Department of Data Analysis and Mathematical Modelling
Faculty of Bioscience Engineering
Ghent University

Supervised Distance Metric Learning for Pattern Recognition

M.Sc. Bac Nguyen Cong

Thesis submitted in partial fulfillment of the requirements for the degree of
Doctor (Ph.D.) of Applied Biological Sciences

Academic year 2018-2019

Supervisors:	Prof. dr. Bernard De Baets Department of Data Analysis and Mathematical Modelling, Ghent University, Belgium
	Prof. dr. Carlos Morell Pérez Department of Computer Science, Universidad Central “Marta Abreu” de Las Villas, Cuba
Examination committee:	Prof. dr. ir. Koen Dewettinck (Chairman) Prof. dr. Stijn Luca Prof. dr. Chris Cornelis Prof. dr. Ann Nowé Prof. dr. Francesc Josep Ferri Rabasa Dr. ir. Michiel Stock
Dean:	Prof. dr. ir. Marc Van Meirvenne
Rector:	Prof. dr. ir. Rik Van de Walle

Bac Nguyen Cong

Supervised Distance Metric Learning for Pattern Recognition

Thesis submitted in partial fulfillment of the requirements for the degree of

Doctor (Ph.D.) of Applied Biological Sciences

Academic year 2018-2019

Dutch translation of the title:
Gesuperviseerd Leren van Afstandsfuncties voor Patroonherkenning

Please refer to this work as follows:
B. Nguyen (**2019**). Supervised Distance Metric Learning for Pattern Recognition, Ph.D. Thesis, Faculty of Bioscience Engineering, Ghent University, Ghent, Belgium. ISBN 9789463571968.

Acknowledgements

First of all, I would like to express my gratitude to Prof. Bernard De Baets and Prof. Carlos Morell Pérez, my two promoters, for their guidance and supervision throughout the present thesis. To Bernard, I really respect your tremendous ability to think, write, and speak clearly. Thank you for many insightful comments and continuous encouragement. Without any doubt, I have learned a lot from your example. To Carlos, I am very grateful for introducing me to the world of machine learning and data science. Thank you both for being always supportive to me through my ups and downs over the last four years. It was a pleasure to work with both of you. I would also like to thank all members of the doctoral committee for their time and effort dedicated to read this thesis.

Thanks for financial support go to the Special Research Fund - Doctoral Scholarships for Candidates from Developing Countries, Ghent University.

Many people contributed to this work in one way or the other. More than one year was carried out at the Centro de Estudio de Informática, Universidad Central de Las Villas. I sincerely thank all members of the group for providing a great working environment. During my Ph.D., I also had an opportunity to enjoy a research stay at the Pattern Recognition and Computer Vision group, Universitat de València, hosted by Prof. Francesc J. Ferri. I would like to thank him and the members of the lab for their hospitality and helpful discussions.

To all of my colleagues and friends, thank you so much for sharing your friendship and good moments. To José, the Mexican "annoying" guy. I still can't stand your Mexican voice when you're singing, but deeply appreciate your help at the beginning of the doctorate. To Christina, thanks for listening to all of my complaints and boring stories. To David, always surprised with a lot of "scientific discoveries" (and also thanks for your massages). To Andreia and Niels, the organizers of many international cool activities. To Aisling, the co-founder of Friday's beer. To Michael, cheerful and full of funny stories. To Lizet for your help and support when I arrived in Belgium. To Petter for his kind help of Dutch translation of the thesis summary. To Trang for the cover design. And to our colleagues within the department, including Gang, Mengzi, Juan Pablo, Zengyuan, Alejandra, Gisele, Shuyun, Wenwen, Dailé, Pamela, Tiago, Raúl, Guillermo, and many others, thanks for the friendly atmosphere. I also want to thank the administrative and technical staffs of the department, including Timpe, Ruth, and Jan.

To my second "family" in Belgium, Duc Anh, Trang, Hang, Phuong Anh, and Chien. During my Ph.D., I also got to know many new friends, including Trinh, Manh, Quan, Vu, Tri, Lan Anh, Nhi, Minh, and many others. I am grateful to

have shared so many great times with you. Thanks for all the Vietnamese foods and funny jokes.

To Nghia, my love who always believes me. Thank you for coming into my life.

Finally, my gratitude for my family. Thanks to my brother Hong. To my parents who brought me into this world. I could not even be here without you. Thanks for your sincere love and understanding.

Bac Nguyen Cong
April 2019

Contents

Summary

Much like in other modeling disciplines does the distance metric used (a measure for dissimilarity) play an important role in the growing field of machine learning. Often, predefined distance metrics (e.g. the Euclidean one) are used to perform such measurement. Unfortunately, most of them ignore any statistical properties that might be estimated from the data. The notion of a good distance metric changes when one moves from one domain to another. For instance, in the problem of computing the dissimilarity for human images, two images could be considered as being similar due to one of the following reasons, the two images are taken from two persons with the same gender, the same age, or the same race. Clearly, it is difficult to use the same distance metric for gender, age, and race since two images might be similar in one case, while being dissimilar in the other case. For this reason, most research efforts have been devoted to automatically learn a good distance metric from data. Depending on the availability of training data, distance metric learning methods can be divided into three categories: supervised, semi-supervised, and unsupervised. Supervised methods often use the heuristic that examples belonging to the same class should be close to each other, while those from different classes should be farther apart. Semi-supervised methods use the information in the form of pairwise similarity or dissimilarity constraints. Unsupervised methods learn a distance metric that preserves the geometric relationships (i.e., distance) between most of the training data for the purpose of unsupervised dimensionality reduction. In this thesis, we focus on supervised distance metric learning. The main aim is to develop efficient and scalable algorithms for solving distance metric learning problems under different types of supervision. The proposed algorithms are supported by empirical as well as theoretical studies.

Nederlandstalige samenvatting

Net zoals in andere modelleerdisciplines speelt de gebruikte afstandsmetriek (een maat voor dissimilariteit) een belangrijke rol in het groeiende domein van machinaal leren. Vaak worden vooraf gedefinieerde afstandsmetrieken (bijvoorbeeld de Euclidische metriek) gebruikt om zulke metingen uit te voeren. Jammer genoeg negeren de meeste van deze metrieken statistische eigenschappen die afgeleid kunnen worden uit de data. De notie van een goede afstandsmetriek varieert van probleem tot probleem. Bijvoorbeeld, wanneer men de dissimilariteit van twee afbeeldingen met mensen wil berekenen, kunnen deze gelijkaardig zijn om de volgende redenen: de twee afbeeldingen zijn genomen van personen met hetzelfde geslacht, leeftijd of afkomst. Het is duidelijk dat het moeilijk is om dezelfde afstandsmetriek te gebruiken voor geslacht, leeftijd en afkomst, aangezien twee afbeeldingen voor het ene geval gelijkaardig kunnen zijn, maar verschillend in een ander geval. Om deze reden wordt veel onderzoek verricht naar het automatisch leren van een goede afstandsmetriek op basis van data. Methoden om een afstandsmetriek te leren kunnen onderverdeeld worden in drie categorieën: gesuperviseerd, semi-gesuperviseerd en ongesuperviseerd. Gesuperviseerde methoden gebruiken vaak de heuristiek dat voorbeelden die tot dezelfde klasse behoren dicht bij elkaar horen te liggen, terwijl voorbeelden die tot een verschillende klasse behoren verder weg van elkaar zouden moeten liggen. Semi-gesuperviseerde methoden gebruiken de informatie in de vorm van paarsgewijze similariteits- of dissimilariteitsvoorwaarden. Ongesuperviseerde methoden leren een afstandsmaat die geometrische relaties (i.e., de afstand) behouden tussen de meeste van de training data om ongesuperviseerde dimensionaliteitsreductie te kunnen uitvoeren. Deze thesis behandelt het leren van een afstandsmetriek op een gesuperviseerde manier. De doelstelling is om efficiënte en schaalbare algoritmes te ontwikkelen voor het oplossen van problemen eigen aan het leren van een afstandmetriek, en dit onder verschillende types van supervisie. De voorgestelde algoritmes worden onderbouwd door zowel empirische als theoretische resultaten.

List of acronyms

k-NN	k-Nearest-Neighbor
mmLMNN	Multiple Metric Learning for Large Margin Nearest Neighbor
ADAMENN	Adaptive Metric Nearest Neighbor
CCCP	Concave-Convex Procedure
CMLP	Constrained Large Margin Local Projection
CMML	Clustered Multi-Metric Learning
DANN	Discriminant Adaptive Nearest Neighbor
DC	Difference of Convex function
DMLMJ	Distance Metric Learning through Maximization of the Jeffrey divergence
DML-dc	Distance Metric Learning using Difference of Convex functions programming
DML-eig	Distance Metric Learning with Eigenvalue Optimization
ITML	Information-Theoretic Metric Learning
LDA	Linear Discriminant Analysis
LDML	Logistic Discriminant Metric Learning
LDMLR	Large-scale Distance Metric Learning for k-NN Regression
LMDML	Large-Margin Distance Metric Learning
LMNN	Large Margin Nearest Neighbor
LMSL	Large Margin Subspace Learning
KEEL	Knowledge Extraction based on Evolutionary Learning
KDMLSC	Kernel-based Distance Metric Learning for Supervised
KISSME	Keep It Simple and Straightforward Metric
NCA	Neighborhood Component Analysis
OASIS	Online Algorithm for Scalable Image Similarity
ODML	Ordinal Distance Metric Learning
PCA	Principal Component Analysis
PRDC	Probabilistic Relative Distance Comparison
POLA	Pseudo-metric Online Learning Algorithm
PSD	Positive Semidefinite
RCA	Relevant Component Analysis
SCML	Sparse Compositional Metric Learning
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
XQDA	Cross-view Quadratic Discriminant Analysis

List of Figures

List of Tables

PART I

INTRODUCTION AND PRELIMINARIES

1 Introduction

1.1. A general overview

Learning a distance metric to measure the closeness of examples is an important research topic in machine learning and pattern recognition. This is also referred to as distance metric learning. Using a good distance metric can lead to great improvements in performance of many fundamental distance-based algorithms such as k-nearest-neighbor (k-NN) classification (Cover and Hart, 1967), k-means clustering (Lloyd, 1982), and kernel regression (Benedetti, 1977). This is motivated by the fact that standard distance metrics (such as the Euclidean one) are often not appropriate as they fail to capture certain specific characteristics of the problem. Recently, many efforts have been devoted to finding a good distance metric for a given problem (Kulis, 2012; Bellet et al., 2015; Moutafis et al., 2017). The idea consists in adjusting a distance metric using the information contained in the training data to satisfy requirements of the application in question. For instance, in a classification setting, a good distance metric should make examples of the same class being close together, while keeping examples of different classes far apart (Davis et al., 2007; Weinberger and Saul, 2009; Ying and Li, 2012; Nguyen et al., 2017c). In information retrieval applications, it should bring the most relevant objects close to the query object given by the users according to the relevance of information of objects (McFee and Lanckriet, 2010).

Among different methods, learning a Mahalanobis distance metric is one of the most successful and well-studied approaches due to its simplicity and flexibility. One can see the Mahalanobis distance metric as a generalization of the Euclidean distance metric, which allows for rotation and scaling of features. Mahalanobis distance metric learning has been widely used in different contexts, such as classification (Weinberger and Saul, 2009; Nguyen et al., 2017c), regression (Nguyen et al., 2016), subspace learning (Peng et al., 2017, 2018), semi-supervised clustering (Yin et al., 2010; Wang et al., 2013), unsupervised learning (Cinbis et al., 2011), learning to rank (McFee and Lanckriet, 2010), etc.

A common guiding principle for learning a distance metric is that the distances between similar examples should be small, while the distances between dissimilar examples should be large. Additionally, there are also several requirements for a good distance metric learning method: (1) it should reflect the true similarity relationships between examples in order to generalize well to unseen examples; (2) it should be easy to implement and to compute efficiently; (3) it should be flexible enough to handle different learning settings and data types.

Unfortunately, most of the existing distance metric learning methods are lacking in at least one of the above requirements. In this thesis, we explore various large-scale optimization techniques for distance metric learning problems under different types of supervision. Our primary focus is on learning the Mahalanobis distance metric. In the next section, we provide a roadmap of the problems considered in this thesis and our contributions.

1.2. Research contributions and structure

This thesis will be divided into five parts: one introductory part (I), three central parts (II-IV), and one concluding part (V). Parts II-IV contain the main contributions, which concern the development of novel supervised distance metric learning algorithms for different learning settings. Each of these parts focuses on a limited number of research objectives and can be read independently. As a typical way of validating algorithms in machine learning, each of the proposed algorithms will be tested on different types of applications. To make the thesis more accessible to readers, Part I includes two chapters (1-2) that introduce basic machine learning concepts, some mathematical tools and well-known results in distance metric learning. Readers who are less familiar with machine learning and mathematical optimization are strongly encouraged to read this part. Part II consists of three chapters (3-5) that exploit the use of binary similarity information (e.g., pairwise constraints) in order to learn a distance metric. Part III consists of five chapters (6-10) that exploit the discriminative nature of relative information (e.g., triplet constraints). Part IV consists of one chapter (11) that exploits the use of kernel learning for clustering. Finally, Part V includes one chapter (12) describing some concluding remarks and possible research directions. Most of the results in this thesis have already been published or submitted for publication in peer-reviewed international journals. Chapters 3, 4, 5, 6, 7, 8, 9, 10, and 11 have been described in Nguyen et al. (2017c); Nguyen and De Baets (2019); Nguyen et al. (2019b, 2018b); Nguyen and De Baets (2018a); Nguyen et al. (2019a, 2016, 2018a), respectively. An overview of the thesis structure is visualized in Fig. 1.1. In this section, the three main parts are briefly introduced. Moreover, for each of the main parts, the key research objectives and contributions are formulated.

1.2.1. Part II: Distance metric learning using pairwise constraints

Problem setting

Information in the form of similarity constraints is often used as natural supervisory information in distance metric learning. Given a set of constraints, distance metric

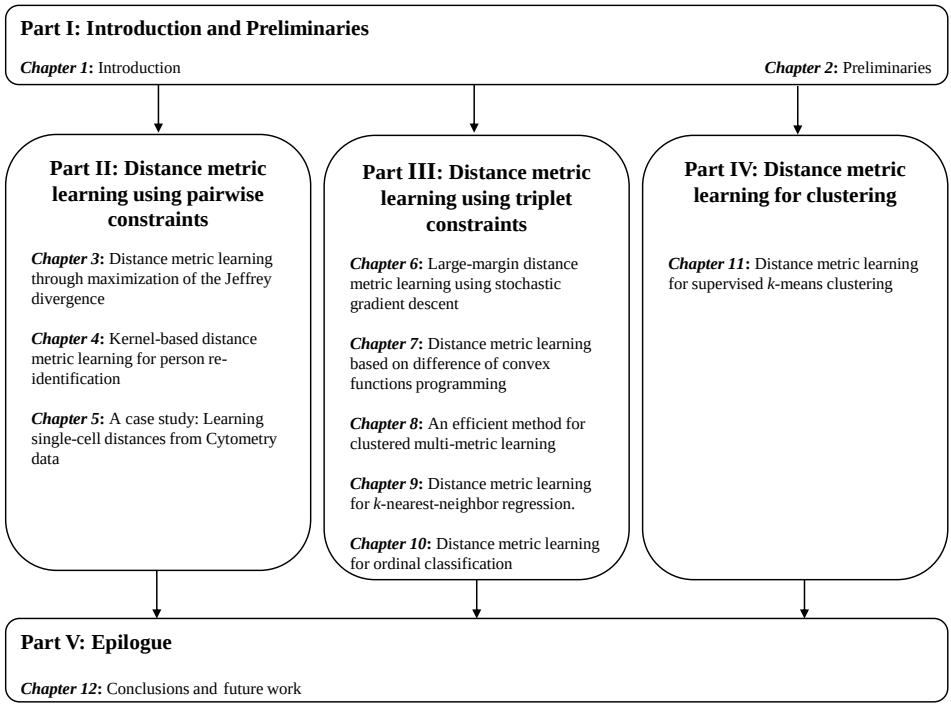


Figure 1.1: A roadmap to the thesis (an arrow from one part to another indicates that the former part is prerequisite material for understanding the latter)

learning aims to find a solution that satisfies as many constraints as possible. In Part II of this thesis, we will develop learning methods based on pairwise constraints, which contain similar (must-link) and dissimilar (cannot-link) pairs of examples. For many learning tasks, pairwise constraints may be extracted with minimal effort or even automatically (Bar-Hillel et al., 2005). A good distance metric should bring examples from similar pairs close to each other while keeping examples from dissimilar pairs far apart. As the number of pairwise constraints can be very large, e.g. $O(N^2)$ pairs can be constructed from N training examples, selecting meaningful pairwise constraints becomes a key issue in order to improve the performance. Besides, learning a distance metric on a large-scale data set raises several issues related to the scalability of the time and space complexity.

Motivated by the above arguments, Part II of this thesis focuses on the following objectives.

Objective II.1: The development of a distance metric learning framework based on local pairwise constraints, which provides a closed-form solution rather than using tedious optimization procedures.

Objective II.2: The exploitation of kernels that allows distance metric learning to effectively handle more complex and high-dimensional data sets.

Objective II.3: The illustration of the proposed framework by means of a case study.

Objectives II.1 and II.2 are the main topics of Chapter 3. Objective II.2 is the main topic of Chapter 4. The last objective is considered in Chapter 5. In the following, we will briefly describe these chapters.

A brief overview of Part II

In Chapter 3, we formulate the problem of learning a linear transformation through maximization of the Jeffrey divergence between two multivariate Gaussian distributions derived from local pairwise constraints. Rather than using tedious optimization procedures, we prove a closed-form solution, which is easy to implement and tractable for large-scale problems. We further derive a kernelized version to tackle nonlinear problems.

In Chapter 4, we present an extension to the well-known KISSME algorithm (Köstinger et al., 2012), an effective distance metric learning method using pairwise constraints to improve the re-identification performance. KISSME is very efficient in terms of training time since it only requires two inverse covariance matrix estimations. However, a linear transformation induced by KISSME may not be powerful enough for more complex problems. We show how to kernelize the KISSME method, resulting in a nonlinear transformation, which is suitable for many real-world applications. To further apply the proposed kernel method efficiently when data are collected sequentially, we introduce a fast incremental version that learns a dissimilarity function in the feature space without estimating the inverse covariance matrices.

As a study case, we explore the use of distance metric learning for the analysis of flow cytometry data in Chapter 5. We evaluate the potential of a learned distance metric in quantifying single-cell distances in a data-driven way. In particular, two different cytometry applications are considered, the first being flow cytometry in the field of synthetic microbial ecology and the second being mass cytometry or Cytometry by Time-Of-Flight (CyTOF) for the analysis of human cells. Results indicate that a learned distance metric can significantly improve the cell population identification.

1.2.2. Part III: Distance metric learning using triplet constraints

Problem setting

Despite the popularity of pairwise constraints, there are two major drawbacks. First, when examples in similar pairs lie in different clusters belonging to the same class, these pairwise constraints may mislead the learning algorithm. Second, it would be more natural for human labelers to compare objects like "A is more similar to B than to C" rather than deciding whether two objects are similar or not. In Part III of this thesis, we will develop distance metric learning methods based on these relative comparisons or triplet constraints. Such constraints are less restrictive than pairwise constraints in the sense that there is no assumption regarding the membership of examples to any class. Instead, it assumes only the proximity of the examples. In the simplest case, triplet constraints can be obtained from any three examples if two of them belong to the same class, which is different from that of the third example. Due to the large number of triplet constraints, one may randomly select a subset of constraints and feed them into the learning algorithm in order to reduce the computational burden. However, such random selection has a few drawbacks. First, it does not consider the most discriminative parts in the feature space (e.g., the boundaries between classes), which can be used to improve the performance. Second, the selected constraints remain the same during the training process, without taking into account the current distance metric. While triplet constraints have been successfully applied to standard classification problems, using such constraints for other learning settings has remained largely unexplored in the literature.

To overcome the above shortcomings, Part III of this thesis considers the use of triplet constraints in different learning settings. In particular, we focus on the following objectives.

Objective III.1: The development of a scalable distance metric learning method based on stochastic gradient descent for nearest-neighbor classification.

Objective III.2: The mathematical formulation of learning a distance metric as a nonconvex optimization problem, as well as the theoretical analysis of the algorithm.

Objective III.3: The development of an effective distance metric learning method that learns multiple distance metrics instead of a single global one, making it more robust to heterogeneously distributed data.

Objective III.4: The development of an efficient distance metric learning for regression settings, where the output space is continuous.

Objective III.5: The development of a distance metric learning method for

ordinal classification settings, where there exists ordering information among class labels.

The above objectives are the main topics of Chapters 6 to 10. In the following, we will briefly describe these chapters.

A brief overview of part III

One of the fundamental challenges in distance metric learning is the positive semidefiniteness constraint on the Mahalanobis matrix. Semidefinite programming is commonly used to enforce this constraint, but it becomes computationally intractable on large-scale data sets. In Chapter 6, we develop an efficient distance metric learning algorithm based on stochastic gradient descent. It employs the principle of margin maximization to learn the distance metric with the goal of improving k-NN classification. Our algorithm can avoid the computation of the full gradient and ensure that the learned matrix remains within the cone of positive semidefinite (PSD) matrices after each iteration. Unlike the method developed in Chapter 3, no assumption about the distribution of the data is required, making it more practical on real-world problems.

Convex optimization has become very popular in distance metric learning over the last few years, because of its empirical performance and because it facilitates a deeper mathematical analysis. Unfortunately, in many practical settings, convexity is not always guaranteed, and one has to resort to nonconvex optimization. In Chapter 7, we exploit the use of nonconvex optimization to learn a distance metric. Similarly to the method proposed in Chapter 6, our distance metric learning framework aims to minimize the misclassification rate of the nearest-neighbor classifier. Due to the use of the ramp loss function, our objective function for margin maximization has a strong ability to avoid the influence of outliers. In particular, the distance metric learning problem is formulated as an instance of difference of convex functions (DC) programming. We show that the generalization error analysis of the proposed approach has an important theoretical implication in explaining that minimizing the objective function may improve the generalization performance of nearest-neighbor classification.

Although there has been an increasing interest in distance metric learning, learning a global distance metric is insufficient to obtain satisfactory results when dealing with heterogeneously distributed data. In Chapter 8, we propose an efficient method that learns multiple local distance metrics instead of a single global one. More specifically, the training examples are divided into several disjoint clusters, in each of which a distance metric is trained to separate the data locally. Moreover, a global distance metric is introduced to capture the common structure among all the clusters, which requires that the distance metric in each cluster should be as close as possible to the global one. On the one hand, the global distance

metric serves as a regularization that controls overfitting; on the other hand, it can lead to the propagation of side-information among clusters, resulting in a more robust and stable model. By learning multiple distance metrics jointly within a single unified optimization framework, our method consistently outperforms single distance metric learning methods, while being more efficient than other state-of-the-art multi-metric learning methods.

In Chapter 9, we present a distance metric learning method for k-nearest-neighbor regression. We define the constraints based on triplets, which are built from the neighborhood of each training example, to learn the distance metric. The resulting optimization problem can be formulated as a convex quadratic program. Our proposed method is simple to implement, and it ensures very fast training, which can be computationally tractable for large-scale data sets.

We further consider distance metric learning for ordinal classification, a problem setting in-between nominal classification and metric regression where the goal is to predict labels from an ordinal scale. Usually, there is a clear ordering of the classes, but the absolute distances between them are unknown. Disregarding the ordering information, this kind of problems is commonly treated as a multi-class classification problem, although this is not appropriate from a semantic point of view. In Chapter 10, we propose a distance metric learning approach for ordinal classification by incorporating local triplet constraints containing the ordering information into a conventional large-margin distance metric learning approach.

1.2.3. Part IV: Distance metric learning for supervised clustering

Problem setting

Part IV of this thesis considers the supervised clustering setting, a problem of training a clustering algorithm with some supervised information so that it can produce a desirable clustering for unseen data (Finley and Joachims, 2005; Daumé and Marcu, 2005). Unlike traditional clustering problems, which are usually referred to as unsupervised clustering, here we have sets of examples and complete clusterings over these sets. By training the distance metric to obtain correct clusterings on supervised data, we expect the distance-based algorithm to cluster unseen data in a similar fashion. Due to the fact that most of the existing semi-supervised methods simply attempt to satisfy the constraints derived from a small amount of labeled data for a single problem, it is usually not reasonable to transfer the knowledge learned from a set of training labels to another set of testing labels (Finley and Joachims, 2005). Supervised clustering can be viewed as a special case of multi-class classification in the sense that both approaches try to classify related examples into

the same class and unrelated examples into different classes (Finley and Joachims, 2005, 2008). Nevertheless, supervised clustering can also be used for problems containing new labels that have not been seen during the training, which seems impossible with multi-class classification.

Therefore, it is important to develop new algorithmic solutions for supervised clustering. In particular, the objectives of Part IV are twofold.

Objective IV.1: The mathematical formulation of learning a distance metric on the kernel space for supervised clustering.

Objective IV.2: The development of a scalable optimization method to efficiently solve this problem.

These objectives are the main topics of Chapter 11, which will be described next.

A brief overview of part IV

In Chapter 11, a kernel-based distance metric learning method is developed to improve the practical use of k-means clustering. In particular, given a set of related data sets with known partitions, we aim to learn a distance metric that will lead to these partitions when k-means clustering is performed. Learning the Mahalanobis distance metric is considered as a structured learning problem. Unlike existing kernel-based methods, we enforce the low-rank constraint on the solution by adding the trace norm to improve the generalization ability. Given the corresponding optimization problem, we derive a meaningful Lagrange dual formulation and introduce an efficient algorithm in order to reduce the training complexity. Our formulation is simple to implement, allowing a large-scale distance metric learning problem to be solved in a computationally tractable way.

1.3. Evaluation methodology

Evaluation is an essential part of any machine learning development. As commonly done in machine learning, the evaluation of a distance metric learning algorithm will be performed on synthetic and benchmark data sets of different sizes and complexities. The benchmark data sets are taken from the machine learning repositories and are available for download. All data sets contain numeric features without missing values.[1] Computational experiments will concern the predictive performance and running times of the proposed methods. Source codes of all methods developed

[1] There are occasionally a few data sets with nominal features, which are encoded as integers. Since our experiments mainly concern the performance between different distance metric learning methods, these nominal features have very little effects on the performance. All methods are trained on the same set of features.

in this thesis can be found at https://github.com/bacnguyencong. These implementations can be loaded in the scientific computing environment Matlab so that any non-expert user can easily use or test the developed models on their own data.

In addition to the performance comparison, the null hypothesis significance testing is considered to analyze the behavior of the proposed methods. This is typically carried out by the Friedman test on the null hypothesis that there is no statistically significant difference among all the competing methods (Demšar, 2006). When the Friedman test rejects the null hypothesis, multiple comparisons are carried out to establish which are the significant differences among the competing algorithms. These multiple comparisons are usually based on the mean-ranks post-hoc tests. Despite its popularity, several issues have been found when applying this hypothesis testing framework (Benavoli et al., 2017, 2016). For instance, the outcome of the test depends on the set of competing algorithms. Algorithms A and B might be declared significantly different in one pool of algorithms and not if the pool contains other algorithms. Many machine learning researchers simply ignore this null hypothesis significance testing in order to avoid such paradoxical situations. Since the approach to perform the statistical comparison of multiple algorithms in machine learning is not yet well developed, we keep using the traditional one introduced by Demšar (2006) in several chapters. The reason is simply that we try to follow a common practice of the scientific journals in which the corresponding chapter was published.

1.4. Notational conventions

For the sake of convenience, the following notations are used throughout the thesis.

- **Scalars**
  Scalars are denoted by lowercase or uppercase letters, such as k, n, and D.
- **Sets**
  Generic sets are denoted by calligraphic uppercase letters, such as $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{V}$. The cardinality of the set $\mathcal{X}$ is denoted by $|\mathcal{X}|$. We use $\mathbb{R}$ to denote the set of real numbers, $\mathbb{R}_+$ to denote the set of nonnegative real numbers. The set of real D-dimensional vectors is denoted by $\mathbb{R}^D$, and the set of real $m \times n$ matrices is denoted by $\mathbb{R}^{m \times n}$. We use $\mathbb{S}^{m \times m}$ to denote the set of $m \times m$ symmetric matrices.
- **Vectors**
  Vectors are assumed to be column vectors and denoted by boldface lowercase letters, such as $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$. The transpose of a vector $\mathbf{x}$ is denoted by $\mathbf{x}^\top$ and the i-th element is denoted as x_i. The inner product between two vectors $\mathbf{u}$

and $\mathbf{v}$ is denoted by $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^\top \mathbf{v}$. Moreover, the ℓ_1-norm of a D-dimensional vector $\mathbf{x}$ is defined by $\|\mathbf{x}\|_1 = \sum_{i=1}^{D} |x_i|$ and the ℓ_2-norm of $\mathbf{x}$ is defined as $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^\top \mathbf{x}}$.

- **Matrices**
  Matrices are denoted by boldface capital letters, such as $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. The trace of a matrix $\mathbf{A}$ is denoted by $\mathrm{tr}(\mathbf{A})$. The identity matrix is denoted by $\mathbf{I}$. The diagonal vector of a square matrix $\mathbf{M}$ is denoted by $\mathrm{diag}(\mathbf{M})$. We will use $\mathbf{K}_{i.}$ to refer to the i-th row vector and $\mathbf{K}_{.j}$ to refer to the j-th column vector of a matrix $\mathbf{K}$. The Hadamard product $\circ$ of two matrices of the same dimension is defined as $(\mathbf{A} \circ \mathbf{B})_{ij} = A_{ij} B_{ij}$. The Frobenius norm $\|.\|_F$ of a matrix $\mathbf{A}$ is defined as $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |A_{ij}|^2}$.

- **Functions**
  A generic function f with domain $\mathcal{X}$ and co-domain $\mathcal{Y}$ is denoted by $f \colon \mathcal{X} \to \mathcal{Y}$. The hinge function $[.]_+ \colon \mathbb{R} \to \mathbb{R}_+$ is defined as $[x]_+ = \max(0, x)$. The gradient of f is denoted as ∇f.

2 Preliminaries

In this chapter, we introduce some relevant backgrounds in machine learning, which are sufficient to understand the contributions of this thesis. We first introduce the supervised learning setting and the main results of statistical learning theory. We then present the basics of distance metric learning as well as its role in machine learning algorithms. Subsequently, a variety of applications using distance metric learning is discussed. Lastly, we review a number of optimization algorithms and see how they can be applied in the context of distance metric learning.

Most of the material that is presented in this chapter can be found in the textbooks *Statistical Learning Theory* (Vapnik, 1998) and *Convex Optimization* (Boyd and Vandenberghe, 2004).

2.1. Supervised learning

Learning from data is a fundamental problem and has a long and successful history. Inspired by humans' capabilities in recognizing patterns, the goal of machine learning is to make computers learn to solve intellectual tasks from experience (data) in an automatic way (Duda et al., 2012). One of the first models realizing this idea was the Perceptron introduced by Rosenblatt (1962). Given a set of examples, it constructs a rule to separate data into two different categories. The generalization ability of this model was successfully tested on the digit recognition problem. During the 1990s, kernel-based learning algorithms (Cortes and Vapnik, 1995; Vapnik, 1998) have proven a great success in machine learning. The most representative kernel-based classifier is the Support Vector Machine (SVM) (Cortes and Vapnik, 1995) (also called Support Vector Network). More recently, deep learning has achieved remarkable success in various domains (Goodfellow et al., 2016). Since the time of the Perceptron, machine learning has gained an immense importance in everyday applications. In addition, significant efforts have been made to develop the mathematical foundations of machine learning (Vapnik, 1998; Mohri et al., 2012; Anthony and Bartlett, 2009). Nowadays, results of a machine learning model must be confirmed by theoretical and experimental studies on different tasks. Depending on the structure of data, one can divide machine learning problems into two main categories: *supervised* and *unsupervised* learning. In a supervised setting, data contain examples (typically a vector) along with their desired outputs (also called labels). In an unsupervised setting, the labels are not available and a machine learning model must be constructed solely on the unlabeled data. This thesis will focus on the first category, supervised learning. Interested readers can refer to the well-written book on unsupervised learning by Bishop (2006).

2.1.1. Problem statement

Assume that data are generated from an unknown generator, the goal of supervised learning is to infer a hypothesis (an approximation of the generator) using a limited number of training examples. In the following, we review basic notions of statistical learning theory, a well-known framework pioneered by Vapnik (1998).

Formally, the selection of a desired hypothesis is based on a set of n independent and identically distributed (i.i.d.) pairs

$$\big\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\big\}$$

drawn according to some unknown distribution $p(\mathbf{x}, y) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x})$. Here, $\mathbf{x} \in \mathbb{R}^D$ is a vector in the input space $\mathcal{X}$ and y is the associated output in the output space $\mathcal{Y}$. A learning algorithm tries to construct an approximation, which provides for a given generator, the best prediction to the outputs. In other words, the goal is to find an appropriate hypothesis $f\colon \mathcal{X} \to \mathcal{Y}$, where $f \in \mathcal{H}$, a set of admissible hypotheses, which achieves the best results in prediction. Therefore, estimating the hypothesis f corresponds to minimizing the following expected error

$$R(f) = \mathbb{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y)}\big[L(y, f(\mathbf{x}))\big] = \int L(y, f(\mathbf{x}))\, d\, p(\mathbf{x}, y)\,,$$

with L a loss function which incurs a penalty if $f(\mathbf{x}) \neq y$. For instance, the simplest loss function in classification is the misclassification error, defined as

$$L(y, f(\mathbf{x})) = \begin{cases} 0\,, & \text{if } f(\mathbf{x}) = y\,, \\ 1\,, & \text{if } f(\mathbf{x}) \neq y\,. \end{cases}$$

Directly optimizing the misclassification error is hard, even for a simple hypothesis space (Ben-David et al., 2003). For this reason, *surrogate loss* functions are often used, such as the hinge loss, the exponential loss, and the logistic loss.

The problem of learning, therefore, is to find a hypothesis that minimizes the expected error when the probability distribution $p(\mathbf{x}, y)$ is unknown, given the observed data.

2.1.2. Empirical risk minimization

One can easily show that good predictions of the training data are necessary conditions to perform well on unseen data. This leads to the principle of Empirical Risk Minimization (ERM). In general, the risk $R(f)$ cannot be computed because the distribution $p(\mathbf{x}, y)$ is unknown. However, it can be approximated by averaging

the loss function on the training examples, the so-called empirical risk,

$$R_{\text{emp}}(f) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) \,.$$

The ERM principle suggests that the learning algorithm should choose a hypothesis f that minimizes the empirical risk

$$f^* = \underset{f \in \mathcal{H}}{\text{minimize}}\; R_{\text{emp}}(f) \,.$$

While the ERM principle may work well in practice, selecting the hypothesis space $\mathcal{H}$ must be done carefully in order to make a good model. Without knowledge of the task, it is often difficult to select an appropriate $\mathcal{H}$.

2.1.3. Structural risk minimization

Minimizing the empirical risk is a necessary condition, but what are the sufficient conditions? Why is the rule that is correct for training examples also correct for unseen examples? Statistical learning theory provides an answer to these questions. In particular, with a probability at least $1 - \sigma$, the expected error of any $f \in \mathcal{H}$ is bounded by

$$R(f) \leq R_{\text{emp}}(f) + \sqrt{\frac{h\left(\log \frac{2n}{h} + 1\right) - \log \frac{\sigma}{4}}{n}} \,, \tag{2.1}$$

where h is the Vapnik-Chervonenkis (VC)-dimension of the hypothesis space $\mathcal{H}$, which measures the capacity of $\mathcal{H}$. Clearly, a small value of the empirical error does not necessarily imply a small value of the expected error. The generalization-error bound in (2.1) suggests that, disregarding the logarithmic factors, in order to achieve a good generalization performance, we need to minimize both the empirical error and the ratio between the VC-dimension and the number of training examples at the same time. In other words, it is very important to make a good trade-off between minimizing the empirical risk and choosing an appropriate value for the VC-dimension of the function space $\mathcal{H}$, known as the bias-variance trade-off. There are two reasons that make the true risk of a hypothesis being much larger than its empirical risk. The first one is due to a too simple model, which is referred to as *underfitting*. The second one is due to a too complex model, which is referred to as *overfitting*.

One solution to this problem is based on the Structural Risk Minimization (SRM) principle (Vapnik, 1998). This principle tries to find an optimal relationship between the empirical error estimated by the hypothesis chosen from a set of hypotheses and the capacity of that set of hypotheses (see Fig. 2.1). Let $\Gamma_1, \ldots, \Gamma_l$

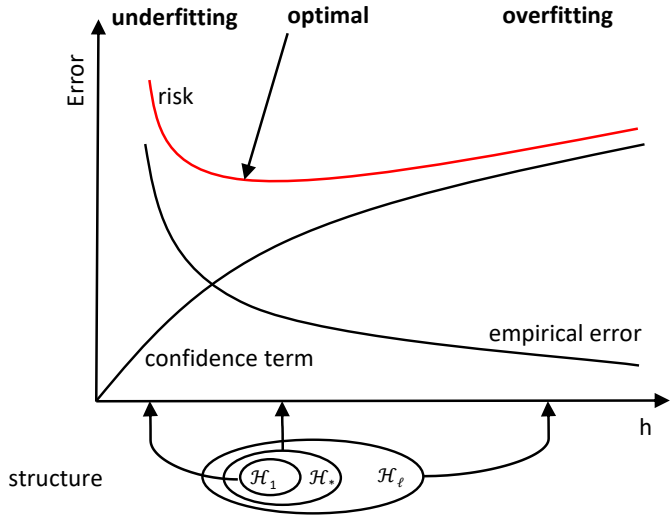


Figure 2.1: An illustration of the Structured Risk Minimization (SRM) principle. If the hypothesis space $\mathcal{H}_j$ has a small complexity, then the model capacity is small, but the empirical error is large (underfitting). Otherwise, if the complexity of $\mathcal{H}_j$ is large, then the model capacity is large, while the empirical error is small (overfitting). The optimal expected error at $\mathcal{H}_*$ is achieved by making a good trade-off between the empirical error and the model capacity.

be the equivalence classes of all admissible hypotheses in $\mathcal{H}$, where two hypotheses belong to the same equivalence class if they separate the training examples in the same way. In doing this, we split our set containing an infinite number of admissible hypotheses $\mathcal{H}$ into a set containing a finite number of equivalence classes Γ_i, where $i \in \{1, \ldots, l\}$. In order to perform the SRM principle, we first create a structure on these equivalence classes, which is a set of nested subsets of hypotheses

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \cdots \subset \mathcal{H}\,,$$

such that $h_1 \leq h_2 \leq \cdots \leq h$, where h_j is the VC-dimension of $\mathcal{H}_j$. The minimization of the right-hand side of (2.1) can be performed as follows: we first choose an element of the structure to control the VC-dimension, then choose a hypothesis in this element that minimizes the empirical error. In order to build this structure, we need to associate with each equivalence class some value characterizing the capacity of the class. For instance, in the case of SVM, each equivalence class is associated with the largest margin of the hypothesis belonging to this class, since a high value of the margin corresponds to a low value of the VC-dimension.

2.1.4. Regularized risk minimization

An implicit way of working with nested hypothesis spaces is the principle of regularized risk minimization (RRM). Instead of minimizing the empirical error $R_{\text{emp}}(f)$ and then expressing the generalization ability of the resulting model f using some capacity measure of the underlying hypothesis class $\mathcal{H}$, we can directly minimize the so-called regularized risk

$$R_{\text{reg}}(f) = \underset{f \in \mathcal{H}}{\text{minimize}} \quad R_{\text{emp}}(f) + \lambda\, \Omega(f)$$

where the hyperparameter $\lambda \geq 0$ controls the trade-off between minimizing the empirical error and punishing hypotheses with large fluctuation through the regularizer Ω. The regularization is used to penalize "complex" hypotheses and to break the tie between hypotheses that have the same empirical error.

After having presented the supervised learning setting and the statistical learning framework, we now turn to distance metric learning, which is the main focus of this thesis.

2.2. Distance metric learning

The notion of similarity between objects or examples plays a key role in several machine learning tasks. There is often no obvious way of defining a (dis)similarity measure. Rather than using a default distance metric such as the Euclidean one, it is desirable to learn a distance metric that satisfies certain conditions, depending on the application domain. If some side-information is given, for instance, as provided by human labelers, it can be used to optimize an appropriate criterion requiring that the distances between similar examples (e.g. examples of the same class) are smaller than those between dissimilar examples (e.g. examples of different classes). In this section, we first give the definition of distance metrics. Next, we introduce a brief overview of some of the applications of distance metric learning employed in various domains. Finally, we formulate the problem of learning a distance metric as a mathematical optimization problem and then discuss some common optimization techniques used in the literature.

2.2.1. Definitions

Distance usually refers to some degree of closeness of two objects, e.g., length, gap, time, or rank difference. Here, we consider the mathematical notion of this term. We start by introducing the definition of what is a distance metric.

Definition 2.1. *A distance metric is a function* $d\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ *that satisfies, for any* $\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_l \in \mathcal{X}$*:*

(i) Non-negativity: $d(\mathbf{x}_i, \mathbf{x}_j) \geq 0$*;*

(ii) Symmetry: $d(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_j, \mathbf{x}_i)$*;*

(iii) Triangle inequality: $d(\mathbf{x}_i, \mathbf{x}_j) + d(\mathbf{x}_j, \mathbf{x}_l) \geq d(\mathbf{x}_i, \mathbf{x}_l)$*;*

(iv) Distinguishability: $d(\mathbf{x}_i, \mathbf{x}_j) = 0 \Leftrightarrow \mathbf{x}_i = \mathbf{x}_j$*.*

The above definition was introduced by Fréchet (1906) to define a metric space $(\mathcal{X}, d)$, which is a special case of a general topological space. If one of these properties fails while others hold, the corresponding functions are given a different name. For instance, if the second property fails, then d is called a quasi metric. If the last property fails, we talk about pseudometrics. In distance metric learning, we do not distinguish between pseudometrics and metrics. Two notions are used interchangeably in the literature. It is worth noting that the triangle inequality property becomes very important to speedup learning algorithms such as nearest neighbor search (Wang, 2011b) and k-means clustering (Elkan, 2003a). Some examples of distance metrics are listed below (see Deza and Deza, 2006, for more concrete examples).

Example 2.1. The Minkowski distance metrics are a family of distance metrics induced by the ℓ_p-norms, given by

$$d_p(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_p = \left(\sum_{i=1}^{D} |u_i - v_i|^p \right)^{1/p}$$

with $p \geq 1$. Some widely used distance metrics are

(i) $p = 1$, the Manhattan distance metric $d(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{D} |u_i - v_i|$,

(ii) $p = 2$, the Euclidean distance metric $d(\mathbf{u}, \mathbf{v}) = \sqrt{\sum_{i=1}^{D} |u_i - v_i|^2}$,

(iii) $p \to \infty$, the Chebyshev distance metric $d(\mathbf{u}, \mathbf{v}) = \max_{i=1,\ldots,D} |u_i - v_i|$.

Example 2.2. The Hamming distance metric defines the distance between two D-dimensional vectors as the number of positions at which their values differ,

$$d(\mathbf{u}, \mathbf{v}) = \left| \left\{ i \in \{1, \ldots, D\} \mid u_i \neq v_i \right\} \right| .$$

As mentioned in the preceding chapter, distance metric learning will be mostly formulated as a semidefinite program. We hence need to define positive semidefinite matrices.

Definition 2.2. *A symmetric matrix* $\mathbf{M} \in \mathbb{R}^{D \times D}$ *is PSD, denoted by* $\mathbf{M} \succcurlyeq 0$*, if*

for any vector $\mathbf{x} \in \mathbb{R}^D$*, the following condition holds*

$$\mathbf{x}^\top \mathbf{M} \mathbf{x} \geq 0 \,.$$

Most of the studies in this field pay particular attention to the Mahalanobis distance metric (Mahalanobis, 1936) because it can be conveniently optimized by deriving a convex formulation with the guarantee of finding the global optimum (Weinberger and Saul, 2009; Xing et al., 2002; Davis et al., 2007; Shen et al., 2012). Besides, it provides good generalization performance (Shi et al., 2014; Guo and Ying, 2014; Jin et al., 2009). The Mahalanobis distance metric originally refers to a distance measure that incorporates the correlation between features

$$d_{\boldsymbol{\Sigma}^{-1}}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \mathbf{x}_j)} \,,$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are random vectors of the same Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix

$$\boldsymbol{\Sigma} = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\top \,.$$

In the distance metric learning literature, the Mahalanobis distance metric is defined as follows.

Definition 2.3. *The Mahalanobis distance between two vectors* $\mathbf{x}_i$ *and* $\mathbf{x}_j$ *in* $\mathbb{R}^D$ *with respect to a PSD matrix* $\mathbf{M}$ *is computed as*

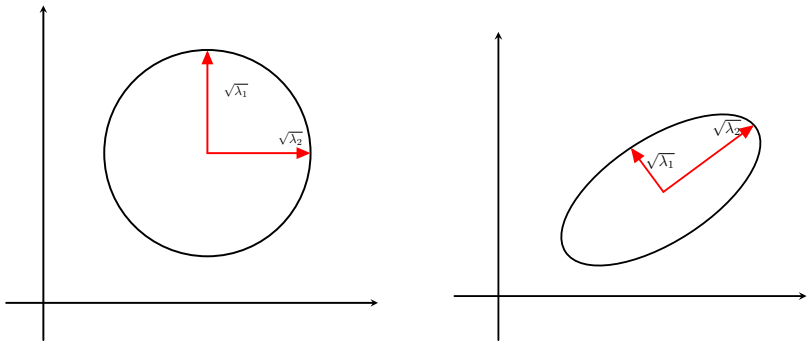
$$d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j)} \,.$$

One can decompose $\mathbf{M}$ as $\mathbf{M} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$ using the eigenvalue decomposition, where $\mathbf{U}$ is a matrix containing all eigenvectors of $\mathbf{M}$ and $\boldsymbol{\Lambda}$ is a diagonal matrix containing all eigenvalues of $\mathbf{M}$ on its diagonal. Let $\mathbf{L} = \mathbf{U}\boldsymbol{\Lambda}^{1/2}$, then the Mahalanobis distance can be viewed as the Euclidean distance in the transformed space after performing a linear transformation $\mathbf{x}'_i = \mathbf{L}^\top \mathbf{x}_i$,

$$\begin{aligned} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) &= (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{L}\mathbf{L}^\top (\mathbf{x}_i - \mathbf{x}_j) \\ &= (\mathbf{L}^\top \mathbf{x}_i - \mathbf{L}^\top \mathbf{x}_j)^\top (\mathbf{L}^\top \mathbf{x}_i - \mathbf{L}^\top \mathbf{x}_j) \\ &= \|\mathbf{x}'_i - \mathbf{x}'_j\|^2 \,. \end{aligned}$$

We can interpret $\mathbf{x}'_i$ as a projected point in a new coordinate system defined by the orthogonal matrix $\mathbf{U}$, which is shifted and rotated w.r.t. the original coordinates (see Fig. 2.2). Thus, the Mahalanobis distance exactly captures the idea of learning a global linear transformation. From another perspective, this is equivalent to first applying a whitening transformation to the data, making a set of uncorrelated

examples having a unit variance, and then measuring the Euclidean distances in the transformed space.



(a) Euclidean **(b)** Mahalanobis

Figure 2.2: An illustration of the Euclidean and the Mahalanobis distance metrics in the 2-dimensional space. The major axes of the ellipse are defined by the eigenvectors $\mathbf{u}_i$ of $\mathbf{M}$ and the corresponding eigenvalues λ_i. For the Euclidean case, both λ_1 and λ_2 are identical.

2.2.2. Problem statement

The main objective of distance metric learning is to find a proper distance metric using the information contained in the training data, in order to bring similar examples closer and to push dissimilar examples farther away. A distance metric learning approach is usually formulated as a constrained optimization problem,

$$\underset{\mathbf{W}}{\text{minimize}} \quad f(\mathbf{W}) = \lambda\Omega(\mathbf{W}) + L(\mathbf{W}, \mathcal{R})\,, \tag{2.2}$$

where $\Omega(\mathbf{W})$ is the regularizer, $L(\mathbf{W}, \mathcal{R})$ is the loss term penalizing the violation of $\mathbf{W}$ over a set of constraints $\mathcal{R}$, and $\lambda \geq 0$ is the regularization parameter. Depending on the parameterization of $\mathbf{W}$ (i.e., learning a Mahalanobis matrix $\mathbf{M}$ or a linear transformation matrix $\mathbf{L}$), we might have additional constraints. While this formulation is different for each approach, the constraints are often of one of the following types:

(i) Pairwise constraints:
$\mathcal{S} = \big\{\, (\mathbf{x}_i, \mathbf{x}_j) \mid \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ should be similar} \,\big\}$,
$\mathcal{D} = \big\{\, (\mathbf{x}_i, \mathbf{x}_j) \mid \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ should be dissimilar} \,\big\}$.

(ii) Relative constraints:
$\mathcal{T} = \big\{\, (\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_l) \mid \mathbf{x}_i \text{ should be more similar to } \mathbf{x}_j \text{ than to } \mathbf{x}_l \,\big\}$.

(iii) Quadruplewise constraints:
$\mathcal{Q} = \big\{\, (\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_l, \mathbf{x}_m) \mid d(\mathbf{x}_i, \mathbf{x}_j) \text{ should be smaller than } d(\mathbf{x}_l, \mathbf{x}_m) \,\big\}$.

The latter has been widely used in computer vision (Law et al., 2013). This type of constraints is easy to find when the ordering information of classes such as $y_l \prec y_i \prec y_j \prec y_m$ is given. Hence, the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ should be smaller than the distance between $\mathbf{x}_l$ and $\mathbf{x}_m$. In this thesis, we will pay more attention to the first and second kinds of constraints. It is important to note that one can obtain relative constraints whenever pairwise constraints are available. This can easily be done by choosing $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_l)$ such that $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}$ and $(\mathbf{x}_i, \mathbf{x}_l) \in \mathcal{D}$, but one cannot always obtain pairwise constraints when relative constraints are available (Wang et al., 2013). In other words, relative constraints are weaker than pairwise constraints. However, human labelers may respond inconsistently in deciding whether two objects are similar or not, but they may all agree on comparing objects like "$\mathbf{x}_i$ is more similar to $\mathbf{x}_j$ than to $\mathbf{x}_l$," which is more natural for human labelers. Clearly, this relative comparison provides more general semantic relationships between examples than the preceding binary form. Relative or triplet constraints have been empirically demonstrated to be effective for distance metric learning (Li et al., 2017; Nguyen et al., 2016; Parameswaran and Weinberger, 2010). However, selecting which triplets should be used for training turns out to be very important in order to achieve a good performance.

2.2.3. Distance-based learning algorithms

Recent years have witnessed an increasing interest in the use of supervised learning techniques for pattern recognition. Computers have demonstrated a recognition rate better than or comparable to human performance in several tasks, such as visual recognition and recommendation systems (Goodfellow et al., 2016). Despite this success, many machine learning algorithms often require embedding of data points into some space. Algorithms such as k-NN classification and neural networks consider the embedding space to be $\mathbb{R}^D$, while kernel methods such as SVMs consider the embedding space to be a Hilbert space. In any case, the notion of distance metric has to be carefully considered. Without any additional knowledge, the Euclidean distance metric is often used. However, learning a good distance metric that fits the data well can provide a promising solution to increase the performance of distance-based algorithms. Below we describe in detail the role of distance metric in several classic machine learning algorithms.

k-nearest-neighbor classification

In pattern recognition, the k-nearest-neighbor (k-NN) algorithm (Cover and Hart, 1967) is among the simplest and most popular classifiers. It is a non-parametric method and does not make any assumption about the data distribution. Essentially, the k-NN classification rule assigns a test example to the majority class label of its k nearest neighbors. In the simple case of $k = 1$, it just assigns the class label

of its nearest neighbor. According to Cover (1968), the k-NN classifier has an asymptotic error rate that converges to the Bayes error rate as $k \to \infty$ and $k/n \to 0$, where n is the number of training examples. We refer to Mclachlan (2004) for a more detailed discussion of k-NN classification. Despite its simplicity, the k-NN classifier is well suited for multi-class classification problems with a large number of training examples, which are relatively common in many pattern recognition applications.

Given a training set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, we first define $r(\mathbf{x}, j)$ as a function that returns the index of the j-th nearest neighbor of an example $\mathbf{x}$ in the training set, which is defined by

$$r(\mathbf{x}, j) = \begin{cases} \underset{i=1,\ldots,n}{\operatorname{argmin}} \quad d(\mathbf{x}, \mathbf{x}_i) & \text{if } j = 1; \\ \underset{i=1,\ldots,n}{\operatorname{argmin}} \quad d(\mathbf{x}, \mathbf{x}_i) \text{ and } i \notin \{r(\mathbf{x}, 1), \ldots, r(\mathbf{x}, j-1)\} & \text{otherwise,} \end{cases}$$

where d denotes the distance metric defined on $\mathcal{X}$. Then, the posterior distribution of a given example is defined as

$$p(y|\mathbf{x}) = \frac{\sum_{\mathbf{x}' \in \mathcal{V}(\mathbf{x})} [\![y' = y]\!]}{k},$$

where $\mathcal{V}(\mathbf{x})$ is the set of k nearest examples of $\mathbf{x}$ and $[\![.]\!]$ is an indicator function that takes value 1 if its argument is true, and value 0 otherwise. Following the decision rule of maximum a posteriori (MAP), an example is classified into the most common class among its k nearest neighbors.

k-means clustering

Clustering is an important task in pattern recognition for data analysis. Among various clustering techniques, k-means clustering (Lloyd, 1982) is one of the most popular and most efficient techniques for general clustering tasks. The goal is to partition a set of examples into disjoint clusters based on some notion of (dis)similarity, such that related examples belong to the same cluster, while unrelated examples belong to different clusters (Huang et al., 2014).

Given a set of n examples $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, the goal of k-means clustering is to find an assignment of these examples into k disjoint sets, which leads to a minimal sum of squared distances between the examples and their corresponding cluster center. Let $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_k] \in \mathbb{R}^{D \times k}$ be k center vectors and $\mathbf{Y} \in \{0, 1\}^{k \times n}$ denote the assignment matrix where $Y_{c,i} = 1$ if example $\mathbf{x}_i$ belongs to the c-th cluster, otherwise $Y_{c,i} = 0$. Following Peng and Wei (2007), the objective of k-means

clustering can be formulated as

$$\begin{array}{ll} \underset{\mathbf{Y},\mathbf{Z}}{\text{minimize}} & \sum_{i=1}^{n}\sum_{c=1}^{k} Y_{c,i}\, d^2(\mathbf{x}_i, \mathbf{z}_c) \\ \text{subject to} & \mathbf{Y} \in \{0,1\}^{k\times n}\,, \\ & \operatorname{rank}(\mathbf{Y}) = k\,, \\ & \mathbf{Y}^\top \mathbf{1} = \mathbf{1}\,, \\ & \mathbf{Z} \in \mathbb{R}^{D\times k}\,. \end{array} \tag{2.3}$$

This problem is a mixed integer program with a nonlinear objective function, which is NP-hard (Aloise et al., 2009). This is due to the fact that the constraints are discrete and the objective function is nonconvex and nonlinear, making the problem very challenging.

Learning with kernels

Kernel learning algorithms typically attempt to learn a kernel matrix over the data (Abbasnejad et al., 2012). A nonlinear kernel can address limitations of linear methods by implicitly mapping the nonlinearly-distributed data to a high-dimensional feature space (Schölkopf and Smola, 2001). By formulating kernel learning as a distance metric learning problem, one can learn the kernel matrix without any assumption on the form of the kernel that implicitly generated it (Jain et al., 2012). Thus, the resulting kernel matrix can generalize well to unseen examples.

Consider a mapping ϕ from the input space $\mathcal{X}$ into a high-dimensional space (the so-called Reproducing Kernel Hilbert Space) $\mathcal{F}$, i.e. $\phi\colon \mathcal{X} \to \mathcal{F}$. Note that the dimensionality of $\mathcal{F}$ can be very high or even infinite. In such case, it becomes hard to learn directly from the feature space due to the computational bottleneck. To this end, kernel methods implicitly perform ϕ by replacing the inner product with a positive semidefinite function $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$. Therefore, they do not necessarily compute the new representations. Several kernel functions, such as polynomials, χ^2, and exponential χ^2 kernel functions, have been successfully employed (see Schölkopf and Smola, 2001, for more examples of kernel functions). In the context of distance metric learning, the Gaussian kernel is one of the most often used kernel functions, given by

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{d^2(\mathbf{x}_i, \mathbf{x}_j)}{\sigma^2}\right)$$

with σ the kernel width. In this case, one can learn a better representation in order to improve the performance of kernel methods (Weinberger and Tesauro, 2007).

2.3. Why should we care about distance metric learning?

Distance metric learning has become a field in itself in the machine learning community. It has gained popularity in the last decades and has been the main topic of several workshops at leading conferences such as NIPS, ICML, and ECML/PKDD. Interested readers may refer to the surveys, such as Kulis (2012) and Bellet et al. (2015), for further details on this topic. Below, we briefly highlight some of the most important applications using distance metric learning.

2.3.1. Information retrieval

Ranking is a central problem in information retrieval. The goal is to provide the user with a ranking containing the most relevant documents according to his/her query. Given a good distance metric, a straightforward solution is achieved by sorting the training set by increasing the distance from the query, in which relevant documents are at the front of the list, while irrelevant documents are at the end. Distance metric learning can be viewed as a special case of the query-by-example paradigm in information retrieval. Many advances have been made in recent years to improve the distance metric used for ranking (Lebanon, 2006; Lee et al., 2008; McFee and Lanckriet, 2010; Lim et al., 2013; Paisitkriangkrai et al., 2015).

2.3.2. Computer vision

Computer vision is the most successful domain for distance metric learning. In image classification, it involves learning an appropriate distance metric, and then applying a simple nearest-neighbor classifier to tag new images (Frome et al., 2007b; Hoi et al., 2006). The distance between images of the same category should be less than the distance between images of different categories. In person re-identification, given an image of a person, the main task is to identify the person from images taken at a different location and/or from a different viewpoint across non-overlapping cameras. It is important to remark that when a person disappears from one camera, he/she can be recognized from other cameras. Distance metric learning can implicitly suppress those cross-view variations between images (Hirzer et al., 2012b; Köstinger et al., 2012). This is motivated by the fact that standard distance metrics, such as the Euclidean and Manhattan distance metrics, are not reliable and flexible enough because they usually assume that all features are from the same domain with the same scale. Consequently, they become more sensitive to irrelevant features and fail to preserve the geometric characteristics of the data (Nguyen et al., 2017c). Moreover, distance metric learning has been applied in face recognition (Guillaumin et al., 2009), visual tracking (Jiang et al.,

2012), human activity recognition (Tran and Sorokin, 2008), and human body pose estimation (Kulis et al., 2009b).

2.3.3. Dimensionality reduction

High-dimensional data arise in many important data mining applications, such as mining texts, sounds, images, gene expression profiles, fMRI data, etc. These application domains share the important property that examples are described by thousands of features. It is well known that the performance of many machine learning algorithms degrades as the number of features grows. This is often referred to as the curse of dimensionality. Concretely, when working with high-dimensional data sets, we might encounter some difficulties, such as the empty space phenomenon, concentration of distances, or the peaking phenomenon (François, 2008). The empty space phenomenon leads to poor and non-smooth approximation of the true probability density function because of the sparsity of the input space. Concentration of distances is the phenomenon that the distances between all different samples are roughly equal, so it is very difficult to draw conclusions from the data. The peaking phenomenon relates to the number of parameters of a model. The parameters cannot be correctly estimated when the number of parameters becomes large compared to the data size.

To handle such real-world data adequately, one needs to transform the high-dimensional data into a meaningful representation of reduced dimensionality (Van Der Maaten et al., 2009). Dimensionality reduction can alleviate the curse of dimensionality and other undesirable properties of high-dimensional spaces. The main idea is to learn an underlying low-dimensional space where geometric relationships (e.g., distance) between most of the observed examples are preserved. Principal Component Analysis (PCA) (Jolliffe, 2005) and Linear Discriminant Analysis (LDA) (Fisher, 1936) are typical examples of distance metric learning for dimensionality reduction. Other nonlinear dimensionality reduction methods include ISOMAP (Tenenbaum et al., 2000), Locally Linear Embedding (LLE) (Roweis and Saul, 2000), and Laplacian Eigenmap (Belkin and Niyogi, 2003).

2.3.4. Transfer learning and domain adaptation

Recent results have shown that training on one domain and then testing on another domain often results in poor performance (Saenko et al., 2010). Transfer learning aims at improving the ability of exploiting commonalities between different learning tasks in order to share statistical strength and to transfer knowledge across tasks (Pan and Yang, 2010). Once the distance metric is learned, we can use it for other tasks as well, for which the source and target tasks are related and share some common structure. The idea of transfer distance metric learning is to learn a

distance metric from one task and then apply it for other related tasks due to the lack of knowledge (Zhang and Yeung, 2010; Hu et al., 2015a), e.g. the unavailability of labeled data. In this case, we assume that source and target tasks share the same distance metric, which captures some common structure that allows to measure the distance between examples (Luo et al., 2014). In the related domain adaptation setup, the distributions of source and targets are different. The idea is to learn a transformation that maps the data from one domain to the another (Kulis et al., 2011).

2.3.5. Bioinformatics

Problems in bioinformatics usually involve comparing sequences such as DNA, protein or time series. Learning a good distance metric can significantly increase the performance. In this case, structured-distance metrics such as edit-distance metrics are often used for strings or time series (Bellet et al., 2011, 2012). DNA microarrays measure the expression levels of a huge number of genes simultaneously, where the goal is to classify tissue samples according to their gene expression levels. Based on these predictions, we can diagnose and predict various genetic disorders including cancer. The k-NN classifier using distance metric learning has shown superior results in some cases (Xiong and Chen, 2006; Takeuchi et al., 2009). In addition, distance metric learning algorithms have recently been applied to enzyme search (Kato and Nagano, 2010).

2.4. Optimization techniques for distance metric learning

Mathematical optimization plays a central role in this thesis. This section serves as a general introduction to the optimization techniques used for learning Mahalanobis distance metrics. As mentioned before, the Mahalanobis distance metric can be parametrized in terms of the matrix $\mathbf{L}$ or the matrix $\mathbf{M}$. We should take into account the fact that $\mathbf{L}$ uniquely defines $\mathbf{M}$, while $\mathbf{M}$ defines $\mathbf{L}$ up to rotation (which does not affect the calculation of distances). This equivalence suggests two general approaches for distance metric learning: one can estimate a linear transformation matrix $\mathbf{L}$ or estimate a PSD matrix $\mathbf{M}$. Note that in the first approach, optimization is unconstrained, whereas in the second approach, it is necessary to enforce the positive semidefiniteness constraint on $\mathbf{M}$. Although it is usually more difficult to solve an optimization problem with many constraints, the second approach has certain advantages. It can lead to convex optimization problems with positive semidefiniteness constraints. It is beneficial to work with convex optimization problems, as they can be solved in polynomial time (Boyd and Vandenberghe, 2004). In general, there is no universally accepted optimization technique. Depending

on the purpose of learning, the authors of most existing methods design their own optimization techniques tailored to the individual methods. Below we will discuss some of the most relevant optimization techniques used in distance metric learning.

2.4.1. Semidefinite programming

Problem (2.2) belongs to the family of semidefinite programming problems, which aims to minimize a linear objective function over the intersection of the cone of PSD matrices with an affine-linear space (also called a *spectrahedron*). A general semidefinite program can be expressed as

$$\begin{aligned} \underset{\mathbf{W}}{\text{minimize}} \quad & \langle \mathbf{W}, \mathbf{C} \rangle \\ \text{subject to} \quad & \langle \mathbf{W}, \mathbf{A}_t \rangle \leq b_t \,, \quad t = 1, \ldots, m \\ & \mathbf{W} \succcurlyeq 0 \,, \end{aligned}$$

where matrices $\mathbf{A}_t$ and scalars b_t define m linear constraints. Semidefinite programming can be seen as an extension of linear programming where the componentwise inequalities between vectors are replaced by matrix inequalities. Most of semidefinite programming solvers are based on primal-dual interior-point methods (Wright, 1997). Popular solvers include the Python Software for Convex Optimization (CVXOPT) developed by Dahl and Vandenberghe (2004) and the Matlab Software for Disciplined Convex Programming (CVX) developed by Grant and Boyd (2014). However, general-purpose solvers usually need to calculate the Hessian matrix, which requires a memory complexity of $O(D^4)$ and a time complexity of $O(D^{6.5})$ in the worst case. For some real-world applications, they become almost intractable.

2.4.2. Gradient descent

Gradient descent (Cauchy, 1847) is one of the simplest iterative first-order optimization algorithms for unconstrained optimization problems. To find a local minimum, gradient descent operates as follows. At the t-th iteration, it takes the gradient (or an approximation of the gradient) ∇f of the objective function f at a current solution $\mathbf{W}_t$. Then, it steps proportional to the negative of the gradient, i.e.,

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta_t \nabla f(\mathbf{W}_t) \,,$$

where $\eta_t > 0$ denotes the step size. One can choose the step size in different ways. A simple approach is to set η_t to a small fixed number. Another approach, referred

to as line search, is to evaluate $f(\mathbf{W} - \eta_t \nabla f(\mathbf{W}_t))$ for several values of η_t and select the one that achieves the smallest objective function value.

Since the domain of a distance metric learning problem typically is the cone of PSD matrices, standard gradient descent cannot be trivially applied. To this end, one can factorize the Mahalanobis matrix $\mathbf{M}$ as $\mathbf{L}\mathbf{L}^\top$, then optimize the objective function over the matrix $\mathbf{L}$ instead of $\mathbf{M}$. By setting the number of columns of $\mathbf{L}$ to be less than the dimensions (rectangular matrices), we simultaneously perform distance metric learning and dimentionality reduction.

As an example, consider the Neighborhood Component Analysis (NCA) model proposed by Goldberger et al. (2005). NCA aims to optimize the expected leave-one-out error of a stochastic nearest-neighbor classifier in the projected space induced by $\mathbf{M}$. Let us define the probability of $\mathbf{x}_i$ being the neighbor of $\mathbf{x}_j$ as

$$p_{ij} = \begin{cases} \dfrac{\exp(-\|\mathbf{L}^\top \mathbf{x}_i - \mathbf{L}^\top \mathbf{x}_j\|^2)}{\sum_{l \neq i} \exp(-\|\mathbf{L}^\top \mathbf{x}_i - \mathbf{L}^\top \mathbf{x}_l\|^2)}\,, & \text{if } i \neq j\,; \\ 0\,, & \text{if } i = j\,. \end{cases}$$

Hence, the probability that $\mathbf{x}_i$ is correctly classified can be computed as

$$p_i = \sum_{\{j | y_j = y_i\}} p_{ij}\,.$$

NCA learns a linear transformation $\mathbf{L}$ by maximizing the expected number of examples correctly classified, i.e.,

$$\underset{\mathbf{L}}{\text{maximize}} \quad f(\mathbf{L}) = \sum_{i=1}^{N} p_i\,. \tag{2.4}$$

The gradient of f can be computed as

$$\nabla f(\mathbf{L}) = -2 \sum_{i=1}^{N} \left(p_i \sum_{l \neq i} p_{il} \mathbf{x}_{il} \mathbf{x}_{il}^\top - \sum_{\{j | y_j = y_i\}} p_{ij} \mathbf{x}_{ij} \mathbf{x}_{ij}^\top \right) \mathbf{L}\,,$$

where $\mathbf{x}_{il} = (\mathbf{x}_i - \mathbf{x}_l)(\mathbf{x}_i - \mathbf{x}_l)^\top$. Given the gradient, one can easily solve problem (2.4) by directly applying gradient descent optimization techniques, such as L-BFGS (Bertsekas, 1999).

2.4.3. Projected gradient descent

To guarantee the convexity, one might apply the projected gradient descent method (Goldstein, 1964). Instead of applying just a gradient step, we apply a

gradient descent followed by an orthogonal projection onto the PSD cone as

$$\begin{aligned}\mathbf{M}_{t+1/2} &= \mathbf{M}_t - \eta_t \nabla f(\mathbf{M}_t)\,,\\ \mathbf{M}_{t+1} &= \underset{\mathbf{A}\succcurlyeq 0}{\operatorname{argmin}} \quad \left\|\mathbf{A} - \mathbf{M}_{t+1/2}\right\|_F\,.\end{aligned}$$

The following lemma helps us to compute the projection in the second step.

Lemma 2.1. *Let* $\mathbf{M}$ *be a symmetric matrix in* $\mathbb{R}^{D\times D}$*, its projection onto the cone of PSD matrices has the closed-form solution*

$$\Pi_{\mathbb{S}^+}(\mathbf{M}) = \sum_{i:\lambda_i>0} \lambda_i \mathbf{u}_i \mathbf{u}_i^\top\,,$$

where $(\lambda_i, \mathbf{u}_i)$ *is the* i*-th pair of eigenvalue and eigenvector of* $\mathbf{M}$*.*

Proof. The proof can be found in (Boyd and Vandenberghe, 2004). □

Similarly to gradient descent, it turns out that one should carefully select the step size in order to guarantee the convergence of projected gradient descent.

Due to its simplicity and effectiveness, projected gradient descent has been widely used in the distance metric learning literature. As an example, consider the large-margin nearest neighbor (LMNN) model introduced by (Weinberger and Saul, 2009). The authors aim to learn a Mahalanobis distance metric for k-NN classification by exploiting the local structure of the data. LMNN does not make any assumption about the distribution of the data, making it one of the most widely-used distance metric learning methods (Torresani and Lee, 2007; Parameswaran and Weinberger, 2010; Nguyen et al., 2017b). The goal is to learn a distance metric under which each training example has k nearest neighbors that share the same class label (i.e., target neighbors), while pushing away those examples with different class labels (i.e., impostors). Finally, the problem is formulated as an instance of semidefinite programming

$$\underset{\mathbf{M}\succcurlyeq 0}{\text{minimize}} \quad (1-\mu) \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) + \mu \sum_{(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_l)\in\mathcal{R}} \Big[1 + d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) - d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_l)\Big]_+$$

where $\mathcal{S}$ denotes the set of pairwise constraints for minimizing the distance between $\mathbf{x}_i$ and its target neighbors, $\mathcal{R}$ denotes the set of triplet constraints for reducing the number of impostors, and $\mu \in [0,1]$ denotes the regularization hyperparameter. The authors developed a special-purpose solver based on projected subgradient descent to make LMNN more practical on large-scale problems. Although LMNN performs well in practice, it is sensitive to the way of selecting the target neighbors and cannot be applied to high-dimensional problems.

2.4.4. Stochastic gradient descent

One practical difficulty with gradient descent is that computing the gradient can be costly, especially when the number of examples is large. To make the gradient-based optimization technique tractable for large-scale problems, one should take into account not only the number of iterations, but also the computational cost of each iteration. Stochastic gradient descent (Bottou, 1991) provides a way to avoid the full-gradient computation by considering only a single loss term at a time, i.e., it minimizes the empirical loss based on one constraint or a small subset of constraints (mini-batch). By randomly choosing an example at each iteration, stochastic gradient descent can directly optimize the expected loss and remove the time complexity dependency on the size of problem.

In particular, one can rewrite the objective function in (2.2) as

$$f(\mathbf{W}) = \frac{1}{|\mathcal{R}|} \sum_{i=1}^{|\mathcal{R}|} \ell(\mathbf{W}, r_i) \,,$$

where $\ell(\mathbf{W}, r_i)$ denotes the loss function penalizing the violation of one constraint $r_i \in \mathcal{R}$. Instead of computing the gradient of $f(\mathbf{W})$ exactly, stochastic gradient descent estimates this gradient based on the gradient of $\ell(\mathbf{W}, r_i)$. At the t-th iteration, it operates as follows

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta_t \nabla \ell(\mathbf{W}_t, r_i) \,.$$

Although stochastic gradient descent has been successfully applied to speed up training time, it can still be computationally expensive when learning a Mahalanobis matrix because the solutions need to lie within the cone of PSD matrices.

There have been a few attempts to make stochastic gradient descent more practical in the context of distance metric learning. As an example, Qian et al. (2015a) introduced an approach to reduce the number of updates in stochastic gradient descent in order to improve the computational efficiency. More specifically, it computes the gradient $\nabla \ell(\mathbf{W}_t, r_i)$ and samples a binary random variable with a probability

$$p(Z_t = 1) = |\ell(\mathbf{W}_t, r_i)| \,.$$

The distance metric is only updated when $Z_t = 1$. Note that constraints with a large loss will have a high chance to be used for updating the distance metric.

2.4.5. Frank-Wolfe algorithms

The Frank-Wolfe algorithm was firstly developed by Frank and Wolfe (1956). In literature, it is also known as the conditional gradient algorithm or the reduced gradient algorithm. The Frank-Wolfe algorithm is an iterative procedure to minimize a convex and continuously differentiable function over a compact and convex set. At the current solution $\mathbf{W}_t$, the algorithm considers the linearization of the objective function f, and optimizes this linear function over the same domain, i.e.,

$$\begin{aligned}\mathbf{A} &= \operatorname*{argmin}_{\mathbf{W}} \quad f(\mathbf{W}_t) + \langle \nabla f(\mathbf{W}_t), \mathbf{W} - \mathbf{W}_t \rangle \,, \\ \mathbf{W}_{t+1} &= (1-\gamma_t)\mathbf{W}_t + \gamma_t \mathbf{A} \,,\end{aligned}$$

where γ_t is the step size. It is important to note that the minimizer of this linear function is at a vertex of the feasible domain. This helps to maintain the sparsity of the solution. For instance, in applications of low-rank approximation, one typically starts with a low-rank solution and increases the rank by at most one after each step (Jaggi, 2013). The latter is in contrast with projected gradient descent, which often starts with a high-rank solution and projects onto a low-rank space.

As an example, consider the Distance Metric Learning with Eigenvalue Optimization (DML-eig) model introduced by Ying and Li (2012). The authors proposed to maximize the minimal squared distances between dissimilar examples while keeping an upper bound for the sum of squared distances between similar examples, i.e.,

$$\begin{array}{ll}\underset{\mathbf{M}\succcurlyeq 0}{\text{maximize}} & \min\limits_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \\ \text{subject to} & \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \leq 1 \,.\end{array} \tag{2.5}$$

Adopting the authors' notation, let $\mathbf{X}_t = (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top$ and $\mathbf{X}_{\mathcal{S}} = \sum_{t\in\mathcal{S}} \mathbf{X}_t$, problem (2.5) can be rewritten as

$$\begin{array}{ll}\underset{\mathbf{M}\succcurlyeq 0}{\text{maximize}} & \min\limits_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \langle \mathbf{X}_t, \mathbf{M}\rangle \\ \text{subject to} & \langle \mathbf{X}_{\mathcal{S}}, \mathbf{M}\rangle \leq 1 \,.\end{array}$$

It has been shown that the above problem is equivalent to

$$\min_{\mathbf{u}\in\Delta} \quad \max_{\mathbf{S}\in\mathcal{P}} \left\langle \sum_{t\in\mathcal{D}} u_t \tilde{\mathbf{X}}_t, \mathbf{S} \right\rangle = \min_{\mathbf{u}\in\Delta} \quad \lambda_{\max}\left(\sum_{t\in\mathcal{D}} u_t \tilde{\mathbf{X}}_t \right) , \tag{2.6}$$

where

$$\Delta = \left\{ \mathbf{u} \in \mathbb{R}^{|\mathcal{D}|} \mid u_t \geq 0, \sum_{t\in\mathcal{D}} u_t = 1 \right\},$$

$$\mathcal{P} = \left\{ \mathbf{S} \succcurlyeq 0 \mid \operatorname{tr}(\mathbf{S}) = 1 \right\}.$$

$\tilde{\mathbf{X}}_t = \mathbf{X}_{\mathcal{S}}^{-1/2} \mathbf{X}_t \mathbf{X}_{\mathcal{S}}^{-1/2}$ and $\lambda_{\max}$ is the maximal eigenvalue. Problem (2.6) is well known as minimizing the maximal eigenvalue of a symmetric matrix (Overton, 1988). To make problem (2.6) more tractable, its smoothed version was introduced

$$\underset{\mathbf{S} \in \mathcal{P}}{\text{minimize}} \quad f(\mathbf{S}) = \mu \log \left(\sum_{t \in \mathcal{D}} \exp(-\langle \tilde{\mathbf{X}}_t, \mathbf{S} \rangle / \mu) \right),$$

where $\mu > 0$ denotes the smoothing parameter. To solve this positive semidefinite program over the spectrahedron $\mathcal{P}$, the authors use the Frank-Wolfe algorithm extended by Hazan (2008). At each step, finding the minimizer for linear functions is efficient because it corresponds to finding the largest singular vector of the gradient $\nabla f(\mathbf{S}_t)$, which can be approximated in $O(D^2)$ (Golub and Van Loan, 1996).

2.4.6. Bregman projection

Bregman projection refers to a family of iterative first-order algorithms developed by Bregman (1967). The idea is to optimize an objective function by choosing one constraint at each iteration, then performing a projection so that the chosen constraint is satisfied. We first introduce the Bregman matrix divergence, a measure of closeness between two matrices w.r.t. a strictly convex and differentiable function ϕ,

$$B_\phi(\mathbf{X}, \mathbf{Y}) = \phi(\mathbf{X}) - \phi(\mathbf{Y}) - \operatorname{tr}\big((\mathbf{X} - \mathbf{Y})^\top \nabla \phi(\mathbf{Y})\big).$$

For instance, by setting $\phi(\mathbf{X}) = \|\mathbf{X}\|_F^2$, the Bregman divergence results in the well-known squared Frobenius norm $\|\mathbf{X} - \mathbf{Y}\|_F^2$. At each iteration, the Bregman projection at a current solution $\mathbf{X}_t$ can be found by solving

$$\begin{aligned} &\text{minimize} && B_\phi(\mathbf{X}, \mathbf{X}_t) \\ &\text{subject to} && \operatorname{tr}(\mathbf{X}\mathbf{A}_i) \leq b_i \,. \end{aligned}$$

Note that only one constraint is involved in each iteration, making the optimization simpler. Instead of an orthogonal projection as in projected gradient descent, here the Bregman projection is being minimized. This optimization procedure is repeated by cycling through all constraints. Censor and Zenios (1997) showed that the method converges to a globally optimal solution under mild conditions.

As an example, consider the information-theoretic metric learning (ITML) model (Davis et al., 2007). The authors have successfully integrated pairwise constraints into a framework for learning a Mahalanobis distance metric. The idea

consists in minimizing the differential relative entropy between two multivariate Gaussian distributions subject to the pairwise constraints. More specifically, it regularizes the distance metric to be as close as possible to a given Mahalanobis distance metric, parameterized by $\mathbf{M}_0$. The closeness between $\mathbf{M}$ and $\mathbf{M}_0$ is measured using the LogDet divergence, which is defined as

$$\mathrm{LogDet}(\mathbf{M}, \mathbf{M}_0) = \mathrm{tr}(\mathbf{M}\mathbf{M}_0^{-1}) - \log(\det(\mathbf{M}\mathbf{M}_0^{-1})) - D\,,$$

where det denotes the determinant of the matrix. Note that the LogDet divergence corresponds to the Bregman divergence over positive definite matrices by setting $\phi(\mathbf{X}) = \log(\det(\mathbf{X}))$ (see Kulis et al., 2009c). In practice, $\mathbf{M}_0$ is often set to be the identity matrix, and thus, the regularization tries to keep the learned distance metric close to the Euclidean distance metric. Finally, must-link constraints (denoted by $\mathcal{M}$) and cannot-link constraints (denoted by $\mathcal{D}$) are introduced into the optimization problem as follows

$$\begin{array}{ll} \underset{\mathbf{M}\succcurlyeq 0}{\text{minimize}} & \mathrm{LogDet}(\mathbf{M}, \mathbf{M}_0) + \lambda \sum_{i,j} \xi_{ij} \\ \text{subject to} & \forall (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M} \colon d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \leq u + \xi_{ij}\,, \\ & \forall (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D} \colon d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq l - \xi_{ij}\,, \\ & \xi_{ij} \geq 0\,, \end{array} \tag{2.7}$$

where $u, l \geq 0$ are threshold parameters, λ is the regularization hyper-parameter, and ξ_{ij} are slack variables. Since the LogDet divergence is finite if and only if $\mathbf{M}$ is positive definite, it suggests a cheap way to satisfy the positive definiteness constraint on $\mathbf{M}$. Indeed, ITML does not need to explicitly constrain the learned matrices to be positive definite. The Bregman projection converges to the global minimum, and the resulting distance metric performs well in practice. However, ITML is sensitive to the way of choosing $\mathbf{M}_0$, which is usually done by hand.

PART II

DISTANCE METRIC LEARNING USING PAIRWISE CONSTRAINTS

3 Distance metric learning through maximization of the Jeffrey divergence

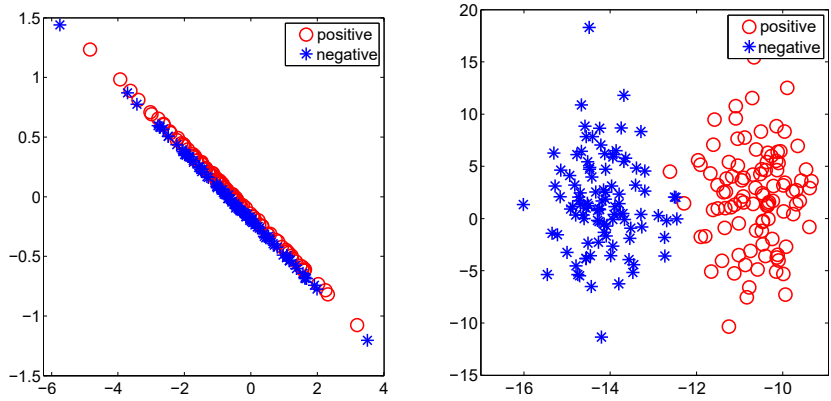
In this chapter, we propose an optimization framework for distance metric learning via linear transformations by maximizing the Jeffrey divergence between two multivariate Gaussian distributions derived from local pairwise constraints. In our method, the distance metric is trained on positive and negative difference spaces, which are built from the neighborhood of each training example, so that the local discriminative information is preserved. We show how to solve this problem with a closed-form solution rather than using tedious optimization procedures. The solution is easy to implement, and tractable for large-scale problems. Experimental results are presented for both a linear and a kernelized version of the proposed method for k-NN classification. We obtain classification accuracies superior to the state-of-the-art distance metric learning methods in several cases while being competitive in others.

The material of this chapter is based on the following publication:
Nguyen, B., Morell, C., and De Baets, B. (2017c). Supervised distance metric learning through maximization of the Jeffrey divergence. *Pattern Recognition*, 64:215–225

3.1. Motivation

We begin by introducing a simple two-class classification problem that motivates the key ideas in the proposed method. For this purpose, we construct a two-dimensional data set, containing 100 positive examples and 100 negative examples (see Fig. 3.1(a)). Both positive examples and negative examples follow a Gaussian distribution with means $\boldsymbol{\mu}_1 = (-1.250; 0.205)$ and $\boldsymbol{\mu}_2 = (0.60; 0.07)$, respectively, and the same covariance matrix $\boldsymbol{\Sigma} = (1.96, -0.55; -0.55, 0.16)$. The training accuracy of 5-NN using the Euclidean distance metric on this data set is very poor, only 64.0%. However, this performance can be dramatically improved by applying a linear transformation to the original data. In particular, using our method (as we will describe later) we obtain the linear transformation $\mathbf{A} = (20.11, -3.02; 70.22, 0.63)$, and consequently, the training accuracy is increased to 97.5% (see the resulting transformed data in Fig. 3.1(b)).

The key question is how to find such a linear transformation $\mathbf{A}$ (or, equivalently, the corresponding Mahalanobis matrix $\mathbf{M}$). Some insights can be obtained when carefully observing how the differences are distributed. Let us informally define



(a) Before **(b)** After

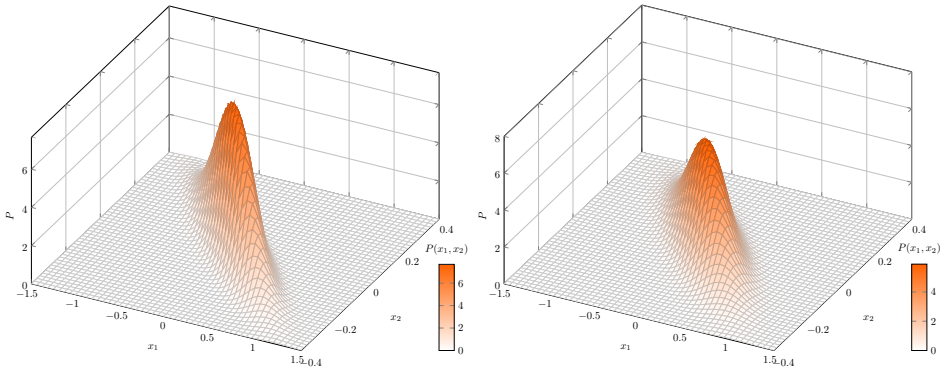
Figure 3.1: A synthetic data set illustrating the poor performance of the k-NN classifier using the Euclidean distance metric. The data set consists of 200 examples drawn from two aligned strips, each defining a different class. The red circles denote positive examples, whereas the blue asterisks denote negative examples. (a) data set before applying the linear transformation, (b) data set after applying the linear transformation.

the positive (resp. negative) *difference space* as the set of all differences $(\mathbf{x}_i - \mathbf{x}_j)$ between an example $\mathbf{x}_i$ and its nearest neighbors $\mathbf{x}_j$ from the same (resp. different) class (see Section 3.2 for the formal definitions). Here, we use five nearest neighbors with the same class label and five nearest neighbors with different class labels for each training example. Figure 3.2 shows the probability density function[1] of data belonging to the positive (Fig. 3.2(a)) and negative (Fig. 3.2(b)) difference spaces. It allows us to see how the differences are distributed before applying the linear transformation.

There is a slight difference between these two distributions. However, this difference clearly reveals itself after applying the linear transformation specified by $\mathbf{A}$ (see Fig. 3.3). Note that our illustration here is based on $k = 5$, but the same phenomenon occurs for other values of k. This particular example suggests a way to find such linear transformation, namely the one that maximizes the difference between these two distributions. The intuition is based on a two-class classification problem, however, it can be also used for multi-class classification problems since the difference spaces are built independently for any number of classes. In the rest of this chapter, we develop this idea. In short, our main contributions are the following.

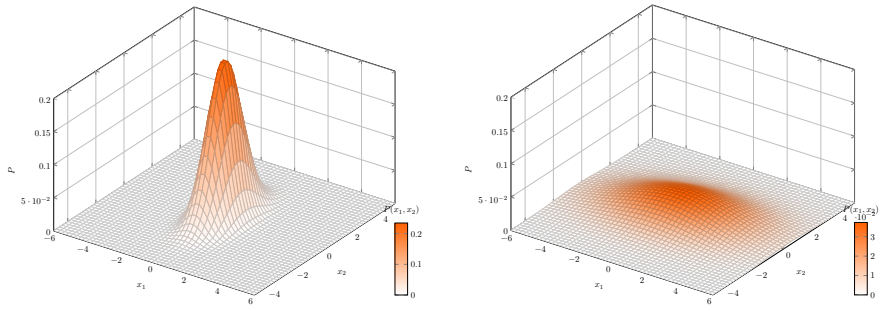
(i) We propose a novel distance metric learning method aimed at finding a linear transformation that maximizes the Jeffrey divergence between two multivariate Gaussian distributions derived from local pairwise constraints.

[1] For illustrative purposes, we use maximum likelihood to estimate the probability density function assuming that the data are normally distributed.



(a) Positive difference space **(b)** Negative difference space

Figure 3.2: Visualization of the probability density functions of the difference spaces before applying the linear transformation.



(a) Positive difference space **(b)** Negative difference space

Figure 3.3: Visualization of the probability density functions of the difference spaces after applying the linear transformation.

We formulate this task as an unconstrained optimization problem and show that it can be solved analytically (Subsection 3.3.1).

(ii) While the proposed method is limited to learn a global linear transformation, we extend it into a kernelized version to tackle nonlinear problems. We show that the kernelized version of the proposed method is more efficient and highly flexible by using the "kernel trick" (Subsection 3.3.2).

(iii) The resulting distance metric, when used in conjunction with k-NN, leads to significant improvements in the classification accuracy. We provide an extensive experimental validation to support this claim (Section 3.5). Several state-of-the-art distance metric learning methods (Section 3.4) have been used for a fair comparison.

3.2. Definitions

We first introduce some definitions in order to develop our proposal.

Definition 3.1 (k-positive neighborhood)**.** *Let $k \in \mathbb{N}$ such that $k \geq 1$. The k-positive neighborhood of $\mathbf{x}_i \in \mathcal{X}$ is the set $\mathcal{V}_k^+(\mathbf{x}_i)$ consisting of the k nearest neighbors of $\mathbf{x}_i$ in the input space $\mathcal{X} \setminus \{\mathbf{x}_i\}$, whose class label is equal to y_i.*

Definition 3.2 (k-negative neighborhood)**.** *Let $k \in \mathbb{N}$ such that $k \geq 1$. The k-negative neighborhood of $\mathbf{x}_i \in \mathcal{X}$ is the set $\mathcal{V}_k^-(\mathbf{x}_i)$ consisting of the k nearest neighbors of $\mathbf{x}_i$ in the input space $\mathcal{X}$, whose class label is not equal to y_i.*

The set of all possible differences for any example $\mathbf{x}_i$ and its k-positive neighborhood is called the k-positive difference space. The set of all possible differences for any example $\mathbf{x}_i$ and its k-negative neighborhood is called the k-negative difference space. They are formally defined hereafter.

Definition 3.3 (k-positive difference space)**.** *The k-positive difference space is the following set:*

$$\mathcal{S} = \left\{ \mathbf{x}_i - \mathbf{x}_j \mid \mathbf{x}_i \in \mathcal{X} \text{ and } \mathbf{x}_j \in \mathcal{V}_k^+(\mathbf{x}_i) \right\}.$$

Definition 3.4 (k-negative difference space)**.** *The k-negative difference space is the following set:*

$$\mathcal{D} = \left\{ \mathbf{x}_i - \mathbf{x}_j \mid \mathbf{x}_i \in \mathcal{X} \text{ and } \mathbf{x}_j \in \mathcal{V}_k^-(\mathbf{x}_i) \right\}.$$

3.3. Proposed method

Motivated by the toy example above, the proposed method is based on learning a linear transformation that maximizes the difference between the probability distribution on the positive difference space and that on the negative difference space. Such difference is often measured by the well-known Kullback-Leibler divergence (Kullback and Leibler, 1951), which is widely used in many machine learning applications, such as information retrieval (Bigi et al., 2000), text categorization (Bigi, 2003), particularly in the classification of multimedia data with support vector machines (Moreno et al., 2004). In the distance metric learning context, the Kullback-Leibler divergence was introduced by Davis et al. (2007) in ITML and later it was motivated in several other distance metric learning methods (Qi et al., 2009; Jain et al., 2009; Mei et al., 2014; Globerson and Roweis, 2006). Since the Kullback-Leibler divergence can yield substantially different values by changing the order of its arguments, in this work we use the symmetric Kullback-Leibler divergence (also called Jeffrey divergence).

3.3.1. Problem formulation

Let P denote the distribution of the differences in the positive difference space and let Q denote the distribution of the differences in the negative difference space. We assume that P and Q are multivariate Gaussian distributions with zero mean[2] and covariance matrices $\mathbf{\Sigma}_{\mathcal{S}}$ and $\mathbf{\Sigma}_{\mathcal{D}}$, respectively. As described by Duda et al. (2012), linear combinations of jointly normally distributed random variables are normally distributed, even if the variables are not independent. Suppose we perform a linear transformation $\mathbf{x}' = \mathbf{A}^\top \mathbf{x}$, then the transformed distributions $P_{\mathbf{A}}$ and $Q_{\mathbf{A}}$ have zero mean and covariance matrices $\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{S}} \mathbf{A}$ and $\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{D}} \mathbf{A}$, respectively. Our goal is to find the linear transformation that maximizes the Jeffrey divergence between $P_{\mathbf{A}}$ and $Q_{\mathbf{A}}$:

$$\underset{\mathbf{A} \in \mathbb{R}^{D \times m}}{\operatorname{argmax}} \quad f(\mathbf{A}) = \mathrm{KL}(P_{\mathbf{A}}, Q_{\mathbf{A}}) + \mathrm{KL}(Q_{\mathbf{A}}, P_{\mathbf{A}})\,. \tag{3.1}$$

As shown in A.1, the Jeffrey divergence between $P_{\mathbf{A}}$ and $Q_{\mathbf{A}}$ can be calculated as:

$$f(\mathbf{A}) = \frac{1}{2}\mathrm{tr}\left((\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{S}} \mathbf{A})^{-1}(\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{D}} \mathbf{A}) + (\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{D}} \mathbf{A})^{-1}(\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{S}} \mathbf{A})\right) - m\,.$$

Since the parameter m in $f(\mathbf{A})$ is constant, we can simplify problem (3.1) to:

$$\underset{\mathbf{A} \in \mathbb{R}^{D \times m}}{\operatorname{argmax}} J(\mathbf{A}) = \mathrm{tr}\left((\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{S}} \mathbf{A})^{-1}(\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{D}} \mathbf{A}) + (\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{D}} \mathbf{A})^{-1}(\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{S}} \mathbf{A})\right)\,. \tag{3.2}$$

Taking the derivative of $J(\mathbf{A})$ with respect to $\mathbf{A}$, by using (Petersen and Pedersen, 2012, Eq. (2.4.4)), we obtain

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{A}} J(\mathbf{A}) \\
&= -2\mathbf{\Sigma}_{\mathcal{S}}\mathbf{A}(\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{S}}\mathbf{A})^{-1}\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{D}}\mathbf{A}(\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{S}}\mathbf{A})^{-1} + 2\mathbf{\Sigma}_{\mathcal{D}}\mathbf{A}(\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{S}}\mathbf{A})^{-1} \\
&\quad - 2\mathbf{\Sigma}_{\mathcal{D}}\mathbf{A}(\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{D}}\mathbf{A})^{-1}\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{S}}\mathbf{A}(\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{D}}\mathbf{A})^{-1} + 2\mathbf{\Sigma}_{\mathcal{S}}\mathbf{A}(\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{D}}\mathbf{A})^{-1} \\
&= \underbrace{\left(2\mathbf{\Sigma}_{\mathcal{D}}\mathbf{A}\mathbf{\Sigma}_{2\mathcal{S}}^{-1} - 2\mathbf{\Sigma}_{\mathcal{S}}\mathbf{A}\mathbf{\Sigma}_{2\mathcal{S}}^{-1}\mathbf{\Sigma}_{2\mathcal{D}}\mathbf{\Sigma}_{2\mathcal{S}}^{-1}\right)}_{\text{first term}} + \underbrace{\left(2\mathbf{\Sigma}_{\mathcal{S}}\mathbf{A}\mathbf{\Sigma}_{2\mathcal{D}}^{-1} - 2\mathbf{\Sigma}_{\mathcal{D}}\mathbf{A}\mathbf{\Sigma}_{2\mathcal{D}}^{-1}\mathbf{\Sigma}_{2\mathcal{S}}\mathbf{\Sigma}_{2\mathcal{D}}^{-1}\right)}_{\text{second term}},
\end{aligned}$$

where $\mathbf{\Sigma}_{2\mathcal{S}} = \mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{S}} \mathbf{A}$ and $\mathbf{\Sigma}_{2\mathcal{D}} = \mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{D}} \mathbf{A}$. The optimal matrix $\mathbf{A}$ should satisfy $\partial J(\mathbf{A})/\partial \mathbf{A} = \mathbf{0}$. Although it seems very complex to solve $\partial J(\mathbf{A})/\partial \mathbf{A} = \mathbf{0}$ for $\mathbf{A}$, the first and second terms can be made zero separately as follows. For the first

[2] In machine learning, Gaussian distributions are widely used to model continuous random variables as they incorporate the least amount of prior knowledge into a model (Goodfellow et al., 2016). In practice, if an example $\mathbf{x}_i$ belongs to the neighborhood of an example $\mathbf{x}_j$, then $\mathbf{x}_j$ usually belongs to the neighborhood of $\mathbf{x}_i$ as well. As a consequence, in practice, the distributions will be symmetric with zero mean.

term, it should hold

$$\boldsymbol{\Sigma}_{\mathcal{S}}^{-1}\boldsymbol{\Sigma}_{\mathcal{D}}\mathbf{A} = \mathbf{A}\boldsymbol{\Sigma}_{2\mathcal{S}}^{-1}\boldsymbol{\Sigma}_{2\mathcal{D}}\,. \tag{3.3}$$

For the second term, it should hold

$$\boldsymbol{\Sigma}_{\mathcal{D}}^{-1}\boldsymbol{\Sigma}_{\mathcal{S}}\mathbf{A} = \mathbf{A}\boldsymbol{\Sigma}_{2\mathcal{D}}^{-1}\boldsymbol{\Sigma}_{2\mathcal{S}}\,. \tag{3.4}$$

Before solving these problems, we would like to introduce a theorem, which will help us to find the solution to these problems.

Theorem 3.1. *Let $\mathbf{A} \in \mathbb{R}^{D\times m}$ be a matrix of m linearly independent eigenvectors of $\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_2$. Then it holds that*

$$\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_2\mathbf{A} = \mathbf{A}(\mathbf{A}^\top\boldsymbol{\Sigma}_1\mathbf{A})^{-1}(\mathbf{A}^\top\boldsymbol{\Sigma}_2\mathbf{A})\,.$$

Proof. Let $\mathbf{D} \in \mathbb{R}^{m\times m}$ be the diagonal matrix containing the corresponding m eigenvalues of $\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_2$, then it holds by definition

$$\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_2\mathbf{A} = \mathbf{A}\mathbf{D}\,. \tag{3.5}$$

Multiplying both sides of Eq. (3.5) by the matrix $\mathbf{A}^\top\boldsymbol{\Sigma}_1$ yields

$$\mathbf{A}^\top\boldsymbol{\Sigma}_2\mathbf{A} = \mathbf{A}^\top\boldsymbol{\Sigma}_1\mathbf{A}\mathbf{D}\,,$$

or, equivalently,

$$(\mathbf{A}^\top\boldsymbol{\Sigma}_1\mathbf{A})^{-1}(\mathbf{A}^\top\boldsymbol{\Sigma}_2\mathbf{A}) = \mathbf{D}\,. \tag{3.6}$$

Multiplying both sides of Eq. (3.6) by $\mathbf{A}$ yields

$$\mathbf{A}(\mathbf{A}^\top\boldsymbol{\Sigma}_1\mathbf{A})^{-1}(\mathbf{A}^\top\boldsymbol{\Sigma}_2\mathbf{A}) = \mathbf{A}\mathbf{D}\,. \tag{3.7}$$

Substituting (3.5) into (3.7) leads to

$$\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_2\mathbf{A} = \mathbf{A}(\mathbf{A}^\top\boldsymbol{\Sigma}_1\mathbf{A})^{-1}(\mathbf{A}^\top\boldsymbol{\Sigma}_2\mathbf{A})\,.$$

This concludes the proof. □

Consequently, we have the following corollary.

Corollary 3.1. *Let $\mathbf{A} \in \mathbb{R}^{D\times m}$ be a matrix of m linearly independent eigenvectors of $\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_2$ and $\mathbf{D} \in \mathbb{R}^{m\times m}$ be the diagonal matrix containing the corresponding m eigenvalues. Then it holds that*

$$\operatorname{tr}\Big((\mathbf{A}^\top\boldsymbol{\Sigma}_1\mathbf{A})^{-1}(\mathbf{A}^\top\boldsymbol{\Sigma}_2\mathbf{A})\Big) = \operatorname{tr}(\mathbf{D})\,.$$

Proof. By definition, it holds that

$$\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_2\mathbf{A} = \mathbf{A}\mathbf{D}\,. \tag{3.8}$$

Substituting $\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_2\mathbf{A} = \mathbf{A}(\mathbf{A}^\top\boldsymbol{\Sigma}_1\mathbf{A})^{-1}(\mathbf{A}^\top\boldsymbol{\Sigma}_2\mathbf{A})$ into Eq. (3.8) leads to

$$\mathbf{A}(\mathbf{A}^\top\boldsymbol{\Sigma}_1\mathbf{A})^{-1}(\mathbf{A}^\top\boldsymbol{\Sigma}_2\mathbf{A}) = \mathbf{A}\mathbf{D}\,. \tag{3.9}$$

Multiplying both sides of Eq. (3.9) by $\mathbf{A}^\top$ yields

$$(\mathbf{A}^\top\boldsymbol{\Sigma}_1\mathbf{A})^{-1}(\mathbf{A}^\top\boldsymbol{\Sigma}_2\mathbf{A}) = \mathbf{D}\,,$$

and hence

$$\mathrm{tr}\left((\mathbf{A}^\top\boldsymbol{\Sigma}_1\mathbf{A})^{-1}(\mathbf{A}^\top\boldsymbol{\Sigma}_2\mathbf{A})\right) = \mathrm{tr}(\mathbf{D})\,.$$

We conclude the proof. □

Note that Theorem 3.1 and Corollary 3.1 hold even when some eigenvalues are zero. In order to satisfy Eqs. (3.3) and (3.4) simultaneously, according to Theorem 3.1, $\mathbf{A}$ can be a matrix of eigenvectors of both $\boldsymbol{\Sigma}_{\mathcal{S}}^{-1}\boldsymbol{\Sigma}_{\mathcal{D}}$ and $\boldsymbol{\Sigma}_{\mathcal{D}}^{-1}\boldsymbol{\Sigma}_{\mathcal{S}}$, because the latter two share the same eigenvectors and their diagonal matrices of corresponding eigenvalues are $\boldsymbol{\Lambda}$ and $\boldsymbol{\Lambda}^{-1}$ as they are related by $(\boldsymbol{\Sigma}_{\mathcal{S}}^{-1}\boldsymbol{\Sigma}_{\mathcal{D}})^{-1} = \boldsymbol{\Sigma}_{\mathcal{D}}^{-1}\boldsymbol{\Sigma}_{\mathcal{S}}$. Therefore, by selecting m linearly independent eigenvectors of $\boldsymbol{\Sigma}_{\mathcal{S}}^{-1}\boldsymbol{\Sigma}_{\mathcal{D}}$, we satisfy both equations simultaneously and consequently it holds that $\partial J(\mathbf{A})/\partial\mathbf{A} = \mathbf{0}$. According to Corollary 3.1, the value of $J(\mathbf{A})$ is:

$$\begin{aligned}
J(\mathbf{A}) &= \mathrm{tr}\left((\mathbf{A}^\top\boldsymbol{\Sigma}_{\mathcal{S}}\mathbf{A})^{-1}(\mathbf{A}^\top\boldsymbol{\Sigma}_{\mathcal{D}}\mathbf{A}) + (\mathbf{A}^\top\boldsymbol{\Sigma}_{\mathcal{D}}\mathbf{A})^{-1}(\mathbf{A}^\top\boldsymbol{\Sigma}_{\mathcal{S}}\mathbf{A})\right) \\
&= \mathrm{tr}\left((\mathbf{A}^\top\boldsymbol{\Sigma}_{\mathcal{S}}\mathbf{A})^{-1}(\mathbf{A}^\top\boldsymbol{\Sigma}_{\mathcal{D}}\mathbf{A})\right) + \mathrm{tr}\left((\mathbf{A}^\top\boldsymbol{\Sigma}_{\mathcal{D}}\mathbf{A})^{-1}(\mathbf{A}^\top\boldsymbol{\Sigma}_{\mathcal{S}}\mathbf{A})\right) \\
&= \mathrm{tr}(\boldsymbol{\Lambda}) + \mathrm{tr}(\boldsymbol{\Lambda}^{-1}) \\
&= \sum_{i=1}^{m}\left(\lambda_i + \frac{1}{\lambda_i}\right)\,.
\end{aligned}$$

In order to maximize $J(\mathbf{A})$, we must select the m linearly independent eigenvectors of $\boldsymbol{\Sigma}_{\mathcal{S}}^{-1}\boldsymbol{\Sigma}_{\mathcal{D}}$ corresponding to the m largest values of $(\lambda_i + 1/\lambda_i)$. The eigenvalue λ_i of $\boldsymbol{\Sigma}_{\mathcal{S}}^{-1}\boldsymbol{\Sigma}_{\mathcal{D}}$ can be found by solving the equation

$$\boldsymbol{\Sigma}_{\mathcal{D}}\mathbf{w}_i = \lambda_i\boldsymbol{\Sigma}_{\mathcal{S}}\mathbf{w}_i\,,$$

where $\mathbf{w}_i$ is the corresponding eigenvector. It is also known as a generalized eigenvalue problem. Note that we might select the target dimensionality m using a validation set (subset of the training set) to find the dimensionality that realizes the best performance.

As shown in (Fukunaga, 1990), each eigenvalue λ_i of $\mathbf{\Sigma}_{\mathcal{S}}^{-1}\mathbf{\Sigma}_{\mathcal{D}}$ represents the ratio between the variances of Q and P along its corresponding eigenvector $\mathbf{w}_i$. Let $\omega_i^{\mathcal{S}}$ be the variance of P and let $\omega_i^{\mathcal{D}}$ be the variance of Q along $\mathbf{w}_i$. When these variances are the same, λ_i becomes 1 and $(\lambda_i + 1/\lambda_i)$ becomes 2, which is the minimum value. Otherwise, when $\omega_i^{\mathcal{S}}$ is larger or smaller than $\omega_i^{\mathcal{D}}$, then λ_i becomes smaller or larger than 1. Consequently, $(\lambda_i + 1/\lambda_i)$ becomes larger than 2 in any case. This intuition yields a straightforward explanation of why our formulation is effective in extracting features that realize significant differences between two distributions P and Q.

Using maximum likelihood estimation, the covariance matrices $\mathbf{\Sigma}_{\mathcal{S}}$ and $\mathbf{\Sigma}_{\mathcal{D}}$ are computed as follows:

$$\mathbf{\Sigma}_{\mathcal{S}} = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{n} \sum_{\mathbf{x}_j \in \mathcal{V}_k^+(\mathbf{x}_i)} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top , \tag{3.10}$$

$$\mathbf{\Sigma}_{\mathcal{D}} = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{n} \sum_{\mathbf{x}_j \in \mathcal{V}_k^-(\mathbf{x}_i)} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top . \tag{3.11}$$

We refer to the proposed method as *Distance Metric Learning through Maximization of the Jeffrey divergence* (DMLMJ). A simplified pseudo-code implementation of DMLMJ is given in Algorithm 1.

Algorithm 1 DMLMJ

Input: Training set $\mathcal{X}$, $\mathcal{Y}$; number of neighbors k; desired dimensionality m
Output: $\mathbf{A}$

1: Build the difference spaces
2: $\quad \mathcal{S} \leftarrow \{\mathbf{x}_i - \mathbf{x}_j \mid \mathbf{x}_i \in \mathcal{X} \text{ and } \mathbf{x}_j \in \mathcal{V}_k^+(\mathbf{x}_i)\}$,
3: $\quad \mathcal{D} \leftarrow \{\mathbf{x}_i - \mathbf{x}_j \mid \mathbf{x}_i \in \mathcal{X} \text{ and } \mathbf{x}_j \in \mathcal{V}_k^-(\mathbf{x}_i)\}$.
4: Estimate the covariance matrices
5: $\quad \mathbf{\Sigma}_{\mathcal{S}} \leftarrow \frac{1}{|\mathcal{S}|} \sum_{\mathbf{u}_i \in \mathcal{S}} \mathbf{u}_i \mathbf{u}_i^\top$,
6: $\quad \mathbf{\Sigma}_{\mathcal{D}} \leftarrow \frac{1}{|\mathcal{D}|} \sum_{\mathbf{u}_i \in \mathcal{D}} \mathbf{u}_i \mathbf{u}_i^\top$.
7: Find $\mathbf{W}$ and $\boldsymbol{\lambda}$ of $\mathbf{\Sigma}_{\mathcal{S}}^{-1}\mathbf{\Sigma}_{\mathcal{D}}$ using the generalized eigenvalue decomposition.
8: Construct $\mathbf{A}$, whose columns are the m column vectors $\mathbf{w}_i \in \mathbf{W}$ corresponding to the m largest values of $(\lambda_i + 1/\lambda_i)$.

3.3.2. Nonlinear distance metric learning

Many real-world data sets contain nonlinearities that linear transformations are unable to capture (He et al., 2004; Torresani and Lee, 2007). In this section, we will derive the kernelized version of DMLMJ to tackle nonlinear problems. The idea of kernelization is to learn a linear transformation in the nonlinear feature space induced by a kernel function. This idea has been successfully applied in many other

contexts, including nonlinear kernel principal component analysis (Schölkopf et al., 1998), kernel Fisher discriminant analysis (Mika et al., 1999), and particularly in SVMs (Schölkopf and Smola, 2001).

Let ϕ be a nonlinear function that maps the input space from $\mathbb{R}^D$ into the feature space $\mathcal{F}$,

$$\begin{aligned} \phi\colon \mathbb{R}^D &\to \mathcal{F} \\ \mathbf{x} &\mapsto \phi(\mathbf{x})\,. \end{aligned}$$

As a result, each training example is mapped into a potentially nonlinear feature space, in which we can perform linear transformations. Let $\mathbf{\Phi} = (\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n))$ be the matrix whose columns are the images of the n training examples under ϕ. Given two points $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^D$, the function that returns the inner product between their images in $\mathcal{F}$ is known as the kernel function, $\ker(\mathbf{u}, \mathbf{v}) = \langle \phi(\mathbf{u}), \phi(\mathbf{v}) \rangle$. To find the linear transformation $\mathbf{A}$ in the feature space $\mathcal{F}$, we aim to solve the following problem

$$\underset{\mathbf{A} \in \mathbb{R}^{N \times m}}{\operatorname{argmax}}\ J(\mathbf{A}) = \operatorname{tr}\Big((\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{S}}^{\Phi} \mathbf{A})^{-1} (\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{D}}^{\Phi} \mathbf{A}) + (\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{D}}^{\Phi} \mathbf{A})^{-1} (\mathbf{A}^\top \mathbf{\Sigma}_{\mathcal{S}}^{\Phi} \mathbf{A}) \Big)\,, \quad (3.12)$$

where the covariance matrices $\mathbf{\Sigma}_{\mathcal{S}}^{\Phi}$ and $\mathbf{\Sigma}_{\mathcal{D}}^{\Phi}$ are defined as:

$$\mathbf{\Sigma}_{\mathcal{S}}^{\Phi} = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{n} \sum_{\phi(\mathbf{x}_j) \in \mathcal{V}_k^+(\phi(\mathbf{x}_i))} \Big[\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j) \Big] \Big[\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j) \Big]^\top,$$

$$\mathbf{\Sigma}_{\mathcal{D}}^{\Phi} = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{n} \sum_{\phi(\mathbf{x}_j) \in \mathcal{V}_k^-(\phi(\mathbf{x}_i))} \Big[\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j) \Big] \Big[\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j) \Big]^\top.$$

Note that the dimensionality of $\mathcal{F}$ can be very high or even infinite. In such case, it becomes hard to estimate $\mathbf{A}$, $\mathbf{\Sigma}_{\mathcal{S}}^{\Phi}$ and $\mathbf{\Sigma}_{\mathcal{D}}^{\Phi}$ directly in the feature space due to the increased computational complexity. Moreover, the mapping ϕ is usually unknown. To overcome these problems, we use the same trick as in (Schölkopf et al., 1998) for kernel principal component analysis. Instead of explicitly expressing the linear transformation, we find a solution that lies in the span of all training examples. That is, each column vector $\mathbf{w}_i$ of $\mathbf{A}$ is represented as a linear combination of training examples in $\mathcal{F}$:

$$\mathbf{w}_i = \sum_{j=1}^{n} B_{ji} \phi(\mathbf{x}_j)\,,$$

where the matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$ contains the expansion coefficients. Now, we need to find the matrix $\mathbf{B}$. Let $\mathbf{U} = \mathbf{\Phi}^\top \mathbf{\Sigma}_{\mathcal{S}}^{\mathbf{\Phi}} \mathbf{\Phi}$ and $\mathbf{V} = \mathbf{\Phi}^\top \mathbf{\Sigma}_{\mathcal{D}}^{\mathbf{\Phi}} \mathbf{\Phi}$, then the objective

function in (3.12) becomes:

$$\begin{aligned} J(\mathbf{A}) &= \operatorname{tr}\left((\mathbf{A}^\top \boldsymbol{\Sigma}_{\mathcal{S}}^{\Phi}\mathbf{A})^{-1}(\mathbf{A}^\top \boldsymbol{\Sigma}_{\mathcal{D}}^{\Phi}\mathbf{A}) + (\mathbf{A}^\top \boldsymbol{\Sigma}_{\mathcal{D}}^{\Phi}\mathbf{A})^{-1}(\mathbf{A}^\top \boldsymbol{\Sigma}_{\mathcal{S}}^{\Phi}\mathbf{A})\right) \\ &= \operatorname{tr}\left((\mathbf{B}^\top \Phi^\top \boldsymbol{\Sigma}_{\mathcal{S}}^{\Phi}\Phi\mathbf{B})^{-1}(\mathbf{B}^\top \Phi^\top \boldsymbol{\Sigma}_{\mathcal{D}}^{\Phi}\Phi\mathbf{B}) + (\mathbf{B}^\top \Phi^\top \boldsymbol{\Sigma}_{\mathcal{D}}^{\Phi}\Phi\mathbf{B})^{-1}(\mathbf{B}^\top \Phi^\top \boldsymbol{\Sigma}_{\mathcal{S}}^{\Phi}\Phi\mathbf{B})\right) \\ &= \operatorname{tr}\left((\mathbf{B}^\top \mathbf{U}\mathbf{B})^{-1}(\mathbf{B}^\top \mathbf{V}\mathbf{B}) + (\mathbf{B}^\top \mathbf{V}\mathbf{B})^{-1}(\mathbf{B}^\top \mathbf{U}\mathbf{B})\right). \end{aligned}$$

Hence, problem (3.12) can be rewritten as:

$$\underset{\mathbf{B}\in\mathbb{R}^{n\times m}}{\operatorname{argmax}} \quad J(\mathbf{B}) = \operatorname{tr}\left((\mathbf{B}^\top \mathbf{U}\mathbf{B})^{-1}(\mathbf{B}^\top \mathbf{V}\mathbf{B}) + (\mathbf{B}^\top \mathbf{V}\mathbf{B})^{-1}(\mathbf{B}^\top \mathbf{U}\mathbf{B})\right).$$

Analogously to (3.2), this problem can be solved by finding the eigenvectors of $\mathbf{U}^{-1}\mathbf{V}$. The maximizing solution is a matrix containing the m eigenvectors of $\mathbf{U}^{-1}\mathbf{V}$ corresponding to the m largest values of $(\lambda_i + 1/\lambda_i)$, where λ_i are eigenvalues of $\mathbf{U}^{-1}\mathbf{V}$. We refer to this method as *Kernel Distance Metric Learning through Maximization of the Jeffrey divergence* (KDMLMJ). A simplified pseudo-code implementation of KDMLMJ is given in Algorithm 2.

Moreover, the matrices $\mathbf{U}$ and $\mathbf{V}$ can be expressed as:

$$\begin{aligned} \mathbf{U} &= \boldsymbol{\Phi}^\top \boldsymbol{\Sigma}_{\mathcal{S}}^{\boldsymbol{\Phi}} \boldsymbol{\Phi} \\ &= \frac{1}{|\mathcal{S}|}\sum_{i=1}^{n} \sum_{\phi(\mathbf{x}_j)\in\mathcal{V}_k^+(\phi(\mathbf{x}_i))} \left[\boldsymbol{\Phi}^\top\phi(\mathbf{x}_i) - \boldsymbol{\Phi}^\top\phi(\mathbf{x}_j)\right]\left[\boldsymbol{\Phi}^\top\phi(\mathbf{x}_i) - \boldsymbol{\Phi}^\top\phi(\mathbf{x}_j)\right]^\top \\ &= \frac{1}{|\mathcal{S}|}\sum_{i=1}^{n} \sum_{\phi(\mathbf{x}_j)\in\mathcal{V}_k^+(\phi(\mathbf{x}_i))} \left[K(\mathbf{x}_i) - K(\mathbf{x}_j)\right]\left[K(\mathbf{x}_i) - K(\mathbf{x}_j)\right]^\top, \end{aligned} \tag{3.13}$$

and

$$\begin{aligned} \mathbf{V} &= \boldsymbol{\Phi}^\top \boldsymbol{\Sigma}_{\mathcal{D}}^{\boldsymbol{\Phi}} \boldsymbol{\Phi} \\ &= \frac{1}{|\mathcal{D}|}\sum_{i=1}^{n} \sum_{\phi(\mathbf{x}_j)\in\mathcal{V}_k^-(\phi(\mathbf{x}_i))} \left[\boldsymbol{\Phi}^\top\phi(\mathbf{x}_i) - \boldsymbol{\Phi}^\top\phi(\mathbf{x}_j)\right]\left[\boldsymbol{\Phi}^\top\phi(\mathbf{x}_i) - \boldsymbol{\Phi}^\top\phi(\mathbf{x}_j)\right]^\top \\ &= \frac{1}{|\mathcal{D}|}\sum_{i=1}^{n} \sum_{\phi(\mathbf{x}_j)\in\mathcal{V}_k^-(\phi(\mathbf{x}_i))} \left[K(\mathbf{x}_i) - K(\mathbf{x}_j)\right]\left[K(\mathbf{x}_i) - K(\mathbf{x}_j)\right]^\top, \end{aligned} \tag{3.14}$$

where

$$\begin{aligned} K(\mathbf{u}) = \boldsymbol{\Phi}^\top\phi(\mathbf{u}) &= \left[\langle\phi(\mathbf{x}_1),\phi(\mathbf{u})\rangle,\ldots,\langle\phi(\mathbf{x}_n),\phi(\mathbf{u})\rangle\right]^\top \\ &= \left[\ker(\mathbf{x}_1,\mathbf{u}),\ldots,\ker(\mathbf{x}_n,\mathbf{u})\right]^\top. \end{aligned}$$

Since $\mathbf{U}$ and $\mathbf{V}$ are expressed in terms of inner products, we can use a kernel

function for mapping all examples and apply a kernel trick as in support vector machines (Schölkopf and Smola, 2001), or kernel principal component analysis (Schölkopf et al., 1998). Possible kernel functions are Gaussian radial basis function kernels (RBF), $\ker(\mathbf{u}, \mathbf{v}) = \exp(-\|\mathbf{u} - \mathbf{v}\|^2/\sigma)$, or polynomial kernels, $\ker(\mathbf{u}, \mathbf{v}) = \langle \mathbf{u}, \mathbf{v} \rangle^b$, for some positive constants $\sigma \in \mathbb{R}$ and $b \in \mathbb{N}$, respectively (see Schölkopf and Smola, 2001) and the references therein for other kernel functions).

Finally, the Mahalanobis distance in the feature space $\mathcal{F}$ is computed as:

$$\begin{aligned} d^2_{\mathbf{M}}\big(\phi(\mathbf{u}), \phi(\mathbf{v})\big) &= \Big\| \mathbf{A}^\top \big(\phi(\mathbf{u}) - \phi(\mathbf{v})\big) \Big\|^2 \\ &= \Big[\mathbf{A}^\top \big(\phi(\mathbf{u}) - \phi(\mathbf{v})\big) \Big]^\top \Big[\mathbf{A}^\top \big(\phi(\mathbf{u}) - \phi(\mathbf{v})\big) \Big] \\ &= \Big[\mathbf{B}^\top \big(\Phi^\top \phi(\mathbf{u}) - \Phi^\top \phi(\mathbf{v})\big) \Big]^\top \Big[\mathbf{B}^\top \big(\Phi^\top \phi(\mathbf{u}) - \Phi^\top \phi(\mathbf{v})\big) \Big] \\ &= \Big[K(\mathbf{u}) - K(\mathbf{v}) \Big]^\top \mathbf{B}\mathbf{B}^\top \Big[K(\mathbf{u}) - K(\mathbf{v}) \Big] . \end{aligned}$$

Algorithm 2 KDMLMJ

Input: Training set $\mathcal{X}$, $\mathcal{Y}$; number of neighbors k; desired dimensionality m; kernel function ker

Output: $\mathbf{B}$

1: Compute the matrix $\mathbf{U}$ as in (3.13) and the matrix $\mathbf{V}$ as in (3.14) using the kernel function ker.
2: Find the eigenvector matrix $\mathbf{W}$ and the eigenvalue vector $\boldsymbol{\lambda}$ of $\mathbf{U}^{-1}\mathbf{V}$ using the generalized eigenvalue decomposition.
3: Construct the matrix $\mathbf{B}$, whose columns are the m column vectors $\mathbf{w}_i \in \mathbf{W}$ corresponding to the m largest values of $(\lambda_i + 1/\lambda_i)$.

3.3.3. Regularization

To get a stable solution $(\lambda_i + 1/\lambda_i)$, where λ_i are eigenvalues of $\boldsymbol{\Sigma}_{\mathcal{S}}^{-1}\boldsymbol{\Sigma}_{\mathcal{D}}$, the covariance matrices $\boldsymbol{\Sigma}_{\mathcal{S}}$ and $\boldsymbol{\Sigma}_{\mathcal{D}}$ are required to be non-singular, which is clearly not always the case. Similarly as Mika et al. (1999), we propose a regularization technique to avoid this problem by adding some constant value $\alpha \in]0, 1[$ to the diagonal of the covariance matrix. That is, instead of using the covariance matrix directly, we use

$$\hat{\boldsymbol{\Sigma}} = (1 - \alpha)\boldsymbol{\Sigma} + \alpha \mathbf{I} .$$

Essentially, it renders the covariance matrix positive definite. Therefore, we make sure that all eigenvalues are sufficiently far from zero, and as a consequence, avoid numerical instability in computing the inverse.

3.3.4. Computational complexity

We analyze the computational complexity of DMLMJ. The difference spaces $\mathcal{S}$ and $\mathcal{D}$ can be built with a time complexity of $O(kn^2D)$. Next, we compute the covariance matrices $\mathbf{\Sigma}_{\mathcal{S}}$ and $\mathbf{\Sigma}_{\mathcal{D}}$ in Eqs. (3.10) and (3.11) with a time complexity of $O(knD^2)$. The generalized eigenvalue decomposition for $\mathbf{\Sigma}_{\mathcal{S}}^{-1}\mathbf{\Sigma}_{\mathcal{D}}$ can be performed in $O(D^3)$. Summarizing, the overall time complexity for DMLMJ is $O(kn^2D + knD^2 + D^3)$.

Analogously, we analyze the computational complexity of KDMLMJ. We first compute the kernel matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$, where $K_{ij} = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$, with a time complexity of $O(n^2D)$. Then the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ in the feature space $\mathcal{F}$ can be computed in $O(1)$ as:

$$\begin{aligned} d(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)) &= \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\| \\ &= \sqrt{\langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_i)\rangle - 2\langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle + \langle\phi(\mathbf{x}_j), \phi(\mathbf{x}_j)\rangle} \\ &= \sqrt{K_{ii} - 2K_{ij} + K_{jj}}\,. \end{aligned}$$

Therefore, we can find the positive and negative neighborhoods for each training example with a time complexity of $O(kn)$. The computation of the matrices $\mathbf{U}$ and $\mathbf{V}$ in Eqs. (3.13) and (3.14) can be performed in $O(kn^3)$. The time complexity of the generalized eigenvalue decomposition for $\mathbf{U}^{-1}\mathbf{V}$ is $O(n^3)$. Summarizing, the overall time complexity for KDMLMJ is $O(n^2D + kn^3)$.

3.4. Related work

In order to take into account the positive semidefiniteness constraint, distance metric learning methods are mostly formulated as convex semidefinite programs. However, standard semidefinite programming solvers (Boyd and Vandenberghe, 2004) do not scale well when the number of examples or the dimensionality is high, due to the expensive computational cost in each iteration. A number of methods have been proposed to reduce this heavy computational burden. Weinberger and Saul (2009) suggested an efficient solver based on the projected subgradient descent method, but it requires an eigen-decomposition to preserve the positive semidefiniteness of the solution after each iteration. To avoid this eigen-decomposition, Ying and Li (2012) used the Frank-Wolfe method, which only requires the largest eigenvalue and corresponding eigenvector, to learn the distance metric. A similar solution requiring only the computation of the largest eigenvalue and corresponding eigenvector, based on a boosting-like method, was presented by Shen et al. (2012) to learn a linear positive combination of rank-one matrices. An alternative method proposed by Davis et al. (2007) was based on the iterative Bregman projection,

where no eigen-decomposition is required. Shi et al. (2014) formulated distance metric learning as learning a sparse combination of positive semidefinite rank-one matrices.

Another paradigm aims to learn a distance metric through learning a linear transformation. Since the positive semidefiniteness constraint in this case is not required, the optimization problem can be efficiently solved by a first-order method, such as gradient descent. However, the problem may be no longer convex, thus suffering from spurious local minima. Consequently, the solution will depend on the initialization point. Some popular methods include neighborhood component analysis (Goldberger et al., 2005), large margin component analysis (Torresani and Lee, 2007), and multi-task low-rank metric learning (Yang et al., 2011).

Unfortunately, the above methods typically require a very large number of iterations for large-scale problems. In practice, it is impossible to satisfy all constraints through online learning or stochastic optimization techniques. That is why learning a Mahalanobis distance metric for large data sets becomes a tremendous challenge as learning is limited by computational resources.

On the other hand, eigenvalue methods have been widely used to learn linear transformations since they only need to compute an eigen-decomposition. The most popular methods include principal component analysis (Jolliffe, 2005) and Fisher's linear discriminant analysis (Fisher, 1936). These methods can also be used in a nonlinear input space by applying the kernel trick (Schölkopf et al., 1998; Mika et al., 1999). Bar-Hillel et al. (2005) proposed a simple and efficient method, called relevant component analysis (RCA), for semi-supervised applications. RCA computes the Mahalanobis distance metric as a whitening transformation of the within chunklet covariance matrix, which is built from the pairwise similarity constraints. Yeung and Chang (2006) extended RCA by incorporating both similarity and dissimilarity constraints. Hoi et al. (2006) proposed discriminative component analysis (DCA), where the objective function is based on the ratio of determinants between the within and the discriminative chunklet covariance matrices. A method similar to DCA was introduced in (Xiang et al., 2008) by using the ratio of traces between the covariance matrices as objective function, requiring an iterative method to find the linear transformation. Despite our method being also an eigenvalue method, it differs significantly from previous methods in the way of finding the covariance matrices as well as in the objective function. By considering the local constraints derived from the neighborhood of each training example, our method can significantly improve the performance of k-NN, as will be shown next.

3.5. Experiments

In this section, we describe some experiments to evaluate the effectiveness of distance metric learning methods. We compare the proposed methods to the baseline Euclidean distance metric and four state-of-the-art distance metric learning methods, including ITML (Davis et al., 2007), LMNN (Weinberger and Saul, 2009), DML-eig (Ying and Li, 2012) and SCML (Shi et al., 2014). First, we use 27 data sets of different sizes to evaluate the linear distance metric learning methods. Second, we conduct an experiment to evaluate the capability of our method to perform dimensionality reduction. Finally, we use two synthetic highly nonlinear data sets to evaluate the kernelized version of DMLMJ.

3.5.1. Experimental settings

In order to make fair comparisons, we use the following configurations throughout this section. All experiments are empirically tested in the context of 5-NN and they are carried out on a PC with 4 Intel Core i5-3570 CPUs (3.40 GHz) and 8GB RAM. We use the source codes implemented in Matlab of ITML[3], LMNN[4], DML-eig[5] and SCML[6] supplied by the authors, and tune their parameters to get the best results. The source codes of DMLMJ and KDMLMJ are available online[7]. For DMLMJ and KDMLMJ, the k-positive neighborhood and k-negative neighborhood are all based on $k = 5$ (see 3.5.4 for a more detailed analysis of the influence of the number of neighbors). The regularization parameter is set to $\alpha = 0.001$. The target dimensionality m for DMLMJ is tuned using cross-validation.

The first general trend is that the classification accuracy of k-NN using the Euclidean distance metric is significantly improved when using the Mahalanobis distance metric learned by DMLMJ. In general, DMLMJ performs competitively compared with other state-of-the-art methods (Subsection 3.5.2).

The second general trend is that DMLMJ can perform distance metric learning and dimensionality reduction simultaneously. It outperforms other distance metric learning methods using principal component analysis (PCA) (Jolliffe, 2005) to reduce the dimensionality. Moreover, it is an order of magnitude faster than the competing methods (Subsection 3.5.3).

The third general trend is that KDMLMJ can perform well on highly nonlinear data sets, whereas a simple linear transformation cannot improve the performance of k-NN (Subsection 3.5.5).

[3] http://www.cs.utexas.edu/~pjain/itml/download/itml-1.2.tar.gz
[4] http://www.cse.wustl.edu/~kilian/code/files/mLMNN2.3.zip
[5] http://secamlocal.ex.ac.uk/people/staff/yy267/dml-eig-copy.zip
[6] http://mloss.org/media/code_archive/SCMLv1.11.zip
[7] http://users.ugent.be/~bacnguye/DMLMJ.zip

3.5.2. Linear distance metric learning

We compare the linear distance metric learning methods on 27 data sets from the Knowledge Extraction based on Evolutionary Learning (KEEL) (Triguero et al., 2017) machine learning repository[8]. The information of these data sets is summarized in Table A.1. The classification accuracies are obtained by averaging over five runs of 10-fold cross-validation. All divisions of the data sets are randomly split by the KEEL evaluation package. The features of these data sets are normalized into the interval $[0, 1]$.

Table 3.1 shows the average classification accuracies obtained by the competing methods. On each data set, we rank the methods based on their classification accuracy. We assign rank 1 to the method obtaining the highest accuracy, and rank 2 to the method obtaining the second higher accuracy, and so on. The average ranks of the competing methods are listed in the last row of Table 3.1. To detect whether there are significant differences among the results, we follow the recommendations by Demšar (Demšar, 2006) for statistical comparisons of classifiers over multiple data sets.

Firstly, we employ the Friedman test (Friedman, 1940) at a confidence level of $\alpha = 0.05$ to test the null hypothesis that all the distance metric learning methods obtain the same results on average. The p-value for the Friedman test is 0.01274. Since the p-value is less than the confidence level α, we reject the null hypothesis. Therefore, we apply the Bonferroni-Dunn test (Dunn, 1961) to detect which distance metric learning method performs equivalently or significantly different from the best-ranked method (i.e., DMLMJ, which obtained the lowest rank). The Bonferroni-Dunn test can identify significant differences between the control method (in our case, the best-ranked method) and other methods by computing a critical difference. Two distance metric learning methods are significantly different in performance if their corresponding average ranks differ by at least the critical difference:

$$\mathrm{CD} = q_\alpha \times \sqrt{\frac{n_c(n_c+1)}{6n_t}} = 2.576 \times \sqrt{\frac{6 \times (6+1)}{6 \times 27}} = 1.3116\,,$$

where n_c and n_t are the number of competing methods and the number of data sets, respectively, and q_α is the critical value (Sheskin, 2007). Figure 3.4 graphically represents the significant differences among the performances of the different distance metric learning methods. Any distance metric learning method with rank outside this marked area is significantly different from the control method (i.e., DMLMJ).

Additionally, we also apply Holm's step-down procedure (Holm, 1979) to compare the best-ranked method with the remaining methods. Table 3.2 presents

[8] http://sci2s.ugr.es/keel/datasets.php

Table 3.1: Classification accuracies on the KEEL data sets.

Data set	Euclidean	ITML	LMNN	DML-eig	SCML	DMLMJ
APP	85.00	86.00	**88.82**	87.00	86.91	87.91
BAL	86.24	91.84	84.64	87.52	**94.25**	92.63
BAN	89.28	89.34	89.34	89.17	**89.36**	89.26
BUP	64.28	62.05	61.90	62.47	65.05	**65.69**
ION	85.17	87.17	**89.75**	84.90	86.33	**89.75**
IRI	95.33	94.67	96.00	96.67	**97.33**	95.33
LED	**70.40**	69.80	69.80	69.40	65.00	67.80
LET	95.55	95.37	96.72	84.42	96.54	**97.50**
MAG	83.60	83.73	83.74	83.15	**84.79**	84.30
MON	94.75	89.43	97.04	**100.00**	99.55	99.55
MOV	75.28	74.72	**82.50**	67.22	63.33	81.94
OPT	98.75	98.70	**99.04**	97.44	97.21	99.00
PAG	95.78	96.03	96.24	95.29	**96.56**	95.78
PHO	87.75	87.75	87.43	**87.84**	87.49	87.79
PIM	73.32	72.93	73.19	73.06	72.92	**73.84**
RIN	69.12	81.54	69.22	84.31	80.12	**87.28**
SAT	90.78	90.71	91.28	89.54	89.08	**91.79**
SEG	95.41	96.36	96.23	**96.84**	95.97	95.84
SON	84.52	81.69	84.05	**85.05**	80.19	**85.05**
SPA	87.77	87.91	**90.08**	89.82	88.04	89.39
TEX	98.49	99.29	**99.89**	98.98	99.58	99.51
TWO	96.99	97.08	96.97	**97.54**	97.09	97.28
VEH	71.75	73.77	77.89	72.81	75.89	**80.97**
VOW	94.85	91.82	95.35	94.65	94.04	**95.45**
WDB	97.01	96.83	96.30	**97.36**	96.48	95.95
WIN	95.49	96.67	97.78	96.63	**98.86**	98.33
WIS	97.09	96.80	**97.10**	96.96	96.67	96.51
Rank	4.167	4.056	2.944	3.648	3.574	**2.611**

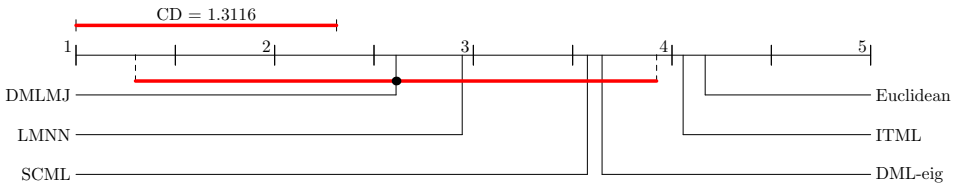


Figure 3.4: Comparison of the control method against the others with the Bonferroni-Dumn test. All methods with ranks outside the marked interval are significantly different from the control.

the z-value, p-value, and adjusted α for the Holm test at a confidence level of $\alpha = 0.5$. According to Table 3.2, the Holm test rejects hypotheses 4 and 5 since the corresponding p-value is less than the adjusted α. But hypotheses 1 to 3 cannot be rejected.

The statistical results allow us to draw the following conclusions. First, DMLMJ significantly outperforms ITML, but it only shows a slightly better behavior compared to LMNN, DML-eig and SCML in the context of k-NN. Second, the Mahalanobis distance metric learned by DMLMJ consistently outperforms the Euclidean distance metric.

Table 3.2: Holm post-hoc test for the competing methods with $\alpha = 0.05$.

i	Method	z-value	p-value	Holm's adjusted α	Hypothesis
5	Euclidean	3.0551	0.0023	0.0100	Rejected
4	ITML	2.8368	0.0046	0.0125	Rejected
3	DML-eig	2.0367	0.0417	0.0167	Accepted
2	SCML	1.8912	0.0586	0.0250	Accepted
1	LMNN	0.6547	0.5127	0.0500	Accepted

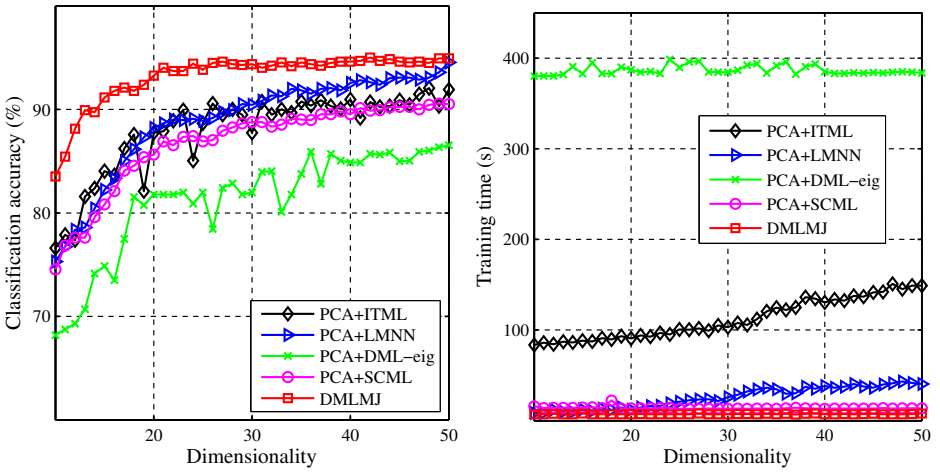
3.5.3. Dimensionality reduction

We compare the performance of k-NN using DMLMJ against other distance metric learning methods using PCA as a preprocessing step to reduce the dimensionality. The main purpose is to highlight the fact that our distance learning method with supervised information obtains better results when the dimensionality is reduced. Our experiment is based on the Isolet (Isolated Letter Speech Recognition) data set (Cole and Fanty, 1990), which consists of 6238 training examples, 1559 test examples with 617 features and 26 classes corresponding to 26 spoken letters. The Isolet data set has been used in various distance metric learning studies such as (McFee and Lanckriet, 2010; Parameswaran and Weinberger, 2010). More details about the features can be found in (Cole and Fanty, 1990). Training and test sets were predefined[9]. All features are continuous, real values, and scaled into the range $[-1, 1]$.

Figure 3.5(a) illustrates the classification accuracy of k-NN based on different distance metric learning methods with a varying number of features. We observe that DMLMJ performs better than other methods on this data set. When the dimensionality is small, all methods perform poorly as a consequence of the loss of information from the original feature space, however, DMLMJ is still much more effective. PCA discards the valuable class label information contained in the

[9] Available at https://archive.ics.uci.edu/ml/datasets/ISOLET

training set, and the projection made by PCA may intertwine the useful features and noisy features, thus leading to the poor performance of methods based on PCA.



(a) Accuracy **(b)** Training time

Figure 3.5: Experimental results on the Isolet data set. (a) Classification accuracy vs. dimensionality, (b) Training time vs. dimensionality

Figure 3.5(b) illustrates the training time (in seconds) of these five methods. Clearly, our method is an order of magnitude faster than the other methods.

3.5.4. Influence of the choice of the difference spaces

In this subsection, we study the influence of the choice of the difference spaces on the performance of DMLMJ. Since the difference spaces are built from the nearest neighbors of each training example, it is interesting to compare the performance in experiments using different neighborhood sizes. Let k_1, k_2 denote the number of neighbors for constructing the positive and negative difference spaces, respectively. Figure 3.6 shows the accuracy of 5-NN classification on the *balance* data set with different numbers of neighbors k_1 and k_2. From the figure, we can see that when $k_1 \gg k_2$ the classification accuracy is very low. This can be explained by the fact that the positive neighborhoods are more likely to undergo divergence than the negative neighborhoods, which implies that DMLMJ will extract the features that maximize the variance between examples of the same class and minimize the variance between examples of different classes. Consequently, the performance of k-NN cannot be improved. On the other hand, if more examples of different classes are considered to build the negative difference space, the classification accuracy is significantly increased. When k_1 and k_2 approach 100, the difference spaces tend

to use the information from the whole data set instead of using only information contained in the neighborhoods. In this case, DMLMJ performs similarly to other global distance metric learning methods, such as distance metric learning for clustering (Xing et al., 2002) and ITML (Davis et al., 2007). The performance is relatively stable when k_1 and k_2 are small enough to find the local discriminative information from the neighborhoods.

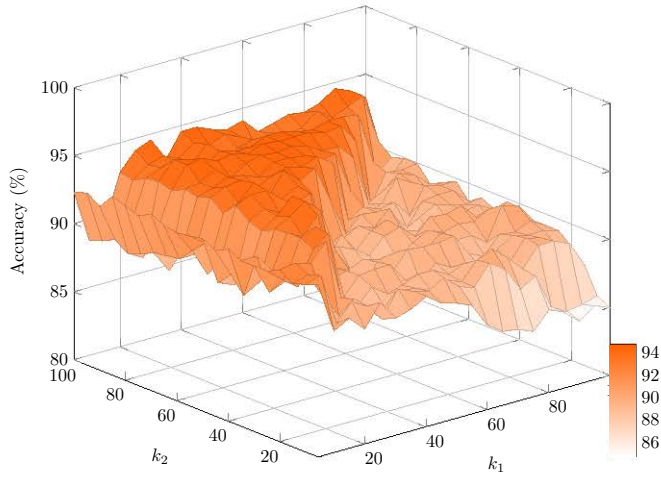


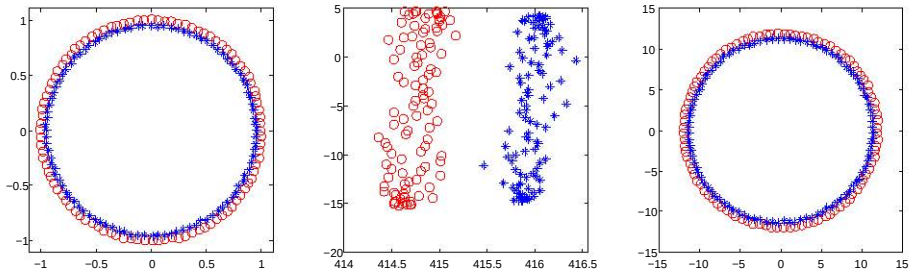
Figure 3.6: Experimental results on the balance data set. Classification accuracy of the 5-NN classifier versus the number of neighbors used for constructing the difference spaces, where k_1 denotes the number of neighbors used in the positive difference space and k_2 denotes the number of neighbors used in the negative difference space.

3.5.5. Nonlinear distance metric learning

To illustrate the potential of KDMLMJ, we conduct experiments on two synthetic two-dimensional data sets shown in Figs. 3.7(a) and 3.8(a). The first one consists of 200 examples drawn from two concentric circles. The second one consists of 500 examples drawn from two banana-shaped distributions. All examples belonging to the same class are shown in the same style and color. Similar experiments on these data sets were discussed by Weinberger and Saul (2009), and by Baghshah and Shouraki (2010b). According to the nonlinear structure in these data sets, a linear transformation may not suffice to improve the classification accuracy of k-NN.

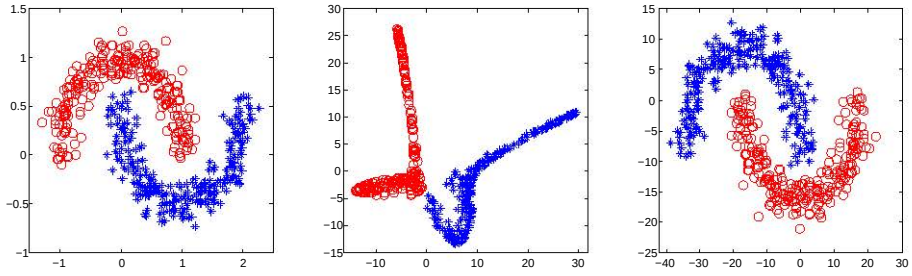
In this experiment, the RBF kernel, $\ker(\mathbf{u}, \mathbf{v}) = \exp(-\|\mathbf{u}-\mathbf{v}\|^2/\sigma)$, where $\sigma > 0$ is the kernel width, is adopted for the KDMLMJ method. The parameter σ is tuned by cross-validation on the training set considering as set of values $\{2^{-15}, \ldots, 2^3\}$. For a visual representation, the data sets are plotted in the transformed space using the nonlinear transformation learned by KDMLMJ (see Figs. 3.7(b) and 3.8(b))

and the linear transformation learned by DMLMJ (see Figs. 3.7(c) and 3.8(c)). Our illustration here is based on DMLMJ, but the same phenomenon occurs for the other linear distance metric learning methods. According to Figs. 3.7 and 3.8, KDMLMJ outperforms DMLMJ on both data sets since it is able to produce a highly nonlinear decision boundary through the use of the kernel function.



(a) Original space **(b)** Transformed space **(c)** Transformed space

Figure 3.7: Illustration of a synthetic data set drawn from two concentric circles: (a) original space, (b) transformed space learned by KDMLMJ using an RBF kernel, and (c) transformed space learned by DMLMJ.



(a) Original space **(b)** Transformed space **(c)** Transformed space

Figure 3.8: Illustration of a synthetic data set drawn from two banana-shaped distributions: (a) original space, (b) transformed space learned by KDMLMJ using an RBF kernel, and (c) transformed space learned by DMLMJ.

3.6. Conclusion

In this chapter, we have developed a novel linear transformation method for distance metric learning. We have shown that learning a linear transformation can be formulated as maximizing the Jeffrey divergence between two distributions derived

from local pairwise constraints. Then we have demonstrated that this problem is equivalent to solving a generalized eigenvalue decomposition problem with a closed-form solution. We have also developed the kernelized version of the proposed method to handle nonlinear data sets. The experimental results on the synthetic and real-world data sets demonstrate that the proposed method performs competitively compared with other state-of-the-art distance metric learning methods, while being an order of magnitude faster in training.

4 Kernel-based distance metric learning for person re-identification

Person re-identification is a fundamental task in many computer vision and image understanding systems. Due to appearance variations from different camera views, person re-identification still poses an important challenge. In the literature, KISSME has already been introduced as an effective distance metric learning method using pairwise constraints to improve the re-identification performance. Computationally, it only requires two inverse covariance matrix estimations. However, the linear transformation induced by KISSME is not powerful enough for more complex problems. We show that KISSME can be kernelized, resulting in a nonlinear transformation, which is suitable for many real-world applications. Moreover, the proposed kernel method can be used for learning distance metrics from structured objects without having a vectorial representation. The effectiveness of our method is validated on five publicly available data sets. To further apply the proposed kernel method efficiently when data are collected sequentially, we introduce a fast incremental version that learns a dissimilarity function in the feature space without estimating the inverse covariance matrices. The experiments show that the latter variant can obtain competitive results in a computationally efficient manner.

The material of this chapter is based on the following publication:
Nguyen, B. and De Baets, B. (2019). Kernel distance metric learning using pairwise constraints for person re-identification. *IEEE Transactions on Image Processing*, 28(2):589–600

4.1. Motivation

In recent years, the deployment of camera networks has grown exponentially in wide-area public spaces, such as railway stations, airports, and office buildings. As a result, many applications in person re-identification demand fast and effective techniques that are capable of accurately searching images from video surveillance (see e.g. Bedagkar-Gala and Shah, 2014, and the references therein). Given an image of a person, the main task in person re-identification is to identify the person from images taken at a different location and/or from a different viewpoint across non-overlapping cameras. It is important to remark that when a person disappears from one camera, he/she can be recognized from other cameras. A good system should be able to keep track of a person throughout the network, i.e. the appearances of the same person from different cameras have to be matched.

Person re-identification is a highly challenging problem, even for humans, due to the difficulty in characterizing the appearance and computing the similarity between images (Bedagkar-Gala and Shah, 2014; Paisitkriangkrai et al., 2015; Liao et al., 2015). These difficulties are mainly caused by changing view angles, resolution, lighting, occlusions, and so on. See Fig. 4.1 for an illustration of challenges in person re-identification.

Figure 4.1: An illustration of challenges in person re-identification (from left to right): different backgrounds, resolution, pose, view angle, lighting, partial occlusion, and similar clothings.

In order to find the correct match for a *probe* image from a set of *gallery* images captured by different cameras, two steps are employed. We first extract features from both probe and gallery images using a suitable feature extraction method. The identification results are then obtained by ranking the similarities between the probe and gallery images. Accordingly, the re-identification performance is measured by the top rank k matching rate, which is the percentage of probe images with correct matches found in the top-k ranked gallery images. That is why person re-identification can be formulated as a ranking problem (Prosser et al., 2010). Consequently, having an effective feature representation and a good distance metric can improve significantly the performance of re-identification (Liao et al., 2015).

Most of the existing studies focus on extracting more relevant or informative features that are able to discriminate different appearance patterns. A number of effective methods have been proposed to perform feature extraction for an image, including the scale invariant feature transform (SIFT) (Lowe, 2004), the ensemble of local features (ELF) (Gray and Tao, 2008), local binary patterns (LBP) (Ojala et al., 2002), Fisher vectors (LDFV) (Ma et al., 2012), and weighted histograms of

overlapping stripes (WHOS) (Lisanti et al., 2015). These handcrafted descriptors allow to significantly improve the performance of person re-identification. However, computing a set of representative and robust features is not always an easy task due to cross-view variations in appearance of images. Another interesting approach is to use a tensor representation rather than a vectorial representation for the input data (Tao et al., 2018, 2016a). Under several realistic viewing changes, most visual features and their combinations are neither stable nor reliable. In contrast to using complex handcrafted features computed from the raw images, deep convolutional neural networks (DCNNs) have been exploited to learn a set of representative features that captures the variability of person appearance across views (Ahmed et al., 2015; Xiao et al., 2016; Ding et al., 2015). One of the major problems with DCNNs is that they often require the availability of a huge number of images to obtain a model that is generalizable to data beyond the training set.

A recent trend tries to learn a good distance metric by implicitly suppressing those cross-view variations between images (Hirzer et al., 2012b; Köstinger et al., 2012). This is motivated by the fact that standard distance metrics, e.g. the Euclidean or Manhattan distance metric, are not reliable and flexible enough because they usually assume that all features are from the same domain with the same scale. Consequently, they become more sensitive to irrelevant features and fail to preserve the geometric characteristics of the data (Nguyen et al., 2017c). An ideal distance metric should accurately reflect the true underlying relationships between images, i.e. small distances for similar images and large distances for dissimilar or unrelated images. Previous studies (Yang et al., 2016; Paisitkriangkrai et al., 2015; Hirzer et al., 2012b; Köstinger et al., 2012; Zhao et al., 2017; Sun et al., 2017) have shown that optimizing a distance metric can significantly improve the performance of person re-identification.

The distance metric used may not fully reflect human judgments of dissimilarity without additional information from the users or from the training examples, such as class labels. One way to provide this information is through a set of constraints. As is common in person re-identification, we describe the information in the form of must-link and cannot-link pairwise constraints. Must-link constraints, e.g. images of the same person, are used to specify that the two examples should be in the same class. Cannot-link constraints, e.g. images of different persons, are used to specify that the two examples should be in different classes. These pairwise constraints have the following advantages that enable them to be applied in a wide range of application domains. First, collecting fully labeled training examples is a difficult task and also time-consuming. Particularly, annotating images with identity from every camera is prohibitively expensive in a large camera network. Second, it is often easier to collect pairwise relations, which are usually expressed in the form of pairwise constraints. The pairwise relations can be obtained, for instance, through interacting with the users by asking feedback whether two images are from the same person or not. Unlike the general procedure of asking feedback in the form of

annotating images with exact labels, the users are not required to have experience or prior knowledge with the data set.

Given a set of constraints, distance metric learning is mostly cast as solving a convex optimization problem over the cone of positive semidefinite matrices. While many efforts (Shen et al., 2012; Weinberger and Saul, 2009) have been devoted to reduce the computational complexity of semidefinite programming, they still require an expensive iterative optimization procedure. Based on a statistical inference perspective, Köstinger et al. (2012) introduced a pairwise distance metric learning approach named KISSME to avoid this computational burden. KISSME has the advantage of being simple and obtains a good recognition rate in person re-identification (Yang et al., 2014). One of the main problems is that KISSME may yield rather poor estimates of covariance matrices when the number of constraints is small, thus leading to a poor generalization ability. Several extensions (Tao et al., 2016b, 2015) have been proposed to address this problem, however, they are still limited to the use of a linear transformation and cannot capture the nonlinear structure of the input space. It is also important to note that KISSME can suffer from the curse of dimensionality in high-dimensional settings, just like other conventional distance metric learning methods that parameterize the distance metric by a matrix that scales quadratically with the dimensionality.

A common guiding principle for learning a distance metric from pairwise constraints is that the distances between examples in must-link constraints should be small, while the distances between those in cannot-link constraints should be large. Additionally, there are also several requirements for a good distance metric learning method: (1) it should reflect the true similarity relationships between examples in order to generalize well to unseen examples; (2) it should be easy to implement and to compute efficiently; (3) it should be flexible enough to handle different learning settings and data types. Based on these considerations, this chapter presents the following two main contributions:

(i) We propose the use of kernels for KISSME, named k-KISSME, which allows to capture the nonlinear structure in a data set. Our method operates in the kernel spaces, yielding a highly flexible distance metric. Compared to the original KISSME method, k-KISSME is not only more robust, but can also be used for naturally structured objects that have no vectorial representation.

(ii) Most of the kernel methods employ a "batch" setting, i.e. all examples need to be available during training. Unfortunately, in applications like video surveillance where images are collected sequentially, processing the whole data set upon the arrival of a new pairwise constraint can be computationally expensive. To alleviate this computational burden, we present an incremental update strategy for k-KISSME.

In the next section, we briefly review some of the most relevant works on person re-identification. In Section 4.3, we will revisit KISSME. Its kernel version, i.e.

k-KISSME, is presented in Section 4.4. Subsequently, we show that k-KISSME can be incrementally updated by relaxing the positive semidefiniteness constraint. Experiments on person re-identification benchmarks are conducted in Section 4.5, followed by some concluding remarks in Section 4.6.

4.2. Related work

In this section, we briefly review various relevant methods for learning an optimal distance metric in supervised settings that have been successfully applied to person re-identification tasks.

Typically, the supervision is induced in the form of pairwise constraints, i.e. must-link and cannot-link constraints. In the context of face identification, Guillaumin et al. (2009) introduced logistic discriminant metric learning (LDML), which aims to make the distances between examples of similar pairs smaller than the distances between those of dissimilar pairs. Based on pairwise constraints, Davis et al. (2007) formulated distance metric learning as a LogDet optimization problem, which can enforce the positive semidefiniteness constraint automatically to avoid the projection onto the positive semidefinite cone. Interestingly, Hirzer et al. (2012b) showed that relaxing the positive semidefiniteness constraint can dramatically simplify the problem of learning a Mahalanobis distance metric while still guaranteeing promising results. Recently, Sun et al. (2017) presented a person re-identification framework based on distance metric learning with latent variables. Yang et al. (2016) used only must-link constraints to learn an effective similarity function. The method most closely related to ours is the KISSME method proposed by Köstinger et al. (2012), which will be discussed in Section 4.3. To perform KISSME in high-dimensional settings, Liao et al. (2015) employed the generalized Rayleigh quotient to find a discriminant low-dimensional subspace in which to perform the KISSME method. Tao et al. (2017) showed that the performance of the latter can be further improved when using deep learning features in conjunction with handcrafted features. Another extension of KISSME was proposed by Tao et al. (2016b), including a smoothing technique to improve the estimation of the covariance matrices. Zhao et al. (2018) considered a QR decomposition that maps the data into a low-dimensional space and subsequently perform KISSME to learn a robust Mahalanobis matrix in the projected space.

Triplet constraints are another common form of supervision, i.e. object $\mathbf{x}_i$ is more similar to object $\mathbf{x}_j$ than to object $\mathbf{x}_l$. Weinberger and Saul (2009) introduced the large-margin nearest neighbor (LMNN) method that aims to pull target neighbors (of the same class) close together while pushing impostor neighbors (of different classes) far apart. LMNN performs well for k-nearest-neighbor (k-NN) classification. In order to handle the rejection case for k-NN, which is quite common in person re-identification tasks, Dikmen et al. (2011) proposed LMNN

with rejection (LMNN-R). Similarly, Zheng et al. (2013) proposed a probabilistic relative distance comparison (PRDC) method that maximizes the probability of a correct-match pair having a smaller distance than that of an incorrect-match pair.

Due to large variations in pose and illumination changes, it is unlikely that a linear transformation induced by the Mahalanobis distance metric can discriminate individuals satisfactorily. Instead of operating directly in the original input space, Xiong et al. (2014) introduced the use of kernels in order to learn a distance metric in the feature space. In doing so, we obtain a more flexible linear transformation in the feature space, which can be applied inductively to new examples. Although kernelized versions of various distance metric learning methods exist (Nguyen et al., 2017c; Davis et al., 2007; Jain et al., 2012), kernelizing a distance metric learning method is not always a trivial and straightforward task. In this chapter, we show how to kernelize KISSME, making it more efficient and robust to person re-identification tasks.

4.3. KISSME revisited

To motivate our approach, we briefly review KISSME as introduced in (Köstinger et al., 2012). Let us consider the difference $\mathbf{x}_i - \mathbf{x}_j$ between two examples $\mathbf{x}_i$ and $\mathbf{x}_j$. Consequently, two disjoint probability spaces of differences are defined, Ω_0 for differences of examples from different classes and Ω_1 for those from the same class. Let p_0 and p_1 denote the probability density functions of differences in Ω_0 and Ω_1, respectively. A possible way to verify whether or not $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same class is through the use of a log-likelihood ratio statistic:

$$\sigma(\mathbf{x}_i, \mathbf{x}_j) = \log\left(\frac{p_0(\mathbf{x}_i - \mathbf{x}_j)}{p_1(\mathbf{x}_i - \mathbf{x}_j)}\right). \tag{4.1}$$

A high value of $\sigma(\mathbf{x}_i, \mathbf{x}_j)$ indicates that $\mathbf{x}_i$ and $\mathbf{x}_j$ likely belong to different classes. In contrast, a low value of $\sigma(\mathbf{x}_i, \mathbf{x}_j)$ indicates that $\mathbf{x}_i$ and $\mathbf{x}_j$ likely belong to the same class. Assuming that the differences in Ω_0 and Ω_1 are normally distributed with zero mean, Eq. (4.1) can be rewritten as

$$\begin{aligned}\sigma(\mathbf{x}_i, \mathbf{x}_j) &= \log\left(\frac{\frac{1}{(2\pi)^{D/2}|\mathbf{\Sigma}_0|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{\Sigma}_0^{-1}(\mathbf{x}_i - \mathbf{x}_j)\right)}{\frac{1}{(2\pi)^{D/2}|\mathbf{\Sigma_1}|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{\Sigma}_1^{-1}(\mathbf{x}_i - \mathbf{x}_j)\right)}\right) \\ &= \frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\top(\mathbf{\Sigma}_1^{-1} - \mathbf{\Sigma}_0^{-1})(\mathbf{x}_i - \mathbf{x}_j) + \log\left(\frac{|\mathbf{\Sigma}_1|}{|\mathbf{\Sigma}_0|}\right),\end{aligned}$$

where $\mathbf{\Sigma}_0$ and $\mathbf{\Sigma}_1$ denote the covariance matrices of p_0 and p_1, respectively. Note that the zero mean assumption was also argued by Moghaddam et al. (2000) in a similar formulation as for each sample $\mathbf{x}_i - \mathbf{x}_j$ there always exists a sample $\mathbf{x}_j - \mathbf{x}_i$.

Since the constant terms do not affect the log-likelihood ratio statistic for use in statistical hypothesis testing, we can simplify it to

$$\sigma(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\top (\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\Sigma}_0^{-1})(\mathbf{x}_i - \mathbf{x}_j)\,.$$

Finally, learning a Mahalanobis distance metric amounts to estimating two inverse covariance matrices, i.e. $\mathbf{M} = \boldsymbol{\Sigma}_0^{-1} - \boldsymbol{\Sigma}_1^{-1}$, as σ and $d_{\mathbf{M}}$ share very similar properties. To guarantee that $d_{\mathbf{M}}$ is a distance metric, we use instead the projection of $(\boldsymbol{\Sigma}_0^{-1} - \boldsymbol{\Sigma}_1^{-1})$ onto the cone of PSD matrices. Using maximum likelihood estimation, the covariance matrices $\boldsymbol{\Sigma}_0$ and $\boldsymbol{\Sigma}_1$ are computed as follows

$$\boldsymbol{\Sigma}_0 = \frac{1}{n_0} \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top\,, \tag{4.2}$$

$$\boldsymbol{\Sigma}_1 = \frac{1}{n_1} \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top\,, \tag{4.3}$$

where $n_0 = |\mathcal{D}|$ and $n_1 = |\mathcal{S}|$.

As another alternative to the use of $\sigma(\mathbf{x}_i, \mathbf{x}_j)$, one may argue that a high value of $p_1(\mathbf{x}_i - \mathbf{x}_j)$ can indicate that $\mathbf{x}_i$ and $\mathbf{x}_j$ likely belong to the same class and a low value of $p_1(\mathbf{x}_i - \mathbf{x}_j)$ can indicate that $\mathbf{x}_i$ and $\mathbf{x}_j$ likely belong to different classes. Accordingly, the distance metric is only parameterized by the inverse of the covariance matrix $\boldsymbol{\Sigma}_1$, which is defined as the Mahalanobis distance between an example and a normal distribution. From this point of view, KISSME can be regarded as an extension of relevant component analysis (RCA) (Bar-Hillel et al., 2005), a simple method for learning distance metrics using only must-link constraints.

Although KISSME is very effective on low-dimensional data sets, it quickly becomes intractable when increasing the number of features. This is due to the fact that KISSME has a high memory complexity $O(D^2)$, which is prohibitive for many applications that involve thousands of features. Besides, computing the inverse covariance matrices is expensive and tends to be an ill-posed inverse problem as the covariance matrices are likely to be singular in higher dimensions. Next, we consider the idea of using kernels to overcome these limitations.

4.4. Kernel distance metric learning

In this section, we propose a nonlinear variant of KISSME. By introducing a regularizer into the covariance matrices, our method k-KISSME becomes more robust and stable. Moreover, to avoid recomputation of k-KISSME on the arrival of a new constraint, which is computationally expensive, an incremental version of k-KISSME is developed.

4.4.1. Kernel KISSME

The idea of kernel methods is to implicitly perform a nonlinear map ϕ from the input space $\mathcal{X}$ into a high-dimensional feature space $\mathcal{F}$, i.e. $\phi\colon \mathcal{X} \to \mathcal{F}$, by replacing the inner product with an appropriate positive semidefinite function. Formally, for any PSD kernel matrix $\mathbf{K}$, there exists a nonlinear map ϕ such that $K_{ij} = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$. The matrix $\mathbf{K}$ can be computed efficiently using a kernel function $\mathcal{K}$ that computes the inner product between two examples in the feature space without carrying out the explicit map, i.e. $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$. Several kernel functions, such as polynomials, radial basis functions, and exponential χ^2 kernel functions, have been successfully used in the context of distance metric learning (Nguyen et al., 2017c; Davis et al., 2007; Xiong et al., 2014). Motivated by the fact that kernel methods can overcome many limitations of their linear counterpart, in this subsection, we describe how to kernelize KISSME. Clearly, a direct computation of the inverse covariance matrices $\mathbf{\Sigma}_0^{-1}$ and $\mathbf{\Sigma}_1^{-1}$ is not feasible since the dimensionality of $\mathcal{F}$ is too high, or even infinite.

Assuming that the pairwise constraints in $\mathcal{S}$ and $\mathcal{D}$ are given, we start by introducing some notations. Let $\mathbf{1}_i$ be a column vector that has the value 1 at the i-th entry and 0 at the other entries. Let $\mathbf{B}_0$ (resp. $\mathbf{B}_1$) be an $n \times n$ diagonal matrix whose diagonal vector contains at the i-th entry the number of constraints in $\mathcal{D}$ (resp. $\mathcal{S}$) of which the first element is $\mathbf{x}_i$, i.e.

$$\begin{aligned}
\operatorname{diag}(\mathbf{B}_0)_i &= \left|\{j \mid j \in \{1, \dots, n\} \text{ and } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}\}\right|, \\
\operatorname{diag}(\mathbf{B}_1)_i &= \left|\{j \mid j \in \{1, \dots, n\} \text{ and } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}\}\right|.
\end{aligned}$$

Let $\mathbf{E}_0$ (resp. $\mathbf{E}_1$) be an $n \times n$ diagonal matrix whose diagonal vector contains at the j-th entry the number of constraints in $\mathcal{D}$ (resp. $\mathcal{S}$) of which the second element is $\mathbf{x}_j$, i.e.

$$\begin{aligned}
\operatorname{diag}(\mathbf{E}_0)_j &= \left|\{i \mid i \in \{1, \dots, n\} \text{ and } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}\}\right|, \\
\operatorname{diag}(\mathbf{E}_1)_j &= \left|\{i \mid i \in \{1, \dots, n\} \text{ and } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}\}\right|.
\end{aligned}$$

Let $\mathbf{W}_0$ (resp. $\mathbf{W}_1$) be an $n \times n$ matrix whose entry at the i-th row and j-th column is 1 if $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}$ (resp. $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}$), otherwise it takes value 0. Using the preceding notations, we can rewrite the matrix $\mathbf{\Sigma}_0$ in Eq. (4.2) as

$$\begin{aligned}
\mathbf{\Sigma}_0 &= \frac{1}{n_0} \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}} \left(\mathbf{x}_i \mathbf{x}_i^\top - \mathbf{x}_j \mathbf{x}_i^\top - \mathbf{x}_i \mathbf{x}_j^\top + \mathbf{x}_j \mathbf{x}_j^\top\right) \\
&= \frac{1}{n_0} \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}} \left(\mathbf{X}\mathbf{1}_i\mathbf{1}_i^\top\mathbf{X}^\top - \mathbf{X}\mathbf{1}_j\mathbf{1}_i^\top\mathbf{X}^\top - \mathbf{X}\mathbf{1}_i\mathbf{1}_j^\top\mathbf{X}^\top + \mathbf{X}\mathbf{1}_j\mathbf{1}_j^\top\mathbf{X}^\top\right) \\
&= \frac{1}{n_0} \mathbf{X} \left(\mathbf{B}_0 - \mathbf{W}_0^\top - \mathbf{W}_0 + \mathbf{E}_0\right) \mathbf{X}^\top
\end{aligned}$$

$$= \frac{1}{n_0}\mathbf{X}\mathbf{H}_0\mathbf{X}^\top,$$

where $\mathbf{H}_0 = \mathbf{B}_0 - \mathbf{W}_0^\top - \mathbf{W}_0 + \mathbf{E}_0$. Similarly, we can rewrite the covariance matrix $\mathbf{\Sigma}_1$ in Eq. (4.3) as

$$\mathbf{\Sigma}_1 = \frac{1}{n_1}\mathbf{X}\mathbf{H}_1\mathbf{X}^\top,$$

where $\mathbf{H}_1 = \mathbf{B}_1 - \mathbf{W}_1^\top - \mathbf{W}_1 + \mathbf{E}_1$. Note that $\mathbf{\Sigma}_0$ and $\mathbf{\Sigma}_1$ can be singular due to the lack of sufficient pairwise constraints. Therefore, to avoid the problem of inverting a singular matrix, we propose the use of a regularizing term by adding some small positive constant value ϵ to the diagonals of $\mathbf{\Sigma}_0$ and $\mathbf{\Sigma}_1$, i.e.

$$\widehat{\mathbf{\Sigma}}_0 = \epsilon\mathbf{I} + \frac{1}{n_0}\mathbf{X}\mathbf{H}_0\mathbf{X}^\top, \quad \widehat{\mathbf{\Sigma}}_1 = \epsilon\mathbf{I} + \frac{1}{n_1}\mathbf{X}\mathbf{H}_1\mathbf{X}^\top. \tag{4.4}$$

According to Friedman (1989), this method can obtain a more robust and stable estimation than using maximum likelihood estimation. To evaluate the inverses of these matrices, we consider the Kailath formula (Petersen and Pedersen, 2012) given by

$$(\mathbf{A}+\mathbf{B}\mathbf{D})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{I}+\mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{D}\mathbf{A}^{-1}. \tag{4.5}$$

Applying Eq. (4.5) to the covariance matrices in (4.4), results in

$$\widehat{\mathbf{\Sigma}}_0^{-1} = \frac{1}{\epsilon}\mathbf{I} - \frac{1}{n_0\epsilon^2}\mathbf{X}\mathbf{H}_0\left(\mathbf{I} + \frac{1}{n_0\epsilon}\mathbf{X}^\top\mathbf{X}\mathbf{H}_0\right)^{-1}\mathbf{X}^\top,$$
$$\widehat{\mathbf{\Sigma}}_1^{-1} = \frac{1}{\epsilon}\mathbf{I} - \frac{1}{n_1\epsilon^2}\mathbf{X}\mathbf{H}_1\left(\mathbf{I} + \frac{1}{n_1\epsilon}\mathbf{X}^\top\mathbf{X}\mathbf{H}_1\right)^{-1}\mathbf{X}^\top.$$

Finally, the difference between these two inverse covariance matrices can be computed as

$$\begin{aligned}
&\widehat{\mathbf{\Sigma}}_1^{-1} - \widehat{\mathbf{\Sigma}}_0^{-1} \\
&= \mathbf{X}\left[\frac{1}{n_0\epsilon^2}\mathbf{H}_0\left(\mathbf{I} + \frac{1}{n_0\epsilon}\mathbf{X}^\top\mathbf{X}\mathbf{H}_0\right)^{-1} - \frac{1}{n_1\epsilon^2}\mathbf{H}_1\left(\mathbf{I} + \frac{1}{n_1\epsilon}\mathbf{X}^\top\mathbf{X}\mathbf{H}_1\right)^{-1}\right]\mathbf{X}^\top \\
&= \mathbf{X}\left[\frac{1}{n_0\epsilon^2}\mathbf{H}_0\left(\mathbf{I} + \frac{1}{n_0\epsilon}\mathbf{K}\mathbf{H}_0\right)^{-1} - \frac{1}{n_1\epsilon^2}\mathbf{H}_1\left(\mathbf{I} + \frac{1}{n_1\epsilon}\mathbf{K}\mathbf{H}_1\right)^{-1}\right]\mathbf{X}^\top \\
&= \mathbf{X}\mathbf{C}\mathbf{X}^\top,
\end{aligned}$$

where $\mathbf{K} = \mathbf{X}^\top\mathbf{X}$ denotes the $n \times n$ kernel matrix and

$$\mathbf{C} = \frac{1}{n_0\epsilon^2}\mathbf{H}_0\left(\mathbf{I} + \frac{1}{n_0\epsilon}\mathbf{K}\mathbf{H}_0\right)^{-1} - \frac{1}{n_1\epsilon^2}\mathbf{H}_1\left(\mathbf{I} + \frac{1}{n_1\epsilon}\mathbf{K}\mathbf{H}_1\right)^{-1}.$$

It is easy to see that if $\widehat{\mathbf{\Sigma}}_1^{-1} - \widehat{\mathbf{\Sigma}}_0^{-1}$ is a PSD matrix, then the matrix $\mathbf{C}$ needs to be PSD as well. Hence, we use the projection of $\mathbf{C}$ onto the cone of PSD matrices, i.e. $\widehat{\mathbf{C}} = \Pi_{\mathbb{S}^+}(\mathbf{C})$, to compute the squared distance between two examples $\mathbf{x}_i$ and $\mathbf{x}_j$ as follows

$$\begin{aligned} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) &= (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{X} \widehat{\mathbf{C}} \mathbf{X}^\top (\mathbf{x}_i - \mathbf{x}_j) \\ &= (\mathbf{k}_{\mathbf{x}_i} - \mathbf{k}_{\mathbf{x}_j})^\top \widehat{\mathbf{C}} (\mathbf{k}_{\mathbf{x}_i} - \mathbf{k}_{\mathbf{x}_j}) \,, \end{aligned}$$

where $\mathbf{k}_{\mathbf{x}} = \mathbf{X}^\top \mathbf{x}$. Clearly, the computations above only involve the inner products between examples. Therefore, we can easily replace the inner product by a kernel function to perform distance metric learning in the feature space $\mathcal{F}$. The great advantage is that the linear KISSME method is extended to nonlinear scenarios in a straightforward way through the use of kernel tricks.

Another advantage of this kernelization is that it allows to apply KISSME on data sets containing structured objects on which kernel functions are defined. Since only a kernel function is required, many real-world data without an explicit vectorial representation (e.g., sequences, trees, and general graph-structured data) can be effectively dealt within our kernel-based framework. Several attempts have been made to design efficient kernel functions for such data. For instance, Leslie et al. (2002) adopted the spectrum kernel on sequences for protein sequences. Collins and Duffy (2002) showed how a kernel function can be applied to natural language structures. Gärtner et al. (2003), proposed kernels on labeled graphs with arbitrary structure. As the main focus of this chapter is on person re-identification, interested readers may refer to the survey by Gärtner (2003) for further details on defining kernel functions for structured data.

The overall computational complexity of k-KISSME mainly depends on the computation of the matrix $\mathbf{C}$. Due to the matrix multiplications and matrix inversions, this computation scales as $O(n^3)$. It is worth pointing out that k-KISSME has an advantage for problems where the number of features is significantly larger than the number of examples, i.e. $D \gg n$.

4.4.2. Incremental settings

In person re-identification, a learning method should be less sensitive to appearance changes, such as varying lighting conditions, clothing, poses, and so on. It is desirable to formulate a computationally tractable distance metric learning framework in an incremental setting to address such dynamic behavior. However, to keep the Mahalanobis matrix being PSD, we always need to employ an eigenvalue decomposition, which is computationally expensive if this procedure has to be carried out upon the arrival of every new pairwise constraint. Therefore, instead of learning a Mahalanobis distance metric, we relax the positive semidefiniteness

constraint and focus on learning a dissimilarity function. This relaxation strategy has already been adopted in various distance metric learning studies (Hirzer et al., 2012b; Chechik et al., 2010). We first define the dissimilarity function and then propose an efficient method for incrementally updating this dissimilarity function. As the new pairwise constraint can contain new examples, which are not observed in the training set, we also describe how to add these examples efficiently to the training set.

A dissimilarity function

In order to compute the dissimilarity of two examples in the feature space, it is necessary to redefine the covariance matrices. Since ϵ is a regularization constant, by redefining its value we can rewrite $\widehat{\mathbf{\Sigma}}_0$ and $\widehat{\mathbf{\Sigma}}_1$ in (4.4) as follows

$$\widehat{\mathbf{\Sigma}}_0 = \frac{1}{n_0}(\mathbf{X}\mathbf{H}_0\mathbf{X}^\top + \epsilon_0\mathbf{I})\,, \quad \widehat{\mathbf{\Sigma}}_1 = \frac{1}{n_1}(\mathbf{X}\mathbf{H}_1\mathbf{X}^\top + \epsilon_1\mathbf{I})\,,$$

where ϵ_0 and ϵ_1 are small positive constants. Applying Eq. (4.5) to compute the inverses of these covariance matrices, it yields

$$\begin{aligned}
\widehat{\mathbf{\Sigma}}_0^{-1} &= \frac{n_0}{\epsilon_0}\mathbf{I} - \frac{n_0}{\epsilon_0^2}\mathbf{X}\mathbf{H}_0\left(\mathbf{I} + \frac{1}{\epsilon_0}\mathbf{K}\mathbf{H}_0\right)^{-1}\mathbf{X}^\top\,,\\
\widehat{\mathbf{\Sigma}}_1^{-1} &= \frac{n_1}{\epsilon_1}\mathbf{I} - \frac{n_1}{\epsilon_1^2}\mathbf{X}\mathbf{H}_1\left(\mathbf{I} + \frac{1}{\epsilon_1}\mathbf{K}\mathbf{H}_1\right)^{-1}\mathbf{X}^\top\,.
\end{aligned}$$

Subsequently, the dissimilarity $\mathrm{dis}_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)$ of two examples $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as

$$\begin{aligned}
&\mathrm{dis}_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)\\
&= (\mathbf{x}_i - \mathbf{x}_j)^\top\left(\widehat{\mathbf{\Sigma}}_1^{-1} - \widehat{\mathbf{\Sigma}}_0^{-1}\right)(\mathbf{x}_i - \mathbf{x}_j)\\
&= (\mathbf{x}_i - \mathbf{x}_j)^\top\left[\frac{n_1}{\epsilon_1}\mathbf{I} - \frac{n_1}{\epsilon_1^2}\mathbf{X}\mathbf{H}_1\left(\mathbf{I} + \frac{1}{\epsilon_1}\mathbf{K}\mathbf{H}_1\right)^{-1}\mathbf{X}^\top\right.\\
&\qquad\left. -\frac{n_0}{\epsilon_0}\mathbf{I} + \frac{n_0}{\epsilon_0^2}\mathbf{X}\mathbf{H}_0\left(\mathbf{I} + \frac{1}{\epsilon_0}\mathbf{K}\mathbf{H}_0\right)^{-1}\mathbf{X}^\top\right](\mathbf{x}_i - \mathbf{x}_j)\\
&= \left(\frac{n_1}{\epsilon_1} - \frac{n_0}{\epsilon_0}\right)(\mathbf{x}_i^\top\mathbf{x}_i - 2\mathbf{x}_i^\top\mathbf{x}_j + \mathbf{x}_j^\top\mathbf{x}_j)\\
&\quad + (\mathbf{k}_{\mathbf{x}_i} - \mathbf{k}_{\mathbf{x}_j})^\top\left[\frac{n_0}{\epsilon_0^2}\mathbf{H}_0\left(\mathbf{I} + \frac{1}{\epsilon_0}\mathbf{K}\mathbf{H}_0\right)^{-1}\right.\\
&\qquad\left. -\frac{n_1}{\epsilon_1^2}\mathbf{H}_1\left(\mathbf{I} + \frac{1}{\epsilon_1}\mathbf{K}\mathbf{H}_1\right)^{-1}\right](\mathbf{k}_{\mathbf{x}_i} - \mathbf{k}_{\mathbf{x}_j})\,.
\end{aligned}$$

Note that $\text{dis}_{\mathbf{M}}$ only depends on the inner products and, therefore, it can be learned in the feature space by applying the kernel trick. It is clear that the matrix $\widehat{\mathbf{\Sigma}}_1^{-1} - \widehat{\mathbf{\Sigma}}_0^{-1}$ obtained is not always PSD, consequently, the dissimilarity function $\text{dis}_{\mathbf{M}}$ is not a pseudometric. However, our empirical experiments show that $\text{dis}_{\mathbf{M}}$ obtains competitive results compared to $d_{\mathbf{M}}$, while being significantly faster to compute. Next, we show how to perform an efficient update for $\text{dis}_{\mathbf{M}}$ upon the arrival of a new pairwise constraint.

Updating the dissimilarity function

Incremental learning usually arises in the case that images (examples) are sequentially collected, which is very common in a video surveillance system. An incremental learning system can be constructed, for instance, by adding additional cameras, or in a more general framework, by adding more knowledge from user interactions. It then follows that constraints are incrementally added using pairwise combinations of the new image and the images already in the training set. The following procedure only shows how the dissimilarity function is updated upon the arrival of a single constraint, but it is still possible to update efficiently given a set of constraints in a sequential manner. We consider the arrival of a new pairwise constraint $(\mathbf{x}_i, \mathbf{x}_j)$, which can be a must-link or a cannot-link constraint. Since $\text{dis}_{\mathbf{M}}$ mainly depends on the inverses of the two covariance matrices $\widehat{\mathbf{\Sigma}}_0$ and $\widehat{\mathbf{\Sigma}}_1$, we need to perform an update for these inverses. We will assume that $(\mathbf{x}_i, \mathbf{x}_j)$ is a cannot-link constraint and discuss how to update the inverse of $\widehat{\mathbf{\Sigma}}_0$. The case of a must-link constraint $(\mathbf{x}_i, \mathbf{x}_j)$ can be treated in a similar way.

Let us assume that $\mathbf{x}_i$ and $\mathbf{x}_j$ are examples in the training set, hence, the input matrix $\mathbf{X}$ and the kernel matrix $\mathbf{K}$ remain the same, while the inverse of $\widehat{\mathbf{\Sigma}}_0$ becomes

$$\widehat{\mathbf{\Sigma}}_{\text{new}}^{-1} = \frac{n_0+1}{\epsilon_0}\mathbf{I} - \frac{n_0+1}{\epsilon_0^2}\mathbf{X}\mathbf{H}_{\text{new}}\left(\mathbf{I} + \frac{1}{\epsilon_0}\mathbf{K}\mathbf{H}_{\text{new}}\right)^{-1}\mathbf{X}^\top, \tag{4.6}$$

where

$$\mathbf{H}_{\text{new}} = \mathbf{H}_0 + \mathbf{1}_i\mathbf{1}_i^\top - \mathbf{1}_i\mathbf{1}_j^\top - \mathbf{1}_j\mathbf{1}_i^\top + \mathbf{1}_j\mathbf{1}_j^\top . \tag{4.7}$$

One immediately observes that the inversion of $\widehat{\mathbf{\Sigma}}_{\text{new}}$ in Eq. (4.6) involves the computation of

$$\mathbf{T}_{\text{new}} = \mathbf{H}_{\text{new}}\left(\mathbf{I} + \frac{1}{\epsilon_0}\mathbf{K}\mathbf{H}_{\text{new}}\right)^{-1} .$$

In order to compute $\mathbf{T}_{\text{new}}$ efficiently, we will perform the update for $\mathbf{H}_0$ in four steps instead of one as in Eq. (4.7). In each step, we add only a rank-one matrix

to $\mathbf{H}_0$, while keeping track of the matrix $\mathbf{T}_{\text{new}}$. From Eq. (4.7), it is easy to see that $\mathbf{H}_0$ involves only four types of rank-one matrix update that are $\mathbf{1}_i\mathbf{1}_i^\top$, $-\mathbf{1}_i\mathbf{1}_j^\top$, $-\mathbf{1}_j\mathbf{1}_i^\top$, and $\mathbf{1}_j\mathbf{1}_j^\top$. By abuse of notation, we continue to write $\mathbf{H}_{\text{new}}$ to denote the matrix $\mathbf{H}_0$ after adding one of those rank-one matrices, i.e.

$$\mathbf{H}_{\text{new}} = \mathbf{H}_0 + \alpha \mathbf{1}_a \mathbf{1}_b^\top , \tag{4.8}$$

where $\alpha \in \{-1, 1\}$ and $a, b \in \{i, j\}$. At each step, we also keep track of the matrices

$$\begin{aligned}
\mathbf{Z}_0 &= \mathbf{I} + \frac{1}{\epsilon_0}\mathbf{K}\mathbf{H}_0 , \\
\mathbf{T}_0 &= \mathbf{H}_0 \left(\mathbf{I} + \frac{1}{\epsilon_0}\mathbf{K}\mathbf{H}_0 \right)^{-1} = \mathbf{H}_0 \mathbf{Z}_0^{-1} ,
\end{aligned}$$

and $\mathbf{Z}_0^{-1}$. The reason for doing so is to avoid extra computations by storing the previous computation results in each update. Next, we will show that

$$\mathbf{Z}_{\text{new}} = \mathbf{I} + \frac{1}{\epsilon_0}\mathbf{K}\mathbf{H}_{\text{new}} , \ \mathbf{T}_{\text{new}} = \mathbf{H}_{\text{new}}\mathbf{Z}_{\text{new}}^{-1}, \ \text{ and } \ \mathbf{Z}_{\text{new}}^{-1}$$

can be computed with a complexity of $O(n^2)$ instead of $O(n^3)$ as the naive method. After each step, we set $\mathbf{Z}_0 = \mathbf{Z}_{\text{new}}$, $\mathbf{T}_0 = \mathbf{T}_{\text{new}}$, $\mathbf{H}_0 = \mathbf{H}_{\text{new}}$, and $\mathbf{Z}_0^{-1} = \mathbf{Z}_{\text{new}}^{-1}$ to perform the next step.

We now explain how to perform the update in one step. Substituting Eq. (4.8) into $\mathbf{Z}_{\text{new}}$ gives

$$\begin{aligned}
\mathbf{Z}_{\text{new}} &= \mathbf{I} + \frac{1}{\epsilon_0}\mathbf{K}\mathbf{H}_{\text{new}} = \mathbf{I} + \frac{1}{\epsilon_0}\mathbf{K}(\mathbf{H}_0 + \alpha \mathbf{1}_a \mathbf{1}_b^\top) \\
&= \mathbf{I} + \frac{1}{\epsilon_0}\mathbf{K}\mathbf{H}_0 + \frac{\alpha}{\epsilon_0}\mathbf{K}_{\cdot a}\mathbf{1}_b^\top = \mathbf{Z}_0 + \frac{\alpha}{\epsilon_0}\mathbf{K}_{\cdot a}\mathbf{1}_b^\top .
\end{aligned}$$

The modification on $\mathbf{Z}_{\text{new}}$ involves only the computation of $\mathbf{K}_{\cdot a}\mathbf{1}_b^\top$, which scales as $O(n^2)$. In order to compute $\mathbf{Z}_{\text{new}}^{-1}$, we consider the Sherman-Morrison formula (Petersen and Pedersen, 2012) given by

$$(\mathbf{A} + \mathbf{c}\mathbf{d}^\top)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{c}\mathbf{d}^\top\mathbf{A}^{-1}}{1 + \mathbf{d}^\top\mathbf{A}^{-1}\mathbf{c}} .$$

Accordingly, it follows that

$$\mathbf{Z}_{\text{new}}^{-1} = \mathbf{Z}_0^{-1} - \frac{\alpha \mathbf{Z}_0^{-1}\mathbf{K}_{\cdot a}\mathbf{1}_b^\top\mathbf{Z}_0^{-1}}{\epsilon_0 + \mathbf{1}_b^\top\mathbf{Z}_0^{-1}\alpha\mathbf{K}_{\cdot a}} = \mathbf{Z}_0^{-1} - \mathbf{u}\mathbf{v}^\top ,$$

where $\mathbf{u} = \alpha\mathbf{Z}_0^{-1}\mathbf{K}_{\cdot a}/(\epsilon_0 + \mathbf{1}_b^\top\mathbf{Z}_0^{-1}\alpha\mathbf{K}_{\cdot a})$ and $\mathbf{v}^\top = \mathbf{1}_b^\top\mathbf{Z}_0^{-1}$. Note that both vectors $\mathbf{u}$ and $\mathbf{v}$ are computed in $O(n^2)$, therefore, the computation of $\mathbf{Z}_{\text{new}}^{-1}$ also scales as

$O(n^2)$. Consequently, $\mathbf{T}_{\text{new}}$ can be computed in $O(n^2)$ as

$$\begin{aligned}\mathbf{T}_{\text{new}} &= \mathbf{H}_{\text{new}}\mathbf{Z}_{\text{new}}^{-1} = (\mathbf{H}_0 + \alpha\mathbf{1}_a\mathbf{1}_b^\top)(\mathbf{Z}_0^{-1} - \mathbf{u}\mathbf{v}^\top) \\ &= \mathbf{T}_0 - (\mathbf{H}_0\mathbf{u})\mathbf{v}^\top + \alpha(1 - \mathbf{1}_b^\top\mathbf{u})\mathbf{1}_a\mathbf{v}^\top .\end{aligned}$$

So far, we have assumed that $\mathbf{x}_i$ and $\mathbf{x}_j$ are examples in the training set. Of course, upon the arrival of a new pairwise constraint that is formed by new examples, we should also add these new examples to the training set before performing the above updates. The task now is to keep track of the matrices $\mathbf{Z}_0$, $\mathbf{T}_0$ and $\mathbf{Z}_0^{-1}$ efficiently. Next, we will explain how to perform this task after adding a new example to the training set in $O(n^2)$.

Adding a new example

In the following, we will denote by $\mathbf{x}$ the newly arrived example. Without loss of generality, assuming that $\mathbf{x}$ will be added at the end of the training set, the input matrix $\mathbf{X}$ becomes $\mathbf{X}_{\text{new}} = \begin{pmatrix}\mathbf{X} & \mathbf{x}\end{pmatrix}$. It follows that the kernel matrix $\mathbf{K}$ and the matrix $\mathbf{H}_0$ are changed to

$$\begin{aligned}\mathbf{K}_{\text{new}} &= \begin{pmatrix}\mathbf{X}^\top \\ \mathbf{x}^\top\end{pmatrix}\begin{pmatrix}\mathbf{X} & \mathbf{x}\end{pmatrix} = \begin{pmatrix}\mathbf{K} & \mathbf{X}^\top\mathbf{x} \\ \mathbf{x}^\top\mathbf{X} & \mathbf{x}^\top\mathbf{x}\end{pmatrix}, \\ \mathbf{H}_{\text{new}} &= \begin{pmatrix}\mathbf{H}_0 & \mathbf{0} \\ \mathbf{0} & 0\end{pmatrix},\end{aligned}$$

where $\mathbf{0}$ is a zero matrix with the appropriate dimensions. Note that $\mathbf{K}_{\text{new}}$ and $\mathbf{H}_{\text{new}}$ are computed in $O(n^2)$. Consequently, we should update the matrices $\mathbf{Z}_0$, $\mathbf{T}_0$, and $\mathbf{Z}_0^{-1}$ to make the update procedure in the previous subsection feasible.

We start by computing $\mathbf{Z}_{\text{new}}$ as follows

$$\begin{aligned}\mathbf{Z}_{\text{new}} &= \mathbf{I} + \frac{1}{\epsilon_0}\mathbf{K}_{\text{new}}\mathbf{H}_{\text{new}} = \mathbf{I} + \frac{1}{\epsilon_0}\begin{pmatrix}\mathbf{K} & \mathbf{X}^\top\mathbf{x} \\ \mathbf{x}^\top\mathbf{X} & \mathbf{x}^\top\mathbf{x}\end{pmatrix}\begin{pmatrix}\mathbf{H}_0 & \mathbf{0} \\ \mathbf{0} & 0\end{pmatrix} \\ &= \mathbf{I} + \frac{1}{\epsilon_0}\begin{pmatrix}\mathbf{K}\mathbf{H}_0 & \mathbf{0} \\ \mathbf{x}^\top\mathbf{X}\mathbf{H}_0 & 0\end{pmatrix} = \begin{pmatrix}\mathbf{Z}_0 & \mathbf{0} \\ \frac{1}{\epsilon_0}\mathbf{x}^\top\mathbf{X}\mathbf{H}_0 & 1\end{pmatrix} .\end{aligned}$$

The modification of $\mathbf{Z}_{\text{new}}$ depends only on the computation of $\mathbf{x}^\top\mathbf{X}\mathbf{H}_0$, which scales as $O(n^2)$. Applying block matrix inversion (Petersen and Pedersen, 2012) given by

$$\begin{pmatrix}\mathbf{A} & \mathbf{B} \\ \mathbf{D} & \mathbf{E}\end{pmatrix}^{-1} = \begin{pmatrix}\mathbf{F}^{-1} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{G}^{-1} \\ -\mathbf{G}^{-1}\mathbf{D}\mathbf{A}^{-1} & \mathbf{G}^{-1}\end{pmatrix},$$

where $\mathbf{F} = \mathbf{A} - \mathbf{B}\mathbf{E}^{-1}\mathbf{D}$ and $\mathbf{G} = \mathbf{E} - \mathbf{D}\mathbf{A}^{-1}\mathbf{B}$, we can compute $\mathbf{Z}_{\text{new}}^{-1}$ as follows

$$\mathbf{Z}_{\text{new}}^{-1} = \begin{pmatrix} \mathbf{Z}_0 & \mathbf{0} \\ \frac{1}{\epsilon_0}\mathbf{x}^\top\mathbf{X}\mathbf{H}_0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{Z}_0^{-1} & \mathbf{0} \\ -\frac{1}{\epsilon_0}\mathbf{x}^\top\mathbf{X}\mathbf{H}_0\mathbf{Z}_0^{-1} & 1 \end{pmatrix}.$$

This computation also scales as $O(n^2)$. Finally, $\mathbf{T}_{\text{new}}$ can be decomposed as

$$\begin{aligned} \mathbf{T}_{\text{new}} = \mathbf{H}_{\text{new}}\mathbf{Z}_{\text{new}}^{-1} &= \begin{pmatrix} \mathbf{H}_0 & \mathbf{0} \\ \mathbf{0} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{Z}_0^{-1} & \mathbf{0} \\ -\frac{1}{\epsilon_0}\mathbf{x}^\top\mathbf{X}\mathbf{H}_0\mathbf{Z}_0^{-1} & 1 \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{H}_0\mathbf{Z}_0^{-1} & \mathbf{0} \\ \mathbf{0} & 0 \end{pmatrix} = \begin{pmatrix} \mathbf{T}_0 & \mathbf{0} \\ \mathbf{0} & 0 \end{pmatrix}. \end{aligned}$$

Finally, we set $\mathbf{Z}_0 = \mathbf{Z}_{\text{new}}$, $\mathbf{T}_0 = \mathbf{T}_{\text{new}}$, $\mathbf{H}_0 = \mathbf{H}_{\text{new}}$, $\mathbf{Z}_0^{-1} = \mathbf{Z}_{\text{new}}^{-1}$, $\mathbf{X} = \mathbf{X}_{\text{new}}$, and $\mathbf{K} = \mathbf{K}_{\text{new}}$ to perform the next step.

Pseudocode

To summarize the whole procedure of incorporating a new pairwise constraint $(\mathbf{x}_i, \mathbf{x}_j)$, a pseudocode is given in Algorithm 3. We use $t \in \{0, 1\}$ to denote the type of the constraint $(\mathbf{x}_i, \mathbf{x}_j)$, i.e. $t = 0$ for a cannot-link constraint and $t = 1$ for a must-link constraint.

Algorithm 3 Incremental update for k-KISSME

Input: A pairwise constraint $(\mathbf{x}_i, \mathbf{x}_j)$ of type $t \in \{0, 1\}$;
Output: The updated matrices $\mathbf{Z}_t$, $\mathbf{T}_t$, $\mathbf{Z}_t^{-1}$, $\mathbf{H}_t$, $\mathbf{X}$ and $\mathbf{K}$;

```
for x ← {x_i, x_j} do
    if x ∉ X then                                   ▷ adding a new example
        Insert x into X, then update K and H;
        Update Z_0, T_0, and Z_0^{-1} as in Subsection 4.4.2;
        Update Z_1, T_1, and Z_1^{-1} as in Subsection 4.4.2;
    end if
end for
for (α, a, b) ← {(1, i, i), (1, j, j), (−1, i, j), (−1, j, i)} do
    Update Z_t, T_t, and Z_t^{-1} as in Subsection 4.4.2;
    Update H ← H + α 1_a 1_b^⊤ ;
end for
```

4.5. Experiments

In this section, we evaluate the performance of our method on the task of identifying people for five publicly available data sets from real-world surveillance video. First, we describe the experimental settings. Then, we provide experimental results along with discussions.

4.5.1. Experimental settings

Competing distance metric learning methods

We have implemented **k-KISSME**[1] in Matlab in order to compare its performance with other distance metric learning methods, including the information-theoretic metric learning (**ITML**) (Davis et al., 2007), the large-margin nearest neighbor (**LMNN**) (Weinberger and Saul, 2009), the original **KISSME** (Köstinger et al., 2012), and the cross-view quadratic discriminant analysis (**XQDA**) (Liao et al., 2015). For k-KISSME, we apply the χ^2 kernel (Vedaldi and Zisserman, 2012), given by

$$\mathcal{K}(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{D} \frac{2u_i v_i}{u_i + v_i} .$$

Following Liao et al. (2015), the regularization parameter $\epsilon = 0.001$ is chosen. The constraints are extracted by forming all pairwise combinations of the training examples. As the number of cannot-link constraints can be significantly larger than that of must-link constraints, we use random subsampling to set the number of cannot-link constraints to ten times the number of must-link constraints to prevent very unbalanced problems.

Evaluation protocol

We adopt a *single-shot* experimental setting as evaluation protocol. More specifically, we randomly select all images of p persons to form the test set and the rest to form the training set. Following the same experimental settings as used in (Liao et al., 2015; Köstinger et al., 2012; Xiong et al., 2014), we split each data set into two equal parts, one half for training and the other half for testing. Each test set contains a gallery set and a probe set. We randomly select an image for each person to form the gallery set and use the rest to form the probe set. In order to facilitate the comparison with previously published results, the average cumulative

[1] Source codes are available at
http://users.ugent.be/~bacnguye/k-KISSME.v1.0.zip

matching accuracies at rank 1, 5, 10 and 20 are reported over ten runs to evaluate the performance of a distance metric learning method.

Feature representation

We use the local maximal occurrence representation (LOMO) recently proposed by Liao et al. (2015) to employ feature extraction for all the distance metric learning methods. First, a multiscale Retinex transformation (Jobson et al., 1997) is applied for image processing, resulting in a good representation of color and lightness. Then, LOMO applies the scale-invariant local ternary pattern method (SILTP) (Liao et al., 2010) to avoid intensity scale changes. Specifically, it locally constructs two scales of SILTP histograms and one HSV histogram of pixel features in a sliding window of size 10×10 to address viewpoint variations while maintaining local characteristics of a person. Finally, LOMO applies a log transform to normalize both HSV and SILTP features to unit length and obtains a 26,960-dimensional descriptor for each image. Due to the very high dimensionality, we project the extracted features into a 100-dimensional subspace using principal component analysis (PCA). In addition, we also report the performance of k-KISSME and XQDA based on the raw LOMO features because both of them can operate in a high-dimensional input space without reducing the dimensionality. Empirically, we have found that the results based on the raw LOMO features and its PCA subspace can be significantly different.

4.5.2. Experiments with re-identification benchmark data sets

We conduct extensive experiments on five data sets, including **iLIDS** (Zheng et al., 2009), **CAVIAR4REID** (Cheng et al., 2011), **3DPeS** (Baltieri et al., 2011), **PRID450S** (Roth et al., 2014), and **CUHK01** (Li et al., 2013a), to validate the effectiveness of the proposed k-KISSME method. A brief description of these data sets is given in Table 4.1. These data sets are widely used and provide many challenges in person re-identification, such as pose, viewpoint, background, resolution, and so on. We report the experimental results in two groups: (1) the performance comparison between k-KISSME and other distance metric learning methods using the low-dimensional features, (2) the performance of k-KISSME compared to other state-of-the-art methods. For the first group, we report the cumulative matching rates of all the competing distance metric learning methods based on the same 100-dimensional features using PCA. Additionally, the explicit feature map $\phi(\mathbf{u}) = \hat{\mathbf{u}}$, where $\hat{u}_i = \text{sign}(u_i)\sqrt{|u_i|}$, which resembles the Hellinger kernel embedding (Vedaldi and Zisserman, 2012), is employed to turn KISSME into a baseline nonlinear method (PCA+Helli.+KISSME) in the feature space mapped by ϕ. For the second group, we report the performance of k-KISSME using the raw

Table 4.1: A brief description of the data sets used in our experiments.

Data set	# individuals	# images	p
iLIDS	119	476	60
CAVIAR4REID	72	1,220	36
3DPeS	192	1,011	95
PRID450S	450	900	225
CUHK01	971	3,884	486

LOMO features against previously published results. A more detailed description and evaluation for each data set are described next.

The iLIDS data set[2] contains 476 images of 119 pedestrians taken from two non-overlapping cameras at an airport. For each individual, the number of images varies from 2 to 8. All images are normalized to the same size of 128×48 pixels. Most of them contain several occlusions caused by luggage and people. We randomly choose images of 60 persons to form the test set, i.e. $p = 60$. The performances of k-KISSME and XQDA using the raw LOMO features against several state-of-the-art methods, including LATENT-re-id (Sun et al., 2017), PCCA (Mignon and Jurie, 2012), LFDA (Pedagadi et al., 2013), SVMML (Li et al., 2013b), rPCCA (Xiong et al., 2014), kLFDA (Xiong et al., 2014), MFA (Yan et al., 2007), and DCNNs (Ding et al., 2015), are reported in Table 4.2. As can be seen from the table, k-KISSME consistently outperforms the recent XQDA and other state-of-the-art methods. Even compared to the deep net proposed in (Ding et al., 2015), k-KISSME obtains a higher matching rate for rank 5, 10, and 20. Interestingly, k-KISSME achieves a significantly higher performance at rank 1 on PCA features.

The CAVIAR4REID data set[3] contains 1,220 images of 72 persons taken from two cameras at a shopping center in Lisbon. This data set is particularly designed with the aim of maximizing appearance variations in resolution changes, lighting conditions, and pose changes. There are 50 persons with both camera views and the remaining 22 persons with one camera view. The number of images for each individual varies from 10 to 20. Since the image sizes vary from 39×17 to 144×72 pixels, we normalize all images to the same size of 128×48 pixels in order to extract the same set of features as is done in (Xiong et al., 2014). Table 4.3 shows the cumulative matching accuracy with $p = 36$ for k-KISSME and XQDA using the raw LOMO features against several state-of-the-art methods, including PCCA (Mignon and Jurie, 2012), LFDA (Pedagadi et al., 2013), SVMML (Li et al., 2013b), rPCCA (Xiong et al., 2014), kLFDA (Xiong et al., 2014), MFA (Yan et al., 2007), and RMLLC (Chen et al., 2015). We can observe that k-KISSME obtains a competitive result compared to XQDA and outperforms other state-of-the-art

[2] https://www.gov.uk/guidance/imagery-library-for-intelligent-detection-systems
[3] http://www.lorisbazzani.info/caviar4reid.html

Table 4.2: The top matching rates (%) on the iLIDS data set. The best results are highlighted in boldface.

Method	Rank 1	Rank 5	Rank 10	Rank 20	Ref.
PCA+ITML	45.2	69.2	80.5	90.4	–
PCA+LMNN	43.5	66.6	77.8	88.1	–
PCA+KISSME	40.7	64.6	75.1	86.0	–
PCA+XQDA	42.2	65.7	77.2	89.2	–
PCA+Helli.+KISSME	40.8	64.7	75.5	86.2	–
PCA+k-KISSME	**48.3**	**70.3**	**80.9**	**90.7**	–
PCCA	23.0	51.1	67.0	83.3	(Xiong et al., 2014)
LFDA	32.2	56.0	68.7	81.6	(Xiong et al., 2014)
SVMML	20.8	49.1	65.4	81.7	(Xiong et al., 2014)
rPCCA	26.6	54.3	69.7	84.5	(Xiong et al., 2014)
kLFDA	36.5	64.1	76.5	88.5	(Xiong et al., 2014)
MFA	32.6	58.5	71.5	84.4	(Xiong et al., 2014)
DCNNs	**52.1**	68.2	78.0	88.8	(Ding et al., 2015)
LATENT-re-id	46.2	**70.2**	80.7	91.3	(Sun et al., 2017)
XQDA	43.5	69.9	81.8	93.3	–
k-KISSME	44.0	70.0	**82.6**	**93.4**	–

methods.

The 3DPeS data set[4] contains 1,011 images of 192 individuals captured from 8 different surveillance cameras. This data set is particularly designed for people tracking and person re-identification. The number of images for each individual varies from 2 to 26. Since the image sizes vary from 100×31 to 267×176 pixels, we normalize all images to the same size of 128×48 pixels. Table 4.4 reports the cumulative matching accuracy with $p = 95$ for k-KISSME and XQDA using the raw LOMO features against other state-of-the-art methods, including PCCA (Mignon and Jurie, 2012), LFDA (Pedagadi et al., 2013), SVMML (Li et al., 2013b), rPCCA (Xiong et al., 2014), kLFDA (Xiong et al., 2014), and MFA (Yan et al., 2007). The results show that k-KISSME achieves the best overall performance. In particular, it achieves a recognition rate of 48.7% at rank 1.

The PRID450S data set[5] contains 900 images from 450 single-shot image pairs captured by two different surveillance cameras. It is a very challenging data set due to different viewpoint changes, background interference, and partial occlusion. In our experiment, each image is normalized to 128×48 pixels. Since the PRID450S is a newly constructed data set, there are only a few results reported in the literature. We show the cumulative matching rate with $p = 225$ of k-KISSME and XQDA using

[4] http://imagelab.ing.unimore.it/visor/3dpes.asp
[5] https://www.tugraz.at/institute/icg/research/team-bischof/lrs/downloads/prid450s/

Table 4.3: The top matching rates (%) on the CAVIAR4REID data set. The best results are highlighted in boldface.

Method	Rank 1	Rank 5	Rank 10	Rank 20	Ref.
PCA+ITML	25.3	53.6	71.7	89.6	–
PCA+LMNN	38.2	59.4	71.7	86.4	–
PCA+KISSME	44.3	72.6	85.7	96.6	–
PCA+XQDA	41.8	71.0	84.8	96.2	–
PCA+Helli.+KISSME	44.9	72.8	85.9	96.6	–
PCA+k-KISSME	**46.0**	**74.6**	**86.8**	**96.9**	–
PCCA	29.1	62.5	79.7	94.2	(Xiong et al., 2014)
LFDA	31.7	56.1	70.4	86.9	(Xiong et al., 2014)
SVMML	25.8	61.4	78.6	93.6	(Xiong et al., 2014)
rPCCA	30.4	63.6	80.4	94.5	(Xiong et al., 2014)
kLFDA	36.2	64.0	78.7	92.2	(Xiong et al., 2014)
MFA	37.7	67.2	82.1	94.6	(Xiong et al., 2014)
RMLLC	41.2	**73.5**	85.0	94.4	(Chen et al., 2015)
XQDA	**42.3**	71.8	**86.0**	96.0	–
k-KISSME	41.9	71.5	85.5	**96.1**	–

Table 4.4: The top matching rates (%) on the 3DPeS data set. The best results are highlighted in boldface.

Method	Rank 1	Rank 5	Rank 10	Rank 20	Ref.
PCA+ITML	26.8	51.3	64.9	79.7	–
PCA+LMNN	39.5	62.3	74.6	85.4	–
PCA+KISSME	**45.7**	69.7	79.1	88.2	–
PCA+XQDA	44.4	69.7	**80.0**	**89.4**	–
PCA+Helli.+KISSME	45.3	68.9	78.1	87.9	–
PCA+k-KISSME	**45.7**	**69.8**	79.3	88.2	–
PCCA	36.4	66.3	78.1	88.6	(Xiong et al., 2014)
LFDA	39.1	61.7	71.8	82.6	(Xiong et al., 2014)
SVMML	27.7	58.5	72.1	84.1	(Xiong et al., 2014)
rPCCA	40.4	69.5	80.5	90.0	(Xiong et al., 2014)
kLFDA	48.4	72.5	82.1	89.9	(Xiong et al., 2014)
MFA	42.3	65.3	75.2	84.8	(Xiong et al., 2014)
XQDA	46.7	70.7	81.2	91.0	–
k-KISSME	**48.7**	**72.6**	**83.9**	**92.1**	–

the raw LOMO features compared to some state-of-the-art methods, including EIML (Hirzer et al., 2012a), SCNCN (Yang et al., 2014), ECM (Liu et al., 2015b),

Table 4.5: The top matching rates (%) on the PRID450S data set. The best results are highlighted in boldface.

Method	Rank 1	Rank 5	Rank 10	Rank 20	Ref.
PCA+ITML	30.6	60.4	73.2	85.3	–
PCA+LMNN	45.7	74.9	84.7	91.2	–
PCA+KISSME	41.6	71.3	81.1	89.4	–
PCA+XQDA	**48.7**	**77.7**	**86.1**	**93.2**	–
PCA+Helli.+KISSME	41.7	71.9	80.2	89.8	–
PCA+k-KISSME	47.4	76.4	85.0	91.9	–
EIML	35.0	–	68.0	77.0	(Yang et al., 2014)
SCNCD	41.6	68.9	79.4	87.8	(Yang et al., 2014)
ECM	41.9	66.3	76.9	84.2	(Liu et al., 2015b)
QRKISS	**57.1**	80.7	88.0	–	(Zhao et al., 2018)
XQDA	49.6	77.6	86.3	92.4	–
k-KISSME	53.9	**81.0**	**88.8**	**94.5**	–

and QRKISS (Zhao et al., 2018), in Table 4.5. Clearly, k-KISSME obtains the best performance on most of the reported ranks.

The CUHK01 data set[6] contains 3,884 images of 971 pedestrians captured from two disjoint cameras on a college campus. Each camera has taken two images of every individual. In our experiment, all images are downsized to a resolution of 128×48 pixels to reduce the computation time. We set $p = 486$ in order to facilitate the comparison with other methods. Table 4.6 shows the cumulative matching accuracy of k-KISSME using the raw LOMO features against some state-of-the-art methods, including kLFDA (Xiong et al., 2014), Ensembles (Paisitkriangkrai et al., 2015), IDLA (Ahmed et al., 2015), ImpTrpLoss (Cheng et al., 2016), and DeepRanking (Chen et al., 2016). Clearly, k-KISSME achieves the best matching performance among the competing distance metric learning methods on the PCA features. Despite its simplicity, k-KISSME obtains a better performance than deep learning methods at rank 1, while being less accurate at rank 5, 10, and 20. This result is not surprising since deep learning methods use advanced techniques such as data augmentation (Cheng et al., 2016) and additional training data (Chen et al., 2016) to improve the matching rate as well as to avoid overfitting.

[6] http://www.ee.cuhk.edu.hk/~xgwang/CUHK_identification.html

Table 4.6: The top matching rates (%) on the CUHK01 data set. The best results are highlighted in boldface.

Method	Rank 1	Rank 5	Rank 10	Rank 20	Ref.
PCA+ITML	22.6	40.6	50.4	61.5	–
PCA+LMNN	42.3	61.5	70.5	79.2	–
PCA+KISSME	52.8	73.6	81.2	87.1	–
PCA+XQDA	50.6	72.4	80.5	87.3	–
PCA+Helli.+KISSME	51.9	73.8	80.9	87.6	
PCA+k-KISSME	**54.6**	**75.4**	**82.5**	**88.6**	–
kLFDA	32.8	59.0	69.6	79.2	(Chen et al., 2016)
IDLA	47.5	71.5	80.0	–	(Ahmed et al., 2015)
Ensembles	53.4	76.3	84.4	90.5	(Paisitkriangkrai et al., 2015)
DeepRanking	50.4	75.9	84.1	91.3	(Chen et al., 2016)
ImpTrpLoss	53.7	**84.3**	**91.0**	**96.3**	(Cheng et al., 2016)
XQDA	53.5	75.8	83.9	89.9	–
k-KISSME	**54.3**	74.2	80.5	87.0	–

According to the overall results, we can see that the kernelized version of KISSME leads to significant improvements over the original KISSME. Our method k-KISSME yields the best performance on most data sets, demonstrating a great flexibility and accuracy for matching compared to other competing methods. It is also interesting to note that k-KISSME consistently obtains high rank r matching rates with small values of r. This provides important information for a person re-identification system because the top matched images are usually verified by a human operator (Gray et al., 2007). We also note that k-KISSME outperforms most of the linear methods on the PCA features. The reason for this may lie in the fact that the projection made by PCA may intertwine the useful features and the noisy features. Consequently, the data set may be transformed into a nonlinearly separable problem, thus making it difficult for linear methods. As commonly used in computer vision, an explicit feature map, such as the signed square root, can approximate nonlinear distance metric learning methods in the feature space by linear ones. Although this approach is scalable for large data sets, the improvement is still very limited (see the results of PCA+KISSME and PCA+Helli.+KISSME). In contrast, k-KISSME employs the kernel trick to find a good solution in the implicit feature space, making it more robust for complex tasks.

4.5.3. Running time

The average training times of the competing distance metric learning methods on the low-dimensional as well as the raw LOMO features are shown in Table 4.7. The running time is computed on a laptop with 4 Intel Core i5-5200U CPUs (2.20GHz) and 8GB RAM. Note that the results include the time for computing the kernel matrix. As can be seen from the table, XQDA is the least time consuming on all these data sets, followed by KISSME. It should be noticed that k-KISSME is significantly faster than other iterative methods such as ITML and LMNN on small-sized data sets. The slower speed on large-sized data sets of k-KISSME is a result of computing the kernel matrix. Further running time improvements can be anticipated by using advanced techniques to speed up the calculation of the kernel matrix. Although our method has mainly been implemented in MATLAB, a careful implementation can significantly improve the real computation time. More importantly, k-KISSME can perform efficiently on very high-dimensional data sets, which could be computationally challenging for those methods that directly learn a distance metric from the input space. It requires significantly less memory and a lower training time compared to the deep neural networks. Moreover, k-KISSME is very simple to implement, computationally efficient, and serves our main goal, which is to develop an efficient system for person re-identification.

Table 4.7: Average training time (in seconds) of the competing distance metric learning methods. The best results are highlighted in boldface.

Method	iLIDS	CAVIAR4REID	3DPeS	PRID450S	CUHK01
ITML	78.96	76.03	57.63	63.33	38.68
LMNN	61.23	209.54	231.41	267.51	2,908.68
KISSME	0.09	0.54	0.24	0.06	1.03
XQDA	**0.03**	**0.02**	**0.08**	**0.05**	**0.33**
k-KISSME	1.49	22.65	7.23	1.82	51.17

4.5.4. Experiments with dimensionality

In this subsection, we investigate how the performance of distance metric learning methods varies with different subspace dimensions. For this purpose, we report the matching rate at rank 1 for ITML, LMNN, KISSME, XQDA, and k-KISSME on the iLIDS data set with different dimensions extracted by PCA (see Fig. 4.2). We keep the same experimental settings for all the competing methods. As can be seen from this figure, k-KISSME consistently outperforms other methods over all the reported dimensions. We found that KISSME is very sensitive to the choice of the number of PCA dimensions, yielding a relatively high variance over different dimensions. This behavior was also noted by Xiong et al. (2014). Nevertheless, we observe that XQDA and k-KISSME tend to have a more stable performance over the different dimensions. The latter can be easily explained by the fact that both XQDA and k-KISSME add a small regularizer to the diagonal elements of the covariance matrices, making the estimation more smooth and robust, especially when the dimensionality is increased.

4.5.5. Experiments with incremental learning

We further verify the efficiency and effectiveness of using the incremental update procedure described in Subsection 4.4.2 for k-KISSME. As an illustration, we compare k-KISSME and the method that learns a dissimilarity function (denoted by **k-KISSME (inc)**) in terms of training time and rank 1 matching rate on the CAVIAR4REID and 3DPeS data sets (see Fig. 4.3). The same experimental settings are used. We keep on randomly adding a pairwise constraint on each update. As we pointed out in Subsection 4.4.2, ϵ_0 and ϵ_1 act as hyperparameters that can be used to adjust the regularizing terms of the covariance matrices. For the CAVIAR4REID and 3DPeS data sets, we set the values of ϵ_0 and ϵ_1 to 1 and 0.05, respectively, which yield the best results in most of our experiments. Clearly, these hyperparameters should be determined by the characteristics of the data sets as well as the pairwise constraints.

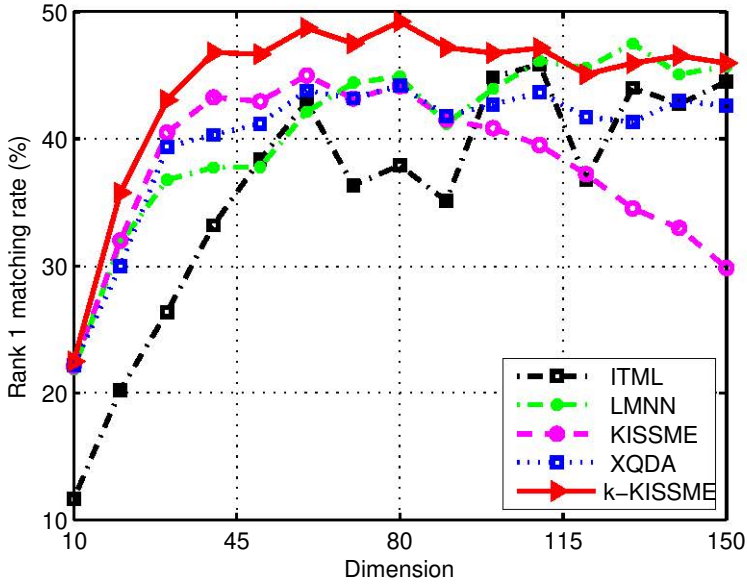
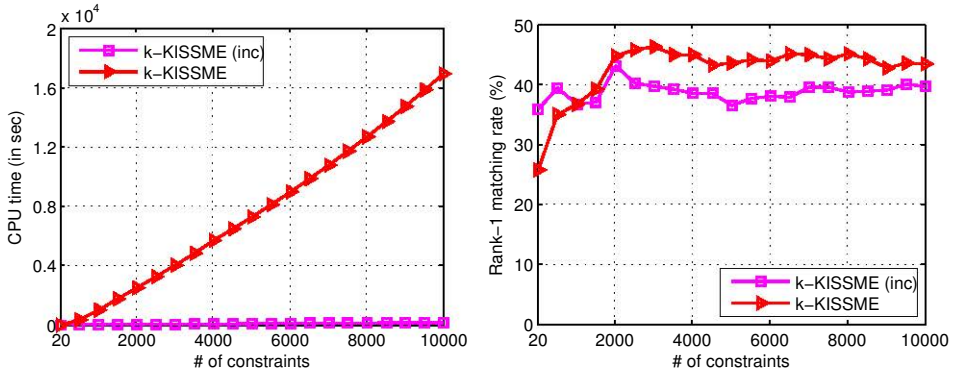


Figure 4.2: Illustration of rank 1 matching rate vs. number of dimensions on the iLIDS data set.

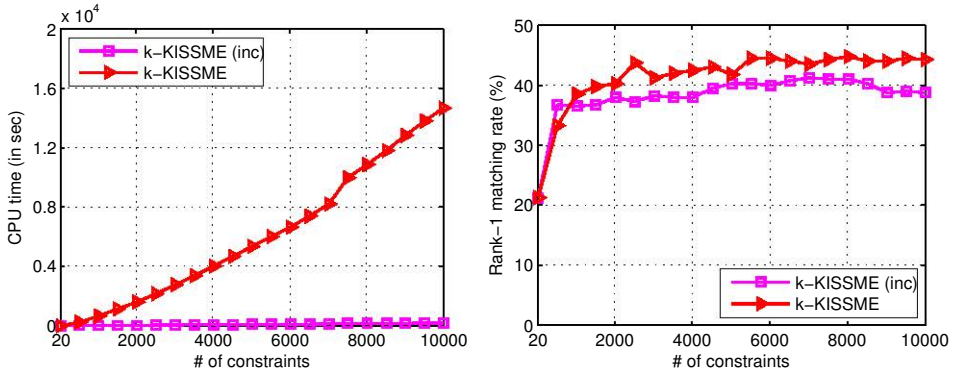
As expected, using the incremental update procedure yields a significant speedup, while obtaining a competitive performance. Like many online learning algorithms, fluctuations in performance are mainly due to the randomness of adding constraints. However, this incremental technique ensures that the similarity function is immediately trained and will become more accurate over time as more new points and pairwise constraints are added. The advantage of incremental k-KISSME is particularly apparent when there is an unbounded stream of possible constraints to learn from.

4.6. Conclusion

Person re-identification is a challenging problem in video surveillance due to the large variations in appearance by using different cameras. To deal with this challenge, we have proposed a distance metric learning method, named k-KISSME, by incorporating kernels into the KISSME method. This allows k-KISSME to operate in a nonlinear feature space induced by a kernel function. As a result, k-KISSME improves the recognition rate and could be applied in learning a distance metric from structural objects without having a vectorial representation. Moreover, we have also introduced a fast version for k-KISSME avoiding expensive recomputations in an incremental setting. Experiments on five real-world data sets have demonstrated the effectiveness of k-KISSME compared to other distance metric



(a) CAVIAR4REID



(b) 3DPeS

Figure 4.3: Illustration of the incremental update procedure on the CAVIAR4REID and 3DPeS data sets (left) training time (in seconds) vs. number of constraints, (right) rank 1 matching rate vs. number of constraints.

learning methods for person re-identification tasks.

5 Case study: Learning single-cell distances from cytometry data

Recent years have seen an increased interest in employing data analysis techniques for the automated identification of cell populations in the field of cytometry. These techniques highly depend on the use of a distance metric to measure the similarities between single cells. Without any additional knowledge, the Euclidean distance metric is commonly used, yielding a suboptimal solution. In this chapter, we exploit the availability of single-cell labels to find an optimal distance metric from the data. The usefulness of such a distance metric is discussed in various applications. We show that current distance-based methods can be improved by using an appropriate Mahalanobis distance metric. In particular, our approach is illustrated for cytometry data from two different origins, i.e. flow cytometry applied to microbial cells and mass cytometry for the analysis of human blood cells. Experiments indicate that the resulting distance metric can significantly improve the cell-population identification.

The material of this chapter is based on the following publication:
Nguyen, B., Rubbens, P., Kerckhof, F.-M., Boon, N., De Baets, B., and Waegeman, W. (2019b). Learning single-cell distances from cytometry data. *Cytometry Part A*, submitted

5.1. Motivation

Due to the fact that the amount of data and the number of dimensions (e.g. the size of multicolor panel designs or the introduction of mass cytometry) are increasing in the field of cytometry, automated data analysis techniques are becoming increasingly popular (O'Neill et al., 2013; Brinkman et al., 2016; Saeys et al., 2016; Rahim et al., 2018). These techniques include a number of preprocessing steps, such as specific transformations and quality controls of the data (Finak et al., 2010; Monaco et al., 2016). They are often followed by dimensionality reduction and clustering techniques to visualize the data or to determine cell populations (Ge and Sealfon, 2012; Amir et al., 2013; Van Gassen et al., 2015; Levine et al., 2015). The latter techniques usually depend on a predefined distance metric in order to measure the distance between single cells. In most cases, a simple choice is the Euclidean distance metric. Other distance metrics, such as the Mahalanobis distance metric, have been considered (Pyne et al., 2009; Aghaeepour et al., 2011; Pouyan et al., 2016), but are less popular.

In addition, the availability of single-cell annotation has opened the door to exploring the use of supervised or semi-supervised machine learning techniques (Rubbens et al., 2017a; Lux et al., 2018). In particular, distance metric learning exploits such available knowledge by learning a distance metric that preserves similarity relationships in the data. The goal is to learn a distance metric that results in small distances between examples of the same class and large distances between examples of different classes. Recent developments in distance metric learning have shown that using an appropriate distance metric can lead to great performance for distance-based techniques (Weinberger and Saul, 2009; Bellet et al., 2015). In addition, once the distance metric is learned, it can be incorporated into downstream multivariate analysis techniques. Therefore, distance metric learning is particularly appealing for experiments in which prior knowledge such as single-cell labeling is available.

In this chapter, we perform an analysis of cytometry data using distance metric learning. More specifically, we determine a Mahalanobis distance metric using the Distance Metric Learning through Maximization of the Jeffrey divergence (DMLMJ) method, which is described in Chapter 3. DMLMJ enables the quantification of single-cell distances in a data-driven way. In order to characterize and validate the functionality of distance metric learning for single-cell data, several experiments are conducted for two different cytometry data sets, one generated by flow cytometry of a synthetic microbial ecosystem and one generated by mass cytometry (CyTOF) for human blood cells. Data are retrieved from experiments that are publicly available. Experiments and evaluation metrics are reported per data set. Readers are referred to the original publications for a full overview of data collection and preprocessing. The performance of DMLMJ is compared to the baseline Euclidean distance metric. Experiments and evaluation metrics are reported per data set. All processed data sets, MATLAB code for DMLMJ and Python scripts can be found at https://github.com/bacnguyencong/CytoDMLMJ.

5.2. Synthetic microbial communities

5.2.1. Data description

Data set 1: *In silico* bacterial communities

Data from 20 individual bacterial cultures measured through flow cytometry (FCM) are retrieved from FlowRepository ID: FR-FCM-ZY6M (Rubbens et al., 2017b). In brief, samples are stained with SYBR Green I and measured subsequently. Most bacterial cultures ($n = 17$) are in early-to-mid stationary phase, the rest ($n = 3$) still are in exponential or linear growth phase. The samples are analyzed on a 3-laser FACSVerse flow cytometer (BD Biosciences), which contains two scatter detectors

(forward and side) and eight fluorescence detectors, in which the FITC-detector (527/32 nm) is the targeted detector. Because peak area, height and width signals are captured, the experiment results in 30 measured variables in total. All variables are considered in DMLMJ, although only a subset of them contain biologically relevant information (Rubbens et al., 2017b). In this way, the functionality of DMLMJ can be evaluated, as we know that a linear transformation should discover this information automatically. A full description of experimental details and preprocessing can be found in Rubbens et al. (2017b). After measurement, samples are denoised in the asinh-transformed bivariate FITC-H – PerCP-Cy5.5-H space, using a robust digital gating strategy (Props et al., 2016). To ensure the quality of the data, the data are additionally filtered using the automated package flowAI (v1.4.4., default settings, target channel = FITC, changepoint detection penalty = 200) (Monaco et al., 2016). A full list of bacterial species and experimental details can be found in Rubbens et al. (2017a,b).

Data set 2: *In silico* autofluorescent microbial communities

Data are collected from FlowRepository ID: FR-FCM-ZYLB (Sgier et al., 2016), in which cyanobacterial and algal cultures are cultured and measured by FCM, using a Beckman-Coulter Gallios flow cytometer. As these microbial populations exhibit autofluorescence, no fluorescence staining is needed. Ten fluorescence and two scatter detectors measure area, height and width signals from the pulse, resulting in 36 variables in total describing the experiment. Only samples that contain more than 500 cells per replicate are considered, resulting in 31 individual strains, which are used for further analysis.

5.2.2. Experimental setup

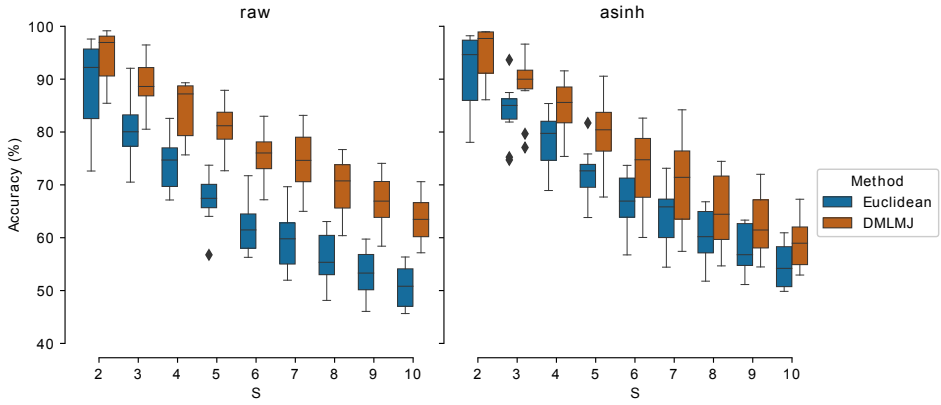
Microbial communities are created in different compositions using a data-aggregation step. In other words, cells are sampled from bacterial populations that are measured individually and combined into artificial communities, so-called *in silico* communities (Rubbens et al., 2017a). The same number of cells ($n = 10,000$ for the first data set, $n = 1,000$ for the second data set) are sampled for every population, distributed over the number of technical replicates that are available. Half of the cells are added to a training set and the other half to the test set for every *in silico* community. The complexity of a community can be expressed in terms of the observed species richness S, denoting the number of distinct microbial populations that are combined in a community. The total number of cells in both training and test set amounts to $S \times n$. The following experiments are conducted for both data sets. *In silico* communities are assembled for every increment of S ranging from two to ten. Performance is evaluated in terms of classification accuracy of cells

that are part of a held-out test set. In all cases, cells are classified according to their phylogeny using k-NN classification based on the Euclidean distance metric or the Mahalanobis distance metric learned by DMLMJ. Experiments are carried out using two different settings, on the raw data and on data for which each variable is asinh-transformed.

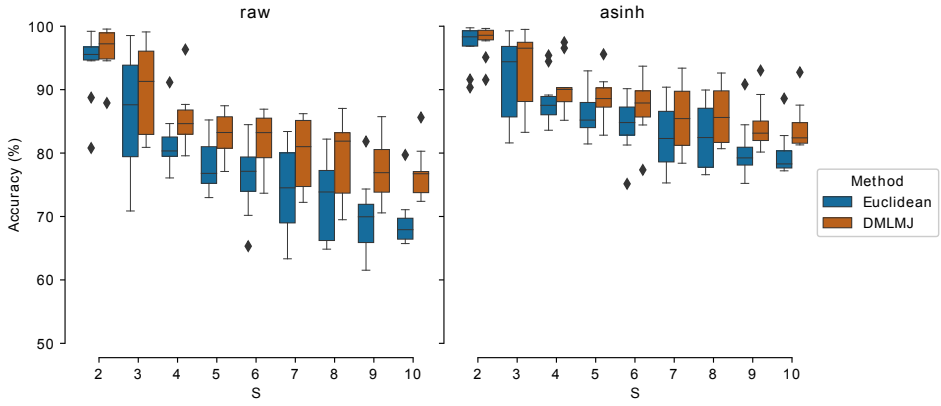
5.2.3. Results

A total of 90 different microbial communities were assembled by using a data-aggregation step, creating *in silico* communities. Ten communities were sampled for every increment of $S = 2, ..., 10$, in which S denotes the total number of microbial populations that were present in a community. Note that these populations were determined beforehand, and upon creation of an *in silico* community, a number of these populations will overlap. The impact of the use of a learned Mahalanobis distance metric on k-NN classification was evaluated in terms of the classification accuracy, which denotes the fraction of correctly labeled cells according to the phylogeny of a single cell. We compared the Mahalanobis distance metric learned by DMLMJ to the Euclidean distance metric in the context of k-NN classification with and without transforming with the use of asinh. The accuracy was evaluated using a held-out test set. This was done for two different data sets, the first containing 20 bacterial populations stained with SYBR Green I, the second containing 31 microbial populations (cyanobacteria and algae) with autofluorescent properties (see Fig. 5.1).

Using the arcsine hyperbolic function as a preprocessing step for microbial flow cytometry data improved classification accuracy for both data sets (average increase in accuracy was 4.1% for data set 1 and 7.9% for data set 2). Performance increased subsequently when DMLMJ was applied on asinh-transformed data (on average 5.4% for data set 1, 2.7% for data set 2). An increase in the number of populations resulted in a drop in accuracy for all methods. DMLMJ was able to boost the performance to a larger extent when applied to data that was not transformed (average increase in accuracy now becomes 11.8% for data set 1 and 5.5% for data set 2). Note that optimizing the distance metric without transforming the data resulted in the best performance for data set 1 (average accuracy over all communities was 77.7%), while for data set 2 a combination of asinh and DMLMJ resulted in the best predictions (which resulted in an average accuracy over all communities of 88.6%). However, when using the Euclidean distance metric, the use of asinh improved identification considerably for both data sets. A number of variables for data set 1 did not contain biologically relevant information, which DMLMJ was able to successfully filter out by means of a linear transformation. We conclude that DMLMJ captured the similarity between examples of the same phylogeny and gave rise to a linear transformation of the data resulting in an improved classification of single cells.



(a) Data set 1: *In silico* bacterial communities



(b) Data set 2: *In silico* autofluorescent microbial communities

Figure 5.1: Classification accuracy of k-NN classification for an increasing population richness S with and without the use of DMLMJ. Each boxplot contains the classification accuracy for ten communities. Each box displays the 25% and 75% quartiles of the classification accuracy, of which the whiskers extend the range to maximal 1,5 times the interquartile range. Points that lie outside this range are visualized as outliers.

5.3. Mass Cytometry

5.3.1. Data description

Data set 1: 13-dimensional CyTOF Data

Data originate from one healthy individual, in which bone marrow mast cells (BMMCs) were analyzed using a 13-color panel. Cell populations were labeled after manual gating using all markers; all markers were used for data analysis.

This data set, as processed by Weber and Robinson (2016), is publicly available on FlowRepository (ID: FR-FCM-ZZPH).

Data set 2: 32-dimensional CyTOF Data

The second CyTOF data set originates from two healthy individuals, in which BMMCs were analyzed using a 32-color panel. Cell populations were labeled after manual gating, which was done using 19 out of the 32 surface markers. All markers were used for data analysis. This data set, as presented by Weber and Robinson (2016), is publicly available on FlowRepository (ID: FR-FCM-ZZPH).

5.3.2. Experimental setup

Following Weber and Robinson (2016), data were preprocessed using an asinh transformation with a standard cofactor of 5, $f(x) = \text{asinh}(x/5)$. Training and test sets were created in a stratified manner. For the first data set, 20,000 labeled cells were added to the training set and test set, respectively. For the second data set, data were divided according to the individual (ID 1 or 2), before the creation of a training and test set, which each contained 15,000 cells. The following experiments were conducted:

1. Single-cell classification was compared using the Euclidean distance metric and the Mahalanobis distance metric learned by DMLMJ in the context of k-NN classification. The hyperparameters were tuned to maximize the average F1-score per cell class, which accounts for imbalanced data sets. The F1-score is calculated as the harmonic average of the precision (which quantifies the number of false positives) and recall (which quantifies the number of false negatives), and lies between zero and one. An F1-score of one resembles perfect cell label classification.
2. The visualization performance was assessed using t-SNE (van der Maaten and Hinton, 2008) on test sets using the Euclidean distance metric and the Mahalanobis distance metric learned by DMLMJ.

5.3.3. Results

DMLMJ was evaluated for two mass cytometry (CyTOF) data sets. Data were first split in a training and test set, after which the Mahalanobis distance metric was determined based on the training sets in function of the average F1-score over all cell populations. Performances of k-NN classification are reported for the test sets (Fig. 5.2). Although identification performance using the Euclidean distance metric is high, DMLMJ improved the performance to some extent.

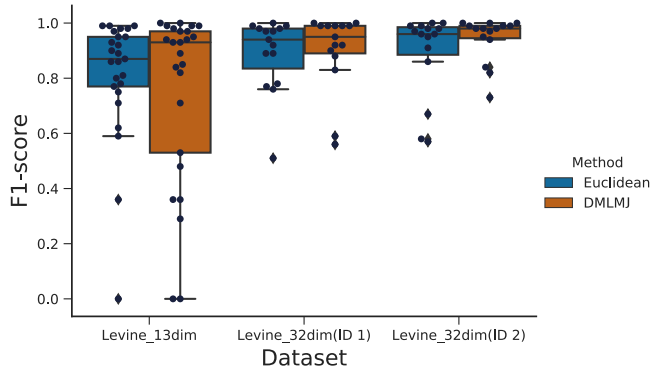


Figure 5.2: F1-score with and without the use of DMLMJ using k-NN classification of single-cell labels for CyTOF data. Boxplots show the distribution of F1-scores per data set and per cell population, in which each cell population is represented by a black dot. Each boxplot displays the 25% and 75% quartiles of the F1-score, of which the whiskers extend the range to maximal 1,5 times the interquartile range. Points that lie outside this range are visualized as outliers.

Next, we employed t-SNE on the test sets for visualization purposes, with and without the use of DMLMJ to visualize the data (Figs. 5.3, 5.4, and 5.5). Although t-SNE already returned an acceptable visualization of the data, DMLMJ improved the visualization to some extent. Most notably, megakaryocyte and erythroblast cells were more separated for the Levine_13dim data set as opposed to a fully unsupervised analysis of the data. In the Levine_32dim data set, CD16+ natural killer (NK) cells were clearly separated from CD16- NK cells for individual 1. For individuals 1 and 2, basophils were separated from plasmacytoid dendritic cells (pDCs) and the CD34+/CD38+/CD13+ hematopoietic stem and progenitor cells (HSPCs). In general, separation between large cell populations that were already separated improved slightly because of DMLMJ.

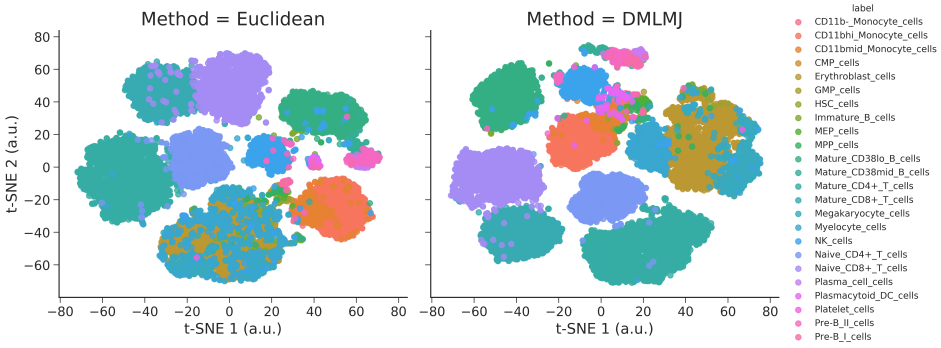


Figure 5.3: Visualization of cell populations using t-SNE the 2-dimensional space for the Levine_13dim data set, with and without the use of DMLMJ.

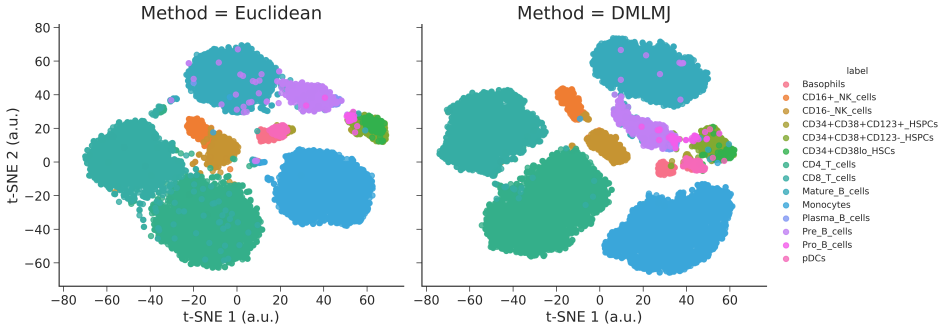


Figure 5.4: Visualization of cell populations using t-SNE the 2-dimensional space for the Levine_32dim data set, with and without the use of DMLMJ.

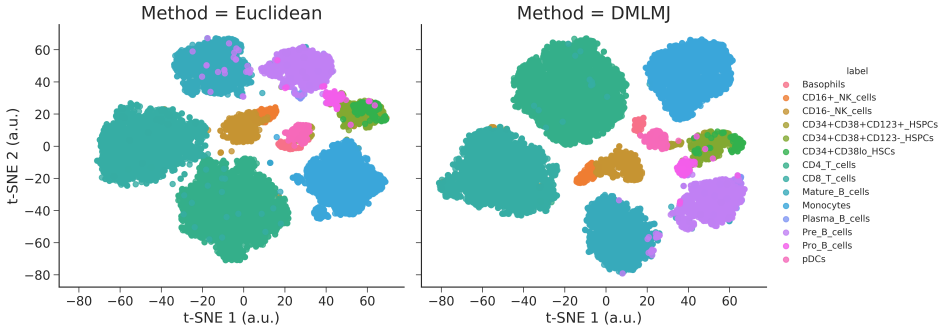


Figure 5.5: Visualization of cell populations using t-SNE in the 2-dimensional space for the Levine_32dim data set, with and without the use of DMLMJ.

5.4. Discussion and conclusion

In this chapter, we have explored the use of Distance Metric Learning through Maximization of the Jeffrey divergence (DMLMJ) for single-cell data analysis. A thorough survey was performed considering the functionality of distance metric learning for different cytometry data sets. While the Euclidean distance metric is often used for identification of cell populations, we have showed that the performance of distance-based multivariate analysis techniques can be improved by employing an appropriate distance metric. A few studies have discussed the impact of alternative distance measures for automated cell population identification. For instance, Van Gassen et al. (2015) reported that the Euclidean distance metric gave the best results compared to the Manhattan and Chebyshev distance metrics for the FlowSOM algorithm. Boddy et al. (2000) noted that a Mahalanobis distance metric consistently resulted in a 4% increase in classification accuracy compared to scaled Euclidean distances for the classification of phytoplankton single cells using neural networks.

Distance metric learning can provide an alternative way to incorporate domain knowledge. When cell population annotation is available for at least one sample, this information can be included in automated cell annotation techniques to analyze samples that have been studied with the same experimental setup. Naturally, the performance of distance metric learning depends on the quality of the manually annotated dataset. We hypothesize that the quality of the data affected the results for the Levine_32dim dataset, in which the distance metric determined for individual 1 resulted in a comparable performance when used for the analysis of individual 2, but this did not hold in the opposite way.

The performance of distance-based data analysis techniques depends on the used distance metric. Distance metric learning provides a solution to improve their performance when some supervised information, e.g. single-cell labels, is available. Since distance metric learning is a robust property of the data setup at hand, it offers a way to incorporate domain knowledge into additional multivariate analyses, which can help to address sources of variability, such as microbial heterogeneity or batch effects in mass cytometry.

PART III

DISTANCE METRIC LEARNING USING TRIPLET CONSTRAINTS

6 Scalable large-margin distance metric learning using stochastic gradient descent

In this chapter, we propose a large-margin-based approach, named Large-Margin Distance Metric Learning (LMDML), for learning a Mahalanobis distance metric. LMDML employs the principle of margin maximization to learn the distance metric with the goal of improving k-NN classification. The main challenge of distance metric learning is the positive semidefiniteness constraint on the Mahalanobis matrix. Semidefinite programming is commonly used to enforce this constraint, but it becomes computationally intractable on large-scale data sets. To overcome this limitation, we develop an efficient algorithm based on stochastic gradient descent (SGD). Our algorithm can avoid the computations of the full gradient and ensure that the learned matrix remains within the positive semidefinite (PSD) cone after each iteration.

The material of this chapter is based on the following publication:
Nguyen, B., Morell, C., and De Baets, B. (2018b). Scalable large-margin distance metric learning using stochastic gradient descent. *IEEE Transactions on Cybernetics*, accepted

6.1. Motivation

Most studies focus on learning a Mahalanobis distance metric due to its wide use in many real-world applications (Bellet et al., 2015). The Mahalanobis distance metric is parametrized by a symmetric positive semidefinite (PSD) matrix $\mathbf{M} \in \mathbb{R}^{D\times D}$, where the distance between two examples $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^D$ is computed as $d_{\mathbf{M}}(\mathbf{u},\mathbf{v}) = \sqrt{(\mathbf{u}-\mathbf{v})^\top \mathbf{M}(\mathbf{u}-\mathbf{v})}$. Alternatively, several authors propose to learn a similarity function instead of a distance metric (Chechik et al., 2010; Liu et al., 2014, 2015a). They focus on learning a bilinear similarity function, parametrized by an arbitrary matrix $\mathbf{M} \in \mathbb{R}^{D\times D}$, where the similarity between two examples $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^D$ is computed as $s_{\mathbf{M}}(\mathbf{u},\mathbf{v}) = \mathbf{u}^\top \mathbf{M}\mathbf{v}$. The bilinear similarity function is very close to the cosine function when $\mathbf{M}$ is set to be the identity matrix and $\mathbf{u}$, $\mathbf{v}$ are normalized to unit length. In general, the matrix $\mathbf{M}$ is not required to be PSD, or not even to be symmetric.

Nevertheless, in both cases, the positive semidefiniteness and the symmetry constraints on the matrix $\mathbf{M}$ may provide a useful regularization tool to prevent over-

fitting on high-dimensional data sets (Chechik et al., 2010). Furthermore, we can project the data into a new space by factorizing $\mathbf{M} = \mathbf{L}\mathbf{L}^\top$, such that the distance or similarity is computed in the transformed space $\widehat{\mathbf{u}} = \mathbf{L}^\top \mathbf{u}$, where the Euclidean distance corresponds to $d_{\mathbf{M}}$ and the dot product corresponds to $s_{\mathbf{M}}$.

In recent years, the computational efficiency of distance metric learning approaches has been substantially improved by using the projected gradient descent (Weinberger and Saul, 2009), coordinate descent (Nguyen et al., 2016), Frank-Wolfe (Ying and Li, 2012), and Bregman projection (Davis et al., 2007) algorithms. However, there still remain some scalability issues in the current literature on learning a distance metric.

The first scalability issue is related to the number of features. Learning a Mahalanobis distance metric requires to estimate a $D \times D$ matrix. This quadratic dependency poses a huge challenge for real-world problems that involve thousands of features, since the performance of a learning algorithm degrades as the number of features grows (Duda et al., 2012). This is often referred to as the "curse of dimensionality." Another limitation arises from the positive semidefiniteness constraint on the Mahalanobis matrix which requires projections onto the PSD cone, which scales as $O(D^3)$. In high-dimensional settings, most of the existing approaches become quickly intractable. Although there exist dimensionality reduction techniques such as principal component analysis (PCA) (Jolliffe, 2005), it may be still impossible to satisfactorily reduce the number of features without a significant loss of information contained in the training data. For these reasons, many distance metric learning algorithms are successful in low-dimensional settings (Bar-Hillel et al., 2005; Wang, 2011a; Weinberger and Saul, 2009; Ying and Li, 2012; Xing et al., 2002; Shen et al., 2012), however, they fail when applied in high-dimensional settings.

The second scalability issue is related to the number of training examples. Considering the increasing amount of data, the computational complexity of a learning algorithm becomes a critical limitation. One solution is to use online learning algorithms, particularly stochastic gradient descent (SGD) (Robbins and Monro, 1951), which considers only a single example at each iteration. The SGD algorithm is significantly more scalable than the batch gradient descent algorithm (Bottou, 1991). However, in the distance metric learning context, both algorithms share a common limitation: they need to make a projection of the Mahalanobis matrix onto the PSD cone after each iteration.

Our distance metric learning algorithm is motivated by these issues. In particular, we propose an efficient strategy based on SGD in which each iteration requires a cheaper computation than performing the eigen-decomposition to keep the solution within the PSD cone. By using a hinge loss function for learning the Mahalanobis distance metric, our algorithm can further reduce the number of updates and projections. In short, our main contributions are the following:

(i) We propose a distance metric learning approach for k-NN classification based on the principle of margin maximization inspired by the margin definition in (Weinberger and Saul, 2009). By considering the trace-norm minimization, our approach can lead to a low-rank solution, thus reducing the risk of overfitting. We refer to the proposed approach as Large-Margin Distance Metric Learning (LMDML).

(ii) To apply LMDML in large-scale settings, we develop an efficient online algorithm based on SGD. Our algorithm, named LMDML-A, keeps the solution always within the PSD cone by computing an appropriate step size in each iteration. We use the Schur complement to find an upper bound of the step size that guarantees that the solution remains within the PSD cone.

The remainder of this chapter is organized as follows. Section 6.2 briefly reviews some related work, focusing on the main problems of large-scale distance metric learning with existing algorithms that are addressed by our algorithm. Section 6.3 introduces our distance metric learning approach (LMDML). Section 6.4 presents an online learning algorithm based on SGD to apply LMDML in large-scale settings. We also analyze the computational complexity of the proposed algorithm and provide some useful recommendations for the implementation to reduce the training time. We conduct extensive experiments to evaluate our algorithm in Section 6.5. Finally, we give a discussion of future work and some conclusions in Section 6.6.

6.2. Related work

Distance metric learning has been successfully used in many different disciplines and applications, such as classification (Weinberger and Saul, 2009), regression (Weinberger and Tesauro, 2007; Nguyen et al., 2016), computer vision (Köstinger et al., 2012; Yang et al., 2014), and so on. Many previous approaches have been proposed, including neighborhood component analysis (NCA) (Goldberger et al., 2005), maximally collapsing metric learning (MCML) (Globerson and Roweis, 2006), information-theoretic metric learning (ITML) (Davis et al., 2007), large margin nearest neighbor classification (LMNN) (Weinberger and Saul, 2009), and metric learning through maximization of the Jeffrey divergence (DMLMJ) (see Chapter 3). However, it becomes a very challenging problem for machines when the number of training examples is large or the dimensionality is high. Below we briefly review some recent work that has attempted to tackle this challenge. To be more specific, we will discuss three keys issues that are addressed by our approach: high-dimensional data, large data sets, and low-rank distance metrics.

The main computational challenge is the positive semidefiniteness constraint, especially when dealing with high-dimensional data. A number of approaches (Davis

et al., 2007; Ying and Li, 2012; Shen et al., 2012; Jin et al., 2009) have been proposed to reduce the expensive cost of projections onto the PSD cone. Hazan and Kale (2012) replaced the projection step by solving a constrained linear program, which is simpler than the projection onto the PSD cone. Mahdavi et al. (2012) addressed this problem by avoiding intermediate projections. Ying and Li (2012) reduced this computational burden by using the Frank-Wolfe algorithm, which requires only the minimum eigenvalue and corresponding eigenvector at each iteration. Unfortunately, the computational efficiency of these approaches is limited for large data sets because the computational cost of each iteration is still expensive.

Recent approaches (Davis et al., 2007; Chechik et al., 2010; Shalev-Shwartz et al., 2004; Gao et al., 2014; Qian et al., 2015a) addressed this challenge using online learning algorithms. In the context of similarity learning, Chechik et al. (2010) introduced an online learning approach, named OASIS, focusing on large-scale data sets with millions of training examples. However, OASIS may increase the risk of overfitting since it does not take into account the possibility that examples lie in a low-dimensional subspace. In contrast to OASIS, our approach enforces the low-rank constraint as well as the positive semidefiniteness constraint on the Mahalanobis matrix. Consequently, our approach would intuitively reduce the risk of overfitting. In more closely related work, Qian et al. (2015a) addressed this challenge by exploiting the SGD algorithm. To reduce the number of projections onto the PSD cone, they proposed a stochastic updating procedure, which consists in giving difficult constraints more chance to be used for updating. Compared to the approach proposed in (Qian et al., 2015a), our approach is simpler and more efficient because it uses a simple hinge loss function to reduce the number of projections (Shalev-Shwartz et al., 2004; Chechik et al., 2010).

Many other approaches (Nguyen et al., 2016; Gao et al., 2014; Schultz and Joachims, 2004; Shi et al., 2014) aim to speed up the training on large-scale data sets by enforcing the learned matrix to be diagonal. These approaches have a significant advantage in computational complexity as well as in memory complexity, making them tractable in large-scale settings. However, the resulting distance metric or similarity function is very restrictive because it neglects the possible correlation between features.

Another research direction focuses on learning a low-rank distance metric via learning a linear transformation. By imposing the low-rank constraint explicitly on the linear transformation matrix, one can limit the number of parameters of the learned distance metric, thus reducing the risk of overfitting. Unfortunately, most of the approaches that focus on learning a linear transformation lead to nonconvex optimization problems (Goldberger et al., 2005; Liu et al., 2013; Lim and Lanckriet, 2014). Hence, they can suffer from spurious local minima and require careful tuning of the matrix rank. In (Zhang and Zhang, 2017), the authors exploit the low-rank structure of intermediate solutions in order to reduce the computation and space

complexity. The projection onto the PSD cone can be addressed efficiently using incremental SVD, but it is not clear how the low-rank constraint is guaranteed at each iteration, and consequently, it may not always result in the desired performance. Another solution is to learn a linear combination of rank-one matrices (Qian et al., 2015b), which enjoys the convexity property. However, it also requires specifying the number of basis matrices. In contrast to these approaches, our approach can find a low-rank Mahalanobis matrix, which induces a low-rank linear transformation, and it ensures to achieve a global convergence since it is formulated as solving a convex optimization problem. Unlike in (Zhang and Zhang, 2017), we do not make any low-rank assumption on the intermediate solutions during training.

6.3. Problem formulation

We will consider the standard supervised classification problem. The set of training examples is denoted by $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i \in \{1, \dots, n\}\} \subset \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{R}^D$ denotes the set of feature vectors and $\mathcal{Y}$ denotes the set of class labels. Let us introduce the definitions of hit examples and miss examples.

Definition 6.1 (Hit examples)**.** *Let $\mathbf{x}_i$ be an example in $\mathcal{X}$. The hit examples of $\mathbf{x}_i$ are the elements of the set $\mathcal{H}(\mathbf{x}_i)$ consisting of the examples in $\mathcal{X} \setminus \{\mathbf{x}_i\}$ that share the same class label with $\mathbf{x}_i$, i.e. $\mathcal{H}(\mathbf{x}_i) = \big\{ \mathbf{x}_j \mid j \in \{1, \dots, n\}, j \neq i, y_j = y_i \big\}$.*

Definition 6.2 (Miss examples)**.** *Let $\mathbf{x}_i$ be an example in $\mathcal{X}$. The miss examples of $\mathbf{x}_i$ are the elements of the set $\mathcal{M}(\mathbf{x}_i)$ consisting of the examples in $\mathcal{X}$ that do not share the same class label with $\mathbf{x}_i$, i.e. $\mathcal{M}(\mathbf{x}_i) = \big\{ \mathbf{x}_j \mid j \in \{1, \dots, n\}, y_j \neq y_i \big\}$.*

Recently, margins have been extensively studied in the distance metric learning context. We can formulate the margin as a function depending on the PSD matrix $\mathbf{M}$. Empirical evidence (Moutafis et al., 2017; Weinberger and Saul, 2009; Parameswaran and Weinberger, 2010; Torresani and Lee, 2007) has demonstrated that distance metric learning approaches employing the principle of margin maximization are more robust than other distance metric learning approaches such as NCA (Goldberger et al., 2005) and RCA (Bar-Hillel et al., 2005).

Due to the nature of the decision rule of the k-NN classifier, each example should share the same class label with the majority of its k nearest neighbors. If we ensure that the neighbors of the same class are closer than the neighbors of the other classes, then the k-NN classifier will be successful. Adopting the same terminology as in (Weinberger and Saul, 2009), we define $\mathcal{T}(\mathbf{x}_i)$, called target neighbors of $\mathbf{x}_i$, as a set of k examples in $\mathcal{H}(\mathbf{x}_i)$ that should be close to $\mathbf{x}_i$ and that share the same class label with $\mathbf{x}_i$ in the training set. The goal is to learn a distance metric that makes the target neighbors of $\mathbf{x}_i$ become its k nearest neighbors. Target neighbors can be selected based on prior knowledge (if available) or simply by searching the

k nearest neighbors with the same class label using the Euclidean distance metric. Note that the target neighbors do not change during the training stage.

We define the margin of a labeled example $\mathbf{x}_i$ for the purpose of measuring the confidence of the k-NN classifier as follows.

Definition 6.3. *Let $\mathbf{x}_i$ be a labeled example in $\mathcal{X}$. The margin of $\mathbf{x}_i$ corresponding to the Mahalanobis distance metric parameterized by $\mathbf{M}$ is defined as:*

$$\phi_{\mathbf{M}}(\mathbf{x}_i) = d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_i^-) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_i^+)\,,$$

where

$$\begin{aligned} \mathbf{x}_i^+ &= \operatorname*{arg\,max}_{\mathbf{x}_j \in \mathcal{T}(\mathbf{x}_i)} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)\,, \\ \mathbf{x}_i^- &= \operatorname*{arg\,min}_{\mathbf{x}_j \in \mathcal{M}(\mathbf{x}_i)} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)\,. \end{aligned} \tag{6.1}$$

As is common in distance metric learning, we use the squared Mahalanobis distances to express the margin, leading to a convex function in terms of $\mathbf{M}$. Roughly speaking, the margin is the difference between the squared distance from $\mathbf{x}_i$ to the nearest example with a different class label and the squared distance from $\mathbf{x}_i$ to the farthest example in its target neighbor set. A similar definition of a margin was also introduced in (Nguyen and Guo, 2008). Using this margin definition, we now turn to develop our distance metric learning approach, which is the result of two fundamental aims.

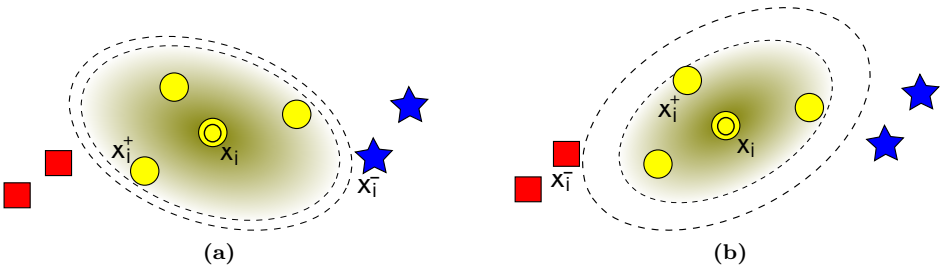


Figure 6.1: Illustration of the intuition behind LMDML. Examples belonging to the same class are denoted in the same color and style. (a) A separating ellipse with a small margin. (b) A separating ellipse with a large margin.

Our first aim is that the learned distance metric should guarantee that many examples have large margins. We follow the large-margin principle that has been successfully used in SVMs (Cortes and Vapnik, 1995), AdaBoost (Schapire et al., 1997), and LMNN (Weinberger and Saul, 2009) algorithms. The aim of our distance metric learning approach is to maximize the sum of all local margins. Using a

hinge loss function with margin one, the latter is equivalent to minimize

$$\epsilon(\mathbf{M}) = \sum_{i=1}^{n} \left[1 + d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_i^+) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_i^-)\right]_+ ,$$

where the function $[.]_+$ denotes the positive part of its argument.

Our second aim is that the learned distance metric should be able to detect irrelevant or noisy features in the input space. Assume that the input examples lie in a low-dimensional subspace $\mathbb{R}^m$, where $m < D$, then the Mahalanobis matrix with low rank m can satisfactorily distinguish any two distinguishable examples (Cong et al., 2014). Thus, a high-rank Mahalanobis matrix may be suffering from overfitting effects as the training data always contain noisy features in practice. To this end, we force the low-rank constraint on the Mahalanobis matrix $\mathbf{M}$. That means, among the Mahalanobis matrices that maximize the margins, we prefer the low-rank ones. As is commonly done, we use the nuclear norm to approximate the rank function. The nuclear norm of the matrix $\mathbf{M}$ is defined as $\|\mathbf{M}\|_* = \sum_i \sigma_i$, where σ_i are the singular values of $\mathbf{M}$. Since $\mathbf{M}$ is PSD, it holds that $\|\mathbf{M}\|_* = \mathrm{tr}(\mathbf{M})$. Note that the trace norm of a PSD matrix is equal to the ℓ_1-norm of its diagonal elements, which is used by the popular Lasso algorithm (Tibshirani, 1996). The theoretical justification of the ℓ_1-norm can be found in (Donoho and Elad, 2003) and the references therein.

Finally, we combine the two aims, the loss function $\epsilon(\mathbf{M})$ based on margins and the low-rank constraint on $\mathbf{M}$, into a single objective function for learning the distance metric. It leads to the following optimization problem

$$\underset{\mathbf{M} \succcurlyeq 0}{\text{minimize}} \quad C\,\mathrm{tr}(\mathbf{M}) + \frac{1}{n} \sum_{i=1}^{n} \left[1 + d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_i^+) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_i^-)\right]_+ , \tag{6.2}$$

where $C > 0$ is a hyper-parameter. We refer to the proposed approach as *Large-Margin Distance Metric Learning*, abbreviated as LMDML. Figure 6.1 illustrates the idea behind LMDML. Although the distance metrics in both cases are able to classify well the example $\mathbf{x}_i$, a better generalization is expected from the distance metric with a larger margin.

Problem (6.2) is a convex semidefinite program, therefore, it can be solved by standard semidefinite optimization algorithms, such as interior-point methods (Boyd and Vandenberghe, 2004). However, these algorithms usually need to calculate the Hessian matrix, which requires a memory complexity of $O(D^4)$ and a time complexity of $O(D^{6.5})$ in the worst case. For some real-world applications, they become almost intractable. Another alternative for minimizing the objective function in (6.2) is to use a simple first-order algorithm such us batch gradient descent as in (Weinberger and Saul, 2009; Xing et al., 2002; Globerson and Roweis, 2006). The use of first-order algorithms can reduce the time complexity per iteration

since they only require the information from the first derivative of the objective function. Unfortunately, these algorithms do not scale well to large data sets because they need the full-gradient computation in each iteration. In the next section, we will introduce our algorithm to overcome these limitations.

6.4. Online distance metric learning

To make the optimization technique tractable for large-scale problems, we should take into account both the number of iterations and the computational cost of each iteration. SGD provides a way to avoid the full-gradient computation by considering only a single loss term at a time. SGD has been successfully exploited in many machine learning algorithms such as SVMs (Shalev-Shwartz et al., 2007), neural networks (Bottou, 1991), and Lasso (Shalev-Shwartz and Tewari, 2011). By randomly choosing an example at each iteration, SGD can directly optimize the expected loss (Bottou, 1991) and remove the time complexity dependency on the size of problem. Besides, SGD is extremely simple to implement and highly scalable, which make it particularly suitable for large-scale learning problems. However, like other gradient descent techniques, it requires a projection step to get the solution back to the PSD cone after each iteration. In this section, we develop an online algorithm based on SGD that needs no projection steps in order to keep the solution within the PSD cone.

6.4.1. Stochastic gradient descent for distance metric learning

For any fixed $C > 0$ in problem (6.2), there is always some choice of $B > 0$ such that the optimal solution of problem (6.2) results in the same objective function value as the optimal solution of

$$\begin{array}{ll} \text{minimize} & f(\mathbf{M}) = \frac{1}{n}\sum_{i=1}^{n}\left[1 + d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_i^+) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_i^-)\right]_+ \\ \text{subject to} & \operatorname{tr}(\mathbf{M}) \leq B\,, \\ & \mathbf{M} \succcurlyeq 0\,. \end{array} \tag{6.3}$$

The formulation of problem (6.2) is also known as the Lagrangian version of problem (6.3). We now focus on solving problem (6.3) due to its interesting expression: the objective function in (6.3) has the form of a sum of loss functions associated with each example used for training, which allows us to use a stochastic optimization algorithm. We refer to the proposed algorithm as LMDML-A.

At the t-th iteration, LMDML-A operates as follows. First, it chooses a random training example $\mathbf{x}_i$ by picking an index $i \in \{1, \ldots, n\}$ uniformly at random. Then,

it replaces the objective function of problem (6.3) with an approximation based on the single randomly picked example,

$$f_i(\mathbf{M}_t) = \left[1 + d^2_{\mathbf{M}_t}(\mathbf{x}_i, \mathbf{x}_i^+) - d^2_{\mathbf{M}_t}(\mathbf{x}_i, \mathbf{x}_i^-)\right]_+ .$$

If $1 + d^2_{\mathbf{M}_t}(\mathbf{x}_i, \mathbf{x}_i^+) \leq d^2_{\mathbf{M}_t}(\mathbf{x}_i, \mathbf{x}_i^-)$, then the subgradient of the above approximate objective function $\nabla f_i(\mathbf{M}_t)$ becomes zero (i.e., there is no need to update the Mahalanobis distance metric). Otherwise, the subgradient $\nabla f_i(\mathbf{M}_t)$ is given by

$$\nabla f_i(\mathbf{M}_t) = (\mathbf{x}_i - \mathbf{x}_i^+)(\mathbf{x}_i - \mathbf{x}_i^+)^\top - (\mathbf{x}_i - \mathbf{x}_i^-)(\mathbf{x}_i - \mathbf{x}_i^-)^\top .$$

Next, we update the matrix $\mathbf{M}_t$ in the direction of the subgradient with a step size $\eta_t = c/\sqrt{t}$, where $c > 0$ is a constant,

$$\begin{aligned}
\mathbf{M}_{t+1/3} &= \mathbf{M}_t - \eta_t \nabla f_i(\mathbf{M}_t) , \\
\mathbf{M}_{t+2/3} &= \Pi_{\mathbb{S}^D_+}(\mathbf{M}_{t+1/3}) , \\
\mathbf{M}_{t+1} &= \min(B/\mathrm{tr}(\mathbf{M}_{t+2/3}), 1)\mathbf{M}_{t+2/3} .
\end{aligned}$$

The last two steps are used to keep the solution within the PSD cone and to satisfy the trace-bound constraint, respectively. According to the Interlacing Theorem (Golub and Van Loan, 1996), the matrix $\mathbf{M}_{t+1/3}$ contains at most one negative eigenvalue. Similarly to the method proposed by Shalev-Shwartz et al. (2004), we can compute $\mathbf{M}_{t+2/3}$ using the following formulation,

$$\mathbf{M}_{t+2/3} = \mathbf{M}_{t+1/3} - \min(\lambda_{\min}, 0)\mathbf{u}_{\min}\mathbf{u}_{\min}^\top ,$$

where $\lambda_{\min}$ is the smallest eigenvalue of $\mathbf{M}_{t+1/3}$ with corresponding eigenvector $\mathbf{u}_{\min}$. Finding the smallest eigenvalue of the symmetric matrix $\mathbf{M}_{t+1/3}$ is equivalent to finding the largest eigenvalue of $-\mathbf{M}_{t+1/3}$. In (Kuczyński and Woźniakowski, 1992), it was shown that the largest eigenvalue of a positive definite matrix can be approximated in $O(D^2)$ time using the Lanczos method or the power method with a random start vector. However, the matrix $-\mathbf{M}_{t+1/3}$ is not always positive definite. To circumvent this, we can add a large enough constant $r > 0$, so that $\mathbf{A} = r\mathbf{I} - \mathbf{M}_{t+1/3}$ becomes positive definite. Finally, the smallest eigenvalue of $\mathbf{M}_{t+1/3}$ becomes $r - \lambda_{\max}(\mathbf{A})$, where $\lambda_{\max}(\mathbf{A})$ is the largest eigenvalue of $\mathbf{A}$ with the corresponding eigenvector $\mathbf{u}_{\min}$.

Although the cost of the projection onto the PSD cone has been reduced, its computational efficiency is limited because it requires a numerical approximation to compute, at each iteration, the smallest eigenvalue and corresponding eigenvector of the updated Mahalanobis matrix. For instance, the Lanczos method requires $O(\log(n)/\sqrt{\gamma})$ iterations (Kuczyński and Woźniakowski, 1992) to approximate a unit vector $\mathbf{u}$ such that $\mathbf{u}^\top\mathbf{A}\mathbf{u}/\lambda_{\max}(\mathbf{A}) \geq 1 - \gamma$. This iterative procedure may

be expensive when the number of features is relatively large. We address this challenge by finding an appropriate step size to keep the solution within the PSD cone. Therefore, we can avoid the projection step of the Mahalanobis matrix onto the PSD cone. This idea of approximating the solution through finding a step size was also introduced in (Jin et al., 2009) for learning a distance metric based on pairwise constraints. In order to compute the step size, Jin et al. (2009) used the conjugate gradient method, which is still slow in high-dimensional settings. We employ the Schur complement to compute an upper bound of the step size.

We begin by introducing two important theorems that help us to develop our proposal. The first theorem is a modification of the generalized inverse theorem (Campbell and Meyer, 1979), which will allow us to compute the pseudo-inverse of a symmetric matrix plus a rank-one symmetric matrix. Thus, the six cases in (Campbell and Meyer, 1979) reduce to only three cases, and the update formulas can be significantly simplified.

Theorem 6.1. *For* $\mathbf{A} \in \mathbb{S}^{D\times D}$, $\mathbf{x} \in \mathbb{R}^D$, *and* $\alpha \in \mathbb{R}$, *let* $\mathbf{k} = \mathbf{A}^\dagger\mathbf{x}$, $\mathbf{h} = \mathbf{x}^\top\mathbf{A}^\dagger$, $\mathbf{u} = (\mathbf{I} - \mathbf{A}\mathbf{A}^\dagger)\mathbf{x}$, $\mathbf{v} = \mathbf{x}^\top(\mathbf{I} - \mathbf{A}^\dagger\mathbf{A})$, *and* $\beta = 1 + \mathbf{x}^\top\mathbf{A}^\dagger\mathbf{x}$. *Then the Moore-Penrose inverse of* $\mathbf{A} + \alpha\mathbf{x}\mathbf{x}^\top$ *is computed as follows.*

(i) If $\mathbf{u} \neq \mathbf{0}$, *then*
$(\mathbf{A} + \alpha\mathbf{x}\mathbf{x}^\top)^\dagger = \mathbf{A}^\dagger - \mathbf{k}\mathbf{u}^\dagger - \mathbf{v}^\dagger\mathbf{h} + (1/\alpha - 1 + \beta)\mathbf{v}^\dagger\mathbf{u}^\dagger$.

(ii) If $\mathbf{u} = \mathbf{0}$ *and* $1 + (\beta - 1)\alpha \neq 0$, *then*
$(\mathbf{A} + \alpha\mathbf{x}\mathbf{x}^\top)^\dagger = \mathbf{A}^\dagger - 1/(1/\alpha - 1 + \beta)\mathbf{k}\mathbf{h}$.

(iii) If $\mathbf{u} = \mathbf{0}$ *and* $1 + (\beta - 1)\alpha = 0$, *then*
$(\mathbf{A} + \alpha\mathbf{x}\mathbf{x}^\top)^\dagger = \mathbf{A}^\dagger - \mathbf{k}\mathbf{k}^\dagger\mathbf{A}^\dagger - \mathbf{A}^\dagger\mathbf{h}^\dagger\mathbf{h} + (\mathbf{k}^\dagger\mathbf{A}^\dagger\mathbf{h}^\dagger)\mathbf{k}\mathbf{h}$.

Proof. The result follows directly from (Campbell and Meyer, 1979, Theorem 3.1.3) and a careful inspection of its proof. □

The second theorem will allow us to find necessary and sufficient conditions that guarantee that a symmetric PSD matrix $\mathbf{A}$ plus two rank-one symmetric matrices of the form $\mathbf{A} - \alpha(\mathbf{a}\mathbf{a}^\top - \mathbf{b}\mathbf{b}^\top)$, where α is a positive scalar and $\mathbf{a}$, $\mathbf{b}$ are real vectors, remains within the PSD cone.

Theorem 6.2. *For* $\mathbf{A} \in \mathbb{S}^{D\times D}$, $\mathbf{a} \in \mathbb{R}^D$, $\mathbf{b} \in \mathbb{R}^D$, *and* $\alpha \in \mathbb{R}$, *where* $\mathbf{A} \succcurlyeq 0$ *and* $\alpha > 0$, *let* $\mathbf{k} = \mathbf{A}^\dagger\mathbf{b}$, $\mathbf{h} = \mathbf{b}^\top\mathbf{A}^\dagger$, $\mathbf{u} = (\mathbf{I} - \mathbf{A}\mathbf{A}^\dagger)\mathbf{b}$, $\mathbf{v} = \mathbf{b}^\top(\mathbf{I} - \mathbf{A}^\dagger\mathbf{A})$, *and* $\beta = 1 + \mathbf{b}^\top\mathbf{A}^\dagger\mathbf{b}$. *Then the following constraint*

$$\mathbf{A} - \alpha(\mathbf{a}\mathbf{a}^\top - \mathbf{b}\mathbf{b}^\top) \succcurlyeq 0 \tag{6.4}$$

is satisfied if and only if any of the following sets of conditions holds (assuming that $1/0 = +\infty$*):*

(i) $\mathbf{u} \neq \mathbf{0}$, $(\mathbf{I} - \mathbf{A}\mathbf{A}^\dagger - \mathbf{u}\mathbf{u}^\dagger)\mathbf{a} = \mathbf{0}$, $1 - \mathbf{a}^\top \mathbf{v}^\dagger \mathbf{u}^\dagger \mathbf{a} > 0$, *and*

$$\alpha \leq (1 - \mathbf{a}^\top \mathbf{v}^\dagger \mathbf{u}^\dagger \mathbf{a})/(\mathbf{a}^\top \mathbf{T}\mathbf{a})\,,$$

where $\mathbf{T} = \mathbf{A}^\dagger - \mathbf{k}\mathbf{u}^\dagger - \mathbf{v}^\dagger \mathbf{h} + (\beta - 1)\mathbf{v}^\dagger \mathbf{u}^\dagger$.

(ii) $\mathbf{u} = \mathbf{0}$, $(\mathbf{I} - \mathbf{A}\mathbf{A}^\dagger)\mathbf{a} = \mathbf{0}$, *and*

$$\alpha \leq 2/(-B + \sqrt{B^2 - 4C})\,,$$

where $B = \beta + \mathbf{a}^\top \mathbf{A}^\dagger \mathbf{a} - 1$, *and* $C = -\mathbf{a}^\top (\mathbf{A}^\dagger(\beta - 1) - \mathbf{k}\mathbf{h})\mathbf{a}$.

Proof. Since $\alpha > 0$, according to the Schur complement theorem (Gallier, 2010, Theorem 4.3), constraint (6.4) is equivalent to

$$\begin{pmatrix} \mathbf{A} + \alpha \mathbf{b}\mathbf{b}^\top & \mathbf{a} \\ \mathbf{a}^\top & 1/\alpha \end{pmatrix} \succcurlyeq 0\,.$$

Note that $(\mathbf{A} + \alpha \mathbf{b}\mathbf{b}^\top) \succcurlyeq 0$, and as a consequence, it is possible to use again the Schur complement theorem. Hence, the following two conditions must hold:

$$1/\alpha - \mathbf{a}^\top (\mathbf{A} + \alpha \mathbf{b}\mathbf{b}^\top)^\dagger \mathbf{a} \geq 0 \tag{6.5}$$

and

$$(\mathbf{I} - (\mathbf{A} + \alpha \mathbf{b}\mathbf{b}^\top)(\mathbf{A} + \alpha \mathbf{b}\mathbf{b}^\top)^\dagger)\mathbf{a} = \mathbf{0}\,. \tag{6.6}$$

In other words, the inequality constraint (6.4) is replaced by conditions (6.5) and (6.6). First, we find an upper bound on α satisfying the inequality constraint (6.5). Since $\alpha > 0$ and $\mathbf{A} \succcurlyeq 0$, implying $1 + (\beta - 1)\alpha > 0$, according to Theorem 6.1, we only have to consider two disjoint cases for the proof.

In the first case, the conditions $\mathbf{u} \neq \mathbf{0}$, (6.5) and (6.6) should be satisfied. Using Theorem 6.1, we have

$$\begin{aligned} (\mathbf{A} + \alpha \mathbf{b}\mathbf{b}^\top)^\dagger &= \mathbf{A}^\dagger - \mathbf{k}\mathbf{u}^\dagger - \mathbf{v}^\dagger \mathbf{h} + (1/\alpha - 1 + \beta)\mathbf{v}^\dagger \mathbf{u}^\dagger \\ &= \mathbf{T} + (1/\alpha)\mathbf{v}^\dagger \mathbf{u}^\dagger\,, \end{aligned} \tag{6.7}$$

hence, condition (6.5) is equivalent to $(1/\alpha)(1 - \mathbf{a}^\top \mathbf{v}^\dagger \mathbf{u}^\dagger \mathbf{a}) \geq \mathbf{a}^\top \mathbf{T}\mathbf{a}$. Since $\mathbf{T}$ is PSD[1], it holds that $\mathbf{a}^\top \mathbf{T}\mathbf{a}$ is nonnegative. Clearly, if $1 - \mathbf{a}^\top \mathbf{v}^\dagger \mathbf{u}^\dagger \mathbf{a} \leq 0$, then there does not exist $\alpha > 0$ that satisfies (6.4). If $\mathbf{a}^\top \mathbf{T}\mathbf{a} \neq 0$, inequality (6.5) becomes

$$\alpha \leq (1 - \mathbf{a}^\top \mathbf{v}^\dagger \mathbf{u}^\dagger \mathbf{a})/(\mathbf{a}^\top \mathbf{T}\mathbf{a})\,. \tag{6.8}$$

[1] When α becomes infinite, the PSD matrix $\left(\mathbf{A} + \alpha \mathbf{b}\mathbf{b}^\top\right)^\dagger = \mathbf{T} + (1/\alpha)\mathbf{v}^\dagger \mathbf{u}^\dagger$ becomes $\mathbf{T}$. Thus, the matrix $\mathbf{T}$ must also be PSD.

Otherwise, if $\mathbf{a}^\top \mathbf{T}\mathbf{a} = 0$, the variable α can take any positive value in order to satisfy (6.4). We can easily include this case in (6.8) by setting $1/0 = +\infty$. Moreover, substituting (6.7) into (6.6) yields

$$\alpha(\mathbf{b}\mathbf{b}^\top \mathbf{T}\mathbf{a}) + (\mathbf{A}\mathbf{v}^\dagger \mathbf{u}^\dagger \mathbf{a})/\alpha + (\mathbf{a} - \mathbf{A}\mathbf{T}\mathbf{a} - \mathbf{b}\mathbf{b}^\top \mathbf{v}^\dagger \mathbf{u}^\dagger \mathbf{a}) = \mathbf{0}\,. \tag{6.9}$$

By definition of the pseudo-inverse (Golub and Van Loan, 1996), it holds that $(\mathbf{A}^\dagger \mathbf{A})^\top = \mathbf{A}^\dagger \mathbf{A}$, and $\mathbf{b}^\top \mathbf{v}^\dagger = 1$. Using $\mathbf{b}^\top \mathbf{k} = \beta - 1$, it is easy to see that

$$\mathbf{b}\mathbf{b}^\top \mathbf{T}\mathbf{a} = \mathbf{b}(\mathbf{b}^\top \mathbf{A}^\dagger \mathbf{a} - (\beta - 1)\mathbf{u}^\dagger \mathbf{a} - \mathbf{b}^\top \mathbf{A}^\dagger \mathbf{a} + (\beta - 1)\mathbf{u}^\dagger \mathbf{a}) = \mathbf{0}\,.$$

Using $\mathbf{A}\mathbf{v}^\dagger = \mathbf{0}$, we obtain $\mathbf{A}\mathbf{v}^\dagger \mathbf{u}^\dagger \mathbf{a} = \mathbf{0}$. Hence, Eq. (6.9) is equivalent to

$$\mathbf{a} - (\mathbf{A}\mathbf{A}^\dagger - \mathbf{A}\mathbf{k}\mathbf{u}^\dagger - \mathbf{A}\mathbf{v}^\dagger \mathbf{h} + (\beta - 1)\mathbf{A}\mathbf{v}^\dagger \mathbf{u}^\dagger)\mathbf{a} - \mathbf{b}\mathbf{u}^\dagger \mathbf{a} = \mathbf{0}\,. \tag{6.10}$$

Using $\mathbf{A}\mathbf{v}^\dagger = \mathbf{0}$, $\mathbf{b}^\top \mathbf{v}^\dagger = 1$, and $\mathbf{A}\mathbf{k} = \mathbf{b} - \mathbf{u}$, Eq. (6.10) becomes $\mathbf{a} - \mathbf{A}\mathbf{A}^\dagger \mathbf{a} + (\mathbf{b} - \mathbf{u})\mathbf{u}^\dagger \mathbf{a} - \mathbf{b}\mathbf{u}^\dagger \mathbf{a} = \mathbf{0}$, and results in $(\mathbf{I} - \mathbf{A}\mathbf{A}^\dagger - \mathbf{u}\mathbf{u}^\dagger)\mathbf{a} = \mathbf{0}$. Therefore, we conclude the first part of Theorem 6.2.

In the second case, the conditions $\mathbf{u} = \mathbf{0}$, (6.5) and (6.6) should be satisfied. Using Theorem 6.1, we have

$$(\mathbf{A} + \alpha \mathbf{b}\mathbf{b}^\top)^\dagger = \mathbf{A}^\dagger - 1/(1/\alpha - 1 + \beta)\mathbf{k}\mathbf{h}\,. \tag{6.11}$$

Substituting (6.11) into (6.5) leads to $1/\alpha - \mathbf{a}^\top(\mathbf{A}^\dagger - 1/(1/\alpha - 1 + \beta)\mathbf{k}\mathbf{h})\mathbf{a} \geq 0$. This results in

$$(1/\alpha)^2 - (1 - \beta + \mathbf{a}^\top \mathbf{A}^\dagger \mathbf{a})/\alpha - \mathbf{a}^\top(\mathbf{A}^\dagger(\beta - 1) - \mathbf{k}\mathbf{h})\mathbf{a} \geq 0\,,$$

or, equivalently,

$$(1/\alpha)^2 + (1/\alpha)B + C \geq 0\,. \tag{6.12}$$

Note that C is nonpositive because $\mathbf{A}^\dagger(\beta - 1) - \mathbf{k}\mathbf{h}$ is PSD[2]. Therefore, (6.12) always has a nonnegative root, $1/\alpha \geq (-B + \sqrt{B^2 - 4C})/2$. If $(-B + \sqrt{B^2 - 4C}) \neq 0$, then we have the following upper bound on α:

$$\alpha \leq 2/(-B + \sqrt{B^2 - 4C})\,. \tag{6.13}$$

Otherwise, if $(-B + \sqrt{B^2 - 4C}) = 0$, then α can take any positive value in order to satisfy (6.4). We can easily include this case in (6.13) by setting $1/0 = +\infty$. Next, we prove that $(\mathbf{A} + \alpha \mathbf{b}\mathbf{b}^\top)(\mathbf{A} + \alpha \mathbf{b}\mathbf{b}^\top)^\dagger = \mathbf{A}\mathbf{A}^\dagger$. Since $\mathbf{u} = \mathbf{0}$, it holds that

[2] When α becomes infinite, the PSD matrix $\left(\mathbf{A} + \alpha \mathbf{b}\mathbf{b}^\top\right)^\dagger = \mathbf{A}^\dagger - 1/(1/\alpha - 1 + \beta)\mathbf{k}\mathbf{h}$ becomes $\mathbf{A}^\dagger - (1/(\beta - 1))\mathbf{k}\mathbf{h}$. Thus, the matrix $\mathbf{A}^\dagger(\beta - 1) - \mathbf{k}\mathbf{h}$ must also be PSD.

$\mathbf{A}\mathbf{A}^\dagger\mathbf{b} = \mathbf{b}$. Using (6.11), one obtains

$$\begin{aligned}
&(\mathbf{A} + \alpha\mathbf{b}\mathbf{b}^\top)(\mathbf{A} + \alpha\mathbf{b}\mathbf{b}^\top)^\dagger \\
&= (\mathbf{A} + \alpha\mathbf{b}\mathbf{b}^\top)(\mathbf{A}^\dagger - 1/(1/\alpha - 1 + \boldsymbol{\beta})\mathbf{k}\mathbf{h}) \\
&= \mathbf{A}\mathbf{A}^\dagger - 1/(1/\alpha - 1 + \boldsymbol{\beta})\mathbf{A}\mathbf{k}\mathbf{h} + \alpha\mathbf{b}\mathbf{b}^\top\mathbf{A}^\dagger - 1/(1/\alpha - 1 + \boldsymbol{\beta})\alpha\mathbf{b}\mathbf{b}^\top\mathbf{k}\mathbf{h} \\
&= \mathbf{A}\mathbf{A}^\dagger + 1/(1/\alpha - 1 + \boldsymbol{\beta})(-\mathbf{A}\mathbf{A}^\dagger\mathbf{b}\mathbf{b}^\top\mathbf{A}^\dagger \\
&\quad + \mathbf{b}\mathbf{b}^\top\mathbf{A}^\dagger + \alpha\mathbf{b}\mathbf{b}^\top\mathbf{A}^\dagger\mathbf{b}\mathbf{b}^\top\mathbf{A}^\dagger - \alpha\mathbf{b}\mathbf{b}^\top\mathbf{A}^\dagger\mathbf{b}\mathbf{b}^\top\mathbf{A}^\dagger) \\
&= \mathbf{A}\mathbf{A}^\dagger\,. \qquad (6.14)
\end{aligned}$$

Substituting (6.14) into (6.6) leads to $(\mathbf{I} - \mathbf{A}\mathbf{A}^\dagger)\mathbf{a} = \mathbf{0}$, and we conclude the second part of Theorem 6.2. □

Given the matrix $\mathbf{M}_t$ and its pseudo-inverse $\mathbf{M}_t^\dagger$, using Theorem 6.2 we can analytically find an upper bound α_t such that for any step size $\eta_t^* \in [0, \alpha_t]$, the matrix

$$\mathbf{M}_t - \eta_t^*\nabla f_i(\mathbf{M}_t) = \mathbf{M}_t - \eta_t^*\left((\mathbf{x}_i - \mathbf{x}_i^+)(\mathbf{x}_i - \mathbf{x}_i^+)^\top - (\mathbf{x}_i - \mathbf{x}_i^-)(\mathbf{x}_i - \mathbf{x}_i^-)^\top\right)$$

remains within the PSD cone. Evaluating efficiently the upper bound of η_t^* requires that we maintain the updated pseudo-inverse matrix $\mathbf{M}^\dagger$. Due to the simplicity of updating $\mathbf{M}$ (since the update makes use of only two rank-one matrices in each iteration), we can derive the update of the pseudo-matrix $\mathbf{M}^\dagger$ from Theorem 6.1 in the following way. First, we update $\mathbf{M}_t$ and $\mathbf{M}_t^\dagger$ by adding the matrix $(\mathbf{x}_i - \mathbf{x}_i^-)(\mathbf{x}_i - \mathbf{x}_i^-)^\top$ as follows:

$$\begin{aligned}
\mathbf{M}_{t+1/3} &= (\mathbf{M}_t + \eta_t^*(\mathbf{x}_i - \mathbf{x}_i^-)(\mathbf{x}_i - \mathbf{x}_i^-)^\top)\,, \\
\mathbf{M}_{t+1/3}^\dagger &= (\mathbf{M}_t + \eta_t^*(\mathbf{x}_i - \mathbf{x}_i^-)(\mathbf{x}_i - \mathbf{x}_i^-)^\top)^\dagger\,.
\end{aligned} \qquad (6.15)$$

Second, we use $\mathbf{M}_{t+1/3}$ and $\mathbf{M}_{t+1/3}^\dagger$ to update $\mathbf{M}_t$ and $\mathbf{M}_t^\dagger$ by adding the matrix $(\mathbf{x}_i - \mathbf{x}_i^+)(\mathbf{x}_i - \mathbf{x}_i^+)^\top$ as follows:

$$\begin{aligned}
\mathbf{M}_{t+2/3} &= (\mathbf{M}_{t+1/3} - \eta_t^*(\mathbf{x}_i - \mathbf{x}_i^+)(\mathbf{x}_i - \mathbf{x}_i^+)^\top)\,, \\
\mathbf{M}_{t+2/3}^\dagger &= (\mathbf{M}_{t+1/3} - \eta_t^*(\mathbf{x}_i - \mathbf{x}_i^+)(\mathbf{x}_i - \mathbf{x}_i^+)^\top)^\dagger\,.
\end{aligned} \qquad (6.16)$$

Using Theorem 6.1, both matrices $\mathbf{M}_{t+1/3}^\dagger$ and $\mathbf{M}_{t+2/3}^\dagger$ are efficiently computed with a time complexity of $O(D^2)$. Finally, we truncate the solution to satisfy the trace-bound constraint:

$$\begin{aligned}
\mathbf{M}_{t+1} &= \min(B/\mathrm{tr}(\mathbf{M}_{t+2/3}), 1)\mathbf{M}_{t+2/3}\,, \\
\mathbf{M}_{t+1}^\dagger &= \max(\mathrm{tr}(\mathbf{M}_{t+2/3})/B, 1)\mathbf{M}_{t+2/3}^\dagger\,.
\end{aligned}$$

Note that all of these operations can be analytically computed with a time complex-

ity of $O(D^2)$, which is faster than those methods that use a numerical approximation procedure to find the smallest eigenvalue and corresponding eigenvector, such as POLA (Shalev-Shwartz et al., 2004), DML-eig (Ying and Li, 2012), SDPMetric (Shen et al., 2010), and BoostMetric (Shen et al., 2012). The pseudo-code of LMDML-A is given in Algorithm 4.

Algorithm 4 LMDML-A

Input: $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, B, c, T for SGD
Output: $\mathbf{M}_T$

Set $\mathbf{M}_1 \leftarrow \mathbf{I}$ and $\mathbf{M}_1^\dagger \leftarrow \mathbf{I}$
for $t \leftarrow 1, \ldots, T-1$ **do**
  Choose $i \in \{1, \ldots, n\}$ uniformly at random
  Find $\mathbf{x}_i^+$ and $\mathbf{x}_i^-$ as in (6.1)
  if $d^2_{\mathbf{M}_t}(\mathbf{x}_i, \mathbf{x}_i^+) + 1 > d^2_{\mathbf{M}_t}(\mathbf{x}_i, \mathbf{x}_i^-)$ **then**
    Find upper bound α_t to make $\mathbf{M}_t - \alpha_t \nabla f_i(\mathbf{M}_t) \succcurlyeq 0$
    Select $0 \leq \eta_t^* \leq \min(\alpha_t, c/\sqrt{t})$
    Compute $\mathbf{M}_{t+1/3}$ and $\mathbf{M}_{t+1/3}^\dagger$ as in (6.15)
    Compute $\mathbf{M}_{t+2/3}$ and $\mathbf{M}_{t+2/3}^\dagger$ as in (6.16)
    Set $\mathbf{M}_{t+1} \leftarrow \min(B/\mathrm{tr}(\mathbf{M}_{t+2/3}), 1)\mathbf{M}_{t+2/3}$
    Set $\mathbf{M}_{t+1}^\dagger \leftarrow \max(\mathrm{tr}(\mathbf{M}_{t+2/3})/B, 1)\mathbf{M}_{t+2/3}^\dagger$
  end if
end for

6.4.2. Convergence analysis

The convergence of SGD has been theoretically studied for decades (see for instance J. Kushner and Yin, 2003, and the references therein). However, these classical convergence bounds require some non-trivial assumptions on the objective function, such as strong convexity or smoothness. Unfortunately, this is not the case for our objective function in problem (6.3). To address this issue, we use a more general convergence result of SGD for non-smooth optimization developed by Shamir and Zhang (2013).

It is well known that the convergence of SGD is dependent on the value of the step size. However, the step size in our method is only upper-bounded using the Schur complement, which involves many constants that cannot be evaluated exactly. As a consequence, the result presented in this section may only provide a conservative estimate of what can be achieved by our method. In practice, we observe that LMDML-A converges faster for a larger step size (but still relatively small to keep the Mahalanobis matrix within the PSD cone, i.e. $\eta_t^* \leq \min(\alpha_t, c/\sqrt{t})$). The following theorem shows that the last iteration of SGD converges to an optimal solution of problem (6.3) with a rate of $O(\log(T)/\sqrt{T})$.

Theorem 6.3. *Let $\mathbf{M}^* \in \mathbb{S}_+^D$ be the optimal solution for the objective function f in (6.3). Let $\mathbf{M}_1, \ldots, \mathbf{M}_T$ be a sequence of matrices such that $\mathbf{M}_1 \in \mathbb{S}_+^D$ and for $t > 0$, $\mathbf{M}_{t+1} = \Pi_{\mathbb{S}_+^D}(\mathbf{M}_t - \eta_t \nabla f_i(\mathbf{M}_t))$, where $\eta_t = c/\sqrt{t}$ and $c > 0$ is a constant. Suppose that for all $t > 0$ and $\mathbf{x}_i \in \mathcal{X}$, it holds that $\|\nabla f_i(\mathbf{M}_t)\|_F \leq G$ and $\mathrm{tr}(\mathbf{M}_t) \leq B$ for some constants G and B. Then, for any $T > 1$, we have*

$$\mathbb{E}\big[f(\mathbf{M}_T) - f(\mathbf{M}^*)\big] \leq \left(\frac{4B^2}{c} + cG^2\right) \frac{2 + \log(T)}{\sqrt{T}}\,.$$

Proof. The result follows directly from (Shamir and Zhang, 2013, Theorem 2) and the corresponding proof. Since $\mathbf{M}_t \in \mathbb{S}_+^D$ and $\mathrm{tr}(\mathbf{M}_t) \leq B$ for all $t > 0$, it holds for any $t, t' > 0$ that

$$\begin{aligned}
\|\mathbf{M}_t - \mathbf{M}_{t'}\|_F &\leq \|\mathbf{M}_t\|_F + \|\mathbf{M}_{t'}\|_F \\
&= \sqrt{\mathrm{tr}(\mathbf{M}_t^\top \mathbf{M}_t)} + \sqrt{\mathrm{tr}(\mathbf{M}_{t'}^\top \mathbf{M}_{t'})} \\
&\leq \mathrm{tr}(\mathbf{M}_t) + \mathrm{tr}(\mathbf{M}_{t'}) \\
&\leq 2B\,,
\end{aligned}$$

where the second inequality holds because $\mathrm{tr}(\mathbf{AB}) \leq \mathrm{tr}(\mathbf{A})\mathrm{tr}(\mathbf{B})$ for all $\mathbf{A}, \mathbf{B} \in \mathbb{S}_+^D$. This concludes the proof. □

Empirically, we have found that the convergence of LMDML-A is slightly slower than that of the standard SGD method (see Section 6.5).

6.4.3. Computational complexity

In this subsection, we analyze the time complexity of LMDML-A. Let n denote the number of training examples, let D denote the dimensionality, and let k denote the number of target neighbors for each training example. We first consider the search for the target neighbors. Using linear search we can easily perform this computation in $O(kn^2D)$. In order to reduce the complexity of searching for the nearest neighbors for large-scale data sets, we can use sophisticated data structures, such as Cover Tree (Beygelzimer et al., 2006), Ball Tree (Omohundro, 1989), and k-d-Tree (Moore, 1991). Next, we analyze the time complexity in each iteration. Recomputation of $\mathbf{x}_i^+$ and $\mathbf{x}_i^-$ as in (6.1) scales as $O(kD^2 + nD^2)$ due to the quadratic time complexity in the dimensionality of computing the Mahalanobis distance. Unfortunately, this computation can make the algorithm impractical on high-dimensional data sets. To reduce this computational burden, we observe that

$$d_{\mathbf{M}}^2(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{M} \mathbf{x}_i + \mathbf{x}_j^\top \mathbf{M} \mathbf{x}_j) - (2\mathbf{u}_i^\top \mathbf{x}_j)\,,$$

where $\mathbf{u}_i = \mathbf{M}\mathbf{x}_i$. For each training example $\mathbf{x}_i$, if we keep track of the value of $\mathbf{x}_i^\top \mathbf{M}\mathbf{x}_i$, then we can reduce the cost of searching for $\mathbf{x}_i^+$ and $\mathbf{x}_i^-$ to $O(kD+nD+D^2)$, since the first term $(\mathbf{x}_i^\top \mathbf{M}\mathbf{x}_i + \mathbf{x}_j^\top \mathbf{M}\mathbf{x}_j)$ can be computed in constant time, the second term $(2\mathbf{u}_i^\top \mathbf{x}_j)$ can be computed in $O(D)$, and the computation of $\mathbf{u}_i$ scales as $O(D^2)$. On the other hand, $\mathbf{M}$ is always updated by adding a rank-one matrix in the following form $\mathbf{M} \leftarrow \mathbf{M} + \alpha \mathbf{u}\mathbf{u}^\top$, where $\alpha \in \mathbb{R}$. Consequently, the value of $\mathbf{x}_i^\top \mathbf{M}\mathbf{x}_i$ can be efficiently updated in $O(D)$ as

$$\begin{aligned}\mathbf{x}_i^\top (\mathbf{M} + \alpha \mathbf{u}\mathbf{u}^\top)\mathbf{x}_i &= \mathbf{x}_i^\top \mathbf{M}\mathbf{x}_i + \alpha(\mathbf{x}_i^\top \mathbf{u})(\mathbf{x}_i^\top \mathbf{u}) \\ &= \mathbf{x}_i^\top \mathbf{M}\mathbf{x}_i + \alpha(\mathbf{x}_i^\top \mathbf{u})^2 \,.\end{aligned}$$

As mentioned in Subsection 6.4.1, LMDML-A requires to update the pseudo-inverse of the Mahalanobis matrix and to find an upper bound on the step size, which scales as $O(D^2)$. Summarizing, the overall time complexity of each iteration for LMDML-A is $O(D^2 + nD)$. The time complexity of one pass over all training examples is $O(nD^2 + n^2D)$.

Although one can notice some improvement, the preceding complexity is still impractical for very large data sets. The bottleneck of LMDML-A is mainly due to the search for $\mathbf{x}_i^-$, which, in theory, is of linear time complexity $O(nD^2)$ in the number of training examples. To overcome this limitation, we observe that $\mathbf{x}_i^-$ usually lies nearby the local neighborhood of $\mathbf{x}_i$, which means that it is not always necessary to search for the whole set containing examples of different classes. Consequently, we can approximate $\mathcal{M}(\mathbf{x}_i)$ with a set of m nearest neighbors of $\mathbf{x}_i$ with different class labels, which can be precomputed using the Euclidean distance metric. Note that this set does not change during the execution of the algorithm. By doing so, the time complexity of searching for $\mathbf{x}_i^-$ is reduced to $O(mD^2)$ and the time complexity of one pass over all training examples is $O(nmD^2 + nkD^2)$.

6.5. Experiments

In this section, we evaluate the effectiveness and efficiency of the proposed algorithm by conducting an extensive set of experiments on standard benchmark data sets. The performance of LMDML-A is compared with other state-of-the-art distance metric learning algorithms in the context of k-NN classification. We will demonstrate that our algorithm is fast and scalable, making it more suitable for large-scale applications. First, we evaluate the effectiveness of the proposed algorithm on fourteen data sets of varying size and difficulty in Subsection 6.5.1. Second, we empirically verify the convergence rate of the proposed algorithm in Subsection 6.5.2. Finally, we compare the performance of LMDML-A with different learning algorithms on large-scale data sets in Subsection 6.5.3. All features are normalized (to have zero mean and unit standard deviation) over the training data

to avoid the influence carried by the scale of each feature for the distance metric or similarity function.

6.5.1. Experiments on the KEEL data sets

In this subsection, we aim to demonstrate the effectiveness of the principle of margin maximization, which inspired our approach. For this purpose, several experiments are carried out using publicly available data sets to compare our approach with five other distance metric learning approaches.

Competing approaches

We will compare the proposed **LMDML-A** algorithm with different distance metric learning algorithms, including the baseline **Euclidean** distance metric, **POLA** (Shalev-Shwartz et al., 2004), **ITML** (Davis et al., 2007), **LMNN** (Weinberger and Saul, 2009), **DML-eig** (Ying and Li, 2012), and **DMLMJ** (Nguyen et al., 2017c). These are the most prominent methods in distance metric learning. LMDML-A is closely related to LMNN in the sense that both methods are particularly designed to maximize the margin of k-NN classification. Other methods, including POLA, ITML, DML-eig, and DMLMJ, use a margin criterion over pairwise constraints, i.e. distances between examples of the same class are smaller than distances between those of different classes.

To get the best results for all algorithms, the hyper-parameters are tuned via cross-validation. For ITML, we set the maximum number of iterations to 10^5 and tune the slack parameter γ considering as set of values $\{10^{-3}, \ldots, 10^3\}$. Based on the authors' recommendation (Davis et al., 2007), the lower bound l and upper bound u are set to be the 5-th and 95-th percentiles, respectively, of the distribution of all distances between training examples. For DML-eig, we set the maximum number of iterations to 1,000. Since POLA, ITML, DML-eig, and DMLMJ are based on pairwise constraints, for a fair comparison, we use the same pairwise constraints for POLA, ITML, DML-eig, and DMLMJ. These constraints are generated by pairing each training example with its k nearest neighbors of the same class and its k nearest neighbors of different classes. For LMNN, we set the maximum number of iterations to 1,000 and tune the trade-off parameter μ considering as set of values $\{0.125, 0.25, 0.5\}$. For LMDML-A, we tune the trace-bound parameter B considering as set of values $\{0.1, 1, 10, 100\}$. We set the maximum number of epochs to 10 for both POLA and LMDML-A. Empirically, we find that a value of 1 as the initial constant of the step size c for LMDML-A can be applied to obtain good results. The source codes in Matlab and C-mex of these approaches are available online from the corresponding

authors' websites[3]. The source codes of LMDML-A can also be downloaded from http://users.ugent.be/~bacnguye/LMDML-A.v1.0.zip.

Data sets and experimental setup

We use fourteen data sets from KEEL (see Table A.1 for a brief description). In particular, a 10-fold cross-validation is employed to estimate the classification accuracy. All partitions of the training and test sets are collected by stratified sampling from all classes. The classification accuracy and the training time are obtained by computing the average over ten runs. Following (Weinberger and Saul, 2009; Ying and Li, 2012), for all data sets, we set $k = 3$ to perform k-NN classification.

Results and discussion

Table 6.1 summarizes the classification accuracy and training time of the competing distance metric learning algorithms on each data set. Note that the training time takes into account the time for tuning the hyper-parameters. On each data set, we rank the competing algorithms based on their classification accuracy, i.e., we assign rank 1 to the algorithm with the highest accuracy, rank 2 to the algorithm with the second highest accuracy, and so on. The average rank based on classification accuracy is shown in the penultimate row of Table 6.1. Based on the experimental results, we can draw some conclusions as follows:

(i) LMDML-A consistently improves the performance of k-NN classification using the Euclidean distance metric on most data sets. In general, our margin-based approach (i.e. LMDML) performs better than other distance metric learning approaches (i.e., POLA, ITML, DML-eig, and LMNN). Both approaches LMNN and ITML perform equally well.

(ii) LMDML-A performs slightly better than POLA and the recent method DMLMJ. Since DMLMJ only involves the computation of solving a generalized eigenvalue decomposition problem, it is very efficient on low-dimensional data sets. In particular, LMDML-A is considerably faster than DMLMJ on large-scale data sets (e.g., *letter* and *magic*).

(iii) LMDML-A is the fastest algorithm on most data sets. In general, POLA, ITML and DML-eig run faster than LMNN because they do not have to perform the full eigen-decomposition at each iteration. Additionally, LMDML-A is significantly more efficient than LMNN on large data sets (e.g., *letter,*

[3] ITML: http://www.cs.utexas.edu/~pjain/itml/
LMNN: http://www.cse.wustl.edu/~kilian/code/code.html
DML-eig: http://empslocal.ex.ac.uk/people/staff/yy267/software.html
DMLMJ: http://users.ugent.be/~bacnguye/DMLMJ.zip

magic, and *ring*), which highlights the advantages of using SGD methods compared to batch gradient descent methods on large data sets.

Table 6.1: Classification accuracy (standard deviation) and training time on the KEEL data sets of the competing algorithms. The best result is highlighted in boldface.

Id	Accuracy							Training time (in seconds)					
	Euclidean	POLA	ITML	LMNN	DML-eig	DMLMJ	LMDML	POLA	ITML	LMNN	DML-eig	DMLMJ	LMDML
APP	83.18 (9.2)	85.00 (6.2)	84.09 (8.5)	84.00 (4.4)	85.00 (8.7)	84.09 (8.5)	**86.91 (6.4)**	1.22	28.06	13.00	0.28	**0.02**	**0.02**
BAL	83.68 (2.2)	90.47 (3.8)	90.51 (3.7)	87.04 (3.1)	85.42 (5.7)	90.24 (3.9)	**90.70 (4.5)**	5.39	37.44	18.01	0.49	**0.17**	0.21
BUP	65.49 (3.4)	65.1 (10.3)	65.47 (6.7)	67.20 (6.1)	64.58 (6.0)	64.8 (11.0)	**67.49 (5.7)**	4.11	32.37	20.43	0.19	**0.08**	0.10
IRI	94.67 (5.3)	96.00 (4.7)	**98.00 (3.2)**	96.00 (4.7)	94.67 (5.3)	96.67 (4.7)	95.33 (4.5)	1.11	30.93	12.52	0.18	**0.03**	**0.03**
LET	94.98 (0.5)	**97.55 (0.4)**	97.23 (0.3)	96.48 (0.5)	87.70 (0.6)	97.51 (0.4)	96.83 (0.5)	223.2	138.8	364.2	872.9	396.4	**58.42**
MAG	83.66 (0.5)	84.15 (0.9)	83.71 (0.8)	83.72 (0.6)	82.91 (1.2)	84.06 (0.7)	**84.93 (0.5)**	267.5	132.6	1007	792.5	238.3	**46.58**
MON	95.85 (3.6)	**100.0 (0.0)**	99.53 (1.0)	97.91 (3.0)	**100.0 (0.0)**	99.07 (2.2)	98.37 (3.6)	3.67	35.75	12.49	0.38	**0.12**	0.14
MOV	80.00 (7.0)	84.17 (5.2)	81.94 (7.1)	83.61 (5.8)	72.50 (9.5)	81.39 (5.4)	**84.72 (5.7)**	58.83	180.9	37.52	2.39	**1.03**	1.26
OPT	97.94 (0.6)	**99.02 (0.4)**	98.35 (0.5)	98.90 (0.3)	97.83 (0.7)	98.83 (0.6)	98.74 (0.6)	166.5	199.4	171.1	58.61	118.8	**26.36**
RIN	71.81 (1.7)	76.22 (1.6)	80.89 (1.0)	71.61 (1.8)	86.38 (1.2)	**87.36 (1.0)**	74.07 (1.6)	145.9	89.16	324.1	101.7	63.14	**15.41**
SEG	95.45 (1.0)	96.36 (1.2)	**97.45 (1.7)**	96.67 (1.1)	95.06 (1.4)	96.80 (1.2)	96.75 (1.2)	24.96	57.37	61.15	6.05	5.97	**2.40**
WDB	97.01 (1.7)	97.52 (1.4)	97.01 (2.6)	96.66 (2.1)	95.61 (2.1)	96.48 (1.4)	**97.54 (2.5)**	12.50	54.10	17.95	1.36	0.64	**0.44**
WIN	96.01 (3.9)	97.71 (4.1)	98.30 (2.7)	96.60 (4.8)	97.19 (4.0)	**98.89 (2.3)**	97.75 (2.9)	1.39	33.80	12.05	0.38	**0.05**	0.07
WIS	**96.78 (1.7)**	96.34 (1.2)	96.34 (2.0)	96.63 (2.1)	95.90 (1.7)	95.90 (0.9)	96.63 (1.8)	5.01	40.40	17.16	0.57	0.33	**0.32**
	5.57	2.93	3.25	4.36	5.64	3.57	**2.68**	921.43	1,091.2	2,089.6	1,838.15	825.1	**151.76**
	Average rank							Total training time					

6.5.2. Evaluation of the convergence

In this subsection, we empirically compare the convergence rate of LMDML-A to that of the standard SGD method (LMDML-S) that solves problem (6.3). Note that LMDML-S only involves the computation of the smallest eigenvalue and corresponding eigenvector at each iteration (see Subsection 6.4.1). As an illustration, we only perform experiments on the *sonar* data set used in Subsection 6.5.1. LMDML-A is configured with the same settings as in the previous experiments. Figure 6.2 shows the convergence rate versus the number of epochs (a full pass through the training set). On the left side of Fig. 6.2, we show the objective function value versus the number of epochs. On the right side of Fig. 6.2, we show the training accuracy based on the 3-NN classifier versus the number of epochs. We observe that our algorithm converges after a few number of epochs. Since the objective function in problem (6.3) is convex, LMDML-A should indeed converge to the optimal objective function value. Once a certain number of epochs is reached, the training accuracy remains more or less the same. As expected, both algorithms LMDML-A and LMDML-S obtain similar results. The convergence of LMDML-A is only slightly slower than that of LMDML-S.

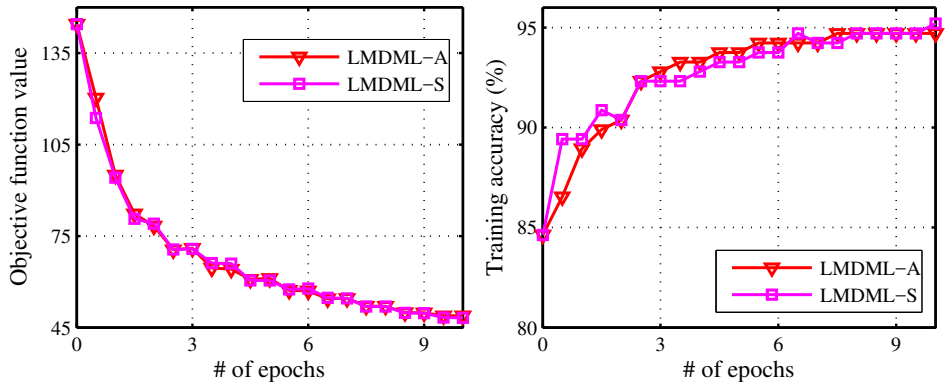


Figure 6.2: Performance illustration of LMDML-A and LMDML-S on the *sonar* data set. Left figure: objective function value vs. number of epochs. Right figure: training accuracy (%) vs. number of epochs.

6.5.3. Experiments on large-scale data sets

In this subsection, we study the behavior of the proposed algorithm in large-scale settings. We compare **LMDML-A** with the baseline **Euclidean** distance metric, **POLA**, **LMDML-S** and **OASIS** (Chechik et al., 2010), a bilinear similarity learning algorithm. OASIS aims to learn a bilinear similarity function over sparse representations for improving image retrieval performance. OASIS is a current

state-of-the-art similarity learning algorithm in large-scale settings. The source codes in Matlab and C-mex of this approach are available online from the corresponding authors' website[4]. For OASIS, only one projection onto the PSD cone is applied. Cross-validation is used for setting the following hyper-parameters: the aggressiveness parameter $C \in \{10^{-9}, \ldots, 10^{2}\}$ for OASIS; and the trace-bound parameter $B \in \{0.1, 1, 10, 100\}$ for LMDML-A and LMDML-S. To get the best performance, we set the maximum number of iterations to 10^6 for OASIS and 10^5 for POLA, LMDML-S, and LMDML-A. OASIS often requires a large number of iterations in order to achieve a good performance. This is due to the fact that the matrix learned by OASIS is not guaranteed to be PSD or even symmetric after each iteration. In our experiments, only one projection onto the PSD cone is applied at the end of OASIS, and as a result, it may lead to a suboptimal solution which is far away from the optimal one.

We carry out an experiment on several publicly available large-scale data sets, including the **Isolet** (Cole and Fanty, 1990), **Connect-4**, **Poker**, **Sensit**, and **Protein** (Chang and Lin, 2011) data sets. Table 6.2 summarizes the information of these data sets. The Isolet[5] data set contains 7,797 examples with 617 features collected from 150 different speakers. They pronounced the name of each letter in the English alphabet twice. The task is to recognize what letter was been uttered. The speakers are formed into sets of 30 speakers each, referred as Isolet1, Isolet2, Isolet3, Isolet4, and Isolet5. The first four subsets are used for training and validation to tune the hyper-parameters. The last subset is used for testing. This data set was used in several distance metric learning studies (Parameswaran and Weinberger, 2010; Qian et al., 2015a; Nguyen et al., 2016). The remaining data sets, including Protein, Connect-4, Poker, and Sensit, were downloaded from LIBSVM[6]. These data sets are challenging because they contain a very large number of training examples with the number of features varying from 10 to 357. For instance, the Poker data set contains up to one million examples, while the Protein data set contains 24,387 examples with 357 features. All training and test sets are predefined, except for the Connect-4 data set where 70% of the data is randomly selected for training and the remaining 30% is used for testing. To make the comparison of the competing algorithms as fair as possible, for each training example, we use 3 nearest neighbors of the same class and 5 nearest neighbors of different classes to generate 15 triplet constraints and 8 pairwise constraints. Due to the very large number of training examples, we also limit the number of miss examples to 5 for both LMDML-S and LMDML-A in order to reduce the time complexity (see discussion in Subsection 6.4.3). All the experiments are repeated five times.

Table 6.3 shows the classification accuracy of the competing algorithms in the

[4] OASIS: http://ai.stanford.edu/~gal/Research/OASIS/
[5] Available at: https://archive.ics.uci.edu/ml/datasets/ISOLET
[6] Available at: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets

Table 6.2: Description of large-scale data sets used in our experiment.

#	Data set	Features	Classes	Training examples	Test examples
1.	Connect-4	126	3	54,045	13,512
2.	Isolet	617	26	6,238	1,559
3.	Poker	10	10	1,000,000	25,010
4.	Protein	357	3	17,766	6,621
5.	Sensit	100	3	78,823	19,705

Table 6.3: Classification accuracy (standard deviation) on large-scale data sets of the competing algorithms. The best result is highlighted in boldface.

#	Euclidean	POLA	OASIS	LMDML-S	LMDML-A
1.	69.17 (0.00)	**78.81 (0.30)**	74.06 (0.01)	78.20 (0.54)	78.21 (0.34)
2.	90.38 (0.00)	92.62 (0.01)	94.42 (0.26)	94.46 (0.38)	**94.47 (0.25)**
3.	61.40 (0.00)	**63.07 (1.26)**	39.07 (0.09)	62.90 (0.99)	62.80 (0.80)
4.	51.87 (0.00)	65.00 (0.03)	**65.97 (0.16)**	64.84 (0.53)	64.74 (0.43)
5.	81.12 (0.00)	81.51 (0.12)	69.11 (0.29)	**81.60 (0.01)**	81.53 (0.02)

context of 3-NN classification. As can be seen, LMDML-A consistently outperforms the baseline Euclidean distance metric, and it maintains a superior or equal performance compared to other algorithms on all data sets. OASIS yields very poor results on the Poker and Sensit data sets. We also observe that LMDML-A and LMDML-S obtain similar results since both algorithms are developed on the basis of the same problem formulation.

Table 6.4 reports the final rank of the Mahalanobis matrix learned by the competing algorithms. The results show that LMDML-A can capture the underlying low-dimensional structure of data, thus reducing the risk of overfitting. These results also confirm the validity of using the trace norm to impose the low-rank constraint on the Mahalanobis matrix.

We further compare the average training time of the competing algorithms in Table 6.5. The training time takes into account the time for tuning the hyper-parameters. Our algorithm is significantly faster than the competing algorithms, especially on the Isolet and the Sensit data sets. When the dimensionality is low (e.g. the Poker data set), there is no significant difference in training time between LMDML-A, LMDML-S, and POLA. Note that the training time of OASIS highly depends on the sparseness of the data sets. This may explain why LMDML-A runs significantly faster than OASIS. Although LMDML-S and LMDML-A have the same time complexity, LMDML-A is still faster than LMDML-S due to the efficient way of keeping the solution within the PSD cone.

Table 6.4: Average rank of the Mahalanobis matrix learned from large-scale data sets by the competing algorithms.

#	Euclidean	POLA	OASIS	LMDML-S	LMDML-A
1.	126.00	82.00	120.00	90.80	87.00
2.	617.00	462.00	607.00	607.80	608.20
3.	10.00	10.00	6.00	10.00	9.00
4.	357.00	356.00	250.00	348.00	333.00
5.	100.00	98.00	100.00	91.00	89.00

Table 6.5: Training time (in seconds) of the competing algorithms on large-scale data sets. The best result is highlighted in boldface.

#	POLA	OASIS	LMDML-S	LMDML-A
1.	1,012.36	1,989.16	1,844.40	**200.66**
2.	11,597.17	8,979.97	13,622.83	**1,816.97**
3.	462.21	1,809.79	430.92	**421.50**
4.	10,755.13	6,842.63	11,641.36	**1,363.93**
5.	953.56	1,863.74	1,332.32	**152.91**

6.6. Discussion and conclusion

We conclude this work by summarizing our main contributions in this chapter and discussing some related work that motivated our approach. The main contribution of this chapter is the proposal of a novel distance metric learning approach for k-NN classification. The intuition behind our approach is based on the principle of margin maximization. In order to make the proposed approach practical on large-scale data sets, we have developed an efficient algorithm to reduce the expensive cost of projections onto the PSD cone. The experimental results have demonstrated that our algorithm is capable of handling large-scale data sets, when the number of examples is large or the dimensionality is high.

Our approach shares the same goals of local learning as LMNN (Weinberger and Saul, 2009) and NCA (Goldberger et al., 2005). Local learning can preserve the discriminative information contained in the neighborhood and allows to capture the local structure of the data. Our approach is based on local learning and follows a similar idea as the margin-maximization principle of SVMs (Cortes and Vapnik, 1995), AdaBoost (Schapire et al., 1997), and LMNN (Weinberger and Saul, 2009).

On the other hand, many distance metric learning approaches based on feature extraction, such as PCA (Jolliffe, 2005), LDA (Fisher, 1936), RCA (Bar-Hillel et al.,

2005), and DMLMJ, require implicit assumptions about the distribution of the data. When these assumptions do not hold, these approaches may extract irrelevant features that are not useful for k-NN classification. In contrast to these approaches, our approach requires only information about the neighborhood of each training example, while no assumption about the distribution of the data is made. For this reason, our approach can preserve the strengths of k-NN classification, which is a nonparametric method, and makes no assumptions about the distribution of the data. Not surprisingly, our approach can boost the performance of k-NN classification.

Our proposed approach is closely related to LMNN (Weinberger and Saul, 2009) in the sense that both approaches solve a convex optimization problem, with the goal of making the k nearest neighbors of each training examples share the same class label, while pushing away examples with different class labels. Unlike LMNN, our approach enforces the low-rank constraint on the Mahalanobis matrix to reduce the risk of overfitting in high-dimensional settings. Due to the simplicity of our margin definition, we can significantly reduce the number of constraints in the problem formulation, and as a consequence, our algorithm is orders of magnitude faster than LMNN.

7 Distance metric learning based on difference of convex functions programming

In this chapter, we develop a supervised distance metric learning method that aims to improve the performance of nearest-neighbor classification. Our method is inspired by the large-margin principle, resulting in an objective function based on a sum of margin violations to be minimized. Due to the use of the ramp loss function, the corresponding objective function is nonconvex, making it more challenging. To overcome this limitation, we formulate our distance metric learning problem as an instance of difference of convex functions (DC) programming. This allows us to design a more robust method than when using standard optimization techniques.

The material of this chapter is based on the following publication:
Nguyen, B. and De Baets, B. (2018a). An approach to supervised distance metric learning based on difference of convex functions programming. *Pattern Recognition*, 81:562–574

7.1. Motivation

Mahalanobis distance metric learning can be formulated within a convex optimization framework, which enjoys significant advantages in that the convexity guarantees to reach the global optimum and is not sensitive to initial conditions. A large number of optimization methods have been proposed to deal with convex optimization problems (Boyd and Vandenberghe, 2004). In particular, convex distance metric learning methods are often cast as solving semidefinite programs, therefore, standard semidefinite programming solvers can be used. In order to make the problem more tractable in large-scale settings, Weinberger and Saul (2009) developed an efficient subgradient descent method based on the active set techniques. Davis et al. (2007) introduced an iterative Bregman projection method to avoid the projection of the Mahalanobis matrix onto the cone of symmetric positive semidefinite (PSD) matrices. Shen et al. (2012) proposed a boosting-based method that learns a linear combination of trace-one rank-one matrices. Recently, Atzmon et al. (2015) suggested an efficient solver based on the block-coordinate descent method to avoid the projection and computation of full gradients. Other methods such as the Frank-Wolfe (Ying and Li, 2012) and the projected gradient descent (Xing et al., 2002) methods have also been employed in the context of

distance metric learning.

Convex optimization has become very popular in the pattern recognition community over the last few decades, because of its empirical performance and because it facilitates a deeper mathematical analysis. Unfortunately, in many practical settings, convexity is not always guaranteed, and one has to resort to nonconvex optimization methods (Collobert et al., 2006b). Various researchers (Mason et al., 2000; Liu et al., 2005; Fujiwara et al., 2017) have argued that using nonconvex loss functions to approximate the misclassification rate can yield a better performance than using convex loss alternatives such as the hinge loss and the exponential loss. Recent research in this direction has provided a number of nonconvex functions in order to alleviate the limitation of convex functions. Shen et al. (2003) and Liu and Shen (2006) proposed a Ψ-learning framework that replaces the hinge loss function in Support Vector Machines (SVMs) by a nonconvex Ψ-loss function. In a similar variant of the Ψ-loss function, Collobert et al. (2006b) and Ertekin et al. (2011) introduced the ramp loss function, which gives a constant penalty for large losses. Both the Ψ-loss and ramp loss functions have been shown to be effective in practice. Therefore, it is important to investigate the use of nonconvex loss functions in the context of distance metric learning. In particular, we pay attention to the ramp loss function, since it can be easily written as a difference of convex functions (DC). Consequently, an effective method for DC programming can be applied to solve the problem. To the best of our knowledge, the method presented in this chapter is the first distance metric learning method that exploits the benefits of DC programming.

Due to the simplicity and effectiveness, this chapter focuses on improving the performance of nearest-neighbor classification. It is well known that the misclassification error rate of the nearest-neighbor classifier converges asymptotically to at most twice the Bayes error rate (Cover and Hart, 1967), however, it is extremely sensitive to noise. In order to overcome the latter drawback, we develop a distance metric learning method making the nearest-neighbor classifier more robust to outliers. In short, our main contributions are summarized as follows:

(i) A distance metric learning framework is proposed to minimize the misclassification rate of the nearest-neighbor classifier. Particularly, our method is inspired by the success of the large-margin principle (Vapnik, 1998). Due to the use of the ramp loss function, our objective function for margin maximization has a strong ability to avoid the influence of outliers.

(ii) Since the objective function can be decomposed into a DC program, a DC algorithm (DCA) (Pham Dinh and Le Thi, 1997) is adopted to solve this problem. Our method iteratively solves a sequence of convex subproblems. We refer to the proposed method as Distance Metric Learning using DC programming (DML-dc).

(iii) We show that the generalization error analysis of the proposed approach has an

important theoretical implication in explaining that minimizing the objective function may improve the generalization performance of nearest-neighbor classification. In particular, the generalization performance is guaranteed via the fat-shattering dimension of Lipschitz classifiers through the combination of a large margin and a low-rank Mahalanobis matrix.

The remainder of this chapter is organized as follows. Section 7.2 gives some preliminaries that will be used throughout this chapter. Section 7.3 briefly reviews some existing approaches that are closely related to our work. Section 7.4 presents our distance metric learning formulation and the corresponding DCA algorithm. Subsequently, Section 7.5 provides the generalization error of the proposed approach using the large-margin criterion. Experimental results are discussed in Section 7.6, followed by some concluding remarks in Section 7.7.

7.2. Preliminaries

To evaluate the performance of a classifier, it does not suffice to consider the training error, but it is also necessary to consider the confidence of the predictions made by the classifier. The margin is one of the geometric measures for evaluating this confidence (Crammer et al., 2003). It provides theoretical generalization bounds on the effectiveness of a classifier, i.e. the higher the confidence is, the lower generalization error the classifier obtains. Many machine learning algorithms have been analyzed using margin concepts, such as SVMs (Vapnik, 1998) and AdaBoost (Schapire et al., 1997).

Given a distance metric d, Crammer et al. (2003) define the margin by which a labeled example $\mathbf{x}_i$ is classified correctly as

$$\phi(\mathbf{x}_i) = d\big(\mathbf{x}_i, \mathrm{NM}(\mathbf{x}_i)\big) - d\big(\mathbf{x}_i, \mathrm{NH}(\mathbf{x}_i)\big)\,, \tag{7.1}$$

where $\mathrm{NM}(\mathbf{x}_i)$ and $\mathrm{NH}(\mathbf{x}_i)$ are the elements of $\mathcal{M}(\mathbf{x}_i)$ and $\mathcal{H}(\mathbf{x}_i)$ (see the definitions of $\mathcal{M}(\mathbf{x}_i)$ and $\mathcal{H}(\mathbf{x}_i)$ in Section 6.3) that are closest to $\mathbf{x}_i$, called nearest miss (NM) and nearest hit (NH), respectively. This margin was originally defined using the Euclidean distance metric for feature selection purposes. The intuition behind this formulation is that it measures how much $\mathbf{x}_i$ can travel in the input space before being misclassified. This margin definition is also adopted implicitly in the well-known RELIEF algorithm (Kira and Rendell, 1992). RELIEF predefines the NH and the NM in the original input space using the Euclidean distance metric, and it leads to a convex optimization problem. The major issue with RELIEF is that the NH and the NM in the original input space are not always the same in the transformed space.

7.3. Related work

Our method is closely related to feature selection methods such as RELIEF (Kira and Rendell, 1992), I-RELIEF (Sun et al., 2010), and SIMBA (Gilad-Bachrach et al., 2004). The reader is referred to (Robnik-Šikonja and Kononenko, 2003) for a more detailed discussion about this family of algorithms in a unified framework. These methods are developed for selecting a set of features that capture the relevant properties of the data by maximizing the margin of nearest-neighbor classification. We use a similar idea of defining the margin as those methods, i.e. the difference between distances of a given example to its NM and NH. However, instead of learning a simple weight vector over the feature set, we learn a full Mahalanobis distance metric. Therefore, the correlations among features are also taken into account, yielding a more powerful model. It is important to note that the NHs and NMs vary when the distance metric is updated, making the optimization problem hard to solve (Sun, 2007). Several approaches have been proposed to avoid this problem. For instance, instead of calculating the NHs and NMs explicitly, Chang (2010) proposed a method using kernel density estimation to estimate the distances to NHs and NMs. By doing so, the problem becomes easier and can be solved by standard optimization techniques such as gradient descent or the EM algorithm. The idea of using kernel density estimation for NHs and NMs is also employed in Neighborhood Components Analysis (NCA) (Goldberger et al., 2005) and Large Margin Subspace Learning (LMSL) (Liu et al., 2013). Based on pairwise constraints, Zhang et al. (2012b) proposed Constrained large Margin Local Projection (CMLP) for multimodal dimensionality reduction. Differently from these methods, we directly minimize the losses based on margin violations defined by NHs and NMs, which may provide a more reliable solution.

Research on distance metric learning has been very active in the past decade (Bellet et al., 2015). Here, we limit ourselves to the discussion of several distance metric learning methods for classification problems. Empirical evidence has demonstrated that distance metric learning methods that employ the large-margin concept usually perform better than other alternatives (Weinberger and Saul, 2009; Shen et al., 2012; Hu et al., 2015b; Zou et al., 2016). One of the most successful methods, namely LMNN (Weinberger and Saul, 2009), aims to learn the distance metric under which each training example has nearest neighbors that share the same class label (i.e. target neighbors), while pushing away those examples with different class labels (i.e. impostor neighbors). The main drawback of LMNN is that the target neighbors are specified a priori and remain unchanged during the training process. Consequently, the performance of LMNN heavily depends on the choice of the target neighbors, since these might be quite different under the optimal distance metric for k-NN classification. In contrast to LMNN, our method adaptively updates the target and impostor neighbors during the training process.

Viewed from an alternative perspective, our method shares the same goals of local learning as LMNN (Weinberger and Saul, 2009), NCA (Goldberger et al., 2005), LDDM (Mu et al., 2013), and DMLMJ (see Chapter 3). Unlike global distance metric learning methods, which usually try to satisfy all the constraints, these local methods only use the neighborhood information, resulting in a suitable model for local classifiers like the nearest-neighbor classifier. This is due to the fact that nearest-neighbor classifiers are mostly influenced by the examples that are close to the test examples. A common guiding principle for local distance metric learning methods is to satisfy the local constraints derived from the neighborhood of each training example (Ying and Li, 2012). While the previous methods are based on local learning, they completely differ in their problem formulation. Therefore, it is unclear whether they have the same theoretical properties as ours.

Recently, there have been several attempts to make the distance metric robust to outliers, which become particularly problematic in noisy data classification. For instance, Wang et al. (2014a) introduced an objective function based on the ℓ_1-norm instead of the usual squared ℓ_2-norm which could be highly influenced by outliers. Similarly to Xiang et al. (2008), they aim to find a linear transformation that minimizes the ratio between the distances of the examples in the must-link constraints and those in the cannot-link constraints. Alternatively, to reduce the influence of noisy examples, Mu et al. (2013) proposed an ensemble framework that combines locally learned distance metrics for the final prediction. Unlike these methods, we use the ramp loss function, which has the strong ability of suppressing the influence of outliers (Collobert et al., 2006b; Ertekin et al., 2011).

7.4. Proposed method

The previous section recalled the importance of margins in order to develop a consistent classifier, i.e. a classifier with a misclassification rate that converges to the best possible. In this section, we present a distance metric learning method that intends to maximize the margin of the nearest-neighbor classifier. We first define the margin in Eq. (7.1) based on the Mahalanobis distance metric. Our distance metric learning method then aims to maximize the margin of each training example. Although the formulation is nonconvex, we show that it belongs to the class of DC programming problems. Subsequently, an efficient algorithm is introduced to solve this problem. Finally, we discuss the convergence and computational complexity of the proposed algorithm.

7.4.1. Problem formulation

Formally, the Mahalanobis distance between two feature vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ takes the following form:

$$d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j)}\,,$$

where $\mathbf{M} \succcurlyeq 0$ is a PSD matrix. Due to the linearity in $\mathbf{M}$, the squared Mahalanobis distance metric $d^2_{\mathbf{M}}$ is usually used to minimize the distances between similar examples $(\mathbf{x}_i, \mathbf{x}_j)$ and to maximize the distances between dissimilar examples $(\mathbf{x}_i, \mathbf{x}_l)$ simultaneously (Weinberger and Saul, 2009; Davis et al., 2007; Shen et al., 2012; Ying and Li, 2012). This goal is often formulated as maximizing:

$$\begin{aligned} &d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \\ &= \big[d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l) - d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)\big]\big[d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l) + d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)\big]\,. \end{aligned} \tag{7.2}$$

Even though the squared distance and the distance are monotonically related, it is not the case that maximizing the difference of squared distances necessarily amounts to maximizing the difference of distances, because the summation in (7.2) will break the monotonicity property. For this reason, we will consider the Mahalanobis distance metric $d_{\mathbf{M}}$, which is a concave function of $\mathbf{M}$ on $\mathbb{S}^D_+$. Accordingly, we can rewrite the margin in Eq. (7.1) based on the distance metric $d_{\mathbf{M}}$ as

$$\begin{aligned} \phi_{\mathbf{M}}(\mathbf{x}_i) &= d_{\mathbf{M}}\big(\mathbf{x}_i, \mathrm{NM}_{\mathbf{M}}(\mathbf{x}_i)\big) - d_{\mathbf{M}}\big(\mathbf{x}_i, \mathrm{NH}_{\mathbf{M}}(\mathbf{x}_i)\big) \\ &= \big[-d_{\mathbf{M}}\big(\mathbf{x}_i, \mathrm{NH}_{\mathbf{M}}(\mathbf{x}_i)\big)\big] - \big[-d_{\mathbf{M}}\big(\mathbf{x}_i, \mathrm{NM}_{\mathbf{M}}(\mathbf{x}_i)\big)\big] \\ &= g_i(\mathbf{M}) - h_i(\mathbf{M})\,, \end{aligned}$$

where

$$\begin{aligned} g_i(\mathbf{M}) &= -\min\big\{d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \mid \mathbf{x}_j \in \mathcal{H}_i\big\}\,, \\ h_i(\mathbf{M}) &= -\min\big\{d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \mid \mathbf{x}_j \in \mathcal{M}_i\big\}\,, \end{aligned}$$

are convex functions of $\mathbf{M}$ on $\mathbb{S}^D_+$.

After having defined the margin, distance metric learning can be performed within the large-margin framework. As is commonly done, the large-margin method maximizes the margin of the example with the smallest margin for a separable classification problem (Schölkopf and Smola, 2001), i.e. the goal is to maximize $\min\big\{\phi_{\mathbf{M}}(\mathbf{x}_i) \mid x_i \in \mathcal{X}\big\}$. To deal with the nonseparable case, we introduce a soft margin to relax this condition, i.e. some examples are allowed to violate their margins by adding additional penalty terms to the objective function. For this

purpose, we formulate our distance metric learning problem as follows:

$$\underset{\mathbf{M}\succcurlyeq 0}{\text{minimize}} \quad \lambda \operatorname{tr}(\mathbf{M}) + \frac{1}{n}\sum_{i=1}^{n} \ell\big(\phi_{\mathbf{M}}(\mathbf{x}_i)\big)\,, \tag{7.3}$$

where ℓ is a loss function penalizing the margin violations and $\lambda > 0$ is a hyper-parameter controlling the trade-off between the margin violations and the regularization. The reason for using the trace norm is that the trace of $\mathbf{M}$ is equal to its nuclear norm, therefore, minimizing the trace norm can lead to a low-rank solution (Kulis, 2012). As a result, it helps to reduce the risk of overfitting. Moreover, the trace norm has been shown to be effective in several distance metric learning studies (Jain et al., 2010; Liu et al., 2015a).

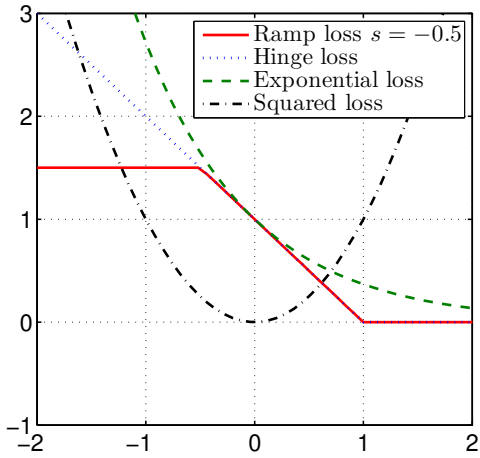


Figure 7.1: An illustration of the ramp loss function with $s = -0.5$ and some convex loss functions

Minimizing the objective function in (7.3) with respect to $\mathbf{M}$ implicitly increases the margins $\phi_{\mathbf{M}}(\mathbf{x}_i)$. However, with the soft margin, misclassified examples like outliers also tend to have a large margin loss because the misclassification penalty is unbounded. As a consequence, they will have a dominant effect on the decision rule of the classifier. In order to alleviate this problem, we consider the ramp loss or truncated hinge loss function (Collobert et al., 2006b), given by

$$R_s(z) = \max\big\{0, 1 - z\big\} - \max\big\{0, s - z\big\}\,,$$

where $s < 1$ is a parameter; we refer to Fig. 7.1 for an illustration of the ramp loss function compared to the hinge loss function. The idea behind the ramp loss is to truncate large losses with the constant s, making the classifier more robust to outliers (Collobert et al., 2006b; Ertekin et al., 2011). Moreover, the ramp loss function has been shown to give better results than the hinge loss function (Chapelle

et al., 2009; Collobert et al., 2006a). Applying the ramp loss function to $\phi_{\mathbf{M}}(\mathbf{x}_i)$ yields

$$\begin{aligned}
&\ell(\phi_{\mathbf{M}}(\mathbf{x}_i)) \\
&= \max\left\{0, 1 - g_i(\mathbf{M}) + h_i(\mathbf{M})\right\} - \max\left\{0, s - g_i(\mathbf{M}) + h_i(\mathbf{M})\right\} \\
&= \max\left\{1 + h_i(\mathbf{M}), g_i(\mathbf{M})\right\} - g_i(\mathbf{M}) - \max\left\{s + h_i(\mathbf{M}), g_i(\mathbf{M})\right\} + g_i(\mathbf{M}) \\
&= \max\left\{1 + h_i(\mathbf{M}), g_i(\mathbf{M})\right\} - \max\left\{s + h_i(\mathbf{M}), g_i(\mathbf{M})\right\}.
\end{aligned}$$

In order to simplify the mathematical notation, let us define

$$\begin{aligned}
G(\mathbf{M}) &= \lambda \operatorname{tr}(\mathbf{M}) + \frac{1}{n}\sum_{i=1}^{n} \max\left\{1 + h_i(\mathbf{M}), g_i(\mathbf{M})\right\}, \\
H(\mathbf{M}) &= \frac{1}{n}\sum_{i=1}^{n} \max\left\{s + h_i(\mathbf{M}), g_i(\mathbf{M})\right\},
\end{aligned}$$

then the objective function in (7.3) can be decomposed into a convex part $G(\mathbf{M})$ and a concave part $-H(\mathbf{M})$. Finally, problem (7.3) can be cast as an instance of DC programming, given by

$$\underset{\mathbf{M} \succcurlyeq 0}{\text{minimize}} \quad G(\mathbf{M}) - H(\mathbf{M}). \tag{7.4}$$

Problem (7.4) is a nonsmooth nonconvex optimization problem, which is difficult to solve in general. Fortunately, a DC program can be solved globally using optimization methods such as branch and bound (Horst and Thoai, 1999), but this can be slow in practice. Next, we will explain how to find a local minimizer for this problem using DCA (Pham Dinh and Le Thi, 1997), which allows to solve large-scale DC programs.

7.4.2. Algorithm

As a starting point for explaining the proposed algorithm, we give a brief introduction to DCA, one of the most effective algorithms for solving DC programs. Essentially, the idea is to linearize the concave part and subsequently solve the convex subproblem. When the objective function is differentiable, DCA can be seen as the Concave-Convex Procedure (CCCP) (Yuille and Rangarajan, 2002). Such algorithms have already been used in SVMs (Collobert et al., 2006b; Ertekin et al., 2011), clustering (Bagirov et al., 2016), regression (Pham Dinh et al., 2014), and so on. We refer the reader to (Pham Dinh and Le Thi, 1997) and the references therein for further details on DCA. Although DCA converges to local optima, Pham Dinh and Le Thi (1997) showed that, in practice, it often converges to the global one. The theoretical results on DCA, such as its convergence properties, can be applied directly to our algorithm. A pseudocode of our algorithm DML-dc is

given in Algorithm 5. Next, we will explain the two basic steps in each iteration of DML-dc.

Algorithm 5 DML-dc: Distance Metric Learning using DC programming

Input: Parameter ϵ
Output: $\mathbf{M}_{t+1} \succcurlyeq 0$

1: Let $\mathbf{M}_0 \succcurlyeq 0$ be an initial solution
2: Set the iteration counter $t \leftarrow 0$
3: **repeat**
4: Linearize the concave part by computing $\mathbf{U}_t \in \partial H(\mathbf{M}_t)$
5: Compute $\mathbf{M}_{t+1}$ by solving the following convex semidefinite program

$$\mathbf{M}_{t+1} \leftarrow \underset{\mathbf{M} \succcurlyeq 0}{\text{minimize}} \quad G(\mathbf{M}) - \langle \mathbf{M}, \mathbf{U}_t \rangle$$

6: Increase the iteration counter $t \leftarrow t + 1$
7: **until** $\|\mathbf{M}_t - \mathbf{M}_{t+1}\|_F \leq \epsilon$

Linearizing the concave part

Assume that $\mathbf{M}_t$ is the solution at the t-th iteration in Algorithm 5. The main idea of DCA is to replace in the original DC program (7.4), at the current solution $\mathbf{M}_t$, the second component $H(\mathbf{M})$ with its affine minorization, given by

$$H_l(\mathbf{M}) = H(\mathbf{M}_t) + \langle \mathbf{M} - \mathbf{M}_t, \mathbf{U}_t \rangle \,, \; \mathbf{U}_t \in \partial H(\mathbf{M}_t) \,.$$

By doing so, problem (7.4) can be approximated by solving a convex program since $H_l(\mathbf{M})$ is a linear function of $\mathbf{M}$. In order to approximate the concave part, we need to compute the subgradient $\mathbf{U}_t$. Note that $H(\mathbf{M})$ is a sum of pointwise maxima of convex functions, which is nondifferentiable. To find the subgradient of a maximum of functions at a point, we can choose one of the subgradients of any function that achieves the maximum at that point. Particularly, in our implementation, $\mathbf{U}_t$ is chosen as

$$\mathbf{U}_t \in \frac{1}{n} \left[\sum_{i \in \mathcal{A}_t} \partial h_i(\mathbf{M}_t) + \sum_{i \in \overline{\mathcal{A}_t}} \partial g_i(\mathbf{M}_t) \right] ,$$

where $\mathcal{A}_t$ is the set consisting of indices i satisfying the condition $s + h_i(\mathbf{M}_t) > g_i(\mathbf{M}_t)$ and $\overline{\mathcal{A}}_t$ is the set consisting of indices i that do not satisfy this condition, i.e. $s + h_i(\mathbf{M}_t) \leq g_i(\mathbf{M}_t)$. Similarly, we can compute a subgradient $p_i(\mathbf{M}_t) \in \partial g_i(\mathbf{M}_t)$

and a subgradient $q_i(\mathbf{M}_t) \in \partial h_i(\mathbf{M}_t)$ as follows[1]:

$$p_i(\mathbf{M}_t) = \begin{cases} -\dfrac{\left[\mathbf{x}_i - \mathrm{NH}_{\mathbf{M}_t}(\mathbf{x}_i)\right]\left[\mathbf{x}_i - \mathrm{NH}_{\mathbf{M}_t}(\mathbf{x}_i)\right]^\top}{2d_{\mathbf{M}_t}\left(\mathbf{x}_i, \mathrm{NH}_{\mathbf{M}_t}(\mathbf{x}_i)\right)}, & \text{if } d_{\mathbf{M}_t}\left(\mathbf{x}_i, \mathrm{NH}_{\mathbf{M}_t}(\mathbf{x}_i)\right) \neq 0\,; \\ \mathbf{0}, & \text{otherwise}\,; \end{cases}$$

$$q_i(\mathbf{M}_t) = \begin{cases} -\dfrac{\left[\mathbf{x}_i - \mathrm{NM}_{\mathbf{M}_t}(\mathbf{x}_i)\right]\left[\mathbf{x}_i - \mathrm{NM}_{\mathbf{M}_t}(\mathbf{x}_i)\right]^\top}{2d_{\mathbf{M}_t}\left(\mathbf{x}_i, \mathrm{NM}_{\mathbf{M}_t}(\mathbf{x}_i)\right)}, & \text{if } d_{\mathbf{M}_t}\left(\mathbf{x}_i, \mathrm{NM}_{\mathbf{M}_t}(\mathbf{x}_i)\right) \neq 0\,; \\ \mathbf{0}, & \text{otherwise}\,; \end{cases}$$

where $\mathbf{0}$ denotes a matrix of zeros. Finally, the subgradient of $H(\mathbf{M})$ at $\mathbf{M}_t$ is computed as

$$\mathbf{U}_t = \frac{1}{n}\left[\sum_{i \in \mathcal{A}_t} p_i(\mathbf{M}_t) + \sum_{i \in \overline{\mathcal{A}_t}} q_i(\mathbf{M}_t)\right].$$

Solving the convex subproblem

After obtaining a subgradient $\mathbf{U}_t$ of $H(\mathbf{M})$ at $\mathbf{M}_t$, we can replace $H(\mathbf{M})$ by its linearizion. Therefore, problem (7.4) is approximated by the following convex semidefinite program:

$$\mathbf{M}_{t+1} = \operatorname*{argmin}_{\mathbf{M} \succcurlyeq 0} \quad G(\mathbf{M}) - H(\mathbf{M}_t) - \langle \mathbf{M} - \mathbf{M}_t, \mathbf{U}_t \rangle\,,$$

or, equivalently,

$$\mathbf{M}_{t+1} = \operatorname*{argmin}_{\mathbf{M} \succcurlyeq 0} \quad G(\mathbf{M}) - \langle \mathbf{M}, \mathbf{U}_t \rangle\,. \tag{7.5}$$

Semidefinite programming can be used to solve problem (7.5), however, it does not scale well on large data sets. To overcome this limitation, we will consider first-order algorithms. The reason for choosing such algorithms is that they only require the first derivative of the objective function and, therefore, they can reduce the time complexity per iteration. In particular, we perform the projected subgradient descent due to its simplicity and effectiveness. This algorithm has already been applied to distance metric learning by Weinberger and Saul (2009),Globerson and Roweis (2006), and Xing et al. (2002).

At the k-th iteration, let $\mathbf{M}_k$ be the current solution of (7.5). Our algorithm operates as follows. First, we compute a subgradient $\mathbf{G}_k$ of the objective function

[1] A matrix $\mathbf{S}$ is a subgradient of a function f at $\mathbf{X}$ if for all $\mathbf{Z}$ in the domain of f, the following condition is satisfied: $f(\mathbf{Z}) \geq f(\mathbf{X}) + \langle \mathbf{Z} - \mathbf{X}, \mathbf{S} \rangle$.

in (7.5). In our implementation, $\mathbf{G}_k$ is chosen as follows:

$$\mathbf{G}_k = \lambda \mathbf{I} + \sum_{i \in \mathcal{B}_k} p_i(\mathbf{M}_k) + \sum_{i \in \overline{\mathcal{B}_k}} q_i(\mathbf{M}_k) - \mathbf{U}_t \,, \tag{7.6}$$

where $\mathcal{B}_k$ is the set consisting of indices i satisfying the condition $1 + h_i(\mathbf{M}_k) > g_i(\mathbf{M}_k)$ and $\overline{\mathcal{B}_k}$ is the set consisting of indices i that do not satisfy this condition, i.e. $1 + h_i(\mathbf{M}_k) \leq g_i(\mathbf{M}_k)$. Note that $p_i(\mathbf{M}_k)$ and $q_i(\mathbf{M}_k)$ are subgradients of $h_i(\mathbf{M}_k)$ and $g_i(\mathbf{M}_k)$, respectively, as defined in the previous subsection. Subsequently, we can update the Mahalanobis matrix in the direction of the subgradient $\mathbf{G}_k$ as

$$\mathbf{M}_{k+1/2} = \mathbf{M}_k - \eta \mathbf{G}_k \,,$$

where η is the step size. Finally, to keep the solution within the cone of PSD matrices, we perform the following projection:

$$\mathbf{M}_{k+1} = \Pi_{\mathbb{S}^D_+}(\mathbf{M}_{k+1/2}) = \mathbf{V}_{k+1/2} \max\{\mathbf{\Lambda}_{k+1/2}, \mathbf{0}\} \mathbf{V}_{k+1/2}^\top \,,$$

where $\mathbf{\Lambda}_{k+1/2}$ is a diagonal matrix whose diagonal vector contains the eigenvalues of $\mathbf{M}_{k+1/2}$ and $\mathbf{V}_{k+1/2}$ is the corresponding eigenvector matrix. One can easily see that the complexity of this algorithm at each iteration scales as $O(N^2D + D^3)$. This is mainly due to the computation of NHs and NMs, which scales as $O(N^2D)$. To reduce this computational burden, we observe that for each training example, its NH and NM lie very nearby. Consequently, we only need to check a subset of examples that likely contains the NH and NM in order to simplify the computation. More specifically, for each training example $\mathbf{x}_i$, we restrict $\mathcal{H}_i$ to be the set containing the m nearest examples of the same class and $\mathcal{M}_i$ to be the set containing the m nearest examples of different classes. These subsets are dynamically updated after a certain number of iterations. By doing so, we reduce the complexity of our algorithm to $O(NmD + D^3)$ per iteration. See Algorithm 6 for a pseudocode summary of these steps.

7.4.3. Convergence and computational complexity

Our algorithm DML-dc has a linear convergence, which is derived from the general convergence properties of DCA (Pham Dinh and Le Thi, 1997). It is also worth pointing out that the objective function value in (7.4) is decreasing, even without considering any line search at each iteration. Due to the computation of NH and NM for each training example, the complexity of computing a subgradient of $H(\mathbf{M})$ scales as $O(N^2D + ND^2)$. However, we observe that DML-dc converges after very few iterations (less than 15 iterations). In each iteration of DML-dc, we also have to find the solution of a convex semidefinite program, which scales as

Algorithm 6 Projected Subgradient Descent for solving (7.5)

Input: Parameters T, η, $\mathbf{U}_t$, m
Output: $\mathbf{M}_T \succcurlyeq 0$

Let $\mathbf{M}_0 \succcurlyeq 0$ be an initial solution
Initialize all sets $\mathcal{H}_i$ and $\mathcal{M}_i$ containing only the m nearest examples
for $k \leftarrow 0$ **to** $T-1$ **do**
    if $(k+1 \mod \textit{some\_constant}) = 0$ **then**
        Recompute all sets $\mathcal{H}_i$ and $\mathcal{M}_i$
    end if
    Compute the subgradient $\mathbf{G}_k$ as in Eq. (7.6)
    Set $\mathbf{M}_{k+1/2} \leftarrow \mathbf{M}_k - \eta \mathbf{G}_k$
    Project onto the PSD cone as $\mathbf{M}_{k+1} \leftarrow \Pi_{\mathbb{S}_+^D}(\mathbf{M}_{k+1/2})$
end for

$O(T(NmD + D^3))$. Therefore, the total complexity of DML-dc at each iteration scales as $O(N^2D + ND^2 + TNmD + TD^3)$.

7.5. Theoretical analysis

In this section, we use the large-margin principle to analyze the effectiveness of the method introduced in Section 7.4. The main purpose of the analysis is to explain the link between the margin and the generalization error of the metric-based method when no assumptions are made about the underlying data distribution. We provide the generalization error for the case of the nearest-neighbor classifier for binary classification problems. The latter restriction is due to the fact that we will rely on results from statistical learning theory that are available for such problems.

We recall some basic concepts of statistical learning theory. Let $\mathcal{Z}$ be an input space in $\mathbb{R}^D$. We assume that the training data are generated independently according to a probability distribution P on $\mathcal{Z} \times \{-1, 1\}$. Let $\mathcal{F}$ be a class of real-valued functions (hypotheses) defined on $\mathcal{Z}$. For the hypothesis $f \in \mathcal{F}$, let $\mathrm{er}_P(f)$ denote the probability that a random couple $(\mathbf{x}, y) \in \mathcal{Z} \times \{-1, 1\}$ is misclassified, i.e.

$$\mathrm{er}_P(f) = \mathbb{P}\left(\mathrm{sgn}\big(f(\mathbf{x})\big) \neq y\right) ,$$

where sgn(.) is a threshold function that takes value -1 if its argument is negative, and value 1 otherwise. We define the empirical error of f on the training set $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i \in \{1, \ldots, n\}\}$ with respect to $\gamma > 0$ as

$$\hat{\mathrm{er}}_{\mathcal{D}}^{\gamma}(f) = \frac{1}{n} \left| \left\{ i \in \{1, \ldots, n\} \mid y_i f(\mathbf{x}_i) < \gamma \right\} \right| .$$

We formally define the hypothesis of the nearest-neighbor classifier as follows. Let $\mathrm{NN}_{\mathbf{M}}$ be a real-valued function such that the sign of $\mathrm{NN}_{\mathbf{M}}(\mathbf{x})$ is the class label of $\mathbf{x}$ assigned by the nearest-neighbor classifier using the Mahalanobis distance metric $d_{\mathbf{M}}$ on $\mathcal{D}$. The magnitude of $\mathrm{NN}_{\mathbf{M}}(\mathbf{x})$ is given by

$$|\mathrm{NN}_{\mathbf{M}}(\mathbf{x})| = d_{\mathbf{M}}\big(\mathbf{x}, \mathrm{NM}_{\mathbf{M}}(\mathbf{x})\big) - d_{\mathbf{M}}\big(\mathbf{x}, \mathrm{NH}_{\mathbf{M}}(\mathbf{x})\big)\,, \tag{7.7}$$

where $\mathrm{NH}_{\mathbf{M}}(\mathbf{x})$ is the nearest example of $\mathbf{x}$ that shares the same class label with $\mathbf{x}$, and $\mathrm{NM}_{\mathbf{M}}(\mathbf{x})$ is the nearest example of $\mathbf{x}$ whose class label is different from the class label of $\mathrm{NH}_{\mathbf{M}}(\mathbf{x})$. The main result of this section is summarized in the following theorem.

Theorem 7.1. *Let $\mathcal{D}$ be the training set containing n examples that are generated independently according to a probability distribution P on $\mathcal{Z} \times \{-1, 1\}$. Assume that the data space $\mathcal{Z}$ lies inside a ball of radius R induced by the Euclidean norm on $\mathbb{R}^D$. Then, with probability at least $1 - \sigma$, for any Mahalanobis distance metric $d_{\mathbf{M}}$ with $\mathrm{tr}(\mathbf{M}) \leq B$ and for any $\gamma > 0$ such that $\hat{\mathrm{er}}^{\gamma}_{\mathcal{D}}(\mathrm{NN}_{\mathbf{M}}) = 0$, the true error of $\mathrm{NN}_{\mathbf{M}}$ can be bounded by*

$$\mathrm{er}_P(\mathrm{NN}_{\mathbf{M}}) \leq \frac{2}{n}\left(c \log_2\left(\frac{34en}{c}\right)\log_2(578n) + \log_2\left(\frac{4}{\sigma}\right)\right).$$

Furthermore, if $\hat{\mathrm{er}}^{\gamma}_{\mathcal{D}}(\mathrm{NN}_{\mathbf{M}}) \neq 0$, with probability at least $1 - \sigma$, we have that

$$\mathrm{er}_P(\mathrm{NN}_{\mathbf{M}}) \leq \hat{\mathrm{er}}^{\gamma}_{\mathcal{D}}(\mathrm{NN}_{\mathbf{M}}) + \sqrt{\frac{2}{n}\left(c \ln\left(\frac{34en}{c}\right)\log_2(578n) + \ln\left(\frac{4}{\sigma}\right)\right)}.$$

In both cases, the constant c satisfies $c \leq \left(\dfrac{128R\sqrt{B}}{\gamma}\right)^{\mathrm{rank}(\mathbf{M})}$.

Remark 7.1. Theorem 7.1 shows that maximizing the lower bound γ of the margin defined by the hypothesis $\mathrm{NN}_{\mathbf{M}}$ will make the nearest-neighbor classifier able to generalize well to unseen data. Therefore, by maximizing $\phi_{\mathbf{M}}$ on the training examples, we attempt to maximize the margin of the nearest-neighbor classifier. Additionally, Theorem 7.1 also provides a theoretical justification to enforce the low-rank constraint in our distance metric learning method as the rank of $\mathbf{M}$ appears in the exponent of the bound as well.

Before giving the proof of Theorem 7.1, let us introduce some useful lemmas.

Lemma 7.1. *The hypothesis of the nearest-neighbor classifier using the Mahalanobis distance metric $d_{\mathbf{M}}$ on the training set $\mathcal{D}$ can be rewritten as*

$$\mathrm{NN}_{\mathbf{M}}(\mathbf{x}) = d_{\mathbf{M}}(\mathbf{x}, \mathcal{X}_{y=-1}) - d_{\mathbf{M}}(\mathbf{x}, \mathcal{X}_{y=1})\,,$$

where $\mathcal{X}_{y=-1}$ and $\mathcal{X}_{y=1}$ denote the subsets containing the training examples whose class labels are -1 and 1, respectively.

Proof. Let $\mathbf{z}_1$ and $\mathbf{z}_2$ be the nearest points of $\mathbf{x}$ using the Mahalanobis distance metric $d_{\mathbf{M}}$ on $\mathcal{X}_{y=-1}$ and $\mathcal{X}_{y=1}$, respectively. We have to consider two different cases in order to prove this lemma.

In the first case, if $d_{\mathbf{M}}(\mathbf{x}, \mathbf{z}_1) \geq d_{\mathbf{M}}(\mathbf{x}, \mathbf{z}_2)$, then $\mathrm{NN}_{\mathbf{M}}(\mathbf{x}) \geq 0$ and

$$\begin{aligned} |\mathrm{NN}_{\mathbf{M}}(\mathbf{x})| &= d_{\mathbf{M}}(\mathbf{x}, \mathbf{z}_1) - d_{\mathbf{M}}(\mathbf{x}, \mathbf{z}_2) \\ &= d_{\mathbf{M}}\big(\mathbf{x}, \mathrm{NM}_{\mathbf{M}}(\mathbf{x})\big) - d_{\mathbf{M}}\big(\mathbf{x}, \mathrm{NH}_{\mathbf{M}}(\mathbf{x})\big)\,. \end{aligned}$$

In the second case, if $d_{\mathbf{M}}(\mathbf{x}, \mathbf{z}_1) < d_{\mathbf{M}}(\mathbf{x}, \mathbf{z}_2)$, then $\mathrm{NN}_{\mathbf{M}}(\mathbf{x}) < 0$ and

$$\begin{aligned} |\mathrm{NN}_{\mathbf{M}}(\mathbf{x})| &= d_{\mathbf{M}}(\mathbf{x}, \mathbf{z}_2) - d_{\mathbf{M}}(\mathbf{x}, \mathbf{z}_1) \\ &= d_{\mathbf{M}}(\mathbf{x}, \mathrm{NM}_{\mathbf{M}}(\mathbf{x})) - d_{\mathbf{M}}(\mathbf{x}, \mathrm{NH}_{\mathbf{M}}(\mathbf{x}))\,. \end{aligned}$$

□

Lemma 7.2. *Let $\mathbf{x}$ be a point inside a ball of radius R induced by the Euclidean norm on $\mathbb{R}^D$ and let $\mathbf{M} \in \mathbb{R}^{D \times D}$ be a PSD matrix such that $\mathrm{tr}(\mathbf{M}) \leq B$. Then it holds that $\|\mathbf{L}^\top \mathbf{x}\| \leq R\sqrt{B}$, where $\mathbf{M} = \mathbf{L}\mathbf{L}^\top$.*

Proof. Since $\mathbf{L}^\top \mathbf{x}$ is a vector, it holds that $\|\mathbf{L}^\top \mathbf{x}\| = \|\mathbf{L}^\top \mathbf{x}\|_F$. Using the submultiplicative property of the Frobenius norm (Golub and Van Loan, 1996), we obtain $\|\mathbf{L}^\top \mathbf{x}\|_F \leq \|\mathbf{L}^\top\|_F \|\mathbf{x}\|_F = \sqrt{\mathrm{tr}(\mathbf{L}\mathbf{L}^\top)}\|\mathbf{x}\| \leq R\sqrt{B}$. □

Lemma 7.3. *Let $\mathcal{X}$ be a set of points in $\mathbb{R}^D$ and let $\mathbf{M} \in \mathbb{R}^{D \times D}$ be a PSD matrix. Then for any $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^D$, it holds that*

$$|d_{\mathbf{M}}(\mathbf{x}_1, \mathcal{X}) - d_{\mathbf{M}}(\mathbf{x}_2, \mathcal{X})| \leq d_{\mathbf{M}}(\mathbf{x}_1, \mathbf{x}_2)\,.$$

Proof. Let $\mathbf{z}_1$ and $\mathbf{z}_2$ be the nearest points of $\mathbf{x}_1$ and $\mathbf{x}_2$ using the Mahalanobis distance metric $d_{\mathbf{M}}$ on $\mathcal{X}$, respectively. By definition of $d_{\mathbf{M}}(\mathbf{x}_1, \mathcal{X})$, it holds that

$$d_{\mathbf{M}}(\mathbf{x}_1, \mathcal{X}) = d_{\mathbf{M}}(\mathbf{x}_1, \mathbf{z}_1) \leq d_{\mathbf{M}}(\mathbf{x}_1, \mathbf{z}_2)\,.$$

Using the triangle inequality of the distance metric $d_{\mathbf{M}}$, we have that

$$\begin{aligned} d_{\mathbf{M}}(\mathbf{x}_1, \mathbf{z}_2) &\leq d_{\mathbf{M}}(\mathbf{x}_1, \mathbf{x}_2) + d_{\mathbf{M}}(\mathbf{x}_2, \mathbf{z}_2) \\ &= d_{\mathbf{M}}(\mathbf{x}_1, \mathbf{x}_2) + d_{\mathbf{M}}(\mathbf{x}_2, \mathcal{X})\,. \end{aligned}$$

The last expression implies

$$d_{\mathbf{M}}(\mathbf{x}_1, \mathcal{X}) - d_{\mathbf{M}}(\mathbf{x}_2, \mathcal{X}) \leq d_{\mathbf{M}}(\mathbf{x}_1, \mathbf{x}_2)\,. \tag{7.8}$$

Exchanging the roles of $\mathbf{x}_1$ and $\mathbf{x}_2$, we obtain in a similar way

$$d_{\mathbf{M}}(\mathbf{x}_2, \mathcal{X}) - d_{\mathbf{M}}(\mathbf{x}_1, \mathcal{X}) \leq d_{\mathbf{M}}(\mathbf{x}_1, \mathbf{x}_2)\,. \tag{7.9}$$

From (7.8) and (7.9), we conclude the proof. □

Lemma 7.4. *Let $\mathcal{D}$ be a training set containing labeled examples in $\mathcal{Z} \times \{-1, 1\}$ and let $\mathbf{M} \in \mathbb{R}^{D \times D}$ be a PSD matrix. Then, for any $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^D$, it holds that*

$$|\mathrm{NN}_{\mathbf{M}}(\mathbf{x}_1) - \mathrm{NN}_{\mathbf{M}}(\mathbf{x}_2)| \leq 2 d_{\mathbf{M}}(\mathbf{x}_1, \mathbf{x}_2)\,.$$

Proof. Using Lemma 7.1, we can write

$$\begin{aligned} |\mathrm{NN}_{\mathbf{M}}(\mathbf{x}_1) - \mathrm{NN}_{\mathbf{M}}(\mathbf{x}_2)| &= |d_{\mathbf{M}}(\mathbf{x}_1, \mathcal{X}_{y=-1}) - d_{\mathbf{M}}(\mathbf{x}_1, \mathcal{X}_{y=1}) \\ &\quad - d_{\mathbf{M}}(\mathbf{x}_2, \mathcal{X}_{y=-1}) + d_{\mathbf{M}}(\mathbf{x}_2, \mathcal{X}_{y=1})| \\ &\leq |d_{\mathbf{M}}(\mathbf{x}_1, \mathcal{X}_{y=-1}) - d_{\mathbf{M}}(\mathbf{x}_2, \mathcal{X}_{y=-1})| \\ &\quad + |d_{\mathbf{M}}(\mathbf{x}_1, \mathcal{X}_{y=1}) - d_{\mathbf{M}}(\mathbf{x}_2, \mathcal{X}_{y=1})|\,. \end{aligned}$$

Using Lemma 7.3, we obtain

$$\begin{aligned} |d_{\mathbf{M}}(\mathbf{x}_1, \mathcal{X}_{y=-1}) - d_{\mathbf{M}}(\mathbf{x}_2, \mathcal{X}_{y=-1})| &\leq d_{\mathbf{M}}(\mathbf{x}_1, \mathbf{x}_2)\,, \\ |d_{\mathbf{M}}(\mathbf{x}_1, \mathcal{X}_{y=1}) - d_{\mathbf{M}}(\mathbf{x}_2, \mathcal{X}_{y=1})| &\leq d_{\mathbf{M}}(\mathbf{x}_1, \mathbf{x}_2)\,, \end{aligned}$$

and hence $|\mathrm{NN}_{\mathbf{M}}(\mathbf{x}_1) - \mathrm{NN}_{\mathbf{M}}(\mathbf{x}_2)| \leq 2 d_{\mathbf{M}}(\mathbf{x}_1, \mathbf{x}_2)\,.$ □

The result in Theorem 7.1 can be proved using the theorem presented by Bartlett (1998), which provides a way to relate the generalization error to the empirical error at a margin γ.

Theorem 7.2. *(Bartlett, 1998) Let $\mathcal{D}$ be the training set containing n examples that are generated independently according to a probability distribution P on $\mathcal{Z} \times \{-1, 1\}$. Let $\mathcal{F}$ be a set of real-valued functions mapping $\mathcal{X}$ to $\mathbb{R}$ and define $c = \mathrm{fat}(\mathcal{F}, \mathcal{X}, \gamma/16)$. Then, with probability at least $1 - \sigma$, for any $f \in \mathcal{F}$ that has margin at least γ on all examples of $\mathcal{D}$, it holds that*

$$\mathrm{er}_P(f) \leq \frac{2}{n}\left(c \log_2\left(\frac{34en}{c}\right)\log_2(578n) + \log_2\left(\frac{4}{\sigma}\right)\right)\,.$$

Furthermore, if $\hat{\mathrm{er}}^{\gamma}_{\mathcal{D}}(f) \neq 0$, with probability at least $1 - \sigma$, we have that

$$\mathrm{er}_P(f) \leq \hat{\mathrm{er}}^{\gamma}_{\mathcal{D}}(f) + \sqrt{\frac{2}{n}\left(c \ln\left(\frac{34en}{c}\right)\log_2(578n) + \ln\left(\frac{4}{\sigma}\right)\right)}\,.$$

We now present the proof of Theorem 7.1.

Proof. In order to use Theorem 7.2, we need to compute the fat-shattering dimension of the nearest-neighbor classifier using the Mahalanobis distance metric $d_{\mathbf{M}}$. According to Lemma 7.4, the function $\mathrm{NN}_{\mathbf{M}}$ is Lipschitz continuous with Lipschitz constant $L = 2$ on the metric space $(\mathcal{X}, d_{\mathbf{M}})$. Therefore, the fat-shattering dimension of the nearest neighbor classifier can be bounded by the covering number $\mathcal{N}(\mathcal{X}, \gamma/32, d_{\mathbf{M}})$ (see Bartlett, 1998, Theorem 13).

In general, it is difficult to estimate the covering number for an arbitrary metric space. There are only a few results on covering numbers, e.g., the fact that the covering number of a closed ball of radius R induced by the Euclidean norm on $\mathbb{R}^D$ can be bounded by $(4R/\epsilon)^D$ (see Cucker and Smale, 2002, for instance), where ϵ is the radius of the disks covering the ball.

Since the Mahalanobis distance metric $d_{\mathbf{M}}$ can be seen as the Euclidean distance metric in the transformed space $\mathcal{X}'$ by performing the linear transformation $\mathbf{x}' = \mathbf{L}^\top \mathbf{x}$, where $\mathbf{M} = \mathbf{L}\mathbf{L}^\top$ and $\mathrm{rank}(\mathbf{M}) = \mathrm{rank}(\mathbf{L})$, the covering number in the metric space $(\mathcal{X}, d_{\mathbf{M}})$ can be seen as the covering number in the transformed metric space $(\mathcal{X}', d_{\mathbf{I}})$. In particular, if $\mathcal{X}$ is a closed ball of radius R induced by the Euclidean norm in $\mathbb{R}^D$, then due to Lemma 7.2, the transformed metric space $\mathcal{X}'$ is a closed ball of radius $R\sqrt{B}$ in $\mathbb{R}^{\mathrm{rank}(\mathbf{M})}$. Hence, according to Cucker and Smale (2002), the covering number $\mathcal{N}(\mathcal{X}, \gamma/32, d_{\mathbf{M}})$ is bounded by $(128R\sqrt{B}/\gamma)^{\mathrm{rank}(\mathbf{M})}$, proving Theorem 7.1. □

7.6. Experiments

In this section, several experiments are conducted to evaluate the effectiveness of our method in the context of nearest-neighbor classification. First, we carry out experiments on various classification benchmark data sets. Second, we conduct additional experiments on real images to validate the robustness of our method. Third, we report results on a synthetic data set containing noise to demonstrate the benefit of using the ramp loss function. Finally, we empirically verify the convergence rate of the proposed algorithm. The experimental settings are detailed in the next subsection.

7.6.1. Experimental settings

We compare the following distance metric learning methods:

1. **Euclidean**: The baseline Euclidean distance metric, which corresponds to the case when the Mahalanobis matrix is the identity matrix.

2. **ITML**: Information-theoretic metric learning (Davis et al., 2007). This method learns a Mahalanobis distance metric in a global sense, i.e., it satisfies all pairwise constraints, while minimizing the differential relative entropy between two multivariate Gaussian distributions to keep the solution as close as possible to a given Mahalanobis distance metric. The formulation results in a convex optimization problem, which can be solved using the Bregman projection algorithm. For ITML, the slack parameter γ is chosen considering as set of values $\{10^{-3}, \ldots, 10^{3}\}$.

3. **LMMCC**: Learning a Mahalanobis matrix for data clustering and classification (Xiang et al., 2008). This method maximizes the ratio of the sum of distances between examples in the cannot-link pairs and the sum of distances between those in the must-link pairs. Due to the orthogonality constraint, the problem cannot be analytically solved. To this end, the authors developed an iterative procedure to find the solution in an efficient way.

4. **LMNN**: Large margin nearest neighbor classification (Weinberger and Saul, 2009). As previously mentioned, LMNN aims to realize that the nearest neighbors of each training example share the same class label, while pushing away examples of other classes. The authors developed an efficient solver based on the subgradient descent method. For LMNN, the trade-off parameter μ is chosen considering as set of values $\{0.125, 0.25, 0.5\}$. Following Weinberger and Saul (Weinberger and Saul, 2009), we use principal components analysis (PCA) as a preprocessing step for LMNN.

5. **DML-eig**: Distance metric learning with eigenvalue optimization (Ying and Li, 2012). This method learns a Mahalanobis distance metric by solving a convex optimization problem, which is inspired on the distance metric learning method for clustering introduced by Xing et al. (2002). The authors proposed an efficient solver based on the Frank-Wolf algorithm, which requires only the computation of the largest eigenvalue and corresponding eigenvector in each iteration to keep the solution within the PSD cone.

6. **DMLMJ**: Distance metric learning through maximization of the Jeffrey divergence. This method learns a linear transformation that maps the input data to a new space, in which the Jeffrey divergence between two Gaussian distributions derived from local constraints is maximized. For DMLMJ, we use five nearest neighbors to estimate the difference spaces.

7. **DML-dc**: Distance metric learning using DC programming described in Algorithm 5. For DML-dc, we tune the trade-off parameter λ considering as set of values $\{0.001, 0.01, 0.1, 1\}$. Based on empirical observations, the parameter s is set to -1, which yields the best results in most of our experiments.

The source codes of these methods are available online from the corresponding

authors' websites[2]. The source code in MATLAB of DML-dc can also be downloaded from http://users.ugent.be/~bacnguye/DML-dc.v1.0.zip.

7.6.2. Benchmark data sets

We perform experiments on fifteen benchmark data sets from the KEEL repository and three data sets from LIBSVM[3], namely mnist (MNIST), pendigits (PEN), and satimage (SAT). All data sets are normalized to have zero mean and unit variance over the training data. Experimental results are obtained by averaging over 10 runs. In each run, the classification accuracies are computed using a 5-fold cross-validation except for the three data sets from LIBSVM where the training and test sets are predefined. Due to the high dimensionality of the MNIST data set, PCA is employed as a preprocessing step to reduce the dimensionality to 100.

The results obtained by the competing methods are reported in Table 7.1. In general, the performance of nearest-neighbor classification is improved using distance metric learning methods. This result confirms that having a good distance metric can lead to improvements for metric-based problems. Our method is competitive with other state-of-the-art distance metric learning methods: ITML, LMMCC, LMNN, DML-eig, and DMLMJ. In most of the cases, DML-dc obtains the best performance. Among the competing methods, LMMCC performs slightly worse than the others. This can be explained by the fact that LMMCC aims to satisfy all possible pairwise constraints, which may constitute a difficult problem. In contrast, local methods such as LMNN, DMLMJ and DML-dc perform quite well in most cases, resulting in a significant improvement in the overall performance of nearest-neighbor classification. As expected, DML-dc outperforms LMNN on various data sets, e.g. *balance*, *bupa*, *monk-2*, and *ring*. This is due to the fact that LMNN predefines the target neighbors using the Euclidean distance metric, whereas DML-dc adaptively updates the target neighbors during the training process. To give a fair comparison, on each data set, we also rank the competing methods based on their classification accuracy. The method with the highest accuracy is assigned rank 1, the one with the second highest accuracy is assigned rank 2, and so on. The average rank of each method over all data sets is presented in the last row of Table 7.1. From these results, we can see that DML-dc achieves the best average rank, demonstrating its robustness and stability for classification tasks.

[2] ITML: http://www.cs.utexas.edu/~pjain/itml/
LMMCC: https://sites.google.com/site/feipingnie
LMNN: http://www.cse.wustl.edu/~kilian/code/code.html
DML-eig: http://empslocal.ex.ac.uk/people/staff/yy267/software.html
DMLMJ: http://users.ugent.be/~bacnguye/DMLMJ.zip
[3] LIBSVM: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Table 7.1: Classification accuracies (standard deviations) of the competing distance metric learning methods on the KEEL data sets. Best results are highlighted in boldface.

Id	Euclidean	ITML	LMMCC	LMNN	DML-eig	DMLMJ	DML-dc
APP	83.07 (3.84)	79.26 (6.29)	79.31 (6.90)	83.03 (5.37)	**83.90 (6.50)**	81.17 (6.56)	82.99 (4.38)
BAL	78.88 (2.69)	90.52 (2.23)	83.52 (2.30)	79.52 (1.93)	82.24 (5.58)	90.40 (2.19)	**90.72 (1.93)**
BUP	64.93 (4.02)	58.55 (4.42)	60.29 (4.18)	61.16 (4.02)	**65.51 (2.38)**	60.29 (8.67)	**65.51 (4.02)**
IRI	94.67 (4.47)	95.33 (3.80)	94.67 (5.06)	94.67 (4.47)	94.00 (3.65)	94.67 (4.47)	**96.00 (3.65)**
LET	95.05 (0.41)	95.38 (0.54)	96.36 (0.23)	96.47 (0.14)	95.09 (0.57)	**97.58 (0.10)**	96.99 (0.27)
MAG	81.57 (0.41)	81.71 (0.46)	80.07 (0.16)	81.46 (0.40)	81.27 (0.58)	81.74 (0.42)	**81.96 (0.49)**
MON	73.84 (2.45)	82.39 (14.66)	92.34 (9.55)	85.42 (13.36)	**100.0 (0.00)**	98.60 (1.91)	98.14 (3.03)
MOV	84.44 (5.50)	82.78 (6.48)	85.56 (4.00)	**87.78 (3.73)**	79.72 (6.26)	83.06 (3.60)	83.33 (5.20)
OPT	97.99 (0.52)	97.92 (0.52)	97.95 (0.55)	98.61 (0.55)	98.47 (0.18)	98.67 (0.51)	**98.68 (0.34)**
RIN	74.61 (1.50)	79.30 (2.00)	75.96 (1.09)	74.95 (1.14)	83.05 (1.12)	84.43 (1.13)	**84.57 (1.00)**
SEG	96.15 (0.91)	96.75 (1.00)	95.71 (1.24)	96.67 (1.25)	94.94 (0.91)	96.93 (0.86)	**97.06 (0.82)**
SON	84.59 (4.50)	85.55 (4.63)	84.60 (4.13)	84.16 (4.27)	84.12 (2.79)	84.62 (5.82)	**86.52 (6.50)**
WDB	96.13 (1.33)	**97.01 (1.00)**	94.90 (0.40)	96.31 (2.28)	95.78 (1.15)	96.66 (1.16)	96.31 (1.57)
WIN	94.97 (3.60)	96.62 (2.36)	97.21 (3.40)	**98.86 (2.56)**	96.08 (2.51)	97.78 (3.62)	98.32 (1.54)
WIS	95.61 (1.87)	95.90 (2.23)	95.76 (2.16)	**96.19 (1.97)**	95.90 (2.35)	95.90 (1.42)	95.61 (1.63)
MNIST	97.16 (0.00)	97.16 (0.00)	97.08 (0.00)	97.56 (0.00)	85.64 (0.00)	97.39 (0.00)	**97.71 (0.00)**
PEN	97.40 (0.00)	97.60 (0.00)	97.46 (0.00)	97.43 (0.00)	**97.80 (0.00)**	97.68 (0.00)	97.71 (0.00)
SAT	88.80 (0.00)	88.80 (0.00)	84.05 (0.00)	89.70 (0.00)	88.90 (0.00)	90.05 (0.00)	**90.35 (0.00)**
Rank	5.27	4.27	5.11	3.72	4.58	2.94	**2.08**

The training time of the competing methods on each data set is reported in Table 7.2. Note that the training time also includes the time for tuning hyper-parameters. All the experiments are run in MATLAB using the same PC. DML-eig is the fastest method because it does not require tuning any hyper-parameter, however, its performance is much lower than other distance metric learning methods. LMMCC also runs quite fast as it has an exponential convergence rate. DMLMJ is the third fastest method since it only needs to perform an eigenvalue decomposition in order to find the linear transformation. Our method is only slightly slower than ITML and LMNN in most cases. This slowness is a result of the fact that DML-dc requires solving several convex subproblems. It is also important to note that our method has mainly been implemented in MATLAB, but further running time improvements can be anticipated. For instance, using intelligent data structures like Ball-Trees and Kd-Trees can speed up the search of nearest neighbors; using C-mex functions can speed up functions written in MATLAB; using online learning techniques can efficiently solve the convex subproblems. Clearly, a careful implementation can make a significant difference in the real computation time.

Table 7.2: Training time (in seconds) of the competing distance metric learning methods on the KEEL data sets. Best results are highlighted in boldface.

Id	ITML	LMMCC	LMNN	DML-eig	DMLMJ	DML-dc
APP	8.36	0.06	7.38	0.28	**0.03**	13.17
BAL	62.57	**0.02**	12.03	0.47	0.11	25.96
BUP	9.88	**0.01**	10.88	0.26	**0.06**	19.55
IRI	42.94	**0.01**	7.59	0.20	0.03	9.70
LET	243.33	28.79	121.96	**8.09**	95.85	1,439.89
MAG	64.84	15.41	728.50	**6.99**	57.14	634.91
MON	11.73	**0.01**	5.23	0.44	0.08	21.16
MOV	523.98	**0.11**	15.53	1.04	0.41	100.71
OPT	368.01	**1.57**	72.61	8.94	35.50	715.46
RIN	21.79	2.80	111.61	**1.42**	20.69	577.38
SEG	132.50	**0.27**	27.71	0.38	1.59	111.64
SON	9.42	**0.05**	9.52	0.64	0.15	56.97
WDB	12.38	**0.02**	9.78	0.53	0.23	44.62
WIN	52.67	**0.01**	6.86	0.39	0.05	16.33
WIS	9.10	**0.02**	9.91	0.33	0.15	24.68
MNIST	2,280.02	1,546.18	5,013.76	**188.28**	9,815.32	15,826.41
PEN	105.30	7.64	59.98	**7.00**	69.06	520.94
SET	112.83	8.13	848.47	**2.82**	49.93	326.61

7.6.3. Experiments on image data sets

To demonstrate the effectiveness of the proposed method, we further compare DML-dc with ITML, LMMCC, LMNN, DML-eig, and the baseline Euclidean distance metric on two image data sets. The first one is the *Coil-100*[4] data set (Nene et al., 1996), which has been widely used in the object recognition literature (Zou et al., 2012; Liu and Srivastava, 2002). This data set consists of 100 objects. Each object comes with 72 images, which are obtained by rotating the object every 5 degrees w.r.t. a fixed color camera (some examples are shown in Fig. 7.2(a)). We convert all images to grayscale and downsample them to 32×32 pixels. Each image is then represented by a 1024-dimensional feature vector. Due their high-dimensional nature, these feature vectors are reduced to 100-dimensional feature vectors using PCA.

The second one is the Extended Yale B (Y-Faces)[5] data set (Soleimani and Matwin, 2016), consisting of 2,424 frontal face images of 39 individuals, which were taken under different illumination conditions (some examples are shown in Fig. 7.2(b)). For each individual, 64 images were captured (a few individuals are represented with fewer images). We use the cropped images and resize them to 32×32 pixels. Apart from pixel features, we also adopt LBP features (Ahonen et al., 2006) computed from local regions to represent each face image (Y-Faces+LBP). Due to the high dimensionality, PCA is employed to obtain a 100-dimensional feature vector for each image. This data set has been used in several distance metric learning studies (Weinberger and Saul, 2009; Yu et al., 2014).

Table 7.3 shows the average classification accuracy and standard deviation of the competing methods. The results here are reported using 5-fold cross-validation in the context of nearest-neighbor classification. As we can see from the results, using the Mahalanobis distance metric leads to a great improvement in the performance of nearest-neighbor classification over the Euclidean distance metric. Interestingly, when using LBP features, DML-dc is still able to improve the performance. We observe that our method outperforms other competing methods, demonstrating the effectiveness of the proposed method. It is important to note that some methods, including LMMCC and DML-eig, cannot even perform better than the Euclidean distance metric.

[4] Available at: http://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php
[5] Available at: http://vision.ucsd.edu/~iskwak/ExtYaleDatabase/ExtYaleB.html

(a) Coil-100

(b) Extended Yale B

Figure 7.2: Examples of the images in (a) the Coil-100 and (b) the Extended Yale B data sets.

Table 7.3: Classification accuracies (standard deviations) of the competing distance metric learning methods on the *Coil-100* and *Y-Faces* data sets. Best results are highlighted in boldface.

Data set	Euclidean	ITML	LMMCC	LMNN	DML-eig	DMLMJ	DML-dc
Coil-100	96.46 (0.23)	98.31 (0.27)	96.43 (0.23)	98.46 (0.28)	93.28 (0.90)	98.36 (0.28)	**99.36 (0.21)**
Y-Faces	90.39 (0.53)	92.49 (0.92)	90.59 (0.75)	93.36 (0.48)	84.28 (2.11)	94.18 (0.47)	**94.97 (0.38)**
Y-Faces+LBP	98.93 (0.23)	97.81 (2.60)	98.84 (0.31)	98.35 (0.76)	95.50 (5.13)	99.38 (0.39)	**99.46 (0.31)**

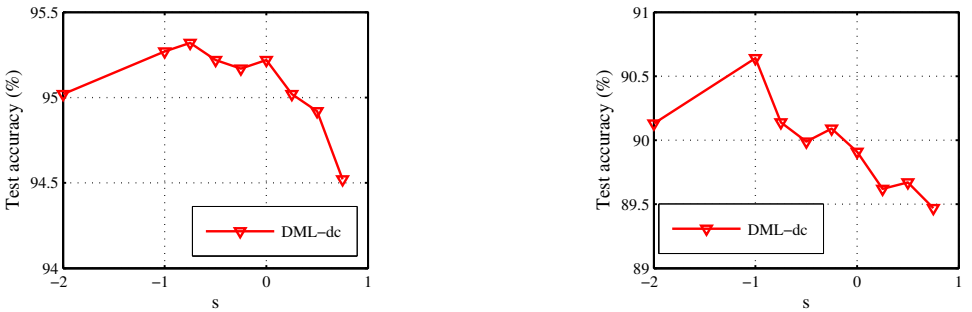
7.6.4. Sensitivity to noise

This subsection aims to compare the sensitivity to noise of the competing methods. For this purpose, we carry out an experiment on handwritten digit recognition (USPS) (Hull, 1994). This data set contains 16×16 grayscale images of the numbers 0–9 written on postal materials. All features are normalized into the interval $[0, 1]$. Since the number of features is large, PCA is employed to reduce the dimensionality to 100 in order to avoid a high computational burden. The training set consists of 7,291 examples and the test set consists of 2,007 examples. Following Ertekin et al. (2011), we generate synthetic data sets with different noise levels by randomly changing the class labels of the training examples. Each noise level corresponds to a different subset of training examples of which the class label will be changed. In our experiment, the percentage of noise is varied from 1 to 10 percent of the training examples. Test accuracies of each method are reported in Table 7.4 by averaging the results over 10 runs. Note that we use the same test set for all methods.

As we can see from this table, all methods (except DML-eig) perform competitively on the original data set without noise. LMMCC is sensitive to the presence of outliers due to the fact that its objective function is based on the squared ℓ_2-norm distances, making the covariance matrices are very sensitive to outliers. This has also been observed by Wang et al. (2014a). Both DMLMJ and ITML are less sensitive to noise. DMLMJ uses a set of nearest neighbors to estimate the difference spaces instead of using only one nearest neighbor, resulting in a distance metric that is less affected by noisy neighbors. ITML randomly chooses a set of pairwise constraints to learn the distance metric. Therefore, the probability of selecting "wrong" pairwise constraints is low when the noise level is low. In contrast, there is a significant decrease in performance for LMNN when the noise level increases. This behavior is caused by the "wrong" target neighbors in the training set. Our method alleviates the effect of these "wrong" target neighbors by using the ramp loss function, making it robust to noise.

Table 7.4: Classification accuracies (standard deviations) of the competing distance metric learning methods on the USPS data set with noise. Best results are highlighted in boldface.

%noise	Euclidean	ITML	LMMCC	LMNN	DML-eig	DMLMJ	DML-dc
0	94.52 (0.00)	94.53 (0.14)	94.47 (0.00)	94.57 (0.00)	91.88 (0.00)	94.67 (0.00)	**95.27 (0.00)**
1	93.53 (0.24)	93.70 (0.33)	93.45 (0.22)	92.27 (0.48)	91.67 (0.55)	94.15 (0.21)	**94.29 (0.23)**
2	92.60 (0.38)	92.74 (0.35)	92.55 (0.42)	90.66 (0.25)	90.47 (0.94)	**93.52 (0.38)**	93.07 (0.35)
5	89.89 (0.35)	89.98 (0.38)	89.82 (0.56)	87.54 (1.24)	86.67 (0.92)	90.59 (0.42)	**90.64 (0.38)**
7	88.05 (0.69)	87.73 (0.93)	87.89 (0.73)	85.22 (1.30)	83.71 (0.87)	88.68 (0.39)	**88.74 (0.46)**
10	85.23 (0.43)	85.70 (0.53)	85.12 (0.53)	81.88 (0.75)	80.61 (0.81)	85.46 (0.60)	**86.10 (0.54)**



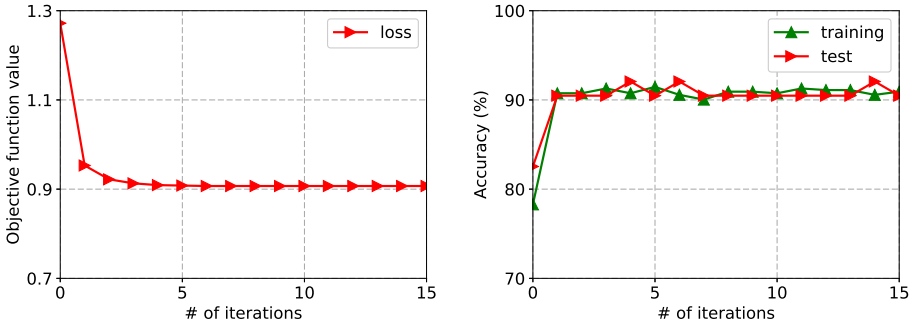
(a) USPS **(b)** USPS + 5% noise

Figure 7.3: Classification accuracy of DML-dc versus different values of s in the ramp loss function.

We also study the behavior of DML-dc when varying the value of s in the ramp loss function. For this purpose, we report the performance of DML-dc on the USPS data set against different values of s in Fig. 7.3. When s takes large negative values, the ramp loss becomes the hinge loss, i.e., it cannot help to remove the outliers from the data. On the other hand, increasing the value of s to be close to 1 may prevent the influences of misclassified examples, which are the most informative examples. As a consequence, this will decrease the generalization performance of DML-dc.

7.6.5. Convergence rate

In this subsection, we empirically verify the convergence rate of DML-dc. As an illustration, Fig 7.4 shows the convergence of DML-dc on the *balance* data set. We show the objective function value and the classification accuracy versus the number of iterations. The training accuracy is computed using leave-one-out cross-validation. As we can see from this figure, DML-dc converges after only five iterations. Both training and test accuracies remain more or less the same once a certain number of iterations is reached.



(a) Objective function value **(b)** Accuracy

Figure 7.4: An illustration of the convergence rate for DML-dc on the *balance* data set: (a) objective function value versus number of iterations and (b) classification accuracy versus number of iterations.

7.7. Conclusion

In this chapter, we have proposed a large-margin distance metric learning method for nearest-neighbor classification. In contrast to previous work on margin maximization in distance metric learning, our method replaces the traditional convex loss function with the ramp loss function, making it more robust in the presence of

noise. To deal with both the nonconvexity and the nonsmoothness of the objective function, an efficient DC programming algorithm has been introduced. It amounts to solving a sequence of convex optimization problems and usually requires a few iterations only. Furthermore, the theoretical foundations of DML-dc have been analyzed, proving that our method yields a good generalization ability to unseen examples. Finally, we provided empirical results demonstrating the effectiveness of our method on several standard benchmark data sets. In general, DML-dc is only slightly slower, however, it performs better than other state-of-the-art distance metric learning methods for nearest-neighbor classification.

8 An efficient method for clustered multi-metric learning

Although there has been an increasing interest in the distance metric learning field, learning a global distance metric is insufficient to obtain satisfactory results when dealing with heterogeneously distributed data. A simple solution to tackle this kind of data is based on kernel embedding methods. However, it quickly becomes computationally intractable as the number of examples increases. In this chapter, we propose an efficient method that learns multiple local distance metrics instead of a single global one. More specifically, the training examples are divided into several disjoint clusters, in each of which a distance metric is trained to separate the data locally. Additionally, a global regularization is introduced to preserve some common properties of different clusters in the learned metric space. By learning multiple distance metrics jointly within a single unified optimization framework, our method consistently outperforms single distance metric learning methods, while being more efficient than other state-of-the-art multi-metric learning methods.

The material of this chapter is based on the following publication:
Nguyen, B., Ferri, F. J., Morell, C., and De Baets, B. (2019a). An efficient method for clustered multi-metric learning. *Information Sciences*, 471:149–163

8.1. Motivation

A successful application of distance metric learning is to improve the performance of k-NN classification. Despite its simplicity, k-NN is well suited for multi-class problems with very large numbers of training examples. It is well known that the performance of k-NN crucially depends on the choice of distance metric (Davis et al., 2007; Mu et al., 2013). Although there is a large amount of works on distance metric learning, most of them simply learn a global distance metric. In many real-world applications, such methods may fail to handle the nonlinearity inherent in the data, especially data from a multimodal distribution. In such cases, there exists no single distance metric that appropriately satisfies all the constraints derived from the data.

A simple solution is to use kernel embedding methods (Schölkopf and Smola, 2001). The idea is to map the input data into a high-dimensional feature space, in which a linear transformation could separate well the data. Many kernel distance metric learning methods have been developed, including ITML (Davis

et al., 2007), LMCA (Torresani and Lee, 2007), and KDMLMJ (see Chapter 3). A general kernel-based framework for distance metric learning can be found in (Jain et al., 2012). By learning a distance metric in the kernel-induced feature space, we can capture any nonlinearity in the original feature space. Following the representer theorem (Schölkopf et al., 2001), the optimal distance metric is implicitly represented by a matrix, which scales quadratically with the number of training examples. As a consequence, the computational burden of these kernel-based methods limits their application to large-scale data sets. Another solution is to learn multiple distance metrics, referred to as multi-metric learning, where each distance metric captures a different region of the data. In the literature, multi-metric learning has recently been studied (Bohné et al., 2014; Parameswaran and Weinberger, 2010; Ramanan and Baker, 2011; Shi et al., 2014; Weinberger and Saul, 2009). However, the scalability on large data sets is not satisfactorily addressed.

Motivated by the above discussion, we propose a novel method, namely clustered multi-metric learning (CMML), for heterogeneously distributed data. In particular, we first divide the data into several clusters using, for instance, k-means clustering (Hartigan and Wong, 1979), then learn a single distance metric for each cluster based on triplet constraints. Moreover, a global distance metric is introduced to capture the common structure among all the clusters, which requires that the distance metric in each cluster should be as close as possible to the global one. On the one hand, the global distance metric serves as a regularization that controls overfitting; on the other hand, it can lead to the propagation of side-information among clusters, resulting in a more robust and stable model. To make CMML scalable for large data sets, we adopt the block-coordinate descent algorithm (Tseng, 2001), which enables us to solve the optimization problem efficiently. For each block, we develop an efficient algorithm based on stochastic gradient descent (SGD). The proposed algorithm only needs the computation of the smallest eigenvalue and corresponding eigenvector of the Mahalanobis matrix at each iteration. Due to the convexity of the stated optimization problem, our algorithm is guaranteed to converge to an optimal solution.

The remainder of this chapter is organized as follows. In Section 8.2, we briefly review several related works. In Section 8.3, we discuss our problem formulation for clustered multi-metric learning and present the proposed algorithm. In Section 8.4, we conduct extensive experiments on various standard benchmark data sets to validate the effectiveness of the proposed algorithm. Finally, we provide some concluding remarks and suggestions for future work in Section 8.5.

8.2. Related work

In the literature, various approaches have been proposed to learn multiple distance metrics that efficiently handle heterogeneously distributed data. The common idea is to locally adjust the distance metric to the properties of the training data in each region. For instance, one distance metric is learned for each class (Weinberger and Saul, 2009), for each training example (Frome et al., 2007a,b; Mu et al., 2013), or for each test example (Domeniconi et al., 2001; Hastie and Tibshirani, 1996). According to Ramanan and Baker (2011), these local distance metrics can provide an approximation to the geodesic distance computed by a metric tensor that defines a possibly different distance metric at each point in the input space. The result may explain the advantage of using multiple distance metrics over a single global one. Below, we review some relevant approaches for each of those categories.

In an early work, Hastie and Tibshirani (1996) used local linear discriminant analysis (LDA) to estimate a distance metric from the neighborhood of each test example (DANN). Similarly, Domeniconi et al. (2001) proposed adaptive metric nearest neighbor (ADAMENN), which learns a local distance metric for each test example such that its neighborhood is elongated along less relevant feature dimensions and shrunk along the most influential ones. Although DANN and ADAMENN can significantly improve the performance of k-NN classification, training a distance metric for each test example is computationally expensive, especially when the number of test examples is large. To reduce this computational burden, Weinberger and Saul (2009) partitioned the training data into clusters and learned a single distance metric for each cluster using LMNN (mmLMNN). Unlike DANN and ADAMENN, mmLMNN learns all the local distance metrics within a unified optimization framework, making them meaningfully comparable for purposes of retrieval and classification. Bohné et al. (2014) proposed LMLML, which first partitions the input space by a Gaussian mixture model and subsequently learns a local distance metric associated with each cluster. In another work, Frome et al. (2007b) jointly learned a weight vector for each training example, yielding local distance functions that capture the relationship in the neighborhoods. Mu et al. (2013) proposed the local discriminative distance metrics (LDDM) algorithm, which learns a distance metric from the neighborhood of each training example. In order to reduce the risk of overfitting and high training cost, Wang et al. (2012) restricted each local distance metric as a linear combination of only a few basis matrices. This framework can be seen as learning a smooth metric matrix function over the data manifold. Similarly, Shi et al. (2014) decomposed the Mahalanobis matrix as a weighted sum of rank-one matrices and learned a smooth function that maps any example to the weighted sum defining its local distance metric. Our method CMML differs from the above methods by the use of global regularization, which assumes that distance metrics from different clusters share some common properties.

Other recent research has focused on dealing with multiple feature representations for multimodal classification problems (Hu et al., 2018; Liang et al., 2018; Zhang et al., 2017). The idea is to extract knowledge from multiple sources representing the same example in order to improve the performance of using only a single source. Unlike conventional methods that learn a distance metric on the concatenated features, these methods jointly learn different distance metrics for different feature representations (modalities). In (Zhang et al., 2017), the distance metric in each modality is defined as the product of an individual matrix from a modality and a global matrix shared across different modalities. Similarly, Hu et al. (2018) forced to learn a shared representation for different modalities in order to preserve their common properties. Instead of learning from different feature representations, our method tries to learn different distance metrics from different regions of the input space.

Multi-task multi-metric learning is also related to our method in the sense that both frameworks learn multiple distance metrics from different subproblems. More specifically, each cluster can be seen as a single task and the global regularization corresponds to the common structure, which is shared by each task. Along with this research direction, there have been several efforts to improve the classification performance. For instance, Parameswaran and Weinberger (2010) extended LMNN (Weinberger and Saul, 2009) to the multi-task paradigm, following the formulation of multi-task SVMs (Evgeniou and Pontil, 2004). Yang et al. (2013) introduced the geometry-preserving criterion among the related tasks based on the von Neumann divergence between two matrices. Recently, Hao et al. (2017) proposed to learn multiple similarity functions for related tasks simultaneously from the triplet constraints via online learning. Zheng et al. (2017) proposed the hierarchical multi-task sparse distance metric learning algorithm that can learn a tree of multiple sparse distance metrics hierarchically over a visual tree. Even though multi-task multi-metric learning methods and our method use information from several subproblems, there exist crucial differences between them in the problem formulation as well as in the objective. In multi-task learning, each task is an independent sample of possibly different distributions, while in our method, each cluster is a disjoint subset of the same sample from the same distribution. More importantly, the objective of multi-task learning is to improve the performance of all tasks simultaneously by enforcing a common regularization. Our objective is to simplify the original problem by partitioning it into several smaller subproblems.

In particular, our method is inspired by an extension of support vector machines (namely CSVM) by Gu and Han (2013). This method is developed using several linear support vector machines to handle nonlinearly distributed data. Instead of learning separating hyperplanes, CMML learns different Mahalanobis distance metrics based on local triplet constraints. Note that these triplet constraints can involve examples from different clusters, while in CSVM, only examples from the

same cluster are used for training individual support vector machines. Due to the positive semidefiniteness constraint, our problem is more difficult to solve than the one in (Gu and Han, 2013). To this end, we propose an efficient algorithm that can quickly converge to an optimal solution.

8.3. Clustered multi-metric learning

In this section, we develop a multi-metric learning method that learns multiple distance metrics in order to tackle heterogeneously distributed data. In particular, the training data set is divided into several clusters such that each training example belongs to only one of the clusters. These clusters should be representative and contain enough discriminative information (i.e. triplet constraints). In this chapter, we use k-means as the baseline clustering algorithm due to its simplicity and efficiency. The prediction of an unseen example is performed using the local distance metric learned from its corresponding cluster. Next, we will describe in detail our formulation as well as the optimization procedures for training.

8.3.1. Problem formulation

We will consider the standard classification problem defined in $\mathbb{R}^D$. Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ denote the training set, where $\mathbf{x}_i \in \mathbb{R}^D$ denotes the i-th training example with its corresponding label y_i. The Mahalanobis distance between two examples $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as

$$d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j)}\,,$$

where $\mathbf{M} \succcurlyeq 0$. Assume that the training set is divided into $T > 1$ disjoint clusters $\mathcal{D} = \mathcal{C}^{(1)} \cup \cdots \cup \mathcal{C}^{(T)}$. For each cluster $\mathcal{C}^{(c)}$, we aim at learning a Mahalanobis distance metric $d_{\mathbf{M}^{(c)}}$ that satisfies a given set of triplet constraints

$$\mathcal{T}^{(c)} = \left\{(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_l) \mid \mathbf{x}_i \in \mathcal{C}^{(c)} \text{ and } \mathbf{x}_i \text{ is closer to } \mathbf{x}_j \text{ than to } \mathbf{x}_l\right\}.$$

In particular, our formulation aims to satisfy the following criteria. First, similarly to the single distance metric learning case, each local distance metric should satisfy as many triplet constraints as possible. Let $d_{\mathbf{M}^{(c)}}$, where $\mathbf{M}^{(c)} \succcurlyeq 0$, denote the local distance metric for the c-th cluster, then we aim to enforce

$$d_{\mathbf{M}^{(c)}}(\mathbf{x}_i, \mathbf{x}_l) > d_{\mathbf{M}^{(c)}}(\mathbf{x}_i, \mathbf{x}_j)$$

for any $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_l) \in \mathcal{T}^{(c)}$, $1 \leq c \leq T$, which can be achieved by maximizing the

margin

$$d^2_{\mathbf{M}^{(c)}}(\mathbf{x}_i, \mathbf{x}_l) - d^2_{\mathbf{M}^{(c)}}(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{M}^{(c)}, \mathbf{Z}_r \rangle \,,$$

where $\mathbf{Z}_r = (\mathbf{x}_i - \mathbf{x}_l)(\mathbf{x}_i - \mathbf{x}_l)^\top - (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top$ with a subscript r to denote the triplet $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_l)$. As the triplet constraints are enumerable, by abusing the notation slightly, we will also call a triplet constraint as r. Several loss functions, such as the square or logistic loss, can be used to maximize the above margin. In particular, we consider to minimize the hinge loss with margin one, i.e. $\max(1 - \langle \mathbf{M}^{(c)}, \mathbf{Z}_r \rangle, 0)$. The margin is set to one since its value only has an impact on the scale of $\mathbf{M}^{(c)}$ and not on the performance of nearest-neighbor retrieval.

Second, following the intuition behind multi-task learning (Gu and Han, 2013; Parameswaran and Weinberger, 2010; Yang et al., 2013), an appropriate sharing of information among different distance metrics may result in several benefits. It might allow for the propagation of side-information among clusters, thus avoiding overfitting in each cluster. For this purpose, let $d_{\mathbf{M}^{(0)}}$, where $\mathbf{M}^{(0)} \succcurlyeq 0$, denote the global distance metric, then we enforce each local distance metric $d_{\mathbf{M}^{(c)}}$ to be similar to $d_{\mathbf{M}^{(0)}}$ using the squared Frobenius norm, i.e. $\|\mathbf{M}^{(0)} - \mathbf{M}^{(c)}\|_F^2$, $1 \leq c \leq T$. By doing so, all local distance metrics are related with each other. Additionally, a trace-norm regularization is applied to the global distance metric, which implicitly imposes the low-rank constraint (Recht et al., 2010) on $\mathbf{M}^{(0)}$, and as a result, it also reduces the risk of overfitting. Note that this regularization may affect all local distance metrics since these are enforced to be close to the global one.

Summarizing, we can formulate our multi-metric learning problem as an instance of the following optimization problem

$$\begin{array}{ll} \underset{\mathbf{M}^{(0)},\ldots,\mathbf{M}^{(T)}}{\text{minimize}} & \alpha \operatorname{tr}\big(\mathbf{M}^{(0)}\big) + \frac{1}{T}\sum_{c=1}^{T}\left[\frac{\beta}{2}\left\|\mathbf{M}^{(0)} - \mathbf{M}^{(c)}\right\|_F^2 + \frac{1}{N^{(c)}}\sum_{r\in\mathcal{T}^{(c)}}\xi_r\right] \\ \text{subject to} & \langle \mathbf{M}^{(c)}, \mathbf{Z}_r \rangle \geq 1 - \xi_r,\ \xi_r \geq 0\,, \forall r \in \mathcal{T}^{(c)}, c \in \{1,\ldots,T\} \\ & \mathbf{M}^{(0)}, \ldots, \mathbf{M}^{(T)} \succcurlyeq 0\,, \end{array} \tag{8.1}$$

where $N^{(c)}$ denotes the number of constraints in $\mathcal{T}^{(c)}$, $\alpha > 0$ and $\beta > 0$ are hyper-parameters, and ξ_r are slack variables. Clearly, problem (8.1) is jointly convex with respect to all parameters in $\mathcal{V} = \{\mathbf{M}^{(0)}, \ldots, \mathbf{M}^{(T)}\}$. It can be seen that when increasing α (while keeping β relatively small), the matrix $\mathbf{M}^{(0)}$ tends to $\mathbf{0}$. Consequently, our multi-metric learning problem amounts to learning independent distance metrics, which are trained in each cluster separately. On the other hand, when increasing β (while keeping α relatively small), all the local distance metrics tend to be similar. Consequently, the above formulation can be thought of as a generalization framework between learning one single distance metric and learning several independent ones.

For illustrative purposes, Fig. 8.1 shows the main idea behind our method. There are three clusters, each of which contains three classes. Examples belonging to the same class are denoted by the same shape; the red, green and blue colors are used to represent the training examples, while the yellow color is used to represent the test examples. On the left-hand side of the figure, if the distance metrics are trained independently, $\mathbf{M}^{(3)}$ is easily overfitted to the training data, since the third cluster contains very few training examples and as a result, it cannot represent the distribution of the data accurately. On the right-hand side of the figure, by jointly learning all the distance metrics, we expect that the global distance metric can capture the shared information between different clusters. In this case, the distance metric defined by $\mathbf{M}^{(3)}$ tends to be similar to the global one, making the prediction more reliable.

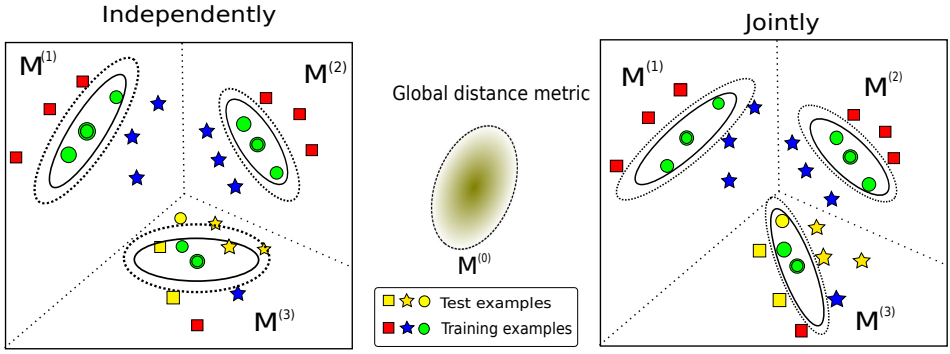


Figure 8.1: An illustration of CMML. Examples belonging to the same class are denoted by the same shape. Left-hand side: all local distance metrics are trained independently. Right-hand side: all local distance metrics are jointly trained.

8.3.2. Optimization solver

Although problem (8.1) is convex, it is very expensive to directly solve it using standard semidefinite programming techniques (Boyd and Vandenberghe, 2004). Another common solution is to use first-order algorithms such as batch gradient descent as in (Weinberger and Saul, 2009; Xing et al., 2002). However, these algorithms are not scalable in practical settings. This is mainly due to the large number of triplets as well as the positive semidefiniteness constraints. To address this computational burden, we adopt the bock-coordinate descent method (Tseng, 2001) to solve problem (8.1) in a more efficient manner. In particular, we solve the problem based on a single distance metric, while keeping the remaining distance metrics unchanged. This optimization procedure is cycled over all parameters in $\mathcal{V}$ until it converges, i.e. the objective function corresponding to problem (8.1)

$$J(\mathbf{M}^{(0)}, \ldots, \mathbf{M}^{(T)})$$

$$= \alpha \operatorname{tr}\big(\mathbf{M}^{(0)}\big) + \frac{1}{T}\sum_{c=1}^{T}\left[\frac{\beta}{2}\left\|\mathbf{M}^{(0)} - \mathbf{M}^{(c)}\right\|_F^2 + \frac{1}{N^{(c)}}\sum_{r\in\mathcal{T}^{(c)}} \max\big(1 - \langle \mathbf{M}^{(c)}, \mathbf{Z}_r\rangle, 0\big)\right] \tag{8.2}$$

no longer decreases in successive iterations. All the distance metrics are initialized using the identity matrix. Algorithm 7 briefly summarizes the optimization procedure of our method. As shown below, the global distance metric can be obtained as a closed-form solution, while each local distance metric will require a further optimization procedure.

Algorithm 7 Block-coordinate descent to solve problem (8.1)

Input: $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, $\{\mathcal{T}^{(c)}\}_{c=1}^T$, α, β, ϵ
Output: $\mathcal{V}_s = \{\mathbf{M}_s^{(c)}\}_{c=0}^T$

```
Initialize M_0^(c) ← I for c = 0, ..., T                ▷ Initialize all parameters
Compute the objective function J_0 in (8.2) at V_0 = {M_0^(c)}_{c=0}^T
Set s ← 0
repeat
    Increase the iteration counter s ← s + 1
    Update the global distance metric
    Set M_s^(0) ← M_*^(0) using Eq. (8.4)
    Update the local distance metrics
    for c ← 1, ..., T do
        Run Algorithm 8 to obtain M_s^(c)
    end for
    Compute the objective function J_s in (8.2) at V_s = {M_s^(c)}_{c=0}^T
until |J_s − J_{s−1}| < ε
```

Solving for the global distance metric

Keeping $\mathcal{V} \setminus \mathbf{M}^{(0)}$ fixed, we can obtain the matrix $\mathbf{M}^{(0)}$ by solving the following optimization problem

$$\mathbf{M}_*^{(0)} = \underset{\mathbf{M}^{(0)}\succcurlyeq 0}{\operatorname{argmin}} \quad \alpha \operatorname{tr}\big(\mathbf{M}^{(0)}\big) + \frac{1}{T}\sum_{c=1}^{T}\frac{\beta}{2}\left\|\mathbf{M}^{(0)} - \mathbf{M}^{(c)}\right\|_F^2, \tag{8.3}$$

which admits, in fact, a simple closed-form solution.

Theorem 8.1. *The optimal solution to problem* (8.3) *is given by*

$$\mathbf{M}_*^{(0)} = \mathcal{P}_{\mathbb{S}^+}\left(\frac{1}{T}\sum_{c=1}^{T}\mathbf{M}^{(c)} - \frac{\alpha}{\beta}\mathbf{I}\right). \tag{8.4}$$

Proof. Multiplying the objective function in (8.3) with $2/\beta$ and using the standard

properties of the Frobenius norm yield

$$\underset{\mathbf{M}^{(0)} \succcurlyeq 0}{\operatorname{argmin}} \quad \frac{2\alpha}{\beta}\operatorname{tr}\big(\mathbf{M}^{(0)}\big) + \big\|\mathbf{M}^{(0)}\big\|_F^2 - \frac{2}{T}\sum_{c=1}^{T}\Big\langle \mathbf{M}^{(0)}, \mathbf{M}^{(c)}\Big\rangle + \frac{1}{T}\sum_{c=1}^{T}\big\|\mathbf{M}^{(c)}\big\|_F^2 .$$

Adding a constant term

$$\Big\|\frac{1}{T}\sum_{c=1}^{T}\mathbf{M}^{(c)} - \frac{\alpha}{\beta}\mathbf{I}\Big\|_F^2 - \frac{1}{T}\sum_{c=1}^{T}\big\|\mathbf{M}^{(c)}\big\|_F^2$$

to the objective function, we obtain the following equivalent problem

$$\begin{aligned}\underset{\mathbf{M}^{(0)} \succcurlyeq 0}{\operatorname{argmin}} \quad & \big\|\mathbf{M}^{(0)}\big\|_F^2 - 2\Big\langle \mathbf{M}^{(0)}, \frac{1}{T}\sum_{c=1}^{T}\mathbf{M}^{(c)} - \frac{\alpha}{\beta}\mathbf{I}\Big\rangle + \Big\|\frac{1}{T}\sum_{c=1}^{T}\mathbf{M}^{(c)} - \frac{\alpha}{\beta}\mathbf{I}\Big\|_F^2 \\ \equiv \quad & \Big\|\mathbf{M}^{(0)} - \Big(\frac{1}{T}\sum_{c=1}^{T}\mathbf{M}^{(c)} - \frac{\alpha}{\beta}\mathbf{I}\Big)\Big\|_F^2 .\end{aligned}$$

The latter is known as the nearest PSD matrix approximation problem under the Frobenius norm (Higham, 1988), and as a result, the optimal solution is given by Eq. (8.4). □

According to Theorem 8.1, we can easily find the optimal solution for problem (8.3) in $O(D^3)$ by performing only one projection onto the cone of PSD matrices.

Solving for the local distance metric

Keeping $\mathcal{V} \setminus \mathbf{M}^{(c)}$ fixed, we update $\mathbf{M}^{(c)}$, $1 \le c \le T$, by solving the following problem

$$\begin{aligned}\underset{\mathbf{M}^{(c)}}{\text{minimize}} \quad & \tfrac{\beta}{2}\Big\|\mathbf{M}^{(c)} - \mathbf{M}^{(0)}\Big\|_F^2 + \tfrac{1}{N^{(c)}}\textstyle\sum_{r\in\mathcal{T}^{(c)}}\xi_r \\ \text{subject to} \quad & \langle \mathbf{M}^{(c)}, \mathbf{Z}_r\rangle \ge 1 - \xi_r,\ \xi_r \ge 0\,, \forall r \in \mathcal{T}^{(c)} \\ & \mathbf{M}^{(c)} \succcurlyeq 0\,.\end{aligned} \tag{8.5}$$

In order to solve this problem, we employ stochastic subgradient descent (SGD), which has been widely used for neural networks (Bottou, 1991) and SVMs (Shalev-Shwartz et al., 2007). Unlike batch gradient descent methods (Parameswaran and Weinberger, 2010; Weinberger and Saul, 2009), SGD consists in drawing an example at random and optimizing the objective function based on that example, avoiding the full-gradient computation. In particular, SGD is suitable for large-scale learning problems since its computational complexity does not depend on the size of problem.

We start by writing the objective function in (8.5) as a sum of loss functions associated with each triplet constraint, i.e.

$$F(\mathbf{M}^{(c)}) = \frac{\beta}{2}\left\|\mathbf{M}^{(c)} - \mathbf{M}^{(0)}\right\|_F^2 + \frac{1}{N^{(c)}} \sum_{r \in \mathcal{T}^{(c)}} \max\left(1 - \langle \mathbf{M}^{(c)}, \mathbf{Z}_r \rangle, 0\right).$$

Using this formulation, our algorithm performs as follows. At the t-th iteration, we replace the objective function $F(\mathbf{M}^{(c)})$ with an approximation based on a single triplet constraint $r \in \mathcal{T}^{(c)}$, i.e.

$$f_t(\mathbf{M}^{(c)}) = \frac{\beta}{2}\left\|\mathbf{M}^{(c)} - \mathbf{M}^{(0)}\right\|_F^2 + \max\left(1 - \langle \mathbf{M}^{(c)}, \mathbf{Z}_r \rangle, 0\right).$$

We consider the subgradient at $\mathbf{M}_t^{(c)}$, given by

$$\nabla_t^{(c)} = \beta(\mathbf{M}_t^{(c)} - \mathbf{M}^{(0)}) - [\![\langle \mathbf{M}_t^{(c)}, \mathbf{Z}_r \rangle < 1]\!]\, \mathbf{Z}_r\,, \tag{8.6}$$

where $[\![.]\!]$ denotes the indicator function which takes value 1 if its argument is true and 0 otherwise. Subsequently, we update $\mathbf{M}_t^{(c)}$ to $\mathbf{M}_{t+1}^{(c)}$ by setting

$$\begin{aligned} \mathbf{M}_{t+1/2}^{(c)} &= \mathbf{M}_t^{(c)} - \eta_t \nabla_t^{(c)}\,, \\ \mathbf{M}_{t+1}^{(c)} &= \mathcal{P}_{\mathbb{S}^+}(\mathbf{M}_{t+1/2}^{(c)})\,, \end{aligned}$$

where $\eta_t > 0$ denotes the step size. Since the projection of $\mathbf{M}_{t+1/2}^{(c)}$ onto the cone of PSD matrices scales as $O(D^3)$, the latter update can be computationally expensive, especially when the dimensionality is high. Due to the fact that $F(\mathbf{M}^{(c)})$ is β-strongly convex, by setting $\eta_t = 1/(\beta t)$ as is commonly done (Shalev-Shwartz et al., 2007), we obtain

$$\mathbf{M}_{t+1/2}^{(c)} = \Big(1 - \frac{1}{t}\Big)\mathbf{M}_t^{(c)} + \frac{1}{t}\mathbf{M}^{(0)} + \eta_t [\![\langle \mathbf{M}_t^{(c)}, \mathbf{Z}_r \rangle < 1]\!]\, \mathbf{Z}_r\,. \tag{8.7}$$

Since $\mathbf{Z}_r$ is the difference of two rank-one matrices, $\mathbf{M}_{t+1/2}^{(c)}$ has at most a single negative eigenvalue (Golub and Van Loan, 1996). As a consequence, $\mathbf{M}_{t+1}^{(c)}$ can be efficiently computed using the following formulation,

$$\mathbf{M}_{t+1}^{(c)} = \mathbf{M}_{t+1/2}^{(c)} - \min\big(\lambda_{\min}, 0\big)\mathbf{u}_{\min}\mathbf{u}_{\min}^\top\,,$$

where $\lambda_{\min}$ is the smallest eigenvalue of $\mathbf{M}_{t+1/2}^{(c)}$ with corresponding eigenvector $\mathbf{u}_{\min}$. Using the Lanczos method or the power method (Golub and Van Loan, 1996) with a random start vector, $\lambda_{\min}$ and $\mathbf{u}_{\min}$ can be approximated in $O(D^2)$. A similar idea to reduce the computational burden of this projection was also introduced in (Shalev-Shwartz et al., 2004). Consequently, the computational complexity of updating the matrix $\mathbf{M}^{(c)}$ per iteration scales as $O(D^2)$ instead of

$O(D^3)$. The entire optimization procedure is summarized in Algorithm 8, where $K > 0$ denotes the maximum number of iterations. Note that we initialize the local distance metric with $\mathbf{M}^{(0)}$.

Algorithm 8 Stochastic subgradient descent to solve problem (8.5)

Input: $\mathcal{T}^{(c)}$, $\mathbf{M}^{(0)}$, β, K
Output: $\mathbf{M}_{K+1}^{(c)}$

Set $\mathbf{M}_1^{(c)} \leftarrow \mathbf{M}^{(0)}$
for $t \leftarrow 1, dots, K$ **do**
  Choose $r \in \mathcal{T}^{(c)}$ uniformly at random
  Set $\eta_t \leftarrow \frac{1}{\beta t}$
  if $\langle \mathbf{Z}_r, \mathbf{M}_t^{(c)} \rangle < 1$ **then**
    Set $\mathbf{M}_{t+1/2}^{(c)} := \left(1 - \frac{1}{t}\right)\mathbf{M}_t^{(c)} + \frac{1}{t}\mathbf{M}^{(0)} + \eta_t \mathbf{Z}_r$
    Compute $(\lambda_{\min}, \mathbf{u}_{\min})$ for $\mathbf{M}_{t+1/2}^{(c)}$
    Set $\mathbf{M}_{t+1}^{(c)} \leftarrow \mathbf{M}_{t+1/2}^{(c)} - \min\left(\lambda_{\min}, 0\right)\mathbf{u}_{\min}\mathbf{u}_{\min}^\top$
  else
    Set $\mathbf{M}_{t+1}^{(c)} \leftarrow \left(1 - \frac{1}{t}\right)\mathbf{M}_t^{(c)} + \frac{1}{t}\mathbf{M}^{(0)}$
  end if
end for

8.3.3. Convergence

In this subsection, we show that the proposed method converges to an optimal solution. Typically, the convergence of block-coordinate descent requires that the objective function is strictly convex (or quasiconvex and hemivariate) and differentiable (Sargent and Sebastian, 1973) because the method may otherwise get stuck at a nonstationary point for a nondifferentiable function. Unfortunately, this is not the case for the objective function in Eq. (8.2). The latter forces us to use the convergence result of block-coordinate descent for nonsmooth optimization developed by Tseng (2001) when the nondifferentiable part is separable.

In particular, we make use of Proposition 5.1 (Tseng, 2001) (see A.2) by extending the objective function in Eq. (8.2). We start by introducing an indicator function σ that takes value 0 if its argument is true and $+\infty$ otherwise. The resulting extended-valued function is then

$$L(\mathbf{M}^{(0)}, \ldots, \mathbf{M}^{(T)}) = L_0(\mathbf{M}^{(0)}, \ldots, \mathbf{M}^{(T)}) + L_{T+1}(\mathbf{M}^{(0)}) + \sum_{c=1}^{T} L_c(\mathbf{M}^{(c)}),$$

where

$$L_0(\mathbf{M}^{(0)},\dots,\mathbf{M}^{(T)}) = \frac{\beta}{2T}\sum_{c=1}^{T}\left\|\mathbf{M}^{(0)}-\mathbf{M}^{(c)}\right\|_F^2,$$

$$L_c(\mathbf{M}^{(c)}) = \sigma(\mathbf{M}^{(c)}\succcurlyeq 0) + \frac{1}{N^{(c)}T}\sum_{r\in\mathcal{T}^{(c)}}\max\left(1-\langle\mathbf{M}^{(c)},\mathbf{Z}_r\rangle,0\right),$$

$$c = 1\dots,T,$$

$$L_{T+1}(\mathbf{M}^{(0)}) = \sigma(\mathbf{M}^{(0)}\succcurlyeq 0) + \alpha\mathrm{tr}\big(\mathbf{M}^{(0)}\big).$$

We demonstrate that the following conditions are satisfied:

(B1) L_0 is continuous since the squared Frobenius norm is continuous.

(B2) For each $t\in\{0,1,\dots,T\}$ and $\mathbf{M}^{(c)}$, $c\neq t$, the function at each coordinate block $\ell_t(\mathbf{M}^{(t)}) = L(\mathbf{M}^{(0)},\dots,\mathbf{M}^{(t)},\dots,\mathbf{M}^{(T)})$ is quasiconvex and hemivariate. Due to the fact that L is jointly convex, ℓ_t is quasiconvex. Since the squared Frobenius norm is strictly quasiconvex, it is easy to show that ℓ_t is also hemivariate (Ortega and Rheinboldt, 1979).

(B3) $L_0,\dots,L_{T+1}$ are lower semicontinuous. Since $\mathbb{S}^+$ is a closed set, the extended-valued function L_c, $1\leq c\leq T+1$, remains lower semicontinuous (Hiriart-Urruty and Lemaréchal, 2012). Clearly, L_0 is lower semicontinuous because it is a continuous function.

(C2) Since L_0 contains only the squared Frobenius norm, dom $L_0 = \mathbb{R}^{D\times D}\times\cdots\times\mathbb{R}^{D\times D}$ and dom $L_c = \mathbb{R}^{D\times D}$, $1\leq c\leq T+1$.

We have shown above that $L_0,\dots,L_{T+1}$ satisfy assumptions (B1)–(B3) and (C2) in Proposition 5.1 (Tseng, 2001). In our block-coordinate descent method, the essentially cyclic rule (Tseng, 2001) is employed. Moreover, for each small block, we employ SGD in Algorithm 8, which has an $O(\log K/K)$ convergence rate. This result directly follows from (Shamir and Zhang, 2013).

Theorem 8.2 ((Shamir and Zhang, 2013))**.** *Let $\mathbf{M}_*^{(c)}$ be an optimal solution for problem* (8.5)*. Consider a sequence of PSD matrices $\mathbf{M}_1^{(c)},\dots,\mathbf{M}_K^{(c)}$ such that $\mathbf{M}_{t+1}^{(c)} = \mathcal{P}_{\mathbb{S}^+}(\mathbf{M}_t - \frac{1}{\beta t}\nabla_t^{(c)})$ for $t\geq 1$. Assume that $\mathbb{E}\big[\|\nabla_t^{(c)}\|_F^2\big]\leq G^2$ for all t, then, for any $K>1$, it holds that*

$$\mathbb{E}\big[F(\mathbf{M}_K^{(c)}) - F(\mathbf{M}_*^{(c)})\big] \leq \frac{17G^2(1+\log K)}{\beta K}.$$

Therefore, our block-coordinate descent method is guaranteed to converge to an optimal solution.

8.3.4. Computational complexity

The computational complexity of CMML involves the identification of the clusters and solving the overall optimization problem. First, we analyze the complexity of partitioning the training data using a clustering algorithm. One of the reasons for choosing k-means clustering is its simplicity and efficiency. It is well known that the complexity of k-means scales as $O(N * T * D * I)$, where I denotes the number of iterations (Hartigan and Wong, 1979) and T denotes the number of clusters. In our experiments, we employ the standard implementation for k-means with a maximum of 100 iterations, but further improvements can be anticipated, for instance, using the triangle inequality (Elkan, 2003b) to speed up k-means.

Empirically, we have found that the block-coordinate descent method to solve our optimization problem converges after very few iterations (less than 20 iterations) in the outer loop. In each iteration, we need to solve a sequence of convex semidefinite programs, each of which scales as $O(K * D^2)$ for the local distance metric case and $O(D^3)$ for the global distance metric case. Summarizing, the overall complexity of block-coordinate descent at each iteration scales as $O(T * K * D^2 + D^3)$.

8.3.5. Testing phase

Once the clusters are defined and the corresponding local distance metrics are trained, we need to select a proper local distance metric which will be employed to classify a test example. This selection can be done efficiently by determining to which cluster the test example belongs. Essentially, it consists in selecting the cluster whose center is the nearest center with respect to the test example. The same distance metric should be used to perform k-means clustering as well as to determine the cluster. Depending on the application, there may exist some auxiliary information or prior knowledge about the distance metric that naturally leads to a good partition. In the absence of prior knowledge, a simple strategy is to adopt the Euclidean distance metric. Note that unlike other approaches such as PLML (Wang et al., 2012) and SCML (Shi et al., 2014), which combine all the basis distance metrics to define the local distance metric, CMML only uses a single local distance metric from one cluster.

An ideal distance between two examples should be computed as the geodesic distance using different local distance metrics on a Riemannian manifold (Ramanan and Baker, 2011; Shi et al., 2014; Wang et al., 2012). However, this requires a computationally expensive optimization and complicated algorithm. Therefore, we only adopt the local distance metric of the test example to compute its nearest neighbors, which results in a very fast search. An intuitive explanation for this simple strategy is that the class label assigned by the k-NN classifier for a test example only depends on its neighborhood. If the local discriminative information

is preserved by using an appropriate distance metric, then we can still improve the performance of k-NN classification. Next, we will show how to select the triplet constraints in order to preserve the local discriminative information.

8.3.6. Strategy of selecting triplet constraints

Since conventional clustering methods only consider the similarity between examples, they often group examples of the same class together. As a consequence, the clusters tend to be pure and the local distance metrics result in a trivial solution. To avoid this situation, we ensure that triplet constraints contain examples from all classes. More specifically, we generate triplet constraints by using k nearest neighbors of the same class and k nearest neighbors of different classes for each training example (Ying and Li, 2012). This has the advantage of avoiding the large number of triplet constraints, which scales as $O(N^3)$ for all possible triplets, making the optimization algorithm more tractable. Moreover, it exploits the fact that if each training example is surrounded by k neighbors of the same class, then the k-NN classification will succeed (Weinberger and Saul, 2009). Therefore, only triplet constraints derived from the neighborhood of each training example should be considered. When no prior knowledge is available, the search for the nearest neighbors is based on, for instance, the Euclidean distance. Note that the nearest neighbors are searched in the entire training set. Therefore, the selection of triplet constraints is not influenced by the clustering algorithm.

After having obtained a set of triplet constraints $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_l)$, we can divide it into T subsets of constraints according to the cluster membership of $\mathbf{x}_i$. As mentioned above, one triplet constraint may involve examples of more than one cluster. By satisfying all triplet constraints simultaneously, we implicitly learn the local distance metrics in a "global" sense. This strategy also makes sure that there is no cluster with an empty set of constraints, since there always exists a set of triplet constraints for each training example. We expect that CMML can perform consistently well in terms of classification accuracy, even on small data sets.

8.4. Experiments

In this section, we conduct extensive experiments on several publicly available data sets to show the effectiveness of our proposed method in terms of classification performance and running time. As is commonly done in various distance metric learning studies (Weinberger and Saul, 2009; Ying and Li, 2012), all experimental results are reported in the context of k-NN classification with $k = 3$. Next, we will detail the experimental settings and results.

8.4.1. Experimental settings

We compare the following distance metric learning methods:

1. **Euclidean**: The baseline Euclidean distance metric.
2. **LMNN**[1] (Weinberger and Saul, 2009): It is a representative method using the large-margin principle to improve the performance of k-NN classification. The authors proposed an efficient solver based on projected subgradient descent. As proposed in (Weinberger and Saul, 2009), we set the maximum number of iterations to 1,000 and tune the trade-off parameter μ considering as set of values $\{0.125, 0.25, 0.5\}$.
3. **ITML**[2] (Davis et al., 2007): This method minimizes the LogDet divergence between two matrices while satisfying pairwise constraints on the distance metric. The authors introduced a fast and scalable algorithm based on the Bregman projection. We set the maximum number of iterations to 10^5 and tune the slack parameter γ considering as set of values $\{10^{-3}, \ldots, 10^3\}$.
4. **DML-eig**[3] (Ying and Li, 2012): Inspired by the work in (Xing et al., 2002), this method learns a Mahalanobis distance metric by solving a convex optimization problem. The solver is based on an eigenvalue optimization framework, which requires only the computation of the maximum eigenvalue in each iteration.
5. **mmLMNN**[4] (Weinberger and Saul, 2009): This method learns several distance metrics in different clusters of the input space, where each class corresponds to a cluster. The distance to a target example is computed using the local distance metric associated with the cluster to which the target example belongs. Similarly to LMNN, a convex optimization framework is developed to learn the local distance metrics simultaneously.
6. **kmLMNN**: This is a simple baseline. We first divide the training set into several clusters using k-means, then learn a Mahalanobis distance metric for each cluster using LMNN. Note that all the distance metrics are learned independently.
7. **DANN** (Hastie and Tibshirani, 1996): As mentioned above, it is a state-of-the-art multi-metric learning method. Following Hastie and Tibshirani (1996), the number of nearest neighbors used to estimate the distance metric is set to $\max(N/5, 50)$ and the regularization parameter ϵ is set to 1. For each test example, DANN is trained with 5 iterations.

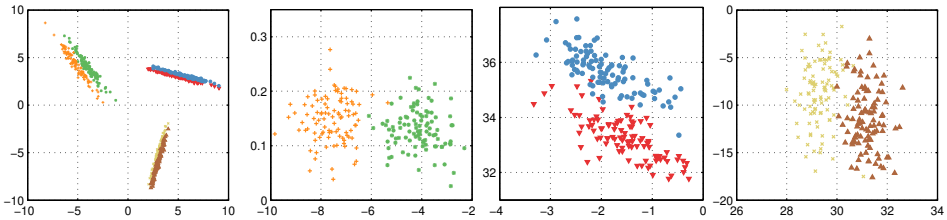
[1] http://www.cse.wustl.edu/~kilian/code/code.html
[2] http://www.cs.utexas.edu/~pjain/itml/
[3] http://empslocal.ex.ac.uk/people/staff/yy267/software.html
[4] http://www.cse.wustl.edu/~kilian/code/code.html

8. **SCML**[5] (Shi et al., 2014): For learning multiple local distance metrics, SCML learns a smooth function that maps an example to a weighted sum defining its local distance metric. We tune the hyperparameter β considering as set of values $\{10^{-6}, \ldots, 10^{-2}\}$. Following the authors, kernel PCA is performed to learn the embedding of each example so that the weights can vary nonlinearly. The number of basis matrices is set to 400 and the embedding dimension to 40. Similarly to CMML, the triplet constraints are generated using three nearest neighbors of the same class and three nearest neighbors of different classes for each training example.

9. **CMML**: We tune the hyper-parameter α considering as set of values $\{0.001, 0.1, 10\}$ and β considering as set of values $\{0.001, 0.01, 0.1\}$. For the SGD algorithm in Algorithm 8, we set the maximum number of iterations to 10^5. The Euclidean distance metric is used to perform k-means and to determine the corresponding cluster for each test example. The source code in MATLAB of CMML can be downloaded from https://github.com/bacnguyencong/CMML.

8.4.2. A synthetic data set

We first illustrate CMML with a synthetic data set for which a single distance metric is not sufficient to improve the performance of k-NN classification (see Fig. 8.2(a)). The data set consists of six classes, each of which is randomly generated from a bivariate normal distribution. Examples of the same class are represented with the same color and style. The data set is partitioned into three clusters in order to train CMML. Figures 8.2(b) to 8.2(d) show how examples of the same class are grouped together in the transformed spaces induced by the corresponding local distance metrics. These results confirm that CMML can fit well the distance metric over different regions of the input space.



(a) Synthetic data **(b)** Cluster 1 **(c)** Cluster 2 **(d)** Cluster 3

Figure 8.2: An illustration of CMML on a synthetic data set: (a) Original data generated by normal distributions, (b)-(d) Projection of the data in the space induced by each local distance metric.

[5] http://researchers.lille.inria.fr/abellet/code.html

8.4.3. Benchmark KEEL data sets

We use eighteen benchmark data sets from the Knowledge Extraction based on Evolutionary Learning (KEEL) (Triguero et al., 2017) (see Table A.1). All features are normalized to have zero mean and unit standard deviation over the training data. The average classification accuracy and standard deviation are estimated using 5-fold cross-validation. Each experiment is repeated five times to remove the randomness in the sampling process. All partitions for training and testing are collected by stratified sampling from all classes. To get the best results for all methods, the hyper-parameters are tuned via internal validation using only the training data. For CMML and kmLMNN, we set the number of clusters to 3 for small data sets ($N \leq 1,000$) and 10 for large data sets. In the next section, we empirically show that CMML is not very sensitive to this number.

The average classification accuracies are shown in Table 8.2. For each data set, we assign rank 1 to the method with the highest accuracy, rank 2 to the one with the second highest accuracy, and so on. The average rank for each method over all data sets is reported in the last row of Table 8.2. From the results, we can see that the performance of k-NN classification is significantly improved by using the distance metrics learned from the data. Generally, the methods that learn multiple distance metrics (i.e. mmLMNN, DANN, SCML, and CMML) outperform the methods that learn a single distance metric (i.e. ITML, LMNN, and DML-eig). According to the average rank, CMML performs the best among the competing methods, followed by SCML as second best. Interestingly, DANN does not always obtain a better performance than that of single distance metric learning methods. This is mainly due to the lack of training examples, since DANN only estimates the distance metric based on information from the neighborhood of each test example. In most cases, mmLMNN performs slightly better than LMNN. There are relatively few cases in which kmLMNN outperforms LMNN. This means that jointly learning multiple distance metrics (i.e. mmLMNN and CMML) can perform better than learning multiple distance metrics separately (i.e. mkLMNN). The results also confirm that CMML consistently performs well, even on small data sets.

Table 8.1: Unadjusted p-value and adjusted p-values according to the Wilcoxon test and different post-hoc tests over eighteen data sets based on classification accuracy using CMML as the control method.

Method	pUnadj	pBonf	pHolm	pHoch	pHomm	Hypothesis
Euclidean	8.2439E-7	6.5951E-6	6.5951E-6	6.5951E-6	6.5951E-6	Rejected
kmLMNN	2.3407E-5	1.8725E-4	1.6384E-4	1.6384E-4	1.4044E-4	Rejected
DML-eig	3.9923E-5	3.1939E-4	2.3954E-4	2.3954E-4	2.3954E-4	Rejected
ITML	0.0019	0.0153	0.0096	0.0085	0.0057	Rejected
DANN	0.0032	0.0253	0.0126	0.0085	0.0063	Rejected
LMNN	0.0035	0.0279	0.0126	0.0085	0.0070	Rejected
mmLMNN	0.0042	0.0339	0.0126	0.0085	0.0085	Rejected
SCML	0.0106	0.0847	0.0126	0.0106	0.0106	Rejected

Table 8.2: Classification accuracies (standard deviation) of the competing distance metric learning methods on the KEEL data sets. The best results are highlighted in boldface.

Id	Single metric				Multiple metrics				
	Euclidean	ITML	LMNN	DML-eig	mmLMNN	kmLMNN	DANN	SCML	CMML
APP	83.03(5.4)	83.98(5.3)	82.08(4.0)	84.89(5.3)	82.08(4.0)	80.17(4.1)	84.89(5.3)	83.98(5.3)	**84.98(2.5)**
BAL	83.04(2.0)	91.84(2.3)	87.84(2.2)	87.68(3.1)	86.72(1.8)	88.80(3.4)	**95.52(0.9)**	87.68(2.6)	92.48(1.7)
BUP	65.51(3.3)	63.19(4.3)	62.90(2.2)	62.32(5.3)	65.22(3.1)	64.35(3.5)	69.57(5.1)	**70.72(4.7)**	66.96(9.6)
ION	84.90(4.8)	87.19(2.6)	90.03(3.9)	85.48(4.7)	**94.29(2.7)**	87.18(3.3)	86.90(3.4)	87.75(4.9)	87.76(4.5)
IRI	94.00(3.7)	94.67(3.0)	96.00(1.5)	95.33(3.0)	95.33(1.8)	94.67(3.8)	95.33(3.0)	96.00(2.8)	**96.67(2.4)**
MON	96.07(2.7)	98.84(2.0)	97.22(2.5)	**100.0(0.0)**	97.22(2.5)	97.22(1.8)	93.75(1.8)	97.68(2.7)	**100.0(0.0)**
SON	85.12(3.4)	85.10(5.7)	**86.97(5.7)**	84.12(2.3)	86.05(4.0)	80.73(7.5)	59.63(8.3)	83.64(7.1)	85.56(5.1)
VEH	70.33(2.1)	79.90(3.5)	78.37(1.3)	71.04(1.3)	75.77(1.8)	78.02(1.8)	**83.33(0.5)**	77.89(2.3)	77.90(2.3)
VOW	95.76(1.5)	96.46(1.2)	96.77(1.6)	94.95(1.4)	**98.38(0.7)**	97.17(0.8)	95.56(2.3)	96.36(1.4)	97.97(1.6)
WDB	**97.19(0.7)**	97.01(1.8)	96.66(1.3)	96.31(1.4)	97.01(1.0)	96.66(1.9)	88.40(3.4)	**97.19(1.3)**	**97.19(1.0)**
WIN	95.51(1.5)	97.75(1.3)	96.62(1.3)	97.21(2.0)	97.75(1.3)	94.95(2.3)	94.38(2.0)	**98.33(2.5)**	97.76(1.3)
LET	94.61(0.5)	97.04(0.3)	96.23(0.3)	87.61(0.4)	97.22(0.3)	96.20(0.4)	95.86(0.5)	96.33(0.5)	**97.29(0.2)**
MAG	83.33(0.5)	83.30(0.3)	83.39(0.6)	82.70(0.2)	82.39(0.6)	83.30(0.4)	**84.51(0.7)**	84.00(0.6)	84.40(0.5)
PAG	96.69(0.4)	96.67(0.4)	96.80(0.4)	95.74(0.9)	96.35(0.4)	96.64(0.5)	96.69(0.6)	96.56(0.4)	**96.88(0.4)**
PHO	88.29(0.6)	88.38(0.6)	88.01(0.6)	88.55(0.7)	88.27(0.7)	88.55(0.8)	**88.88(0.5)**	88.45(0.8)	88.58(0.6)
RIN	70.89(1.6)	80.38(0.8)	71.12(1.7)	86.41(1.0)	**95.20(0.4)**	82.66(1.3)	93.41(0.5)	73.69(1.4)	86.47(0.9)
SPA	91.10(0.5)	91.97(0.7)	92.28(0.4)	92.15(1.1)	92.30(0.6)	92.04(0.4)	90.49(1.0)	91.04(0.3)	**92.36(0.2)**
TWO	96.39(0.4)	96.22(0.5)	96.54(0.4)	**97.32(0.3)**	96.51(0.3)	95.38(0.8)	97.24(0.3)	97.19(0.4)	96.86(0.4)
Rank	6.69	5.03	4.86	5.94	4.81	6.06	4.89	4.53	**2.19**

In order to determine whether there exist significant differences in classification performance among the results reported in Table 8.2, we follow the recommendations made by Demšar (2006). First, the Friedman test is employed at a confidence level of $\alpha = 0.05$ with the null hypothesis that all the competing methods obtain the same results on average. Since the p-value was 1.2423×10^{-4}, we reject the null hypothesis. This implies that there exist statistically significant differences between at least two competing methods. Subsequently, we employ the Wilcoxon signed-rank test and several post-hoc tests, including Bonferroni-Dunn, Holm, Hochberg, and Hommel, to determine whether a competing method performs equivalently or significantly different from the control method (i.e. CMML, which has the lowest rank). In order to compensate for multiple comparisons (Demšar, 2006), the p-values in post-hoc tests are adjusted. If the adjusted p-value for a particular null hypothesis is less than $\alpha = 0.05$, then that hypothesis is rejected. Table 8.1 reports the unadjusted p-value (pUnadj) computed by the Wilcoxon signed-rank test, the adjusted p-values computed by the Bonferroni-Dunn (pBonf), Holm (pHolm), Hochberg (pHoch), and Hommel (pHomm) tests. These test results show that CMML significantly outperforms the other competing methods (except in one case, namely SCML for the Bonferroni-Dunn test with a confidence level of $\alpha = 0.05$).

Tables 8.3 and 8.4 report the average running time of the competing methods in terms of training as well as testing time, respectively. To facilitate a comparison among the competing methods, we also show the total running time in the last row of these tables. Note that the training time reported in Table 8.3 takes into consideration the time for tuning the hyper-parameters. DANN only estimates the distance metric in the test phase. Clearly, DML-eig is the fastest method since it learns a single distance metric and does not require tuning any hyper-parameter. In most cases, CMML runs faster than other multi-metric learning methods (i.e. SCML, mmLMNN and kmLMNN) although the latter two do not require tuning hyper-parameters. Because the number of clusters is small, CMML is significantly faster than ITML and LMNN in terms of training time. The overall running time in the test phase of mmLMNN, kmLMNN, and CMML is approximately equal to that of single distance metric learning methods. In contrast, DANN requires a running time proportional to the number of test examples. This may limit the application of DANN to real-world problems when the number of test examples is relatively large. SCML requires to compute the embedding of each test example in the feature space using kernel PCA, making the test process relatively slow on large data sets.

8.4.4. Real data sets

In this subsection, we evaluate our method on several real challenging data sets, including USPS (Hull, 1994), MNIST (Lecun et al., 1998), and ISOLET (Cole

Table 8.3: Training time (in seconds) of the competing methods on the KEEL data sets. Best results are highlighted in boldface.

#	Single disance metric			Multiple distance metrics			
	ITML	LMNN	DML-eig	mmLMNN	kmLMNN	SCML	CMML
1.	44.38	20.78	**0.31**	9.22	10.97	9.53	0.68
2.	60.63	27.19	**0.43**	20.72	21.26	102.36	2.35
3.	51.59	27.38	**0.16**	3.31	20.48	39.66	3.40
4.	65.91	32.67	**1.32**	16.04	21.16	45.6	17.90
5.	49.66	19.81	**0.20**	10.67	12.59	13.43	0.33
6.	55.91	17.83	**0.50**	8.97	18.08	53.02	1.63
7.	155.11	29.99	**1.42**	16.68	20.55	27.25	54.66
8.	67.39	43.12	**0.59**	29.54	23.47	118.46	70.74
9.	69.81	29.61	**0.87**	29.26	23.96	137.26	18.52
10.	73.34	25.80	**1.42**	14.89	15.72	167.98	15.66
11.	53.77	19.69	**0.48**	10.09	0.85	24.69	1.14
12.	175.19	294.38	224.77	686.34	238.6	1305.10	**171.53**
13.	162.26	798.58	169.78	886.87	526.74	1513.75	**119.29**
14.	83.23	256.25	**9.51**	149.58	104.66	228.13	22.53
15.	79.03	191.51	**9.27**	232.69	138.82	269.87	23.14
16.	93.00	253.55	18.13	121.28	210.00	593.48	173.31
17.	222.17	133.75	**9.49**	66.99	59.67	512.7	487.23
18.	93.89	60.02	**18.17**	24.09	81.11	633.40	84.20
Total	1656.27	2281.91	466.82	2337.23	1548.69	5795.67	1268.24

Table 8.4: Testing time (in seconds) of the competing methods on the KEEL data sets.

#	Single disance metric			Multiple distance metrics				
	ITML	LMNN	DML-eig	mmMNN	kmLMNN	DANN	SCML	CMML
1.	1.47E-03	1.50E-03	1.38E-03	6.06E-03	4.01E-03	0.17	0.15	2.87E-03
2.	3.45E-03	4.15E-03	3.70E-03	6.67E-03	6.08E-03	0.86	1.13	4.50E-03
3.	1.81E-03	2.20E-03	1.70E-03	2.67E-03	3.38E-03	0.40	0.29	3.57E-03
4.	2.60E-03	3.10E-03	3.49E-03	5.66E-03	6.13E-03	1.05	0.35	5.03E-03
5.	1.22E-03	1.24E-03	1.04E-03	2.27E-03	2.84E-03	0.16	0.13	2.49E-03
6.	2.56E-03	2.61E-03	2.66E-03	3.25E-03	5.75E-03	0.55	0.39	5.06E-03
7.	2.89E-03	2.24E-03	2.03E-03	4.07E-03	5.29E-03	0.93	0.29	5.52E-03
8.	4.72E-03	4.74E-03	5.04E-03	6.81E-03	0.01	2.49	1.48	8.08E-03
9.	5.13E-03	5.48E-03	5.18E-03	9.13E-03	0.01	3.03	2.1	8.93E-03
10.	3.22E-03	4.85E-03	3.40E-03	5.02E-03	8.40E-03	1.85	1.72	6.94E-03
11.	2.39E-03	1.78E-03	3.64E-03	2.41E-03	2.72E-03	0.23	0.19	2.71E-03
12.	1.24	1.27	1.25	1.74	1.33	588.13	568.74	1.42
13.	1.12	1.14	1.13	0.67	1.17	387.47	484.9	1.16
14.	0.08	0.09	0.09	0.06	0.10	42.49	28.44	0.10
15.	0.08	0.09	0.09	0.05	0.10	35.17	30.31	0.09
16.	0.15	0.17	0.16	0.10	0.17	93.04	60.57	0.18
17.	0.07	0.07	0.07	0.05	0.10	93.70	21.21	0.10
18.	0.19	0.17	0.16	0.10	0.17	86.73	92.11	0.19
Total	2.96	3.03	2.98	2.82	3.20	1338.45	1294.50	3.30

and Fanty, 1990). The first two correspond to handwritten digit recognition problems while the third one is about letter speech recognition. These data sets are widely used in several distance metric learning studies (Shalev-Shwartz et al., 2004; Weinberger and Saul, 2009; Yang et al., 2013). The USPS[6] data set contains

[6] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

7,291 examples of digits for training and 2,007 for testing of size 16×16 pixels. The MNIST[7] data set contains 60,000 examples of digits for training and 10,000 for testing of size 28×28 pixels. All partitions for training and testing are predefined. The ISOLET data set was collected from 150 different speakers (each one spoke the name of each letter of the alphabet twice), where the objective is to recognize what letter was been uttered. There are 26 classes corresponding to the letters of the alphabet. Since the data set consists of five groups, the first four groups are used for training and the last one for testing. We summarize the characteristics of the data sets in Table 8.5. Due to the high dimensionality, PCA is employed as a preprocessing step to reduce the dimensionality of the input data to 100. All the competing methods are configured with the same settings as in the preceding experiments.

Table 8.5: Description of real data sets used in our experiments.

#	Data set	Features	Classes	Training	Test
1.	ISOLET	617	26	6,238	1,559
2.	MNIST	784	10	60,000	10,000
3.	USPS	256	10	7,291	2,007

Table 8.6 shows the classification accuracy on the test sets of the competing methods. Clearly, distance metric learning leads to a significant improvement in letter speech recognition. We observe that CMML performs competitively with mmLMNN, while it performs significantly better than other competing methods. It is interesting to see that the baseline kmLMNN method also performs well on the ISOLET and MNIST data sets. However, it obtains a very poor performance on the USPS data set.

Table 8.6: Classification accuracies of the competing distance metric learning methods on real data sets.

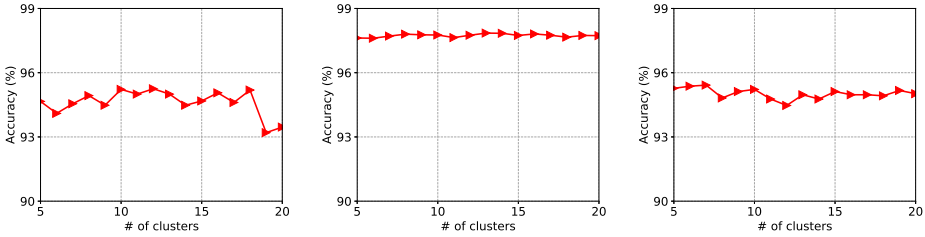
#	Single distance metric				Multiple distance metrics				
	Euclidean	ITML	LMNN	DML-eig	mmLMNN	kmLMNN	DANN	SCML	CMML
1.	90.19	94.87	95.15	91.28	95.13	94.23	94.42	94.68	**95.25**
2.	97.33	97.51	97.66	73.50	**98.68**	97.67	96.36	97.41	97.73
3.	94.87	93.57	94.77	90.33	95.12	92.92	92.03	94.88	**95.22**

We also conduct experiments to investigate the classification performance of CMML under changes in the number of clusters. For this purpose, we vary the number of clusters from 5 to 20. Figure 8.4 illustrates the classification accuracy of CMML versus the numbers of clusters on the test sets. As expected, CMML obtains a stable performance with different number of clusters. This is mainly

[7] http://yann.lecun.com/exdb/mnist/

due to the strategy of selecting triplet constraints, which guarantees that all local distance metrics are correlated.

It is important to mention that the training time of CMML is approximately proportional to the number of clusters (see Subsection 8.3.4). Since the prediction of CMML is not very sensitive to the number of clusters, we could choose a low number of clusters to balance between the effectiveness and efficiency.

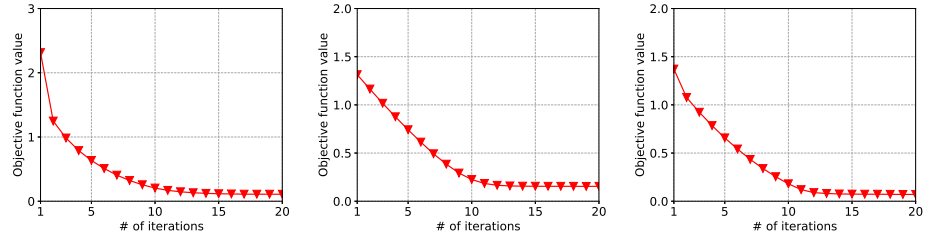


(a) ISOLET **(b)** MNIST **(c)** USPS

Figure 8.3: Classification accuracy of CMML versus number of clusters on real data sets.

8.4.5. Convergence

We further verify the convergence of block-coordinate descent. For this purpose, we empirically show the convergence of CMML on three real data sets, namely ISOLET, MNIST, and USPS (see Fig. 8.4). As an illustration, the hyper-parameters are set as follows: $\alpha = 0.01$ and $\beta = 0.1$. As can be seen from the figure, CMML converges after only 10 iterations. The objective function values remain more or less the same when a certain number of iterations is reached.



(a) ISOLET **(b)** MNIST **(c)** USPS

Figure 8.4: Convergence of CMML versus number of iterations on real data sets.

8.5. Conclusion

In this chapter, we have developed a multi-metric learning method to handle heterogeneously distributed data. A divide-and-conquer strategy has been proposed to address this kind of data. More specifically, data are divided into several clusters, in each of which a distance metric is trained to separate the data locally. We have also introduced an additional global distance metric, which requires the local distance metrics being similar to the global one. This global regularization allows for sharing information between clusters. Experimental results on several benchmark data sets show that the proposed CMML method outperforms single distance metric learning methods and other multi-metric learning methods while having a low computational cost.

9 Distance metric learning for k-nearest-neighbor regression

This chapter presents a distance metric learning method for k-nearest neighbors regression. We define the constraints based on triplets, which are built from the neighborhood of each training example, to learn the distance metric. The resulting optimization problem can be formulated as a convex quadratic program. Quadratic programming has as disadvantage that it does not scale well in large-scale settings. To reduce the time complexity of training, we propose a novel dual coordinate descent method for this type of problem. Experimental results on several regression data sets show that our method obtains a competitive performance when compared with the state-of-the-art distance metric learning methods, while being an order of magnitude faster.

The material of this chapter is based on the following publication:
Nguyen, B., Morell, C., and De Baets, B. (2016). Large-scale distance metric learning for k-nearest neighbors regression. *Neurocomputing*, 214:805–814

9.1. Motivation

One of the oldest and simplest regression methods is k-nearest neighbors regression (k-NNR) (Cover and Hart, 1967). The k-NNR method attributes the same weight to all neighbors, ignoring the similarity between the test example and its neighbors. To counter this issue, we can assign higher weights to more similar neighbors. The weight of each training example can be computed using a kernel function, which depends on the distance (as opposed to similarity) between itself and the test example. A common distance metric used to measure the distance between two examples is the Euclidean distance metric. However, using this distance metric may not be appropriate for every application domain, because it does not take into account the correlation between attributes and it ignores the presence of irrelevant attributes (Weinberger and Saul, 2009). The ideal distance metric should preserve the similarity relationships in the data, i.e., similar examples should be close to each other and dissimilar examples should be far away from each other. In this work, we focus on the Mahalanobis distance metric, which provides a well-studied and successful framework for distance metric learning. Using the Mahalanobis distance metric is a flexible way to learn an appropriate distance metric, thus allowing for arbitrary linear rotations and scaling of attributes.

One of the most important requirements for a distance metric learning method

is that the algorithm should be fast and scalable. Most of the previous optimization algorithms for distance metric learning tend to require a larger computation time for large-scale problems, where they may become computationally intractable. In particular, learning a Mahalanobis distance metric typically requires estimating a matrix with D^2 parameters (where D is the data dimensionality). This quadratic dependency poses a huge challenge for learning a good Mahalanobis distance metric in high-dimensional settings. Here, we consider a distance metric defined by a diagonal matrix to simplify the distance metric learning problem.

For this purpose, we extend the distance metric learning method proposed by Schultz and Joachims (2004) to the regression setting. The original method is intended to be used in information retrieval applications, but it is general enough to be applied in other contexts. This method uses a diagonal matrix and avoids costly projections of the Mahalanobis matrix onto the positive semidefinite cone. The formulation results in a quadratic program, which can be solved by standard quadratic programming solvers. However, the general-purpose solvers tend to scale poorly in the number of constraints. Motivated by these reasons, we:

(i) Extend the previous work of Schultz and Joachims (2004), so that it can be used in the context of k-NNR. We refer to the proposed method as *large-scale distance metric learning for k-nearest neighbors regression* (LDMLR).

(ii) Introduce a novel strategy to define the constraints. Instead of randomly selecting triplet constraints to satisfy an application-specific criterion, we extract the constraints from the local neighborhood of each training example, which allow us to preserve discriminative information from this neighborhood.

(iii) Propose a special solver for this type of optimization problem. The proposed method is simple to implement, and it ensures very fast training, which can be computationally tractable for large-scale data sets.

(iv) Conduct an empirical study evaluating our method on twenty data sets. The experiments show that the proposed method is comparable with the state-of-the-art distance metric learning methods in terms of regression accuracy, while being much more efficient in terms of training time.

The remainder of this chapter is organized as follows. In Section 9.2, we introduce our distance metric learning method for regression. The proposed method is based on solving a constrained optimization problem. First, we describe the selection of constraints in Subsection 9.2.1. Second, we formulate our proposal as a quadratic program in Subsection 9.2.2. To make problem solving more straightforward and effective than a general-purpose solver, we present a dual coordinate descent method in Subsection 9.2.3. In Section 9.3, we discuss the related work to highlight the main differences of our method compared to existing methods, focusing on the problems that are addressed by our method. In Section 9.4, we evaluate the capabilities of the proposed method by comparing its regression

accuracy on several public standard data sets to that of various state-of-the-art distance metric learning methods. Finally, we present some concluding remarks in Section 9.5.

9.2. Distance metric learning for regression

In this section, we focus on distance metric learning in the regression setting. In contrast to the classification setting, in this case the output space $\mathcal{Y}$ can be a set of real values. It turns out to be difficult to build two sets of pairwise constraints $\mathcal{S}$ and $\mathcal{D}$. In order to preserve the similarity relationships among the data in the transformed space, we learn a distance metric that satisfies a set of relative constraints $\mathcal{T}$. First, we present a strategy for selecting the triplet constraints making up the set $\mathcal{T}$. Then, we formulate our distance metric learning problem based on those triplet constraints. Finally, we present a coordinate gradient descent method in order to solve our distance metric learning problem.

9.2.1. Selection of triplet constraints

The major assumption underlying example-based learning methods, such as k-NN (Cover and Hart, 1967), is the commonsense principle suggesting that "similar problems have similar solutions." This "similarity hypothesis" serves as a basic inference paradigm. In the classification context, it is translated into the assertion that similar examples have similar class labels. In the regression context, it means that two examples that are close to each other in the input space are also close in the output space. The intuition behind our method is to learn a distance metric that guarantees the fulfillment of the given principle for each training example. It may be difficult to globally satisfy all constraints for each example, so we will keep track of each example $\mathbf{x}^{(i)}$ only by means of its neighborhood $\mathcal{V}(\mathbf{x}^{(i)})$, which is the set of the nearest neighbors of $\mathbf{x}^{(i)}$ in $\mathcal{X}$. More specifically, our method will identify a set of triplet constraints in the neighborhood of each example that enforce the resulting distance metric to agree with the similarity hypothesis. A distance metric induces, for any example, a total ordering of its neighborhood. Such ordering will be correct if the outputs associated with the neighbors can be ranked in the same order. That is, an example $\mathbf{x}^{(i)}$ should be closer to $\mathbf{x}^{(j)}$ than to $\mathbf{x}^{(k)}$ if $y^{(i)}$ is closer to $y^{(j)}$ than to $y^{(k)}$.

Let us define the order-preserving function F_Ω as a real-valued function on the metric space Ω associated with a distance metric d_Ω for a triplet $\delta_\Omega^{ijk} = (\mathbf{u}^{(i)}, \mathbf{u}^{(j)}, \mathbf{u}^{(k)}) \in \Omega^3$ as

$$F_\Omega(\delta_\Omega^{ijk}) = d_\Omega(\mathbf{u}^{(i)}, \mathbf{u}^{(j)}) - d_\Omega(\mathbf{u}^{(i)}, \mathbf{u}^{(k)})\,.$$

The triplets in the input space $\mathcal{X}$ and the output space $\mathcal{Y}$ are denoted as $\delta_{\mathcal{X}}^{ijk} = (\mathbf{x}^{(i)}, \mathbf{x}^{(j)}, \mathbf{x}^{(k)})$ and $\delta_{\mathcal{Y}}^{ijk} = (y^{(i)}, y^{(j)}, y^{(k)})$, respectively. In the absence of prior knowledge or information about the distance metric, we can use the Euclidean distance metric on both spaces $\mathcal{X}$ and $\mathcal{Y}$, in order to compute the order-preserving function. We define $\pi(i, j, k)$ as the indicator function given by

$$\pi(i,j,k) = \begin{cases} 1 & , \text{ if } F_{\mathcal{Y}}(\delta_{\mathcal{Y}}^{ijk}) < 0\,; \\ 1 & , \text{ if } F_{\mathcal{Y}}(\delta_{\mathcal{Y}}^{ijk}) = 0 \text{ and } F_{\mathcal{X}}(\delta_{\mathcal{X}}^{ijk}) < 0\,; \\ 0 & , \text{ otherwise}\,. \end{cases}$$

Finally, we define the set $\mathcal{T}$ of triplet constraints as:

$$\mathcal{T} = \left\{ (\mathbf{x}^{(i)}, \mathbf{x}^{(j)}, \mathbf{x}^{(k)}) \mid \mathbf{x}^{(j)}, \mathbf{x}^{(k)} \in \mathcal{V}(\mathbf{x}^{(i)}) \text{ and } \pi(i,j,k) = 1 \right\}.$$

In order to build the set $\mathcal{T}$, we need to find the neighborhood of each example in the training set. Using linear nearest neighbor search, the time complexity is quadratic in terms of the number of examples $O(s^2)$. To reduce the cost of the nearest neighbor search, for bigger data sets, we can use sophisticated tree data structures, such as Cover Tree (Beygelzimer et al., 2006), Ball Tree (Omohundro, 1989) or k-d-trees (Moore, 1991).

9.2.2. Problem formulation

Similarly as Schultz and Joachims (2004), we are interested in learning the Mahalanobis distance metric parameterized by a linear transformation $\mathbf{A}$ and a diagonal matrix $\mathbf{W}$, such that

$$d^2_{\mathbf{A},\mathbf{W}}(\mathbf{u}, \mathbf{v}) = (\mathbf{u} - \mathbf{v})^\top \mathbf{A}\mathbf{W}\mathbf{A}^\top (\mathbf{u} - \mathbf{v})\,.$$

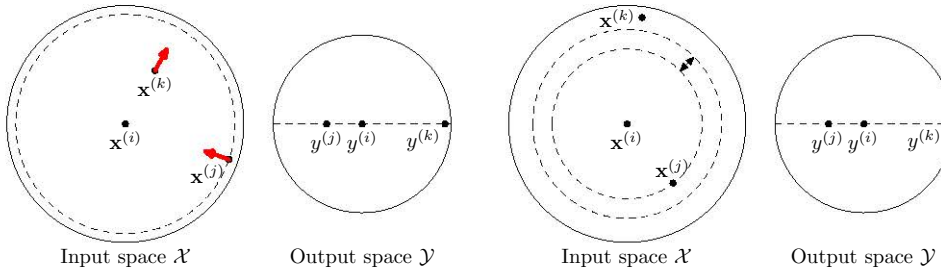
In order to guarantee that $d_{\mathbf{A},\mathbf{W}}$ is a distance metric, $\mathbf{W}$ has to be a diagonal matrix with non-negative values and $\mathbf{A}$ can be any real matrix of rank m, where $\mathbf{A} \in \mathbb{R}^{D \times m}$, $\mathbf{W} \in \mathbb{R}^{m \times m}$, and $m \leq D$. The magnitude of each diagonal element of $\mathbf{W}$ represents the relevance of the corresponding attribute in the input space after applying the linear transformation $\mathbf{A}$.

In particular, for the setting that example $\mathbf{x}^{(i)}$ is the i-th column of the matrix $\mathbf{A}$, we obtain

$$\begin{aligned} d^2_{\mathbf{A},\mathbf{W}}(\mathbf{u}, \mathbf{v}) &= \big(\mathbf{A}^\top(\mathbf{u} - \mathbf{v})\big)^\top \mathbf{W} \big(\mathbf{A}^\top(\mathbf{u} - \mathbf{v})\big) \\ &= \sum_{i=1}^{s} W_{ii} \big(\langle \mathbf{u}, \mathbf{x}^{(i)} \rangle - \langle \mathbf{v}, \mathbf{x}^{(i)} \rangle\big)^2. \end{aligned} \tag{9.1}$$

Equation (9.1) only depends on the inner products of two vectors, hence we can

apply the "kernel trick" to do the mapping implicitly by a kernel function instead of inner products (Schölkopf and Smola, 2001). Kernel functions can represent inner products in a high- or even infinite-dimensional space, which provides a flexible way to learn non-linear transformations in the input space.



(a) Before training **(b)** After training

Figure 9.1: Illustration of the intuition behind our distance metric learning method for k-NNR. examples $\mathbf{x}^{(j)}$ and $\mathbf{x}^{(k)}$ are nearest neighbors of $\mathbf{x}^{(i)}$. Before learning, the triplet constraint $(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}, \mathbf{x}^{(k)})$ is violated ($y^{(i)}$ is closer to $y^{(j)}$ than to $y^{(k)}$, but $\mathbf{x}^{(i)}$ is closer to $\mathbf{x}^{(k)}$ than to $\mathbf{x}^{(j)}$). After learning, the new distance metric induces the same ranking and example $\mathbf{x}^{(k)}$ is pushed away from $\mathbf{x}^{(j)}$ by a safe margin.

The learned distance metric should make sure that for each triplet constraint $(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}, \mathbf{x}^{(k)})$ in $\mathcal{T}$, it holds that in the transformed space the squared distance between examples $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(k)}$ is at least $\epsilon > 0$ greater than the squared distance between $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ (see Fig. 9.1). Furthermore, a good distance metric should be able to remove noisy attributes in training data, while leading to a reduced dimension. For this purpose, among all distance metrics that satisfy the set of constraints $\mathcal{T}$, we are only interested in the one that results in a minimum ℓ_1-norm of the eigenvalues of $\mathbf{AWA}^\top$. However, it is difficult to use this norm directly, so we use the squared ℓ_2-norm of the eigenvalues of $\mathbf{AWA}^\top$, which is equal to the squared Frobenius norm of $\mathbf{AWA}^\top$. Finally, we aim to solve the following optimization problem

$$\begin{aligned} \underset{\mathbf{W} \succeq 0}{\text{minimize}} \quad & \tfrac{1}{2}\|\mathbf{AWA}^\top\|_F^2 + C \textstyle\sum_{(ijk)} \xi_{ijk} \\ \text{subject to} \quad & \forall (\mathbf{x}^{(i)}, \mathbf{x}^{(j)}, \mathbf{x}^{(k)}) \in \mathcal{T}\,, \\ & d^2_{\mathbf{A},\mathbf{W}}(\mathbf{x}^{(i)}, \mathbf{x}^{(k)}) - d^2_{\mathbf{A},\mathbf{W}}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \geq \epsilon - \xi_{ijk}\,, \\ & \xi_{ijk} \geq 0\,, \end{aligned} \tag{9.2}$$

where ξ_{ijk} are slack variables used to penalize the constraints that cannot be satisfied, and the parameter $C > 0$ controls the trade-off between the slack variable penalties and the squared Frobenius norm.

Furthermore, the distance metric $d_{\mathbf{A},\mathbf{W}}$ can be expressed in the following

form:

$$d^2_{\mathbf{A},\mathbf{W}}(\mathbf{u},\mathbf{v}) = \mathbf{w}^\top \Big[(\mathbf{A}^\top \mathbf{u} - \mathbf{A}^\top \mathbf{v}) \circ (\mathbf{A}^\top \mathbf{u} - \mathbf{A}^\top \mathbf{v}) \Big] ,$$

where $\mathbf{w}$ is the diagonal vector of the matrix $\mathbf{W}$. Let us define:

$$\begin{aligned} \mathbf{L} &= (\mathbf{A}^\top \mathbf{A}) \circ (\mathbf{A}^\top \mathbf{A}) , \\ \boldsymbol{\Delta}^{\mathbf{u},\mathbf{v}} &= (\mathbf{A}^\top \mathbf{u} - \mathbf{A}^\top \mathbf{v}) \circ (\mathbf{A}^\top \mathbf{u} - \mathbf{A}^\top \mathbf{v}) , \\ \mathbf{z}^{(ijk)} &= \boldsymbol{\Delta}^{\mathbf{x}^{(i)},\mathbf{x}^{(k)}} - \boldsymbol{\Delta}^{\mathbf{x}^{(i)},\mathbf{x}^{(j)}} . \end{aligned}$$

Since $\mathbf{z}^{(ijk)}$ are enumerable, we can change the superscripts (ijk) to an enumerable set $\{1, 2, \ldots, n\}$. Hence, problem (9.2) can be rewritten as:

$$\begin{array}{ll} \underset{\mathbf{w}}{\text{minimize}} & \frac{1}{2}\mathbf{w}^\top \mathbf{L}\mathbf{w} + C \sum_{i=1}^n \xi_i \\ \text{subject to} & \mathbf{w}^\top \mathbf{z}^{(i)} \geq \epsilon - \xi_i , \\ & \xi_i \geq 0, \quad i = 1, ..., n , \\ & w_j \geq 0, \quad j = 1, ..., m . \end{array} \tag{9.3}$$

Note that $\mathbf{L}$ is positive definite, therefore problem (9.3) is a convex quadratic program.

For the constraints, we introduce multipliers $\boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{t} \geq 0$ for the Lagrangian function $\mathcal{L}\colon \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, and obtain:

$$\begin{aligned} \mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{t}) = \frac{1}{2}\mathbf{w}^\top \mathbf{L}\mathbf{w} + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \lambda_i (\mathbf{w}^\top \mathbf{z}^{(i)} - \epsilon + \xi_i) \\ - \sum_{i=1}^n \mu_i \xi_i - \sum_{i=1}^m t_i w_i . \end{aligned}$$

To minimize the objective function in (9.3), we have to find the saddle point of the function $\mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{t})$, i.e., we have to minimize over the primal variables $\mathbf{w}, \boldsymbol{\xi}$ and maximize over the dual variables $\boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{t}$. Setting the derivatives with respect to the primal variables equal to zero yields the following equations:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{t}) = 0 ,$$

resulting in

$$\mathbf{w} = \mathbf{L}^{-1} \left(\sum_{i=1}^n \lambda_i \mathbf{z}^{(i)} + \mathbf{t} \right) , \tag{9.4}$$

and

$$\frac{\partial \mathcal{L}}{\partial \xi_i}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{t}) = C - \lambda_i - \mu_i = 0\,,$$

resulting in

$$C = \lambda_i + \mu_i\,. \tag{9.5}$$

Substituting these expressions back into $\mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{t})$ leads to the Wolfe dual optimization problem which has as objective function (to be maximized):

$$\begin{aligned}
g(\boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{t}) &= \frac{1}{2}\mathbf{w}^\top \mathbf{L}\mathbf{w} + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\lambda_i(\mathbf{w}^\top \mathbf{z}^{(i)} - \epsilon + \xi_i) - \sum_{i=1}^{n}\mu_i\xi_i - \sum_{i=1}^{m} t_i w_i \\
&= \frac{1}{2}\mathbf{w}^\top \mathbf{L}\mathbf{w} + \underbrace{\sum_{i=1}^{n}\xi_i(C - \lambda_i - \mu_i)}_{=0} - \sum_{i=1}^{n}\lambda_i \mathbf{w}^\top \mathbf{z}^{(i)} + \epsilon\sum_{i=1}^{n}\lambda_i - \mathbf{w}^\top \mathbf{t} \\
&= \frac{1}{2}\mathbf{w}^\top \mathbf{L}\mathbf{w} - \mathbf{w}^\top \mathbf{L}\underbrace{\mathbf{L}^{-1}\left(\sum_{i=1}^{n}\lambda_i \mathbf{z}^{(i)} + \mathbf{t}\right)}_{=\mathbf{w}} + \epsilon\sum_{i=1}^{n}\lambda_i \\
&= -\frac{1}{2}\mathbf{w}^\top \mathbf{L}\mathbf{w} + \epsilon\sum_{i=1}^{n}\lambda_i \\
&= -\frac{1}{2}\left(\sum_{i=1}^{n}\lambda_i(\mathbf{z}^{(i)})^\top\right)\mathbf{L}^{-1}\left(\sum_{i=1}^{n}\lambda_i \mathbf{z}^{(i)}\right) - \left(\frac{1}{2}\mathbf{t}^\top \mathbf{L}^{-1}\mathbf{t}\right) \\
&\qquad - \left(\mathbf{t}^\top \mathbf{L}^{-1}\sum_{i=1}^{n}\lambda_i \mathbf{z}^{(i)}\right) + \epsilon\sum_{i=1}^{n}\lambda_i\,.
\end{aligned}$$

To simplify the notation of the function $g(\boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{t})$, let us define the functions $h(\mathbf{t}) = \frac{1}{2}\mathbf{t}^\top \mathbf{L}^{-1}\mathbf{t}$ and $l(\mathbf{t}, \boldsymbol{\lambda}) = \mathbf{t}^\top \mathbf{L}^{-1}\sum_{i=1}^{n}\lambda_i \mathbf{z}^{(i)}$. We also define the matrix $\mathbf{H} \in \mathbb{R}^{n\times n}$ given by $H_{ij} = (\mathbf{z}^{(i)})^\top \mathbf{L}^{-1}(\mathbf{z}^{(j)})$. From (9.5), since $\mu_i \geq 0$ implies $\lambda_i \leq C$, the primal problem (9.3) is converted into the dual problem:

$$\begin{aligned}
\underset{\boldsymbol{\lambda}, \mathbf{t}}{\text{minimize}} \quad & f(\boldsymbol{\lambda}, \mathbf{t}) = \tfrac{1}{2}\boldsymbol{\lambda}^\top \mathbf{H}\boldsymbol{\lambda} + h(\mathbf{t}) + l(\mathbf{t}, \boldsymbol{\lambda}) - \epsilon\textstyle\sum_{i=1}^{n}\lambda_i \\
\text{subject to} \quad & C \geq \lambda_i \geq 0, \quad i = 1, \ldots, n\,, \\
& t_j \geq 0, \quad j = 1, \ldots, m\,.
\end{aligned} \tag{9.6}$$

Problem (9.6) can be solved by standard quadratic programming solvers, such as CPLEX (ILOG, 2012) and MOSEK (Mosek, 2010), but they quickly tend to scale poorly in the number of constraints. In the following subsection, we will describe a dual coordinate descent method to solve this problem more efficiently and faster than a general-purpose solver.

9.2.3. Learning a distance metric with coordinate descent

The coordinate descent method is a powerful optimization technique that has been used to solve a number of machine learning tasks, such as linear SVMs (Hsieh et al., 2008) and Lasso regression (Friedman et al., 2007). In general, the coordinate descent method is efficient if each single iteration can be performed with a minimal cost. The convergence analysis of coordinate descent methods can be found in (Shalev-Shwartz and Zhang, 2013; Luo and Tseng, 1992). In this subsection, we describe our coordinate descent method for solving problem (9.6).

Notations

We introduce some notations to express the optimization procedure conveniently. The optimization procedure starts from an initial point $(\boldsymbol{\lambda}^0, \mathbf{t}^0) \in \mathbb{R}^{n+m}$ and generates a sequence of vectors $\{(\boldsymbol{\lambda}^k, \mathbf{t}^k)\}_{k=0}^{\infty}$. We refer to an outer iteration as the process of updating from $(\boldsymbol{\lambda}^k, \mathbf{t}^k)$ to $(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k+1})$. At each outer iteration, $(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k+1})$ is constructed by sequentially updating each component of $(\boldsymbol{\lambda}^k, \mathbf{t}^k)$. In other words, in each outer iteration we have $n+m$ inner iterations, so that $\lambda_1, \ldots, \lambda_n, t_1, \ldots, t_m$ are updated. Thus, each outer iteration generates vectors $(\boldsymbol{\lambda}^{k,i}, \mathbf{t}^k) \in \mathbb{R}^{n+m}$, for $i = 1, \ldots, n+1$, and $(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j}) \in \mathbb{R}^{n+m}$, for $j = 1, \ldots, m+1$, such that $\boldsymbol{\lambda}^{k,1} = \boldsymbol{\lambda}^k$, $\boldsymbol{\lambda}^{k,n+1} = \boldsymbol{\lambda}^{k+1}$, $\mathbf{t}^{k,1} = \mathbf{t}^k$, $\mathbf{t}^{k,m+1} = \mathbf{t}^{k+1}$, and

$$\boldsymbol{\lambda}^{k,i} = \left(\lambda_1^{k+1}, \ldots, \lambda_{i-1}^{k+1}, \lambda_i^k, \ldots, \lambda_n^k\right)^\top ,$$
$$\mathbf{t}^{k,j} = \left(t_1^{k+1}, \ldots, t_{j-1}^{k+1}, t_j^k, \ldots, t_m^k\right)^\top .$$

First, we update each component of $\boldsymbol{\lambda}^k$ and then we update each component of $\mathbf{t}^k$. We describe in detail the method for updating from $(\boldsymbol{\lambda}^{k,i}, \mathbf{t}^k)$ to $(\boldsymbol{\lambda}^{k,i+1}, \mathbf{t}^k)$ and from $(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j})$ to $(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j+1})$ in each inner iteration.

Solving for $(\boldsymbol{\lambda}^{k,i+1}, \mathbf{t}^k)$

To update from $(\boldsymbol{\lambda}^{k,i}, \mathbf{t}^k)$ to $(\boldsymbol{\lambda}^{k,i+1}, \mathbf{t}^k)$, we solve the following one-variable subproblem:

$$\begin{aligned} \underset{d \in \mathbb{R}}{\text{minimize}} \quad & f\left(\lambda_1^{k+1}, \ldots, \lambda_{i-1}^{k+1}, d + \lambda_i^k, \lambda_{i+1}^k, \ldots, \lambda_n^k, t_1^k, \ldots, t_m^k\right) \\ & \equiv f(\boldsymbol{\lambda}^{k,i} + d\mathbf{e}^{(i)}, \mathbf{t}^k) \\ \text{subject to} \quad & C \geq d + \lambda_i^k \geq 0\,, \end{aligned} \tag{9.7}$$

where $\mathbf{e}^{(i)} = (0, \ldots, 1, \ldots, 0)^\top$. The objective function in (9.7) is a simple quadratic function of variable d:

$$f(\boldsymbol{\lambda}^{k,i} + d\mathbf{e}^{(i)}, \mathbf{t}^k) = \frac{1}{2} H_{ii} d^2 + \nabla_i f(\boldsymbol{\lambda}^{k,i}, \mathbf{t}^k) d + \text{constant},$$

where $\nabla_i f(\boldsymbol{\lambda}^{k,i}, \mathbf{t}^k)$ is the i-th component of the gradient ∇f, which is defined as:

$$\begin{aligned}
\nabla_i f(\boldsymbol{\lambda}^{k,i}, \mathbf{t}^k) &= \frac{1}{2} \sum_{j=1}^{n} H_{ij} \lambda_j^{k,i} + \frac{1}{2} \sum_{j=1}^{n} H_{ji} \lambda_j^{k,i} + \frac{\partial l}{\partial \lambda_i^{k,i}} (\boldsymbol{\lambda}^{k,i}, \mathbf{t}^k) - \epsilon \\
&= \sum_{j=1}^{n} H_{ij} \lambda_j^{k,i} + (\mathbf{z}^{(i)})^\top \mathbf{L}^{-1} \mathbf{t}^k - \epsilon \\
&= \sum_{j=1}^{n} (\mathbf{z}^{(i)})^\top \mathbf{L}^{-1} \mathbf{z}^{(j)} \lambda_j^{k,i} + (\mathbf{z}^{(i)})^\top \mathbf{L}^{-1} \mathbf{t}^k - \epsilon \\
&= (\mathbf{z}^{(i)})^\top \left[\mathbf{L}^{-1} \left(\sum_{j=1}^{n} \lambda_j^{k,i} \mathbf{z}^{(j)} + \mathbf{t}^k \right) \right] - \epsilon \\
&= (\mathbf{z}^{(i)})^\top \mathbf{w} - \epsilon \,. \qquad (9.8)
\end{aligned}$$

It is easy to see that (9.7) has as optimal solution $d = 0$ (i.e., no need to update λ_i^k) if and only if:

$$\nabla_i^P f(\boldsymbol{\lambda}^{k,i} + d\mathbf{e}^{(i)}, \mathbf{t}^k) = 0 \,, \qquad (9.9)$$

where $\nabla_i^P f(\boldsymbol{\lambda}, \mathbf{t})$ is the projected gradient (Lin and Moré, 1999), which is calculated as:

$$\nabla_i^P f(\boldsymbol{\lambda}, \mathbf{t}) = \begin{cases} \nabla_i f(\boldsymbol{\lambda}, \mathbf{t}) & , \text{ if } 0 < \lambda_i < C \,; \\ \min(0, \nabla_i f(\boldsymbol{\lambda}, \mathbf{t})) & , \text{ if } \lambda_i = 0 \,; \\ \max(0, \nabla_i f(\boldsymbol{\lambda}, \mathbf{t})) & , \text{ if } \lambda_i = C \,. \end{cases}$$

We only update in case (9.9) does not hold, i.e., the projected gradient $\nabla_i^P f(\boldsymbol{\lambda}, \mathbf{t}) \neq 0$. Since $f(\boldsymbol{\lambda}^{k,i} + d\mathbf{e}^{(i)}, \mathbf{t}^k)$ is a simple quadratic function of a single variable d, the optimal point that minimizes the objective function $f(\boldsymbol{\lambda}^{k,i} + d\mathbf{e}^{(i)}, \mathbf{t}^k)$ when $H_{ii} > 0$ is:

$$d = -\frac{\nabla_i f(\boldsymbol{\lambda}^{k,i}, \mathbf{t}^k)}{H_{ii}} \,.$$

Subsequently, we need to truncate the argument $(\lambda_i^k + d)$ into the interval $[0, C]$ to obtain a feasible solution:

$$\lambda_i^{k,i+1} = \min\left(\max\left(\lambda_i^k - \frac{\nabla_i f(\boldsymbol{\lambda}^{k,i}, \mathbf{t}^k)}{H_{ii}}, 0\right), C\right). \tag{9.10}$$

If $H_{ii} = 0$, i.e., $(\mathbf{z}^{(i)})^\top \mathbf{L}^{-1}(\mathbf{z}^{(i)}) = 0$, then $\mathbf{z}^{(i)} = \mathbf{0}$. Therefore, due to (9.8), we have that $\nabla_i f(\boldsymbol{\lambda}^{k,i}, \mathbf{t}^k) = -\epsilon$, and the optimal solution of (9.7) is $d = C - \lambda_i^k$, thus $\lambda_i^{k,i+1} = C$. We can also include this case in (9.10) by setting $1/0 = +\infty$.

To calculate $\nabla_i f(\boldsymbol{\lambda}^{k,i}, \mathbf{t}^k)$, we need to update $\mathbf{w}$ throughout the coordinate descent procedure. This can be easily computed as follows:

$$\mathbf{w} \leftarrow \mathbf{w} + d\,\mathbf{L}^{-1}\mathbf{z}^{(i)}.$$

To reduce the complexity of this process, we can pre-compute the values of $\mathbf{L}^{-1}\mathbf{z}^{(i)}$. Hence, this update only takes $O(m)$ operations to update $\mathbf{w}$ in each inner iteration. Finally, we set the value of $(\boldsymbol{\lambda}^{k,i+1}, \mathbf{t}^k)$ as:

$$(\boldsymbol{\lambda}^{k,i+1}, \mathbf{t}^k) \leftarrow (\boldsymbol{\lambda}^{k,i} + d\mathbf{e}^{(i)}, \mathbf{t}^k).$$

Solving for $(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j+1})$

To update from $(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j})$ to $(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j+1})$, we solve the following one-variable subproblem:

$$\begin{aligned} \underset{d\in\mathbb{R}}{\text{minimize}} \quad & f\Big(\lambda_1^{k+1}, \ldots, \lambda_n^{k+1}, t_1^{k+1}, \ldots, t_{j-1}^{k+1}, d + t_j^k, t_{j+1}^k, \ldots, t_m^k\Big) \\ & \equiv f(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j} + d\mathbf{e}^{(j)}) \\ \text{subject to} \quad & d + t_j^k \geq 0, \end{aligned} \tag{9.11}$$

where $\mathbf{e}^{(j)} = (0, \ldots, 1, \ldots, 0)^\top$. The objective function (9.11) is also a simple quadratic function of a single variable d:

$$f(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j} + d\mathbf{e}^{(j)}) = \frac{1}{2} L_{jj}^{-1} d^2 + \nabla_{(n+j)} f(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j}) d + \text{constant},$$

where $\nabla_{(n+j)} f(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j})$ is the $(n+j)$-th component of the gradient ∇f, which is defined as:

$$\begin{aligned} \nabla_{(n+j)} f(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j}) &= \frac{1}{2}\sum_{i=1}^{m} L_{ij}^{-1} t_i^{k,j} + \frac{1}{2}\sum_{i=1}^{m} L_{ji}^{-1} t_i^{k,j} + \frac{\partial h}{\partial t_j^{k,j}}(\mathbf{t}^{k,j}) \\ &= \sum_{i=1}^{m} L_{ji}^{-1} t_i^{k,j} + \left(\mathbf{L}^{-1}\sum_{i=1}^{n} \lambda_i^{k+1}\mathbf{z}^{(i)}\right)_j \end{aligned}$$

$$= \sum_{i=1}^{m} L_{ji}^{-1} t_i^{k,j} + \Big(T(\boldsymbol{\lambda}^{k+1})\Big)_j \,, \tag{9.12}$$

where $T(\boldsymbol{\lambda}) = \mathbf{L}^{-1} \sum_{i=1}^{n} \lambda_i \mathbf{z}^{(i)}$. As for the objective function in (9.7), we do not need to update t_j^k if and only if:

$$\nabla^P_{(n+j)} f(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j} + d\mathbf{e}^{(j)}) = 0 \,, \tag{9.13}$$

where $\nabla^P_{(n+j)} f(\boldsymbol{\lambda}, \mathbf{t})$ is the projected gradient (Lin and Moré, 1999), which is calculated as:

$$\nabla^P_{(n+j)} f(\boldsymbol{\lambda}, \mathbf{t}) = \begin{cases} \nabla_{(n+j)} f(\boldsymbol{\lambda}, \mathbf{t}) & , \text{ if } t_j > 0 \,; \\ \min(0, \nabla_{(n+j)} f(\boldsymbol{\lambda}, \mathbf{t})) & , \text{ if } t_j = 0 \,. \end{cases}$$

Hence, when (9.13) does not hold, the optimal solution that minimizes the function $f(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j} + d\mathbf{e}^{(j)})$ is given by:

$$d = -\frac{\nabla_{(n+j)} f(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j})}{L_{jj}^{-1}} \,.$$

Subsequently, we need to truncate the argument $(t_j^k + d)$ to the interval $[0, +\infty[$ to obtain a feasible solution of (9.11):

$$t_j^{k,j+1} = \max\left(t_j^k - \frac{\nabla_{(n+j)} f(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j})}{L_{jj}^{-1}}, 0\right) \,.$$

We can calculate $\nabla_{(n+j)} f(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j})$ in a time complexity of $O(m)$. Since the first term $\sum_{i=1}^{m} L_{ji}^{-1} t_i^{k,j}$ in (9.12) can be computed in $O(m)$ and the second term $\big(T(\boldsymbol{\lambda})\big)_j$ can be computed in $O(1)$ if we pre-compute $T(\boldsymbol{\lambda})$ and update it throughout the coordinate descent procedure after updating each $\boldsymbol{\lambda}^{k,i}$, this only takes a time complexity of $O(m)$:

$$T(\boldsymbol{\lambda}) \leftarrow T(\boldsymbol{\lambda}) + (\lambda_i^{k+1} - \lambda_i^k) \mathbf{L}^{-1} \mathbf{z}^{(i)} \,.$$

Finally, we set the value of $(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j+1})$ as:

$$(\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j+1}) \leftarrow (\boldsymbol{\lambda}^{k+1}, \mathbf{t}^{k,j} + d\mathbf{e}^{(j)}) \,.$$

After updating $\mathbf{t}^k$, we must update $\mathbf{w}$ as follows:

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{L}^{-1}(\mathbf{t}^{k+1} - \mathbf{t}^k) \,.$$

Random permutation of subproblems

The above coordinate descent method solves the one-variable subproblems in the order of $\lambda_1, \ldots, \lambda_n, t_1, \ldots, t_m$. As discussed by Chang et al. (2008), when the components of $\boldsymbol{\lambda}$ or the components of $\mathbf{t}$ are correlated, the order of the components may affect the training speed. To improve the convergence, we randomly permute the subproblems at each outer iteration, i.e., at the k-th outer iteration, we construct a random permutation π_λ of $\{1, \ldots, n\}$ and a random permutation π_t of $\{1, \ldots, m\}$, then sequentially solve the subproblems in the order of $\lambda_{\pi_\lambda(1)}, \ldots, \lambda_{\pi_\lambda(n)}, t_{\pi_t(1)}, \ldots, t_{\pi_t(m)}$.

The initial value for $\mathbf{w}$ can be set to $\mathbf{0}$ by using $\boldsymbol{\lambda} = \mathbf{0}$ and $\mathbf{t} = \mathbf{0}$. A brief description is given in Algorithm 9. Our method has time complexity of $O(nm+m^2)$, which is much more efficient than $O(n^3 + nm + n^2 m)$, the time complexity of the normally used coordinate descent method in each outer iteration.

Algorithm 9 A dual coordinate descent method for regression (LDMLR)

- Given initial values of $\boldsymbol{\lambda}, \mathbf{t}$ and the corresponding
  $\mathbf{w} \leftarrow \mathbf{L}^{-1}\left(\sum_{i=1}^n \lambda_i \mathbf{z}^{(i)} + \mathbf{t}\right)$
  $\mathbf{T} \leftarrow \mathbf{L}^{-1} \sum_{i=1}^n \lambda_i \mathbf{z}^{(i)}$
- **While** $\boldsymbol{\lambda}, \mathbf{t}$ are not optimal (outer iteration)
  - Randomly permute $(1, \ldots, n)$ to $(\pi(1), \ldots, \pi(n))$
  - (a) **For** $i = \pi(1), \ldots, \pi(n)$, (inner iteration)
    - (1) $G_\lambda \leftarrow (\mathbf{z}^{(i)})^\top \mathbf{w} - \epsilon$
    - (2) $P_\lambda \leftarrow \begin{cases} G_\lambda & , \text{ if } 0 < \lambda_i < C\,; \\ \min(0, G_\lambda) & , \text{ if } \lambda_i = 0\,; \\ \max(0, G_\lambda) & , \text{ if } \lambda_i = C\,. \end{cases}$
    - (3) **If** $|P_\lambda| \neq 0$, **then**
      $\lambda_i^{old} \leftarrow \lambda_i$
      $\lambda_i \leftarrow \min\left(\max\left(\lambda_i - \frac{G_\lambda}{H_{ii}}, 0\right), C\right)$
      $\mathbf{w} \leftarrow \mathbf{w} + (\lambda_i - \lambda_i^{old})\mathbf{L}^{-1}\mathbf{z}^{(i)}$
      $\mathbf{T} \leftarrow \mathbf{T} + (\lambda_i - \lambda_i^{old})\mathbf{L}^{-1}\mathbf{z}^{(i)}$
  - (b) $\mathbf{t}^{old} \leftarrow \mathbf{t}$
  - Randomly permute $(1, \ldots, m)$ to $(\pi(1), \ldots, \pi(m))$
  - (c) **For** $j = \pi(1), \ldots, \pi(m)$, (inner iteration)
    - (1) $G_t \leftarrow \sum_{i=1}^m L_{ji}^{-1} t_i^{k,j} + T_j$
    - (2) $P_t \leftarrow \begin{cases} G_t & , \text{ if } t_j > 0\,; \\ \min(0, G_t) & , \text{ if } t_j = 0\,. \end{cases}$
    - (3) **If** $|P_t| \neq 0$, **then**
      $t_j \leftarrow \max\left(t_j - \frac{G_t}{L_{jj}^{-1}}, 0\right)$
  - (d) $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{L}^{-1}(\mathbf{t} - \mathbf{t}^{old})$

9.3. Related work

In the literature, a number of methods have been proposed recently to learn a Mahalanobis distance metric (Yang and Xu, 2016; Miao et al., 2015; Wang et al., 2014b; Baghshah and Shouraki, 2010a; Zhang et al., 2012a). However, there are only few methods that are developed to be used in the regression setting. Next, we review some distance metric learning methods that are closely related to our method.

Weinberger and Tesauro (2007) have successfully incorporated distance metric learning into the kernel function, leading to a great improvement of the regression accuracy. The main idea of their method, which is called metric learning for kernel regression (MLKR), is to learn a Mahalanobis matrix for a Gaussian kernel via the minimization of the leave-one-out regression error. Inspired by MLKR, Huang and Sun (2013) introduced kernel regression with sparse metric learning (KR_SML). To reduce the risk of overfitting, KR_SML enforces the sparsity constraint on the Mahalanobis matrix. MLKR and KR_SML can perform well on small data sets, however, they may suffer from local optima since the problems are not convex.

Assi et al. (2014) adapted the large margin nearest neighbor method (Weinberger and Saul, 2009) for regression settings (LMNNR). It learns a Mahalanobis matrix by solving a semidefinite program, and it has the same formulation as the original method for classification. The optimization process is also based on triplet constraints, but they only focus on the violated triplets. That is, they look for triplets that have an ordering in the input space $\mathcal{X}$ that is different from the ordering in the output space $\mathcal{Y}$. The main drawback of this work is the lack of regularization. It may lead to overfitting in case of high-dimensional data sets.

Problem (9.2) resulting from our formulation has the same form as the one proposed by Schultz and Joachims (2004). Their work uses a random procedure to select triplet constraints. That is, it randomly selects three examples $\mathbf{x}^{(i)}$, $\mathbf{x}^{(j)}$, and $\mathbf{x}^{(k)}$ from the training set. If $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ are from the same class and $\mathbf{x}^{(k)}$ is from a different class, then they add the triplet $(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}, \mathbf{x}^{(k)})$ to $\mathcal{T}$. This selection may lead to a loss of the information contained in the discarded training examples. However, in our work, we propose a different way to select the triplet constraints. We just focus on the neighborhood of each example, and try to preserve its local similarity relationships.

Taking into account the similarities of our formulation with support vector machines (SVM) (Schölkopf and Smola, 2001), we proposed a solver that is inspired by the coordinate descent method used to train large-scale linear SVM (Hsieh et al., 2008). The authors of the latter work also solved the dual problem with the coordinate descent method, and the key technique for making coordinate descent iterations fast is to keep track of the vector $\mathbf{w}$ during optimization. The main

difference with (Hsieh et al., 2008) is that in our formulation the first term of the objective function contains an embedded positive definite matrix $\mathbf{L}$ and the additional constraints $w_j \geq 0$, for $j = 1, \ldots, m$. Therefore, our problem is more difficult to solve than the one in (Hsieh et al., 2008).

9.4. Experiments

In this section, we present some experiments to highlight the advantages of the proposed method in regression settings. We compare the performance of **LDMLR** with **LMNNR** (Assi et al., 2014), **MLKR** (Weinberger and Tesauro, 2007), **MLRC** (Schultz and Joachims, 2004) and k-NNR using the Euclidean distance metric (**Euclidean**). All methods were implemented in Matlab, and the experiments were carried out on a PC with 4 Intel Core i5-3570 CPUs (3.40 GHz) and 8GB RAM.

9.4.1. Data description and configuration

In our experiments, we used twenty regression data sets (see Table 9.1 for a brief description). These data sets represent an important challenge for the selected methods. The first 16 data sets were taken from the *Data for Evaluating Learning in Valid Experiments* (DELVE)[1] collection. These data sets were generated by two synthetic robot arms[2]. Half of the sixteen data sets contain 32 input attributes and the other half contain 8 input attributes. Each data set was randomly split into four disjoint sets containing 2048 examples, 1024 examples were used for training and 1024 examples were used for testing. The final result for each data set was the average of the four individual runs. We chose the DELVE data sets for assessing the performance of our method since there are several published papers on these data sets (Weinberger and Tesauro, 2007; Huang and Sun, 2013; Williams and Rasmussen, 1996). The last four data sets were taken from the *Knowledge Extraction based on Evolutionary Learning* (KEEL) (Triguero et al., 2017) machine learning repository[3]. These data sets cover a range from 3 to 85 attributes and from 2178 to 22784 examples. The KEEL package randomly splits each data set to perform 5-cross validation. We conducted experiments on the KEEL data sets to also evaluate our method on more complex problems. For LDMLR, the hyperparameter ϵ is set to $\epsilon = 1$.

The predicted output value $\hat{y}^{(j)}$ for a test example $\mathbf{x}^{(j)}$ is computed as the

[1] DELVE: http://www.cs.toronto.edu/~delve/data/datasets.html
[2] For more details on the specific data sets, see:
http://www.cs.toronto.edu/~delve/data/pumadyn/desc.html
and http://www.cs.toronto.edu/~delve/data/kin/desc.html
[3] KEEL: http://sci2s.ugr.es/keel/datasets.php

locally weighted average of the values of its nearest neighbors in the training set using the Gaussian kernel, which is defined as:

$$K(\mathbf{u}, \mathbf{v}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(- \frac{d(\mathbf{u}, \mathbf{v})}{\sigma^2} \Big) ,$$

where d is the learned Mahalanobis distance metric. Thus, we can compute $\hat{y}^{(j)}$ as:

$$\hat{y}^{(j)} = \frac{\sum_{i=1}^{s} \mathbf{1}\big(\mathbf{x}^{(i)} \in \mathcal{V}(\mathbf{x}^{(j)})\big) y^{(i)} K\big(\mathbf{x}^{(j)}, \mathbf{x}^{(i)}\big)}{\sum_{i=1}^{s} \mathbf{1}\big(\mathbf{x}^{(i)} \in \mathcal{V}(\mathbf{x}^{(j)})\big) K\big(\mathbf{x}^{(j)}, \mathbf{x}^{(i)}\big)} ,$$

where $\mathbf{1}(.)$ is a function that takes value 1 if its argument is true, otherwise it takes value 0. In our experiments, we only considered $\sigma = 1$ and five neighbors of each test example. For LDMLR, we also used five nearest neighbors to get the triplet constraints; the transformation matrix $\mathbf{A}$ was set to the identity matrix. To improve the quadratic nearest neighbor search for some big data sets, we used a sophisticated data structure named Ball Tree (Omohundro, 1989). The trade-off parameter C in the objective function of LDMLR and MLRC was tuned by cross-validation on the training set considering as set of values $\{0.001, 0.01, \ldots, 1000\}$. For MLRC, we use a dual active-set method to solve the optimization problem. We adapt MLRC to be used in regression settings by randomly selecting triplets of the type $(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}, \mathbf{x}^{(k)})$, where $|y^{(i)} - y^{(j)}| < |y^{(i)} - y^{(k)}|$.

To compare the performance of the proposed method with other related methods, we use the root mean squared error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{s_t} \sum_{j=1}^{s_t} (y^{(j)} - \hat{y}^{(j)})^2} ,$$

where s_t denotes the number of test examples.

9.4.2. Methodology

To compare the performance of several methods over multiple data sets, we follow the recommendation in (Demšar, 2006; García and Herrera, 2008) associated with the computation of the p-value. First, we apply the non-parametric Friedman test (Friedman, 1940), which is equivalent to the repeated-measures ANOVA (Fisher, 1959), to test the null hypothesis that all methods obtain the same results on average. After the Friedman test rejects the null hypothesis, we can apply a post-hoc test to analyze which methods perform significantly different from the best-ranked method. To this end, we apply the Bonferroni-Dunn test (Dunn, 1961), which permits to identify significant differences between a control method (in our case, the best-ranked method) and other methods. One method performs

Table 9.1: Description of the data sets used in the experiment

#	Data set	Features	Examples	#	Data set	Features	Examples
1.	kin8fh	8	1024	11.	puma8nh	8	1024
2.	kin8fm	8	1024	12.	puma8nm	8	1024
3.	kin8nh	8	1024	13.	puma32fh	32	1024
4.	kin8nm	8	1024	14.	puma32fm	32	1024
5.	kin32fh	32	1024	15.	puma32nh	32	1024
6.	kin32fm	32	1024	16.	puma32nm	32	1024
7.	kin32nh	32	1024	17.	house	16	22784
8.	kin32nm	32	1024	18.	pole	25	14998
9.	puma8fh	8	1024	19.	quake	3	2178
10.	puma8fm	8	1024	20.	tic	85	9822

significantly different from the best-ranked method if the corresponding average rank differs by, at least, a critical distance (CD), which is calculated as:

$$\mathrm{CD} = q_\alpha \times \sqrt{\frac{n_c(n_c+1)}{6n_t}}, \tag{9.14}$$

where n_c and n_t are the number of methods and the number of data sets, respectively, and q_α is the critical value (Sheskin, 2007). Finally, we use Holm's step-down procedure (Holm, 1979) to complement the multiple comparison statistical analysis.

9.4.3. Experimental results and discussion

Table 9.4 shows the training time for each method on each data set. Clearly, LDMLR is significantly faster than LMNNR, MLKR, and MLRC. In Table 9.3, we highlight the lowest RMSE on each data set and the average rank (**Rank**) of each method according to the Friedman test at a confidence level of 0.05. Since the p-value for the Friedman test was 0.00194, we rejected the null hypothesis that all competing methods obtained the same results on average. Therefore, to detect which distance metric learning method performed significantly different from the best-ranked method (i.e., LDMLR), we applied the Bonferroni-Dunn test at a confidence level of $\alpha = 0.05$. The performance of two methods is significantly different if their corresponding average ranks differ by at least the critical difference:

$$\mathrm{CD} = q_\alpha \times \sqrt{\frac{n_c(n_c+1)}{6n_t}} = 2.498 \times \sqrt{\frac{5\times 6}{6\times 20}} = 1.249\,.$$

Figure 9.2 visualizes the significant differences among the RMSE values of the different distance metric learning methods. Any distance metric learning method with a rank outside this marked area is significantly different from the control method (i.e., LDMLR).

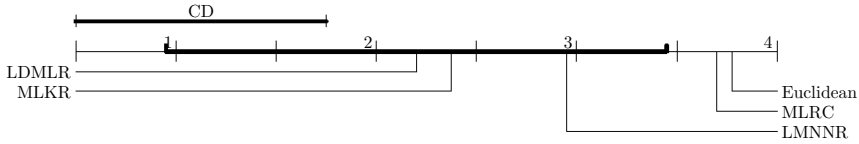


Figure 9.2: Visualization of the post-hoc Bonferroni-Dunn test of RMSE.

We also applied Holm's step-down procedure at a confidence level of $\alpha = 0.05$ to compare the best-ranked method (i.e., LDMLR) with the remaining methods. Table 9.2 presents the z-value, p-value, and adjusted α for the Holm test. The methods are ordered with respect to the p-value. The Holm test rejected the first and second hypothesis (i.e., Euclidean and MLRC obtain the same results on average as LDMLR) since the corresponding p-value was smaller than the adjusted α. But the third and fourth hypothesis (i.e., RLMNN and MLKR obtain the same results on average as LDMLR) could not be rejected as the corresponding p-value was greater than the adjusted α.

Table 9.2: Holm's post-hoc test for the competing methods with $\alpha = 0.05$ (control method: LDMLR).

Method	z-value	p-value	Holm's adjusted α	Hypothesis
Euclidean	3.1500	0.0016	0.0125	Rejected
MLRC	3.0000	0.0027	0.0167	Rejected
RLMNN	1.5000	0.1336	0.0250	Accepted
MLKR	0.3500	0.7263	0.0500	Accepted

The statistical results allow us to conclude some trends:

(i) Our strategy for building the triplet constraints leads to an improved performance compared with the subsampling strategy proposed in Schultz and Joachims (2004) (see the RMSE results of LDMLR and MLRC).

(ii) Our dual coordinate gradient descent is significantly faster than a general-purpose solver for finding the solution of a quadratic program (see the training time results of LDMLR and MLRC). Since the time complexity of LDMLR is $O(mn + m^2)$ in each outer iteration, we could expect LDMLR to perform well for more complex problems, where other methods, such as LMNNR and MLKR, could not achieve a good performance due to the high computational complexity in each iteration.

(iii) LDMLR exhibits a slightly better behavior compared to LMNNR and MLKR. This unexpected result is due to the fact that our method learns only a diagonal matrix, which has fewer parameters than other methods that learn a full Mahalanobis matrix.

(iv) Methods that learn a full Mahalanobis matrix require a memory complexity of $O(D^2)$, which poses a huge challenge when handling high-dimensional problems. In contrast, LDMLR only stores a diagonal matrix with a memory complexity of $O(D)$.

Table 9.3: Experimental results in terms of RMSE.

#	Euclidean	RLMNN	MLKR	MLRC	LDMLR
1.	0.051563	0.051220	**0.049009**	0.054875	0.056518
2.	0.028276	0.027087	**0.016588**	0.029336	0.028129
3.	0.196502	0.197256	**0.190883**	0.21574	0.202931
4.	0.150332	0.149030	**0.109057**	0.205514	0.139877
5.	0.360485	0.341728	**0.317247**	0.363681	0.340263
6.	0.272334	0.238165	**0.144129**	0.303363	0.237400
7.	0.483901	**0.481393**	0.521417	0.512054	0.503052
8.	0.446595	**0.440956**	0.447151	0.467792	0.449575
9.	3.688479	3.684056	3.508015	3.455432	**3.496472**
10.	1.813290	1.737894	1.191496	1.213079	**1.158852**
11.	4.152913	4.137456	3.573474	**3.481092**	3.394903
12.	2.994171	2.881655	1.294109	1.573102	**1.174007**
13.	0.024956	0.024781	0.025927	0.022591	**0.022444**
14.	0.012350	0.010897	**0.004860**	0.005406	0.005235
15.	0.036197	0.036081	0.030225	0.035769	**0.028683**
16.	0.029167	0.028801	**0.011677**	0.029050	0.017354
17.	39249.24	42230.38	39249.24	41832.60	**38582.66**
18.	8.067374	**6.251509**	7.521751	29.647121	7.419996
19.	0.200677	0.200868	0.201301	0.205264	**0.200547**
20.	0.250528	0.248829	0.261938	**0.248818**	0.250846
Rank	3.775000	2.950000	2.375000	3.70000	**2.200000**

9.5. Conclusion

In this chapter, we have proposed a new distance metric learning method to improve the performance of k-NNR. This was accomplished by formulating a convex optimization problem. To solve this problem, we have developed a special solver based on the coordinate descent method. We evaluated the method on a

Table 9.4: Experimental results in terms of training time (in seconds).

#	RLMNN	MLKR	MLRC	LDMLR
1.	9.81000	2.38556	217.518	**0.77911**
2.	9.89502	2.44267	213.220	**0.81447**
3.	9.70097	2.51613	213.497	**0.77781**
4.	8.09799	2.50395	213.150	**0.78570**
5.	24.8306	6.50735	454.399	**1.82979**
6.	21.9150	6.43434	460.688	**1.63494**
7.	20.3224	6.55719	325.564	**1.55986**
8.	18.2022	6.30261	299.474	**1.81043**
9.	8.42433	2.46434	207.913	**0.71277**
10.	8.72317	2.47085	200.182	**0.78939**
11.	10.2451	2.38182	211.207	**0.86834**
12.	8.62751	2.44820	209.708	**0.72931**
13.	16.8085	6.35133	266.689	**1.45746**
14.	15.2522	6.34046	195.784	**1.59970**
15.	15.7750	6.42428	270.898	**1.59220**
16.	15.1270	6.62855	271.122	**1.50747**
17.	1301.84	7832.16	2052.01	**70.4357**
18.	274.734	577.405	2041.11	**36.6199**
19.	133.610	3.43870	1919.36	**0.56642**
20.	183.197	1760.30	1828.70	**31.9248**

collection of twenty publicly available data sets. Experimental results show that our method outperforms the standard k-NNR using the Euclidean distance metric, while obtaining comparable results compared to the state-of-the-art methods MLKR and LMNNR. The results also show that our method is an order of magnitude faster than its competitors. The proposed method also enjoys a significant advantage in memory complexity, making it more practical for real-world applications, which are not tractable using the full Mahalanobis distance metric learning methods. Future work will include a deeper theoretical analysis, the inclusion of kernel functions and a deeper experimental analysis on high-dimensional data sets.

10 Distance metric learning for ordinal classification

Ordinal classification is a problem setting in-between nominal classification and metric regression, where the goal is to predict classes of an ordinal scale. Usually, there is a clear ordering of the classes, but the absolute distances between them are unknown. Disregarding the ordering information, this kind of problems is commonly treated as multi-class classification problems, although this is not appropriate from a semantic point of view. Exploring such ordering information can help to improve the effectiveness of classifiers. In this chapter, we propose a distance metric learning approach for ordinal classification by incorporating local triplet constraints containing the ordering information into a conventional large-margin distance metric learning approach. Specifically, our approach tries to preserve, for each training example, the ordinal relationship as well as the local geometry structure of its neighbors, which is suitable for use in local distance-based algorithms such as k-NN classification. Different from previous works that usually learn distance metrics by weighing the distances between training examples according to their class label differences, the proposed approach can directly satisfy the ordinal relationships where no assumptions about the distances between classes are made.

The material of this chapter is based on the following publication:
Nguyen, B., Morell, C., and De Baets, B. (2018a). Distance metric learning for ordinal classification based on triplet constraints. *Knowledge-Based Systems*, 142:17–28

10.1. Motivation

Ordinal classification (also called ordinal regression) has recently become an important research topic as a consequence of the growing amount of human preference information in many real-world applications, such as human age estimation (Chang et al., 2011), face recognition (Kim and Pavlovic, 2010), medical research (Pérez-Ortiz et al., 2014), social sciences (Fullerton and Xu, 2012), and so on. This learning task consists in predicting a target variable on an ordinal scale. Very often, for qualifying objects, humans prefer using ordinal labels instead of continuous scores. An ordinal variable can represent, for instance, the grades of students, which are usually represented in linguistic terms such as *bad*, *average*, *good*, and *excellent*. Clearly, *average* is more preferable than *bad*, and *good* is better than both. In contrast to nominal (binary or multi-class) classification, there exists a

linear ordinal relationship among the different class labels, which can be written as $bad \prec average \prec good \prec excellent$, where $\prec$ denotes the order relation.

At least two important challenges can be identified from the differences between nominal classification and ordinal classification (Gutiérrez and García, 2016). First, it is important to consider the ordering information among class labels. An ordinal classifier not only needs to recognize what class the data belong to, but it should also preserve the ordinal relationships of the data. Second, the measure used for assessing the performance of ordinal classifiers should take into account the deviations of the predictions from the true class labels as well as the consistency of the relative ordering among the class labels. In the literature, different types of solutions have been proposed to address this kind of problems. According to Gutiérrez et al. (2016), ordinal classification approaches can be divided into three categories: naïve approaches, threshold approaches, and ordinal binary decomposition approaches.

Naïve approaches treat ordinal classification problems as standard classification or regression problems by making some simplifying assumptions on the class labels. By ignoring the ordinal relationship among classes, unseen data can be classified using conventional nominal classifiers, such as support vector machines (Vapnik, 1998) and neural networks (Anthony and Bartlett, 2009). Although it is possible to directly use nominal classifiers, the classification accuracy may be limited due to the loss of training information. Another early effort in this category was proposed by Kramer et al. (2001). The idea is to transform the target variable into a real-valued and continuous variable, and then solve the ordinal classification problem using regression trees. The main drawback of this approach is that the true distances among the class labels are unknown in most cases, and as a consequence, it is difficult to estimate an appropriate function to map the class labels.

Threshold approaches try to overcome the aforementioned limitation by automatically estimating the distances between the class labels. They assume that the target variable is a one-dimensional latent continuous variable and learn thresholds that divide the real line into consecutive intervals. Each class label corresponds to an interval delimited by these thresholds. Different from naïve approaches, the distances between class labels are estimated during the learning process. Approaches belonging to this category are also popular in problems of learning to rank (Shashua and Levin, 2002). Within this context, Chu and Keerthi (2005) extended support vector machines to deal with ordinal classification. Two approaches were proposed to find parallel discriminant hyperplanes that separate examples of different classes. An interesting property of the second approach is that the constraints on thresholds are implicitly satisfied at the optimal solution.

Ordinal binary decomposition approaches, on the other hand, transform an ordinal classification problem into a set of binary (two-class) classification problems, which are separately solved by binary classifiers (Frank and Hall, 2001; Lin and Li,

2012). Hence, class labels are predicted by combining the binary outputs. Using the ranking information contained in the class labels, a simple definition of the set of binary classification problems can be made, for instance, by asking "if the class label of an example $\mathbf{x}$ is greater than q" (Frank and Hall, 2001). It is important to note that, apart from the way of defining two-class classification problems, the performance of approaches belonging to this category also depends on the way of combining the outputs.

Distance metric learning is now a well-established discipline in pattern recognition, but much of the attention has been focused on classification and clustering (Bellet et al., 2015). Due to the significant different characteristics of ordinal classification (Li et al., 2015), most of the previous works in distance metric learning cannot be directly applied for dealing with ordinal classification. One solution is to use additional constraints that indicate the ordinal relationships among examples of different classes, e.g., "examples A and B are closer in preference than A and C," which imposes ranking constraints $A \prec B \prec C$ or $C \prec B \prec A$. In the context of distance metric learning, these triplet constraints mean "A should be closer to B than to C." Therefore, by learning a distance metric that satisfies the ordinal relationships, we can address ordinal classification problems using distance-based classifiers. In particular, we aim to incorporate local triplet constraints into a conventional large-margin distance metric learning approach to improve the performance of k-nearest-neighbor (k-NN) classification (Cover and Hart, 1967). Although distance metric learning has been explored in some previous studies for ordinal classification (Li et al., 2012a, 2015; Fouad and Tino, 2013; Tian et al., 2016), to the best of our knowledge, there is no existing work that directly formulates this problem using triplet constraints. In short, our main contributions are summarized as follows:

(i) To bridge the gap between distance metric learning and ordinal classification, we propose a large-margin distance metric learning approach by adapting the ordering information on the training examples. Unlike previous works, the proposed approach makes no assumptions about the distances between the classes, thus leading to a better performance.

(ii) To further capture nonlinear structures in a complex data set, we extend the proposed approach to a kernelized version. We first map the input space into a Hilbert space by a nonlinear kernel function, then the distance metric is learned in the transformed space.

(iii) To validate the effectiveness of the proposed approach, we conduct extensive experiments using public benchmark data sets for ordinal classification. We show that the proposed approach improves the performance of k-NN classification.

The remainder of this chapter is organized as follows. Section 10.2 describes some basic concepts and background in distance metric learning. Subsequently, we

review some distance metric learning approaches closely related to ours, focusing specifically on the case of ordinal classification. Section 10.3 introduces the main idea of the proposed approach and its nonlinear kernelized version. Subsequently, an analysis of the computational complexity of the proposed approach is discussed. Section 10.4 discusses some issues and challenges in evaluating the performance of ordinal classifiers. Section 10.5 provides the experimental results, analyzing different aspects of the proposed approach. Finally, some concluding remarks and future work are presented in Section 10.6.

10.2. Preliminaries

The main goal of distance metric learning approaches is to estimate a distance metric that satisfies some application-specific requirements, but all follow the same guiding principle: similar examples should be close together and dissimilar examples should be far away from each other. In this section, we briefly describe the general framework of distance metric learning. An overview of challenges in distance metric learning is also provided.

10.2.1. Notations

We start by introducing some notations that will be used throughout this chapter. Let $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, n\}$ denote the training set, where example $\mathbf{x}_i$ belongs to the input space $\mathcal{X} \subseteq \mathbb{R}^D$ and its corresponding class label y_i belongs to the output space $\mathcal{Y} = \{C_1, \ldots, C_r\}$. An ordinal classification setting can be seen as a special case of nominal classification where there exists a linear order relation among the class labels, i.e., $C_1 \prec \cdots \prec C_r$.

10.2.2. Problem definition

Among the many different ways of learning a distance measure, we aim at finding the Mahalanobis distance metric (Weinberger and Saul, 2009), due to its simplicity and flexibility in incorporating the correlation between different features into the distance. The Mahalanobis distance metric can be seen as the Euclidean distance metric after performing a linear transformation on the input space. In other words, learning a Mahalanobis distance metric corresponds to learning a linear transformation (Nguyen et al., 2017c). Formally, the squared Mahalanobis distance between two examples $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as

$$d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j) = \langle \mathbf{M}, (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top \rangle \,,$$

where $\mathbf{M} \succcurlyeq 0$ is a symmetric positive semidefinite (PSD) matrix.

In the general distance metric learning framework, given a set of constraints $\mathcal{R}$, one can formulate the problem as a constrained optimization,

$$\underset{\mathbf{M}\succcurlyeq 0}{\text{minimize}} \quad \alpha\,\text{reg}(\mathbf{M}) + \ell(\mathbf{M}, \mathcal{R})\,, \tag{10.1}$$

with reg a regularization function to reduce the risk of overfitting, ℓ a loss function penalizing violations of constraints in $\mathcal{R}$, and $\alpha \geq 0$ a trade-off parameter. Essentially, our objective is to learn dissimilarities between examples by determining the importance of the input features and their correlations.

There have been several successful approaches for solving problem (10.1). The main challenge is to satisfy the positive semidefiniteness constraint on the matrix $\mathbf{M}$. Using a convex loss function and a convex regularizer, problem (10.1) then turns into a convex semidefinite program which can be solved in polynomial time by standard semidefinite programming methods (Boyd and Vandenberghe, 2004). The drawback of this technique is that it quickly becomes computationally intractable on large-scale data sets. To further reduce the computational complexity, Weinberger and Saul (2009) proposed an efficient solver based on the subgradient descent method. Davis et al. (2007) introduced an iterative Bregman projection method. More advanced optimization methods such as the Frank-Wolfe algorithm (Ying and Li, 2012), the block-coordinate descent (Qi et al., 2009; Atzmon et al., 2015), and the Boosting-like algorithm (Shen et al., 2012) are also studied in the context of distance metric learning.

In the literature, the constraints in $\mathcal{R}$ are usually represented in a pairwise (Xing et al., 2002) or triplet form (Schultz and Joachims, 2004). A pairwise constraint (i, j, y_{ij}) indicates whether two given examples, $\mathbf{x}_i$ and $\mathbf{x}_j$, belong to the same class ($y_{ij} = 1$) or not ($y_{ij} = -1$). A triplet constraint (i, j, l) indicates a relative comparison among three given examples such as "$\mathbf{x}_i$ is more similar to $\mathbf{x}_j$ than to $\mathbf{x}_l$." Triplet constraints can be seen as an extension of pairwise constraints since two pairwise constraints of the form $(i, j, 1)$ and $(i, l, -1)$ imply a triplet constraint (i, j, l), but not vice versa. That is, given a triplet constraint (i, j, l), it does not necessarily mean that $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same class, neither that $\mathbf{x}_i$ and $\mathbf{x}_l$ belong to different classes. Due to this flexibility, we will exploit triplet constraints to impose the ordinal relationships on learning a Mahalanobis distance metric. In the next section, we show how to implement this idea.

10.2.3. Related work

Although there is a large number of works in distance metric learning (Weinberger and Saul, 2009; Shen et al., 2012; Nguyen et al., 2016), most of them only concentrate on nominal classification. One of the most popular methods is the large margin nearest neighbor (LMNN) proposed by Weinberger and Saul (2009) for improving

the performance of k-NN classification. In essence, LMNN aims to maximize margins between target neighbors and all examples of different classes, leading to solve a convex semidefinite program. Compared with LMNN, our approach also follows the large-margin principle, but the formulation is quite different. While LMNN ignores any ordering information present in the data and solely focuses on increasing the classification performance of k-NN, our approach tries to preserve the ordinal relationships, entailing an even larger increase in performance. Our main contribution is the use of the matrix trace norm, which yields a simple explanation in the hypothesis space, thus reducing the risk of overfitting. Additionally, to avoid a large number of unnecessary constraints, we introduce a strategy to select only the local triplet constraints derived from neighborhoods of each training example.

Recently, several proposals have been formulated to reduce the gap between distance metric learning and ordinal classification. Compared to nominal distance metric learning approaches, ordinal approaches take the ordering information into account, thus resulting in more accurate predictions, especially when data sets are small (Gutiérrez et al., 2016). Below, we review some distance metric learning approaches particularly designed for dealing with ordinal classification.

Xiao et al. (2009) introduce a distance metric learning approach with the goal of preserving the local neighborhoods in the semantic space for human age estimation. To be more specific, the learned distance metric tries to keep the pairwise distances between examples of the same class in the local neighborhoods unchanged, while pushing examples of different classes far away. The limitation of such methodology is that the ordinal relationships with examples of different classes is neglected.

Fouad and Tino (2013) extend the information-theoretic metric learning (ITML) method (Davis et al., 2007) for ordinal classification by weighing pairwise constraints according to the differences between their class labels (OITML). However, this method may lead to suboptimal performance due to the assumption that the learned distance metric should be as close as possible to a predefined distance metric, e.g., the Euclidean distance metric.

Li et al. (2015) propose an ordinal distance metric learning method (LDMLR) for image ranking. The idea is to preserve the local structure for each training example with its target neighbors, while the ordinal relationships are preserved by weighing the distances between training examples according to the differences between their class labels. Nevertheless, this simple strategy may not be powerful enough for preserving the ordinal relationships.

Tian et al. (2016) attempt to preserve the ordinal relationships by minimizing distances between examples of the same class and simultaneously making all classes orderly distributed in the transformed space. The main limitation of this approach is that many hyper-parameters need to be tuned, which can make the selection

difficult in practice. Compared to this approach, our approach is much simpler and easier to implement.

Most of the existing approaches are based on the idea of weighing distances between examples to preserve the ordinal relationships among the class labels. Assuming that the class labels are represented by a set of consecutive integers, the weighing function depends on the differences between the class labels. This implies that all pairs of consecutive class labels in the ordinal scale have the same distance, which is not the case in many real-world ordinal classification problems (Gutiérrez et al., 2016). Therefore, this assumption is not always reasonable, and as a consequence, it may cause serious errors in the estimation of distance metrics. In contrast to these approaches, our approach tries to naturally preserve the ordinal relationships using triplet constraints, where no assumptions about the true distances between class labels are made.

10.3. Distance metric learning in ordinal settings

In this section, we present a novel distance metric learning approach based on the large-margin principle to improve the performance of k-NN in ordinal classification settings. We substitute the loss function in (10.1) by the widely applied hinge loss function resulting in a convex optimization framework. By imposing triplet constraints, the ordering of the data set is locally preserved, which makes it suitable for local methods such as k-NN classification. We also extend the proposed approach into a kernelized version to capture nonlinearities in the data. In the following, we will describe the proposed approach and its extension in more detail.

10.3.1. Linear distance metric learning

For the purpose of improving the performance of k-NN classification, our distance metric learning approach is based on the following considerations. First, each training example should share the same class label with its k nearest neighbors due to the majority voting rule of k-NN classification. Second, the ordinal relationships between a training example and its neighbors should be preserved in terms of distances, which makes the k-NN classifier more reliable in ordinal classification settings.

Similar to many existing distance metric learning approaches for nominal classification (Weinberger and Saul, 2009; Shi et al., 2014), the learned distance metric should preserve the local structure of each training example. That is, each training example is surrounded by k examples with the same class label, namely *target neighbors* (Weinberger and Saul, 2009). The target neighbors can be specified as k nearest neighbors of the same class using the Euclidean distance metric. For

the sake of simplicity, let η_{ij} be an indicator variable that takes value one when example $\mathbf{x}_j$ is a target neighbor of example $\mathbf{x}_i$, otherwise it takes value zero. Ideally, we would like to make distances from $\mathbf{x}_i$ to its target neighbors $\mathbf{x}_j$ smaller than to all examples $\mathbf{x}_l$ of different classes by at least a safe margin. However, this may cause a large number of constraints, which is computationally challenging. Instead, we approximate these full constraints by considering only local constraints in the neighborhood of each training example. Let $\mathcal{N}(\mathbf{x}_i)$ be the neighborhood of $\mathbf{x}_i$, which is a set of its nearest neighbors. The local neighborhood constraint can be defined as the following set of triplet constraints

$$\mathcal{R}_1 = \left\{ (i,j,l) \mid i,j,l \in \{1,\dots,n\}, \mathbf{x}_l \in \mathcal{N}(\mathbf{x}_i), \eta_{ij} = 1, \text{ and } y_i \neq y_l \right\}. \quad (10.2)$$

On the other hand, for each training example, the ordering information in its neighborhood should be preserved as well. Instead of considering only large margins between target neighbors and examples of different classes, we also keep a safe margin between different classes, thus making the ordinal relationships of classes explicit. That is, the larger the class dissimilarity of two examples is, the larger the distance between these two examples is. In other words, if $y_l \prec y_j \prec y_i$ or $y_i \prec y_j \prec y_l$, then examples of class y_l should be separated from those of class y_j by at least a safe margin. This consideration leads to the following sets of triplet constraints

$$\mathcal{R}_2 = \left\{ (i,j,l) \mid i,j,l \in \{1,\dots,n\}, \mathbf{x}_j, \mathbf{x}_l \in \mathcal{N}(\mathbf{x}_i) \text{ and } y_l \prec y_j \prec y_i \right\}, \quad (10.3)$$

$$\mathcal{R}_3 = \left\{ (i,j,l) \mid i,j,l \in \{1,\dots,n\}, \mathbf{x}_j, \mathbf{x}_l \in \mathcal{N}(\mathbf{x}_i) \text{ and } y_i \prec y_j \prec y_l \right\}. \quad (10.4)$$

It is important to remark that we cannot determine any triplet constraint when $y_l \prec y_i \prec y_j$ or $y_j \prec y_i \prec y_l$, since the true distances between class labels are unknown. In these cases, it is not clear whether the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ should be smaller than the distance between $\mathbf{x}_i$ and $\mathbf{x}_l$ or not.

Finally, combining all triplet constraints given in (10.2), (10.3), and (10.4) we obtain the following set of constraints

$$\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3 .$$

Inspired by the large-margin principle, we aim to estimate a Mahalanobis distance metric that satisfies, for any triplet constraint (i,j,l) in $\mathcal{R}$, the inequality

$$d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + 1 \leq d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l) . \quad (10.5)$$

The margin in (10.5) is set to 1 since its value only has an impact on the scale of $\mathbf{M}$ and not on the performance of k-NN classification. In order to allow for some inequalities to be violated, a soft margin is introduced to deal with the non-separable case. Following the general framework for distance metric learning

in (10.1), we define our problem as

$$\begin{aligned} \underset{\mathbf{M},\boldsymbol{\xi}}{\text{minimize}} \quad & f_L(\mathbf{M}) = \alpha \operatorname{tr}(\mathbf{M}) + \tfrac{1}{m} \sum_{(i,j,l)\in\mathcal{R}} \xi_{ijl} \\ \text{subject to} \quad & d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq 1 - \xi_{ijl}\,, \\ & \xi_{ijl} \geq 0 \;\text{ for }\; (i,j,l) \in \mathcal{R}\,, \\ & \mathbf{M} \succcurlyeq 0\,, \end{aligned} \tag{10.6}$$

where m denotes the number of constraints in $\mathcal{R}$ and ξ_{ijl} are slack variables to handle the case of soft margins for the sake of computational feasibility. In addition, the trace norm on $\mathbf{M}$ yields a sparse solution in eigenspectrum (Recht et al., 2010), thus our approach can simultaneously perform feature selection along with distance metric learning. This regularization function is also an implementation of Occam's razor principle according to which one should favor the simplest explanation consistent with the training examples (Vapnik, 1998).

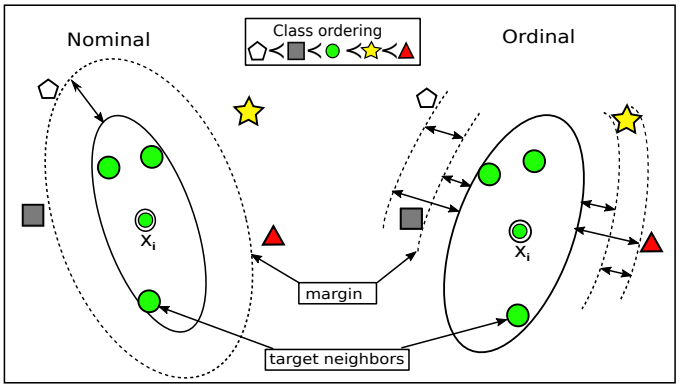


Figure 10.1: An illustration of distance metric learning for nominal (left) and ordinal (right) classification. Examples from different classes are represented as different shapes filled with different colors. The ellipse represents all examples having the same distance to example $\mathbf{x}_i$.

Table 10.1: Constraints derived from nominal and ordinal distance metric learning approaches with respect to example $\mathbf{x}_i$ in Fig. 10.1.

Nominal	Ordinal
d(◉,●)<d(◉,■)	d(◉,●)<d(◉,■)
d(◉,●)<d(◉,⬠)	d(◉,●)<d(◉,⬠) d(◉,■)<d(◉,⬠)
d(◉,●)<d(◉,☆)	d(◉,●)<d(◉,☆)
d(◉,●)<d(◉,▲)	d(◉,●)<d(◉,▲) d(◉,☆)<d(◉,▲)

For illustrative purposes, Fig. 10.1 shows the differences between the nominal (left-hand side) and ordinal (right-hand side) approaches. In both approaches, example $\mathbf{x}_i$ is surrounded by three target neighbors (shown as circles filled with green color). Although the nominal approach has a larger margin separating its target neighbors and other neighbors of different classes than that of our ordinal approach, the latter should be more favorable since the ordinal relationships are preserved. More concretely, example $\mathbf{x}_i$ is closer to the square example than to the pentagon example and $\mathbf{x}_i$ is closer to the star example than to the triangle example. Therefore, the probability of misclassifying $\mathbf{x}_i$ as pentagon is less than that of misclassifying it as square and the probability of misclassifying $\mathbf{x}_i$ as triangle is less than that of misclassifying it as star. It is important to note that the ordinal relationships are no longer guaranteed in the nominal approach. Additionally, Table 10.1 lists all constraints derived from the nominal distance metric learning approach and those from our ordinal approach. Clearly, our approach imposes more useful constraints than the nominal approach, thus making it more suitable for ordinal classification tasks.

To further understand the objective function in (10.6), we rewrite it into another form,

$$\begin{aligned} f_L(\mathbf{M}) &= \alpha \operatorname{tr}(\mathbf{M}) + \frac{1}{m} \sum_{(i,j,l)\in\mathcal{R}} \Big[1 + d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l)\Big]_+ \\ &= \alpha \operatorname{tr}(\mathbf{M}) + \frac{1}{m} \sum_{(i,j,l)\in\mathcal{R}} \Big[1 + \operatorname{tr}(\mathbf{M}\mathbf{X}_{ij}) - \operatorname{tr}(\mathbf{M}\mathbf{X}_{il})\Big]_+, \end{aligned}$$

where $\mathbf{X}_{ij} = (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top$ and $[z]_+ = \max(z, 0)$. Hence, the subgradient of f_L can be computed as

$$g_L(\mathbf{M}) = \frac{\partial}{\partial \mathbf{M}} f_L = \alpha \mathbf{I} + \frac{1}{m} \sum_{(i,j,l)\in\mathcal{R}} \beta_{ijl}(\mathbf{X}_{ij} - \mathbf{X}_{il})\,, \tag{10.7}$$

where

$$\beta_{ijl} = \begin{cases} 1 & , \text{ if } 1 + \operatorname{tr}(\mathbf{M}\mathbf{X}_{ij}) - \operatorname{tr}(\mathbf{M}\mathbf{X}_{il}) > 0\,; \\ 0 & , \text{ otherwise}\,. \end{cases}$$

To find the solution for problem (10.6), we employ the projected subgradient descent method due to its simplicity and effectiveness (Weinberger and Saul, 2009). Let $\mathbf{M}_t$ be the solution at the t-th iteration, then $\mathbf{M}_{t+1}$ can be computed as

$$\mathbf{M}_{t+1} = \mathbf{M}_t - \gamma g_L(\mathbf{M}_t)\,,$$

where γ denotes the step size. To guarantee that $\mathbf{M}_{t+1}$ is PSD, we project it onto the cone of PSD matrices using eigendecomposition. First, we perform the

eigendecomposition of $\mathbf{M}_{t+1}$ as

$$\mathbf{M}_{t+1} = \mathbf{V}_{t+1}\boldsymbol{\Delta}_{t+1}\mathbf{V}_{t+1}^{\top}\,.$$

Then, the projection of $\mathbf{M}_{t+1}$ onto the cone of PSD matrices can be computed as

$$\mathbf{M}_{t+1}^{*} = \mathbf{V}_{t+1}\max(\boldsymbol{\Delta}_{t+1}, 0)\mathbf{V}_{t+1}^{\top}\,.$$

A brief overview of our algorithm is given in Algorithm 10. We refer to the proposed algorithm as LODML.

Algorithm 10 Linear distance metric learning for ordinal classification

Input: $\{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, n\}$, k, γ;
Output: $\mathbf{M}_t \succcurlyeq 0$;

```
Set M_0 ← I and t ← 0;
Construct a set of local triplet constraints R;
while not converged do
    Compute the subgradient g_L(M_t) using (10.7);
    Set M_{t+1} ← M_t − γ g_L(M_t);
    Project M_{t+1} onto the PSD cone;
    Set t ← t + 1;
end while
```

10.3.2. Nonlinear distance metric learning

A simple Mahalanobis distance metric might not always be appropriate for a supervised learning problem (Schölkopf and Smola, 2001), especially in solving complex problems. In this subsection, we extend LODML into a kernelized version, which can overcome this limitation. Similarly as in (McFee et al., 2011), the kernelized algorithm can be performed as follows.

Let ϕ denote a function, possibly nonlinear, that maps each example from the input space $\mathcal{X} \subseteq \mathbb{R}^D$ into a feature space $\mathbb{F}$ with a high, possibly infinite, dimensionality. Instead of learning directly a distance metric in the input space, we are interested in learning a distance metric in the feature space mapped by ϕ.

Since $\mathbf{M} \succcurlyeq 0$, we can factorize the matrix $\mathbf{M}$ into $\mathbf{M} = \mathbf{A}^{\top}\mathbf{A}$. According to the representer theorem (Schölkopf et al., 2001), the optimal linear transformation $\mathbf{A}$, corresponding to the optimal Mahalanobis matrix $\mathbf{M}$, lies within the span of all training examples mapped by the function ϕ,

$$\mathbf{A} = \widehat{\mathbf{A}}\boldsymbol{\Phi}^{\top}\,,$$

where $\boldsymbol{\Phi} = (\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n))$ and $\widehat{\mathbf{A}}$ is the coefficient matrix. Thus, the Maha-

lanobis distance between two examples $\mathbf{x}_i$ and $\mathbf{x}_j$ in the feature space $\mathbb{F}$ can be computed as follows

$$\begin{aligned} d^2_{\mathbf{M}}(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)) &= (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^\top \mathbf{M} (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) \\ &= (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^\top \mathbf{\Phi} \widehat{\mathbf{A}}^\top \widehat{\mathbf{A}} \mathbf{\Phi}^\top (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) \\ &= \left[\mathbf{\Phi}^\top (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))\right]^\top \widehat{\mathbf{A}}^\top \widehat{\mathbf{A}} \left[\mathbf{\Phi}^\top (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))\right] \\ &= (\mathbf{K}_i - \mathbf{K}_j)^\top \widehat{\mathbf{M}} (\mathbf{K}_i - \mathbf{K}_j) \\ &= d^2_{\widehat{\mathbf{M}}}(\mathbf{K}_i, \mathbf{K}_j)\,, \end{aligned}$$

where $\mathbf{K_i} = \mathbf{\Phi}^\top \phi(\mathbf{x}_i)$ and $\widehat{\mathbf{M}} = \widehat{\mathbf{A}}^\top \widehat{\mathbf{A}}$. In a similar way, we can rewrite the regularization function in (10.6) as

$$\begin{aligned} \operatorname{tr}(\mathbf{M}) &= \operatorname{tr}(\mathbf{\Phi} \widehat{\mathbf{A}}^\top \widehat{\mathbf{A}} \mathbf{\Phi}^\top) \\ &= \operatorname{tr}(\widehat{\mathbf{A}}^\top \widehat{\mathbf{A}} \mathbf{\Phi}^\top \mathbf{\Phi}) \\ &= \operatorname{tr}(\widehat{\mathbf{M}} \mathbf{K})\,, \end{aligned}$$

where $\mathbf{K} = \mathbf{\Phi}^\top \mathbf{\Phi} = (\mathbf{K}_1, \ldots, \mathbf{K}_n)$ is the kernel matrix. Let $\widehat{\mathcal{R}}$ be the set containing triplet constraints in the feature space using the Euclidean distance metric by setting $\widehat{\mathbf{A}}$ to be the identity matrix $\mathbf{I}$, our distance metric learning problem (10.6) can be reformulated as

$$\begin{aligned} \underset{\widehat{\mathbf{M}}, \boldsymbol{\xi}}{\text{minimize}} \quad & f_K(\widehat{\mathbf{M}}) = \alpha \operatorname{tr}(\widehat{\mathbf{M}} \mathbf{K}) + \tfrac{1}{m} \sum_{(i,j,l) \in \widehat{\mathcal{R}}} \xi_{ijl} \\ \text{subject to} \quad & d^2_{\widehat{\mathbf{M}}}(\mathbf{K}_i, \mathbf{K}_l) - d^2_{\widehat{\mathbf{M}}}(\mathbf{K}_i, \mathbf{K}_j) \geq 1 - \xi_{ijl}\,, \\ & \xi_{ijl} \geq 0 \;\text{ for } (i,j,l) \in \widehat{\mathcal{R}}\,, \\ & \widehat{\mathbf{M}} \succcurlyeq 0\,. \end{aligned} \tag{10.8}$$

Note that problem (10.8) has a very similar form as problem (10.6) in the sense that if we slightly change the regularization function and substitute each training example in (10.6) by a column vector of the kernel matrix, then we obtain problem (10.8). Most interestingly, the kernel matrix $\mathbf{K}$ can be evaluated using the kernel trick (Schölkopf and Smola, 2001), which consists in replacing the inner product by an arbitrary kernel function. Thereby, we can avoid explicitly expressing the mapping ϕ, which is usually unknown and difficult to estimate due to the high dimensionality.

Similar to LODML, we perform the projected subgradient descent method to solve problem (10.8). First, we remove the slack variables and rewrite the objective

function in (10.8) as follows

$$f_K(\widehat{\mathbf{M}}) = \alpha \operatorname{tr}(\widehat{\mathbf{M}}\mathbf{K}) + \frac{1}{m} \sum_{(i,j,l)\in\widehat{\mathcal{R}}} \Big[1 + d^2_{\widehat{\mathbf{M}}}(\mathbf{K}_i, \mathbf{K}_j) - d^2_{\widehat{\mathbf{M}}}(\mathbf{K}_i, \mathbf{K}_l)\Big]_+$$
$$= \alpha \operatorname{tr}(\widehat{\mathbf{M}}\mathbf{K}) + \frac{1}{m} \sum_{(i,j,l)\in\widehat{\mathcal{R}}} \Big[1 + \operatorname{tr}(\widehat{\mathbf{M}}\widehat{\mathbf{X}}_{ij}) - \operatorname{tr}(\widehat{\mathbf{M}}\widehat{\mathbf{X}}_{il})\Big]_+ ,$$

where $\widehat{\mathbf{X}}_{ij} = (\mathbf{K}_i - \mathbf{K}_j)(\mathbf{K}_i - \mathbf{K}_j)^\top$. Hence, the subgradient of f_K can be computed as

$$g_K(\widehat{\mathbf{M}}) = \frac{\partial}{\partial \widehat{\mathbf{M}}} f_K = \alpha\mathbf{K} + \frac{1}{m} \sum_{(i,j,l)\in\widehat{\mathcal{R}}} \widehat{\beta}_{ijl}(\widehat{\mathbf{X}}_{ij} - \widehat{\mathbf{X}}_{il}) , \tag{10.9}$$

where

$$\widehat{\beta}_{ijl} = \begin{cases} 1 & , \text{ if } 1 + \operatorname{tr}(\widehat{\mathbf{M}}\widehat{\mathbf{X}}_{ij}) - \operatorname{tr}(\widehat{\mathbf{M}}\widehat{\mathbf{X}}_{il}) > 0 ; \\ 0 & , \text{ otherwise} . \end{cases}$$

Algorithm 11 Nonlinear distance metric learning for ordinal classification

Input: $\{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, n\}$, k, γ, $\mathbf{K}$;
Output: $\widehat{\mathbf{M}}_t \succcurlyeq 0$;

1: Set $\widehat{\mathbf{M}}_0 \leftarrow \mathbf{I}$ and $t = 0$;
2: Construct a set of local triplet constraints $\widehat{\mathcal{R}}$;
3: **while** not converged **do**
4: Compute the subgradient $g_K(\widehat{\mathbf{M}}_t)$ using (10.9);
5: Set $\widehat{\mathbf{M}}_{t+1} \leftarrow \widehat{\mathbf{M}}_t - \gamma\, g_K(\widehat{\mathbf{M}}_t)$;
6: Project $\widehat{\mathbf{M}}_{t+1}$ onto the PSD cone;
7: Set $t \leftarrow t + 1$;
8: **end while**

Let $\widehat{\mathbf{M}}_t$ be the solution at the t-th iteration, then the solution $\widehat{\mathbf{M}}_{t+1}$ can be computed as

$$\widehat{\mathbf{M}}_{t+1} = \widehat{\mathbf{M}}_t - \gamma g_K(\widehat{\mathbf{M}}_t) .$$

Subsequently, we need to project $\widehat{\mathbf{M}}_{t+1}$ onto the cone of PSD matrices. First, we perform the eigendecomposition of $\widehat{\mathbf{M}}_{t+1}$ as

$$\widehat{\mathbf{M}}_{t+1} = \widehat{\mathbf{V}}_{t+1}\widehat{\boldsymbol{\Delta}}_{t+1}\widehat{\mathbf{V}}_{t+1}^\top .$$

Then, the projection of matrix $\mathbf{M}_{t+1}$ onto the PSD cone can be computed as

$$\widehat{\mathbf{M}}^*_{t+1} = \widehat{\mathbf{V}}_{t+1} \max(\widehat{\boldsymbol{\Delta}}_{t+1}, 0)\widehat{\mathbf{V}}_{t+1}^\top .$$

Algorithm 11 gives a brief description of our algorithm. We refer to the proposed algorithm as KODML.

10.3.3. Computational complexity

We now analyze the computational complexity of LODML presented in Algorithm 10. The search for target neighbors can be performed in $O(kn^2 + Dn^2)$ using linear nearest neighbor search. Similarly, we can find the neighborhood for each training example in $O(vn^2 + Dn^2)$ where v denotes the size of neighborhood. The time complexity of computing the subgradient g_L as in (10.7) is $O(mD^2)$. The projection of $\mathbf{M}$ onto the cone of PSD matrices scales as $O(D^3)$. The overall time complexity of LODML per iteration is $O(D^3 + mD^2)$.

Note that the computation of subgradient g_L requires to perform the outer products $\mathbf{X}_{ij}$, which scale quadratically with the dimensionality $O(mD^2)$. When the number of triplet constraints is large, the computation of g_L using brute-force methods might be limited due to the high computational cost. Let $\mathcal{A}_t$ denote the set of active constraints, containing all triplet constraints (i, j, l) that satisfy the following inequality

$$d^2_{\mathbf{M}_t}(\mathbf{x}_i, \mathbf{x}_j) + 1 > d^2_{\mathbf{M}_t}(\mathbf{x}_i, \mathbf{x}_l)\,.$$

Following Weinberger and Saul (2009), we observe that only the differences between two sets of active constraints are required to compute g_L which enables us to reduce the computational burden of g_L as follows

$$g_L(\mathbf{M}_{t+1}) = g_L(\mathbf{M}_t) - \frac{1}{m}\mathbf{U}_t\,,$$

where

$$\mathbf{U}_t = \sum_{(i,j,l)\in\mathcal{A}_t\setminus\mathcal{A}_{t+1}} (\mathbf{X}_{ij} - \mathbf{X}_{il}) - \sum_{(i,j,l)\in\mathcal{A}_{t+1}\setminus\mathcal{A}_t} (\mathbf{X}_{ij} - \mathbf{X}_{il})\,.$$

That is, the update of $g_L(\mathbf{M}_{t+1})$ is computed by adding contributions of triplet constraints that become active and subtracting the contributions of triplets that are no longer active.

The computational complexity of KODML presented in Algorithm 11 can be analyzed as follows. We first compute the kernel matrix $\mathbf{K}$ in $O(Dn^2)$. The target neighbors and neighborhood of each training example are performed in the feature space using the Euclidean distance metric

$$\begin{aligned} d(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)) &= \sqrt{(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^\top (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))} \\ &= \sqrt{K_{ii} - 2K_{ij} + K_{jj}}\,, \end{aligned}$$

which scales as $O(1)$. Thus, the time complexity of searching target neighbors and neighborhoods is $O(kn^2 + vn^2)$. The regularization function $\mathrm{tr}(\widehat{\mathbf{M}}\mathbf{K})$ in (10.8) can be computed in $O(n^2)$. The computation of subgradient g_K in (10.9) scales as $O(mn^2)$. The projection of $\widehat{\mathbf{M}}$ onto the cone of PSD matrices scales as $O(n^3)$. Summarizing, the time complexity of KODML per iteration is $O(mn^2+n^3)$. Let $\widehat{\mathcal{A}}_t$ denote the set of active constraints at the t-th iteration. Similar to the computation of g_L, to further reduce the time complexity of computing g_K, we can perform

$$g_K(\widehat{\mathbf{M}}_{t+1}) = g_K(\widehat{\mathbf{M}}_t) - \frac{1}{m}\widehat{\mathbf{U}}_t\,,$$

where

$$\widehat{\mathbf{U}}_t = \sum_{(i,j,l)\in\widehat{\mathcal{A}}_t\setminus\widehat{\mathcal{A}}_{t+1}} (\widehat{\mathbf{X}}_{ij} - \widehat{\mathbf{X}}_{il}) - \sum_{(i,j,l)\in\widehat{\mathcal{A}}_{t+1}\setminus\widehat{\mathcal{A}}_t} (\widehat{\mathbf{X}}_{ij} - \widehat{\mathbf{X}}_{il})\,.$$

The update of g_K can be computed using only the differences between two sets of active constraints.

10.4. Performance evaluation

In this section, we briefly discuss some performance measures for assessing the performance of ordinal classifiers, along with their shortcomings. As the main goal of classification is to produce a hypothesis $h\colon \mathcal{X} \to \mathcal{Y}$, very often, the misclassification rate or Mean Zero-One Error (MZE) is considered (Chu and Keerthi, 2005; Chu and Ghahramani, 2005). Let n_t denote the number of test examples, then the MZE is computed as

$$\mathrm{MZE} = \frac{1}{n_t}\sum_{j=1}^{n_t} [\![y_j \neq h(\mathbf{x}_j)]\!]\,,$$

where $[\![.]\!]$ is the indicator function that takes value one if its argument is true, and zero otherwise. This measure, however, treats every misclassification equally, which is not robust to evaluate the performance of ordinal classifiers (Gutiérrez and García, 2016). For instance, in the system of predicting student grades, labeling a *good* student as *bad* is not the same as labeling it as *average*. To overcome this limitation of MZE, we can use alternative measures that also take into account the magnitude of the prediction errors. One of the most commonly used measures is the Mean Absolute Error (MAE) (Baccianella et al., 2009). Assuming that class labels are represented by numbers, the MAE is computed as

$$\mathrm{MAE} = \frac{1}{n_t}\sum_{j=1}^{n_t} |y_j - h(\mathbf{x}_j)|\,.$$

To be more specific, MAE is defined as the average deviation of the predicted class labels from the true ones. Since the true distances between class labels are usually unknown, the performance given by MAE is strongly influenced by the numerical representation of the class labels.

To avoid this kind of influence, one solution is to consider only the ordinal relationships between the predicted and the true class labels (Gutiérrez and García, 2016). Along this line, the C-index or the concordance index (Gönen and Heller, 2005) is reported as one of the most popular measures. Formally, the C-index is defined as the ratio of the number of concordant pairs to the number of comparable pairs, i.e.

$$\text{C-index} = \frac{1}{\sum_{k<l} n_k n_l} \sum_{y_{j_1} < y_{j_2}} \left([\![h(\mathbf{x}_{j_1}) < h(\mathbf{x}_{j_2})]\!] + \frac{1}{2} [\![h(\mathbf{x}_{j_1}) = h(\mathbf{x}_{j_2})]\!] \right)$$

with n_k and n_l the numbers of test examples having class label k and l, respectively. In the binary case, the C-index corresponds to the Wilcoxon–Mann–Whitney statistic or, equivalently, the area under the receiver operating characteristics (ROC) curve (AUC) (Cortes and Mohri, 2004).

10.5. Experiments

In order to show the effectiveness of using triplet constraints in ordinal settings, we compare the performance of the proposed methods, **LODML** and **KODML**, with other state-of-the-art distance metric learning methods, including the baseline **Euclidean** distance metric, **ITML** (Davis et al., 2007), **LMNN** (Weinberger and Saul, 2009), **DML-eig** (Ying and Li, 2012), **LDMLR** and its kernelized version **KDMLR** (Li et al., 2015). The latter two methods have been especially designed for ordinal classification tasks. All methods are implemented in Matlab[1]. The source codes of LODML and KODML are available online at http://users.ugent.be/~bacnguye/ODMLv1.0.zip. All results are reported in the context of 3-NN classification. To obtain the best results for all methods, the hyper-parameters are tuned via cross-validation based on the MZE, MAE, or C-index, depending on the measure considered. For LMNN, we set the maximum number of iterations to 1,000 and tune the trade-off parameter μ considering as set of values $\{0.125, 0.25, 0.5\}$ as in (Weinberger and Saul, 2009). For ITML, we set the maximum number of iterations to 100,000 and tune the slack parameter γ considering as set of values $\{10^{-3}, \ldots, 10^{3}\}$. Following Li et al. (2015), we set $p = 0.5$ and $\beta = 0.1$ for LDMLR and KDMLR, and tune the hyper-parameter α considering as set of

[1] The source codes are available from the corresponding authors' websites (except LDMLR):
ITML: http://www.cs.utexas.edu/~pjain/itml/download/itml-1.2.tar.gz
LMNN: http://www.cse.wustl.edu/~kilian/code/
DML-eig: http://empslocal.ex.ac.uk/people/staff/yy267/software.html

values $\{10^{-3}, \ldots, 10^{3}\}$. For LODML and KODML, we use 15 nearest neighbors to build the neighborhood and tune the hyper-parameter α considering as set of values $\{10^{-5}, \ldots, 10^{2}\}$. The maximum number of iterations for LDMLR, KDMLR, LODML, and KODML is set to 1,000. The following RBF Gaussian kernel is adopted for the kernelized methods, including KDMLR and KODML, $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma)$. We tune the kernel width σ considering as set of values $\{10^{-3}, \ldots, 10^{3}\}$.

10.5.1. Benchmark data sets

For comparison purposes, we carry out experiments on fifteen real-world ordinal classification data sets, which were also used in (Gutiérrez et al., 2016; Pérez-Ortiz et al., 2016). All these benchmark data sets are extracted from public repositories, including the UCI machine learning repository (Frank and Asuncion, 2010) and the `mldata.org` repository (PASCAL, 2011). We should remark that ordinal data sets are usually created by gathering information from human experts, therefore, they are often of small to moderate size only (Agresti, 2010). Additionally, we use some large data sets provided by Chu and Ghahramani (2005). These data sets were generated from regression problems by discretizing target values into ordinal classes using 10 equal-length bins. One major limitation is that they assume the ordinal classes to be equidistant, which is not always the case in ordinal settings. The characteristics of the data sets are summarized in Table 10.2. In each row, we specify the number of features, classes, and examples. To avoid the influence carried by the scale of features, all features are normalized (to have zero mean and unit standard deviation) over the training data. A 5-fold cross-validation scheme is employed to estimate the performance of the competing methods. All partitions are performed using stratified sampling in order to maintain the original distribution of classes. The results will be obtained by averaging over five runs. According to Cruz-Ramírez et al. (2014), using a single performance measure for ordinal classification might lead to partial or inexact conclusions, therefore, we will report the results using several measures, including the MZE, MAE, and C-index, to compare the performances of the competing methods.

Tables 10.3 to 10.5 show the experimental results based on the selected measures for the linear distance metric learning methods (i.e., Euclidean, ITML, LMNN, DML-eig, LDMLR, and LODML). The average ranks of the competing methods are listed in the penultimate row of these tables. For each data set, we rank the methods based on their performance, i.e., rank 1 is assigned to the best method, rank 2 is assigned to the second best, and so on. We define the average rank of one method as the mean rank over the 23 data sets considered, providing a fair comparison between the competing methods (Demšar, 2006). From the results, it is clear that using distance metric learning methods, the performance of the k-NN classifier is improved. Interestingly, LDMLR yields only competitive results when

Table 10.2: Description of the benchmark data sets used in our experiments.

Id	Data set	Features	Classes	Examples
Real ordinal classification data sets				
ER	ERA	4	9	1,000
ES	ESL	4	9	488
LE	LEV	4	5	1,000
SW	SWD	10	4	1,000
AU	automobile	71	6	205
BA	balance-scale	4	3	625
BO	bondrate	37	5	57
CA	car	21	4	1,728
EU	eucalyptus	91	5	736
NE	newthyroid	5	3	215
PA	pasture	25	3	36
SD	squash-stored	51	3	52
SE	squash-unstored	51	3	52
TA	tae	54	3	151
WI	winequality-red	11	6	1,599
Discretized regression data sets				
AB	abalone	11	10	4,177
B1	bank1	8	10	8,192
B2	bank2	32	10	8,192
CH	calhousing	8	10	20,640
S1	census1	8	10	22,784
S2	census2	16	10	22,784
C1	computer1	12	10	8,192
C2	computer2	21	10	8,192

compared to the state-of-the-art LMNN, which is successfully used for nominal classification. This behavior may be due to the fact that LDMLR tries to preserve the ordinal relationships globally for each training example, which is very difficult to achieve. In general, LODML consistently outperforms ITML, LMNN, DML-eig, and LDMLR.

10.5.2. Statistical analysis of the results

In order to detect whether there are significant differences in performance among the results reported in Tables 10.3 to 10.5, we follow the recommendations made by Demšar (2006). We first perform the Friedman test (Friedman, 1940) at a confidence level of $\alpha = 0.05$ with the null hypothesis that all the competing methods obtain the same results on average. For each performance measure (i.e., MZE, MAE, and C-index), the p-value of the Friedman test with 5 degrees of freedom is shown in the last row of Tables 10.3 to 10.5. Since the p-values are less than the confidence level α, we reject the null hypothesis. This means that there exist statistically significant differences between at least two methods.

Table 10.3: MZE of the linear distance metric learning methods on the benchmark data sets. Best results are highlighted in boldface.

Id	Euclidean	ITML	LMNN	DML-eig	LDMLR	LODML
ER	**0.8280**	**0.8280**	**0.8280**	**0.8280**	**0.8280**	**0.8280**
ES	0.3298	0.3156	0.3196	0.3995	0.3463	**0.3135**
LE	0.4900	0.4900	0.4890	**0.4890**	0.4900	0.4900
SW	0.5300	0.5280	0.5330	0.5270	**0.5260**	0.5290
AU	0.4195	0.3366	**0.2829**	0.3268	0.3707	0.3073
BA	0.1744	0.0848	0.1472	0.1056	0.1824	**0.0496**
BO	0.4697	0.5076	0.4530	0.4727	0.4439	**0.4015**
CA	0.2627	0.0324	0.0405	0.0909	0.2282	**0.0284**
EU	0.5299	0.4199	0.4620	0.4960	0.4483	**0.4185**
NE	0.0465	**0.0279**	0.0326	0.0372	**0.0279**	**0.0279**
PA	0.3607	0.3393	**0.3357**	0.3857	0.3514	**0.3357**
SD	0.3818	0.4000	0.4364	0.4473	0.4964	**0.3655**
SE	0.3455	0.3473	0.4218	0.4273	0.4255	**0.2873**
TA	0.5295	0.4574	0.4699	0.5228	0.4501	**0.4163**
WI	0.4497	0.4534	0.4390	0.4534	0.4497	**0.4321**
AB	0.7826	0.7763	0.7754	0.7766	0.7745	**0.7735**
B1	0.7062	0.5697	0.6078	0.5540	0.6062	**0.5055**
B2	0.8629	0.8252	0.8431	0.8003	0.8529	**0.8002**
CH	0.6598	0.6592	0.6573	0.7092	**0.6293**	0.6302
S1	**0.7508**	0.7569	0.7563	0.7688	**0.7508**	0.7519
S2	0.7398	0.7409	**0.7373**	0.7882	0.7380	**0.7373**
C1	0.5836	0.5833	0.5815	0.5734	0.5730	**0.5690**
C2	0.5584	**0.5341**	0.5481	0.5400	0.5584	0.5354
Rank	4.7174	3.4130	3.3478	4.2826	3.5435	**1.6957**
p-value	1.2678×10^{-6}					

Subsequently, we perform the Wilcoxon signed-rank test (Wilcoxon, 1945) and several post-hoc tests, including Bonferroni-Dunn (Dunn, 1961), Holm (Holm, 1979), Hochberg (Hochberg, 1988), Hommel (Hommel, 1988), to detect whether a competing method performs equivalently or significantly different from the control method (i.e., LODML, which obtained the lowest rank). Using the post-hoc tests, the p-values are adjusted in order to compensate for multiple comparisons (Demšar, 2006). If the adjusted p-value for a particular null hypothesis is less than a confidence level of $\alpha = 0.05$, then that hypothesis is rejected. Tables 10.6 to 10.8 show the unadjusted p-value (**pUnadj**) computed by the Wilcoxon signed-rank test, the adjusted p-values computed by the Bonferroni-Dunn (**pBonf**), Holm (**pHolm**), Hochberg (**pHoch**), and Hommel (**pHomm**) tests. The test results

Table 10.4: MAE of the linear distance metric learning methods on the benchmark data sets. Best results are highlighted in boldface.

Id	Euclidean	ITML	LMNN	DML-eig	LDMLR	LODML
ER	**1.8360**	**1.8360**	**1.8360**	**1.8360**	**1.8360**	**1.8360**
ES	0.3585	**0.3381**	0.3524	0.4240	0.3750	**0.3381**
LE	0.6220	0.6220	**0.6210**	**0.6210**	0.6220	0.6220
SW	0.6130	0.6110	0.6160	0.6120	**0.6090**	0.6160
AU	0.6585	0.4439	**0.3951**	0.5171	0.4780	**0.3951**
BA	0.2128	0.0928	0.1888	0.1328	0.2144	**0.0640**
BO	0.6091	0.5773	0.5394	0.5955	0.5700	**0.4545**
CA	0.3559	0.0353	0.0480	0.1042	0.0634	**0.0336**
EU	0.7772	**0.4784**	0.5571	0.6292	0.5530	0.5041
NE	0.0465	**0.0279**	0.0326	0.0372	**0.0279**	**0.0279**
PA	0.4179	0.3964	**0.3357**	0.4107	0.3500	**0.3357**
SD	0.4000	0.4182	0.4545	0.5055	0.5145	**0.3836**
SE	0.3636	0.3655	0.4600	0.4473	0.4636	**0.3055**
TA	0.7075	0.5957	0.6015	0.6617	0.5428	**0.5215**
WI	0.5297	0.5185	0.5185	0.5347	0.5297	**0.5160**
AB	1.8952	1.8559	1.9054	1.9241	1.9104	**1.8793**
B1	1.2919	0.7601	0.8709	0.7313	1.0919	**0.5941**
B2	2.8633	2.3713	2.6127	2.0912	2.0863	**1.9965**
CH	1.3044	1.1911	1.2873	1.6313	**1.1900**	**1.1900**
S1	**1.8025**	1.8343	1.8319	1.9548	**1.8025**	1.8085
S2	1.6074	1.6091	**1.5943**	2.0235	1.6070	1.5973
C1	0.9441	0.9349	0.9424	0.9366	0.9341	**0.8953**
C2	0.8370	0.8048	0.8163	0.8273	0.8370	**0.7789**
Rank	4.7174	2.9130	3.4348	4.5217	3.6087	**1.8043**
p-value			4.0196×10^{-7}			

confirm that our method significantly outperforms the other competing methods based on all selected performance measures, except in one case, namely for the C-index and ITML.

10.5.3. Influence of using ordering information

Additionally, we compare the performance of the nominal distance metric learning method (i.e., LMNN) and that of our ordinal distance metric learning method (i.e., LODML) when the training size increases. Figure 10.2 illustrates the MZEs and MAEs of LMNN and LODML with a varying number of training examples on the *balance-scale* data set. All results are reported using the same test sets. When

Table 10.5: C-index of the linear distance metric learning methods on the benchmark data sets. Best results are highlighted in boldface.

Id	Euclidean	ITML	LMNN	DML-eig	LDMLR	LODML
ER	**0.6356**	**0.6356**	**0.6356**	**0.6356**	**0.6356**	**0.6356**
ES	0.9183	0.9199	0.9164	0.9066	0.9185	**0.9224**
LE	**0.7416**	**0.7416**	0.7413	0.7413	**0.7416**	**0.7416**
SW	0.7021	0.7022	0.7006	0.7007	**0.7028**	0.6978
AU	0.7591	0.8088	0.8533	0.8018	0.8306	**0.8569**
BA	0.8922	0.9521	0.8969	0.9191	0.8924	**0.9689**
BO	0.5446	**0.7038**	0.6401	0.5725	0.6445	0.6734
CA	0.6511	0.9790	0.9676	0.9379	0.7306	**0.9840**
EU	0.7706	0.8670	0.8447	0.8278	0.8426	**0.8706**
NE	0.9470	0.9623	0.9620	0.9606	**0.9694**	0.9567
PA	0.7729	0.7732	**0.8512**	0.8006	0.7889	0.8324
SD	0.7249	0.7166	0.6954	0.6710	0.7007	**0.7344**
SE	0.7050	0.6993	0.6026	0.6254	0.6189	**0.7818**
TA	0.6206	0.6628	0.6783	0.6265	**0.6960**	0.6907
WI	0.6891	0.7020	**0.7026**	0.6934	0.6891	0.6991
AB	0.7580	**0.7620**	0.7561	0.7538	0.7245	0.7572
B1	0.8618	0.9204	0.9076	0.9183	0.8618	**0.9373**
B2	0.6135	0.6901	0.6561	0.7265	0.6135	**0.7393**
CH	0.8333	0.8423	0.8355	0.7864	0.8433	**0.8492**
S1	**0.7633**	0.7579	0.7589	0.7400	**0.7633**	0.7627
S2	0.7898	0.7878	**0.7917**	0.7237	0.7898	0.7911
C1	0.8842	0.8849	0.8840	0.8857	0.8852	**0.8905**
C2	0.8988	0.9015	0.9012	0.8991	0.8988	**0.9061**
Rank	4.5217	2.8261	3.6522	4.3913	3.6087	**2.0000**
p-value	1.4218×10^{-5}					

Table 10.6: Unadjusted p-value and adjusted p-values according to the Wilcoxon test and different post-hoc tests over 23 data sets based on MZE using LODML as the control method.

Method	pUnadj	pBonf	pHolm	pHoch	pHomm	Hypothesis
Euclidean	4.317E-8	2.158E-7	2.158E-7	2.158E-7	2.158E-7	Rejected
DML-eig	2.741E-6	1.370E-5	1.096E-5	1.096E-5	1.096E-5	Rejected
LDMLR	8.096E-4	0.0040	0.0024	0.0024	0.0024	Rejected
ITML	0.0018	0.0092	0.0037	0.0027	0.0027	Rejected
LMNN	0.0027	0.0137	0.0037	0.0027	0.0027	Rejected

Table 10.7: Unadjusted p-value and adjusted p-values according to the Wilcoxon test and different post-hoc tests over 23 data sets based on MAE using LODML as the control method.

Method	pUnadj	pBonf	pHolm	pHoch	pHomm	Hypothesis
Euclidean	1.289E-7	6.447E-7	6.447E-7	6.447E-7	6.447E-7	Rejected
DML-eig	8.406E-7	4.203E-6	3.362E-6	3.362E-6	3.362E-6	Rejected
LDMLR	0.0011	0.0054	0.0032	0.0032	0.0032	Rejected
LMNN	0.0031	0.0156	0.0062	0.0062	0.0062	Rejected
ITML	0.0445	0.2223	0.0445	0.0445	0.0445	Rejected

Table 10.8: Unadjusted p-value and adjusted p-values according to the Wilcoxon test and different post-hoc tests over 23 data sets based on C-index using LODML as the control method.

Method	pUnadj	pBonf	pHolm	pHoch	pHomm	Hypothesis
Euclidean	4.853E-6	2.426E-5	2.426E-5	2.426E-5	2.426E-5	Rejected
DML-eig	1.460E-5	7.301E-5	5.840E-5	5.840E-5	5.840E-5	Rejected
LMNN	0.0027	0.0137	0.0082	0.0071	0.0055	Rejected
LDMLR	0.0035	0.0177	0.0082	0.0071	0.0071	Rejected
ITML	0.1343	0.6714	0.1343	0.1343	0.1343	Accepted

the training size is large, there is a significant difference between the performances of LMNN and LODML. This is due to the fact that our method (LODML) takes advantage of the ordering information, whereas LMNN simply ignores this meaningful information. Beyond a training set of 200 examples, the difference becomes more apparent. These results support the claim that ordinal classifiers taking the ordering of the classes into account can result in more accurate predictions.

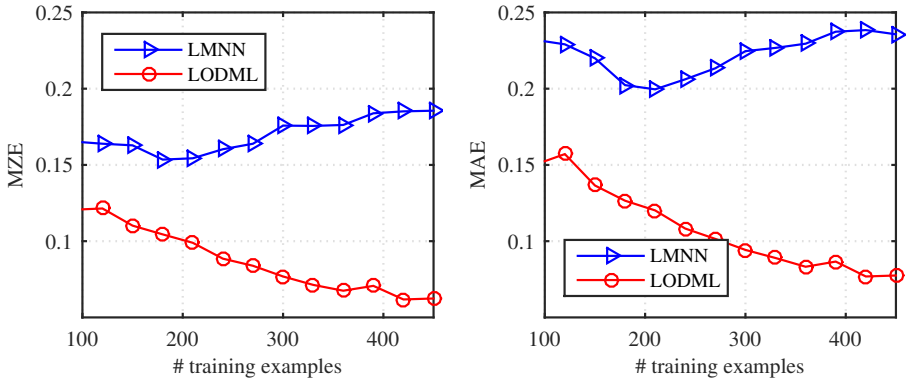


Figure 10.2: MZE and MAE vs. number of training examples on the *balance-scale* data set for LMNN and LODML.

10.5.4. Nonlinear distance metric learning

We further compare the performance of our kernelized method KODML with that of KDMLR on these benchmark data sets. Note that both kernelized methods learn a Mahalanobis matrix that scales quadratically with the number of training examples, making the experiments on large data sets computationally expensive. Due to this reason, we restrict our experiments to small data sets (i.e., $n \leq 1000$). From Table 10.9, we can observe that our kernelized method KODML outperforms KDMLR in most cases. Interestingly, KODML consistently obtain better results than KDMLR based on the C-index. Compared to the linear method LODML, the nonlinear method KODML outperforms KDMLR for nine out of the thirteen data sets considering the MZE.

Table 10.9: MZE, MAE, and C-index of the nonlinear distance metric learning methods on the small benchmark data sets. Best results are highlighted in boldface.

Id	MZE		MAE		C-index	
	KDMLR	KODML	KDMLR	KODML	KDMLR	KODML
ER	0.8260	**0.8250**	1.8280	**1.8100**	0.6361	**0.6376**
ES	**0.3646**	0.3686	**0.3912**	0.3953	0.9139	**0.9160**
LE	0.4900	**0.4810**	0.6220	**0.6050**	0.7416	**0.7508**
SW	0.5270	**0.5160**	0.6120	**0.5950**	0.6991	**0.7076**
AU	0.4000	**0.3512**	0.5512	**0.4732**	0.8109	**0.8345**
BA	0.1280	**0.0480**	0.1664	**0.0640**	0.9139	**0.9692**
BO	**0.3848**	0.3985	**0.4712**	0.5030	0.6204	**0.6251**
EU	0.5501	**0.4810**	0.8511	**0.6183**	0.7506	**0.8246**
NE	0.0419	**0.0233**	0.0419	**0.0233**	0.9486	**0.9762**
PA	0.3107	**0.2821**	0.3393	**0.2821**	0.8295	**0.8795**
SD	0.4036	**0.3573**	0.4418	**0.3855**	0.6893	**0.7624**
SE	0.3073	**0.2673**	0.3073	**0.2673**	0.7383	**0.7813**
TA	**0.4837**	0.5301	**0.6355**	0.7019	**0.6510**	0.5892

10.5.5. Convergence analysis

According to Boyd and Vandenberghe (2004), subgradient descent methods converge to the optimal solution provided that the step size is small enough. In this section, we empirically test the convergence of the proposed methods, LODML and KODML, on the *balance-scale* data set. As shown in Fig. 10.3, the objective function value always decreases in subsequent iterations and both methods converge after a certain number of iterations. To further illustrate their classification performance, we report the test results in terms of MZE and MAE versus the number of iterations

for LODML and KODML in Fig. 10.4. Using the Euclidean distance metric, the 3-NN classifier obtains an MZE = 0.2087 and MAE = 0.2699. We can observe that minimizing the objective functions in (10.6) and (10.8) results in improving the classification performance. After 500 iterations, both LODML and KODML obtain significantly better results than the baseline Euclidean distance metric.

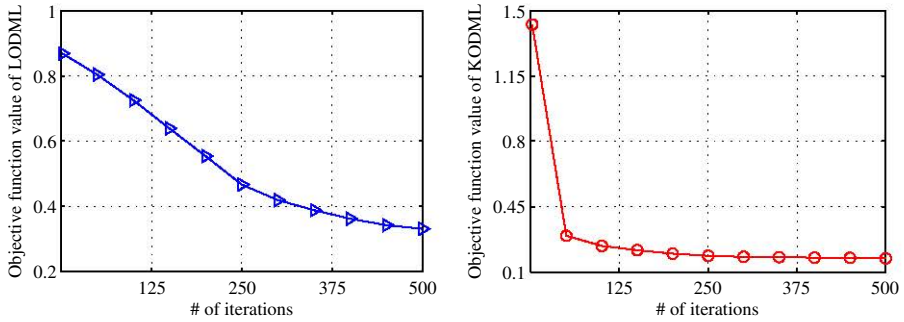


Figure 10.3: Objective function value vs. number of iterations on the *balance-scale* data set for LODML and KODML.

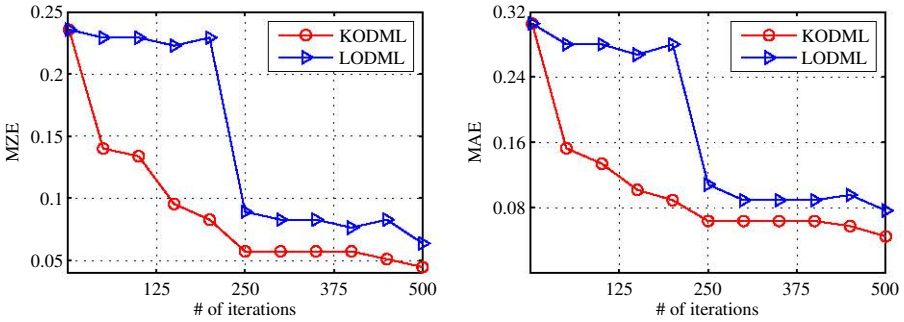


Figure 10.4: Test results (MZE and MAE) vs. number of iterations on the *balance-scale* data set for LODML and KODML.

10.5.6. Influence of neighborhood size

Since our approach uses only the information from the local neighborhood of each training example, it is interesting to analyze the influence of the neighborhood size on the performance. For this purpose, we perform an experiment on the *ESL* data set with a varying number of nearest neighbors used to build the neighborhood of each training example. Figure 10.5 shows the performance of LODML in terms of MAE versus the neighborhood size. From the left panel in this figure, we can observe that an increasing number of neighbors implies an increasing performance,

which becomes relatively stable once the number of neighbors is sufficiently large (but still relatively small). The latter implies that adding more constraints increases the training cost without such benefit. This means the number of neighbors can be kept low, which is important, since the number of triplet constraints could otherwise become very large, leading to a high computational complexity, which is intractable for large-scale problems. From the right panel in Fig. 10.5, we can observe that, when the neighborhood size is relatively small, LODML is much faster than the conventional method LMNN while keeping a similar performance.

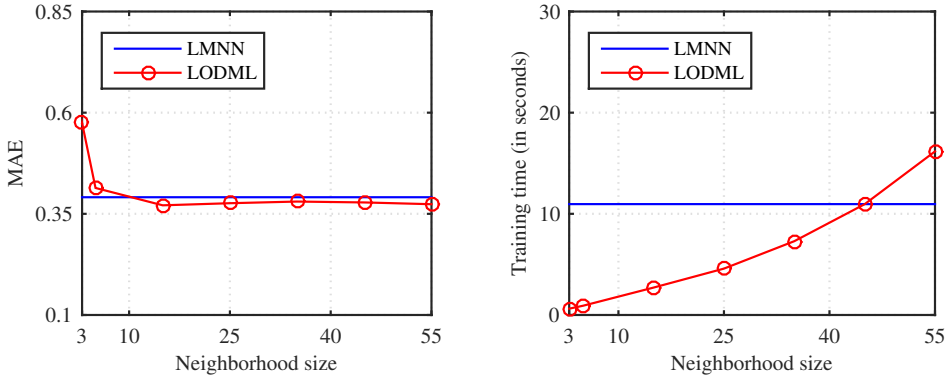


Figure 10.5: MAE and training time vs. the neighborhood size on the *ESL* data set for LODML compared to LMNN.

10.6. Conclusion

We have proposed a novel distance metric learning approach (LODML) for ordinal classification problems. We argued that the ordinal relationships can be preserved by satisfying triplet constraints derived from the local neighborhood of each training example. Compared to previous approaches, our approach does not make any assumption about the absolute distances between the class labels, making it more robust and suitable for ordinal classification tasks. To validate this claim, we have carried out extensive experiments on a set of publicly available benchmark data sets. The experimental results have been analyzed using several standard performance measures, allowing to capture different aspects of the prediction capability of ordinal classifiers. Moreover, we have proposed a kernelized version of LODML to tackle the nonlinearities usually encountered in many complex problems. It is important to point out that our strategy to preserve the ordinal relationships can be also incorporated in other existing distance metric learning approaches. More importantly, our framework can be used to guide further development of distance metric learning in ordinal settings.

PART IV

DISTANCE METRIC LEARNING FOR CLUSTERING

11 Kernel-based distance metric learning for supervised k-means clustering

Finding an appropriate distance metric that accurately reflects the (dis)similarity between examples is a key to the success of k-means clustering. While it is not always an easy task to specify a good distance metric, we can try to learn one based on prior knowledge from some available clustered data sets, an approach that is referred to as supervised clustering. In this chapter, a kernel-based distance metric learning method is developed to improve the practical use of k-means clustering. Given the corresponding optimization problem, we derive a meaningful Lagrange dual formulation and introduce an efficient algorithm in order to reduce the training complexity. Our formulation is simple to implement, allowing a large-scale distance metric learning problem to be solved in a computationally tractable way. Experimental results show that the proposed method yields more robust and better performances on synthetic as well as real-world data sets compared to other state-of-the-art distance metric learning methods.

The material of this chapter is based on the following publication:
Nguyen, B. and De Baets, B. (2018b). Kernel-based distance metric learning for supervised k-means clustering. *IEEE Transactions on Neural Networks and Learning Systems*, accepted

11.1. Introduction

Clustering is an important task in pattern recognition for data analysis. Among various clustering techniques, k-means clustering (Lloyd, 1982) is one of the most popular and most efficient techniques for general clustering tasks. The goal is to partition a set of examples into disjoint clusters based on some notion of (dis)similarity, such that related examples belong to the same cluster, while unrelated examples belong to different clusters (Huang et al., 2014). Despite its apparent simplicity, it is not always clear how to select "related" examples since there are many possible ways of defining the similarity of examples for a given task, e.g. by using different similarity measures or distance metrics. It is well known that the Euclidean distance metric may not be a good choice for a given task because it simply ignores the correlations between features, which usually contain useful discriminative information (Nguyen et al., 2016; Shen et al., 2014; Weinberger and Saul, 2009; Nguyen and De Baets, 2018a). Depending on the application domain, one wishes to learn a distance metric that satisfies some specific requirements. Typical applications include classification (Faraki et al., 2018; Weinberger and Saul,

2009; Goldberger et al., 2005; Nguyen et al., 2018a), regression (Nguyen et al., 2016), clustering (Jia et al., 2016; Wu et al., 2012; Bilenko et al., 2004; Yin et al., 2010), ranking (Lim and Lanckriet, 2014), and kernel learning (Jain et al., 2012). Due to its flexibility in parameterization, we focus on learning a Mahalanobis distance metric for k-means clustering in a supervised setting.

Formally, supervised clustering is the problem of training a clustering algorithm with some supervision information, so that it can produce a desirable clustering for unseen data (Finley and Joachims, 2005; Daumé and Marcu, 2005). Unlike traditional clustering problems, which are usually referred to as unsupervised clustering, here we have sets of examples and complete clusterings over these sets. By adjusting the distance metric to obtain appropriate clusterings on supervised data, one hopes the distance-based algorithm to cluster unseen data in a similar fashion. Supervised clustering is closely related to semi-supervised clustering. In semi-supervised clustering, the supervision information is typically incomplete and is often provided in the form of pairwise constraints (Xing et al., 2002), e.g. "examples $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same cluster" (must-link constraints) or "examples $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to different clusters" (cannot-link constraints). Other kinds of supervision like triplet constraints, e.g. "example $\mathbf{x}_i$ is more similar to example $\mathbf{x}_j$ than to example $\mathbf{x}_l$," have also been considered in the literature (Schultz and Joachims, 2004; Kumar and Kummamuru, 2008). However, most of the existing semi-supervised methods simply attempt to satisfy the constraints derived from a small amount of labeled data for a single problem. Therefore, it is usually not reasonable to transfer the knowledge learned from a set of training labels to another set of testing labels (Finley and Joachims, 2005). On the other hand, supervised clustering can be seen as a special case of multi-class classification in the sense that both approaches try to classify related examples into the same class and unrelated examples into different classes (Finley and Joachims, 2005, 2008). Nevertheless, supervised clustering can also be used for problems containing new labels that have not been seen during the training, which seems impossible with multi-class classification.

Over the last few years, there has been a growing interest in learning a distance metric in order to improve the clustering performance (Finley and Joachims, 2005; Xing et al., 2002; Lajugie et al., 2014; Law et al., 2016). A common assumption is that the available training examples share the same distance metric as that of test examples, which is then used by a distance-based clustering algorithm. For instance, Lajugie et al. (2014) adopted the large-margin structured prediction framework (Tsochantaridis et al., 2005) in a supervised way (LMMLCP) to optimize the objective of clustering through the use of a Mahalanobis distance metric (or, equivalently, a linear transformation). This framework was then applied in different domains such as video segmentation, image segmentation, and detection of change points in DNA sequences. Unfortunately, in many real-world problems, data are often nonlinearly separable, which can be challenging for LMMLCP. Despite

its simplicity and generalization abilities, LMMLCP may fail when dealing with high-dimensional data sets since the number of parameters increases quadratically with the dimensionality. A solution to address this problem is based on kernel embedding methods. An additional benefit of using kernel methods lies in the fact that they have been successfully employed for several types of structured data (even without having a vectorial representation) (Schölkopf and Smola, 2001). Although there exist several kernel-based distance metric learning methods (Jain et al., 2012; Davis et al., 2007; Jain et al., 2010; Wang et al., 2015), they do not necessarily improve the separability of the data for clustering (Yin et al., 2010; Baghshah and Shouraki, 2010a).

To overcome the above shortcomings, we consider the same goal as in (Finley and Joachims, 2005; Lajugie et al., 2014) to learn a Mahalanobis distance metric in the feature space induced by a nonlinear kernel function, making it more flexible and effective. In particular, given a set of related data sets with known partitions, we aim to learn a distance metric that will lead to these partitions when k-means clustering is performed. The main contributions of this work are summarized below:

(i) A nonlinear distance metric learning method for k-means clustering is presented. Although our work is related to that of Lajugie et al. (2014), a novel formulation is introduced such that partitions induced by k-means clustering can be optimized more appropriately in the feature space. More specifically, learning the Mahalanobis distance metric is considered as a structured learning problem.

(ii) Unlike existing kernel-based methods, we enforce the low-rank constraint on the solution by adding the trace norm to improve the generalization ability. As a consequence, the resulting distance metric implicitly performs feature selection (Recht et al., 2010). We refer to the proposed method as Kernel-based Distance Metric Learning for Supervised k-means Clustering (KDMLSC).

(iii) To accelerate the structured support vector machine (SVM) solver, a simple and scalable algorithm is developed to solve the training problem efficiently. Our algorithm is based on the Lagrange dual formulation and converges to an optimal solution. In particular, we employ the block-coordinate descent technique, which iteratively solves each subproblem in an efficient manner.

(iv) Experimental results on synthetic and real-world data sets show that the proposed method is more effective than the one introduced in (Lajugie et al., 2014) and other state-of-the-art distance metric learning methods for clustering tasks.

The remainder of this chapter is organized as follows. We first motivate our method by giving a brief discussion of related works in Section 11.2. Then, we

review the spectral relaxation for k-means clustering in Section 11.3. Our kernel-based distance metric learning method is described in Section 11.4. Results of an extensive experimental evaluation and comparison are presented in Section 11.5, followed by some concluding remarks in Section 11.6.

11.2. Related work

The performance of many clustering methods heavily depends on the choice of the distance metric (Xing et al., 2002; Lajugie et al., 2014), but this choice is generally not treated as a part of the training process. Without any supervision, such distance-based clustering methods may not guarantee to achieve a good partition of unlabeled data. In order to alleviate this problem, several approaches have incorporated prior knowledge into either the distance metric for k-means clustering (Xing et al., 2002; Bar-Hillel et al., 2005; Yin et al., 2010) or the similarity matrix for spectral clustering (Wagstaff et al., 2001; Bach and Jordan, 2003). A common goal is to minimize the distances between related examples and to maximize the distances between unrelated examples. Below, we will discuss some relevant methods that are closely related to ours.

Most of the existing methods fall into the semi-supervised category, where some partial constraints about the clustering are given. In an early work, Xing et al. (2002) formulated the distance metric learning problem as a convex optimization problem subject to a set of pairwise constraints. Despite its popularity, this method obtains a relatively poor performance compared to more recent methods and the resulting formulation is slow to optimize. Bar-Hillel et al. (2005) used must-link constraints only to learn a whitening transformation of the within-chunklet covariance matrix. Their method has the advantage of being simple to implement, but it does not make use of cannot-link constraints, which carry much more discriminative information. Also based on pairwise constraints, Yin et al. (2010) introduced a nonlinear semi-supervised clustering method that makes examples in the must-link constraints belong to the same cluster while those in the cannot-link constraints belong to different clusters. Similarly, Kulis et al. (2009a) extended a probabilistic framework for semi-supervised clustering (Basu et al., 2004) to handle graph-based clustering using a kernel approach. As an alternative, Bilenko et al. (2004) combined k-means clustering and distance metric learning in a unified framework. A fundamental limitation of these semi-supervised methods is that they cannot fully explore additional prior knowledge contained in the partitions when several training sets are available.

In order to overcome the above shortcomings, various supervised clustering methods have been proposed (Finley and Joachims, 2005, 2008; Lajugie et al., 2014; Law et al., 2016; Bach and Jordan, 2003). Unlike semi-supervised clustering, these methods consider all the possible must-link as well as cannot-link constraints

and take into account the global clustering structure of the training data sets. For instance, Bach and Jordan (2003) provided a general framework for learning a similarity matrix for spectral clustering. Given a partition, the objective is to minimize the error between the target partition and the solution derived from the spectral relaxation. Due to the nonconvexity of the objective function, this method may suffer from poor local minima. Based on the structured SVM framework (Tsochantaridis et al., 2005), Finley and Joachims (2005) learned a similarity function to improve the performance of correlation clustering. Analogously, Lajugie et al. (2014) proposed a large-margin distance metric learning method for constrained partitioning problems. However, they provide no kernel extension and neither solve the training problem efficiently. In an extension of the latter, Law et al. (2016) derived a closed-form solution when there exists only a single data set for training. As mentioned earlier, these methods rely on a linear transformation, limiting their applicability for complex or nonlinearly separable data. To overcome this limitation, some kernel-based methods have been introduced (Finley and Joachims, 2008; Baghshah and Shouraki, 2010a; Chatpatanasiri et al., 2010; Yeung and Chang, 2007). For instance, Finley and Joachims (2008) formulated k-means clustering in the feature space by learning a weighting function, but it cannot yield a sufficiently flexible model. In contrast, our method intrinsically provides both desirable properties: (1) learning a distance metric in a nonlinear feature space and (2) optimizing the desired clusterings in a unified framework. Moreover, we try to enforce the low-rank solution in the feature space, making it less sensitive to overfitting.

11.3. Spectral relaxation of k-means clustering

In this section, we recall the spectral relaxation for k-means clustering, which was previously presented in (Zha et al., 2002; Ng et al., 2002). Given a set of examples $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{D \times n}$, the goal of k-means clustering is to find an assignment of these examples into k disjoint sets, which leads to a minimal sum of squared distances between the examples and their corresponding cluster center. Let $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_k] \in \mathbb{R}^{D \times k}$ be k center vectors and $\mathbf{Y} \in \{0, 1\}^{k \times n}$ denote the assignment matrix where $Y_{c,i} = 1$ if example $\mathbf{x}_i$ belongs to the c-th cluster, otherwise $Y_{c,i} = 0$. Following Peng and Wei (2007), the objective of k-means clustering can be formulated as

$$\begin{array}{ll} \underset{\mathbf{Y},\mathbf{Z}}{\text{minimize}} & \sum_{i=1}^{n} \sum_{c=1}^{k} Y_{c,i} \left\| \mathbf{x}_i - \mathbf{z}_c \right\|_F^2 \\ \text{subject to} & \mathbf{Y} \in \{0, 1\}^{k \times n}, \operatorname{rank}(\mathbf{Y}) = k \,, \mathbf{Y}^\top \mathbf{1} = \mathbf{1} \,, \\ & \mathbf{Z} \in \mathbb{R}^{D \times k} \,. \end{array} \tag{11.1}$$

This problem is a mixed integer program with a nonlinear objective function, which is NP-hard (Aloise et al., 2009). This is due to the fact that the constraints are discrete and the objective function is nonconvex and nonlinear, making the problem very challenging. To deal with these difficulties, the k-means clustering algorithm minimizes the objective function in (11.1) using the block-coordinate descent technique. Despite its popularity, the k-means clustering algorithm can be easily prone to local minima. More importantly, it is very sensitive to noise as well as initialization. Although one can find an optimal solution to problem (11.1) by employing the notion of Voronoi partition (Inaba et al., 1994), the computational complexity scales as $O(n^{kD+1})$, which is not practical for medium- and large-sized problems.

Adopting matrix notation, the objective function in (11.1) can be rewritten as $\|\mathbf{X} - \mathbf{Z}\mathbf{Y}\|_F^2$. Given $\mathbf{Y}$, according to Yu and Schuurmans (2011), the optimal value of $\mathbf{Z}$ for problem (11.1) is $\mathbf{Z} = \mathbf{X}\mathbf{Y}^\dagger = \mathbf{X}\mathbf{Y}^\top(\mathbf{Y}\mathbf{Y}^\top)^{-1}$. Thus, problem (11.1) becomes

$$\underset{\mathbf{C}\in\mathcal{C}_k}{\text{minimize}} \quad \|\mathbf{X} - \mathbf{X}\mathbf{C}\|_F^2\,, \tag{11.2}$$

where $\mathcal{C}_k = \{\mathbf{Y}^\top(\mathbf{Y}\mathbf{Y}^\top)^{-1}\mathbf{Y} \mid \mathbf{Y} \in \{0,1\}^{k\times n}, \mathrm{rank}(\mathbf{Y}) = k\,, \mathbf{Y}^\top\mathbf{1} = \mathbf{1}\}$. It is important to remark that each matrix $\mathbf{C} \in \mathcal{C}_k$ (*rescaled equivalence matrix*) has a special structure. That is, $C_{ij} = 1/n_c$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the c-th cluster, otherwise $C_{ij} = 0$. It is easy to see that $\mathbf{C} \succcurlyeq 0$; $\mathbf{C}^2 = \mathbf{C}$, i.e. $\mathbf{C}$ has eigenvalues in $\{0,1\}$; and $\mathrm{tr}(\mathbf{C}) = k$, i.e. the number of eigenvalues being equal to one is the number of clusters. To make problem (11.2) more tractable, we relax the constraints in $\mathcal{C}_k$ to

$$\widehat{\mathcal{C}}_k = \{\mathbf{C} \mid \mathrm{tr}(\mathbf{C}) = k,\, \mathbf{C}^2 = \mathbf{C},\, \mathbf{C} = \mathbf{C}^\top\}\,.$$

Note that the positive semidefiniteness constraint on $\mathbf{C}$ is automatically satisfied. By simplifying the independent terms, we rewrite problem (11.2) as

$$\underset{\mathbf{C}\in\widehat{\mathcal{C}}_k}{\text{maximize}} \quad \langle \mathbf{C}, \mathbf{X}^\top\mathbf{X}\rangle\,, \tag{11.3}$$

since $\|\mathbf{X}-\mathbf{X}\mathbf{C}\|_F^2 = \mathrm{tr}(\mathbf{X}^\top\mathbf{X}(\mathbf{I}-\mathbf{C})(\mathbf{I}-\mathbf{C})^\top) = \mathrm{tr}(\mathbf{X}^\top\mathbf{X}(\mathbf{I}-\mathbf{C}))$. As proved by Zha et al. (2002), problem (11.3) has a closed-form solution, which is the orthogonal projector onto the k leading eigenvectors of $\mathbf{X}^\top\mathbf{X}$. Once the relaxed solution is obtained, a possible way to estimate $\mathbf{Y}$ is to run the k-means clustering algorithm over the k leading eigenvectors of $\mathbf{X}^\top\mathbf{X}$ as suggested in (Ng et al., 2002).

11.4. Proposed method

In this section, we formulate the problem of learning a Mahalanobis distance metric for k-means clustering in the structured SVM framework (Tsochantaridis et al., 2005). Our method operates in the feature space induced by a nonlinear kernel function. Subsequently, we present the optimization algorithm to solve the subproblems derived from each iteration of the structured SVM solver. Finally, a heuristic simplification is introduced in order to make our algorithm scalable when increasing the number of training examples.

11.4.1. Problem formulation

Let $\mathcal{X}$ denote an input space containing all possible sets of examples and let $\mathcal{Y}$ denote an output space containing all possible partitions of those sets. The set of training sets is denoted by $\mathcal{D} = \{(\mathbf{X}_i, \mathbf{Y}_i)\}_{i=1}^{m} \subset \mathcal{X} \times \mathcal{Y}$, which consists of m sets of training examples $\mathbf{X}_i = [\mathbf{x}_{i,1}, \dots, \mathbf{x}_{i,n_i}] \in \mathbb{R}^{D \times n_i}$ and their corresponding clustering $\mathbf{Y}_i = [\mathbf{y}_{i,1}, \dots, \mathbf{y}_{i,n_i}] \in \{0,1\}^{k_i \times n_i}$. Following the structured output prediction scheme, the goal is to learn a clustering function $h\colon \mathcal{X} \to \mathcal{Y}$ such that, given a set of examples $\mathbf{Q} \in \mathcal{X}$, its corresponding clustering is computed as

$$h(\mathbf{Q}) = \operatorname*{argmax}_{\mathbf{O} \in \mathcal{Y}} \quad F_{\mathbf{M}}(\mathbf{Q}, \mathbf{O})\,,$$

where $F_{\mathbf{M}}\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ denotes a linear function parameterized by $\mathbf{M}$, which characterizes the relationship between $\mathbf{Q}$ and a clustering output $\mathbf{O}$. Here, we adopt the k-means objective function in (11.3) to define $F_{\mathbf{M}}$, which should give the highest value for the correct clustering output. In other words, given a training set $\mathcal{D}_i = (\mathbf{X}_i, \mathbf{Y}_i)$, we aim at learning a Mahalanobis matrix $\mathbf{M} \succcurlyeq 0$ that satisfies

$$\forall \mathbf{C} \in \widehat{\mathcal{C}}_{k_i} \setminus \{\mathbf{C}_i\}: \quad \langle \mathbf{M}, \mathbf{X}_i \mathbf{C}_i \mathbf{X}_i^\top \rangle > \langle \mathbf{M}, \mathbf{X}_i \mathbf{C} \mathbf{X}_i^\top \rangle\,, \tag{11.4}$$

where $\mathbf{C}_i = \mathbf{Y}_i^\top (\mathbf{Y}_i \mathbf{Y}_i^\top)^{-1} \mathbf{Y}_i$ is the rescaled equivalence matrix (Lajugie et al., 2014). By appropriately adjusting $\mathbf{M}$, we can force the correct clustering $\mathbf{Y}_i$ (or, equivalently, the matrix $\mathbf{C}_i$) of $\mathbf{X}_i$ to have the highest score under the parameterized k-means objective in (11.3). However, if the data are nonlinearly separable, the resulting matrix $\mathbf{M}$ may not be powerful enough to employ the desired clusterings. As a result, we are interested in learning a distance metric in a nonlinear feature space to alleviate this problem.

Formally, let $\mathbf{X} = [\mathbf{X}_1, \dots, \mathbf{X}_m] \in \mathbb{R}^{D \times n}$ with $n = \sum_{i=1}^{m} n_i$ be the input matrix containing all training examples and $\mathbf{\Phi} = [\mathbf{\Phi}_1, \dots, \mathbf{\Phi}_m]$, where $\mathbf{\Phi}_i = [\phi(\mathbf{x}_{i,1}), \dots, \phi(\mathbf{x}_{i,n_i})]$, be the transformed matrix containing all training examples in the feature space $\mathcal{F}$ induced by a nonlinear function $\phi\colon \mathbb{R}^D \to \mathcal{F}$. The goal is to learn $\mathbf{M}$ in the feature space $\mathcal{F}$. Following the Representer Theorem (Schölkopf

et al., 2001), the optimal linear transformation induced by $\mathbf{M}$ lies within the span of all training examples. Accordingly, the optimal matrix $\mathbf{M}$ has the following form

$$\mathbf{M} = \mathbf{\Phi}\mathbf{W}\mathbf{\Phi}^\top ,$$

where $\mathbf{W} \in \mathbb{R}^{n\times n}$ and $\mathbf{W} \succcurlyeq 0$. The latter condition is to guarantee that $\mathbf{M}$ is PSD. By doing so, learning $\mathbf{M}$ amounts to learning $\mathbf{W}$. Alternatively, the problem can be viewed as learning a parameterized kernel function $\mathcal{K}_{\mathbf{W}}(\mathbf{u},\mathbf{v}) = \phi(\mathbf{u})^\top \mathbf{W}\phi(\mathbf{v})$ given some input kernel function $\mathcal{K}(\mathbf{u},\mathbf{v}) = \phi(\mathbf{u})^\top\phi(\mathbf{v})$ (for a more detailed discussion see Jain et al., 2012). Consequently, we rewrite the constraints in (11.4) as

$$\forall \mathbf{C} \in \widehat{\mathcal{C}}_{k_i} \setminus \{\mathbf{C}_i\} : \quad \langle \mathbf{W}, \mathbf{K}_i\mathbf{C}_i\mathbf{K}_i^\top \rangle > \langle \mathbf{W}, \mathbf{K}_i\mathbf{C}\mathbf{K}_i^\top \rangle ,$$

where $\mathbf{K} = \mathbf{\Phi}^\top\mathbf{\Phi} = [\mathbf{\Phi}^\top\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}^\top\mathbf{\Phi}_m] = [\mathbf{K}_1, \ldots, \mathbf{K}_m]$ denotes the kernel matrix. The kernel trick allows us to implicitly compute the dot products in $\mathcal{F}$ without mapping the input examples into $\mathcal{F}$. Following the large-margin framework of structured SVM (Tsochantaridis et al., 2005), we formulate our nonlinear distance metric learning problem for supervised clustering as

$$\begin{array}{ll} \underset{\mathbf{W}}{\text{minimize}} & r(\mathbf{W}) + \gamma \sum_{i=1}^m \beta_i \\ \text{subject to} & \forall \mathbf{C} \in \widehat{\mathcal{C}}_{k_i} \setminus \{\mathbf{C}_i\}: \\ & \langle \mathbf{W}, \mathbf{K}_i(\mathbf{C}_i - \mathbf{C})\mathbf{K}_i^\top \rangle \geq \ell(\mathbf{C},\mathbf{C}_i) - \beta_i , \\ & \beta_i \geq 0 , \quad \text{for} \quad i = 1, \ldots, m ; \\ & \mathbf{W} \succcurlyeq 0 , \end{array} \tag{11.5}$$

where $\ell\colon \widehat{\mathcal{C}}_k \times \widehat{\mathcal{C}}_k \to \mathbb{R}^+$ is a loss function that penalizes the violation of clustering; $r\colon \mathbb{S}^+ \to \mathbb{R}^+$ is a regularization function; $\beta_i \geq 0$ are slack variables; and $\gamma \geq 0$ is a hyperparameter.

Regularization function: In order to reduce the risk of overfitting, the regularization function in (11.5) is defined as

$$r(\mathbf{W}) = \frac{1}{2}\|\mathbf{W}\|_F^2 + \lambda \operatorname{tr}(\mathbf{W}) ,$$

where $\lambda \geq 0$ is a hyperparameter. This is also known as Elastic-Net regularization (Li et al., 2012b). The main reasons for selecting such regularization are the following. First, the Frobenius norm can lead to fast, simple and scalable optimization. Second, minimizing the trace norm can yield a sparse solution in eigenspectrum (Recht et al., 2010). The trace norm has been extensively studied in (Jain et al., 2010) for learning a kernel matrix. Since both the Frobenius and trace norm are convex functions, problem (11.5) results in a convex semidefinite program. Combining these two norms can be considered as a trade-off between

sparsity and efficiency, leading to stability of an optimization framework.

Loss function: Let $\mathbf{P}$ and $\mathbf{Q}$ denote two rescaled equivalence matrices in $\widehat{\mathcal{C}}_{k_i}$, then the following loss function is considered

$$\ell(\mathbf{P}, \mathbf{Q}) = \|\mathbf{P} - \mathbf{Q}\|_F^2 = \operatorname{tr}(\mathbf{P}) + \operatorname{tr}(\mathbf{Q}) - 2\operatorname{tr}(\mathbf{PQ})\,. \tag{11.6}$$

As explained in (Lajugie et al., 2014), unlike the loss function associated with the Rand index (Finley and Joachims, 2005), the Frobenius norm loss function takes into account the size of the clusters, avoiding the domination of the problem by the largest clusters. This loss function has already been used to measure the dissimilarity between two partitions (Bach and Jordan, 2003).

Since the cardinality of $\mathcal{Y}$ is exponential in the number of training examples, the structured SVM optimization is used to solve problem (11.5). More specifically, our algorithm is an adaptation of the 1-slack margin-rescaling cutting-plane algorithm proposed by Joachims et al. (2009). A pseudocode is given in Algorithm 12.

Algorithm 12 Cutting plane algorithm for 1-slack formulation

Input: Training set $\{(\mathbf{X}_i, \mathbf{Y}_i)\}_{i=1}^m$, λ, ϵ
Output: A matrix $\mathbf{W} \succcurlyeq 0$

Compute $\mathbf{K}_i$ and $\mathbf{C}_i$ using $\mathbf{X}_i$ and $\mathbf{Y}_i$
$\mathcal{S} \leftarrow \emptyset$
repeat

$$\begin{aligned} (\mathbf{W}, \xi) \quad &\leftarrow \underset{\mathbf{W}\succcurlyeq 0,\, \xi \geq 0}{\operatorname{argmin}} \quad \|\mathbf{W}\|_F^2 + \lambda \operatorname{tr}(\mathbf{W}) + \gamma\, \xi \\ \text{subject to} \quad &\forall (\overline{\mathbf{C}}_1, \ldots, \overline{\mathbf{C}}_n) \in \mathcal{S} : \\ &\tfrac{1}{m} \left\langle \mathbf{W}, \textstyle\sum_{i=1}^m \mathbf{K}_i (\mathbf{C}_i - \overline{\mathbf{C}}_i) \mathbf{K}_i^\top \right\rangle \\ &\qquad\qquad \geq \tfrac{1}{m} \textstyle\sum_{i=1}^m \ell(\mathbf{C}_i, \overline{\mathbf{C}}_i) - \xi \end{aligned} \tag{11.7}$$

  for $i = 1, \ldots, m$ **do**
    $\widehat{\mathbf{C}}_i \leftarrow \underset{\widehat{\mathbf{C}} \in \widehat{\mathcal{C}}_k}{\operatorname{argmax}} \quad \ell(\mathbf{C}_i, \widehat{\mathbf{C}}) + \langle \widehat{\mathbf{C}} - \mathbf{C}_i, \mathbf{K}_i^\top \mathbf{W} \mathbf{K}_i \rangle$
  end for
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{(\widehat{\mathbf{C}}_1, \ldots, \widehat{\mathbf{C}}_m)\}$

until $\dfrac{1}{m} \displaystyle\sum_{i=1}^m \ell(\mathbf{C}_i, \widehat{\mathbf{C}}_i) - \dfrac{1}{m} \left\langle \mathbf{W}, \displaystyle\sum_{i=1}^m \mathbf{K}_i (\mathbf{C}_i - \widehat{\mathbf{C}}_i) \mathbf{K}_i^\top \right\rangle \leq \xi + \epsilon$

Essentially, the proposed algorithm iteratively constructs a working set of constraints $\mathcal{S}$. In each iteration, it finds the most violated constraint and if the violation is larger than a desired precision $\epsilon > 0$, then the constraint is added to the working set. Subsequently, it optimizes the problem based on the current working set of constraints. The algorithm terminates when no constraint is added to $\mathcal{S}$. In other words, the solution found by our algorithm is correct up to a certain approximation that depends on ϵ. In most machine learning applications, tolerating

the optimal solution by a small value can be acceptable. The theoretical analysis for the correctness, convergence rate, and scaling behavior of the structured SVM algorithm can be found in (Joachims et al., 2009). This algorithm is efficient whenever the most violated constraint (which is called the separation oracle) can be found efficiently from the working set of constraints.

Finding the separation oracle can be expressed as

$$\underset{\widehat{\mathbf{C}} \in \widehat{\mathcal{C}}_{k_i}}{\text{maximize}} \quad \ell(\mathbf{C}_i, \widehat{\mathbf{C}}) + \langle \widehat{\mathbf{C}} - \mathbf{C}_i, \mathbf{K}_i^\top \mathbf{W} \mathbf{K}_i \rangle \,,$$

which is equivalent to

$$\underset{\widehat{\mathbf{C}} \in \widehat{\mathcal{C}}_{k_i}}{\text{maximize}} \quad \langle \widehat{\mathbf{C}}, \mathbf{I} + \mathbf{K}_i^\top \mathbf{W} \mathbf{K}_i - 2\mathbf{C}_i \rangle \,.$$

Similarly to problem (11.3), the solution to this problem is the orthogonal projector onto the k_i leading eigenvectors of $\mathbf{I} + \mathbf{K}_i^\top \mathbf{W} \mathbf{K}_i - 2\mathbf{C}_i$.

11.4.2. A dual approach to distance metric learning

Algorithm 12 requires the solution of a semidefinite program at each iteration. In this subsection, we will show how to solve problem (11.7) efficiently. For simplicity in notation, let t denote the number of constraints and let the i-th combination of rescaled equivalence matrices in $\mathcal{S}$ be $(\overline{\mathbf{C}}_1^{(i)}, \ldots, \overline{\mathbf{C}}_m^{(i)})$. We will denote $\mathbf{S}_i = \frac{1}{m} \sum_{j=1}^m \mathbf{K}_j (\mathbf{C}_j - \overline{\mathbf{C}}_j^{(i)}) \mathbf{K}_j^\top$ and $l_i = \frac{1}{m} \sum_{j=1}^m \ell(\mathbf{C}_j, \overline{\mathbf{C}}_j^{(i)})$ for $i = 1, \ldots, t$. Using these notations, problem (11.7) can be rewritten as

$$\begin{array}{ll} \underset{\mathbf{W}}{\text{minimize}} & \frac{1}{2}\|\mathbf{W}\|_F^2 + \lambda \operatorname{tr}(\mathbf{W}) + \gamma\xi \\ \text{subject to} & \langle \mathbf{W}, \mathbf{S}_i \rangle \geq l_i - \xi \,, \text{ for } i = 1, \ldots, t \,; \\ & \xi \geq 0 \,; \\ & \mathbf{W} \succcurlyeq 0 \,. \end{array} \tag{11.8}$$

Note that this formulation contains only one slack variable ξ. In order to solve the above problem, we introduce the Lagrange multipliers $\mathbf{V} \succcurlyeq 0$, $\alpha \geq 0$, and $\mu_i \geq 0$ for $i = 1, \ldots, t$, and obtain its Lagrangian as

$$\begin{aligned} &\mathcal{L}(\underbrace{\mathbf{W}, \xi}_{\text{primal}}, \underbrace{\mathbf{V}, \boldsymbol{\mu}, \alpha}_{\text{dual}}) \\ &= \frac{1}{2}\|\mathbf{W}\|_F^2 + \lambda \operatorname{tr}(\mathbf{W}) + \gamma\xi \\ &\qquad - \sum_{i=1}^t \mu_i \Big[\langle \mathbf{W}, \mathbf{S}_i \rangle - l_i + \xi \Big] - \alpha\xi - \langle \mathbf{V}, \mathbf{W} \rangle \,. \end{aligned}$$

Setting the derivatives of $\mathcal{L}$ w.r.t. the primal variables (i.e. $\mathbf{W}$ and ξ) equal to zero yields the following equations

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}}(\mathbf{W}, \boldsymbol{\xi}, \mathbf{V}, \boldsymbol{\mu}, \alpha) = 0\,,$$

resulting in

$$\mathbf{W} = \mathbf{V} + \sum_{i=1}^{t} \mu_i \mathbf{S}_i - \lambda \mathbf{I}\,, \tag{11.9}$$

and

$$\frac{\partial \mathcal{L}}{\partial \xi}(\mathbf{W}, \boldsymbol{\xi}, \mathbf{Z}, \boldsymbol{\mu}, \alpha) = 0\,,$$

resulting in

$$\sum_{i=1}^{t} \mu_i = \gamma - \alpha\,, \tag{11.10}$$

which implies

$$\sum_{i=1}^{t} \mu_i \leq \gamma\,. \tag{11.11}$$

From (11.9), it follows that

$$\langle \mathbf{V}, \mathbf{W} \rangle = \|\mathbf{W}\|_F^2 - \sum_{i=1}^{t} \mu_i \langle \mathbf{W}, \mathbf{S}_i \rangle + \lambda \operatorname{tr}(\mathbf{W})\,. \tag{11.12}$$

Substituting Eqs. (11.9)–(11.12) back into the Lagrangian, we get the following dual problem

$$\begin{array}{ll} \underset{\mathbf{V},\,\boldsymbol{\mu}}{\text{minimize}} & \frac{1}{2}\left\| \mathbf{V} + \sum_{i=1}^{t} \mu_i \mathbf{S}_i - \lambda \mathbf{I} \right\|_F^2 - \sum_{i=1}^{t} \mu_i l_i \\ \text{subject to} & \sum_{i=1}^{t} \mu_i \leq \gamma\,, \mu_i \geq 0\,, \text{for } i = 1, \ldots, t\,; \\ & \mathbf{V} \succcurlyeq 0\,. \end{array} \tag{11.13}$$

Note that the strong duality holds since the primal problem in (11.8) is a convex program and satisfies Slater's condition (Boyd and Vandenberghe, 2004). This implies that we can solve the primal problem by solving the dual. Although problem (11.13) still has a positive semidefiniteness constraint, we can address it efficiently using the block-coordinate descent algorithm (Tseng, 2001). More specifically, we first fix $\mathbf{V}$ and solve the dual problem in μ_i for $i = 1, \ldots, t$. Then, we fix $\boldsymbol{\mu}$ and solve the dual problem in $\mathbf{V}$. By updating $\mathbf{V}$ and $\boldsymbol{\mu}$ alternatingly, we

can find the optimal solution of the dual problem. Next, we will explain in detail how to perform these updates.

The optimization procedure starts from an initial point $(\mathbf{V}^{(0)}, \boldsymbol{\mu}^{(0)})$ and generates a sequence of solutions $\{(\mathbf{V}^{(k)}, \boldsymbol{\mu}^{(k)})\}_{k=1}^{\infty}$. By fixing $\mathbf{V}^{(k)}$, problem (11.13) becomes the following convex quadratic program:

$$\begin{array}{ll} \underset{\boldsymbol{\mu}}{\text{minimize}} & f(\boldsymbol{\mu}) = \frac{1}{2}\sum_{i=1}^{t}\sum_{j=1}^{t}\mu_i\mu_j\langle \mathbf{S}_i, \mathbf{S}_j\rangle + \sum_{i=1}^{t}\mu_i a_i \\ \text{subject to} & \sum_{i=1}^{t}\mu_i \leq \gamma\,, \mu_i \geq 0\,, \text{for } i = 1, \ldots, t \end{array} \tag{11.14}$$

with $a_i = \langle \mathbf{V}^{(k)} - \lambda\mathbf{I}, \mathbf{S}_i\rangle - l_i$, for $i = 1, \ldots, t$. Now, we need to update each component of $\boldsymbol{\mu}^{(k)}$ by minimizing (11.14) over each of $\mu_1, \ldots, \mu_t$, while fixing the remaining components at their last updated values. We start by introducing some notations in order to simplify the description. Let

$$\boldsymbol{\mu}^{(k,i)} = [\mu_1^{(k+1)}, \ldots, \mu_{i-1}^{(k+1)}, \mu_i^{(k)}, \ldots, \mu_t^{(k)}]\,,$$

thus, updating $\boldsymbol{\mu}^{(k,i)}$ to $\boldsymbol{\mu}^{(k,i+1)}$ can be carried out by solving the following one-variable subproblem

$$\begin{array}{ll} \underset{d}{\text{minimize}} & f(\boldsymbol{\mu}^{(k,i)} + d\mathbf{e}^{(i)}) \\ \text{subject to} & -\mu_i^{(k)} \leq d \leq \gamma - \sum_{j=1}^{i-1}\mu_j^{(k+1)} - \sum_{j=i}^{t}\mu_j^{(k)} \end{array} \tag{11.15}$$

with $\mathbf{e}^{(i)} = [0, \ldots, 0, 1, 0, \ldots, 0]^\top$ a vector with all entries equal to 0, except for the i-th entry being equal to 1. It is important to note that the objective function of (11.15) is a quadratic function in d, i.e.

$$f(\boldsymbol{\mu}^{(k,i)} + d\mathbf{e}^{(i)}) = \left[\frac{1}{2}\langle \mathbf{S}_i, \mathbf{S}_i\rangle\right] d^2 + \nabla_i f(\boldsymbol{\mu}^{(k,i)})d + \text{constant}\,,$$

where $\nabla_i f$ is the i-th component of the gradient of f, which is given by

$$\nabla_i f(\boldsymbol{\mu}) = a_i + \sum_{j=1}^{t}\mu_j\langle \mathbf{S}_i, \mathbf{S}_j\rangle = a_i + \left\langle \mathbf{S}_i, \sum_{j=1}^{t}\mu_j\mathbf{S}_j\right\rangle. \tag{11.16}$$

It is easy to see that problem (11.15) has as optimal solution $d = 0$ (i.e. no need to update $\mu_i^{(k)}$) if and only if the projected gradient (Lin and Moré, 1999) of f at the i-component $\nabla_i^P f(\boldsymbol{\mu}^{(k,i)})$ equals 0, which is defined as

$$\nabla_i^P f(\boldsymbol{\mu}) = \begin{cases} \min(\nabla_i f(\boldsymbol{\mu}), 0) & \text{, if } \mu_i = 0\,; \\ \max(\nabla_i f(\boldsymbol{\mu}), 0) & \text{, if } \gamma = \sum_{j=1}^{t}\mu_j\,; \\ \nabla_i f(\boldsymbol{\mu}) & \text{, if } 0 < \mu_i < \gamma - \sum_{j=1, j\neq i}^{t}\mu_j\,. \end{cases}$$

When $\nabla_i^P f(\boldsymbol{\mu}^{(k,i)}) \neq 0$, we need to find the optimal solution d of problem (11.15).

If $\langle \mathbf{S}_i, \mathbf{S}_i \rangle > 0$, then the optimal solution is given by

$$d = \min\left(\max\left(-\mu_i^{(k)}, -\frac{\nabla_i^P f(\boldsymbol{\mu}^{(k,i)})}{\langle \mathbf{S}_i, \mathbf{S}_i \rangle}\right), \gamma - \sum_{j=1}^{i-1} \mu_j^{(k+1)} - \sum_{j=i}^{t} \mu_j^{(k)}\right). \tag{11.17}$$

If $\langle \mathbf{S}_i, \mathbf{S}_i \rangle = 0$, then it follows that $\mathbf{S}_i = \mathbf{0}$. From (11.16), if $\mathbf{S}_i = \mathbf{0}$, then $\nabla_i f(\boldsymbol{\mu}^{k,i}) = -l_i \leq 0$. Since d is bounded in an interval, the optimal solution is given by

$$d = \gamma - \sum_{j=1}^{i-1} \mu_j^{(k+1)} - \sum_{j=i}^{t} \mu_j^{(k)} .$$

By setting $1/0 = +\infty$, we can also include this case into (11.17). From (11.16), the computation of $\nabla_i f(\boldsymbol{\mu})$ scales as $O(tn^2)$, which is very expensive when t is large. To reduce this computational burden, we define

$$\mathbf{T} = \sum_{j=1}^{t} \mu_j \mathbf{S}_j ,$$

hence, $\nabla_i f(\boldsymbol{\mu}) = a_i + \langle \mathbf{S}_i, \mathbf{T} \rangle$, which scales as $O(n^2)$. After updating $\mu_i^{(k)}$, we can efficiently keep track of $\mathbf{T}$ by

$$\mathbf{T} \leftarrow \mathbf{T} + d\mathbf{S}_i .$$

Note that this computation takes only $O(n^2)$. Consequently, the complexity of computing $\nabla_i f(\boldsymbol{\mu})$ is reduced to $O(n^2)$ instead of $O(tn^2)$.

After having computed $\boldsymbol{\mu}^{(k+1)}$, problem (11.13) can be simplified as

$$\begin{array}{ll} \underset{\mathbf{V}}{\text{minimize}} & \frac{1}{2}\left\| \mathbf{V} + \sum_{i=1}^{t} \mu_i^{(k+1)} \mathbf{S}_i - \lambda \mathbf{I} \right\|_F^2 \\ \text{subject to} & \mathbf{V} \succcurlyeq 0 . \end{array}$$

This is known as the nearest PSD matrix approximation problem under the Frobenius norm (Higham, 1988). Consequently, it has a closed-form solution

$$\mathbf{V}^{(k+1)} = \mathcal{P}^+ \left(\lambda \mathbf{I} - \sum_{i=1}^{t} \mu_i^{(k+1)} \mathbf{S}_i \right) = \mathcal{P}^+ \left(\lambda \mathbf{I} - \mathbf{T} \right) ,$$

where $\mathcal{P}^+$ denotes the projection onto the cone of PSD matrices. From Eq. (11.9), we can easily compute $\mathbf{W}$, which is also guaranteed to be PSD.

Summarizing, the computational complexity of this algorithm to perform the update from $(\mathbf{V}^{(k)}, \boldsymbol{\mu}^{(k)})$ to $(\mathbf{V}^{(k+1)}, \boldsymbol{\mu}^{(k+1)})$ scales as $O(tn^2 + n^3)$. A pseudocode is given in Algorithm 13. In our implementation, we initialize the values of $\mathbf{V}$ and $\boldsymbol{\mu}$

to be zero. Indeed, $\mathbf{V}$ could be initialized by any PSD matrix. It is straightforward to see that $\boldsymbol{\mu} = \mathbf{0}$ implies that $\mathbf{T} = \mathbf{0}$. Therefore, we can avoid the expensive cost of initializing $\mathbf{T}$, which scales as $O(tn^2)$.

Algorithm 13 A block-coordinate descent algorithm for distance metric learning

Input: $\{(\mathbf{S}_i, l_i)\}_{i=1}^t$, λ, γ
Output: $\mathbf{W} \succcurlyeq 0$

Initialize the values of $\mathbf{V} \leftarrow \mathbf{0}$, $\boldsymbol{\mu} \leftarrow \mathbf{0}$, $\mathbf{T} \leftarrow \mathbf{0}$
while $\mathbf{V}$ and $\boldsymbol{\mu}$ are not optimal **do**
  for $i \leftarrow 1, \ldots, t$ **do** ▷ solving for $\boldsymbol{\mu}$
    $G \leftarrow \langle \mathbf{V} - \lambda \mathbf{I}, \mathbf{S}_i \rangle - l_i + \langle \mathbf{S}_i, \mathbf{T} \rangle$
    $P \leftarrow \begin{cases} \min(G, 0) & \text{, if } \mu_i = 0\,; \\ \max(G, 0) & \text{, if } \gamma = \sum_{j=1}^t \mu_j\,; \\ G & \text{, otherwise}\,. \end{cases}$
    if $P \neq 0$ **then**
      $d \leftarrow \min\Big(\max\big(-\mu_i, -\frac{P}{\langle \mathbf{S}_i, \mathbf{S}_i \rangle}\big), \gamma - \sum_{j=1}^t \mu_j\Big)$
      $\mu_i \leftarrow \mu_i + d$
      $\mathbf{T} \leftarrow \mathbf{T} + d\mathbf{S}_i$
    end if
  end for
  $\mathbf{V} \leftarrow \mathcal{P}^+(\lambda \mathbf{I} - \mathbf{T})$ ▷ solving for $\mathbf{V}$
end while
$\mathbf{W} \leftarrow \mathbf{V} + \mathbf{T} - \lambda \mathbf{I}$

11.4.3. Learning a Mahalanobis distance metric for large-scale problems

The above formulation learns a full matrix $\mathbf{W}$ of size $n \times n$, which poses a huge challenge in terms of computational as well as space complexity. When the number of examples is high, it quickly becomes intractable. In order to deal with this issue, one can simplify the distance metric learning problem by considering $\mathbf{W}$ as a diagonal matrix (Nguyen et al., 2016; Schultz and Joachims, 2004; Lajugie et al., 2014). The space complexity is then reduced to $O(n)$. Another advantage is computational simplicity as the positive semidefiniteness constraint is no longer required. Instead, all the elements of $\mathbf{W}$ are required to be nonnegative. Without too much additional effort, we can slightly modify the above algorithm to solve this problem. It turns out that the projection onto the PSD cone amounts to enforcing negative diagonal elements of $\mathbf{W}$ to be zero. The computational complexity of updating all coordinates scales as $O(tn)$. As a consequence of its simplicity, the resulting distance metric of this approach is very restrictive because it neglects the possible correlations between features.

In this subsection, we employ a simple heuristic method to reduce the dimen-

sionality for the kernel implementation. Let the singular value decomposition of the kernel matrix $\mathbf{K}$ be $\mathbf{K} = \mathbf{U}\mathbf{\Sigma}\mathbf{H}^\top$. Kulis (2012) showed that if $\mathbf{W}^*$ is an optimal solution of (11.8), then it admits the following form

$$\mathbf{W}^* = \mathbf{U}\mathbf{W}'\mathbf{U}^\top,$$

where $\mathbf{W}' \succcurlyeq 0$. Since $\mathbf{U}$ is orthogonal, we can rewrite problem (11.8) in terms of $\mathbf{W}'$ as

$$\begin{array}{ll} \underset{\mathbf{W}'}{\text{minimize}} & \frac{1}{2}\|\mathbf{W}'\|_F^2 + \lambda \operatorname{tr}(\mathbf{W}') + \gamma\xi \\ \text{subject to} & \langle \mathbf{W}', \mathbf{U}^\top \mathbf{S}_i \mathbf{U} \rangle \geq l_i - \xi\,, \text{ for } i = 1, \ldots, t\,; \\ & \xi \geq 0\,; \\ & \mathbf{W}' \succcurlyeq 0\,. \end{array} \tag{11.18}$$

This result leads to a simple method to learn $\mathbf{W}$. That is, we first apply a transformation to the input $\mathbf{U}^\top \mathbf{S}_i \mathbf{U}$ and then run Algorithm 13 to find $\mathbf{W}'$. A more detailed discussion of this result can be found in (Jain et al., 2012; Kulis, 2012; Chatpatanasiri et al., 2010). Instead of projecting using all eigenvectors and eigenvalues of the kernel matrix $\mathbf{K}$ to estimate $\mathbf{U}$, here we only use the P leading eigenvectors and eigenvalues. As a consequence, $\mathbf{W}'$ has a size of $P \times P$. In other words, the number of parameters to optimize changes from $O(n^2)$ to $O(P^2)$, making our algorithm more tractable when n is large (up to ten thousand of training examples) and $P \ll n$.

11.5. Experiments

In this section, we present experimental results on both synthetic and real-world data sets, showing the effectiveness of our method against other supervised clustering methods as well as state-of-the-art distance metric learning methods. All the results here are reported in the context of k-means clustering. To account for the sensitivity of k-means clustering, we measure the performance over ten different initial cluster centroid positions and report the result with the lowest squared sum. Next, we describe the experimental settings and give a more detailed overview of our experiments on each data set.

11.5.1. Experimental settings

The following distance metric learning methods are considered:

(1) **Euclidean**: The baseline Euclidean distance metric, which corresponds to the case when the Mahalanobis matrix is the identity matrix.

(2) **MMC**[1] (Xing et al., 2002): This method aims to minimize the sum of squared distances between examples in must-link constraints while keeping the distances between those in cannot-link constraints at least one unit. A simple projected gradient descent was proposed to solve the corresponding optimization problem. Unfortunately, MMC is still limited to small-sized and low-dimensional problems due to the expensive projection onto the cone of PSD matrices.

(3) **ITML**[2] (Davis et al., 2007): This method introduces the use of the LogDet divergence regularization, providing a cheap way to preserve the positive semidefiniteness constraint. For ITML, the slack parameter γ is tuned considering as set of values $\{10^{-3}, \ldots, 10^{3}\}$.

(4) **LMNN**[3] (Weinberger and Saul, 2009): The main aim of this method is to improve the performance of k-nearest-neighbor classification by minimizing distances of one example to its target neighbors and maximizing distances to its impostor neighbors. For LMNN, the hyperparameter μ is tuned considering as set of values $\{0.125, 0.25, 0.5\}$. In our experiment, we set the number of target neighbors to 3.

(5) **LMMLCP**[4] (Lajugie et al., 2014): This supervised clustering method is the most closely related to our work. To get the best results, we tune the hyperparameter C considering as set of values $\{10^{-2}, \ldots, 10^{2}\}$.

(6) **MLCA**[5] (Law et al., 2016): This method provides a closed-form solution for the supervised clustering problem. Although the method is efficient, it is limited to the case where there is only a single set of training data.

(7) **KDMLSC**: In our experiments, we set the parameter P (see Subsection 11.4.3) to be the number of features D. As a result, KDMLSC will learn $D \times D$ parameters as other linear methods, making a fair comparison. The RBF kernel is adopted, $\mathcal{K}(\mathbf{u}, \mathbf{v}) = \exp(-\|\mathbf{u} - \mathbf{v}\|^2/\sigma)$. For KDMLSC, we tune the hyperparamters λ considering as set of values $\{10^{-2}, \ldots, 10^{0}\}$, γ as set of values $\{10^{2}, \ldots, 10^{4}\}$, and σ as set of values $\{10^{-3}, \ldots, 10^{0}\}$. Following Joachims et al. (2009), we set $\epsilon = 0.1$ as a stopping criterion.

In our experiment, we consider all possible pairwise combinations of the training examples for those distance metric learning methods that are based on pairwise constraints (i.e. MMC and ITML). For hyperparameter optimization, a grid search is performed for all possible combinations of values. The same validation strategy is used for all the competing methods. For each set of training data containing m sets of training examples, a validation data set containing m subsets is created. In particular, we randomly subsample 10% from each subset of training data to form

[1] http://www.cs.cmu.edu/~epxing/papers/
[2] http://www.cs.utexas.edu/users/pjain/itml/
[3] https://www.cs.cornell.edu/~kilian/code/code.html
[4] http://www.di.ens.fr/~lajugie/
[5] https://github.com/MarcTLaw/MLCA

the validation set. The best validation score corresponds to the lowest Frobenius loss (Eq. (11.6)) on average. Note that other criteria can also be used to select the best parameters.

As noted by Daumé and Marcu (2005), using a single measure to validate the clustering performance might lead to partial or inexact conclusions. In order to make a fair comparison, results for several clustering measures are reported, including (1) Purity (PUR) (Zhou et al., 2013), which computes the average accuracy of the dominating class correctly assigned in each cluster, (2) Normalized Mutual Information (NMI) (Vinh et al., 2009), which computes the dependence between the predicted and ground-truth clusterings under the independence assumption, and (3) Rand Index (RI) (Rand, 1971), which computes the agreement of the predicted with ground-truth clusterings. For ease of notation, let $\widetilde{\mathcal{C}} = \{\widetilde{\mathcal{C}}_1, \ldots, \widetilde{\mathcal{C}}_{\widetilde{k}}\}$ be a clustering result and $\overline{\mathcal{C}} = \{\overline{\mathcal{C}}_1, \ldots, \overline{\mathcal{C}}_{\overline{k}}\}$ be the original partition of a set $\mathcal{X}$ containing N examples. The purity is computed as

$$\mathrm{PUR}(\widetilde{\mathcal{C}}, \overline{\mathcal{C}}) = \frac{1}{N} \sum_{i=1}^{\widetilde{k}} \max_{j=1,\ldots,\overline{k}} N_{ij}$$

with N_{ij} the number of common examples of $\widetilde{\mathcal{C}}_i$ and $\overline{\mathcal{C}}_j$, which is defined as $N_{ij} = |\widetilde{\mathcal{C}}_i \cap \overline{\mathcal{C}}_j|$, where $1 \leq i \leq \widetilde{k}$ and $1 \leq j \leq \overline{k}$. If the number of clusters is large, it is easy to achieve a high purity value. In particular, when each example gets its own cluster, purity becomes 1. Therefore, purity is not a suitable measure to trade off the quality of the clustering against the number of clusters. To make this trade-off, we can instead use NMI. Let $\widetilde{N}_i$ be the number of examples in $\widetilde{\mathcal{C}}_i$ and $\overline{N}_j$ be the number of examples in $\overline{\mathcal{C}}_j$, then NMI is computed as

$$\mathrm{NMI}(\widetilde{\mathcal{C}}, \overline{\mathcal{C}}) = \frac{\sum_{i=1}^{\widetilde{k}} \sum_{j=1}^{\overline{k}} N_{ij} \log \frac{N_{ij} N}{\widetilde{N}_i \overline{N}_j}}{\sqrt{\left(\sum_{i=1}^{\widetilde{k}} \widetilde{N}_i \log \frac{\widetilde{N}_i}{N}\right)\left(\sum_{j=1}^{\overline{k}} \overline{N}_j \log \frac{\overline{N}_j}{N}\right)}}.$$

Let A denote the number of pairs of examples belonging to the same cluster in $\widetilde{\mathcal{C}}$ and $\overline{\mathcal{C}}$, and let B denote the number of pairs of examples belonging to different clusters in $\widetilde{\mathcal{C}}$ and $\overline{\mathcal{C}}$, then RI is computed as

$$\mathrm{RI}(\widetilde{\mathcal{C}}, \overline{\mathcal{C}}) = \frac{2(A+B)}{N(N-1)}.$$

All measures lie in the range of $[0, 1]$, with 1 representing a perfect clustering. Note that there may exist some inconsistency between different measures due to their discrepancy (Vinh et al., 2010). Additionally, we report the Frobenius loss (FRO) in Eq. (11.6), which is minimized by LMMLCP, MLCA, and our method.

11.5.2. Experiments on a synthetic data set

We conduct an experiment to show the clustering performance of our method on a synthetic data set containing nonlinearly separable clusters. More specifically, the training data consist of two data sets (Figs. 11.1(a) and 11.1(b)), each of which contains 300 examples. Examples in one cluster are generated from a bivariate normal distribution with the same covariance but different mean. All examples of the same class are denoted by the same color and style. The clusters in the test set are generated with the same properties as those of the training set. This data set is challenging for linear distance metric learning methods because clusters of the same class are not linearly separable.

Figure 11.1 shows the clustering results of k-means clustering with $k = 2$ on the test set using the Euclidean distance metric (Fig. 11.1(c)) and our method (Fig. 11.1(d)), where examples of the same color and style are predicted as the same cluster. One can observe that KDMLSC clearly obtains a better clustering performance than the Euclidean baseline. Our resulting distance metric allows to obtain a clustering close to the desired one for the test data. We do not report the results of other linear methods (i.e., MCC, ITML, LMNN, LMMLCP, and MLCA) as they are similar to those of the Euclidean baseline on this synthetic data set.

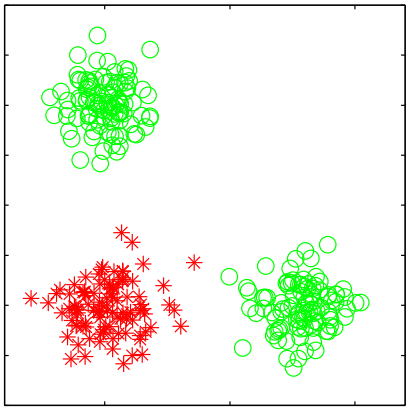
11.5.3. Experiments on handwritten digits data

The USPS digits data set[6] is adopted to create a supervised clustering problem. It contains 9,298 handwritten digit images of size 16×16 pixels. We use the raw pixel representation, i.e., each image is represented by a 256-dimensional feature vector. All examples are normalized to have zero mean and unit variance. Experiments here are performed with a subset of classes (digits 0, 1, 2, 3, 5, 6, and 8). USPS is often used in multiclass classification tasks. To turn it into a supervised clustering task, we use images of digits 2, 5, 6, 8 for training and those of 0, 1, 3 for testing. This data set has also been used as a benchmark for supervised clustering in (Daumé and Marcu, 2005).

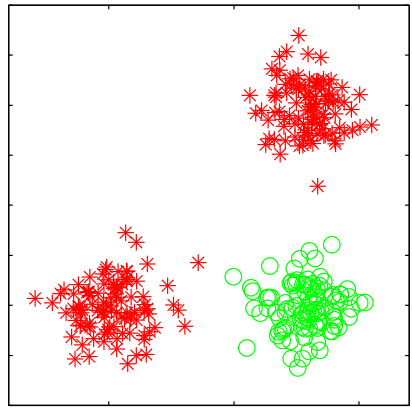
By seeing only images of a few digits in the training set, a supervised clustering method should be able to predict the structure of digits in the test set, even though they have not been seen during training. The results on training as well as test sets are reported in Table 11.1.

We observe that KDMLSC consistently obtains a better performance than other competing methods, only with a slight drop in terms of NMI on the test set. The accuracy (or purity) of the Euclidean baseline on the test set is only 0.87364, while our method achieves 0.94960. The magnitude of the difference shows a substantial

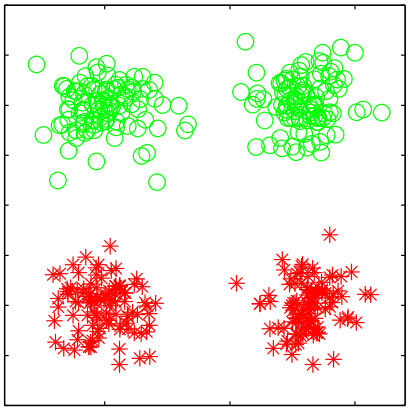
[6] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/



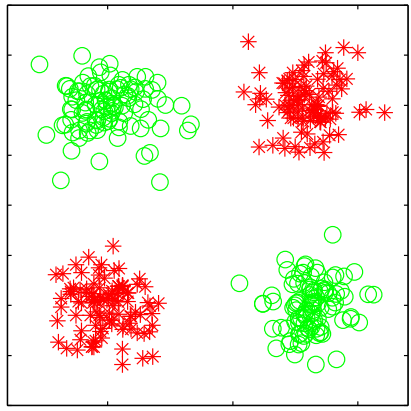
(a) Training data I



(b) Training data II



(c) Euclidean



(d) KDMLSC

Figure 11.1: An illustration of clustering of a nonlinearly separable data set: (a)-(b) training data sets, (c) k-means clustering using the Euclidean distance metric and (d) k-means clustering using the distance metric learned by our method on the test set.

gain. Surprisingly, both LMMLCP and MLCA obtain relatively poor results, even worse than the Euclidean baseline. This can be explained by the fact that the optimization techniques used are not very robust. LMNN obtains similar results as ITML on the test set, but the latter performs significantly better on the training set.

Table 11.1: Performances of the competing methods on the USPS data set based on different measures. The best results are highlighted in boldface.

	Measure	Euclidean	MMC	ITML	LMNN	LMMLCP	MLCA	KDMLSC
Train	RI	0.85629	0.85629	0.96661	0.93437	0.70501	0.71036	**0.98157**
	PUR	0.83434	0.83434	0.96590	0.93341	0.56679	0.55556	**0.98115**
	NMI	0.57997	0.57997	0.87108	0.78167	0.22627	0.31582	**0.92550**
	FRO	2.33078	2.33078	0.53777	0.99219	4.62389	4.37081	**0.30393**
Test	RI	0.86738	0.86738	0.90914	0.91300	0.66596	0.79689	**0.93534**
	PUR	0.87364	0.87364	0.93035	0.92440	0.59643	0.76864	**0.94960**
	NMI	0.75873	0.75873	0.74048	**0.80296**	0.31676	0.63783	0.80097
	FRO	1.11806	1.11806	0.79366	0.80269	2.94509	1.76364	**0.61711**

11.5.4. Experiments on WebKB data

We perform an evaluation on the WebKB data set (Craven et al., 1998), which contains 1,091 web pages retrieved from the computer science departments of the following universities: Cornell, Texas, Washington, and Wisconsin. All web pages are organized according to one of the following topics: faculty page, student page, staff page, department page, research project page, course page, and other. This task has previously been used as a benchmark by Finley and Joachims (2008). Following the setup in (Finley and Joachims, 2008), each web page is represented as a tf-idf vector of dimension 41,131. Due to the high dimensionality, PCA is employed to reduce the dimensionality to 100.

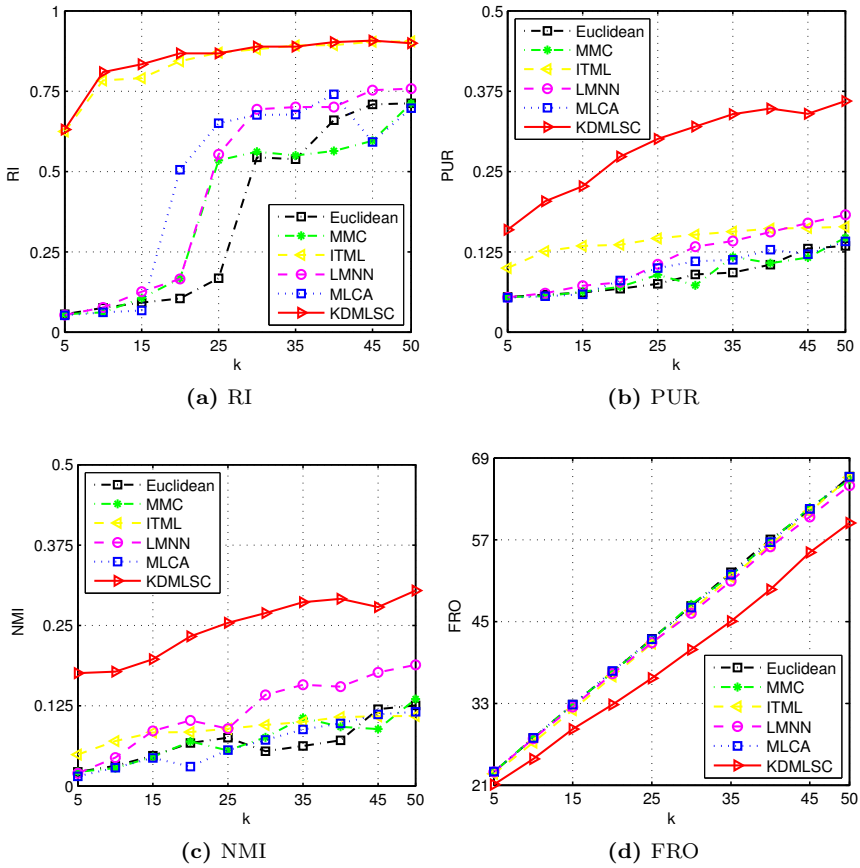
We conduct four leave-one-out experiments in order to evaluate the clustering performance. In each experiment, web pages from one university are considered as a test set and the rest are used for training. The results of all methods are reported in Table 11.2.

Our method yields a good clustering performance and generalization ability. Even though the number of training examples is low, KDMLSC consistently outperforms the Euclidean baseline on all measures, which seems to be inconsistent with MMC, LMMLCP, and MLCA. KDMLSC obtains the lowest values in terms of FRO, except for the case of Cornell University. This is important as the objective of our algorithm is to minimize the Frobenius norm loss during the training phase. Without any metric learning, the clustering performance is only 0.6241 in terms of RI on the case of Wisconsin University, while our method using distance metric learning reaches a rate of 0.7136, thus improving significantly the performance.

Table 11.2: Performances of the competing methods on the WebKB data set based on different measures. The best results are highlighted in boldface.

Method	RI				PUR			
	Cornell	Texas	Washington	Wisconsin	Cornell	Texas	Washington	Wisconsin
Euclidean	0.6978	0.5930	0.6355	0.6241	0.6976	0.5938	0.6917	0.6667
MMC	0.6677	0.6432	0.6297	0.6430	0.6613	0.7148	0.6767	0.7072
ITML	0.7047	0.6619	0.6785	0.6794	0.6976	0.6875	0.6429	0.7259
LMNN	0.7031	0.6683	**0.7336**	0.6892	0.7016	0.7266	**0.7293**	0.6947
LMMLCP	0.6684	0.5873	0.6655	0.6518	0.6129	0.6250	0.6541	0.5981
MLCA	0.6431	0.6131	0.6230	0.6866	0.6129	0.6211	0.5790	0.6355
KDMLSC	**0.7116**	**0.7037**	0.6885	**0.7136**	**0.7137**	**0.7500**	0.6993	**0.7383**
Method	NMI				FRO			
	Cornell	Texas	Washington	Wisconsin	Cornell	Texas	Washington	Wisconsin
Euclidean	0.3039	0.1826	0.3000	0.2857	7.8777	9.2414	8.5047	8.4936
MMC	0.2861	0.3271	0.2655	0.3611	8.2601	7.9442	8.6367	7.9592
ITML	**0.4195**	0.3025	0.2659	0.3521	**7.3427**	8.4864	8.7012	7.9462
LMNN	0.3169	0.3255	**0.3802**	0.3257	7.5877	8.1313	8.0761	8.3339
LMMLCP	0.2285	0.1426	0.2210	0.1664	8.4845	9.1424	8.6441	8.9728
MLCA	0.1622	0.1572	0.1094	0.2428	9.0574	9.1053	9.3684	8.5222
KDMLSC	0.3247	**0.3764**	0.3170	**0.3956**	7.8335	**7.8247**	**8.0556**	**7.5479**

11.5.5. Experiments on text categorization



(a) RI (b) PUR

(c) NMI (d) FRO

Figure 11.2: Performances of the competing methods versus the number of clusters on the 20news data set based on different clustering measures (a)-(d).

The third benchmark is the 20 newsgroups data set (20news), which consists of different UseNet discussion groups collected by Lang (1995) representing diverse areas, including computing, politics, sports, sciences, and so on. This data set has been widely used as a benchmark in machine learning (Weinberger and Saul, 2009; Kumar and Kummamuru, 2008; Zha et al., 2002). We use the bydate version[7], which contains 18,774 documents of 20 groups, with approximately 1,000 documents per group. The 20news data set is divided into two sets, 60% for training and 40% for testing. As noted by Slonim and Tishby (2000), the true clusters are fuzzy due to the fact that some documents are present in more than one group and many of those groups describe similar topics. In our experiments, each document

[7] http://qwone.com/~jason/20Newsgroups/

is represented by a tf-idf vector using the 26,214 most common words in the vocabulary. We then employ PCA to reduce the dimensionality to 100 in order to alleviate the effect of noise and to make other linear methods computationally tractable.

As the class labels of the test data are known, they can be used to analyze the effect of varying the number of clusters. For this purpose, we show the performance of KDMLSC and the competing methods when increasing the number of clusters in Fig. 11.2. Each learning curve represents the performance of the k-means clustering algorithm with different numbers of clusters. Using the code provided by the authors, LMMLCP did not converge within seven days on this large data set, therefore, we did not report its results.

From Fig. 11.2, it is clear that our method achieves a superior performance on different clustering measures. It is important to note that ITML optimizes all possible pairwise constraints derived from the training data, and as a consequence, it obtains a good performance on the RI measure. However, it shows an inconsistent result on the other clustering measures. This can be explained by the fact that ITML treats all pairwise constraints as independent, which is a strong assumption and is not so easily satisfied. Unlike methods based on pairwise constraints, KDMLSC can take advantage of dependencies between pairwise constraints in order to optimize the clustering measures. There is a significant difference in performance between our method and other linear distance metric learning methods. This result confirms the ability of KDMLSC to exploit the cluster structure.

11.5.6. Running time

All the competing methods are implemented in Matlab, running on the same PC. The result of each method is computed with the best hyperparameters. Note that KDMLSC can operate in a high-dimensional input space without reducing the dimensionality, but to allow a fair comparison and faster experimentation, it is trained with the same number of parameters like other methods, which scales quadratically with the number of features. The training times of KDMLSC and the competing methods (MMC, ITML, LMNN, LMMLCP, and MLCA) are reported in Table 11.3.

From Table 11.3, MLCA is the fastest method because it provides a direct analytical solution. However, MLCA cannot achieve a good performance compared to other competing methods. KDMLSC is always the second fastest method and the third fastest one is LMNN. These methods are relatively fast compared to MMC and ITML. This is because the computation time of the latter two methods heavily depends on the number of pairwise constraints, which increases quadratically with the size of the data set. As shown in the table, there is a significant difference in running time between KDMLSC and LMMLCP although both methods are based

Table 11.3: Training time of the competing methods on the data sets used in our experiments (N/A: not available).

Data set	MMC	ITML	LMNN	LMMLCP	MLCA	KDMLSC
USPS	1 h 51 m	1 h 54 m	5 m 15 s	14 h 24 m	0.1 s	2 m 8 s
WebKB	30 m 23 s	3 m 4 s	7 m 35 s	4 m 22 s	0.1 s	1.9 s
20News	6 h 12 m	6 h 51 m	22 m 20 s	N/A	1.8 s	4 m 13 s

on the same framework. Clearly, our method is several orders of magnitude faster. This result demonstrates the efficiency of our dual algorithm in order to learn the distance metric in large-scale settings.

11.6. Conclusion

In this chapter, we have proposed a kernel-based distance metric learning method, namely KDMLSC, for supervised k-means clustering. Our method offers three main advantages over previous methods. First, it learns a distance metric that best fits the nonlinear structure of the data by employing kernel learning. Second, it implicitly reduces the risk of overfitting by incorporating a low-rank constraint on the learned Mahalanobis matrix. Third, our dual algorithm is simple to implement and more scalable to large data sets than most of the semidefinite programming solvers. Although we have only applied this dual algorithm in the context of supervised clustering, it can be used as a general solver for developing new distance metric learning methods that involve semidefinite programming. Experiments across different domains have shown that KDMLSC outperforms the state-of-the-art distance metric learning methods on supervised clustering tasks.

PART V

EPILOGUE

12 Conclusions and future work

A good distance metric mostly depends on the application domain and should yield small distances between similar examples and large distances between dissimilar examples. Recent advances in distance metric learning have demonstrated a promising approach to compute more effective distance metrics for a given problem. However, they either lack scalability, robustness, ability to handle different learning settings and data types, or have no theoretical guarantee on convergence. The methods proposed in this thesis are motivated by an attempt to overcome these shortcomings. Below, we will summarize the main conclusions that can be drawn from this thesis and highlight some pending issues that might be interesting for future research.

12.1. Conclusions and open issues

In this thesis, we addressed some important limitations of existing distance metric learning methods by introducing novel methods for different supervised settings. The applicability of the proposed methods was demonstrated on many synthetic as well as real-world data sets. These methods were implemented in Matlab and the resulting toolboxes were made publicly available[1], so that any non-expert user can easily use or test the developed methods on their own data.

In Chapter 3, we introduced the DMLMJ method that aims at learning a linear transformation through maximization of the Jeffrey divergence between two multivariate Gaussian distributions derived from local pairwise constraints. Learning the linear transformation was formulated as an unconstrained optimization problem, which can be solved analytically. In addition, a kernelized version of DMLMJ was derived to tackle nonlinear problems.

In Chapter 4, we proposed the use of kernels for the KISSME method, allowing to capture the nonlinear structure in the data set. This method operates in the kernel spaces, yielding a highly flexible distance metric. We also presented an incremental update strategy for k-KISSME upon the arrival of a new pairwise constraint, which could be computationally expensive. Despite the promising results, there are still some aspects of k-KISSME and its incremental version that require further efforts. For instance, the computational bottleneck of k-KISSME becomes impractical on large-scale data sets. The latter is endemic to most kernel-based methods and reducing the training set size may be useful in this case. While our incremental update strategy for k-KISSME may be initially sufficient, it would

[1] https://github.com/bacnguyencong

be more interesting to be able to keep the Mahalanobis matrix within the cone of PSD matrices and to perform an update upon arrival of multiple constraints at the same time.

In Chapter 6, we employed the principle of margin maximization to learn the distance metric with the goal of improving the performance of k-NN classification. To make our method scalable on large-data sets, an efficient online algorithm based on SGD, namely LMDML-A, was developed. Our algorithm keeps the solution always within the PSD cone by computing an appropriate step size in each iteration. We use the Schur complement to find an upper bound of the step size that guarantees that the solution remains within the PSD cone. It would be interesting to extend LMDML-A to make it applicable for very high-dimensional data sets, which would be computationally attractive. One may improve the training speed of LMDML-A by considering the possibility of implementing it in parallel and averaging the resulting solutions.

In Chapter 7, we proposed the DML-dc method to minimize the misclassification rate of the nearest-neighbor classifier. Due to the use of the ramp loss function, our objective function for margin maximization has a strong ability to avoid the influence of outliers. Since the objective function can be decomposed into a DC program, we iteratively solved a sequence of convex subproblems using DCA. To further reduce the computational cost of DCA, it would be interesting to explore more advanced optimization techniques (e.g. stochastic gradient) to reduce the computational complexity of solving the convex subproblems.

In Chapter 8, we proposed the CMML method that learned multiple local distance metrics instead of a single global one in order to tackle heterogeneously distributed data. First, data were divided into several clusters using k-means clustering, then a single distance metric was estimated for each cluster based on triplet constraints. Moreover, a global distance metric was introduced to capture the common structure among all the clusters, which required that the distance metric in each cluster should be as close as possible to the global one. To make CMML scalable for large data sets, the block-coordinate descent algorithm was adopted, which enabled us to solve the optimization problem efficiently. We used k-means clustering to partition the input data due to its simplicity and efficiency, but future studies employing other clustering algorithms could be interesting. One may extend CMML by incorporating the clustering procedure into the training process, which could optimally partition the input data.

In Chapter 9, we presented the LDMLR method for k-NN regression. Instead of randomly selecting triplet constraints to satisfy an application-specific criterion, we extracted the constraints from the local neighborhood of each training example, which allowed us to preserve discriminative information from this neighborhood. In order to solve this problem, a special solver based on coordinate gradient descent method was developed.

In Chapter 10, we developed the ODML method for ordinal classification by incorporating local triplet constraints containing the ordering information into a conventional large-margin distance metric learning method. Compared to previous methods, our method did not make any assumption about the absolute distances between the class labels, making it more robust and suitable for ordinal classification tasks.

In Chapter 11, we introduced the KDMLSC method to improve the practical use of k-means clustering. A common assumption of this method is that the available training examples share the same distance metric as that of test examples, which was then used by a distance-based clustering algorithm. In order to reduce the training time, we derived a meaningful Lagrange dual formulation and introduced an efficient algorithm based on block-coordinate descent. Our current implementation using internal cross-validation to select an appropriate kernel function may limit the expressiveness of the resulting KDMLSC method. While kernel-based methods use a single kernel function, in practice it is often more effective to use multiple kernel functions as they can naturally handle multiple data sources. It will be interesting to explore the use of multiple predefined kernel functions to overcome this limitation. Another extension would be to generalize our model to the case where the number of clusters is unknown. In this situation, one may estimate the number of clusters directly from the data by penalizing the objective function of k-means (Lajugie et al., 2014).

12.2. Potential research directions

Several directions have been discussed in the preceding section to improve our methods as well as to establish a more general framework for distance metric learning. In addition, we will explore some potential research directions as follows.

12.2.1. Distance metric learning for extreme classification

The complexity of a machine learning problem is often characterized in terms of the number of examples and the number of features. Recent studies (Deng et al., 2011; Gupta et al., 2014) have shown that increasing the number of labels (classes) also poses a huge challenge. Such challenge appears in many applications, e.g., text classification, ranking, tagging locations, photo and video annotation, where the goal is to learn a predictor that automatically tags a data point with the most relevant labels from an extremely large label set (up to several millions). Problems of this kind are referred to as extreme classification problems. They are mainly of two kinds: single-label classification where each example is assigned only one label and multi-label classification where each example can be associated with many labels. The latter case usually contains a very few labels per example. Consider,

for instance, a problem of tagging Wikipedia articles, where one might wish to tag a document with a couple of relevant labels, which are chosen from a set of more than a million possible tags. Extreme classification has also opened the door to ranking and recommendation systems by reformulating them as multi-label classification problems, where each item to be ranked/recommended is treated as a separate label (Agrawal et al., 2013).

A naive one-versus-all approach is to independently learn a single model for each label. Since the space and time complexities are linear in the number of labels, this approach quickly becomes intractable both for training and prediction. An important characteristic of extreme classification problems is that a large fraction of labels have very few training examples assigned to them, referred to as long-tail label distributions. Unlike conventional classification approaches, distance metric learning becomes a very appealing technique in extreme classification because of its ability to learn the general concept of distance metric (differently from label-specific concepts) and its compatibility with an efficient nearest neighbor search on the learned metric space.

12.2.2. Deep metric learning

With the remarkable success in learning useful semantic representations of data, so-called deep metric learning aims at learning an embedding function through a deep neural network (e.g., a convolutional neural network), which directly optimizes a loss function related to the similarity of examples (Hu et al., 2014; Song et al., 2016, 2017; Sohn, 2016; Wang et al., 2017; Duan et al., 2017). Specifically, embeddings are optimized to pull similar examples close to each other and push dissimilar examples far apart from each other. For instance, Hu et al. (2014) proposed a two-layer discriminative network, which learns a set of nonlinear transformations that make the distances between examples of must-link pairs smaller than a threshold and the distances between those of cannot-link pairs larger than a threshold. Sohn (2016) introduced the multi-class N-pairs loss, which extends the triplet loss by allowing the joint comparison between an example and multiple examples of different classes. By learning from the general concept of similarity instead of category-specific concepts, deep metric learning can naturally deal with problems involving millions of labels, which could be impossible for conventional deep neural networks due to the computational bottleneck (unless advanced techniques, such as hierarchical softmax and negative sampling are employed).

A central issue for deep metric learning consists in collecting and creating meaningful training constraints (e.g., pairwise, triplet, or quadruplet constraints). In this thesis, we have shown that an efficient strategy of building constraints is very critical to the success of the algorithms. This is also indicated in recent literature (Ge, 2018). The usual solution to minimize the loss in an online fashion

with stochastic gradient descent is to randomly sample constraints. Unfortunately, the number of violated constraints decreases if there are more labels (Gupta et al., 2014), leading to slow convergence and low performance. To make learning more effective and efficient, only difficult (or hard) constraints should be considered. However, selecting hard constraints could be a very expensive operation. Designing a more effective sampling strategy to avoid this issue would be a promising direction.

Several loss functions have been developed for deep metric learning, such as contrastive loss, triplet loss, quadruplet loss, and N-pair loss. Compared to softmax loss, these loss functions are more difficult to optimize. Existing loss functions often employ only one negative example while not interacting with the other negative labels. Therefore, a potential research direction is to develop new loss functions with multiple negative examples. In particular, the loss function should take into account the global structure of the embedding space.

12.2.3. Theoretical understanding

Similarly to conventional supervised learning, one may prove generalization bounds by considering each pairwise or triplet constraint as an independent and identically distributed (i.i.d.) sample. However, the i.i.d. assumption is violated in distance metric learning because the constraints are built from the training set. That is why obtaining generalization bounds in distance metric learning is very challenging. Although several recent works have attempted to prove a generalization bound, analyzing the link between the consistency of the learned distance metric and its performance in a given algorithm (classifier, ranking, regression, etc) remains an important open problem. So far, only a few results for linear classification have been obtained using the notion of uniform stability (Jin et al., 2009), algorithmic robustness (Bellet and Habrard, 2015), and Rademacher complexity (Guo and Ying, 2014). As mentioned in Bellet et al. (2015), there is still no theoretical result for k-NN classification using the learned distance metric.

Appendices

A Appendix

A.1. Jeffrey divergence

Let P_1 and P_2 be two D-dimensional multivariate Gaussian distributions with means $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, covariance matrices $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$, and corresponding probability density functions p_1 and p_2, respectively. The Kullback-Leibler divergence between P_1 and P_2 is defined as:

$$\begin{aligned}\mathrm{KL}(P_1, P_2) &= \int \ln\left(\frac{p_1(\mathbf{x})}{p_2(\mathbf{x})}\right) p_1(\mathbf{x})\,\mathrm{d}\mathbf{x} \\ &= \frac{1}{2}\left[\log\left(\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|}\right) - D + \mathrm{tr}\left(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right) + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)\right].\end{aligned}$$

The proof of the latter expression can be found in (Davis and Dhillon, 2007). We now consider the symmetric Kullback-Leibler divergence or Jeffrey divergence between P_1 and P_2:

$$\begin{aligned}\mathrm{JF}(P_1, P_2) &= \mathrm{KL}(P_1, P_2) + \mathrm{KL}(P_2, P_1) \\ &= \frac{1}{2}\left[\log\left(\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|}\right) - D + \mathrm{tr}\left(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right) + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)\right] \\ &\quad + \frac{1}{2}\left[\log\left(\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_2|}\right) - D + \mathrm{tr}\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_2\right) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right] \\ &= \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_2 + \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right) + \frac{1}{2}\mathrm{tr}\left((\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)\right) \\ &\quad + \frac{1}{2}\mathrm{tr}\left((\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right) - D \\ &= \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_2 + \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right) - D + \frac{1}{2}\mathrm{tr}\left((\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1})(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^\top\right).\end{aligned}$$

If $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \mathbf{0}$, then the Jeffrey divergence between P_1 and P_2 reduces to:

$$\mathrm{JF}(P_1, P_2) = \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_2 + \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right) - D\,.$$

A.2. Conditions to guarantee the convergence of block-coordinate descent

Assume that the objective function to be optimized has the following form

$$f(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}) = f_0(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}) + \sum_{k=1}^{n} f_k(\mathbf{x}^{(k)})$$

for some $f_0 \colon \mathbb{R}^{N^{(1)}+\cdots+N^{(n)}} \to \mathbb{R} \cup \{\infty\}$ and some $f_k \colon \mathbb{R}^{N^{(k)}} \to \mathbb{R} \cup \{\infty\}$, $k = 1, \ldots, n$. We refer to each $\mathbf{x}^{(k)}$, $k = 1, \ldots, n$, as a coordinate block of $\mathbf{x} = (x_1, \ldots, x_{N^{(k)}})$. In order to guarantee the convergence of block-coordinate descent, the following conditions are proposed by Tseng (2001):

(B1) f_0 is continuous on $\mathrm{dom} f_0$.

(B2) For each $k \in \{1, \ldots, n\}$ and $\mathbf{x}^{(j)}, j \neq k$, the function $\mathbf{x}^{(k)} \mapsto f(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)})$ is quasiconvex and hemivariate.

(B3) $f_0, f_1, \ldots, f_n$ are lower semicontinuous.

(C1) $\mathrm{dom} f_0$ is open and f_0 tends to ∞ at every boundary point of $\mathrm{dom} f_0$.

(C2) $\mathrm{dom} f_0 = \mathcal{Y}^{(1)} \times \cdots \times \mathcal{Y}^{(n)}$, for some $\mathcal{Y}^{(k)} \subseteq \mathbb{R}^{N^{(k)}}$, $k = 1, \ldots, n$.

Proposition A.1 ((Tseng, 2001))**.** *Suppose that $f, f_0, \ldots, f_n$ satisfy assumptions (B1)-(B3) and that f_0 satisfies either assumption (C1) or (C2). Using the essentially cyclic rule, the block-coordinate descent method converges to an optimal point of f.*

A.3. Data sets

Data sets from the *Knowledge Extraction based on Evolutionary Learning* (KEEL) machine learning repository[1]. These data sets cover a range from 4 to 90 features with the number of examples varying from 106 to 20,000. A brief description of these data sets is given in Table A.1.

Table A.1: A brief description of the KEEL data sets

Id	Data sets	Features	Examples	Classes	Id	Data sets	Features	Examples	Classes
APP	*appendicitis*	7	106	2	PIM	*pima*	8	768	2
BAN	*balance*	4	625	3	RIN	*ring*	20	7400	2
BUP	*banana*	2	5300	2	SAT	*satimage*	36	6435	7
BUP	*bupa*	6	345	2	SEG	*segment*	19	2310	7
ION	*ionosphere*	33	351	2	SON	*sonar*	60	208	2
IRI	*iris*	4	150	3	SPA	*spambase*	57	4597	2
LED	*led7digit*	7	500	10	TEX	*texture*	40	5500	11
LET	*letter*	16	20000	26	TWO	*twonorm*	20	7400	2
MAG	*magic*	10	19020	2	VEH	*vehicle*	18	846	4
MON	*monk-2*	6	432	2	VOW	*vowel*	13	990	11
MOV	*movement_libras*	90	360	15	WDB	*wdbc*	30	569	2
OPT	*optdigits*	64	5620	10	WIN	*wine*	13	178	3
PAG	*page-blocks*	10	5472	5	WIS	*wisconsin*	9	683	2
PHO	*phoneme*	5	5404	2					

[1] KEEL: http://sci2s.ugr.es/keel/datasets.php

Bibliography

Abbasnejad, M. E., Ramachandram, D., and Mandava, R. (2012). A survey of the state of the art in learning the kernels. *Knowledge and Information Systems*, 31(2):193–221.

Aghaeepour, N., Nikolic, R., Hoos, H. H., and Brinkman, R. R. (2011). Rapid cell population identification in flow cytometry data. *Cytometry A*, 79 A:6–13.

Agrawal, R., Gupta, A., Prabhu, Y., and Varma, M. (2013). Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 13–24.

Agresti, A. (2010). *Analysis of Ordinal Categorical Data*. John Wiley & Sons.

Ahmed, E., Jones, M., and Marks, T. K. (2015). An improved deep learning architecture for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3908–3916.

Ahonen, T., Hadid, A., and Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041.

Aloise, D., Deshpande, A., Hansen, P., and Popat, P. (2009). Np-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248.

Amir, E., Davis, K. L., Tadmor, M. D., Simonds, E. F., Levine, J. H., Bendall, S. C., Shenfeld, D. K., Krishnaswamy, S., Nolan, G. P., and Pe'er, D. (2013). viSNE enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia. *Nat Biotechnol*, 31:545–552.

Anthony, M. and Bartlett, P. L. (2009). *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1st edition.

Assi, K. C., Labelle, H., and Cheriet, F. (2014). Modified large margin nearest neighbor metric learning for regression. *IEEE Signal Processing Letters*, 21(3):292–296.

Atzmon, Y., Shalit, U., and Chechik, G. (2015). Learning sparse metrics, one feature at a time. *The Journal of Machine Learning Research*, 1:1–48.

Baccianella, S., Esuli, A., and Sebastiani, F. (2009). Evaluation measures for ordinal regression. In *Proceedings of the 9th International Conference on Intelligent Systems Design and Applications*, pages 283–287.

Bach, F. R. and Jordan, M. I. (2003). Learning spectral clustering. In *Advances in Neural Information Processing Systems 16*, pages 305–312.

Baghshah, M. S. and Shouraki, S. B. (2010a). Kernel-based metric learning for semi-supervised clustering. *Neurocomputing*, 73(7-9):1352–1361.

Baghshah, M. S. and Shouraki, S. B. (2010b). Non-linear metric learning using pairwise similarity and dissimilarity constraints and the geometrical structure of data. *Pattern Recognition*, 43(8):2982–2992.

Bagirov, A. M., Taheri, S., and Ugon, J. (2016). Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems. *Pattern Recognition*, 53:12–24.

Baltieri, D., Vezzani, R., and Cucchiara, R. (2011). 3dpes: 3d people dataset for surveillance and forensics. In *Proceedings of the Joint ACM Workshop on Human Gesture and Behavior Understanding*, pages 59–64.

Bar-Hillel, A., Hertz, T., Shental, N., and Weinshall, D. (2005). Learning a Mahalanobis metric from equivalence constraints. *The Journal of Machine Learning Research*, 6(6):937–965.

Bartlett, P. L. (1998). The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536.

Basu, S., Bilenko, M., and Mooney, R. J. (2004). A probabilistic framework for semi-supervised clustering. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 59–68.

Bedagkar-Gala, A. and Shah, S. K. (2014). A survey of approaches and trends in person re-identification. *Image and Vision Computing*, 32(4):270–286.

Beernaerts, J., Derie, R., Nguyen, B., Vansteenkiste, P., De Baets, B., Deconinck, F., Lenoir, M., De Clercq, D., and Van de Weghe, N. (2019). Assessing the potential of the qualitative trajectory calculus to detect gait pathologies: a case study of children with developmental coordination disorder. *Computer Methods in Biomechanics and Biomedical Engineering*, accepted.

Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps and spectral techniques for embedding and clustering. *Neural Computation*, 15(6):1373–1396.

Bellet, A. and Habrard, A. (2015). Robustness and generalization for metric learning. *Neurocomputing*, 151:259–267.

Bellet, A., Habrard, A., and Sebban, M. (2011). Learning good edit similarities with generalization guarantees. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 188–203.

Bellet, A., Habrard, A., and Sebban, M. (2012). Good edit similarity learning by loss minimization. *Machine Learning*, 89:5–35.

Bellet, A., Habrard, A., and Sebban, M. (2015). Metric learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 9(1):1–151.

Ben-David, S., Eiron, N., and Long, P. M. (2003). On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66:496 – 514.

Benavoli, A., Corani, G., Demšar, J., and Zaffalon, M. (2017). Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18:2653–2688.

Benavoli, A., Corani, G., and Mangili, F. (2016). Should we really use post-hoc tests based on mean-ranks? *The Journal of Machine Learning Research*, 17(1):152–161.

Benedetti, J. K. (1977). On the nonparametric estimation of regression functions. *Journal of the Royal Statistical Society*, 39(2):248–253.

Bertsekas, D. P. (1999). *Nonlinear programming*. Athena Scientific.

Beygelzimer, A., Kakade, S., and Langford, J. (2006). Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 97–104.

Bigi, B. (2003). Using Kullback-Leibler distance for text categorization. In *Proceedings of the 25th European Conference on IR Research*, pages 305–319.

Bigi, B., Mori, R. D., El-Bèze, M., and Spriet, T. (2000). A fuzzy decision strategy for topic identification and dynamic selection of language models. *Signal Processing*, 80(6):1085–1097.

Bilenko, M., Basu, S., and Mooney, R. J. (2004). Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the 21st International Conference on Machine Learning*, pages 81–88.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag, Berlin, Heidelberg.

Boddy, L., Morris, C., Wilkins, M., Al-Haddad, L., Tarran, G., Jonker, R., and Burkill, P. (2000). Identification of 72 phytoplankton species by radial basis function neural network analysis of flow cytometric data. *Mar Ecol Prog Ser*, 195:47–59.

Bohné, J., Ying, Y., Gentric, S., and Pontil, M. (2014). Large margin local metric learning. In *Proceedings of the European Conference on Computer Vision*, pages 679–694.

Bottou, L. (1991). Stochastic gradient learning in neural networks. In *Proceedings of Neuro-Nîmes 91*.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Bregman, L. (1967). The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217.

Brinkman, R. R., Aghaeepour, N., Finak, G., Gottardo, R., Mosmann, T., and Scheuermann, R. H. (2016). Automated analysis of flow cytometry data comes of age. *Cytometry A*, 89:13–15.

Campbell, S. L. and Meyer, C. D. (1979). *Generalized Inverses of Linear Transformations*. SIAM.

Cauchy, A. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Compte Rendu à l'Académie des Sciences*, 25:536–538.

Censor, Y. and Zenios, S. A. (1997). *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press.

Chang, C.-C. (2010). Generalized iterative RELIEF for supervised distance metric learning. *Pattern Recognition*, 43(8):2971–2981.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Chang, K.-W., Hsieh, C.-J., and Lin, C.-J. (2008). Coordinate descent method for large-scale ℓ_2-loss linear support vector machines. *The Journal of Machine Learning Research*, 9:1369–1398.

Chang, K. Y., Chen, C. S., and Hung, Y. P. (2011). Ordinal hyperplanes ranker with cost sensitivities for age estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 585–592.

Chapelle, O., Do, C. B., Teo, C. H., Le, Q. V., and Smola, A. J. (2009). Tighter bounds for structured estimation. In *Advances in Neural Information Processing Systems 21*, pages 281–288.

Chatpatanasiri, R., Korsrilabutr, T., Tangchanachaianan, P., and Kijsirikul, B. (2010). A new kernelization framework for mahalanobis distance learning algorithms. *Neurocomputing*, 73:1570–1579.

Chechik, G., Sharma, V., Shalit, U., and Bengio, S. (2010). Large scale online learning of image similarity through ranking. *The Journal of Machine Learning Research*, 11:1109–1135.

Chen, J., Zhang, Z., and Wang, Y. (2015). Relevance metric learning for person re-identification by exploiting listwise similarities. *IEEE Transactions on Image Processing*, 24(12):4741–4755.

Chen, S., Guo, C., and Lai, J. (2016). Deep ranking for person re-identification via joint representation learning. *IEEE Transactions on Image Processing*, 25:2353–2367.

Cheng, D., Gong, Y., Zhou, S., Wang, J., and Zheng, N. (2016). Person re-identification by multi-channel parts-based cnn with improved triplet loss function. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1335–1344.

Cheng, D. S., Cristani, M., Stoppa, M., Bazzani, L., and Murino, V. (2011). Custom pictorial structures for re-identification. In *Proceedings of the British Machine Vision Conference*, pages 68.1–68.11.

Chu, W. and Ghahramani, Z. (2005). Gaussian processes for ordinal regression. *The Journal of Machine Learning Research*, 6:1019–1041.

Chu, W. and Keerthi, S. S. (2005). New approaches to support vector ordinal regression. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 145–152.

Cinbis, R. G., Verbeek, J., and Schmid, C. (2011). Unsupervised metric learning for face identification in TV video. In *Proceedings of the International Conference on Computer Vision*, pages 1559–1566.

Cole, R. and Fanty, M. (1990). Spoken letter recognition. In *Proceedings of the Third DARPA Speech and Natural Language Workshop*, pages 385–390.

Collins, M. and Duffy, N. (2002). Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, pages 625–632.

Collobert, R., Sinz, F., Weston, J., and Bottou, L. (2006a). Large scale transductive SVMs. *The Journal of Machine Learning Research*, 7:1687–1712.

Collobert, R., Sinz, F., Weston, J., and Bottou, L. (2006b). Trading convexity for scalability. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 201–208.

Cong, Y., Liu, J., Yuan, J., and Luo, J. (2014). *Low-Rank and Sparse Modeling for Visual Analysis*, chapter Low-Rank Online Metric Learning, pages 203–233. Springer International Publishing.

Cortes, C. and Mohri, M. (2004). AUC optimization versus error rate minimization. In *Advances in Neural Information Processing Systems 16*, pages 313–320.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Cover, T. (1968). Estimation by the nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(1):50–55.

Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Crammer, K., Gilad-Bachrach, R., Navot, A., and Tishby, N. (2003). Margin analysis of the LVQ algorithm. In *Advances in Neural Information Processing Systems 15*, pages 479–486.

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., and Slattery, S. (1998). Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, pages 509–516.

Cruz-Ramírez, M., Hervás-Martínez, C., Sánchez-Monedero, J., and Gutiérrez, P. A. (2014). Metrics to guide a multi-objective evolutionary algorithm for ordinal classification. *Neurocomputing*, 135:21–31.

Cucker, F. and Smale, S. (2002). On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39(1):1–49.

Dahl, J. and Vandenberghe, L. (2004). Cvxopt python software for convex optimization.

Daumé, H. and Marcu, D. (2005). A Bayesian model for supervised clustering with the Dirichlet process prior. *The Journal of Machine Learning Research*, 6:1551–1577.

Davis, J. V. and Dhillon, I. S. (2007). Differential entropic clustering of multivariate Gaussians. In *Advances in Neural Information Processing Systems 19*, pages 337–344. MIT Press.

Davis, J. V., Kulis, B., Jain, P., Sra, S., and Dhillon, I. S. (2007). Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning*, pages 209–216.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30.

Deng, J., Satheesh, S., Berg, A. C., and Li, F. (2011). Fast and balanced: Efficient label tree learning for large scale object recognition. In *Advances in Neural Information Processing Systems 24*, pages 567–575.

Deza, M.-M. and Deza, E. (2006). *Dictionary of distances*. Elsevier.

Dikmen, M., Akbas, E., Huang, T. S., and Ahuja, N. (2011). Pedestrian recognition with a learned metric. In *Proceedings of the Asian Conference on Computer Vision*, pages 501–512.

Ding, S., Lin, L., Wang, G., and Chao, H. (2015). Deep feature learning with relative distance comparison for person re-identification. *Pattern Recognition*, 48(10):2993–3003.

Domeniconi, C., Peng, J., and Gunopulos, D. (2001). An adaptive metric machine for pattern classification. In *Advances in Neural Information Processing Systems 13*, pages 458–464.

Donoho, D. L. and Elad, M. (2003). Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. In *Proceedings of the National Academy of Sciences*, pages 2197–2202.

Duan, Y., Lu, J., Feng, J., and Zhou, J. (2017). Deep localized metric learning. *IEEE Transactions on Circuits and Systems for Video Technology*, To appear:In press.

Duda, R. O., Hart, P. E., and Stork, D. G. (2012). *Pattern Classification*. Wiley-Interscience, 2nd edition.

Dunn, O. J. (1961). Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64.

Elkan, C. (2003a). Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th International Conference on International Conference on Machine Learning*, pages 147–153.

Elkan, C. (2003b). Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th International Conference on Machine Learning*, pages 147–153.

Ertekin, S., Bottou, L., and Giles, C. L. (2011). Nonconvex online support vector machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2):368–381.

Evgeniou, T. and Pontil, M. (2004). Regularized multi–task learning. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117.

Faraki, M., Harandi, M. T., and Porikli, F. (2018). Large-scale metric learning: A voyage from shallow to deep. *IEEE Transactions on Neural Networks and Learning Systems*, 29(9):4339–4346.

Finak, G., Perez, J. M., Weng, A., and Gottardo, R. (2010). Optimizing transformations for automated, high throughput analysis of flow cytometry data. *BMC Bioinformatics*, 11:546.

Finley, T. and Joachims, T. (2005). Supervised clustering with support vector machines. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 217–224.

Finley, T. and Joachims, T. (2008). Supervised k-means clustering. Technical report, Department of Computer Science, Cornell University, Ithaca, NY, USA.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188.

Fisher, R. A. (1959). *Statistical Methods and Scientific Inference*. Oliver and Boyd, Edinburgh, second edition.

Fouad, S. and Tino, P. (2013). Ordinal-based metric learning for learning using privileged information. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1–8.

Frank, A. and Asuncion, A. (2010). UCI machine learning repository.

Frank, E. and Hall, M. (2001). A simple approach to ordinal classification. In *Proceedings of the 12th European Conference on Machine Learning*, pages 145–156. Springer Berlin Heidelberg.

Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110.

François, D. (2008). *High-dimensional Data Analysis: From Optimal Metrics to Feature Selection*. PhD thesis, Universit´e catholique de Louvain.

Fréchet, M. M. (1906). Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 22:1–72.

Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332.

Friedman, J. H. (1989). Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92.

Frome, A., Singer, Y., and Malik, J. (2007a). Image retrieval and classification using local distance functions. In *Advances in Neural Information Processing Systems 19*, pages 417–424.

Frome, A., Singer, Y., Sha, F., and Malik, J. (2007b). Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *Proceedings of IEEE International Conference on Computer Vision*, pages 1–8. IEEE.

Fujiwara, S., Takeda, A., and Kanamori, T. (2017). DC algorithm for extended robust support vector machine. *Neural Computation*, 29(5):1406–1438.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*. Academic Press, second edition.

Fullerton, A. S. and Xu, J. (2012). The proportional odds with partial proportionality constraints model for ordinal response variables. *Social Science Research*, 41(1):182–198.

Gallier, J. (2010). The schur complement and symmetric positive semidefinite (and definite) matrices. Technical report, Penn Engineering.

Gao, X., Hoi, S. C. H., Zhang, Y., Wan, J., and Li, J. (2014). SOML: Sparse online metric learning with application to image retrieval. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1206–1212.

García, S. and Herrera, F. (2008). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *The Journal of Machine Learning Research*, 9:2677–2694.

Gärtner, T. (2003). A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5:49–58.

Gärtner, T., Flach, P., and Wrobel, S. (2003). On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop*, pages 129–143.

Ge, W. (2018). Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision*, pages 269–285.

Ge, Y. and Sealfon, S. C. (2012). Flowpeaks: A fast unsupervised clustering for flow cytometry data via k-means and density peak finding. *Bioinformatics*, 28:2052–2058.

Gilad-Bachrach, R., Navot, A., and Tishby, N. (2004). Margin based feature selection - theory and algorithms. In *Proceedings of the 21st International Conference on Machine Learning*, pages 43–50.

Globerson, A. and Roweis, S. T. (2006). Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems 18*, pages 451–458.

Goldberger, J., Roweis, S., Hinton, G., and Salakhutdinov, R. (2005). Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 17*, pages 513–520.

Goldstein, A. A. (1964). Convex programming in hilbert space. *Bulletin of the American Mathematical Society*, 70(5):709–710.

Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations*. Johns Hopkins University Press, 3rd edition.

Gönen, M. and Heller, G. (2005). Concordance probability and discriminatory power in proportional hazards regression. *Biometrika*, 92(4):965–970.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Grant, M. and Boyd, S. (2014). CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx.

Gray, D., Brennan, S., and Tao, H. (2007). Evaluating appearance models for recognition, reacquisition, and tracking. In *Proceedings of the IEEE International Workshop on Performance Evaluation for Tracking and Surveillance*, pages 41–48.

Gray, D. and Tao, H. (2008). Viewpoint invariant pedestrian recognition with an ensemble of localized features. In *Proceedings of the European Conference on Computer Vision*, pages 262–275.

Gu, Q. and Han, J. (2013). Clustered support vector machines. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*, pages 307–315.

Guillaumin, M., Verbeek, J., and Schmid, C. (2009). Is that you? Metric learning approaches for face identification. In *Proceedings of the 12-th International Conference on Computer Vision*, pages 498–505.

Guo, Z. and Ying, Y. (2014). Guaranteed classification via regularized similarity learning. *Neural Computation*, 26(3):497–522.

Gupta, M. R., Bengio, S., and Weston, J. (2014). Training highly multiclass classifiers. *The Journal of Machine Learning Research*, 15:1461–1492.

Gutiérrez, P. A. and García, S. (2016). Current prospects on ordinal and monotonic classification. *Progress in Artificial Intelligence*, 5(3):171–179.

Gutiérrez, P. A., Pérez-Ortiz, M., Sánchez-Monedero, J., Fernández-Navarro, F., and Hervás-Martínez, C. (2016). Ordinal regression methods: Survey and experimental study. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):127–146.

Hao, S., Zhao, P., Liu, Y., Hoi, S. C. H., and Miao, C. (2017). Online multi-task relative similarity learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1823–1829.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.

Hastie, T. and Tibshirani, R. (1996). Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):607–616.

Hazan, E. (2008). Sparse approximate solutions to semidefinite programs. In *Proceedings of the 8-th Latin American Conference on Theoretical informatics*, pages 306–316.

Hazan, E. and Kale, S. (2012). Projection-free online learning. In *Proceedings of the 29th International Conference on Machine Learning*, pages 521–528.

He, J., Li, M., Zhang, H.-J., Tong, H., and Zhang, C. (2004). Manifold-ranking based image retrieval. In *Proceedings of the 12th Annual ACM International Conference on Multimedia*, pages 9–16.

Higham, N. J. (1988). Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and its Applications*, 103:103–118.

Hiriart-Urruty, J.-B. and Lemaréchal, C. (2012). *Fundamentals of Convex Analysis*. Springer Science & Business Media.

Hirzer, M., Roth, P. M., and Bischof, H. (2012a). Person re-identification by efficient impostor-based metric learning. In *Proceedings of the 9-th International Conference on Advanced Video and Signal-Based Surveillance*, pages 203–208.

Hirzer, M., Roth, P. M., Köstinger, M., and Bischof, H. (2012b). Relaxed pairwise learned metric for person re-identification. In *Proceedings of the European Conference on Computer Vision*, pages 780–793.

Hochberg, Y. (1988). A sharper bonferroni procedure for multiple tests of significance. *Biometrika*, 75(4):800–802.

Hoi, S. C. H., Liu, W., Lyu, M. R., and Ma, W.-Y. (2006). Learning distance metrics with contextual constraints for image retrieval. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2072–2078.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70.

Hommel, G. (1988). A stagewise rejective multiple test procedure based on a modified bonferroni test. *Biometrika*, 75(2):383–386.

Horst, R. and Thoai, N. V. (1999). DC programming: Overview. *Journal of Optimization Theory and Applications*, 103(1):1–43.

Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S. S., and Sundararajan, S. (2008). A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th International Conference on Machine Learning*, pages 408–415, New York, NY, USA. ACM.

Hu, J., Lu, J., and Tan, Y.-P. (2014). Discriminative deep metric learning for face verification in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1875–1882.

Hu, J., Lu, J., and Tan, Y. P. (2015a). Deep transfer metric learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 325–333.

Hu, J., Lu, J., and Tan, Y. P. (2018). Sharable and individual multi-view metric learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(9):2281 – 2288.

Hu, J., Lu, J., Yuan, J., and Tan, Y.-P. (2015b). Large margin multi-metric learning for face and kinship verification in the wild. In *Proceedings of the 12th Asian Conference on Computer Vision*, pages 252–267.

Huang, R. and Sun, S. (2013). Kernel regression with sparse metric learning. *Journal of Intelligent and Fuzzy Systems*, 24(4):775–787.

Huang, X., Ye, Y., and Zhang, H. (2014). Extensions of kmeans-type algorithms: A new clustering framework by integrating intracluster compactness and intercluster separation. *IEEE Transactions on Neural Networks and Learning Systems*, 25(8):1433–1446.

Hull, J. J. (1994). A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554.

ILOG (2012). Cplex optimizer.

Inaba, M., Katoh, N., and Imai, H. (1994). Applications of weighted voronoi diagrams and randomization to variance-based k-clustering. In *Proceedings of the 10th Annual Symposium on Computational Geometry*, pages 332–339.

J. Kushner, H. and Yin, G. (2003). *Stochastic Approximation and Recursive Algorithms and Applications*. Springer.

Jaggi, M. (2013). Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning*, pages 427–435.

Jain, P., Kulis, B., Davis, J. V., and Dhillon, I. S. (2012). Metric and kernel learning using a linear transformation. *The Journal of Machine Learning Research*, 13:519–547.

Jain, P., Kulis, B., and Dhillon, I. S. (2010). Inductive regularized learning of kernel functions. In *Advances in Neural Information Processing Systems 23*, pages 946–954.

Jain, P., Kulis, B., Dhillon, I. S., and Grauman, K. (2009). Online metric learning and fast similarity search. In *Advances in Neural Information Processing Systems 21*, pages 761–768.

Jia, H., Cheung, Y., and Liu, J. (2016). A new distance metric for unsupervised learning of categorical data. *IEEE Transactions on Neural Networks and Learning Systems*, 27(5):1065–1079.

Jiang, N., Liu, W., and Wu, Y. (2012). Order determination and sparsity-regularized

metric learning adaptive visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1956–1963.

Jin, R., Wang, S., and Zhou, Y. (2009). Regularized distance metric learning: Theory and algorithm. In *Advances in Neural Information Processing Systems 22*, pages 862–870.

Joachims, T., Finley, T., and Yu, C.-N. J. (2009). Cutting-plane training of structural SVMs. *Machine Learning*, 77:27–59.

Jobson, D. J., Rahman, Z., and Woodell, G. A. (1997). Properties and performance of a center/surround retinex. *IEEE Transactions on Image Processing*, 6:451–462.

Jolliffe, I. (2005). *Principal Component Analysis*. Wiley Online Library.

Kato, T. and Nagano, N. (2010). Metric learning for enzyme active-site search. *Bioinformatics*, 26(21):2698–2704.

Kim, M. and Pavlovic, V. (2010). Structured output ordinal regression for dynamic facial emotion intensity prediction. In *Proceedings of the 11th European Conference on Computer Vision*, pages 649–662.

Kira, K. and Rendell, L. A. (1992). A practical approach to feature selection. In *Proceedings of the 9th International Workshop on Machine Learning*, pages 249–256.

Köstinger, M., Hirzer, M., Wohlhart, P., Roth, P. M., and Bischof, H. (2012). Large scale metric learning from equivalence constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2288–2295.

Kramer, S., Widmer, G., Pfahringer, B., and De Groeve, M. (2001). Prediction of ordinal classes using regression trees. *Fundamenta Informaticae*, 47:1–13.

Kuczyński, J. and Woźniakowski, H. (1992). Estimating the largest eigenvalue by the power and lanczos algorithms with a random start. *SIAM Journal on Matrix Analysis and Applications*, 13(4):1094–1122.

Kulis, B. (2012). Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364.

Kulis, B., Basu, S., Dhillon, I., and Mooney, R. (2009a). Semi-supervised graph clustering: a kernel approach. *Machine Learning*, 74:1–22.

Kulis, B., Jain, P., and Grauman, K. (2009b). Fast similarity search for learned metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2143–2157.

Kulis, B., Saenko, K., and Darrell, T. (2011). What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1785–1792.

Kulis, B., Sustik, M. A., and Dhillon, I. S. (2009c). Low-rank kernel learning with bregman matrix divergences. *The Journal of Machine Learning Research*, 10:341–376.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Kumar, N. and Kummamuru, K. (2008). Semisupervised clustering with metric learning using relative comparisons. *IEEE Transactions on Knowledge and Data Engineering*, 20:496–503.

Lajugie, R., Bach, F., and Arlot, S. (2014). Large-margin metric learning for constrained partitioning problems. In *Proceedings of the 31st International Conference on Machine Learning*, pages 297–305.

Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339.

Law, M. T., Thome, N., and Cord, M. (2013). Quadruplet-wise image similarity learning. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, pages 249–256.

Law, M. T., Yu, Y., Cord, M., and Xing, E. P. (2016). Closed-form training of Mahalanobis distance for supervised clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3909–3917.

Lebanon, G. (2006). Metric learning for text documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:497–508.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, J.-E., Jin, R., and Jain, A. K. (2008). Rank-based distance metric learning: An application to image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Leslie, C., Eskin, E., and Noble, W. S. (2002). The spectrum kernel: A string kernel for svm protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 564–575.

Levine, J. H., Simonds, E. F., Bendall, S. C., Davis, K. L., Amir, E.-A. D., Tadmor, M., Litvin, O., Fienberg, H. G., Jager, A., Zunder, E. R., Finck, R., Gedman, A. L., Radtke, I., Downing, J. R., Pe'er, D., and Nolan, G. P. (2015). Data-driven phenotypic dissection of aml reveals progenitor-like cells that correlate with prognosis. *Cell*, 162:184–197.

Li, C., Liu, Q., Liu, J., and Lu, H. (2012a). Learning distance metric regression for facial age estimation. In *Proceedings of the 21st International Conference on Pattern Recognition*, pages 2327–2330.

Li, C., Liu, Q., Liu, J., and Lu, H. (2015). Ordinal distance metric learning for image ranking. *IEEE Transactions on Neural Networks and Learning Systems*, 26(7):1551–1559.

Li, H., Chen, N., and Li, L. (2012b). Error analysis for matrix elastic-net regularization algorithms. *IEEE Transactions on Neural Networks and Learning Systems*, 23(5):737–748.

Li, M., Wang, Q., Zhang, D., Li, P., and Zuo, W. (2017). Joint distance and similarity measure learning based on triplet-based constraints. *Information Sciences*, 406:119–132.

Li, W., Zhao, R., and Wang, X. (2013a). Human reidentification with transferred metric learning. In *Proceedings of the Asian Conference on Computer Vision*, pages 31–44.

Li, Z., Chang, S., Liang, F., Huang, T. S., Cao, L., and Smith, J. R. (2013b). Learning locally-adaptive decision functions for person verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3610–3617.

Liang, J., Hu, Q., Zhu, P., and Wang, W. (2018). Efficient multi-modal geometric mean metric learning. *Pattern Recognition*, 75:188–198.

Liao, S., Hu, Y., Zhu, X., and Li, S. Z. (2015). Person re-identification by local maximal occurrence representation and metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2197–2206.

Liao, S., Zhao, G., Kellokumpu, V., Pietikainen, M., and Li, S. Z. (2010). Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1301–1306.

Lim, D. and Lanckriet, G. (2014). Efficient learning of Mahalanobis metrics for ranking. In *Proceedings of the Thirty-First International Conference on Machine Learning*, pages 1980–1988.

Lim, D., Lanckriet, G., and McFee, B. (2013). Robust structural metric learning. In *Proceedings of the 30th International Conference on Machine Learning*, pages 615–623.

Lin, C.-J. and Moré, J. J. (1999). Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization*, 9(4):1100–1127.

Lin, H. T. and Li, L. (2012). Reduction from cost-sensitive ordinal ranking to weighted binary classification. *Neural Computation*, 24(5):1329–1367.

Lisanti, G., Masi, I., Bagdanov, A. D., and Bimbo, A. D. (2015). Person re-

identification by iterative re-weighted sparse ranking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37:1629–1642.

Liu, B., Fang, B., Liu, X., Chen, J., Huang, Z., and He, X. (2013). Large margin subspace learning for feature selection. *Pattern Recognition*, 46(10):2798–2806.

Liu, K., Bellet, A., and Sha, F. (2014). Similarity learning for high-dimensional sparse data. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 653–662.

Liu, W., Mu, C., Ji, R., Ma, S., Smith, J. R., and Chang, S.-F. (2015a). Low-rank similarity metric learning in high dimensions. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2792–2799.

Liu, X. and Srivastava, A. (2002). A spectral representation for appearance-based classification and recognition. In *Proceedings of the 16th International Conference onPattern Recognition*, volume 1, pages 37–40.

Liu, X., Wang, H., Wu, Y., Yang, J., and Yang, M. H. (2015b). An ensemble color model for human re-identification. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 868–875.

Liu, Y. and Shen, X. (2006). Multicategory ψ-learning. *Journal of the American Statistical Association*, 101:500–509.

Liu, Y., Shen, X., and Doss, H. (2005). Multicategory ψ-learning and support vector machine: Computational tools. *Journal of Computational and Graphical Statistics*, 14(1):219–236.

Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Luo, Y., Liu, T., Tao, D., and Xu, C. (2014). Decomposition-based transfer distance metric learning for image classification. *IEEE Transactions on Image Processing*, 23(9):3789–3801.

Luo, Z. and Tseng, P. (1992). On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35.

Lux, M., Brinkman, R. R., Chauve, C., Laing, A., Lorenc, A., Abeler-Dörner, L., and Hammer, B. (2018). flowlearn: fast and precise identification and quality checking of cell populations in flow cytometry. *Bioinformatics*, 15:1–9.

Ma, B., Su, Y., and Jurie, F. (2012). Local descriptors encoded by fisher vectors for person re-identification. In *Proceedings of the European Conference on Computer Vision*, pages 413–422.

Mahalanobis, P. C. (1936). On the generalized distance in statistics. In *Proceedings of the National Institute of Sciences (Calcutta) 2*, pages 49–55.

Mahdavi, M., Yang, T., Jin, R., Zhu, S., and Yi, J. (2012). Stochastic gradient descent with only one projection. In *Advances in Neural Information Processing Systems 25*, pages 494–502.

Mason, L., Bartlett, P. L., and Baxter, J. (2000). Improved generalization through explicit optimization of margins. *Machine Learning*, 38(3):243–255.

McFee, B., Galleguillos, C., and Lanckriet, G. (2011). Contextual object localization with multiple kernel nearest neighbor. *IEEE Transactions on Image Processing*, 20(2):570–585.

McFee, B. and Lanckriet, G. R. (2010). Metric learning to rank. In *Proceedings of the 27th International Conference on Machine Learning*, pages 775–782.

Mclachlan, G. J. (2004). *Discriminant Analysis and Statistical Pattern Recognition*. Wiley.

Mei, J., Liu, M., Karimi, H., and Gao, H. (2014). Logdet divergence-based metric learning with triplet constraints and its applications. *IEEE Transactions on Image Processing*, 23(11):4920–4931.

Miao, Y., Tao, X., Sun, Y., Li, Y., and Lu, J. (2015). Risk-based adaptive metric learning for nearest neighbour classification. *Neurocomputing*, 156:33–41.

Mignon, A. and Jurie, F. (2012). PCCA: A new approach for distance learning from sparse pairwise constraints. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 2666–2672.

Mika, S., Rätsch, G., Weston, J., Schölkopf, B., and Müller, K.-R. (1999). Fisher discriminant analysis with kernels. In *Proceedings of the IEEE Intenational Workshop Neural Networks for Signal Processing IX*, pages 41–48.

Moghaddam, B., Jebara, T., and Pentland, A. (2000). Bayesian face recognition. *Pattern Recognition*, 33(11):1771–1782.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of Machine Learning*. The MIT Press.

Monaco, G., Chen, H., Poidinger, M., Chen, J., de Magalhaes, J. P., and Larbi, A. (2016). Flowai: automatic and interactive anomaly discerning tools for flow cytometry data. *Bioinformatics*, 32:2473–2480.

Moore, A. W. (1991). An intoductory tutorial on k-d-trees. In *Extracted from Moore's PhD Thesis-Efficient Memory-based Learning for Robot Control*. Computer Laboratory, University of Cambridge.

Moreno, P. J., Ho, P. P., and Vasconcelos, N. (2004). A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. In *Advances in Neural Information Processing Systems 16*, pages 1385–1392.

Mosek, A. (2010). The MOSEK optimization software.

Moutafis, P., Leng, M., and Kakadiaris, I. A. (2017). An overview and empirical comparison of distance metric learning methods. *IEEE Transactions on Cybernetics*, 47(3):612–625.

Mu, Y., Ding, W., and Tao, D. (2013). Local discriminative distance metrics ensemble learning. *Pattern Recognition*, 46(8):2337–2349.

Nene, S. A., Nayar, S. K., and Murase, H. (1996). Columbia object image library (coil-100). Technical report, Department of Computer Science, Columbia University.

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856.

Nguyen, B. and De Baets, B. (2018a). An approach to supervised distance metric learning based on difference of convex functions programming. *Pattern Recognition*, 81:562–574.

Nguyen, B. and De Baets, B. (2018b). Kernel-based distance metric learning for supervised k-means clustering. *IEEE Transactions on Neural Networks and Learning Systems*, accepted.

Nguyen, B. and De Baets, B. (2019). Kernel distance metric learning using pairwise constraints for person re-identification. *IEEE Transactions on Image Processing*, 28(2):589–600.

Nguyen, B., Ferri, F. J., Morell, C., and De Baets, B. (2019a). An efficient method for clustered multi-metric learning. *Information Sciences*, 471:149–163.

Nguyen, B., Morell, C., and De Baets, B. (2016). Large-scale distance metric learning for k-nearest neighbors regression. *Neurocomputing*, 214:805–814.

Nguyen, B., Morell, C., and De Baets, B. (2017a). Distance metric learning: a two-phase approach. In *Proceedings of the 25th European symposium on artificial neural networks, computational intelligence and machine learning*, pages 123–128.

Nguyen, B., Morell, C., and De Baets, B. (2017b). Distance metric learning with the Universum. *Pattern Recognition Letters*, 100:37–43.

Nguyen, B., Morell, C., and De Baets, B. (2017c). Supervised distance metric learning through maximization of the Jeffrey divergence. *Pattern Recognition*, 64:215–225.

Nguyen, B., Morell, C., and De Baets, B. (2018a). Distance metric learning for ordinal classification based on triplet constraints. *Knowledge-Based Systems*, 142:17–28.

Nguyen, B., Morell, C., and De Baets, B. (2018b). Scalable large-margin distance metric learning using stochastic gradient descent. *IEEE Transactions on Cybernetics*, accepted.

Nguyen, B., Rubbens, P., Kerckhof, F.-M., Boon, N., De Baets, B., and Waegeman, W. (2019b). Learning single-cell distances from cytometry data. *Cytometry Part A*, submitted.

Nguyen, N. and Guo, Y. (2008). Metric learning: A support vector approach. In *The European Conference on Machine Learning*, pages 125–136.

Ojala, T., Pietikainen, M., and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987.

Omohundro, S. M. (1989). *Five Balltree Construction Algorithms*. International Computer Science Institute Berkeley.

O'Neill, K., Aghaeepour, N., Spidlen, J., and Brinkman, R. (2013). Flow cytometry bioinformatics. *PLoS Comput Biol*, 9:e1003365.

Ortega, J. M. and Rheinboldt, W. C. (1979). *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press.

Overton, M. (1988). On minimizing the maximum eigenvalue of a symmetric matrix. *SIAM Journal on Matrix Analysis and Applications*, 9(2):256–268.

Paisitkriangkrai, S., Shen, C., and Van Den Hengel, A. (2015). Learning to rank in person re-identification with metric ensembles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1846–1855.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Parameswaran, S. and Weinberger, K. Q. (2010). Large margin multi-task metric learning. In *Advances in Neural Information Processing Systems 23*, pages 1867–1875.

PASCAL (2011). Pascal (Pattern Analysis, Statistical Modelling and Computational Learning) machine learning benchmarks repository.

Pedagadi, S., Orwell, J., Velastin, S., and Boghossian, B. (2013). Local fisher discriminant analysis for pedestrian re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3318–3325.

Peng, J. and Wei, Y. (2007). Approximating k-means-type clustering via semidefinite programming. *SIAM Journal on Optimization*, 18:186–205.

Peng, X., Lu, C., Yi, Z., and Tang, H. (2018). Connections between nuclear-norm and frobenius-norm-based representations. *IEEE Transactions on Neural Networks and Learning Systems*, 29:218–224.

Peng, X., Lu, J., Yi, Z., and Yan, R. (2017). Automatic subspace learning via principal coefficients embedding. *IEEE Transactions on Cybernetics*, 47(11):3583–3596.

Pérez-Ortiz, M., Cruz-Ramírez, M., Ayllón-Terán, M., Heaton, N., Ciria, R., and Hervás-Martínez, C. (2014). An organ allocation system for liver transplantation based on ordinal regression. *Applied Soft Computing*, 14, Part A:88–98.

Pérez-Ortiz, M., Gutiérrez, P., Carbonero-Ruz, M., and Hervás-Martínez, C. (2016). Semi-supervised learning for ordinal kernel discriminant analysis. *Neural Networks*, 84:57–66.

Petersen, K. B. and Pedersen, M. S. (2012). The matrix cookbook. Technical report, Technical University of Denmark.

Pham Dinh, T., Le, H. M., Le Thi, H. A., and Lauer, F. (2014). A difference of convex functions algorithm for switched linear regression. *IEEE Transactions on Automatic Control*, 59(8):2277–2282.

Pham Dinh, T. and Le Thi, H. A. (1997). Convex analysis approach to DC programming: Theory, algorithms and applications. *Acta Mathematica Vietnamica*, 22(1):289–355.

Pouyan, M. B., Jindal, V., Birjandtalab, J., and Nourani, M. (2016). Single and multi-subject clustering of flow cytometry data for cell-type identification and anomaly detection. *BMC Med Genomics*, 9:41.

Props, R., Monsieurs, P., Mysara, M., Clement, L., and Boon, N. (2016). Measuring the biodiversity of microbial communities by single-cell analysis. *Methods Ecol Evol*, 7:n/a–n/a.

Prosser, B., Zheng, W.-S., Gong, S., and Xiang, T. (2010). Person re-identification by support vector ranking. In *Proceedings of the British Machine Vision Conference*, pages 1–11.

Pyne, S., Hu, X., Wang, K., Rossin, E., Lin, T.-I., Maier, L. M., Baecher-Allan, C., McLachlan, G. J., Tamayo, P., Hafler, D. A., De Jager, P., and Mesirov, J. P. (2009). Automated high-dimensional flow cytometric data analysis. *Proc Natl Acad Sci U S A*, 106(21):8519–8524.

Qi, G.-J., Tang, J., Zha, Z.-J., Chua, T.-S., and Zhang, H.-J. (2009). An efficient sparse metric learning in high-dimensional space via ℓ_1-penalized log-determinant

regularization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 841–848.

Qian, Q., Jin, R., Yi, J., Zhang, L., and Zhu, S. (2015a). Efficient distance metric learning by adaptive sampling and mini-batch stochastic gradient descent (sgd). *Machine Learning*, 99(3):353–372.

Qian, Q., Jin, R., Zhu, S., and Lin, Y. (2015b). Fine-grained visual categorization via multi-stage metric learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3716–3724.

Rahim, A., Meskas, J., Drissler, S., Yue, A., Lorenc, A., Laing, A., Saran, N., White, J., Abeler-Dorner, L., Hayday, A., and Brinkman, R. R. (2018). High throughput automated analysis of big flow cytometry data. *Methods*, 134-135:164–176.

Ramanan, D. and Baker, S. (2011). Local distance functions: A taxonomy, new algorithms, and an evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):794–806.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

Recht, B., Fazel, M., and Parrilo, P. A. (2010). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407.

Robnik-Šikonja, M. and Kononenko, I. (2003). Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning*, 53(1-2):23–69.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan.

Roth, P. M., Hirzer, M., Köstinger, M., Beleznai, C., and Bischof, H. (2014). Mahalanobis distance learning for person re-identification. In *Advances in Computer Vision and Pattern Recognition*, pages 247–267.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326.

Rubbens, P., Props, R., Boon, N., and Waegeman, W. (2017a). Flow cytometric single-cell identification of populations in synthetic bacterial communities. *PLoS One*, 12(1):e0169754.

Rubbens, P., Props, R., Garcia-Timermans, C., Boon, N., and Waegeman, W. (2017b). Stripping flow cytometry: How many detectors do we need for bacterial identification? *Cytometry A*, 91A:1184–1191.

Saenko, K., Kulis, B., Fritz, M., and Darrell, T. (2010). Adapting visual category models to new domains. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, pages 213–226.

Saeys, Y., Van Gassen, S., and Lambrecht, B. (2016). Computational flow cytometry: helping to make sense of high-dimensional immunology data. *Nat Rev Immunol*, 16:449–462.

Sargent, R. W. H. and Sebastian, D. J. (1973). On the convergence of sequential minimization algorithms. *Journal of Optimization Theory and Applications*, 12(6):567–575.

Schapire, R. E., Freund, Y., Barlett, P., and Lee, W. S. (1997). Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings of the 14th International Conference on Machine Learning*, pages 322–330.

Schölkopf, B., Herbrich, R., and Smola, A. J. (2001). A generalized representer theorem. In *Proceedings of the 14th Annual Conference on Computational Learning Theory*, pages 416–426.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319.

Schölkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Schultz, M. and Joachims, T. (2004). Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems 16*, pages 41–48.

Sgier, L., Freimann, R., Zupanic, A., and Kroll, A. (2016). Flow cytometry combined with visne for the analysis of microbial biofilms and detection of microplastics. *Nat Commun*, 7:11587.

Shalev-Shwartz, S., Singer, Y., and Ng, A. Y. (2004). Online and batch learning of pseudo-metrics. In *Proceedings of the 21-st International Conference on Machine Learning*, pages 94–101.

Shalev-Shwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814.

Shalev-Shwartz, S. and Tewari, A. (2011). Stochastic methods for ℓ_1-regularized loss minimization. *The Journal of Machince Learning Research*, 12:1865–1892.

Shalev-Shwartz, S. and Zhang, T. (2013). Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research*, 14(1):567–599.

Shamir, O. and Zhang, T. (2013). Stochastic gradient descent for non-smooth

optimization: Convergence results and optimal averaging schemes. In *Proceedings of the Thirtieth International Conference on Machine Learning*, pages 71–79.

Shashua, A. and Levin, A. (2002). Ranking with large margin principle: Two approaches. In *Advances in Neural Information Processing Systems 15*, pages 937–944.

Shen, C., Kim, J., Liu, F., Wang, L., and van den Hengel, A. (2014). Efficient dual approach to distance metric learning. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2):394–406.

Shen, C., Kim, J., and Wang, L. (2010). Scalable large-margin Mahalanobis distance metric learning. *IEEE Transactions on Neural Networks*, 21(9):1524–1530.

Shen, C., Kim, J., Wang, L., and Van Den Hengel, A. (2012). Positive semidefinite metric learning using boosting-like algorithms. *The Journal of Machine Learning Research*, 13(1):1007–1036.

Shen, X., Tseng, G. C., Zhang, X., and Wong, W. H. (2003). On ψ-learning. *Journal of the American Statistical Association*, 98:724–734.

Sheskin, D. J. (2007). *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC, fourth edition.

Shi, Y., Bellet, A., and Sha, F. (2014). Sparse compositional metric learning. In *Procedings of the 28th AAAI Conference on Artificial Intelligence*, pages 2078–2084.

Slonim, N. and Tishby, N. (2000). Document clustering using word clusters via the information bottleneck method. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 208–215.

Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems 29*, pages 1857–1865.

Soleimani, B. H. and Matwin, S. (2016). Nonlinear dimensionality reduction by unit ball embedding (ube) and its application to image clustering. In *Proceedings of the 15th IEEE International Conference on Machine Learning and Applications*, pages 983–988.

Song, H. O., Jegelka, S., Rathod, V., and Murphy, K. (2017). Deep metric learning via facility location. In *Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition*.

Song, H. O., Xiang, Y., Jegelka, S., and Savarese, S. (2016). Deep metric learning via lifted structured feature embedding. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 4004–4012.

Sun, C., Wang, D., and Lu, H. (2017). Person re-identification via distance metric learning with latent variables. *IEEE Transactions on Image Processing*, 26(1):23–34.

Sun, Y. (2007). Iterative RELIEF for feature weighting: Algorithms, theories, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1035–1051.

Sun, Y., Todorovic, S., and Goodison, S. (2010). Local-learning-based feature selection for high-dimensional data analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1610–1626.

Takeuchi, I., Nakagawa, M., and Seto, M. (2009). Metric learning for dna microarray data analysis. In *Proceedingso of International Workshop on StatisticalMechanical Informatics*.

Tao, D., Cheng, J., Song, M., and Lin, X. (2016a). Manifold ranking-based matrix factorization for saliency detection. *IEEE Transactions on Neural Networks and Learning Systems*, 27:1122–1134.

Tao, D., Guo, Y., Li, Y., and Gao, X. (2018). Tensor rank preserving discriminant analysis for facial recognition. *IEEE Transactions on Image Processing*, 27:325–334.

Tao, D., Guo, Y., Song, M., Li, Y., Yu, Z., and Tang, Y. Y. (2016b). Person re-identification by dual-regularized KISS metric learning. *IEEE Transactions on Image Processing*, 25(6):2726–2738.

Tao, D., Guo, Y., Yu, B., Pang, J., and Yu, Z. (2017). Deep multi-view feature learning for person re-identification. *IEEE Transactions on Circuits and Systems for Video Technology*, To appear(In press):To appear.

Tao, D., Jin, L., Wang, Y., and Li, X. (2015). Person reidentification by minimum classification error-based KISS metric learning. *IEEE Transactions on Cybernetics*, 45(2):242–252.

Tenenbaum, J. B., Vin, D. S., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

Tian, Q., Chen, S., and Qiao, L. (2016). Ordinal margin metric learning and its extension for cross-distribution image data. *Information Sciences*, 349-350:50–64.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58(1):267–288.

Torresani, L. and Lee, K.-C. (2007). Large margin component analysis. In *Advances in Neural Information Processing Systems 19*, pages 1385–1392.

Tran, D. and Sorokin, A. (2008). Human activity recognition with metric learning.

In *Proceedings of the 10th European Conference on Computer Vision: Part I*, pages 548–561.

Triguero, I., González, S., Moyano, J. M., García, S., Alcalá-Fdez, J., Luengo, J., Fernández, A., del Jesús, M. J., Sánchez, L., and Herrera, F. (2017). KEEL 3.0: An open source software for multi-stage analysis in data mining. *International Journal of Computational Intelligence Systems*, 10:1238–1249.

Tseng, P. (2001). Convergence of a block coordinate descent method for non-differentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *The Journal of Machine Learning Research*, 6:1453–1484.

van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *The Journal of Machine Learning Research*, 9:2579–2605.

Van Der Maaten, L., Postma, E., and Van den Herik, J. (2009). Dimensionality reduction: a comparative. *The Journal of Machine Learning Research*, 10:66–71.

Van Gassen, S., Callebaut, B., Van Helden, M. J., Lambrecht, B. N., Demeester, P., Dhaene, T., and Saeys, Y. (2015). Flowsom: Using self-organizing maps for visualization and interpretation of cytometry data. *Cytometry A*, 87A:636–645.

Vapnik, V. N. (1998). *Statistical Learning Theory*. John Wiley & Sons.

Vedaldi, A. and Zisserman, A. (2012). Efficient additive kernels via explicit feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):480–492.

Vinh, N. X., Epps, J., and Bailey, J. (2009). Information theoretic measures for clusterings comparison: Is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1073–1080.

Vinh, N. X., Epps, J., and Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11:2837–2854.

Wagstaff, K., Cardie, C., Rogers, S., and Schrödl, S. (2001). Constrained k-means clustering with background knowledge. In *Proceedings of the 18th International Conference on Machine Learning*, pages 577–584.

Wang, F. (2011a). Semisupervised metric learning by maximizing constraint margin. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(4):931–939.

Wang, F., Zuo, W., Zhang, L., Meng, D., and Zhang, D. (2015). A kernel

classification framework for metric learning. *IEEE Transactions on Neural Networks and Learning Systems*, 26:1950–1962.

Wang, H., Nie, F., and Huang, H. (2014a). Robust distance metric learning via simultaneous l1-norm minimization and maximization. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1836–1844.

Wang, J., Kalousis, A., and Woznica, A. (2012). Parametric local metric learning for nearest neighbor classification. In *Advances in Neural Information Processing Systems 25*, pages 1601–1609.

Wang, J., Zhou, F., Wen, S., Liu, X., and Lin, Y. (2017). Deep metric learning with angular loss. In *The IEEE International Conference on Computer Vision*, pages 2593–2601.

Wang, Q., Yuen, P. C., and Feng, G. (2013). Semi-supervised metric learning via topology preserving multiple semi-supervised assumptions. *Pattern Recognition*, 46(9):2576–2587.

Wang, W., Hu, B.-G., and Wang, Z.-F. (2014b). Globality and locality incorporation in distance metric learning. *Neurocomputing*, 129:185–198.

Wang, X. (2011b). A fast exact k-nearest neighbors algorithm for high dimensional search using k-means clustering and triangle inequality. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1293–1299.

Weber, L. M. and Robinson, M. D. (2016). Comparison of clustering methods for high-dimensional single-cell flow and mass cytometry data. *Cytometry A*, 89:1084–1096.

Weinberger, K. and Tesauro, G. (2007). Metric learning for kernel regression. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, pages 608–615.

Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

Williams, C. K. and Rasmussen, C. E. (1996). Gaussian processes for regression. In *Advances in Neural Information Processing Systems 8*, pages 514–520. MIT press.

Wright, S. J. (1997). *Primal-Dual Interior-Point Methods*. Siam.

Wu, L., Hoi, S. C. H., Jin, R., Zhu, J., and Yu, N. (2012). Learning bregman distance functions for semi-supervised clustering. *IEEE Transactions on Knowledge and Data Engineering*, 24:478–491.

Xiang, S., Nie, F., and Zhang, C. (2008). Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition*, 41(12):3600–3612.

Xiao, B., Yang, X., Xu, Y., and Zha, H. (2009). Learning distance metric for regression by semidefinite programming with application to human age estimation. In *Proceedings of the 17th ACM International Conference on Multimedia*, pages 451–460, New York, NY, USA.

Xiao, T., Li, H., Ouyang, W., and Wang, X. (2016). Learning deep feature representations with domain guided dropout for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1249–1258.

Xing, E. P., Jordan, M. I., Russell, S., and Ng, A. (2002). Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems 14*, pages 505–512.

Xiong, F., Gou, M., Camps, O., and Sznaier, M. (2014). Person re-identification using kernel-based metric learning methods. In *Proceedings of the European Conference on Computer Vision*, pages 1–16.

Xiong, H. and Chen, X.-W. (2006). Kernel-based distance metric learning for microarray data classification. *BMC Bioinformatics*, 7(1):1–11.

Yan, S., Xu, D., Zhang, B., j. Zhang, H., Yang, Q., and Lin, S. (2007). Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):40–51.

Yang, J. and Xu, H. (2016). Metric learning based object recognition and retrieval. *Neurocomputing*, 190:70–81.

Yang, P., Huang, K., and Liu, C.-L. (2011). Multi-task low-rank metric learning based on common subspace. In *Proceedings of the 18th International Conference on Neural Information Processing*, pages 151–159, Berlin, Heidelberg. Springer Berlin Heidelberg.

Yang, P., Huang, K., and Liu, C.-L. (2013). Geometry preserving multi-task metric learning. *Machine Learning*, 92(1):133–175.

Yang, Y., Liao, S., Lei, Z., and Li, S. Z. (2016). Large scale similarity learning using similar pairs for person verification. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 3655–3661.

Yang, Y., Yang, J., Yan, J., Liao, S., Yi, D., and Li, S. Z. (2014). Salient color names for person re-identification. In *Proceedings of the European Conference on Computer Vision*, pages 536–551.

Yeung, D.-Y. and Chang, H. (2006). Extending the relevant component analysis

algorithm for metric learning using both positive and negative equivalence constraints. *Pattern Recognition*, 39(5):1007 – 1010.

Yeung, D. Y. and Chang, H. (2007). A kernel approach for semisupervised metric learning. *IEEE Transactions on Neural Networks*, 18:141–149.

Yin, X., Chen, S., Hu, E., and Zhang, D. (2010). Semi-supervised clustering with metric learning: An adaptive kernel method. *Pattern Recognition*, 43(4):1320–1333.

Ying, Y. and Li, P. (2012). Distance metric learning with eigenvalue optimization. *The Journal of Machine Learning Research*, 13(1):1–26.

Yu, J., Tao, D., Li, J., and Cheng, J. (2014). Semantic preserving distance metric learning and applications. *Information Sciences*, 281:674–686.

Yu, Y. and Schuurmans, D. (2011). Rank/norm regularization with closed-form solutions: Application to subspace clustering. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 778–785.

Yuille, A. L. and Rangarajan, A. (2002). The concave-convex procedure (CCCP). In *Advances in Neural Information Processing Systems 14*, pages 1033–1040.

Zha, H., He, X., Ding, C., Gu, M., and Simon, H. D. (2002). Spectral relaxation for k-means clustering. In *Advances in Neural Information Processing Systems 14*, pages 1057–1064.

Zhang, H., Patel, V. M., and Chellappa, R. (2017). Hierarchical multimodal metric learning for multimodal classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2925–2933.

Zhang, H., Yu, J., Wang, M., and Liu, Y. (2012a). Semi-supervised distance metric learning based on local linear regression for data clustering. *Neurocomputing*, 93:100–105.

Zhang, J. and Zhang, L. (2017). Efficient stochastic optimization for low-rank distance metric learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 933–939.

Zhang, Y. and Yeung, D.-Y. (2010). Transfer metric learning by learning task relationships. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1199–1208.

Zhang, Z., Zhao, M., and Chow, T. W. (2012b). Constrained large margin local projection algorithms and extensions for multimodal dimensionality reduction. *Pattern Recognition*, 45(12):4466–4493.

Zhao, C., Chen, Y., Wei, Z., Miao, D., and Gu, X. (2018). QRKISS: A two-stage metric learning via QR-decomposition and KISS for person re-identification. *Neural Processing Letters*, 1:1–24.

Zhao, R., Oyang, W., and Wang, X. (2017). Person re-identification by saliency learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(2):356–370.

Zheng, W.-S., Gong, S., and Xiang, T. (2009). Associating groups of people. In *Proceedings of the British Machine Vision Conference*, volume 1, pages 1–11.

Zheng, W. S., Gong, S., and Xiang, T. (2013). Reidentification by relative distance comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(3):653–668.

Zheng, Y., Fan, J., Zhang, J., and Gao, X. (2017). Hierarchical learning of multi-task sparse metrics for large-scale image classification. *Pattern Recognition*, 67:97–109.

Zhou, G.-T., Lan, T., Vahdat, A., and Mori, G. (2013). Latent maximum margin clustering. In *Advances in Neural Information Processing Systems 26*, pages 28–36.

Zou, P.-C., Wang, J., Chen, S., and Chen, H. (2016). Margin distribution explanation on metric learning for nearest neighbor classification. *Neurocomputing*, 177:168–178.

Zou, W., Zhu, S., Yu, K., and Ng, A. Y. (2012). Deep learning of invariant features via simulated fixations in video. In *Advances in Neural Information Processing Systems 25*, pages 3203–3211.

Curriculum Vitae

Personalia

Name	Bac Nguyen Cong
Date of birth	January 22, 1989
Place of birth	Bac Ninh, Vietnam
Nationality	Vietnamese
E-mail	nguyencongbacbk@gmail.com

Educations

- 2013-2015: MSc. in Computer Science, Universidad Central "Marta Abreu" de Las Villas, Villa Clara, Cuba.
- 2009-2014: BSc. in Computer Science, Universidad Central "Marta Abreu" de Las Villas, Villa Clara, Cuba.

Research experiences

- Summer 2018: Machine learning software engineer internship, Radix.ai, Brussels, Belgium
- Winter 2017: Visiting researcher, Universitat de València, València, Spain

Languages

- Vietnamese: Native speaker
- Spanish: Fluent
- English: Fluent

Scientific outputs

Publications in international journals (ISI-papers)

- Nguyen, B., Morell, C., and De Baets, B. (2016). Large-scale distance metric learning for k-nearest neighbors regression. *Neurocomputing*, 214:805–814
- Nguyen, B., Morell, C., and De Baets, B. (2017b). Distance metric learning with the Universum. *Pattern Recognition Letters*, 100:37–43
- Nguyen, B., Morell, C., and De Baets, B. (2017c). Supervised distance metric learning through maximization of the Jeffrey divergence. *Pattern Recognition*, 64:215–225
- Nguyen, B. and De Baets, B. (2018b). Kernel-based distance metric learning for supervised k-means clustering. *IEEE Transactions on Neural Networks and Learning Systems*, accepted
- Nguyen, B. and De Baets, B. (2018a). An approach to supervised distance metric learning based on difference of convex functions programming. *Pattern Recognition*, 81:562–574
- Nguyen, B., Morell, C., and De Baets, B. (2018a). Distance metric learning for ordinal classification based on triplet constraints. *Knowledge-Based Systems*, 142:17–28
- Nguyen, B., Morell, C., and De Baets, B. (2018b). Scalable large-margin distance metric learning using stochastic gradient descent. *IEEE Transactions on Cybernetics*, accepted
- Beernaerts, J., Derie, R., Nguyen, B., Vansteenkiste, P., De Baets, B., Deconinck, F., Lenoir, M., De Clercq, D., and Van de Weghe, N. (2019). Assessing the potential of the qualitative trajectory calculus to detect gait pathologies: a case study of children with developmental coordination disorder. *Computer Methods in Biomechanics and Biomedical Engineering*, accepted
- Nguyen, B. and De Baets, B. (2019). Kernel distance metric learning using pairwise constraints for person re-identification. *IEEE Transactions on Image Processing*, 28(2):589–600
- Nguyen, B., Ferri, F. J., Morell, C., and De Baets, B. (2019a). An efficient method for clustered multi-metric learning. *Information Sciences*, 471:149–163
- Nguyen, B., Rubbens, P., Kerckhof, F.-M., Boon, N., De Baets, B., and Waegeman, W. (2019b). Learning single-cell distances from cytometry data. *Cytometry Part A*, submitted

Conference proceedings

- Nguyen, B., Morell, C., and De Baets, B. (2017a). Distance metric learning: a two-phase approach. In *Proceedings of the 25th European symposium on artificial neural networks, computational intelligence and machine learning*, pages 123–128