Exact and Efficient Algorithms for Pairwise Learning

Exacte en efficiënte algoritmes voor paarsgewijs leren

Michiel Stock

Promotoren: prof. dr. Bernard De Baets en prof. dr. Willem Waegeman
Proefschrift voor het behalen van de graad van
Doctor in de Toegepaste Biologische Wetenschappen

Vakgroep Wiskundige Modellering, Statistiek en Bioinformatica
Voorzitter: prof. dr. Bernard De Baets
Faculteit Bio-ingenieurswetenschappen
Academiejaar 2016-2017

Examencommissie

Prof. dr. Colin Janssen, voorzitter
Vakgroep Toegepaste ecologie & milieubiologie
Faculteit Bio-ingenieurswetenschappen
Universiteit Gent

Prof. dr. ir. Wim Van Criekinge, secretaris
Vakgroep Wiskundige modellering, Statistiek en Bio-informatica
Faculteit Bio-ingenieurswetenschappen
Universiteit Gent

Prof. dr. ir. Guy Smagghe
Vakgroep Gewasbescherming
Faculteit Bio-ingenieurswetenschappen
Universiteit Gent

Dr. ir. Jan Verwaeren
Vakgroep Toegepaste Biowetenschappen
Faculteit Bio-ingenieurswetenschappen
Universiteit Gent

Prof. dr. ir. Joni Dambre
Vakgroep Elektronica en Informatiesystemen,
Faculteit Ingenieurswetenschappen en Architectuur
Universiteit Gent

Prof. dr. ir. Pierre Geurts
Department of Electrical engineering and Computer Science,
Institut Montefiore

Prof. dr. Tapio Pahikkala
Department of Information Technology,
Turku University.

Voorwoord

Everything must be made as simple as possible. But not simpler.
- Albert Einstein, on the importance of
tuning the regularization parameter

Sometimes science is more art than science, Morty. A lot of people don't get that.
- Rick Sanchez

Mijn master cel-en genbiotechnologie begon ik met een vol enthousiasme. Eindelijk kon ik de major computationele biologie volgen. Exotisch klinkende vakken zoals 'bio-informatica', 'machine learning' en 'computationele modellen' zouden me leren hoe de grote vragen van *life, the universe and everything* op te lossen met enkel mijn verstand en mijn laptop. Nieuwe eiwitten ontwerpen, kanker genezen, ..., het was allemaal maar slechts een paar lijntjes code verwijderd. Later zou ik leren dat een dergelijke ingesteldheid in wetenschappelijke kringen gekend staat als *physics envy*.

Na zulke hoge verwachtingen kon er natuurlijk enkel maar een teleurstelling volgen.

Het was blijkbaar mogelijk om de 3D structuur van een eiwit te voorspellen. Als je de 3D structuur had van een eiwit dat er op lijkt. Blijkbaar kan je als bio-ingenieur enkel maar dingen voorspellen als je veel data tot je beschikking hebt. Flauw. Empirisch. Valsspelen. We hebben nog altijd labo's nodig. Het had enkele maanden nodig om in te zinken dat die algoritmes eigenlijk een elegante toepassingen zijn van de probabiliteits- en de evolutietheorie. Complexe (levende) systemen kan je enkel maar begrijpen door experimenten te doen en algoritmes zijn krachtige hulpmiddelen om informatie uit die experimenten te halen.

Als eindexamen voor machine learning moesten we een artikel voorstellen aan de groep. Gezien ik toen maar in het eerste masterjaar zat had ik zelf geen leeslijst van artikels en doorzocht dan maar de voorgestelde lijst van artikels. Ik stootte op '*A new pairwise kernel for biological network inference*

with support vector machines' van Jean-Phillipe Vert. Dit artikel besprak een interessante probleemstelling (voorspellen van eiwit-eiwit interacties) en bevatte de goede mix van gekende en eerder enigmatische wiskunde (het gebruik van een mysterieus symbool van een cirkel met een kruis in). Ik presenteerde dit artikel, samen met wat eigen experimenten. Later heb ik er mijn masterproef over gedaan en daarna een doctoraat. Als ik nu die finale presentatie van toen bekijk zie ik dat die ruwweg dezelfde elementen bespreekt als mijn, op het moment van schrijven, half afgewerkte presentatie voor de publieke verdediging. Ik laat het aan de lezer om er het zijne van te denken.

De elementen die mij intrigeerden in mijn onderzoek naar 'paarsgewijs leren' zijn waarschijnlijk hetzelfde voor menig machine learning fanaat: met een methode die op een half blaadje uitgelegd kan worden kan je honderden verschillende problemen benaderen (oplossen?). De probleemstelling, iets voorspellen in functie van twee dingen, was gelukkig breed genoeg om dit van heel veel verschillend uitgangspunten te kunnen benaderen, zeker voor iemand die graag snel iets nieuws probeert. Gelukkig heb ik dan ook de kans gekregen om verschillende boeiende toepassingen binnen de bioingenieurswetenschappen uit te werken, vaak in samenwerkingen met anderen.

Ik hoop dat deze doctoraatsthesis een deel van mijn enthousiasme voor computationele biologie en machine learning kan overbrengen.

Michiel Stock
April 2017

Dankwoord

Het dankwoord is zonder twijfel het meest gelezen hoofdstuk van een doctoraatsthesis. Terecht, want net zoals de bibliografie geeft het aan dat dit werk er niet zou gekomen zijn zonder de hulp van talloze anderen. Net zoals een bibliografie is die ook nooit zo compleet als zou moeten en door tijdsdruk vergeet ik ongetwijfeld verschillende belangrijke personen.

Eerst en vooral wil ik mijn promotoren, Bernard en Willem, bedanken. Ik kon bij hen altijd terecht voor raad over de correcte machine learning terminologie en over hoe lang een spatie nu exact moet zijn. Een doctoraat zonder promotor(s) is als een gezin zonder ouders, ik wens hen nadrukkelijk bedanken voor hun inzet, advies, kennis en, bij momenten, geduld en relativeringsvermogen gedurende de laatste vijf jaar.

Verder zou ik ook graag mijn collega's van de vakgroep bedanken. Mijn bureaugenoten, Gang, Hilde en Peter, voor het tolereren van de immer uitdijende hoeveelheid papier en boeken op mijn bureau en de sporadische ernstige en minder ernstige discussies. Bij uitbreiding wil ik voor al mijn collega's van vakgroep BW10 waardering uiten. Raúl, wiens deadlines het laatste half jaar met de mijne verweven zijn, Aisling om Netflix *recommendations* uit te wisselen, Wouter om mij te leren wat een IPA is, Bram om mijn passie voor koken te delen, Marlies en Jim om me te laten moeien met hun onderzoek, Tinne om de pauzes zoeter te maken en de vele vele anderen die ik elke werkdag in de koffiekamer hoop tegen te komen. Ook mijn waardering voor mede-statistiek-assistent en goede vriend Matthijs, voor het helpen uitdenken van een beetje-maar-niet-te ambigue examenvragen en de vele leuke kookavondjes van de laatste jaren.

Als assistent bij de vakgroep wiskunde zie je zowat alle bio-ingenieurs passeren. Zonder twijfel het leukste aspect van mijn loopbaan als assistent was het uitwerken en begeleiden van de thesissen en studentenprojecten.

Bedankt aan Ayla, Marlies, Wai Kit, Laurentijn, Lander en de talloze projectstudenten om samen iets nieuws uit te proberen.

Deelnemen aan de iGEM competitie was een onverwachts en bijzonder leuk zijspoor. Het project mocht dan wel cool zijn, de mensen, Bram, Bram, Bob, Chari, Griet, David, Maarten, Marjan, Sofie, Steven, Wouter en Wim hebben het onvergetelijk gemaakt. De jamboree in Boston was de leukste trip in het doctoraat, al was het maar door de walvissen. Ik hoop nog vele avonden op *Dungeons & Dragons* avontuur te kunnen gaan!

Mijn waardering voor mijn familie, ouders, schoonouders en oma's die altijd voor mij klaarstaan, ook al dat we misschien niet altijd in dezelfde wereld leven. Oudere en nieuwere vrienden, Bert, Elina, Koen en Hannes, Ine, Ira, Sofie, Simon, Steven, ..., bedankt om er te zijn!

Tenslotte wil ik nog graag mijn partner en beste vriend, Francis, bedanken. Geen paragraaf kan uitdrukken hoe blij ik ben dat je er elke dag bent.

Michiel Stock
April 2017

Samenvatting

In veel toepassingen komt men problemen tegen waar eigenschappen van *paren* van objecten voorspeld moeten worden, zeker in de levenswetenschappen. Bijvoorbeeld, voor een gegeven eiwit en ligand wil men voorspellen of deze een moleculaire interactie vertonen. Hiervoor maakt men typisch gebruik van een dataset met gelabelde paren $\{(u_i, v_i, y_i) \mid i \in \{1 \dots n\}\}$, waar u_i en v_i elk een object zijn die samen een geassocieerd label y_i hebben. Vaak wordt dit soort datasets voorgesteld als een tabel van interacties of als een netwerk of graaf waar het label de connectie tussen de knopen of objecten karakteriseert. Informeel is het doel van paarsgewijs leren om een functie $f(\cdot, \cdot)$ te vinden waarbij $f(u_i, v_i) \approx y_i$ voor elke i.

In deze doctoraatsthesis heb ik verschillende exacte en efficiënte algoritmes voor paarsgewijs leren ontwikkeld en onderzocht. Deze thesis werd gedreven door drie grote onderzoeksvragen. *Hoe kan men paarsgewijs leren gebruiken?* Deze vraag omvat ook de onderliggende vragen hoe we de prestatie van een model realistisch in kunnen schatten en hoe we paarsgewijze modellen efficiënt kunnen trainen, valideren en gebruiken. *Hoe verhoudt paarsgewijs leren zich tot andere machine learning methoden?* En, ten slotte, *hoe kunnen we paarsgewijs leren gebruiken voor vernieuwende toepassingen?*. Dit werk vat vijf jaar onderzoek samen, zowel theoretisch als experimenteel, over het gebruik en de toepassingen van paarsgewijs leren door middel van kernel ridge regressie via het Kronecker product.

Om te beginnen beschrijf ik grondig en formeel wat paarsgewijs leren inhoudt. Via verschillende voorbeelden verduidelijk ik wat paarsgewijze data is en geef aan waarom het belangrijk is om dergelijke data te modelleren.

Ik geef een uitgebreid overzicht van de verschillende wiskundige eigenschappen van paarsgewijze data en hoe dergelijke eigenschappen in het model ingebouwd kunnen worden. Bijvoorbeeld, een symmetrische relatie tussen twee eiwitten A en B betekent dat als eiwit A interageert met een eiwit B, eiwit B bijgevolg ook interageert met eiwit A. Verder bespreek ik hoe dergelijke paarsgewijze modellen gebruikt en geëvalueerd worden. Ik geef ook een overzicht hoe paarsgewijs leren gebruikt wordt om moleculaire netwerken te construeren, interacties tussen soorten te voorspellen en om aanbevelingssystemen systems op te bouwen.

Modellen bouwen op basis van een grote dataset van voorbeelden is het studiedomein van machine learning ('machinaal leren'). Een brede waaier van machine learning-methoden zijn doorheen de jaren ontwikkeld. Dit werk focust zich echter hoofdzakelijk op zogenaamde kernelmethoden. Kernels kunnen gebruikt worden om complexe, niet-lineaire voorstellingen van objecten te genereren. Enkele eenvoudige, maar krachtige, methoden voor regressie, classificatie en dimensiereductie worden besproken. Deze vormen de bouwstenen voor de meer gespecialiseerde methoden voor paarsgewijs leren.

Vertrekkende van kernel ridge regressie kunnen verschillende efficiënte paarsgewijze leeralgoritmes worden ontwikkeld. Belangrijk bij deze is dat de tijdscomplexiteit schaalt met het aantal objecten, niet met het aantal geobserveerde interacties. In een theoretisch luik toon ik aan hoe al deze verschillende leeralgoritmes met elkaar verbonden zijn en bestudeer ik hun eigenschappen met betrekking tot de theorie van statistisch leren. In een computationeel luik ontwikkel ik verschillende vernieuwende algoritmes om efficiënt paarsgewijze modellen te trainen en te evalueren. Alle voorgestelde methoden worden ondersteund door een uitgebreide experimentele analyse waarin verschillende toepassingen gaande van eiwit-ligand interactievoorspelling, restauratie van beelden tot classificatie van documenten aan bod komen.

Voor de meeste praktische toepassingen wordt een paarsgewijs model gebruikt om een aantal relevante objecten te vinden in een databank. Bijvoorbeeld, in geneesmiddelenontwerp wil men een collectie kandidaat-molecules zoeken met een hoge affiniteit voor een bepaald eiwit. Dit probleem wordt ook wel het top-K bevragingsprobleem genoemd. Ik toon aan dat het zogenaamd thresholdalgoritme gebruikt kan worden om het zoekproces drastisch te versnellen. De eigenschappen van dit algoritme worden zowel theoretisch als empirisch geanalyseerd.

Naast het ontwikkelen van een theoretisch raamwerk voor paarsgewijs leren worden twee vernieuwende toepassingen uit de levenswetenschappen besproken: één uit de moleculaire biologie en één uit de ecologie. Een eerste toepassing betreft het annoteren van enzymes. Het is geweten dat de functie van een enzyme grotendeels bepaald wordt door de actieve site (het deel van het eiwit dat met het substraat bindt). Door gebruik te maken van een maat voor verwantschap gedefinieerd over deze actieve site kan men enzymes detecteren met dezelfde of een gelijkaardige functie als een bepaald referentieënzyme. Door een dataset met geannoteerde enzymen te gebruiken kan de prestatie van de annotatie sterk verbeterd worden ten opzichte van een rangschikking die enkel gebaseerd is op verwantschap.

In een tweede toepassing bestudeer ik het detecteren van vals negatieve interacties in species interactie netwerken. Dergelijke netwerken, die bijvoorbeeld plant-mier- of gastheer-parasietinteracties beschrijven, worden meestal verzameld via observatiestudies. In dergelijke studies zijn ontbrekende interacties inherent aanwezig. Via een eenvoudige lineaire filter, een speciaal geval van de meer algemene methodes voor paarsgewijs leren, kunnen vals negatieve interacties accuraat gevonden worden.

Ik ben er van overtuigd dat mijn doctoraatsproefschrift een collectie van nuttige en algemeen toepasbare werktuigen aanbiedt om paarsgewijze predictieproblemen aan te pakken.

Summary

In many applications, not the least in the life sciences, one encounters problems in which one wants to predict properties of *pairs* of objects. For example, given a protein and a ligand, one wants to predict whether they will show a molecular interaction. To this end, one has a dataset of labeled pairs $\{(u_i, v_i, y_i) \mid i \in \{1 \dots n\}\}$, with u_i and v_i two objects and y_i an associated label. Such a dataset is often represented as either a table of interactions or as a network or a graph in which the label characterizes the connection between the objects. The goal of pairwise learning is, informally, to find a function $f(\cdot, \cdot)$, such that $f(u_i, v_i) \approx y_i$ for every i.

In this PhD thesis I study and develop several exact and efficient algorithms for pairwise learning. This dissertation is driven by three main research questions. *How can we use pairwise learning?* This includes the question of how to get a realistic performance estimate and how to efficiently train, validate and use pairwise models. *How is pairwise learning related to other machine learning methods?* And, finally, *How can pairwise learning be used for novel applications?* This work summarizes five years of research, both theoretical as well as empirical, on the use and application of pairwise learning using Kronecker kernel-based ridge regression.

To start, I thoroughly and formally describe pairwise learning. Using ample examples, I explain what pairwise data is and why it is important to model such data. I give an extensive overview of the different mathematical properties that pairwise models could exhibit. For example, if protein A interacts with protein B, then necessarily, protein B interacts with protein A. This is an example of a symmetric relationship; if this is known to occur

in the data, then we desire the model to have such a property as well. Next, I discuss how such pairwise models can be used and by extension, how their performance can be assessed. Modelling properties of pairs of objects is commonplace in many situations. I provide an overview of how pairwise learning is used to infer molecular networks, to predict interactions between species and to build recommender systems.

Learning models based on a dataset of examples are the domain of machine learning. Many machine learning methods exist, though this work focuses predominantly on kernel methods. Kernels can be used to create complex nonlinear representations of objects by computing a dot product between implicit feature representations. Some simple, though powerful methods for regression, classification and dimensionality reduction are presented. These are the foundation for the more specific methods for pairwise learning.

Starting from kernel ridge regression, I show how several efficient pairwise learning algorithms can be derived. A key property of these is that their time complexity scales with the number of objects, not the number of interactions. In a theoretical part, I show how all the different learning methods are related and formally analyze their learning theoretic properties. In a computation part, I derive several novel algorithms to efficiently train and evaluate pairwise models. Finally, I provide an extensive experimental analysis of the algorithms for applications as diverse as drug-target prediction, image restoration and document classification.

For many practical applications, one wants to use a pairwise model to find a set of highly-scoring objects in a database with respect to some query. For example, in drug design the goal is to find a set of drug candidates with a high predicted affinity towards the protein target. This problem is denoted as the top-K inference problem. I demonstrate that the threshold algorithm can greatly speed up the search process by using a precomputed set of sorted lists. The properties of this algorithm are analyzed both theoretically and empirically.

In addition to developing and analyzing a theoretical framework for pairwise learning, two novel applications in the life sciences are investigated, one in molecular biology and one in ecology. A first application is in enzyme annotation. It is well known that the function of an enzyme is largely determined by the physicochemical properties of its active cleft. Using similarities describing this active site, one can find enzymes that have the same or a similar function as the query enzyme. Using a dataset of annotated enzymes,

the performance of the annotation can be greatly improved compared to an unsupervised approach.

As a second application, I study the problem of detecting false negative interactions in species interaction datasets, such as plant-ant or host-parasite networks. Such datasets are often collected by means of observation studies. As a result, they often contain many absent interactions. Using a simple linear filter, a special case of the general models discussed earlier, false negative interactions can be detected with great accuracy.

I am convinced that my PhD dissertation provides a general and complete toolkit for solving pairwise prediction problems.

Contents

1 Introduction **1**

2 What is pairwise learning and why should you be interested? **13**

2.1 Pairwise datasets and pairwise models 13
2.2 Structure and prior knowledge in pairwise data 17
2.2.1 Monadic data 17
2.2.2 Dyadic data 24
2.3 Using and evaluating pairwise models 29
2.3.1 The conditional ranking problem 29
2.3.2 Four settings for prediction 32
2.3.3 Performance metrics for pairwise problems 37
2.4 Motivating examples at different scales 45
2.4.1 In the cell: molecular interactions 45
2.4.2 In the environment: species interactions 47
2.4.3 On your computer: recommender systems 53

3 Learning with kernels **57**

3.1 A quick introduction to machine learning 57
3.2 Kernels and the representer theorem 59
3.3 Regression using kernel ridge regression 63
3.4 Dimension reduction using kernel principal component analysis 66
3.5 Classification using kernel Fisher discriminant analysis 68
3.6 Concluding remarks and the future of kernels 71
3.6.1 Representing distributions using kernels 72

3.6.2 Randomization for approximating kernels and constructing nonlinear representations 73

4 Pairwise learning with Kronecker product kernels 75

4.1 Pairwise learning as multi-task learning 76
4.2 Chapter outline 77
4.3 Related settings to pairwise learning 79
4.4 Pairwise learning using ridge-regression-based methods 83
4.4.1 Independent-task kernel ridge regression 83
4.4.2 Pairwise and Kronecker kernel ridge regression 84
4.4.3 Conditional ranking on pairwise data 86
4.4.4 Two-step kernel ridge regression 88
4.5 Theoretical considerations 91
4.5.1 Equivalence between two-step and other kernel ridge regression methods 91
4.5.2 Universality of the Kronecker product pairwise kernel 97
4.5.3 Spectral interpretation 99
4.5.4 Learning relations using pairwise kernels 107
4.5.5 Kernels without object features 108
4.6 Efficient algorithms for Kronecker kernel-based pairwise learning 113
4.6.1 Efficient hold-out computation 113
4.6.2 Learning with mini batches 120
4.6.3 An iterative transductive algorithm for incomplete data 123
4.7 Experiments 128
4.7.1 Study of regularization for the different settings 129
4.7.2 A study of missing value imputation 136
4.7.3 Comparison of different transfer learning settings . . . 143
4.7.4 Online-learning of hierarchical text classification . . . 149
4.8 Conclusions and perspectives 151

5 Efficient exact top-K inference in pairwise prediction 155

5.1 Introduction and formal problem description 156
5.2 Exact algorithms for top-K inference 161
5.3 Recommending recipes: an illustrative example 174
5.4 Application domains and relationships with separable linear relational (SEP-LR) models 178

5.4.1 Memory-based and model-based collaborative filtering 179
5.4.2 Multi-label classification, multivariate regression and multi-task learning . . . 180
5.4.3 Pairwise learning, content-based filtering and supervised network inference . . . 182
5.5 Experimental results . . . 182
5.5.1 Data dependencies in collaborative filtering . . . 183
5.5.2 Running time and partial threshold algorithm using Uniprot data . . . 185
5.5.3 Behaviour of individual queries . . . 188
5.5.4 Large-scale text classification . . . 190
5.6 Conclusions and future perspectives . . . 191

6 Functional annotation of enzymes using conditional ranking 195
6.1 Introduction . . . 196
6.2 Material and methods . . . 200
6.2.1 Database . . . 200
6.2.2 Similarity measures for cavities . . . 203
6.2.3 Unsupervised ranking . . . 207
6.2.4 Supervised ranking . . . 210
6.2.5 Performance measures for ranking . . . 213
6.2.6 Experimental set-up . . . 214
6.3 Results and discussion . . . 215
6.3.1 Differences between cavity-based similarities and data sets . . . 215
6.3.2 The benefits of supervised ranking . . . 218
6.3.3 Differences between performance measures . . . 219
6.4 Related work . . . 224
6.5 Conclusion . . . 225

7 Detecting false negatives in species interaction datasets 227
7.1 A filter for species interaction matrices . . . 228
7.2 Material and methods . . . 232
7.3 Detecting false negative species interactions . . . 234
7.4 Discussion . . . 241

8 Conclusions and future perspectives 245
8.1 A general system representation (Setting A) . . . 248

8.2 Structured output prediction with labels (Setting D) 250

Bibliography **253**

A Properties of matrices **289**

B Curriculum vitae **297**

List of abbreviations

AUC	area under the ROC curve
CB	CavBase
C-index	concordance index
FP	fingerprints
GIP	Gaussian interaction profile
KRR	kernel ridge regression
LOO	leave-one-out
LPCS	labeled point cloud superposition
MAE	mean absolute error
MCS	maximum common subgraph
MSE	mean squared error
R^2	fraction of explained variance
RMSE	root mean squared error
ROC	receiver operating characteristic
PCA	principal component analysis
SEP-LR	separable linear relational
SVD	singular value decomposition
SW	Smith-Waterman
TSS	two-step single hyperparameter

List of symbols

Scalars

n	number of training instances
m	number of objects of type u
q	number of objects of type v
n'	number of test instances
d	dimensionality of the feature description related to u
r	dimensionality of the feature description related to v
R	rank of a model
K	size of top for inference
λ	regularization parameter
y	label

Matrices, vectors and their operations

$\mathbf{A}$	a matrix
$\mathbf{A}_{i.}$	the i-th row of $\mathbf{A}$ (row vector)
$\mathbf{A}_{.j}$	the j-th column of $\mathbf{A}$ (column vector)
$\mathbf{A}_{ij}$	the element at the i-th row and the j-th column of $\mathbf{A}$ (scalar)
$\mathrm{tr}(\mathbf{A})$	the trace of matrix $\mathbf{A}$
$\mathbf{A}^\top$	the transpose of matrix $\mathbf{A}$
$\mathbf{A}^{-1}$	the inverse of matrix $\mathbf{A}$
$\mathrm{diag}_v(\mathbf{A})$	a vector of diagonal elements of matrix $\mathbf{A}$
$\mathrm{diag}_m(\mathbf{A})$	a diagonal matrix with the diagonal elements of matrix $\mathbf{A}$
$\mathrm{vec}(\cdot)$	the vectorization operator
$\sigma_i(\mathbf{A})$	the i-th eigenvalue of matrix $\mathbf{A}$
$\mathbb{I}_n$	a $n \times n$ identity matrix (dimension will be omitted if clear from context)
$\mathbb{J}_{nm}$	a $n \times m$ matrix of ones (dimension will be omitted if clear from context)
$\mathbf{Y}$	the label matrix
$\mathbf{F}$	the matrix with the predictions corresponding to $\mathbf{Y}$
$\mathbf{y}$	the label vector
$\mathbf{f}$	the matrix with the predictions corresponding to $\mathbf{y}$
$\mathbf{K}$	Gram matrix of the kernel $k(\cdot, \cdot)$
$\mathbf{G}$	Gram matrix of the kernel $g(\cdot, \cdot)$
$\mathbf{\Gamma}$	Gram matrix of the kernel $\Gamma(\cdot, \cdot)$
$\mathbf{H}$	the hat matrix

Objects and sets

x	instance or datapoint
u	object
v	object of another space
$\mathbb{R}$	set of real numbers
$\mathcal{U}$	space of objects u
$\mathcal{V}$	space of objects v
U	set of objects u in training set
V	set of objects v in training set
U'	database of u for inference
S	training set
T	test set
S_v^K	top-K set using query v

Functions

$k(\cdot,\cdot)$	kernel function (usually associated with u)
$g(\cdot,\cdot)$	kernel function (usually associated with v)
$\Gamma((\cdot,\cdot),(\cdot,\cdot))$	pairwise kernel function (usually associated with (u,v))
$\boldsymbol{\phi}(\cdot)$	feature representation associated with k
$\boldsymbol{\psi}(\cdot)$	feature representation associated with g
$\mathbf{p}(\cdot)$	model representation of u
$\mathbf{q}(\cdot)$	model representation of v
$\sigma(\cdot)$	a function to map a real-valued output to $[0,1]$
$f(\cdot,\cdot)$	prediction function
$Q(\cdot,\cdot)$	relation
$\mathcal{L}(\cdot,\cdot)$	loss function
$\varphi(\cdot)$	filter function

1
Introduction

Let us talk about our reading preferences. I like novels, often with a theme of magic realism and nostalgia. In the last years I have developed an appetite for the fantasy genre and I also enjoy the occasional science fiction book. For non-fiction, I used to love reading about evolution, though in these last years I have come across many interesting books about artificial intelligence. Of course, there are superb books about nearly any topic, from cooking to trees.

Although there are generally acclaimed books and, likewise, some books are objectively garbage, you and I will find some books on which we respectfully disagree on how well we like them. People do, generally speaking, not all like the same things. Some people like to read about science, some people do not. Some people like young adult novels, others would not come near them. What people read reveals something about their preferences. Similarly, the ratings tell something about the books as well. By taking a closer look at such data we can hopefully discover some books we have somehow missed.

In the spirit of fun, I have collected some of my friends' book ratings from the social cataloging website Goodreads. Nine persons, including myself, contributed a list of books they have read and, in some cases, rated them on a scale from one to five. Below is a table indicating which books have been read by which persons. For brevity, only the books that were read by at least two persons are given.

Book title	Bart	Cedric	Cons	Hannes	Kate	Koen	Marc	Michiel	Scott
Harry Potter and the Half-Blood Prince (Harry P...	0	1	0	0	1	0	0	1	0
Harry Potter and the Order of the Phoenix (Harr...	0	1	0	0	1	0	0	1	0
Harry Potter and the Sorcerer's Stone (Harry Po...	0	1	0	0	1	0	0	1	0
Harry Potter and the Prisoner of Azkaban (Harry...	0	1	0	0	1	0	0	1	0
Harry Potter and the Goblet of Fire (Harry Pott...	0	1	0	0	1	0	0	1	0
The Hitchhiker's Guide to the Galaxy (Hitchhike...	0	1	1	0	1	0	0	1	0
A Short History of Nearly Everything	0	0	1	0	0	0	0	1	1
The Fellowship of the Ring (The Lord of the Rin...	0	0	1	0	1	0	1	1	0
Perfume: The Story of a Murderer	0	1	0	0	0	1	0	1	0
Angels & Demons (Robert Langdon, #1)	0	0	1	0	1	0	0	1	0
The Da Vinci Code (Robert Langdon, #2)	0	1	0	0	1	1	0	1	0
A Brief History of Time	0	0	0	0	0	1	0	1	1
American Gods (American Gods, #1)	1	0	1	0	1	0	0	1	0
The Great Gatsby	0	1	0	0	1	0	1	1	0
Slaughterhouse-Five	0	1	1	0	0	1	0	0	0
The Catcher in the Rye	1	1	1	0	1	0	1	0	0
1984	0	0	1	0	0	1	1	0	0
The Hobbit	1	1	1	0	1	0	1	1	0
Animal Farm	0	1	1	0	0	0	1	1	0
Lord of the Flies	1	1	0	0	1	1	1	1	0
The Restaurant at the End of the Universe (Hitc...	0	0	1	0	1	0	0	1	0
The God Delusion	0	1	1	0	0	0	0	1	0
The Two Towers (The Lord of the Rings, #2)	0	0	1	0	1	0	1	1	0
Harry Potter and the Chamber of Secrets (Harry ...	0	1	0	0	1	0	0	1	0
Coraline	0	1	0	0	1	0	0	1	0
The Amber Spyglass (His Dark Materials, #3)	1	0	0	0	1	0	0	1	0
The Return of the King (The Lord of the Rings, #3)	0	0	1	0	1	0	1	0	0
Pattern Recognition and Machine Learning	1	1	0	0	0	0	0	1	0
Harry Potter and the Deathly Hallows (Harry Pot...	0	1	0	0	1	0	0	1	0
Watchmen	0	1	1	0	0	0	0	1	0
Outliers: The Story of Success	0	0	1	0	0	0	1	0	1
Moonwalking with Einstein: The Art and Science ...	0	0	1	0	0	0	1	1	0
Will Grayson, Will Grayson	0	0	0	0	1	1	0	1	0
De jongen die nooit heeft bestaan	0	0	0	1	0	1	0	1	0

The pattern in the table above is far from random. For example, some people have read more of the books in this table than others (who might have mainly read books that others have not). An example of the dependence in the books people have read can be found by looking at books of the same series. People have either read all the books of the Harry Potter series or none. Similar observations can be made for the Lord of the Rings books, books by Neil Gaiman etc. The similarity between the books people have read and the ratings they gave to these books can be visualized by similarity matrices, see Figure 1.1. As one can see in the left matrix, some people such as Cons and Kate have

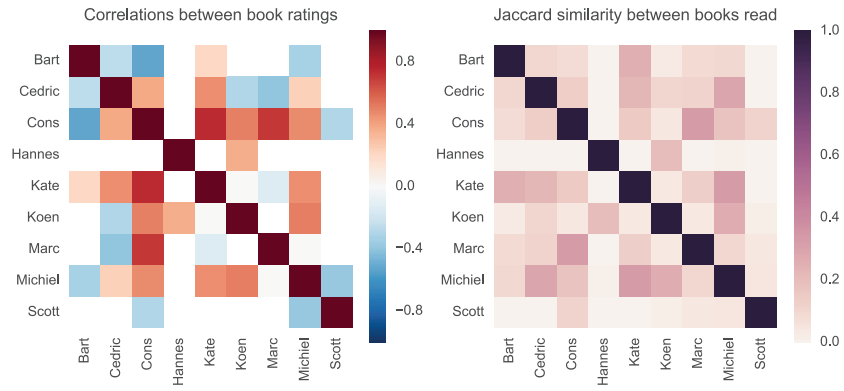


Figure 1.1: (left) Correlations in book ratings for different users. White cells indicate insufficient joint ratings between two persons for computing the Pearson correlation. (right) Jaccard index measuring the similarity in the number of books two persons have both read (calculated only on books read by at least two persons).

very similar ratings for books, while for example Scott and I show a negative correlation. Books rated higher by Scott are on average rated lower by me. To find new stuff to read, it is likely a good idea to start looking for books read by people whose ratings are highly correlated with ones own.

The ratings give direct feedback on whether someone enjoyed a book or not and hence provide the most useful information. The downside is that there are no ratings for most person-book pairs and some people do not even bother to rate the books that they have read. In contrast, by using only the information whether someone has read a book or not, we only have indirect feedback, though this information is available for all person-book combinations.

In addition to the ratings and reading statistics, we can also use some side-information on the books and ratings. In this case, the users can be structured into a social network, depicted in Figure 1.2. Friends often share book recommendations with each other, so the structure of the social network can help understand why people have

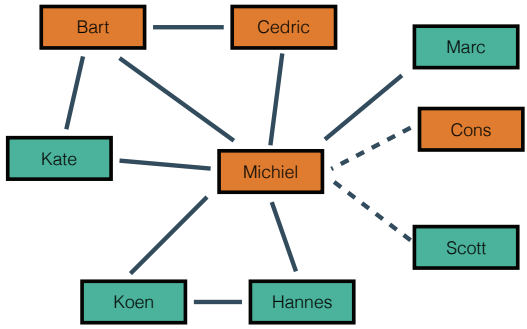


Figure 1.2: Social network for the people involved in the Goodreads survey. Full links between persons indicate that they know each other, dotted links indicate that they are only Goodread friends. Nodes in orange indicate that the person is professionally involved in computer science.

read the same book. Kate has recommended several fantasy books and mangas to me; this emerges as a high Jaccard index between our reading profiles. Often one should take properties of both the reader and as the book into account. Take, for instance, the seminal 'Pattern Recognition and Machine Learning', a standard reference book for machine learning. This book has been read by Cedric, Bart and me, people quite involved in computer science. As such it is a safe bet that this would be a relevant book for Cons. A similar reasoning can be used to recommend 'De jongen die nooit bestaan heeft' to Bart.

The above toy example illustrates the concept of pairwise data and making predictions in this context. This PhD thesis is about models yhat make predictions for *pairs* of objects. Pairwise datasets are often represented as a graph or as a network, where the interaction is the connection or link between two nodes. In this work I will rather put the emphasis on the representation where the pairwise data is structured in a table or matrix. These two representations are depicted in Figure 1.3.

As we will see shortly, pairwise data appear in many settings. My humble example of book ratings is an instance of a recommender

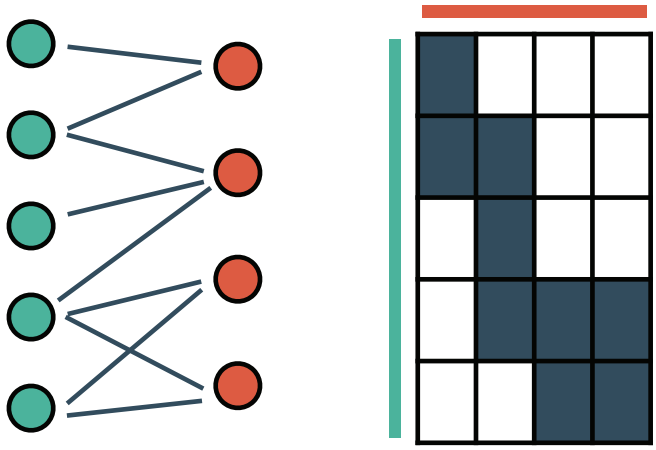


Figure 1.3: Two equivalent representations of a pairwise dataset: as a graph (left) and as a matrix (right). In the graph representation, the nodes correspond to the objects (e.g. persons and books) and the edges to the interactions (e.g. read the book). In the matrix representation, the rows and columns represent the objects, while the values in the cells indicate the interaction or relation.

system, a system used to recommend relevant items to users. Internet companies such as Netflix heavily rely on recommender systems to tempt customers with films and television series. Networks are also omnipresent in, for example, biology. It should be no surprise that many interesting pairwise settings appear in the life sciences. At the molecular scale we find all sorts of molecules interacting with each other in an intricate dance that makes life possible, see Figure 1.4 for an example of a molecular interaction dataset. Also at the macro scale, nothing exists in isolation. All species in the phylogenetic tree form interactions with other species by, for example, "eating", "being eaten by", "chemically disrupting", "hiding in", "tricking into having coitus with", "letting their genetic material being transported by" etc. Some of these interactions are shown in Figure 1.5. All these interactions are of great importance for understanding living systems and great effort is done to collect, interpret and model them.

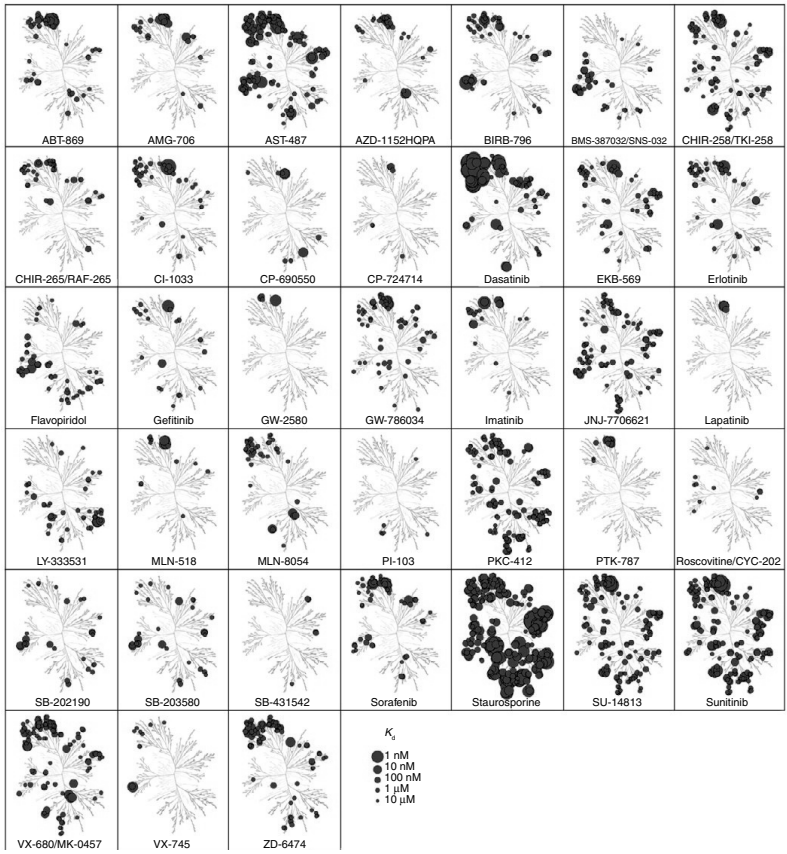


Figure 1.4: Example of a protein-ligand dataset for 38 kinase-inhibitors towards 317 kinases. Kinases found to bind a ligand are marked with red circles, where larger circles indicate higher-affinity binding. There is a large difference in the specificity and selectivity of the kinases for different ligands. The dendrogram hints at the link between the evolution of the proteins and their function. Image from Karaman et al. (2008)



Figure 1.5: Three examples of ecological interactions. (left) Mutualism between a clownfish and a sea anemone. Both partners benefit from the interaction. (middle) Commensalism between a fly and phoretic mites. The mites benefit while the fly has neither advantage nor disadvantage from the interaction. (right) Antagonism between a walnut tree and other plants. The roots of the walnut tree excrete chemicals to hinder the other plants' growth. Images from Wikipedia Commons.

This PhD thesis has three main goals:

1. Investigating the practical issues and bottle-necks of pairwise learning. Here the focus is on how to train and test a pairwise model for situations with two, one or no new objects. A large part of this work is concerned with developing efficient and scalable algorithms for this purpose.

2. Linking pairwise learning methods to other techniques in machine learning that deals with complex objects such as, transfer learning, matrix factorisation, collaborative filtering, multi-task learning and structured output prediction.

3. Exploring applications of pairwise learning, with an emphasis on applications in the life sciences.

Outline of this dissertation

In Chapter 2, I give a general overview of the problem of pairwise learning. I discuss the details and peculiarities of pairwise datasets, pairwise prediction functions and how to correctly evaluate them. These pairwise prediction functions are often used to *rank* objects

conditional on another object. At the end of the chapter I discuss some motivating application fields where pairwise learning is used. I consider this chapter to be the heart of this dissertation, as it discusses the central thesis.

Chapter 3 reviews kernel methods, for the readers who are not familiar with them or in need of a small review of this matter. In Chapter 4, I develop the workhorses of this PhD thesis: models based on pairwise Kronecker kernels. Not for the faint of heart, this chapter shows the motivation and derivation of several pairwise kernel ridge regression methods, analyzes their learning properties and shows some novel algorithms for learning and evaluating these models. All developed theory is supplemented with an extensive experimental section to demonstrate the working and relevance of the methods and algorithms. This chapter is an extended version of a manuscript in preparation. It has also been published as a conference paper (Pahikkala et al., 2014). The Kronecker kernel ridge regression is an established method, but two-step kernel ridge regression and related algorithms were developed during my PhD research.

Pairwise models are often used to generate a ranking of objects, conditional on another object (e.g. generate a ranking of books most relevant for a person). In practice, one is often interested in the top of this ranking, for example the ten most relevant objects. Chapter 5 discusses this problem as top-K inference. I discuss the Threshold Algorithm, a simple algorithm to find the top-K most relevant objects according to some pairwise prediction function. This allows for finding the correct top without needing to score every object in a database. I argue that this algorithm can work in tandem with a variety of important pairwise models. This chapter also ends with a large experimental section to empirically analyze the Threshold Algorithm. This chapter is an adapted version of a journal paper (Stock et al., 2016).

The next two chapters addess applications of pairwise learning for biological problems. In Chapter 6, I use pairwise learning to develop a novel method to functionally annotate enzymes based on structural similarity of their active cleft. This chapter originates from a paper published in IEEE Transactions on Computational Biology and Bioinformatics (Stock et al., 2014). An application in ecology is discussed in Chapter 7. Here I show that by using a very simple linear filter, false negative interactions can be found in species interaction datasets such as plant-pollinator networks of host-parasite networks. This has recently been published in Scientific Reports (Stock et al., 2017). Chapter 8 wraps everything up with some conclusions and future perspectives.

In the last years I had the good fortune to collaborate with many people to tackle some interesting problems. Some of this work is not discussed in depth in this dissertation, but has been published as journal papers. An overview:

- In Stock et al., (2013), I used some techniques from data mining to explore and predict pairwise interactions between methanotrophic and heterotrophic bacteria. This was done in collaboration with dr. Sven Hoefman and dr. Kim Heylen.

- In Costello et al. (2014), I collaborated in a meta-study to predict the effect of several drugs on different cancer cell lines.

- In Gonnelli et al. (2015), I helped to develop Nokoi, a proteomics tool for scoring peptides after mass spectrometry analysis. This allows users to make a correct estimation of the false discovery rate of hits of relevant peptides.

- In De Clercq et al. (2015), I helped to develop a model to recommend ingredients to add some extra oomph to a dish.

- In De Paepe et al. (2016), I helped to develop a model to predict miRNA-mRNA interactions.

On the next pages it is listed how the different chapters and sections contribute to the three main goals of this PhD dissertation.

Goal	Section	Remark
1. Algorithms and good practices of pairwise learning	Chapter 2	General introduction to pairwise learning, the structure of pairwise datasets and how to assess the performance of pairwise prediction functions.
	Section 4.4	Overview of new and established kernel-based methods for pairwise learning.
	Section 4.5.5	Discussion of pairwise learning without object features.
	Section 4.6	Efficient algorithms to train and validate pairwise models.
	Section 4.7	Experimental results, examples on how to train and validate pairwise model using the developed methods.
	Sections 5.1 and 5.2	Discussion of the top-K inference problem and efficient algorithms to solve this problem.
	Chapter 7	Discussion of false negatives in pairwise datasets.
2. Links to other machine learning settings	Section 2.4	Discussion of pairwise learning in different contexts (molecular biology, ecology and collaborative filtering).
	Sections 4.1 and 4.3	Discussion of how different machine learning settings can be seen as pairwise learning.
	Section 5.4	Overview of machine learning problems that can be seen as top-K inference problems.

	Chapter 6	Discussion of a conditional ranking problem in a bioinformatics context.
	Chapter 7	Pairwise learning as a linear filter on matrices. Comparing the linear filter with singular value decomposition.
3. Applications of pairwise learning	Section 2.4	Discussion of pairwise learning applications in molecular biology, ecology and collaborative filtering.
	Section 4.7	Experimental results: pairwise learning in protein-ligand interaction prediction, image restoration and text classification.
	Section 5.3	Toy example of a recipe recommendation engine.
	Section 5.5	Experimental results of top-K inference in collaborative filtering, enzyme annotation and text classification.
	Chapter 6	Development of an enzyme annotation engine using conditional ranking methods.
	Chapter 7	Application of finding false negative interactions in species interaction datasets.

So, let us start with taking a closer look to what pairwise learning really is and how it can be used.

2

What is pairwise learning and why should you be interested?

2.1 Pairwise datasets and pairwise models

This is a PhD thesis about algorithms that can extract patterns from pairwise data, hence it makes sense to start with an overview of what pairwise datasets are comprised of. The datasets that are discussed in this work consist of a set of labeled dyads, where a dyad is simply a pair of objects. Such datasets are used to train, tune and validate statistical models. Simple as this may seem, it shall be opportune to discuss each element of the dataset into more detail. Firstly, we informally define 'an object'.

Definition 2.1 (Object). *We say that $x \in \mathcal{X}$ is an object if it is the smallest uniquely identifiable element of a dataset for which it is meaningful to make a prediction.*

Such a definition may seem to be somewhat vague or even circular, but it is important to see an object as the building block of the dataset. Any smaller part would loose its meaning in a machine learning context. In other works, what I denote here as an object is

sometimes referred to as an instance or data point. Some examples of objects and their spaces:

- A microscopy image from the set of all microscopy images.
- The protein mitogen-activated protein kinase 3 from the set of all human proteins.
- The species *Scathophaga stercoraria* (yellow dung fly) from the set of all pollinators occurring in a certain region.
- Michiel, a user of the social cataloging website `Goodreads.com`.

In all of these cases, we can think of many interesting problems to make a prediction for these objects, e.g. estimating the number of cells that are on the microscopy image, predicting the function of the protein or guessing the age of Michiel based on his reading habits. Note that we make a distinction between the object itself and its representation. The same protein can be represented as a string of amino acids or as a 3D structure. In many cases such representations are complex and structured and the most relevant representation will be highly problem-specific. Moreover, most machine learning algorithms require a numeric feature description of fixed length. The representation of the object, and how this is translated into numeric features, will be discussed in Chapter 3.

A pair of objects (u, v) with $u \in \mathcal{U}$ and $v \in \mathcal{V}$ is called a dyad. We make the distinction between the monadic case, when u and v are of the same domain, i.e. $\mathcal{U} = \mathcal{V}$ and the dyadic case, when the domains of the two objects differ[1]. Monadic relations are typically encountered in domains such as network analysis, for example when predicting a link in a social network or predicting a physical interaction between different proteins. In dyadic cases, the objects come from different

[1]The term 'monadic' is not typically used in this context. In this work I use it to indicate that there is only a single space of objects from which the dyads are constructed.

domains. For example, when predicting whether a person will be interested in a certain book or to predict the binding affinity between a protein and a ligand. It is tempting to assume that the monadic case is merely a simpler version of the dyadic case. As we will see shortly, the monadic case has his own pitfalls.

In pairwise learning, the goal is to predict properties of dyads. To this end, each dyad that is used to build and validate the models has a corresponding label y. Depending on the type of problem, this label can take different forms:

- The label can be real-valued, e.g. predicting the binding energy between two proteins. In this case we deal with a regression problem.
- The label can be a binary variable, e.g. predicting whether a protein will bind to a ligand or not. In such cases, the two outcomes are usually coded as $\{0, 1\}$ or $\{-1, 1\}$. Such problems are referred to as binary classification problems.
- The label can be one of multiple values, e.g. predicting the nature of the interaction between a protein and a ligand (substrate, inhibitor, inducer). This is called a multi-class classification problem. Often, a multi-class classification problem is transformed into multiple binary classification problems (Bishop, 2006).
- The label can be an ordinal value[2], e.g. predicting whether the interaction between two proteins is not occurring, weak, moderate or strong. Such problems are referred to as ordinal regression problems and can be thought of something in between regression and multi-class classification.

Dyadic prediction models are fitted using a set of labeled examples: the training set. This training set $S = \{(u_h, v_h, y_h) \mid h = 1, \ldots, n\}$ is

[2]Ordinal values are values for which there exists a natural ordering, yet cannot be identified with real numbers, for example *freezing, cold, agreeable, hot* and *very hot*.

a set of labeled dyads. Let $U = \{u_i \mid i = 1, \ldots, m\}$ and $V = \{v_j \mid j = 1, \ldots, q\}$ denote, respectively, the sets of distinct objects of both types in the training set with $m = |U|$ and $q = |V|$. In this work, we will often deal with *complete* datasets, defined below.

Definition 2.2 (Complete dataset). *A training set is complete if it contains exactly one labeled example for every dyad* $(u, v) \in U \times V$.

For complete training sets, we introduce a further notation for the matrix of labels $\mathbf{Y} \in \mathbb{R}^{m \times q}$, so that its rows are indexed by the objects in U and the columns by the objects in V. Working with complete matrices often allows for many algorithmic shortcuts when training and validating models.

The goal of pairwise learning or dyadic prediction is to learn a function $f(\cdot, \cdot)$ to make predictions for new dyads (dyads with one of two objects not encountered in the training dataset). In this work, $f : \mathcal{U} \times \mathcal{V} \to \mathbb{R}$, or $f : \mathcal{U}^2 \to \mathbb{R}$ in the monadic case. Such a function is natural for regression problems. It can also be straightforwardly used for binary classification problems, for example, by setting a threshold at zero where negative values denote the one class and positive values the other. In cases that require the probability of belonging to a certain class, we can use an increasing isomorphism $\sigma : \mathbb{R} \to [0, 1]$ to 'squeeze' the output of the prediction function in the unit interval[3]. The logistic map, i.e.

$$\sigma(x) = \frac{1}{1 + e^{-x}}\,, \tag{2.1}$$

would be an obvious choice. When dealing with binary classification problems, we will usually not bother with post-processing the prediction function. I will motivate using this value directly for ranking, which is in many cases more relevant than classification.

[3] An increasing isomorphism is used instead of an increasing function, as the former is by definition invertible. This is a useful property as it implies that every value of $f(u, v)$ is mapped to an unique point in $[0, 1]$.

Many real-world machine learning problems can be naturally represented as pairwise learning problems. In contrast to more traditional learning settings, the goal here consists of making predictions for pairs of objects, each of them being characterized by a feature representation. Amongst others, applications of that kind emerge in biology (e.g. predicting mRNA-miRNA interactions), medicine (e.g. design of personalized drugs), chemistry (e.g. prediction of binding between two types of molecules), ecology (e.g. prediction of host-parasite interactions), social network analysis (e.g. finding links between persons) and recommender systems (e.g. recommending personalized products to users).

2.2 Structure and prior knowledge in pairwise data

In the previous section, I discussed what pairwise datasets and pairwise models are. In this section I will elaborate on some specific properties of monadic and dyadic datasets.

2.2.1 Monadic data

Monadic data is often encountered when dealing with networks or graphs. Networks are one of the principal tools for modelling complex systems in all domains of science (Strogatz, 2001), including protein interactions (Antal et al., 2009), ingredient use in recipes (Ahn et al., 2011) and microbiomes (Faust and Raes, 2012). Many machine learning algorithms have been developed to predict links in such networks. Often, the goal is to predict whether a link occurs between two nodes. This is an example of the pairwise learning setting where one solves a binary classification problem (absence or presence of a link). Using pairwise regression, one can predict a value of the link between two nodes, for example the interaction strength between proteins. When

dealing with a monadic setting, I will use a slightly different terminology: I will often use the term 'relational learning' rather than pairwise learning when discussing monadic problems. What follows is an adapted summary of the work by Waegeman et al. (2012), which discusses the general issues when learning graded relations.

Such data is usually structured as a graph $G = (\mathcal{U}, \mathcal{E}, Q)$, where $\mathcal{U}$ corresponds to the set of vertices (or nodes), $\mathcal{E} \subseteq \mathcal{U}^2$ represents the set of edges or links, and $Q(\cdot, \cdot)$ is a binary relation between two vertices. Depending on the application, the following types of relations are frequently encountered:

- **Crisp relations**: when $Q : \mathcal{U}^2 \to \{0, 1\}$. This corresponds to a pairwise binary classification problem with the two vertices as inputs.

- $[0, 1]$**-valued relations**: when $Q : \mathcal{U}^2 \to [0, 1]$. This is a regression problem, but with a restriction on the image of the prediction function.

- **Ordinal-valued relations**: when Q maps to an ordinal value.

In this section, I will mainly focus on $[0, 1]$-valued relations, a setting that is very important in the context of probability theory, decision theory and fuzzy set theory. Recall that the general monadic prediction function, $f(\cdot, \cdot)$, has a real-valued image. Consider an increasing isomorphism $\nabla : \mathbb{R} \to [0, 1]$ that satisfies $\nabla(x) = 1 - \nabla(-x)$, e.g. the logistic map (Eq. (2.1)). Any monadic pairwise prediction function can be transformed into a $[0, 1]$-valued relation Q by

$$Q(u, u') = \nabla(f(u, u')) \,, \qquad \forall (u, u') \in \mathcal{U}^2 \,.$$

Monadic relations can exhibit several interesting properties. These can be incorporated in the pairwise prediction function as prior knowledge.

A first import property that might hold for relations is symmetry. Symmetric pairwise prediction functions or relations can be defined as follows.

DEFINITION 2.3 (Symmetric function). *A pairwise prediction function* $f : \mathcal{U}^2 \to \mathbb{R}$ *is symmetric if for all* $(u, u') \in \mathcal{U}^2$ *it holds that* $f(u, u') = f(u', u)$.

DEFINITION 2.4 (Symmetric relation). *A binary relation* $Q : \mathcal{U}^2 \to [0, 1]$ *is symmetric if for all* $(u, u') \in \mathcal{U}^2$ *it holds that* $Q(u, u') = Q(u', u)$.

Symmetric relations occur whenever the data can be represented as an undirected graph. For example, protein-protein networks are usually thought of as symmetric; the interaction between protein A and B is the same as the interaction between B and A.

A second often-encountered property is anti-symmetry or reciprocity, defined as follows.

DEFINITION 2.5 (Anti-symmetric function). *A pairwise prediction function* $f : \mathcal{U}^2 \to \mathbb{R}$ *is anti-symmetric if for all* $(u, u') \in \mathcal{U}^2$ *it holds that* $f(u, u') = -f(u', u)$.

DEFINITION 2.6 (Reciprocal relation). *A binary relation* $Q : \mathcal{U}^2 \to [0, 1]$ *is reciprocal if for all* $(u, u') \in \mathcal{U}^2$ *it holds that* $Q(u, u') = 1 - Q(u', u)$.

The transformation $\nabla(\cdot)$ can be used to transform an anti-symmetric pairwise prediction function into a reciprocal relation. Hence, if one can incorporate anti-symmetry in the learned pairwise prediction function, reciprocal relations can be represented. Reciprocal relations occur frequently in domains such as preference learning, choice probabilities, winning probabilities, gene regulation, species competition, etc. For example, in a Pokémon battle, the probability that, say, Raichu will defeat Slowbro is the reciprocal of the probability that

Slowbro will defeat Raichu. Reciprocal relations can be represented as weighted directed graphs, where the weight of an edge determines the direction of that edge. Weights greater than 0.5 represent edges going in one direction, edges with a weight smaller than 0.5 represent an edge going in the other direction.

Furthermore, consider the notions of transitivity and intransitivity. Formally, but without loosing ourselves into details, transitivity for symmetrical and reciprocal relations is defined as follows (Luce and Suppes, 1965; De Baets et al., 2006).

DEFINITION 2.7 (g-stochastic transitivity). *Let $g(\cdot)$ be an increasing $[1/2,1]^2 \to [0,1]$ mapping. A reciprocal relation $Q : \mathcal{U}^2 \to [0,1]$ is g-stochastic transitive if for any $(u_1,u_2,u_3) \in \mathcal{U}^3$*

$$(Q(u_1,u_2) \geq 1/2 \wedge Q(u_2,u_3) \geq 1/2 \Rightarrow Q(u_1,u_3) \geq g(Q(u_1,u_2),Q(u_2,u_3))\,.$$

DEFINITION 2.8 (T-transitivity). *A symmetric relation $Q : \mathcal{U}^2 \to [0,1]$ is T-transitive with T a t-norm if for any $(u_1,u_2,u_3) \in \mathcal{U}^3$*

$$T(Q(u_1,u_2),Q(u_2,u_3)) \leq Q(u_1,u_3)\,.$$

A small Pokémon example will help to clarify the difference between between these types of transitivity. Take the evolution line of Squirtle: Squirtle → Wartortle → Blastoise. Intuitively, g-stochastic transitivity implies that if Squirtle is more likely to be defeated by Wartortle and Wartortle is more likely to be defeated by Blastoise, it implies that Squirtle is more likely to be defeated by Blastoise than the converse. Taking the minimum T-norm, T-transitivity would be that the probability of Blastoise defeating Squirtle is at least the minimum of the probability of Wartortle defeating Squirtle and the probability of Blastoise defeating Wartortle.

Transitivity can roughly be interpreted as a mathematical formulation of the rationale that if A is 'better than' B and B is 'better than' C, then A is 'better than' C. Such reasoning is consistent with certain models of rational behavior. Nevertheless, violations of transitivity have been frequently observed in human behavior, such as game playing (e.g. playing rock-paper-scissors) or in nature, such as competition between bacteria (Kerr et al., 2002; Károlyi et al., 2005; Laird and Schamp, 2006) or the mating choice of lizards (Sinervo and Lively, 1996). See Allesina and Levine (2011) for a theory on how such intransitive relations can hypothetically arise for species interactions when there are multiple limiting factors. Many systems have both transitive and intransitive properties. For example, when predicting which Pokémon will win a battle, there is a clear transitive component. Some Pokémon are inherently stronger than others (Mewtwo vs. Weedle), though there is a complex intransitive component in the system of strengths and weaknesses of types of the Pokémon (illustrated in Figure 2.1). The general framework discussed in this work can learn arbitrary relations which may violate transitivity.

Transitivity can be established in a pairwise prediction setting via the notion of ranking representability, as defined below.

Definition 2.9 (Ranking representability). *A reciprocal or symmetric relation $Q : \mathcal{U}^2 \to [0, 1]$ is ranking representable if there exists a ranking function $h : \mathcal{U} \to \mathbb{R}$ such that for all $(u, u') \in \mathcal{U}^2$ it respectively holds that*

1. $Q(u, u') = \nabla(h(u) - h(u'))$ *(reciprocal case);*
2. $Q(u, u') = \nabla(h(u) + h(u'))$ *(symmetric case).*

The idea behind ranking representability is that there is some latent utility function $h(\cdot)$ from which the relations are constructed. This concept is best explained by means of an example. For the reciprocal case, consider admitting preference for one item over another.

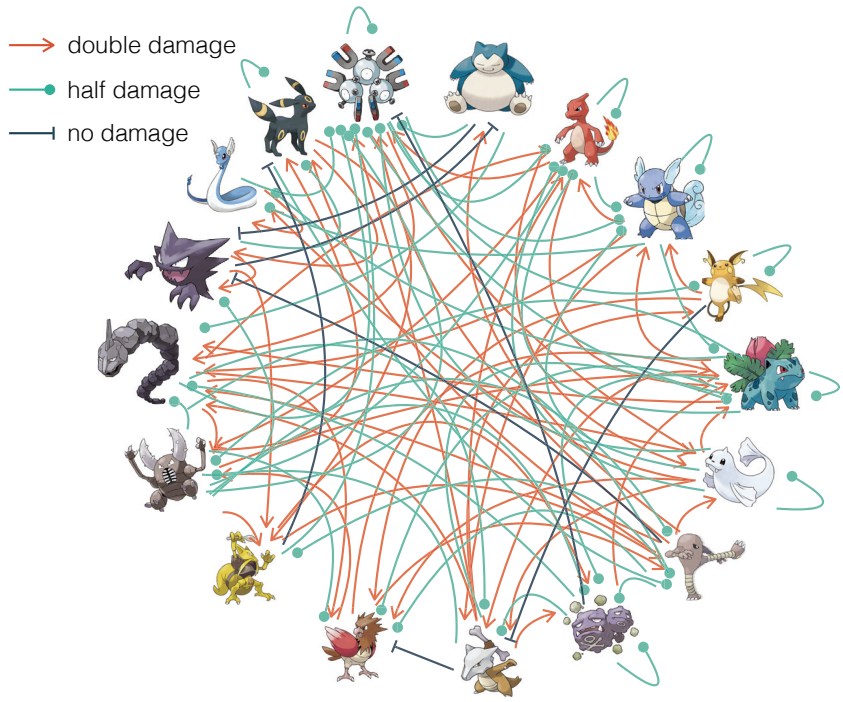


Figure 2.1: Chart of the different strengths and weaknesses of the Pokémon types of generation I and II. The relation modelling the probability of one Pokémon winning against an other Pokémon is both an example of a reciprocal relation as well as of intransitivity. A challenging pairwise prediction problem would be to try to predict which of a pair of Pokémon is likely to win a battle based on the level, type and move set.

A *homo economicus* can give a value (in dollars for example) to each item and his preference for one item over the other is determined by the difference in value[4]. For the symmetric case, consider the artificial example when one would model the quality of a relation between two spouses as being only determined by their joint income. The more money each person makes, the better the relationship. A cynical dat-

[4]Despite this being a concept that forms the basis of much of economics as a discipline, it is generally accepted that this is seldom the case in reality. Humans are, on the whole, not rational (Kahneman, 2011).

ingsite might incorporate this notion in its algorithms for matching people.

The notion of ranking representability is quite interesting for reciprocal relations, but is somewhat artificial for symmetric relations. A more relevant property is Euclidean representability, which states that the relation can be constructed from an underlying Euclidian distance measure. This is defined below.

DEFINITION 2.10 (Euclidean representability). *A symmetric relationship $Q : \mathcal{U}^2 \to [0, 1]$ is Euclidean representable if there exists a function $\mathbf{h} : \mathcal{U} \to \mathbb{R}^z$, with $z \in \mathbb{N}$, such that for all $(u, u') \in \mathcal{U}^2$ it holds that*

$$Q(u_1, u_2) = \nabla((\mathbf{h}(u) - \mathbf{h}(u'))^\top(\mathbf{h}(u) - \mathbf{h}(u'))) .$$

This concept has been used successfully for inferring protein-protein networks (Vert et al., 2007). Here, the idea is that when two proteins have, for example, a similar expression profile, they are more likely to show a physical interaction. By restricting themselves to Euclidean representable relations, rather than the more general case of symmetric relations, the authors recorded a substantial improvement in performance.

Figure 2.2 gives an overview of some of the concepts regarding relations that were discussed in this section. The general method for pairwise learning (and hence also for relational learning) can be applied to learn arbitrary relations. If there is prior knowledge available on the nature of the relation to be learned, one can incorporate this information in the prediction function. Since this implies that the learning involves searching for an optimal function in a much smaller space (only the functions that satisfy the relevant properties), standard learning theory dictates that the learned function should be an improvement compared to the one obtained without using this prior knowledge (Vapnik, 2000). In Section 4.5.4, I will provide the practi-

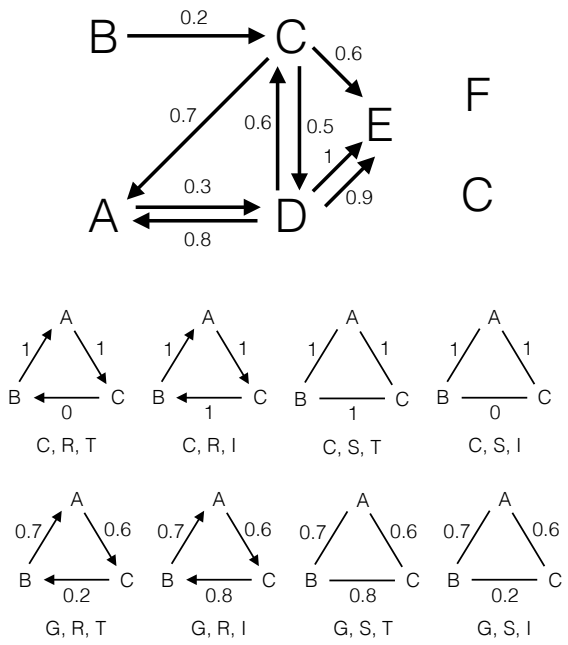


Figure 2.2: (top) Example of a multi-graph representing the most general case, where no additional properties of relations are assumed. (bottom) Examples of eight different types of relations in a graph of cardinality three. The following relational properties are illustrated: (C) crisp, (G) graded, (R) reciprocal, (S) symmetric, (T) transitive and (I) intransitive. For the reciprocal relations, (I) refers to a relation that does not satisfy weak stochastic transitivity, while (T) is showing an example of a relation fulfilling strong stochastic transitivity. Image after Waegeman et al. (2012).

cal tools for learning a prediction function with any of the properties defined above.

2.2.2 Dyadic data

If pairwise learning corresponds to predicting properties of edges in a graph, then dyadic data boils down to predicting properties of the edges of a bipartite graph. I will briefly discuss some potential properties of pairwise functions.

An important property of a pairwise function is bilinearity:

DEFINITION 2.11 (Bilinearity). *A pairwise function f is bilinear with rank R if there exist two vector-valued functions $\mathbf{p} : \mathcal{U} \to \mathbb{R}^R$ and $\mathbf{q} : \mathcal{V} \to \mathbb{R}^R$ such that for all $(u, v) \in \mathcal{U} \times \mathcal{V}$ it holds that*

$$f(u, v) = \mathbf{p}(u)^\top \mathbf{q}(v) \,.$$

Most of the prediction functions that will be discussed in the later chapters are bilinear. Note that as the rank increases, the flexibility of this representation also increases. Using kernels, it will be possible to work with bilinear functions with an arbitrary large or even infinite rank. In Section 4.5.2 it will be proven that it is indeed possible to represent any possible function using this framework.

When the rank R is small, the data is denoted as *low-rank*. This implies that the complete label matrix can be approximated as:

$$\mathbf{Y} \approx \mathbf{PQ} \,,$$

with $\mathbf{P}$ a 'long' (many rows and few columns) and $\mathbf{Q}$ a 'wide' (few rows and many columns) matrix. Splitting a label matrix into two low-rank matrices is called matrix factorization and is a large subfield in machine learning (Barber, 2011). Many algorithms for machine learning exist, including those that enforce the values in the low-rank matrices to be non-negative (Lee and Seung, 1999), sparse (Zou et al., 2006) or 'independent' (MacKay, 2003). A tentative list of matrix factorization algorithms can be found on The Advanced Matrix Factorization Jungle[5]. For an illustration of a practical use of matrix factorization to explore and predict bacterial interactions, see my work: Stock et al., (2013).

Another property of the prediction function is when the two objects act independently. I will denote this as additivity.

[5] `https://sites.google.com/site/igorcarron2/matrixfactorizations`

DEFINITION 2.12 (Additivity). *A pairwise function f is called additive if there exist two functions $f_1 : \mathcal{U} \to \mathbb{R}$ and $f_2 : \mathcal{V} \to \mathbb{R}$ such that for all $(u, v) \in \mathcal{U} \times \mathcal{V}$ it holds that*

$$f(u, v) = f_1(u) + f_2(v) \,.$$

This can be seen as a special case of bilinearity with rank two, by considering

$$\mathbf{p}(u) = \begin{bmatrix} f_1(u) \\ 1 \end{bmatrix} \qquad \mathbf{q}(v) = \begin{bmatrix} 1 \\ f_2(v) \end{bmatrix} . \tag{2.2}$$

This property is not particularly desirable or interesting within the context of this PhD thesis, as u and v can be treated completely independently. In other words, there is a 'best' object u, unrelated to object v. An example where such a setting would arise is in the so-called neutral theory in ecology where the probability of species interactions is determined by the abundances and mobility of the individual species (Blüthgen et al., 2006; Poisot et al., 2015). Additive pairwise models have been applied to protein-ligand interaction predictions and have shown competitive performance (Bleakley and Yamanishi, 2009; Xia et al., 2010; van Laarhoven et al., 2011).

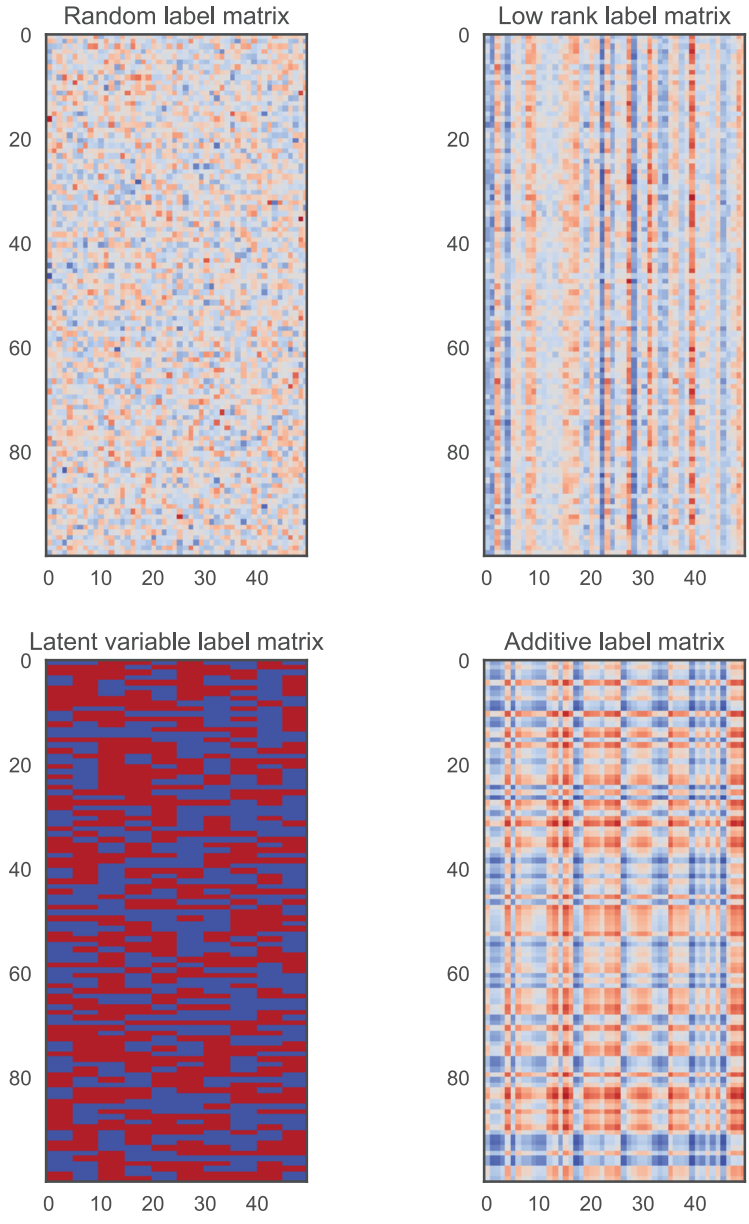


Figure 2.3: Four examples of matrices with a different structure. (top left) A matrix with independently, identically distributed normal values. (top right) A low-rank matrix with a rank of ten. (bottom left) A matrix with for all rows, every consecutive ten values are all either zero or one, with equal probability. (bottom right) A matrix with additive data.

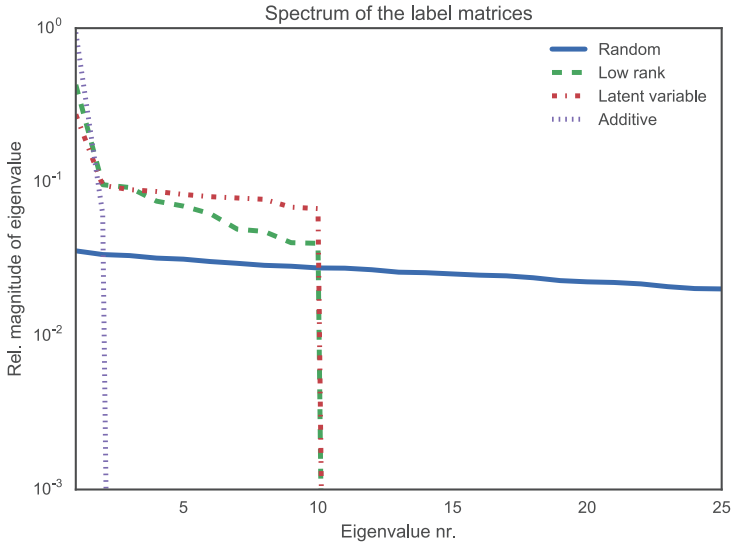


Figure 2.4: The normalized eigenvalue spectra for the four matrices depicted in Figure 2.3.

Figure 2.3 gives an example of a few datasets of variable complexity. The first dataset is a matrix with independently, identically distributed normal values. The second matrix is low-rank with a rank of ten. The third matrix conveys a different pattern: for all rows, every consecutive ten values are all either zero or one with equal probability. The final matrix is an example where the data is additive. An instructive way of studying the information in such matrices is by plotting the eigenvalue spectrum obtained by singular value decomposition. Figure 2.4 shows the normalized eigenvalue spectrum for the four depicted matrices. For the random data, every eigenvalue has an approximately similar magnitude, indicating that there is no low-dimensional representation of this data that retains most of the information. For the other matrices, it is clear that the real complexity of the matrix is much lower, as most of the eigenvalues are equal to zero.

2.3 Using and evaluating pairwise models

2.3.1 The conditional ranking problem

Using a dataset S, we can train a pairwise function $f : \mathcal{U} \times \mathcal{V} \to \mathbb{R}$ to score new dyads. For any practical application it is important to correctly assess how reliable these scorings are. However, in order to provide a meaningful performance metric, it should first be defined how the model will be used.

In this work I will mainly consider applications situated in the life sciences. Typically in those settings, high-quality data is scarce and the function to be learned from a dataset is extremely complex. For example, consider the problem of predicting protein-protein interactions. Proteins interact through a complex interplay of amino acid residues (Aloy and Russell, 2006) and modelling such an interaction mechanistically would require a tremendous amount of quantum physics simulations.

We cannot hope to learn to accurately predict such interactions, unless we can combine very expressive models with unrealistically huge amounts of data. Luckily, we do not need a perfectly-accurate model to be useful in practice. If our model can find general trends in the data that determine the interactions, such information can be used to design experiments. Consider the following scenarios:

- A pharmaceutical company has designed a new drug and wants to know to which proteins it is likely to show affinity to which drug. Using a predictive model, they can select the proteins that are most likely to bind and test these in the lab.

- An ecologist has collected a network containing plant-ant interactions, the model can be used to provide a limited list of plants

to study in detail to find interactions with a certain ant species of interest.

- A large book seller website maintains a database of their users and catalog. When visiting their website you see a recommendation of books you might enjoy.

In all of the above examples, the model aids the experiments but does not replace it. Errors are tolerable, as long as the predictions of the model tend to be (preferably much) better than random.

To put this a bit more formally, through modelling and experimentation, one wants to solve a problem of the form:

$$u^* = \arg\max_{u \in \mathcal{U}} f(u, v)\,. \tag{2.3}$$

Here we denote v as the *query* object, we search for an optimal u^* that matches it as well as possible. Note that the meaning of u and v can be switched depending on the application. Equation (2.3) is known as solving the inverse problem in structured output prediction (Bakir et al., 2007). This is typically a difficult problem to solve, as it is usually an optimization problem over a huge structured space. Depending on the form of the prediction function and structure of object spaces, efficient exact (e.g. Ricci et al. (2008)) or heuristic (e.g. Daumé et al. (2009)) algorithms can be constructed. Rather than searching for the optimal u^* in its complete object space, we will mainly try to solve the more modest problem of searching in a finite subset $U' \subseteq \mathcal{U}$:

$$u^* = \arg\max_{u \in U'} f(u, v)\,. \tag{2.4}$$

Depending on the application U' can be equal to the object set U used for training, it can be a superset or an independent set. We will denote U' as the *database*.

Problem (2.4) will be solved by a *ranking* approach, all objects in the database will be scored conditional on the query and ranked

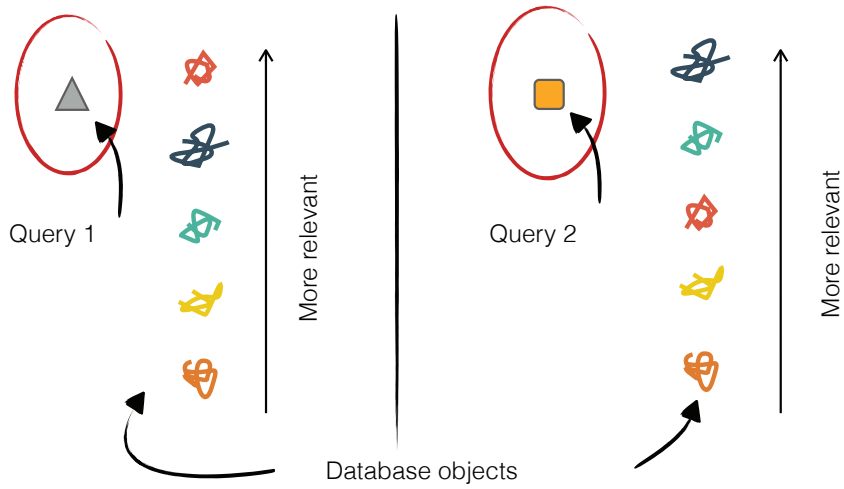


Figure 2.5: Example of the use of a pairwise prediction function. A ligand is used as a query to rank a database of proteins according to the predicted affinity. A different ligand can induce a completely different ranking.

accordingly. Given two objects from the database $u, u' \in \mathcal{U}^2$ and an object $v \in \mathcal{V}$ serving as a query, the ranking of u and u', conditioned on v, is constructed as:

$$u \succeq_v^f u' \Leftrightarrow f(u, v) \geq f(u', v)\,.$$

We will denote this as *conditional ranking*, as a ranking of a set of objects U' is induced *conditional* on a query v using a pairwise prediction function $f(\cdot, \cdot)$. This is shown graphically in Figure 2.5. Either the highest-scoring database object or a set of top-scoring objects are retained.

Now that the practical use of pairwise prediction has been clarified, it is possible to define what is to be considered a good model. *We want to find a pairwise model that tends to give higher scores to more relevant new dyads compared to less relevant new dyads.* In the follow-

ing subsections we will discuss 1) what new dyads mean in pairwise learning and 2) how the performance can be quantified in a ranking setting.

2.3.2 Four settings for prediction

Here, I make an important distinction between several types of pairwise learning settings. Later, in Section 4.3, I provide a literature survey of related machine learning methods that fit in this paradigm.

In traditional supervised machine learning, the model is trained using the training set and its generalization performance is assessed using an independent validation set (Hastie et al., 2001). This is important for both *model selection* (choosing the appropriate method, hyperparameters and structural properties such as kernels or network topology in artificial neural networks) as well as *model validation* (getting a realistic estimate of how well the model performs on new data). To this end, one would actually require three datasets: a training set, a validation set and a test set[6]. If the available dataset S is large, it can be divided in three separate sets, otherwise one has to resort to K-fold cross-validation.

When dealing with pairwise data, it becomes quite tricky to define what 'new data' is. Consider the pairwise dataset of Figure 2.6. The dyad (2, A) is clearly a new dyad as we have no observed label in the training set, even though we have labeled dyads containing both the object 2 and object A, but never together. Intuitively, one feels that this is likely to be relatively easy, as it is only a problem of imputing a missing value in a matrix. Making predictions for the dyads (5, B) or (3, D) seems more challenging, as one of the objects of the dyad has not been encountered in the training set. The hardest case is when both objects in the dyad are new, e.g. (5, D).

[6]It is interesting to remark that a Bayesian approach can, in principle, circumvent the need for a validation set by choosing the hyperparameters to maximize the posterior likelihood (MacKay, 2003; Bishop, 2006; Barber, 2011). In frequentist settings, this would result in overfitting.

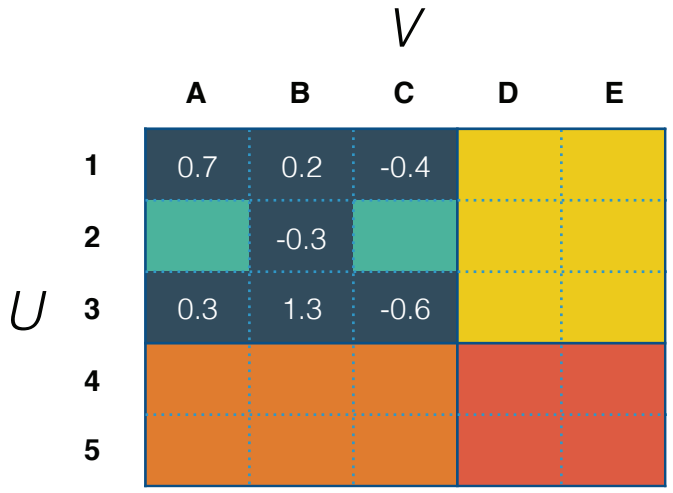


Figure 2.6: Toy pairwise dataset with seven labeled dyads.

Formally, four settings for prediction can be distinguished in pairwise learning:

- **Setting A**: Both u and v are observed during training, as parts of different dyads, but the label of the dyad (u, v) must be predicted;
- **Setting B**: Only v is known during training, while u is not observed in any dyad, and the label of the dyad (u, v) must be predicted;
- **Setting C**: Only u is known during training, while v is not observed in any dyad, and the label of the dyad (u, v) must be predicted;
- **Setting D**: Neither u nor v occur in any training dyad, and the label of the dyad (u, v) must be predicted.

These settings are represented graphically in Figure 2.7. In pairwise learning it is extremely important to implement appropriate

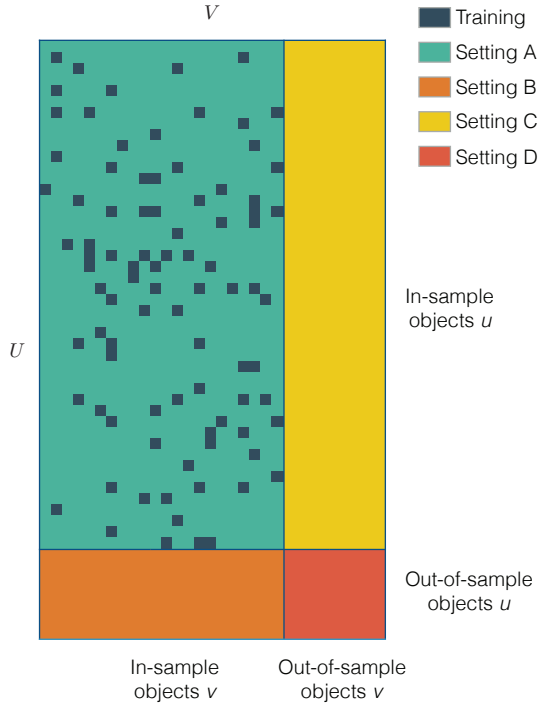


Figure 2.7: An overview of different prediction settings in pairwise learning. When the dataset is complete, the block of Setting A is completely available in the training set.

training and evaluation procedures. Without notions about these settings, one might select the wrong model for the given task or obtain an under- or overestimation of the generalization error. In a large-scale meta-study about biological network identification, it was found that these concepts are vital to correctly evaluate pairwise learning models (Park and Marcotte, 2012). For these reasons, it is advised to always give the performance for the four settings.

To correctly assess the generalization error, I propose using suitable cross-validation schemes. To perform K-fold cross-validation for Setting A, it usually suffices to randomly divide the n dyads in the training set into K disjoint sets. Subsequently, $K-1$ folds are used

for training the model, while the remaining fold is used to compute the performance of the model. This is repeated for each of the folds, so that we obtain K estimates of the performance. A similar scheme can be applied for Setting B (resp. C), by distributing the objects in U (resp. V) in K disjoint sets. Then, the dyads of the test set are allocated according to the folds containing one of their objects. Using these folds, the obtained performance will reflect the capacity of the model to make predictions for new objects of one kind.

For Setting D, performing cross-validation is the most complex. A practical recipe for K-fold cross-validation for this setting is by distributing both the objects in the sets U and V each in K disjoint subsets $U_1, U_2, \ldots, U_K$ and $V_1, V_2, \ldots, V_K$. For cross-validation fold i, all dyads in the dataset that occur in $U_i \times V_i$ are withheld for validation, while the model is trained using the remainder of the training set with all dyads containing objects in U_i or V_i discarded. Note that by using such a scheme, a substantial part of the training data is neither used for training, nor for validation! For K-fold cross-validation, a fraction of $(K-1)^2/K^2$ of the training set will be used for training, a fraction of $1/K^2$ for validation and a fraction of $2K(K-1)/K^2$ will be discarded in each fold, as illustrated in Figure 2.8.

In this work, we often use leave-one-out (LOO) cross-validation, where each 'data point' is withheld once for validation, while the model is trained on the remainder of the dataset. For the different prediction settings in pairwise learning, this boils down to:

- **Setting A**: For every $(i,j) \in \{1,\ldots,m\} \times \{1,\ldots,q\}$, leave out $S_{ij}^{\text{LOO, A}} = \{(u_i, v_j, \mathbf{Y}_{ij})\}$ for testing, while the model is trained on $S_{ij}^{\text{Train, A}} = \{(u_k, v_l, \mathbf{Y}_{kl}) \mid k \in \{1,\ldots,m\}, l \in \{1,\ldots,q\}, (k,l) \neq (i,j)\}$.
- **Setting B**: For every $i \in \{1,\ldots,m\}$, leave out $S_i^{\text{LOO, B}} = \{(u_i, v_j, \mathbf{Y}_{ij}) \mid j \in \{1,\ldots,q\}\}$ for testing, while the model is trained on $S_i^{\text{Train, B}} = \{(u_k, v_j, \mathbf{Y}_{kj}) \mid k \in \{1,\ldots,q\}, j \in \{1,\ldots,q\}, k \neq i\}$.

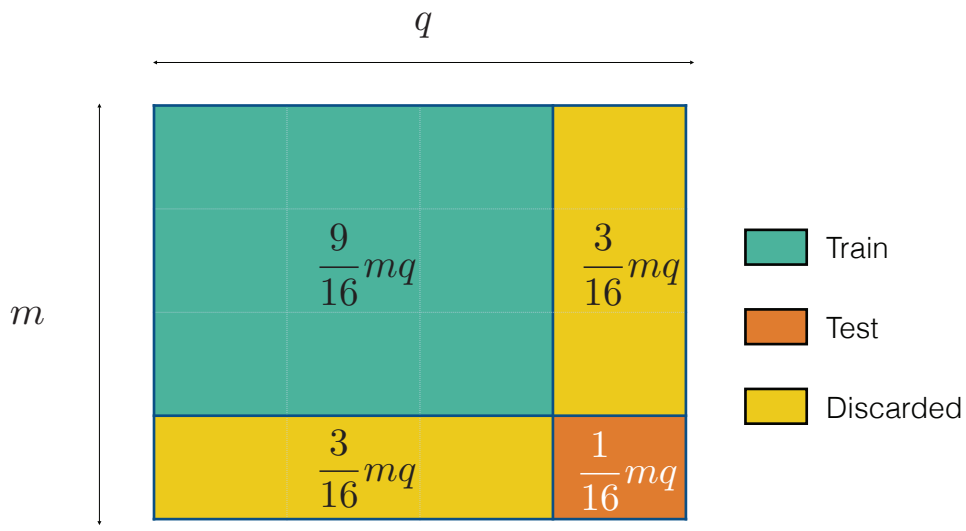


Figure 2.8: Example of taking a fold for four-fold cross-validation for Setting D on a complete dataset. For each fold, a substantial amount of data has to be discarded.

- **Setting C**: For every $j \in \{1, \ldots, q\}$, leave out $S_j^{\text{LOO, C}} = \{(u_i, v_j, \mathbf{Y}_{ij}) \mid i \in \{1, \ldots, m\}\}$ for testing, while the model is trained on $S_j^{\text{Train, C}} = \{(u_i, v_l, \mathbf{Y}_{il}) \mid i \in \{1, \ldots, m\}, l \in \{1, \ldots, q\}, l \neq j\}$.

- **Setting D**: For every $(i, j) \in \{1, \ldots, m\} \times \{1, \ldots, q\}$, leave out $S_{ij}^{\text{LOO, D}} = \{(u_i, v_j, \mathbf{Y}_{ij})\}$ for testing, while the model is trained on $S_{ij}^{\text{Train, D}} = \{(u_k, v_l, \mathbf{Y}_{kl}) \mid k \in \{1, \ldots, m\}, l \in \{1, \ldots, q\}, k \neq i, l \neq j\}$.

These different LOO cross-validation settings for a complete dataset are depicted in Figure 2.9. It can be shown that LOO cross-validation gives a nearly unbiased estimate of the performance, at the cost of a high variance (Hastie et al., 2001). This is because the models trained for the different folds are constructed using very similar datasets. Another potential bottleneck is that LOO cross-validation is often

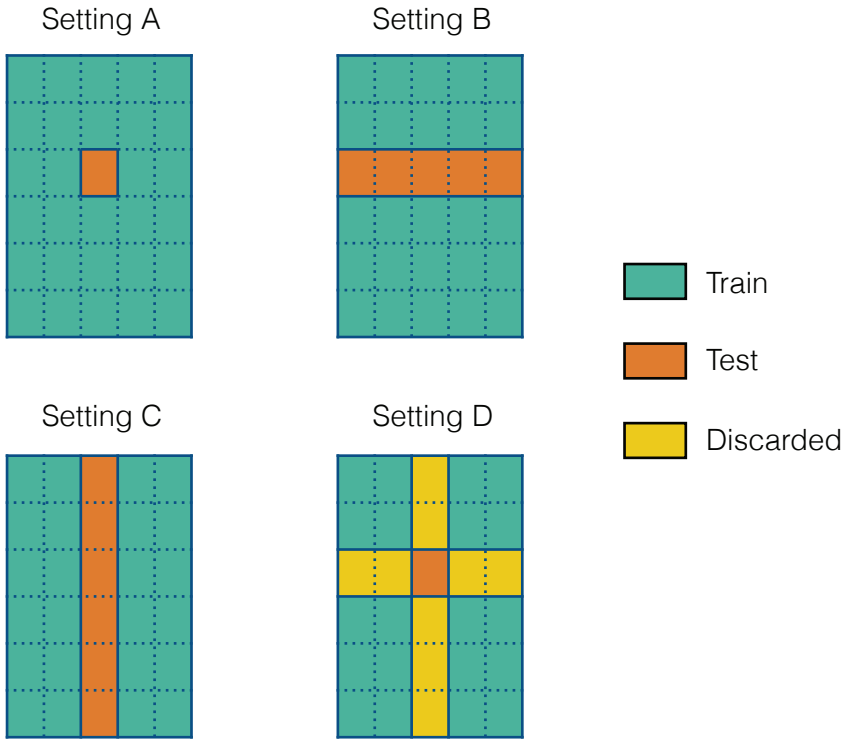


Figure 2.9: Leave-one-out cross-validation for pairwise data. See text for details.

computationally expensive because as many models as there are data points have to be trained (i.e. n, m or q models depending on the setting).

2.3.3 Performance metrics for pairwise problems

In the previous section, I discussed the concept of 'new data' in a pairwise context. Here, I shall discuss how to evaluate the performance of the model for these new dyads. The performance is computed based on a set of n' labels $y_1, \ldots, y_{n'}$ and a set of n' corresponding predictions $f_1, \ldots, f_{n'}$. Most performance metrics used in

traditional machine learning settings are calculated on such an 'unstructured dataset' (i.e. assuming that the instances are independently and identically distributed). In pairwise prediction settings, however, there are sometimes different ways for calculating the performance metric. This is best illustrated on the special case when working with a complete dataset. Here we have the matrix with the labels $\mathbf{Y} = [Y_{ij}]$ and the matrix with the corresponding predictions $\mathbf{F} = [F_{ij}]$. Borrowing some terminology from multi-label classification, three ways for calculating the performance can be distinguished, as illustrated in Figure 2.10:

- **Micro-wise**: the performance metric is calculated over the whole matrix (i.e. the matrix is flattened).
- **Instance-wise**: the performance is calculated for each row separately and averaged.
- **Macro-wise**: the performance is calculated for each column separately and averaged.

Similar to the four different prediction settings discussed in the previous section, the ways of calculating performance matter to obtain a realistic assessment for the application of interest. This is especially relevant in ranking applications. Consider an example in protein-ligand interaction prediction with the ligands (resp. proteins) corresponding to the rows (resp. columns) of the label matrix. The micro-wise performance will quantify how well the model can identify strongly interacting protein-ligand pairs, while the instance-wise metric assesses whether the model can distinguish between ligands with high or low affinity for a given protein. Similarly, the macro-wise metric assesses whether the model can distinguish between proteins with high or low affinity for a given ligand.

In regression settings one usually wants the predicted labels to be close on average to the observed labels. The most widely used

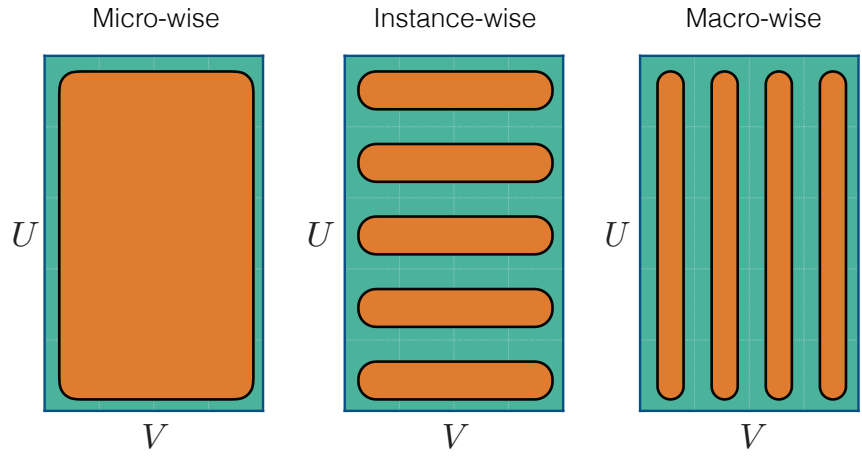


Figure 2.10: The three ways of calculating performance in a complete pairwise dataset. Micro-wise: the performance metric is calculated over the whole matrix, Instance-wise: the performance is calculated for each row separately and averaged, Macro-wise: the performance is calculated for each column separately and averaged.

performance metric for regression is the mean squared error (MSE), calculated as:

$$\sum_{h=1}^{n'} (f(u_h, v_h) - y_h)^2 .$$

Taking the square root of the MSE results in the root mean squared error (RMSE), ensuring that the metric is on the same scale as the labels. The MSE (resp. RMSE) can be compared to the variance (resp. standard deviation) of the labels. The latter corresponds to the baseline performance of using the mean for prediction. This rationale is incorporated in the fraction of explained variance (R^2), which is calculated as one minus the MSE divided by the variance of the labels. Values close to one indicate a very good fit of the model.

As explained above, we will often use our models in a ranking framework. To this end, dyads that are more relevant (have a higher

label) should be higher ranked using a model than dyads that are less relevant. This is measured by the concordance index (C-index) (Gönen and Heller, 2005), also known as the pairwise ranking accuracy:

$$\frac{1}{|\{(k,l) \mid y_k > y_l\}|} \sum_{y_k > y_l} H(f_k - f_l)\,, \tag{2.5}$$

where $H(\cdot)$ denotes the Heaviside step function which returns one if the input is positive, zero if the input is negative and $1/2$ if the input is equal to zero. The C-index is the probability that when randomly selecting two dyads with a different label, the model will give a higher score to the dyad with the highest label. Hence, the C-index is particularly suited to measure how well the model is suited for ranking purposes.

When dealing with binary classification problems, the goal will be to discriminate between a relevant and an irrelevant class, which we shall denote as the positive and negative class, respectively. Because the prediction function returns a real value, we have to apply a threshold to perform a classification based on the returned output. If the predicted score for the dyad is larger than a threshold t, it will be classified as positive, else it will be classified as negative. Positive dyads that are classified as positive are denoted as true positives (TP), negative dyads that are wrongly classified as positives are false positives (FP). Similarly, positive dyads classified as negative are false negatives (FP), while negative dyads classified as negative are true negatives (TP) This is summarized in the table below and in Figure 2.11.

	Predicted positive	Predicted negative
Label positive	TP(t)	FN(t)
Label negative	FP(t)	TN(t)

Note the explicit dependence on the threshold t. This is because there is a trade-off: higher thresholds will result in a more stringent

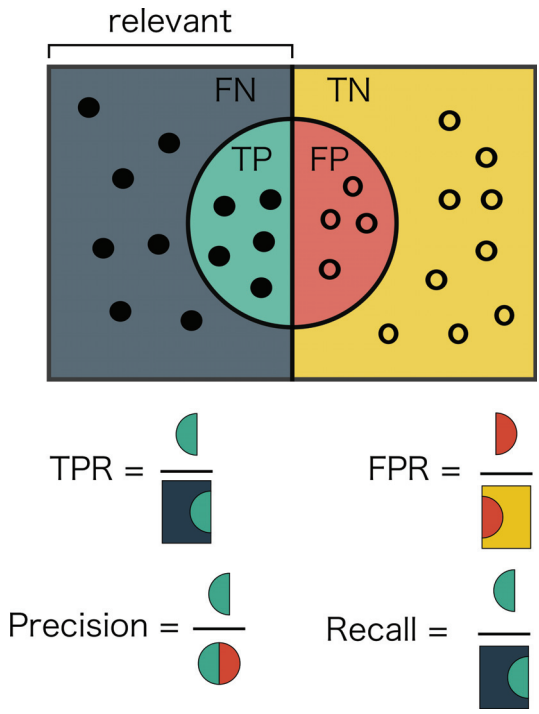


Figure 2.11: Using a classifier to determine whether instances are relevant or not. In reality it is not known whether an instance is truly relevant. Lowering the threshold corresponds in this figure to increasing the radius of the circle: more instances will be retained.

selection, at the cost of potentially not detecting some positive instances. This dilemma is captured by the true positive rate (TPR) and the false positive rate (FPR):

$$\mathrm{TPR}(t) = \frac{\mathrm{TP}(t)}{\mathrm{TP}(t) + \mathrm{FN}(t)} \qquad \mathrm{FPR}(t) = \frac{\mathrm{FP}(t)}{\mathrm{TN}(t) + \mathrm{FP}(t)}\,.$$

The true positive rate is the fraction of the positive dyads that have been detected, while the false positive rate is the fraction of negative dyads that have been wrongly deemed relevant. It is preferable to have a true positive rate that is as large as possible, while having a low false positive rate. This trade-off is studied using a receiver operating

characteristic (ROC) curve, which plots the true positive rate as a function of the false positive rate for varying thresholds (Fawcett, 2004). This is depicted in Figure 2.12 for a simple binary classification problem.

The ROC curve is often summarized by the area under the ROC curve (AUC). Similar to the C-index, which is actually an extension of the AUC, the AUC can be interpreted as the probability that a randomly chosen positive instance receives a higher score than a randomly chosen negative instance. A strong merit of this performance metric is that it is insensitive to the distribution of positives or negatives in the data. Because the AUC can be used for skewed datasets with a small fraction of positive instances, it is very popular in domains such as bioinformatics (Lasko et al., 2005; Schrynemackers et al., 2013)

For some applications, precision and recall are more relevant. They are calculated as

$$\text{precision}(t) = \frac{\text{TP}(t)}{\text{TP}(t) + \text{FP}(t)} \qquad \text{recall}(t) = \frac{\text{TP}(t)}{\text{TP}(t) + \text{FN}(t)} \,.$$

Note that recall is a different name for the TPR. A user of a recommendation system is mainly interested in precision, the fraction of returned instances that are really relevant to her. This is contrasted to the recall which measures the fraction of relevant documents that were detected. Again there is an obvious trade-off between these two metrics, which can be analyzed using a precision-recall curve (illustrated in the bottom plot of Figure 2.12). In contrast to ROC curves, precision-recall curves are sensitive to the class distribution. Precision-recall curves can likewise be summarized by the area under the curve.

For many practical applications, one is only interested in the quality of the top-scoring objects. Metrics such as the AUC treat every part of the ranking as equally important. This can sometimes be misleading; if the test set contains some easy to detect negative ex-

amples, this can lead to a high AUC while the classifier might be bad at detecting positive instances. Such behaviour can be mediated by, for example, putting more emphasis on the top of the ranking in the ROC curve, e.g. Swamidass et al. (2010). In information retrieval, many performance metrics have been designed to specifically evaluate the quality of the top-scored items, including the F-measure, precision-at-k, average precision and discounted cumulative gain.

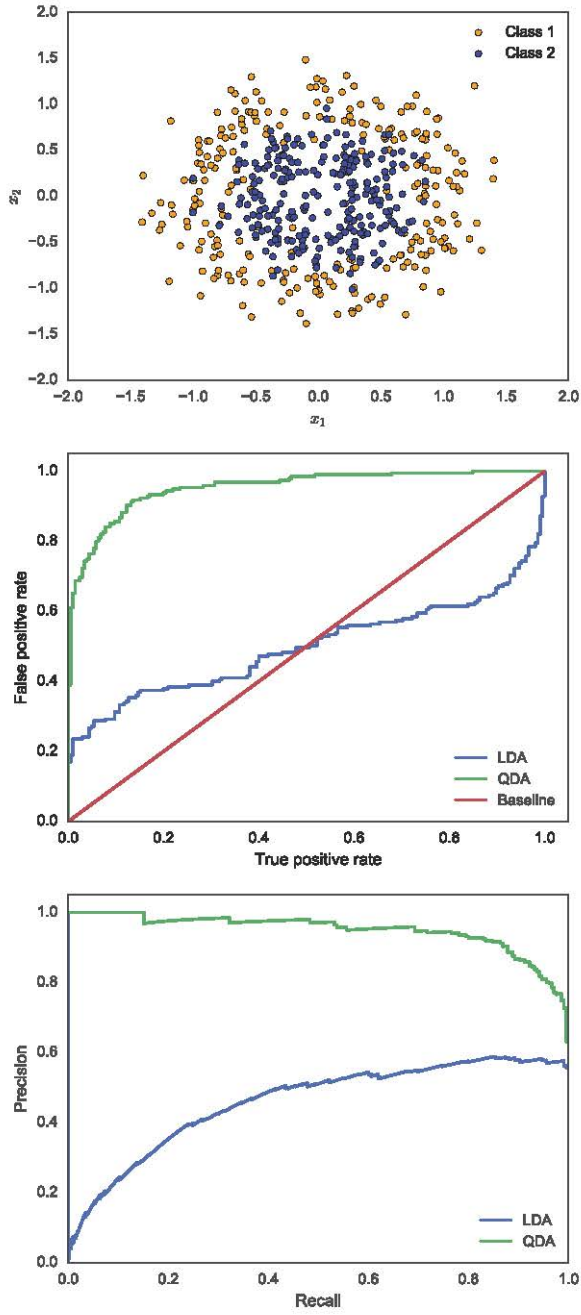


Figure 2.12: Example of a two-dimensional binary classification problem, solved using linear (LDA) and quadratic discriminant analysis (QDA). (top) Scatter plot of the dataset. (middle) The corresponding ROC curve for the two classifiers. (bottom) The precision-recall curve for the two classifiers.

2.4 Motivating examples at different scales

The broadness of pairwise learning, at least as defined in this work, makes that many people have been studying the same and related settings for a long time. Indeed, the kernel-based framework, which is central to this work, was first proposed for inferring protein-protein interactions (Vert et al., 2007), while the notion of retrieving documents based on a query originates from at least the 1950s (Singhal, 2001). It would be impossible to review these huge and rich research domains in a couple of pages while being anything close to being comprehensive. Instead, I will try to give a personal overview of these fields, with emphasis on the big ideas and issues, illustrated with some relevant references.

At the end of this section it should be clear that these domains can naturally be cast in the light of pairwise learning. Later chapters will feature many applications from these domains and will discuss how to resolve some of the raised issues presented in the different domains.

2.4.1 In the cell: molecular interactions

Life at the molecular level is mediated by the interaction of a variety of components, including genes, RNA, proteins, small molecules etc. Systems biology is concerned with how these parts fit together to create a living entity. Many analytical techniques have been developed for large-scale screening of thousands to millions of interactions – see Macarron et al. (2011) for a discussion. These screenings are often unreliable, for example a meta-study has found that only in about 20% of the cases protein-ligand interaction screenings are consistent (Prinz et al., 2011). The optimistic data scientist views this as an opportunity, as it motivates the need for powerful learning algorithms to detect these interactions in noisy data. In the last decades,

many algorithms have been developed for biological network inference, most of them based on machine learning. In this section I give a brief overview of the main ideas used in these algorithms.

Again, I make the distinction between monadic and dyadic interactions:

- **Monadic molecular interactions** include protein-protein interactions, gene regulatory interactions, the links between metabolites in metabolic pathways or even interactions of residues within a protein. Some of these interactions, such as protein-protein interactions, are symmetric, while some, such as the direction of metabolic fluxes, are anti-symmetric.

- **Dyadic molecular interactions** include protein-ligand interactions, mRNA-miRNA interactions, protein-DNA interactions etc.

Machine learning methods to infer molecular interactions can be subdivided into two large classes: unsupervised and supervised methods.

Unsupervised methods, also denoted as *de novo* inference (Vert, 2008), do not use a dataset of known interacting components. Here, only data from the individual components such as expression, phylogeny, sequence, 3D structure, etc. is available. For example, gene regulatory networks can directly be inferred from gene expression data, e.g. (Friedman et al., 2000; Huynh-Thu et al., 2010; Marbach et al., 2012; Stein et al., 2015). Other strategies for unsupervised network inference include looking for similarities in data, for example co-evolution (Pazos and Valencia, 2001) or patterns in sequence or structure (Marcotte, 1999; Pazos and Valencia, 2002; Pitre et al., 2008). On the whole, unsupervised methods are consideren not to be as powerful as supervised methods for biological network inference (Maetschke et al., 2014).

Supervised methods for biological network inference require a so-called gold standard network containing examples of interacting molecules. Here, one usually trains a classifier to distinguish between interacting versus non-interacting pairs using features of the component. The features are typically sequence, expression data, 3D structure, phylogenic profiles, etc.

There are two paradigms for supervised network inference: the global and the local approach. The global approach, shown left in Figure 2.13, builds a single pairwise model based on the complete dataset of available interactions. The local approach, depicted right in Figure 2.13, divides the inference in several smaller classification problems. For each component, a classifier is trained to predict whether other components will interact with the component. None of these methods is generally accepted as the best, though the global approach is, in theory, both more powerful and more computationally demanding, since the interaction network is processed in one step. By now it should be clear that this PhD thesis heavily favours the global approach, though in Section 4.4 I briefly discuss the connection with the local approach.

Many different machine learning methods have been applied for supervised network prediction, though kernel machines (e.g. (Yamanishi et al., 2004; Bock and Gough, 2001; Ben-Hur and Noble, 2005; Vert et al., 2007; Jacob et al., 2008; Chang et al., 2010; Bleakley and Yamanishi, 2009; van Laarhoven et al., 2011; Pahikkala et al., 2015)) and ensembles of trees (e.g. Geurts et al. (2007); Schrynemackers et al. (2015); Sun et al. (2016)) have been particularly popular. For some general reviews, see Vert (2008), Ding et al. (2013) and Schrynemackers et al. (2015).

2.4.2 In the environment: species interactions

Much like the molecular organisation of life, ecosystems are largely determined by the complex interplay of different organisms. No species

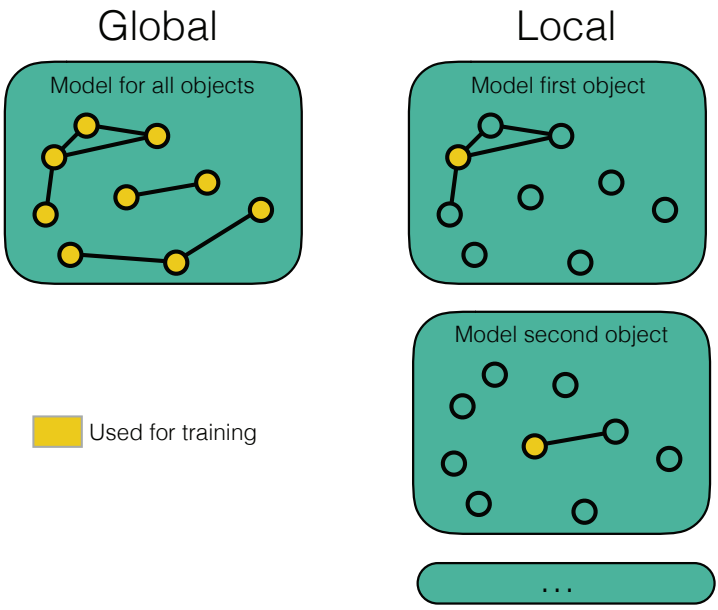


Figure 2.13: The global versus the local approach for supervised network inference. The global approach constructs a single pairwise model whereas the local approach constructs a model for each component.

exists in isolation and understanding the interactions between different species is a key step to comprehend an ecosystem as a whole.

This is more than an academic pursuit, such insight can lead to a rational management of an ecosystem, which lies at the core of bioscience engineering (Poisot et al., 2016). Knowledge of species interactions is commonly used in applications such as biological pest control (selecting natural predators for crop pests), epidemiology (removing transmission vectors) and waste water treatment (selecting the right culture of bacteria to remove the inorganic compounds). Furthermore, anthropogenic environmental destructions, including global warming and pollution, have greatly unhinged the global ecosystem. The large-scale extinction and migration of different species will pose

great challenges for both nature and man. For example, bees, which are responsible for the pollination of roughly 75% of all plant species used for human consumption, have been in sharp decline in the last years (Kluser et al., 2010). In order to bring back balance between man and nature, we require a clear understanding of how nature's components fit together, if only to appreciate its importance and delicacy.

Species interactions are usually represented as a graph or network, where the vertices denote the species and the edges the interaction. For example, directed monadic graphs are often used to represent prey-predator interactions, while undirected dyadic graphs are used to represent plant-ant interactions. In some cases the interaction strength is also given as a weight of the edges, though interaction strength is quite hard to measure and quantify (Wootton and Emmerson, 2005; Berlow et al., 2004). A graph representation will always be an abstraction of reality as not only do species interactions vary in space and time (Poisot et al., 2015), three-way interactions also occur in practice (for example host-pathogen-vector interactions). Species interaction datasets are usually collected through observation studies in the field, often by monitoring certain species and observing its interacting partners. As a consequence, these datasets often contain many false negative interactions (Goldwasser and Roughgarden, 1997; Chacoff et al., 2012; Fründ et al., 2015). In Chapter 7, I will demonstrate how pairwise learning can be used to detect such false negatives.

Species interaction datasets are analysed by calculating statistics that quantify the stability, dynamics or other properties of the network, e.g. Fründ et al. (2015); Oliver et al. (2015) or by using them to construct a mechanistic model, e.g. Gao et al. (2016). For example, the eigenvalue spectrum gives a quantitative description of the nestedness of a species interaction network (Staniczenko et al., 2013). In a bachelor project, we studied how ideas from information theory could be used to analyze the properties of species interaction networks. We

found for example that the variance of information is a good indicator of stability of a network - see Dhiedt et al. (2016).

Biotic interactions have evolved through millions of years to make the species compatible at the anatomical, morphological, physiological and molecular level (Thompson, 1999; Hu et al., 2008; Shimizu et al., 2014; Hadfield et al., 2014). Since the agents interact through physical or chemical mechanisms, it stands to reason that if an appropriate set of species traits and a representable dataset of example interactions is available, a data-driven model should be able to predict these interactions. Such a trait-based approach has recently been advocated (Bartomeus, 2013; Bartomeus et al., 2015; Morales-Castilla et al., 2015; Bartomeus et al., 2016), and was used for example to predict the impact of invading plant species (Gibson et al., 2012). This framework using traits can conceptually be divided into several levels of ecological organization (Bartomeus et al., 2015):

1. **Habitat filtering**: traits determine which species can occur in a certain habitat, determining the probability that two species meet (e.g. influence of climate on vegetation).

2. **Interaction preference**: given species co-occurence, a match of traits determines if two species can interact (e.g. a pathogen is equipped with the molecular machinery to cause a disease in a plant).

3. **Interaction efficiency**: species traits influence the per capita efficiency and impact of an interaction (e.g. plant is very susceptible or tolerant to a certain pathogen).

4. **Emergent properties**: emergent properties inherent to structure of the interaction network will influence functioning and will provide feedback on community dynamics.

This is illustrated in Figure 2.14.

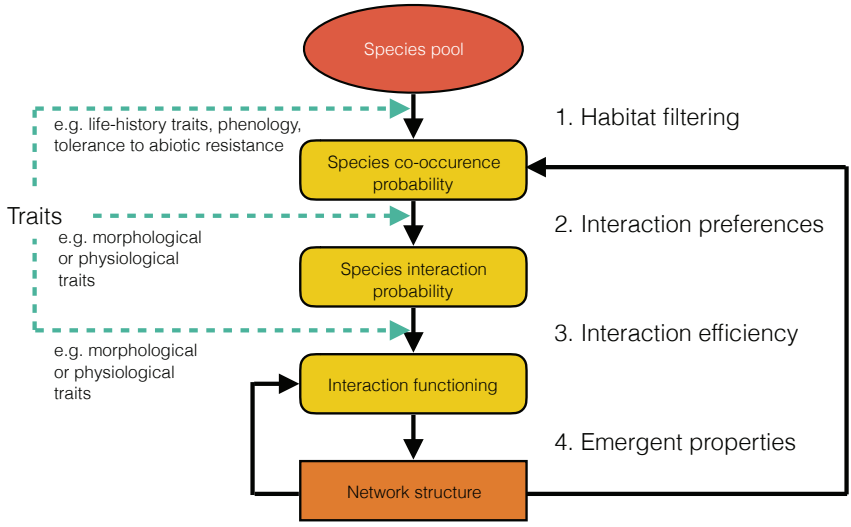


Figure 2.14: Different levels at which species traits influence the structure of interaction networks. See main text for details. Image after Bartomeus et al. (2015).

We consider plant-pollinator interactions as an illustration of a biotic process where traits can be used to predict interactions. Pollination is a very important mutualistic process where both partners benefit: the plant is able to spread its pollen, allowing efficient sexual reproduction and the pollinator receives nutrition in the form of pollen or nectar. Here the two agents have often undergone a long process of co-evolution to fine-tune the specificity and efficiency of the interaction (Hu et al., 2008; Shimizu et al., 2014). For example, bees, quite good at manipulating plant parts and good at seeing blue, white and ultraviolet, are typically attracted to appropriately colored flowers that require some dexterity to open. Butterflies, with their high energy needs and long tongues, go to brightly colored tubular flowers with a high nectar reward. Finally, moths, flying at night, pollinate white or pale flowers which often release strong fragrances. For an entertaining layman introduction to plant-insect interactions, see Goulson (2016).

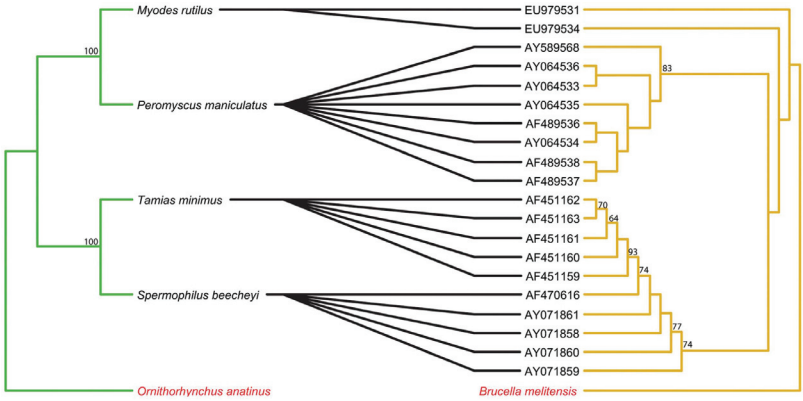


Figure 2.15: Tanglegram of cophylogenetic relationships between New World rodent hosts (left in green) and *Bartonella* (right in yellow). The black lines between the phylogenetic trees indicate that the rodent is a host for the respective bacterium. This illustrates the link between phylogeny and biotic interactions. Image from Lei and Olival (2014).

Since species interactions have arisen as a result of a co-evolutionary process, it follows that using the phylogeny of the different species should convey some information about potential interactions (Peralta, 2016). For example, parasitism and infection are processes where the disease-causing agent is evolved to match its host (while its host in turn evolves to evade the parasite or pathogen, creating an evolutionary arms race). The link between phylogeny and infection has been well studied, e.g. (Poulin et al., 2011; Krasnov et al., 2012; Lei and Olival, 2014; Hadfield et al., 2014) and is illustrated in Figure 2.15. This is interesting for our pairwise learning approach, as phylogenetic information, which is often much easier to obtain than trait-based features.

2.4.3 On your computer: recommender systems

As a final example of pairwise learning, I will briefly discuss recommender systems. For a more comprehensive survey, see e.g. Isinkaye et al. (2015), Lü et al. (2012) and Su and Khoshgoftaar (2009).

In our digital world, there is an explosion of available options: websites to visit, people to connect with, goods to buy, books to read, etc. Recommender systems, sometimes called recommendation engines, are 'systems' that try to recommend relevant items such as websites, books or people to a user. The Google search engine is probably the most obvious recommender system that most people daily use. Many companies spend substantial resources to develop recommendation engines for commercial reasons. This can be a direct gain, such as Amazon, which recommends books to increase their revenue, or it can be slightly more subtle, such as Facebook, which recommends people you might know to extend their network and hence resources.

Recommender systems face the interesting problem that a relevant recommendation is not necessarily a useful one for the user. For example, (almost) everyone likes the Harry Potter books, so there is not much added value for Amazon to recommend the new book, as most interested people will order this anyway. This is called a popular recommendation, and is useful to set a baseline for the performance of the system, but is not really useful in practice. A good recommendation should contain some less obvious items that are unlikely to be found by the users themselves, the diversity versus accuracy trade-off (Herlocker et al., 2004; Zhou et al., 2010). In e-commerce, this is particularly important, as companies get the majority of their revenue from the so-called 'long-tail' goods that are rarely purchased, but yield profit due to their multitude (Anderson, 2006).

There are two types of user feedback that are usually provided to construct a recommender system: explicit and implicit feedback. For explicit feedback the users provide ratings for the items that are used to build and improve the recommendation system, e.g. rating books on

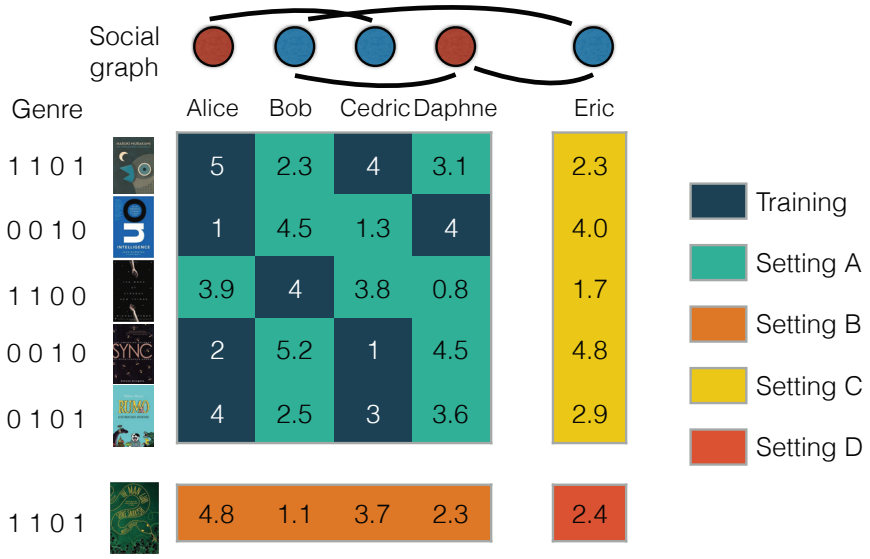


Figure 2.16: Toy dataset to construct a recommendation system from. Four users have each rated for some of the five books on a five-star scale. Using side-features of the users (in the form of a social network) and of the books (in the form of a binary vector encoding the genre), predictions can be made for each of the settings discussed in Section 2.3.2.

Amazon. For implicit feedback the system infers the users' preference based on behaviour of the latter, e.g. which music they listen to on Spotify. For an example of a dataset with explicit feedback (ratings of books), see Figure 2.16.

Recommender systems are usually classified as follows (Lü et al., 2012):

1. **Content-based recommendations**: recommendations are based on a feature description of the users and items.

2. **Collaborative recommendations**: recommendations are based on feedback provided for the items from other users. Within this class, two categories can be distinguished:

   (a) **Memory-based collaborative filtering**: recommended items are those that were preferred by other users who have a similar preference as the user.

   (b) **Model-based collaborative filtering**: recommended items are chosen based on a model such as matrix factorization or a graphical model trained on a dataset of user-item preferences.

3. **Hybrid recommendations**: methods that combine collaborative with content-based information.

Research in recommendation systems is an exciting field with many challenges. One is the cold start problem: how to deal with new users for which no preferences are known yet? How to make the methods scalable with huge amounts of data? How to efficiently update the recommender systems with new information, and how to deal with the changing preferences and trends? As discussed earlier, much of this PhD thesis is about making recommendations. In the later chapters I will illustrate the use of pairwise learning for constructing recommender systems and will address some of these issues.

3 Learning with kernels

3.1 A quick introduction to machine learning

In the previous chapter, pairwise prediction functions were discussed in depth. How to obtain such a pairwise prediction function was left as an open question. There is no complete mechanistic understanding for any of the problems discussed in this work. To predict whether two proteins would interact based on their amino acid sequences, one first has to solve the protein folding problem before using Schrödinger's equation to calculate the exact force fields between the two molecules. Both subproblems are, to state it mildly, rather hard. Hence, these pairwise prediction functions will not be hand-constructed, but will have to be learned using a dataset of observed pairwise interactions.

According to Shawe-Taylor and Cristianini (2004), machine learning is the science of finding stable patterns in data. As a branch of artificial intelligence, machine learning is concerned with designing algorithms that can extract such patterns from datasets.[1] Fitting a

[1] There is a considerate overlap (and not a small amount of friction) between the fields of machine learning and statistics. Both fields are about extracting information from data, with statistics originating from mathematics while machine learning emerged from computer science. Somewhat generalizing, machine learning

model is often called *training*, as the model is imagined to 'learn' from the data. The two most widely studied problems in machine learning are:

- **Supervised learning**: using data with labels, the goal is to find a function that maps the feature space to some label space. For example, predicting whether a picture is of an apple or an orange is an instance of a supervised classification problem.

- **Unsupervised learning**: trying to find patterns such as clusters in or a low-dimensional representation of unlabelled data. Detecting topics in a large corpus of texts is an example of this type of learning.

Machine learning is a rich field and many different types of algorithms exist; linear models, tree-based models, artificial neural networks, to name a few. In this chapter, and in this PhD thesis in general, I will focus on kernel-based methods. These methods allow linear models to be extended for nonlinear problems. Kernel methods are theoretically well founded, with a huge body of literature available to show the properties of kernel-based learning algorithms, see e.g. Aronszajn (1950); Vapnik (2000); Bartlett and Mendelson (2002); Schölkopf and Smola (2002); Shawe-Taylor and Cristianini (2004) for some overviews.

In this chapter I will present an intuitive introduction to kernel methods, rather than providing a rigorous mathematical treatment. For the latter, one better consults one of the many standard reference works on this topic. I will also discuss some algorithms for regression, dimension reduction and classification. The selected algorithms all have a closed-form solution for training. All these methods typically

mainly attempts to mine data for patterns and tries to find good prediction models. The goal of statistics on the other hand is to quantify and reduce uncertainty over the universe, for example to estimate an effect or to prove or disprove a hypothesis. In my opinion, the latter is the most complex problem.

provide an excellent performance and can be implemented in only a few lines of code.

We will momentarily omit the pairwise nature of our data. Instead, the input of our models is $x \in \mathcal{X}$, where $\mathcal{X} = \mathcal{U} \times \mathcal{V}$ if one wants to make the connection to the pairwise setting.

3.2 Kernels and the representer theorem

Kernel functions are mathematical tools to represent and manipulate objects in high-dimensional feature spaces. Suppose that there is some feature map $\phi : \mathcal{X} \to \mathcal{H}$, where $\mathcal{H}$ is a suitable space to represent these objects. The general idea is that in this high-dimensional space, a simple linear model might suffice to describe the patterns in the data. Performing the feature mapping explicitly might be prohibitive, because computing the map is computationally expensive or because $\mathcal{H}$ is an infinite-dimensional space. Using kernels one can perform algebraic operations in this space $\mathcal{H}$ without performing this mapping. First, let us formally define a valid kernel function.

Definition 3.1 (Kernel function). *A function $k : \mathcal{X}^2 \to \mathbb{R}$ is called a kernel function if it is symmetric, that is, $k(x, x') = k(x', x)$ for any two objects $x, x' \in \mathcal{X}^2$, and positive semi-definite, that is,*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0$$

for any $n > 0$, any choice of n objects $x_1, \ldots, x_n \in \mathcal{X}$, and any choice of real numbers $c_1, \ldots, c_n \in \mathbb{R}$.

We will often find it convenient to use a matrix containing the kernel values of all pairs of observations in the training data, called

the Gram matrix,

$$\mathbf{K} = [K_{ij}] = [k(x_i, x_j)] \,.$$

The kernel trick allows us to perform operations in $\mathcal{H}$ without performing an explicit feature mapping.

THEOREM 3.1 (Kernel trick). *(Schölkopf et al., 2004) For any kernel $k : \mathcal{X}^2 \to \mathbb{R}$ on a space $\mathcal{X}$, there exists a Hilbert space $\mathcal{H}$ and a mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{H}$ such that*

$$k(x, x') = \langle \boldsymbol{\phi}(x), \boldsymbol{\phi}(x') \rangle_{\mathcal{H}} \qquad \forall x, x' \in \mathcal{X} \,,$$

where $\langle \mathbf{p}, \mathbf{q} \rangle_{\mathcal{H}}$ represents the dot product in the Hilbert space between any two vectors $\mathbf{p}, \mathbf{q} \in \mathcal{H}$.

Hence, the kernel value between two objects is equivalent to computing the feature representation in the implied Hilbert space and computing the dot product. This is represented in Figure 3.1. The dot product is one of the fundamental operations in linear algebra. Many linear learning algorithms can be *kernelized* by restructuring the algorithm such that the feature descriptions of objects only appear as dot products. These dot products are subsequently replaced by an appropriate kernel. As an elementary example, suppose we want to compute the Euclidean distance between two objects in the Hilbert space:

$$\begin{aligned} D_{\mathcal{H}}(x, x') &= ||\boldsymbol{\phi}(x) - \boldsymbol{\phi}(x')|| \\ &= \sqrt{\langle \boldsymbol{\phi}(x), \boldsymbol{\phi}(x) \rangle_{\mathcal{H}} - 2\langle \boldsymbol{\phi}(x), \boldsymbol{\phi}(x') \rangle_{\mathcal{H}} + \langle \boldsymbol{\phi}(x'), \boldsymbol{\phi}(x') \rangle_{\mathcal{H}}} \,, \end{aligned}$$

then, after applying the kernel trick:

$$D_{\mathcal{H}}(x, x') = \sqrt{k(x, x) - 2k(x, x') + k(x', x')}$$

only needs kernel evaluations.

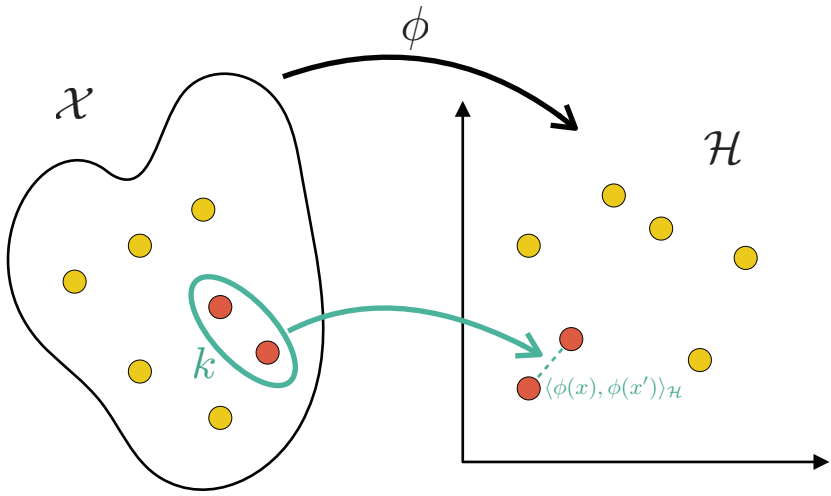


Figure 3.1: Representation of the kernel trick. The value of the kernel function of two pairs of objects in object space is identical to an inner product of the representations of the objects in the implied Hilbert space.

Any linear model in the Hilbert space can be written in dual form using only kernel evaluations with respect to the training data:

$$\begin{aligned} f(x) &= \mathbf{w}^\top \boldsymbol{\phi}(x) \\ &= \sum_{i=1}^{n} a_i k(x, x_i) \\ &= \mathbf{a}^\top \mathbf{k}(x) \end{aligned} \tag{3.1}$$

with $\mathbf{w}$ the primal parameters, $\mathbf{a} = [a_1, \ldots, a_n]^\top$ the dual parameters and $\mathbf{k}(x) = [k(x, x_1), \ldots, k(x, x_n)]^\top$.

Many machine learning algorithms can be formulized as a regularized risk function minimization problem. Here, a function is sought that has a trade-off between having a good fit to the data (i.e. a low value for the loss function) and being a simple function (i.e. having a small norm). The dual form can be motivated by the representer theorem.

THEOREM 3.2 (Representer theorem). *(Schölkopf and Smola, 2002) Denote by* $\Omega : [0, \infty[\rightarrow \mathbb{R}$ *a strictly increasing function, by* $\mathcal{X}$ *a set, and by* $\mathcal{L} : (\mathcal{X} \times \mathbb{R})^n \rightarrow \mathbb{R}$ *an arbitrary loss function. Then each minimizer* $f \in \mathcal{H}$ *of the regularized risk*

$$\mathcal{L}((y_1, f(x_1)), \ldots, (y_n, f(x_n))) + \Omega(||f||_{\mathcal{H}}) \tag{3.2}$$

admits a representation of the form

$$f(x) = \sum_{i=1}^{n} a_i k(x, x_i) \,.$$

The regularized risk minimization problem of Eq. (3.2) is an optimization problem defined over functions, objects of an infinite number of dimensions. The representer theorem is quite powerful in the sense that it allows for optimization problem (3.2) to be reformulated such that only a finite number of parameters of Eq. (3.1) need to be optimized. Furthermore, if the loss function is convex, finding the dual parameters is also a convex problem (Schölkopf and Smola, 2002).

The norm of a function in the Hilbert space can also be given in terms of the kernels and the dual parameters (Shawe-Taylor and Cristianini, 2004):

$$||f||^2_{\mathcal{H}} = \mathbf{a}^\top \mathbf{K} \mathbf{a} \,. \tag{3.3}$$

Kernel functions should reflect the similarity between objects in the problem at hand. For vectorial data, the radial basis function kernel is a popular kernel:

$$k^{\text{RBF}}(\mathbf{x}, \mathbf{x}') = e^{-\frac{||\mathbf{x}-\mathbf{x}'||^2}{\gamma^2}} \,. \tag{3.4}$$

Basically, this kernel gives an exponential transformation of the distance, with the hyperparameter γ the bandwidth of the kernel. Larger values of γ will allow instances that are further from each other to

be represented as more similar according to the kernel compared to smaller bandwidth values.

One of the fortes of kernel-based methods is that they allow to represent symbolic objects such as strings, sets, trees and graphs. Different kernels can be combined to fuse different types of feature descriptions of objects (Schölkopf et al., 2004), such as sequential and structural descriptors of proteins. For an extensive introduction on how to design a suitable kernel for many types of structured data, see Shawe-Taylor and Cristianini (2004).

3.3 Regression using kernel ridge regression

For kernel ridge regression (KRR), one tries to find a prediction function f that minimizes the following regularized risk:

$$\min_f \sum_{i=1}^{n} (f(x_i) - y_i)^2 + \lambda ||f||^2_{\mathcal{H}} \,.$$

Theorem 3.2, together with the expression of the function norm (Eq. (3.3)) allows us to reformulate this optimization problem in terms of the dual parameters:

$$\min_{\mathbf{a}} \, (\mathbf{K}\mathbf{a} - \mathbf{y})^\top (\mathbf{K}\mathbf{a} - \mathbf{y}) + \lambda \mathbf{a}^\top \mathbf{K}\mathbf{a} \,,$$

with the prediction function of the form of Eq. (3.1). A closed-form solution can easily be obtained by taking the derivative with respect

to $\mathbf{a}$ and setting the result equal to zero[2]:

$$\mathbf{a} = (\mathbf{K} + \lambda\mathbb{I})^{-1}\mathbf{y}\,. \tag{3.5}$$

This system can be rewritten using the eigenvalue decomposition of the Gram matrix:

$$\mathbf{K} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^\top\,,$$

with $\mathbf{U}$ an orthonormal matrix containing the eigenvectors as rows and $\boldsymbol{\Sigma}$ a diagonal matrix containing the corresponding real-valued and non-negative eigenvalues $\sigma_1, \ldots, \sigma_n$. Hence, Eq. (3.5) can be rewritten as

$$\mathbf{a} = \mathbf{U}(\boldsymbol{\Sigma} + \lambda\mathbb{I})^{-1}\mathbf{U}^\top\mathbf{y}\,.$$

Since $\boldsymbol{\Sigma}$ is a diagonal matrix, the dual parameters can be obtained cheaply for any value of λ. However, either inverting or computing the eigenvalue decomposition of an $n \times n$ matrix has a time complexity of $\mathcal{O}(n^3)$. This indicates that the practical use of kernel ridge regression (KRR) is limited by the number of training instances.

Note that minimizing a squared loss implies, in principle, that the error is homoscedastic (the variance is the same everywhere) and normally distributed. These assumptions are rarely fulfilled in reality, though rarely matter for obtaining a good predictive model. If the variance of the irreducible error is not constant over the feature space, there will be a higher variance in the trained parameters, though this will not influence their bias. This issue can be approached by applying an appropriate scaling to the labels or by given a higher weight in the

[2]Technically, finding the parameters requires solving the system

$$\mathbf{K}\mathbf{y} = \mathbf{K}(\mathbf{K} + \lambda\mathbb{I})\mathbf{a}\,,$$

hence the given solution is only valid when the Gram matrix is invertible. If $\mathbf{K}$ is singular (only positive semi-definite), then the given solution should be interpreted as using $\lim_{\varepsilon\to 0}(\mathbf{K} + \varepsilon\mathbb{I})$ as Gram matrix.

loss function to instances with lower variance. Using the squared loss is known not to be robust to outliers, but works well in many practical situations.

One can give a Bayesian interpretation to KRR by treating it as a Gaussian process (Rasmussen and Williams, 2006). Here, the kernel function is used to construct a prior distribution on the prediction functions, while λ should be interpreted as the variance of the isotropic, normally distributed noise on the labels. Treating kernel ridge regression as a Gaussian process motivates tuning the hyperparameters by maximizing the posterior likelihood. Furthermore, it allows us to derive the variance of the uncertainty on a prediction $f(x)$:

$$\sigma^2(x) = k(x, x) + \lambda - \mathbf{k}(x)^\top (\mathbf{K} + \lambda \mathbb{I})^{-1} \mathbf{k}(x) \,.$$

The uncertainty of a prediction is often used in Bayesian optimization (Snoek et al., 2012). Bayesian optimization is often used to iteratively improve a variable when each evaluation step is expensive, for example if an evaluation requires to perform a wet-lab experiment or a long simulation. Here, one builds a surrogate model that quickly gives the estimated outcome as well as the uncertainty of this outcome. For instance, consider the task of finding a drug with an as high as possible activity according to a machine learning model. Likely, one would prefer to test a drug with some expected activity, but a low variance of uncertainty, over another drug with a slightly higher predicted activity, but with a much higher variance of uncertainty. Metrics such as expected improvement and upper bound confidence both take the prediction as well as the variance of the prediction into account to select appropriate objects for experimental validation.

3.4 Dimension reduction using kernel principal component analysis

Often objects are represented by a high-dimensional feature vector. Behind seemingly complex data, there is often a simple explanation. For example, the datapoints might lie approximately on a low-dimensional manifold. Finding such low-dimensional representations of a dataset is an important topic in unsupervised learning.

An elementary algorithm for dimension reduction is principal component analysis (PCA). Classical PCA provides a linear projection of the data such that the variance is maximized (Bishop, 2006). Kernel PCA is a nonlinear extension of basic PCA, where the dimension reduction is performed in the implied Hilbert space by the kernel. For PCA, the data is usually assumed to be centred, i.e.

$$\breve{\boldsymbol{\phi}}(x_i) = \boldsymbol{\phi}(x_i) - \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{\phi}(x_j)\,.$$

The corresponding centred Gram matrix is given by

$$\breve{\mathbf{K}} = \mathbf{K} - \frac{1}{n}\mathbb{J}_{n\times n}\mathbf{K} - \frac{1}{n}\mathbf{K}\mathbb{J}_{n\times n} + \frac{1}{n^2}\mathbb{J}_{n\times n}\mathbf{K}\mathbb{J}_{n\times n}\,,$$

with $\mathbb{J}_{n\times n}$ an $n\times n$ matrix filled with ones. The components for kernel PCA in dual space can be obtained by solving the following eigenvalue problem

$$\breve{\mathbf{K}}\mathbf{a}_i = \sigma_i n \mathbf{a}_i\,,$$

where the projection function for new objects x onto the i-th component is again of the form of Eq. (3.1). In Figure 3.2 classical and kernel PCA are demonstrated on a toy dataset.

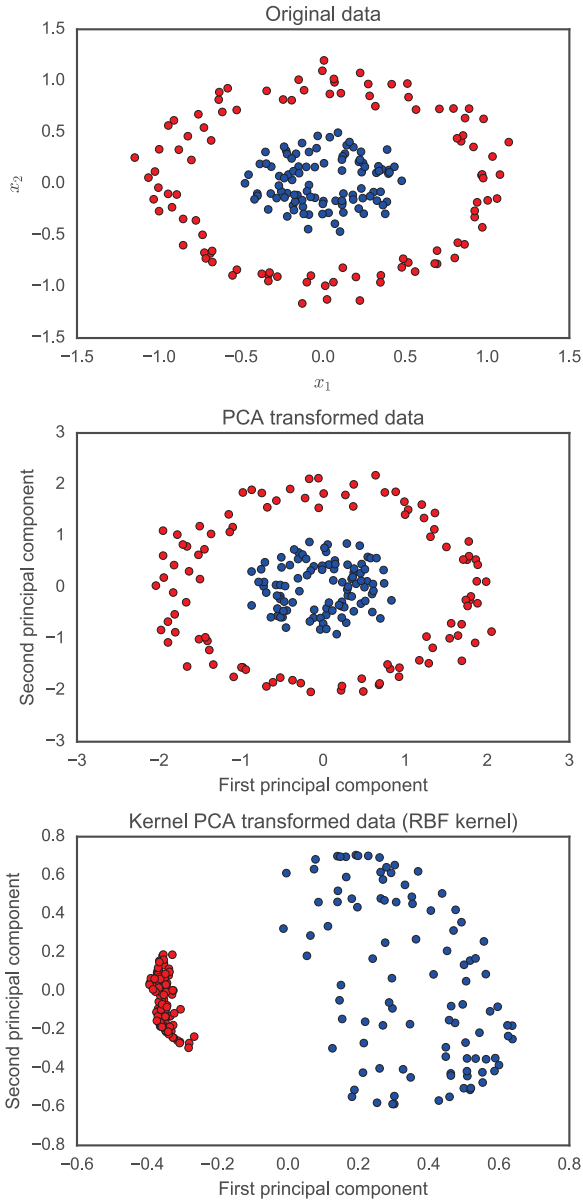


Figure 3.2: (top) A toy dataset with the observations arranged in two concentric circles with different diameters. (middle) Projection of the dataset onto the first two principal components. This does not change the representation as there are no linear trends in the data. (bottom) Projection of the dataset onto the first two principal components in the Hilbert space. The first component can be used to separate the two circular patterns.

3.5 Classification using kernel Fisher discriminant analysis

Kernel methods are ubiquitously applied for classification problems. The support vector machine is perhaps the most well-known example of a kernel-based classifier. The support vector machine learns an optimal hyperplane in the Hilbert space to separate the two classes. This hyperplane is chosen such that the margin (the minimal distance from the instances to the hyperplane) is maximized. To train a support vector machine, one typically has to solve a quadratic programming problem. In the spirit of this PhD thesis, I will briefly discuss kernel Fisher discriminant analysis, for which one only has to solve a linear system of equations. Empirically, it has been found that the performance of both classifiers is comparable (Schölkopf and Smola, 2002).

Fisher discriminant analysis tries to find a projection of the data such that the classes are maximally separated. One can consider it as a type of principal component analysis that takes the class labels into account. The function that performs the projection is the discriminant function and is again in the dual form of Eq. (3.1). Concretely, the points are projected so that the ratio of the variance between classes and the variance within classes (the so-called Rayleigh coefficient) is maximized. The dual parameters for the optimal projection can be obtained by solving the system

$$(\mathbf{BK} + \lambda\mathbb{I})\mathbf{a} = \mathbf{y}\,,$$

with $\mathbf{y}$ the vector of the labels encoded as $y_i = +1$ for instances of the one class (i.e., the positive class) and $y_i = -1$ for instances of the other class (i.e., the negative class). The matrix $\mathbf{B}$ is a symmetric positive-definite matrix to reweigh the Gram matrix. For balanced problems, $\mathbf{B}$ is close to the identity matrix. This motivates using

KRR as a classifier. I refer to standard textbooks for the details - e.g. Shawe-Taylor and Cristianini (2004); Bishop (2006)

The obtained function $f(x)$ thus returns a value such that higher values indicate one class and lower values another. Decisions can be made by choosing an appropriate threshold to discriminate between one class or the other. One can also fit a normal distribution to the scores of the instances for each of the classes. This is motivated by the empirical observation that in high-dimensional feature spaces, the histogram of the scores of the instances can be closely approximated by a normal distribution (Schölkopf and Smola, 2002). Using Bayes' rule one can subsequently obtain posterior probabilities for making predictions. Fisher discriminant analysis is illustrated in Figure 3.3.

Fisher discriminant analysis should not be confused with linear discriminant analysis (Hastie et al., 2001). This is a generative classifier that models every class as a multivariate normal distribution, each with the same covariance structure[3]. Using Bayes' rule, the optimal discriminant function is made. Though very similar to Fisher discriminant analysis, linear discriminant analysis makes strong assumptions on the distribution of the data that the other does not.

[3]The extension which uses a different covariance matrix for every class is called quadratic discriminant analysis.

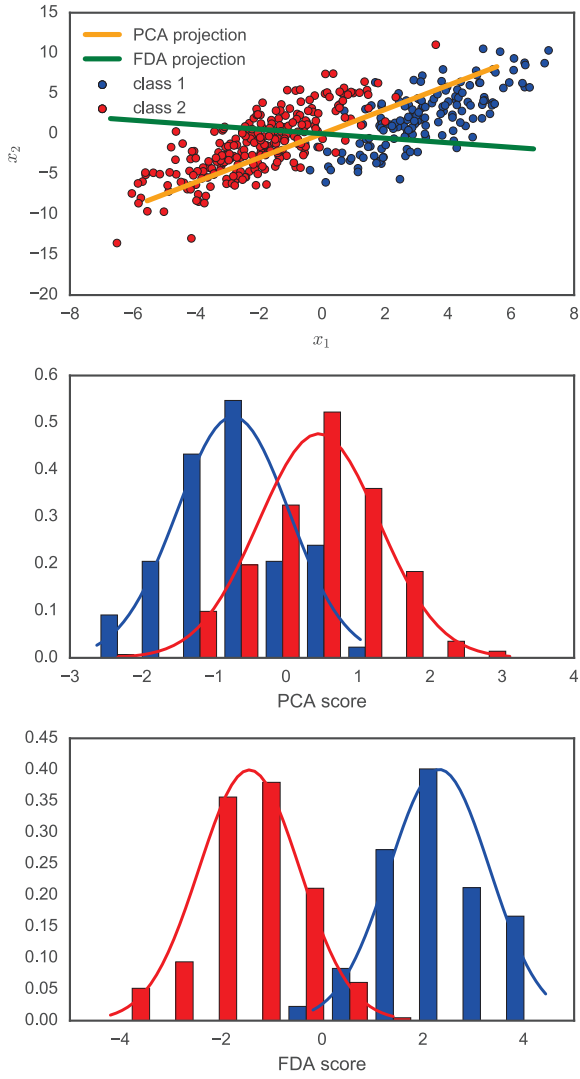


Figure 3.3: (top) A toy binary classification problem. The first principal component (indicated in orange) and the projection for Fisher discriminant analysis (indicated in green) are shown. (middle) Empirical distribution with fitted normal density function for the instances for the two classes, projected using principal component analysis. This does not result in a good discrimination of the classes, as the main direction of the variance of the data does not correspond to a satisfactory separation. (bottom) Empirical distribution with fitted normal density function for the instances for the two classes, projected using Fisher discriminant analysis. This results in an excellent separation of the two classes.

3.6 Concluding remarks and the future of kernels

For many machine learning researchers, kernel methods are synonymous with support vector machines. It is my belief that by usually introducing these two distinct concepts in one discourse, both become harder to understand. A support vector machine is simply a linear method for classification or regression, like logistic regression, which can be extended to nonlinear problems. Admittedly, this method is well suited to kernelization, as it leads to some wonderful properties in sparseness and learning-theoretic guarantees (Vapnik, 2000; Shawe-Taylor and Cristianini, 2004). To me, kernels are tools for thinking about the feature representations of instances, functions and distributions. Linking them to only a single method would not do justice to this versatile construct.

At the moment of writing, kernel methods do not seem to be the most trendy topic in machine learning. Technological advances and new milestones have created a renewed enthusiasm for artificial neural networks - e.g. Bengio et al. (2012); Mnih et al. (2015); Bengio (2016); Lin and Tegmark (2016). As a machine learning researcher, it is hard not to be at least somewhat intrigued by these methods. The idea of just designing a model of any complexity and using gradient-based optimization to train for a specific task is extremely flexible and powerful. However, it is my opinion that kernel methods still have their role to play in problems with a small to intermediate amount of data (which are still omnipresent in the field of computational biology and bioscience engineering). This chapter and by extension this PhD thesis focuses on learning methods with an efficient closed-form solution and provable guarantees of performance. Kernel methods, or more generally, linear methods that work in an extended feature space, are still exciting research topics. To conclude this chapter, I will briefly

discuss two ideas which I believe will allow kernel methods to remain relevant in the coming years.

3.6.1 Representing distributions using kernels

Firstly, there is this elegant idea of representing a distribution of some random variable $X \in \mathcal{X}$ as a point in the Hilbert space (Smola et al., 2007; Song et al., 2013; Kadri et al., 2013). This is done by taking the first moment of the implicit feature representation in the Hilbert space:

$$\boldsymbol{\mu}_X = \mathbb{E}_X[\boldsymbol{\phi}(x)] \,.$$

In practice, one works with the empirical moment

$$\begin{aligned} \hat{\boldsymbol{\mu}}_X &= \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{\phi}(x_i) \\ &= \frac{1}{n} \sum_{i=1}^{n} k(x_i, \cdot) \,. \end{aligned}$$

One can show that the convergence of $\hat{\boldsymbol{\mu}}_X \to \boldsymbol{\mu}_X$ is $\mathcal{O}(\sqrt{m})$, independent of the dimensionality of x or $\boldsymbol{\phi}(x)$! This idea is represented in Figure 3.4. Using kernels to think about distributions is intriguing. We now have a tool for manipulating the distribution of complex objects such as proteins or species interaction networks. Using the Kronecker product, one can model the joint distribution of variables (joint spaces). Using the equivalent of the sum rule, chain rule and Bayes's rule for kernel embeddings, one can easily obtain a representation of a conditional distribution of, for example, $\mathcal{U}$ (e.g. ligands) conditional on $\mathcal{V}$ (e.g. proteins). To perform structured output prediction (Problem (2.3)), one has to solve the inverse problem, either by solving an optimization problem or by learning the inverse mapping (Bakir et al., 2004). In summary, kernels can also be used to

think about complex distributions, much like copulas can be used to study the dependence between variables (Genest and Favre, 2007).

3.6.2 Randomization for approximating kernels and constructing nonlinear representations

A second idea that pairs well with kernels is the recent interest in randomized algorithms. Kernel methods have the reputation of not being feasible for large-scale problems due to the prohibitive time and memory requirements for computing the Gram matrix[4]. A typical solution is to construct a low-rank approximation of the Gram matrix and learn the model in primal form. This can be obtained by the Nyström approximation (Drineas and Mahoney, 2005). Recently, there has been a surge in interest in so-called randomized numerical linear algebra (Drineas and Mahoney, 2016), which uses randomized algorithms to approximately solve large-scale problems in linear algebra. Ideas from this field have led to improved algorithms to efficiently find low-rank approximations of the Gram matrix (Gittens and Mahoney, 2013). Furthermore, random projections in combination with a nonlinear transformation can be seen as an alternative for working with kernels. Denoted as extreme learning machines (Huang et al., 2006), these embody the best of kernel methods without the scalability issues. For example, it has been found that classical PCA on a random nonlinear expansion of a dataset is a fast and reliable alternative for kernel PCA (Lopez-Paz et al., 2014). The exciting part is finding randomized representations for structured objects. For example, the success of reservoir computing is due to the fact that randomized representations can be used to learn time series - e.g. wyffels and Schrauwen (2010).

[4]Storing a Gram matrix requires a memory size of $\mathcal{O}(n^2)$ and typical operations such as inversion and the computation of eigenvalues have a time complexity of $\mathcal{O}(n^3)$.

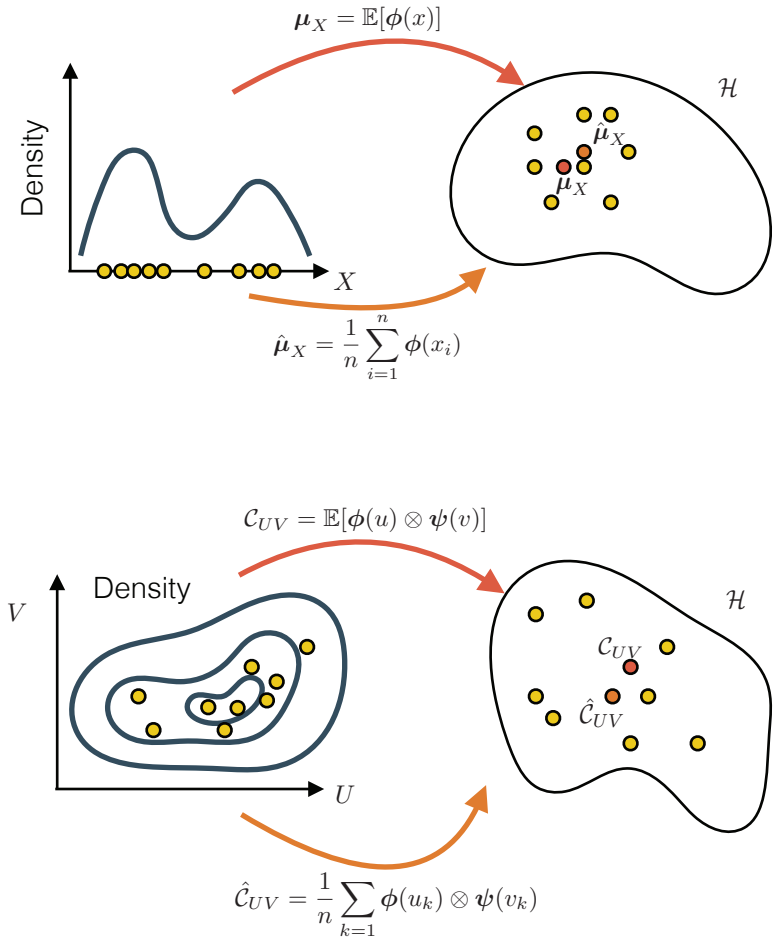


Figure 3.4: Main idea of representing distributions using kernels. (top) A distribution of a variable X can be represented by the first moment $\boldsymbol{\mu}_X$ in the reproducing Hilbert space. This can be estimated based on a dataset using the empirical moment $\hat{\boldsymbol{\mu}}_X$. (bottom) The same idea can be applied for a joint distribution of the variabels $U \times V$.

4

Pairwise learning with Kronecker product kernels

As discussed in depth in Chapter 2, pairwise learning or dyadic prediction concerns the prediction of properties for pairs of objects. It can be seen as an umbrella covering various machine learning problems such as matrix completion, collaborative filtering, multi-task learning, transfer learning, network prediction and zero-shot learning. In this chapter I analyze several kernel-based methods for pairwise learning, with a particular focus on a recently suggested two-step method. I show that this method offers an appealing alternative for commonly-applied Kronecker-based methods that model dyads by means of pairwise feature representations and pairwise kernels. In a series of theoretical results, I establish correspondences between the two types of methods in terms of linear algebra and spectral filtering, and I analyze their statistical consistency. In addition, the two-step method allows us to establish novel algorithmic shortcuts for efficient training and validation on very large datasets. Putting those properties together, I believe that this simple, yet powerful method can become a standard tool for many problems. Extensive experimental results for a range of practical settings are reported.

4.1 Pairwise learning as multi-task learning

As discussed at length in Chapter 2, many real-world machine learning problems can naturally be represented as pairwise learning or dyadic prediction problems. In contrast to more traditional learning settings, the goal here consists of making predictions for pairs of objects, each of them being characterized by a feature representation.

Pairwise learning has strong connections with many other machine learning settings. Especially a link with multi-task learning can be advocated, by calling the first object of the dyad an 'instance' and the second object a 'task'. As a typical multi-task learning example, consider ten schools providing the grades for a selection of their students. Suppose we want to predict the scores for new students in a school. The most straightforward path would be to fit a different model for each school, which uses features of the students. Intuitively, we might do better by building one model which takes all information into account. This can be done by learning a general model that takes both the student and the school as input. The prediction function is thus pairwise: it takes both an instance and a task as input. In multi-task learning, the underlying idea for making the distinction between instances and tasks is that the feature description of the instances is often considered as more informative for making a prediction, while the feature description of the tasks is mainly used to steer learning into the right direction. In the majority of multi-task learning methods, a feature description for tasks is even not given, though often the idea that the tasks can be clustered or are located on a low-dimensional manifold is exploited.

In this chapter I adopt a multi-task learning terminology for pairwise learning. The objects $u \in \mathcal{U}$ will be referred to as instances, while the objects $v \in \mathcal{V}$ will be referred as tasks.

Various types of learning methods can be used to solve prediction problems of that kind. Kernel methods are very popular in bioinformatics, as indicated by the large number of applications in bioinformatics. They can be easily employed for pairwise learning by defining so-called pairwise kernels $\Gamma((u,v),(\bar{u},\bar{v}))$, which measure the similarity between two dyads (u,v) and $(\bar{u},\bar{v})$. Kernels of that kind can be used in tandem with any conventional kernelized learning algorithm, such as support vector machines, KRR and kernel Fisher discriminant analysis. In case of regression or classification tasks, one obtains prediction functions of the following form:

$$f(u,v) = \sum_{h=1}^{n} \alpha_k \Gamma((u,v),(u_h,v_h))\,,$$

with $\alpha_1,\ldots,\alpha_n$ the dual parameters, which are obtained by the learning algorithm – see subsequent section for references and details.

4.2 Chapter outline

In this chapter, I survey the established Kronecker-kernel based ridge regression for pairwise learning, as well as a simple, yet elegant two-step method developed during my PhD. In the first step, a KRR model is trained on auxiliary tasks, and adopted to predict labels for the related target task. Then, in a second step, a second model is constructed, by employing KRR on the target data, augmented with the predictions of the first phase. This method was first presented in a conference paper (Pahikkala et al., 2014). One year later, the same method was independently proposed as a tool to solve zero-shot learning problems (Romera-Paredes and Torr, 2015). From a different but related perspective, Schrynemackers et al. (2015) recently also proposed a similar method for biological network inference, but here tree-based methods instead of kernel methods were used as base learners. Those three papers have confirmed that the two-step method

can obtain state-of-the-art performance on standard benchmarks in various pairwise learning settings. In this chapter, I will not put the emphasis on demonstrating the performance of this method. In contrast, the main focus is rather on a theoretical, computational and experimental analysis and justification of the two-step method, since I believe that it has a lot of potential. It will become clear that the two-step method is much simpler to implement than efficient Kronecker models, while manifesting more flexible model selection capabilities. It will also be shown to be applicable to more heterogeneous transfer learning settings.

This chapter can be subdivided in several parts that describe different aspects of the kernel-based methods for pairwise learning. Before going into mathematical details, Section 4.3 gives an overview of related settings in pairwise learning. I identified the four different prediction settings discussed in Section 2.3.2, for which it is crucial to make a subdivision further in this chapter. In Section 4.4 different KRR-based methods that can be used for pairwise learning are described, including the novel two-step method. Subsequently, I show in Section 4.5 via linear algebra and spectral filtering that this approach is closely related to other kernel methods, but with slightly different regularization mechanisms. I also formally prove universal approximation properties for two-step KRR. In Section 4.6 I use a specific decomposition to derive novel algorithmic shortcuts for leave-one-out cross-validation and for updating existing models with new training instances or tasks. A very important merit of the two-step method will be that there are closed-form shortcuts for cross-validation for any of the prediction settings discussed in Section 4.3, in contrast to the other methods. Furthermore, I propose a simple iterative algorithm to deal with incomplete datasets. In the experiments (Section 4.7) I consider several dyadic prediction problems, studying generalization for the different settings as well as efficient implementations for training and testing the models. The results show that the two-step method

can be highly beneficial when there is no labeled data at all, or only a small amount of labeled data available for the target task, while in settings where there is a significant amount of labeled data available for the target task, an independent-task model suffices. Furthermore, I showcase the tools for performing cross-validation for the different settings and updating the model with extra instances or tasks.

4.3 Related settings to pairwise learning

In this section I provide a quick recap of the four settings discussed in Section 2.3.2. This will allow to give a brief overview of related methods, and to perform a more detailed analysis in the upcoming sections. In pairwise learning it is extremely important to implement appropriate training and evaluation procedures. For example, in a large-scale meta-study about biological network identification, it was found that these concepts are vital to correctly evaluate pairwise learning models (Park and Marcotte, 2012). Four settings can be distinguished:

- **Setting A**: Both u and v are observed during training, as parts of different dyads, but the label of the dyad (u, v) must be predicted.

- **Setting B**: Only v is known during training, while u is not observed in any dyad, and the label of the dyad (u, v) must be predicted.

- **Setting C**: Only u is known during training, while v is not observed in any dyad, and the label of the dyad (u, v) must be predicted.

- **Setting D**: Neither u nor v appear in any training dyad, but the label of the dyad (u, v) must be predicted.

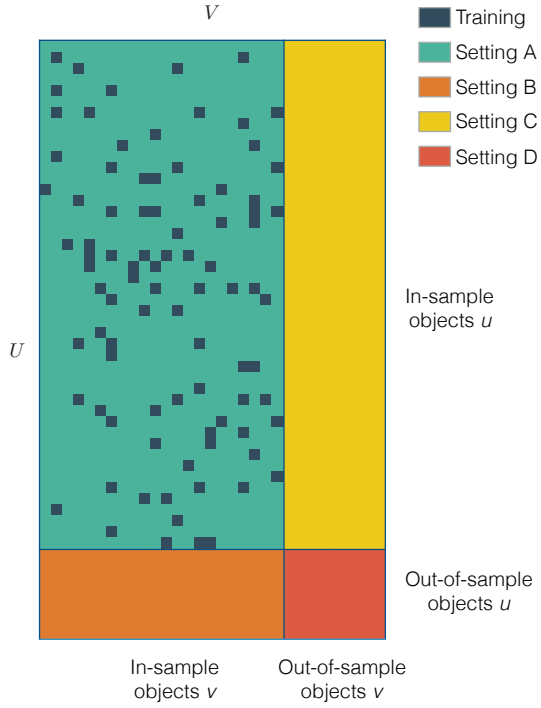


Figure 4.1: An overview of different prediction settings in pairwise learning. I will mainly deal with complete datasets, where for each combination of training instance and task we have exactly one observation in the training data. See main text for links with different machine learning settings.

These settings are represented graphically in Figure 4.1. Setting A, the matrix completion problem, is of all four settings by far the most studied setting in the machine learning literature. Motivated by applications in collaborative filtering and link prediction, matrix factorization and related techniques are often applied to complete partially observed matrices. Missing values correspond to couples (u, v) that are not observed during training - see e.g. Larochelle et al. (2008) for a review. Many of these matrix completion algorithms do not incorporate side features (features of the instances and tasks) and make assumptions on the structure of the true label matrix by, for example,

assuming that the completed matrix is low rank or has a low nuclear norm (Candes and Recht, 2008; Mazumder et al., 2010). Recently, a framework based on bipartite graphs was proposed, which exploits the network structure for transductive link prediction (Liu and Yang, 2015). If one uses the notion of 'latent features' (which can be implemented by means of a delta kernel), one can exploit both the structure of the label matrix as well as the side-features. Some interesting work has been done to unify these two approaches, using both the structure of the matrix, as well as features, e.g. Basilico and Hofmann (2004a); Abernethy et al. (2008); Menon and Elkan (2011).

Settings B and C are very similar, and a variety of machine learning methods can be applied to these settings. From a recommender systems viewpoint, those settings resemble the cold start problem (new user or new item), for which hybrid and content-based filtering techniques are often applied – see e.g. Adams et al. (2010); Fang and Si (2011); Menon and Elkan (2010); Shan and Banerjee (2010); Zhou et al. (2012) for a non-exhaustive list. From a bioinformatics viewpoint, Settings B and C are often analyzed using graph-based methods that take the structure of a biological network into account – see e.g. Schrynemackers et al. (2013) for a recent review. When the features of the tasks are negligible or unavailable, while those of the instances are informative, Setting B can be interpreted as a multi-label classification problem (in case of binary labels), a multi-output regression problem (in case of continuous labels) or, more generally, as a multi-task learning problem. Here, most techniques encode dependency between the tasks by means of a suitable loss function or by jointly regularizing the different tasks (Dembczynski et al., 2012). Prior knowledge on the tasks can be incorporated by using a feature description of the tasks that captures the relations between the tasks. Setting C is closer to transfer learning, in which one wants to generalize to new tasks. Here as well, a large number of applicable methods are available in the literature (Pan and Yang, 2010).

Matrix factorization and hybrid filtering strategies are not applicable to Setting D. I will refer to this setting as the *zero-shot learning* problem. This setting finds important applications in domains such as bioinformatics and chemistry – see experiments. Compared to the other three settings, Setting D has received less attention in the literature (but it is gaining rapidly in popularity, see e.g. Larochelle et al. (2008); Park and Chu (2009); Menon and Elkan (2010); Palatucci et al. (2009); Pahikkala et al. (2013); Rohrbach et al. (2011)). In the experimental section, I will investigate the transition phase between Settings C and D, when v occurs very few times in the training dataset, while u of the dyad (u, v) is only observed in the prediction phase. I refer to this setting as an *almost zero-shot learning* problem.

Full and almost zero-shot learning problems can only be solved by considering feature representations of dyads. For Setting D Kronecker-based kernel methods are often employed (Vert et al., 2007; Brunner and Fischer, 2012). They have been successfully applied in order to solve problems such as product recommendation (Basilico and Hofmann, 2004a; Park and Chu, 2009), enzyme annotation Stock et al., (2014), prediction of protein-protein interactions (Ben-Hur and Noble, 2005; Kashima et al., 2009) or protein-nucleic acid interactions (Pelossof et al., 2015) interactions, drug design (Jacob and Vert, 2008), prediction of game outcomes (Pahikkala et al., 2010) and document retrieval (Pahikkala et al., 2013). For classification and regression problems, a standard recipe consists of plugging pairwise kernels in support vector machines, KRR, or any other kernel method. Efficient optimization approaches based on gradient descent (Park and Chu, 2009; Kashima et al., 2009) and closed-form solutions (Pahikkala et al., 2013) have been proposed. In the next section, I will review those methods more in detail.

4.4 Pairwise learning using ridge-regression-based methods

4.4.1 Independent-task kernel ridge regression

Suppose that the training set is complete and that for each task we have a set of m labeled dyads $U = \{u_i\}_{i=1}^m$. Let $\mathbf{Y}_{\cdot j} \in \mathbb{R}^m$ be the labels of task v_j and $k(\cdot, \cdot)$ be a suitable kernel function which quantifies the similarity between the different instances. Since a separate and independent model is trained for each task, we will denote this setting as independent task (IT) kernel ridge regression[1]. For each task v_j, one would like to learn a function of the form

$$f_j^{\mathrm{IT}}(u) = \sum_{i=1}^{m} a_{ij}^{\mathrm{IT}} k(u, u_i)\,,$$

with a_{ij}^{IT} parameters that minimize a suitable objective function. In the case of KRR, this objective function is the squared loss with an L_2-complexity penalty. The parameters for the individual tasks using KRR can be found jointly by minimizing the following objective function (Bishop, 2006):

$$J(\mathbf{A}^{\mathrm{IT}}) = \mathrm{tr}[(\mathbf{K}\mathbf{A}^{\mathrm{IT}} - \mathbf{Y})^\top (\mathbf{K}\mathbf{A}^{\mathrm{IT}} - \mathbf{Y})] + \lambda_u \mathrm{tr}[\mathbf{A}^{\mathrm{IT}\top} \mathbf{K}\mathbf{A}^{\mathrm{IT}}]\,, \quad (4.1)$$

with $\mathbf{A}^{\mathrm{IT}} = [a_{ij}^{\mathrm{IT}}] \in \mathbb{R}^{m \times q}$ and $\mathbf{K} \in \mathbb{R}^{m \times m}$ the Gram matrix associated with the kernel function $k(\cdot, \cdot)$ for the instances. For simplicity, I assume the same regularization parameter λ_u for each task v, though extensions to different penalties for different tasks are straightforward. This basic setting assumes no crosstalk between the tasks, as each model is fitted independently. The optimal coefficients that minimize

[1] Recall that in biological network construction, this is denoted as a local model, see Section 2.4.1.

Eq. (4.1) can be found by solving the following linear system:

$$(\mathbf{K} + \lambda_u \mathbb{I})\, \mathbf{A}^{\mathrm{IT}} = \mathbf{Y}\,. \tag{4.2}$$

Using the singular value decomposition of the Gram matrix, this system can be solved for any value of λ_u with a time complexity of $\mathcal{O}(m^3 + m^2 q)$.

4.4.2 Pairwise and Kronecker kernel ridge regression

Suppose one has prior knowledge about which tasks are more similar, quantified by a kernel function $g(\cdot, \cdot)$ defined over the tasks. Several authors (see Alvarez et al. (2012); Baldassarre et al. (2012) and references therein) have extended KRR to incorporate task correlations via matrix-valued kernels. However, most of this literature concerns kernels for which the tasks are fixed at training time. An alternative approach, allowing for the generalization to new tasks more straightforwardly by means of such a task kernel, is to use a pairwise kernel $\Gamma\left((u, v), (\bar{u}, \bar{v})\right)$. Pairwise kernels provide a prediction function of type

$$f(u, v) = \sum_{h=1}^{n} \alpha_h \Gamma\left((u, v), (u_h, v_h)\right)\,, \tag{4.3}$$

where $\boldsymbol{\alpha} = [\alpha_h]$ are parameters that minimize the following objective function similar to the one used for independent task KRR:

$$J(\boldsymbol{\alpha}) = (\boldsymbol{\Gamma}\boldsymbol{\alpha} - \mathbf{y})^{\top}(\boldsymbol{\Gamma}\boldsymbol{\alpha} - \mathbf{y}) + \lambda \boldsymbol{\alpha}^{\top}\boldsymbol{\Gamma}\boldsymbol{\alpha}\,. \tag{4.4}$$

The minimizer can also be found by solving a system of linear equations:

$$(\boldsymbol{\Gamma} + \lambda \mathbb{I})\, \boldsymbol{\alpha} = \mathbf{y}\,, \tag{4.5}$$

with $\mathbf{\Gamma}$ the Gram matrix. The most commonly used pairwise kernel is the Kronecker product pairwise kernel (Basilico and Hofmann, 2004a; Oyama and Manning, 2004; Ben-Hur and Noble, 2005; Park and Chu, 2009; Hayashi et al., 2012; Bonilla et al., 2007; Pahikkala et al., 2013). This kernel is defined as

$$\Gamma^{\mathrm{KK}}\left((u,v),(\bar{u},\bar{v})\right) = k\left(u,\bar{u}\right) g\left(v,\bar{v}\right) , \tag{4.6}$$

a product of the data kernel $k(\cdot,\cdot)$ and the task kernel $g(\cdot,\cdot)$. Many other variations of pairwise kernels have been considered to incorporate prior knowledge on the nature of the relations (e.g. Vert et al. (2007); Pahikkala et al. (2010); Waegeman et al. (2012); Pahikkala et al. (2013)) or for more efficient calculations in certain settings, e.g. Kashima et al. (2010).

Let $\mathbf{G} \in \mathbb{R}^{q\times q}$ be the Gram matrix for the tasks. Then, for a complete training set, the Gram matrix for the instance-task pairs is the Kronecker product $\mathbf{\Gamma} = \mathbf{G} \otimes \mathbf{K}$. Often it is infeasible to use this kernel directly due to its large size. When the dataset is complete, the prediction function (Eq. (4.3)) can be written as

$$f^{\mathrm{KK}}(u,v) = \sum_{i=1}^{m}\sum_{j=1}^{q} a_{ij}^{\mathrm{KK}} k(u,u_i) g(v,v_j) . \tag{4.7}$$

The matrix $\mathbf{F}$ containing the predictions for the training data using a pairwise kernel can be obtained by a linear transformation of the training labels:

$$\begin{aligned} \mathrm{vec}(\mathbf{F}) &= \mathbf{\Gamma}\left(\mathbf{\Gamma} + \lambda\mathbb{I}\right)^{-1}\mathrm{vec}(\mathbf{Y}) && (4.8)\\ &= \mathbf{H}^{\Gamma}\mathrm{vec}(\mathbf{Y}) , && (4.9) \end{aligned}$$

with vec the vectorization operator, which stacks the columns of a matrix into a vector. In the statistical literature $\mathbf{H}^{\Gamma} = \mathbf{\Gamma}\left(\mathbf{\Gamma} + \lambda\mathbb{I}\right)^{-1}$ is referred to as the so-called hat matrix (Hastie et al., 2001), which transforms the measurements into estimates. As a special case of

the Kronecker KRR, we also retrieve ordinary Kronecker kernel least-squares (OKKLS), when the objective function of Eq. (4.4) has no regularization term (i.e. $\lambda = 0$).

Several authors have pointed out that, while the size of the system in Eq. (4.5) is considerably large, its solutions for the Kronecker product kernel can be found efficiently via tensor algebraic optimization (Van Loan, 2000; Martin and Van Loan, 2006; Kashima et al., 2009; Raymond and Kashima, 2010; Pahikkala et al., 2013; Alvarez et al., 2012). This is because the eigenvalue decomposition of a Kronecker product of two matrices can easily be computed from the eigenvalue decomposition of the individual matrices. The time complexity scales roughly with $\mathcal{O}(m^3 + q^3)$, which is required for computing the singular value decomposition of both the instance and task kernel matrices (see Property A.6 in the appendix), but the complexities can be scaled down even further by using sparse kernel matrix approximation.

However, these computational short-cuts only concern the case in which the training set is complete. If some of the instance-task pairs in the training set are missing or if there are several occurrences of certain pairs, one has to resort, for example, to gradient-descent-based training approaches (Park and Chu, 2009; Kashima et al., 2009; Pahikkala et al., 2013). While the training can be accelerated via tensor algebraic optimization, those techniques still remain considerably slower than the approach based on eigenvalue decomposition. In Section 4.6.3, I provide an alternative algorithm to deal with incomplete data.

4.4.3 Conditional ranking on pairwise data

As discussed in Section 2.3, a pairwise prediction function is often useful in a ranking setting. Kronecker kernel-based regression can be optimized for ranking (Pahikkala et al., 2013). To this end, rather than minimizing a squared loss as in Eq. (4.4), it is more appropriate

to minimize a ranking loss:

$$\mathcal{L}(f,S) = \sum_{v_j \in V} \sum_{\substack{u_i \in U \\ u_k \in U \\ y_{ij} \geq y_{kj}}} H(f(u_i,v_j) - f(u_k,v_j))\,, \tag{4.10}$$

with $H(\cdot)$ the Heaviside step function. This loss function quantifies the number of mismatches in each ranking of the objects in U for all $v \in V$. As Eq. (4.10) is neither convex nor differentiable, we will instead optimize an approximation. To motivate this approximation, we follow the rationale that if two dyads (u,v) and (u',v) have a similar label, their respective predictions should also be similar. This can be translated into the following loss function:

$$\mathcal{L}(f,S) = \sum_{v_j \in V} \sum_{\substack{u_i \in U \\ u_k \in U}} (y_{ij} - y_{kj} - f(u_i,v_j) + f(u_k,v_j))^2\,, \tag{4.11}$$

which can be interpreted as a convex and differentiable approximation of Eq. (4.10).

The loss function of Eq. (4.11) can be represented in matrix form using the centring matrix

$$\mathbf{C}_l = \mathbb{I} - \frac{1}{l}\mathbb{J}_{l\times l}^{\top}\,,$$

with $\mathbb{J}_{l\times l} \in \mathbb{R}^{l\times l}$ an $l \times l$ matrix for which every entry is equal to 1. The matrix $\mathbf{C}_l$ is a symmetric idempotent $l \times l$ matrix and multiplying it with a vector substracts the mean of the vector entries from all elements of the vector. Moreover, the following equality holds for any vector $\mathbf{c} = [c_i] \in \mathbb{R}^l$:

$$\frac{1}{2l^2}\sum_{i,j=1}^{l}(c_i - c_j)^2 = \frac{1}{l}\mathbf{c}^{\top}\mathbf{C}_l\mathbf{c}\,.$$

For a complete dataset and using a model of the form of Eq. (4.7), loss function (4.11) can be written as

$$(\mathrm{vec}(\mathbf{Y}) - (\mathbf{G} \otimes \mathbf{K})\mathrm{vec}(\mathbf{A}^{\mathrm{CR}}))^\top (\mathbb{I} \otimes \mathbf{C}_m)(\mathrm{vec}(\mathbf{Y}) - (\mathbf{G} \otimes \mathbf{K})\mathrm{vec}(\mathbf{A}^{\mathrm{CR}}))\,,$$

with $\mathbf{A}^{\mathrm{CR}}$ the matrix containing the dual parameters for the conditional ranking model. If L_2 regularization is applied, analogous to Eq. (4.4), the dual parameters that minimize this regularized approximation of the ranking loss can be obtained by solving the following linear system:

$$(\mathbf{G} \otimes (\mathbf{C}_m\mathbf{K}) + \lambda\mathbb{I})\mathrm{vec}(\mathbf{A}^{\mathrm{CR}}) = \mathrm{vec}(\mathbf{C}_m\mathbf{Y})\,.$$

Note that even though $\mathbf{C}_m\mathbf{K}$, the product of two symmetric matrices, is not necessarily symmetric, it is diagonalizable and its eigenvalues are nonnegative, see Property A.4. Hence, solving this system also has a time complexity of again roughly $\mathcal{O}(m^3 + q^3)$.

4.4.4 Two-step kernel ridge regression

Clearly, independent-task ridge regression can generalize to new instances, but not to new tasks as no dependence between these tasks is encoded in the model. Kronecker KRR on the other hand can be used for all four prediction settings depicted in Figure 4.1. But since our definition of 'instances' and 'tasks' is purely conventional, nothing is preventing us from building a model using the kernel function $g(\cdot, \cdot)$ to generalize to new tasks for the same instances. By combining two ordinary KRRs, one for generalizing to new instances and one that generalizes to new tasks, one can indirectly predict for new dyads.

More formally, suppose one wants to make a prediction for the dyad (u, v). Let $\mathbf{k} \in \mathbb{R}^m$ denote the vector of instance kernel evaluations between the instances in the training set and an instance in the test set, i.e. $\mathbf{k}(u) = (k(u, u_1), \ldots, k(u, u_m))^\top$. Likewise, $\mathbf{g} \in \mathbb{R}^q$ represents the vector of task kernel evaluations between the target task

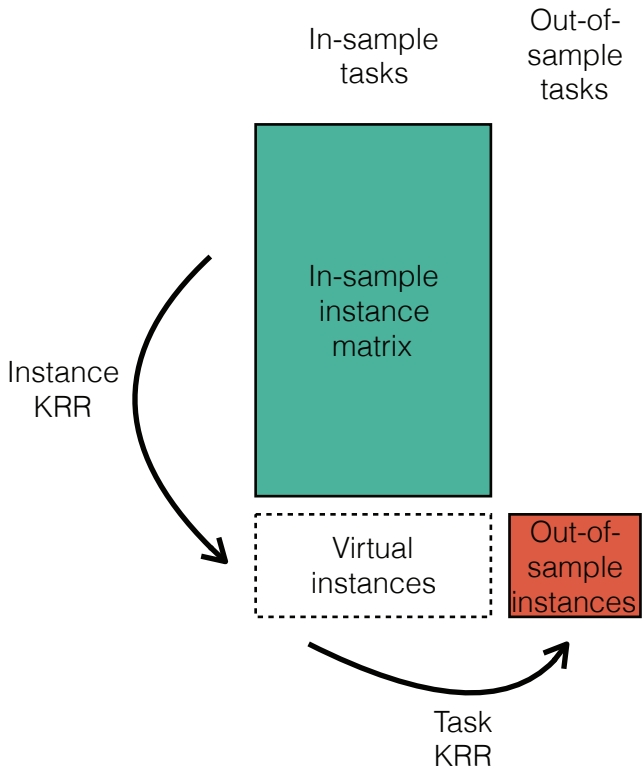


Figure 4.2: Principle of two-step KRR. In a first step, a virtual prediction is made for the auxiliary tasks for new instances using a first KRR model. A second KRR model is trained using this data and this model is used to make predictions for new tasks.

and the auxiliary tasks, i.e. $\mathbf{g}(v) = (g(v, v_1), \ldots, g(v, v_q))^\top$. Based on the parameters found by solving Eq. (4.2), we can make a prediction for the new instance u for all the auxiliary tasks:

$$\mathbf{f}_V(u) = \mathbf{k}^\top \left(\mathbf{K} + \lambda_u \mathbb{I}\right)^{-1} \mathbf{Y}\,, \tag{4.12}$$

with λ_u the specific regularization parameter for the instances. This vector of predictions $\mathbf{f}_V(u)$ can be used as a set of labels in an intermediate step to train a second model for generalizing to new tasks for

the same instance. Thus, using the task kernel and a regularization parameter for the tasks λ_v, one obtains:

$$f^{\mathrm{TS}}(u, v) = \mathbf{g}^\top \left(\mathbf{G} + \lambda_v \mathbb{I}\right)^{-1} \mathbf{f}_V(u)^\top ,$$

or, by making use of Eq. (4.12), the prediction is given by

$$f^{\mathrm{TS}}(u, v) = \mathbf{k}^\top \left(\mathbf{K} + \lambda_u \mathbb{I}\right)^{-1} \mathbf{Y} \left(\mathbf{G} + \lambda_v \mathbb{I}\right)^{-1} \mathbf{g} \tag{4.13}$$

$$= \mathbf{k}^\top \mathbf{A}^{\mathrm{TS}} \mathbf{g} , \tag{4.14}$$

with $\mathbf{A}^{\mathrm{TS}}$ the dual parameters. I call this method two-step (TS) KRR and it was independently proposed as "embarrassingly simple zero-shot learning" by Romera-Paredes and Torr (2015). It is represented in Figure 4.2. Superficially, this approach resembles alternating least-squares (Zachariah and Sundin, 2012), though the latter is an iteratively trained model mainly used for Setting A to obtain a low-rank representation. The two-step method, on the other hand, has a closed-form solution for the model parameters and allows for some computational techniques discussed later in this work. Two-step KRR can be used for any of the settings discussed in Section 4.3. Note that in practice there is no need to explicitly calculate $\mathbf{f}_V$, nor does it matter if in the first step one uses a model for new tasks and in the second step for instances, or the other way around.

This model can be cast in a similar form as the pairwise prediction function of Eq. (4.7) by making use of Property A.5. Thus for two-step KRR the parameters are given by

$$\mathbf{A}^{\mathrm{TS}} = \left(\mathbf{K} + \lambda_u \mathbb{I}\right)^{-1} \mathbf{Y} \left(\mathbf{G} + \lambda_v \mathbb{I}\right)^{-1} . \tag{4.15}$$

The time complexity for two-step KRR is the same as for Kronecker KRR: $\mathcal{O}(m^3 + q^3)$. The parameters can also be found by computing the eigenvalue decomposition of the two Gram matrices. Starting from these eigenvalue decompositions, it is possible to directly obtain the

dual parameters for any values of the regularization hyperparameters λ_u and λ_v. Because of its conceptual simplicity, it is quite straightforward to use two-step KRR for certain cases when the label matrix is not complete, in contrast to the Kronecker KRR, see experiments in Section 4.7. The computational advantages of this method will be discussed in Section 4.6.

4.5 Theoretical considerations

In this section I will show that two-step KRR can be seen as using KRR with special kinds of pairwise kernel matrices, depending on the prediction setting. I will show that Setting A is a transductive setting, while Setting D is merely a special case of (Kronecker) KRR. I will also study the different learning algorithms from a spectral filtering point of view, showing that two-step KRR uses a special decomposable filter. From these observations I will prove the universality and admissibility of the methods. Finally, I will briefly discuss pairwise kernels for learning relations and kernels when no features of the objects are available.

4.5.1 Equivalence between two-step and other kernel ridge regression methods

The relation between two-step ridge regression and independent-task ridge regression is given in the following theorem.

THEOREM 4.1 (Setting B). *When the Gram matrix of the tasks* $\mathbf{G}$ *is full rank and* λ_v *is set to zero, independent-task KRR and two-step KRR return the same predictions for any given training task:*

$$f_j^{\mathrm{IT}}(\cdot) \equiv f^{\mathrm{TS}}(\cdot, v_j)\,.$$

Table 4.1: Overview of the different learning methods discussed in this chapter. For each method, the dual form of the prediction function and the linear system for estimating the parameters is given. Here, the data is assumed to be complete.

KRR method	Dual prediction function	System for dual parameters
Independent-task	$f_j^{\mathrm{IT}}(u) = \sum_{i=1}^{m} a_{ij}^{\mathrm{IT}} k(u, u_i)$	$(\mathbf{K} + \lambda_u \mathbb{I})\, \mathbf{A}^{\mathrm{IT}} = \mathbf{Y}$
Kronecker kernel	$f^{\mathrm{KK}}(u, v) = \sum_{i=1}^{m} \sum_{j=1}^{q} a_{ij}^{\mathrm{KK}} k(u, u_i) g(v, v_j)$	$(\mathbf{G} \otimes \mathbf{K} + \lambda \mathbb{I})\, \mathrm{vec}(\mathbf{A}^{\mathrm{KK}}) = \mathrm{vec}(\mathbf{Y})$
Ordinary Kronecker kernel least-squares	$f^{\mathrm{OKKLS}} = \sum_{i=1}^{m} \sum_{j=1}^{q} a_{ij}^{\mathrm{OKKLS}} k(u, u_i) g(v, v_j)$	$(\mathbf{G} \otimes \mathbf{K})\mathrm{vec}(\mathbf{A}^{\mathrm{OKKLS}}) = \mathrm{vec}(\mathbf{Y})$
Conditional ranking	$f^{\mathrm{CR}}(u, v) = \sum_{i=1}^{m} \sum_{j=1}^{q} a_{ij}^{\mathrm{CR}} k(u, u_i) g(v, v_j)$	$(\mathbf{G} \otimes (\mathbf{C}_m \mathbf{K}) + \lambda \mathbb{I})\, \mathrm{vec}(\mathbf{A}^{\mathrm{CR}} = \mathrm{vec}(\mathbf{C}_m \mathbf{Y})$
Two-step	$f^{\mathrm{TS}}(u, v) = \sum_{i=1}^{m} \sum_{j=1}^{q} a_{ij}^{\mathrm{TS}} k(u, u_i) g(v, v_j)$	$(\mathbf{K} + \lambda_u \mathbb{I})\, \mathbf{A}^{\mathrm{TS}}\, (\mathbf{G} + \lambda_v \mathbb{I}) = \mathbf{Y}$

Proof. The prediction for the independent-task KRR is given by:

$$f_j^{\mathrm{IT}}(u) = [\mathbf{k}^\top(\mathbf{K} + \lambda_u \mathbb{I})^{-1}\mathbf{Y}]_j \,.$$

For two-step KRR, it follows from Eq. (4.13) that

$$\begin{aligned} f_j^{\mathrm{TS}}(u) &= [\mathbf{k}^\top(\mathbf{K} + \lambda_u \mathbb{I})^{-1}\mathbf{Y}\mathbf{G}^{-1}\mathbf{G}]_j \\ &= [\mathbf{k}^\top(\mathbf{K} + \lambda_u \mathbb{I})^{-1}\mathbf{Y}]_j \,. \end{aligned}$$

□

When $\mathbf{G}$ is singular, the q outputs for the different tasks are projected on a lower-dimensional subspace by two-step KRR. This means that a dependence between the tasks is enforced, even with $\lambda_v = 0$.

The connection between two-step and Kronecker KRR is established by the following results.

THEOREM 4.2 (Setting A). *Consider the following pairwise kernel matrix:*

$$\boldsymbol{\Xi} = \mathbf{G} \otimes \mathbf{K} \left(\lambda_u \lambda_v \mathbf{I} \otimes \mathbf{I} + \lambda_v \mathbf{I} \otimes \mathbf{K} + \lambda_u \mathbf{G} \otimes \mathbf{I}\right)^{-1} \,.$$

The predictions for the training data $\mathbf{F}$ *using pairwise KRR (Eq. (4.8)) with the above pairwise kernel and regularization parameter* $\lambda = 1$ *correspond to those obtained with two-step KRR using the kernel matrices* $\mathbf{K}$, $\mathbf{G}$ *with respective regularization parameters* λ_u *and* λ_v.

Proof. I will write the corresponding empirical risk minimization of Eq. (4.4) from the perspective of value regularization. Since Setting A is an imputation setting, we directly search for the optimal predicted label matrix $\mathbf{F}$, rather than the optimal parameter matrix. Starting from the objective function for KRR, the predictions for the training data are obtained through minimizing the following variational

problem:

$$
\begin{aligned}
J'(\mathbf{F}) &= \mathrm{vec}(\mathbf{F}-\mathbf{Y})^\top \mathrm{vec}(\mathbf{F}-\mathbf{Y}) + \mathrm{vec}(\mathbf{F})^\top \boldsymbol{\Xi}^{-1} \mathrm{vec}(\mathbf{F}) \qquad (4.16)\\
&= \mathrm{vec}(\mathbf{F}-\mathbf{Y})^\top \mathrm{vec}(\mathbf{F}-\mathbf{Y}) \\
&\quad + \mathrm{vec}(\mathbf{F})^\top \left(\mathbf{G}\otimes\mathbf{K}\left(\lambda_u\lambda_v \mathbf{I}\otimes\mathbf{I} + \lambda_v \mathbf{I}\otimes\mathbf{K} + \lambda_u \mathbf{G}\otimes\mathbf{I}\right)^{-1}\right)^{-1} \mathrm{vec}(\mathbf{F}) \\
&= \mathrm{vec}(\mathbf{F}-\mathbf{Y})^\top \mathrm{vec}(\mathbf{F}-\mathbf{Y}) \\
&\quad + \mathrm{vec}(\mathbf{F})^\top \left(\mathbf{G}^{-1}\otimes\mathbf{K}^{-1}\left(\lambda_u\lambda_v \mathbf{I}\otimes\mathbf{I} + \lambda_v \mathbf{I}\otimes\mathbf{K} + \lambda_u \mathbf{G}\otimes\mathbf{I}\right)\right) \mathrm{vec}(\mathbf{F}) \\
&= \mathrm{vec}(\mathbf{F}-\mathbf{Y})^\top \mathrm{vec}(\mathbf{F}-\mathbf{Y}) \\
&\quad + \mathrm{vec}(\mathbf{F})^\top \left(\lambda_u\lambda_v \mathbf{G}^{-1}\otimes\mathbf{K}^{-1} + \lambda_u \mathbf{I}\otimes\mathbf{K}^{-1} + \lambda_v \mathbf{G}^{-1}\otimes\mathbf{I}\right) \mathrm{vec}(\mathbf{F}) \\
&= \mathrm{tr}((\mathbf{F}-\mathbf{Y})^\top(\mathbf{F}-\mathbf{Y}) + \lambda_u\lambda_v \mathbf{F}^\top\mathbf{K}^{-1}\mathbf{F}\mathbf{G}^{-1} \\
&\quad + \lambda_u \mathbf{F}^\top\mathbf{K}^{-1}\mathbf{F} + \lambda_v \mathbf{F}^\top\mathbf{F}\mathbf{G}^{-1})\,.
\end{aligned}
$$

The derivative with respect to $\mathbf{F}$ is:

$$
\begin{aligned}
\frac{\partial J'(\mathbf{F})}{\partial \mathbf{F}} &= 2(\mathbf{F}-\mathbf{Y} + \lambda_u\lambda_v \mathbf{K}^{-1}\mathbf{F}\mathbf{G}^{-1} + \lambda_u \mathbf{K}^{-1}\mathbf{F} + \lambda_v \mathbf{F}\mathbf{G}^{-1}) \\
&= 2(\lambda_u \mathbf{K}^{-1} + \mathbf{I})\mathbf{F}(\lambda_v \mathbf{G}^{-1} + \mathbf{I}) - 2\mathbf{Y}\,.
\end{aligned}
$$

Setting it to zero and solving with respect to $\mathbf{F}$ yields:

$$
\begin{aligned}
\mathbf{F} &= (\lambda_u \mathbf{K}^{-1} + \mathbf{I})^{-1}\mathbf{Y}(\lambda_v \mathbf{G}^{-1} + \mathbf{I})^{-1} \\
&= \mathbf{K}(\mathbf{K} + \lambda_u \mathbf{I})^{-1}\mathbf{Y}(\mathbf{G} + \lambda_v \mathbf{I})^{-1}\mathbf{G}\,.
\end{aligned}
$$

Comparing with Eq. (4.15), we note that $\mathbf{F} = \mathbf{K}\mathbf{A}^{\mathrm{TS}}\mathbf{G}$, which proves the theorem. □

Here, I have assumed that $\mathbf{K}$ and $\mathbf{G}$ are invertible. The kernel $\boldsymbol{\Xi}$ can always be obtained as long as $\mathbf{K}$ and $\mathbf{G}$ are positive semi-definite. The relevance of the above theorem is that it formulates two-step KRR as an empirical risk minimization problem for Setting A (Eq. (4.16)). It is important to note that the pairwise kernel matrix $\boldsymbol{\Xi}$ only appears in the regularization term of this variational problem.

The loss function is only dependent on the predicted values $\mathbf{F}$ and the label matrix $\mathbf{Y}$. Using two-step KRR for Setting A when dealing with incomplete data is thus well defined. The empirical risk minimization problem of Eq. (4.16) can be modified so that the squared loss only takes the observed dyads into account:

$$J'(\mathbf{F}) = \sum_{(\mathbf{d},\mathbf{t},y)\in S} (y - f(\mathbf{d},\mathbf{t}))^2 + \mathrm{vec}(\mathbf{F})^\top \boldsymbol{\Xi}^{-1} \mathrm{vec}(\mathbf{F}) \,, \tag{4.17}$$

with S the training set of labeled dyads. In this case, one ends up with a transductive setting. This explains why Setting A is the most easy setting to predict for, as in transductive learning one only has to predict for a finite number of dyads known during training. This is in contrast to inductive learning in which the model has to generalize to make predictions for any new dyad, a harder problem (Chapelle et al., 2006). See Rifkin and Lippert (2007b); Johnson and Zhang (2008) for a more in-depth discussion.

Two-step and Kronecker KRR also coincide in an interesting way for zero-shot learning problems (e.g. the special case in which there is no labeled data available for the target task). This, in turn, allows us to show the consistency of two-step KRR via its universal approximation and spectral regularization properties. The theorem below shows the relation between two-step KRR and ordinary Kronecker KRR for Setting D.

THEOREM 4.3 (Setting D). *Let us consider a zero-shot learning setting with a complete training set. Let* $f^{\mathrm{TS}}(\cdot,\cdot)$ *be a model trained with two-step KRR and* $f^{\mathrm{OKKLS}}(\cdot,\cdot)$ *be a model trained with ordinary Kronecker kernel least-squares regression (OKKLS) using the following pairwise kernel function on* $\mathcal{U} \times \mathcal{V}$*:*

$$\Upsilon\left((u,v),(\bar{u},\bar{v})\right) = \left(k\left(u,\bar{u}\right) + \lambda_u \delta\left(u,\bar{u}\right)\right)\left(g\left(v,\bar{v}\right) + \lambda_v \delta\left(v,\bar{v}\right)\right) \tag{4.18}$$

where δ is the delta kernel whose value is 1 if the arguments are equal and 0 otherwise. Then for making predictions for instances $u \in \mathcal{U} \setminus U$ and tasks $v \in \mathcal{V} \setminus V$ not seen in the training set, it holds that $f^{\mathrm{TS}}(v,u) = f^{\mathrm{OKKLS}}(v,u)$.

Proof. From Eq. (4.13) we have the following dual model for prediction:

$$f^{\mathrm{TS}}(u,v) = \sum_{i=1}^{m}\sum_{j=1}^{q} a_{ij}^{\mathrm{TS}} k(u,u_i) g(v,v_j)\,,$$

with $\mathbf{A}^{\mathrm{TS}} = [a_{ij}^{\mathrm{TS}}]$ the matrix of parameters. Similarly, the dual representation of the OKKLS (see Eq. (4.7)), using a parametrization $\mathbf{A}^{\mathrm{OKKLS}} = [a_{ij}^{\mathrm{OKKLS}}]$, is given by

$$\begin{aligned} f^{\mathrm{OKKLS}}(u,v) &= \sum_{i=1}^{m}\sum_{j=1}^{q} a_{ij}^{\mathrm{OKKLS}} \Upsilon\left((u,v),(u_i,v_j)\right) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{q} a_{ij}^{\mathrm{OKKLS}} (k(u,u_i) + \lambda_u \delta(u,u_i))(g(v,v_j) \\ &\qquad + \lambda_v \delta(v,v_j))) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{q} a_{ij}^{\mathrm{OKKLS}} k(u,u_i) g(v,v_j)\,. \end{aligned}$$

In the last step we used the fact that $u \neq u_i$ and $v \neq v_j$ to drop the delta kernels. Hence, we need to show that $\mathbf{A}^{\mathrm{TS}} = \mathbf{A}^{\mathrm{OKKLS}}$.

By Eq. (4.15) and denoting $\widetilde{\mathbf{G}} = (\mathbf{G} + \lambda\mathbb{I})^{-1}$ and $\widetilde{\mathbf{K}} = (\mathbf{K} + \lambda\mathbb{I})^{-1}$, we observe that the model parameters $\mathbf{A}^{\mathrm{TS}}$ of the two-step model can also be obtained in the following closed form:

$$\mathbf{A}^{\mathrm{TS}} = \widetilde{\mathbf{K}}\mathbf{Y}\widetilde{\mathbf{G}}\,. \tag{4.19}$$

The kernel matrix of Υ for Setting D can be expressed as: $\boldsymbol{\Upsilon} = (\mathbf{G} + \lambda_v \mathbb{I}) \otimes (\mathbf{K} + \lambda_u \mathbb{I})$. The OKKLS problem with kernel Υ being

$$\begin{aligned}&\mathrm{vec}(\mathbf{A}^{\mathrm{OKKLS}}) = \\ &\quad \underset{\mathbf{A}^{\mathrm{OKKLS}} \in \mathbb{R}^{m \times q}}{\arg\min} \left(\mathrm{vec}(\mathbf{Y}) - \boldsymbol{\Upsilon}\mathrm{vec}(\mathbf{A}^{\mathrm{OKKLS}})\right)^{\top} \left(\mathrm{vec}(\mathbf{Y}) - \boldsymbol{\Upsilon}\mathrm{vec}(\mathbf{A}^{\mathrm{OKKLS}})\right) ,\end{aligned}$$

its minimizer can be expressed as

$$\begin{aligned}\mathrm{vec}(\mathbf{A}^{\mathrm{OKKLS}}) &= \boldsymbol{\Upsilon}^{-1}\mathrm{vec}(\mathbf{Y}) = \left((\mathbf{G} + \lambda_v \mathbb{I})^{-1} \otimes (\mathbf{K} + \lambda_u \mathbb{I})^{-1}\right) \mathrm{vec}(\mathbf{Y}) \\ &= \mathrm{vec}\left((\mathbf{K} + \lambda_u \mathbb{I})^{-1} \mathbf{Y} (\mathbf{G} + \lambda_v \mathbb{I})^{-1}\right) = \mathrm{vec}\left(\widetilde{\mathbf{K}} \mathbf{Y} \widetilde{\mathbf{G}}\right) .\end{aligned} \tag{4.20}$$

Here, we again make use of Property A.5. From Eq. (4.20) it then follows that $\mathbf{A}^{\mathrm{TS}} = \mathbf{A}^{\mathrm{OKKLS}}$, which proves the theorem. □

4.5.2 Universality of the Kronecker product pairwise kernel

Here I consider the universal approximation properties of Kronecker KRR and, by Theorems and , of two-step KRR. This is a necessary step in showing the consistency of this method. First, recall the concept of universal kernel functions.

Definition 4.1. **(Steinwart, 2002)** *A continuous kernel $k(\cdot, \cdot)$ on a compact metric space $\mathcal{X}$ (i.e. $\mathcal{X}$ is closed and bounded) is called universal if the reproducing kernel Hilbert space (RKHS) induced by $k(\cdot, \cdot)$ is dense in $C(\mathcal{X})$, where $C(\mathcal{X})$ is the space of all continuous functions $f : \mathcal{X} \to \mathbb{R}$.*

The universality property indicates that the hypothesis space induced by a universal kernel can approximate any continuous function on the input space $\mathcal{X}$ to be learned arbitrarily well, given that the available set of training data is large and representative enough, and

the learning algorithm can efficiently find this approximation from the hypothesis space (Steinwart, 2002). In other words, the learning algorithm is consistent in the sense that, informally put, the hypothesis learned by it gets closer to the function to be learned while the size of the training set gets larger. The consistency properties of two-step KRR are considered in more detail in Section 4.5.3.

Next, I consider the universality of the Kronecker product pairwise kernel. The following result is a straightforward modification of some of the existing results in the literature (e.g. Waegeman et al. (2012)), but it is presented here for self-sufficiency. This theorem is mainly related to Setting D, while it also covers the other settings as special cases.

THEOREM 4.4. *The kernel* $\Gamma^{\mathrm{KK}}((\cdot,\cdot),(\cdot,\cdot))$ *on* $\mathcal{U}\times\mathcal{V}$ *defined in Eq. (4.6) is universal if the instance kernel* $k(\cdot,\cdot)$ *on* $\mathcal{U}$ *and the task kernel* $g(\cdot,\cdot)$ *on* $\mathcal{V}$ *are both universal.*

Proof. Let us define

$$\mathcal{A}\otimes\mathcal{B}=\{t \mid t(u,v)=u(u)v(v), u\in\mathcal{A}, v\in\mathcal{B}\} \tag{4.21}$$

for compact metric spaces $\mathcal{U}$ and $\mathcal{V}$ and sets of functions $\mathcal{A}\subset C(\mathcal{U})$ and $\mathcal{B}\subset C(\mathcal{V})$. I observe that the RKHS of the kernel Γ can be written as $\mathcal{H}(k)\otimes\mathcal{H}(g)$, where $\mathcal{H}(k)$ and $\mathcal{H}(g)$ are the RKHS of the kernels $k(\cdot,\cdot)$ and $g(\cdot,\cdot)$, respectively.

Let $\epsilon>0$ and let $t\in C(\mathcal{U})\otimes C(\mathcal{V})$ be an arbitrary function which can, according to Eq. (4.21), be written as $t(u,v)=u(u)v(v)$, where $u\in C(\mathcal{U})$ and $v\in C(\mathcal{V})$. By definition of the universality property, $\mathcal{H}(k)$ and $\mathcal{H}(g)$ are dense in $C(\mathcal{U})$ and $C(\mathcal{V})$, respectively. Therefore, there exist functions $\bar{u}\in\mathcal{H}(k)$ and $\bar{v}\in\mathcal{H}(g)$ such that

$$\max_{u\in\mathcal{U}}|\bar{u}(u)-u(u)|\leq\bar{\epsilon},\qquad\max_{v\in\mathcal{V}}|\bar{v}(v)-v(v)|\leq\bar{\epsilon}\,,$$

where $\bar{\epsilon}$ is a constant for which it holds that

$$\max_{u\in\mathcal{U},v\in\mathcal{V}} \left\{ |\bar{\epsilon}\, u(u)| + |\bar{\epsilon}\, v(v)| + \bar{\epsilon}^2 \right\} \leq \epsilon\,.$$

Note that, according to the extreme value theorem, the maximum exists due to the compactness of $\mathcal{U}$ and $\mathcal{V}$ and the continuity of the functions $u(\cdot)$ and $v(\cdot)$. Now we have

$$\begin{aligned}
&\max_{u\in\mathcal{U},v\in\mathcal{V}} |t(u,v) - \bar{u}(u)\bar{v}(v)| \\
&\leq \max_{u\in\mathcal{U},v\in\mathcal{V}} \left\{ |t(u,v) - u(u)v(v)| + |\bar{\epsilon}\, u(u)| + |\bar{\epsilon}\, v(v)| + \bar{\epsilon}^2 \right\} \\
&= \max_{u\in\mathcal{U},v\in\mathcal{V}} \left\{ |\bar{\epsilon}\, u(u)| + |\bar{\epsilon}\, v(v)| + \bar{\epsilon}^2 \right\} \leq \epsilon,
\end{aligned}$$

which confirms the density of $\mathcal{H}(k) \otimes \mathcal{H}(g)$ in $C(\mathcal{U}) \otimes C(\mathcal{V})$.

The space $\mathcal{U}\times\mathcal{V}$ is compact if both $\mathcal{U}$ and $\mathcal{V}$ are compact according to Tikhonov's theorem. It is straightforward to see that $C(\mathcal{U}) \otimes C(\mathcal{V})$ is a subalgebra of $C(\mathcal{U} \times \mathcal{V})$, it separates points in $\mathcal{U} \times \mathcal{V}$, it vanishes at no point of $C(\mathcal{U} \times \mathcal{V})$, and it is therefore dense in $C(\mathcal{U} \times \mathcal{V})$ due to the Stone-Weierstraß theorem. Thus, $\mathcal{H}(k) \otimes \mathcal{H}(g)$ is also dense in $C(\mathcal{U} \times \mathcal{V})$, and Γ is a universal kernel on $\mathcal{U} \times \mathcal{V}$. □

4.5.3 Spectral interpretation

In Theorem 4.5.2 I have shown the relation between single-task, two-step, and Kronecker KRR for Setting D. In this section I will study the difference between single-task, Kronecker and two-step KRR from the point of view of spectral regularization. The above shown universal approximation properties of this kernel are also connected to the consistency properties of two-step KRR, as is elaborated in more detail in this section.

Learning by spectral regularization originates from the theory of ill-posed problems. This paradigm is well studied in domains such as image analysis (Bertero and Boccacci, 1998) and, more recently, in machine learning – e.g. Lo Gerfo et al. (2008). Here, one wants to

find the parameters $\boldsymbol{\alpha}$ of the data-generating process given a set of noisy measurements $\mathbf{y}$ such that

$$\boldsymbol{\Gamma}\boldsymbol{\alpha} \approx \mathbf{y}\,, \tag{4.22}$$

with $\boldsymbol{\Gamma}$ a Gram matrix with eigenvalue decomposition $\boldsymbol{\Gamma} = \mathbf{W}\boldsymbol{\Lambda}\mathbf{W}^\top$. At first glance, one can find the parameters $\boldsymbol{\alpha}$ by inverting $\boldsymbol{\Gamma}$:

$$\begin{aligned}\boldsymbol{\alpha} &= \boldsymbol{\Gamma}^{-1}\mathbf{y} \\ &= \mathbf{W}\boldsymbol{\Lambda}^{-1}\mathbf{W}^\top\mathbf{y}\,.\end{aligned}$$

If $\boldsymbol{\Gamma}$ has small eigenvalues, the inverse becomes highly unstable: small changes in the feature description of the label vector will lead to huge changes in $\boldsymbol{\alpha}$. Spectral regularization deals with this problem by generalizing the inverse by a so-called filter function to make solving Eq. (4.22) well-posed. The following definition of a spectral filter-based regularizer is standard in the machine learning literature (see e.g. Lo Gerfo et al. (2008) and references therein). Note that we assume $\Gamma((\cdot,\cdot),(\cdot,\cdot))$ being bounded with $\kappa > 0$ such that $\sup_{\mathbf{x}\in\mathcal{X}} \sqrt{\Gamma(\mathbf{x},\mathbf{x})} \leq \kappa$, assuring that the eigenvalues of the Gram matrix $\boldsymbol{\Gamma}$ are in $[0,\kappa^2]$.

Definition 4.2 (Admissible regularizer). *A function $\varphi_\lambda : [0,\kappa^2] \to \mathbb{R}$, parameterized by $0 < \lambda \leq \kappa^2$, is an admissible regularizer if there exist constants $D, B, \gamma \in \mathbb{R}$ and $\bar{\nu}, \gamma_\nu > 0$ such that*

$$\sup_{0<\sigma\leq\kappa^2} |\sigma\varphi_\lambda(\sigma)| \leq D, \quad \sup_{0<\sigma\leq\kappa^2} |\varphi_\lambda(\sigma)| \leq \frac{B}{\lambda}, \quad \sup_{0<\sigma\leq\kappa^2} |1-\sigma\varphi_\lambda(\sigma)| \leq \gamma\,,$$

$$\textit{and} \sup_{0<\sigma\leq\kappa^2} \frac{\lambda^\nu}{\sigma^\nu}|1-\sigma\varphi_\lambda(\sigma)| \leq \gamma_\nu, \quad \textit{for any } \nu \in]0,\bar{\nu}]\,,$$

where the constant γ_ν does not depend on λ.

The constant $\bar{\nu}$ is in the literature called the qualification of the regularizer and it is related to the consistency properties of the learning method as described in more detail below.

The spectral filter is a matrix function that acts as a stabilized generalization of a matrix inverse. Hence, Eq. (4.22) can be solved by

$$\begin{aligned}\boldsymbol{\alpha} &= \varphi_\lambda(\boldsymbol{\Gamma})\mathbf{y} \\ &= \mathbf{W}\varphi_\lambda(\boldsymbol{\Lambda})\mathbf{W}^\top \mathrm{vec}(\mathbf{Y})\,.\end{aligned}$$

Similarly, the noisy measurements can be filtered to obtain a better estimation of the true labels:

$$\begin{aligned}\mathbf{f} &= \boldsymbol{\Gamma}\boldsymbol{\alpha} \\ &= \mathbf{W}\boldsymbol{\Lambda}\mathbf{W}^\top\mathbf{W}\varphi_\lambda(\boldsymbol{\Lambda})\mathbf{W}^\top\mathrm{vec}(\mathbf{Y}) \\ &= \mathbf{W}\boldsymbol{\Lambda}\varphi_\lambda(\boldsymbol{\Lambda})\mathbf{W}^\top\mathrm{vec}(\mathbf{Y})\,.\end{aligned}$$

The spectral interpretation allows for using a more general form of the hat matrix (Eq. (4.9)):

$$\mathbf{H}^\Gamma = \mathbf{W}\boldsymbol{\Lambda}\varphi_\lambda(\boldsymbol{\Lambda})\mathbf{W}^\top\,.$$

For example, the filter function corresponding to the Tikhonov regularization, as used for independent-task KRR, is given by

$$\varphi_\lambda^{\mathrm{TIK}}(\sigma) = \frac{1}{\sigma+\lambda}\,,$$

with the ordinary least-squares approach corresponding to $\lambda = 0$. Several other learning approaches, such as spectral cut-off, iterated Tikhonov and $L2$ Boosting, can also be expressed as filter functions, but cannot be expressed as a penalized empirical error minimization problem analogous to Eq. (4.4) (Lo Gerfo et al., 2008). The spectral interpretation can also be used to motivate novel learning algorithms.

Many authors have expanded this framework to multi-task settings, e.g. Argyriou et al. (2007, 2010); Baldassarre et al. (2012). I shall translate the pairwise learning methods from Section 4.4 to this spectral regularization context. Let us denote the eigenvalue decomposition of the instance and task kernel matrices as

$$\mathbf{K} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^\top \qquad \text{and} \qquad \mathbf{G} = \mathbf{V}\mathbf{S}\mathbf{V}^\top .$$

Let $\mathbf{u}_i$ denote the i-th eigenvector of $\mathbf{K}$ and $\mathbf{v}_j$ the j-th eigenvector of $\mathbf{G}$. The eigenvalues of the kernel matrix obtained with the Kronecker product kernel on a complete training set can be expressed as the Kronecker product $\boldsymbol{\Lambda} = \mathbf{S} \otimes \boldsymbol{\Sigma}$ of the eigenvalues $\boldsymbol{\Sigma}$ and $\mathbf{S}$ of the instance and task kernel matrices. For the models of this work, it is opportune to define a pairwise filter function over the representation of the instances and tasks.

Both of the factor kernels are assumed to be bounded, and hence we can write that all the eigenvalues ς of the Kronecker product kernel can be factorized as the product of the eigenvalues of the instance and task kernels as follows:

$$\varsigma = \sigma s \qquad \text{with } 0 \leq \sigma, s \leq a\sqrt{\varsigma} \text{ and } 1 \leq a < \infty\,, \tag{4.23}$$

where σ, s denote the eigenvalues of the factor kernels and a the constant determined as the product of $\sup_{u\in\mathcal{U}} \sqrt{k(u,u)}$ and $\sup_{v\in\mathcal{V}} \sqrt{g(v,v)}$.

Definition 4.3 (Pairwise spectral filter). *We say that a function $\varphi_\lambda : [0, \kappa^2] \to \mathbb{R}$, parameterized by $0 < \lambda \leq \kappa^2$, is a pairwise spectral filter if it can be written as*

$$\varphi_\lambda(\varsigma) = \vartheta_\lambda(\sigma, s)$$

for some function $\vartheta_\lambda : [0, a\sqrt{\varsigma}]^2 \to \mathbb{R}$ with $1 \leq a < \infty$, and it is an admissible regularizer for all possible factorizations of the eigenvalues as in Eq. (4.23).

Since the eigenvalues of a Kronecker product of two matrices are just the scalar product of the eigenvalues of the matrices, the filter function for Kronecker KRR is given by

$$\vartheta_\lambda^{\mathrm{KK}}(s,\sigma) = \varphi_\lambda^{\mathrm{TIK}}(\sigma s) = \frac{1}{(\sigma s + \lambda)}\,, \tag{4.24}$$

where σ and s are the eigenvalues of $\mathbf{K}$ and $\mathbf{G}$, respectively. The admissibility of this filter is a well-known result, since it is simply the Tikhonov regularizer for the pairwise Kronecker product kernel.

Instead of considering two-step KRR from the kernel point of view, one can also cast it into the spectral filtering regularization framework. We start from Eq. (4.15) in vectorized form:

$$\begin{aligned}
\mathrm{vec}(\mathbf{A}) &= \left((\mathbf{G} + \lambda_v \mathbb{I})^{-1} \otimes (\mathbf{K} + \lambda_u \mathbb{I})^{-1}\right) \mathrm{vec}(\mathbf{Y}) \\
&= \left((\mathbf{V}\mathbf{S}\mathbf{V}^\top + \lambda_v \mathbb{I})^{-1} \otimes (\mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top + \lambda_u \mathbb{I})^{-1}\right) \mathrm{vec}(\mathbf{Y}) \\
&= \left((\mathbf{V}\varphi_{\lambda_v}^{\mathrm{TIK}}(\mathbf{S})\mathbf{V}^\top) \otimes (\mathbf{U}\varphi_{\lambda_u}^{\mathrm{TIK}}(\mathbf{\Sigma})\mathbf{U}^\top)\right) \mathrm{vec}(\mathbf{Y}) \\
&= \left((\mathbf{V} \otimes \mathbf{U})(\varphi_{\lambda_v}^{\mathrm{TIK}}(\mathbf{S}) \otimes \varphi_{\lambda_u}^{\mathrm{TIK}}(\mathbf{\Sigma}))(\mathbf{V} \otimes \mathbf{U})^\top\right) \mathrm{vec}(\mathbf{Y})\,.
\end{aligned}$$

Hence, one can interpret two-step KRR with a complete training set for Setting D as a spectral filtering regularization based learning algorithm that uses the pairwise Kronecker product kernel with the following filter function.

$$\begin{aligned}
\vartheta_\lambda^{\mathrm{TS}}(s,\sigma) &= \varphi_{\lambda_v}^{\mathrm{TIK}}(s)\varphi_{\lambda_u}^{\mathrm{TIK}}(\sigma) \\
&= \frac{1}{(\sigma + \lambda_u)(s + \lambda_v)} \\
&= \frac{1}{\sigma s + \lambda_v \sigma + \lambda_u + \lambda_v \lambda_u}\,.
\end{aligned} \tag{4.25}$$

The validity of this filter is characterized by the following theorem.

Theorem 4.5. *The filter function $\vartheta_\lambda^{\mathrm{TS}}(\cdot,\cdot)$ is admissible with $D = B = \gamma = 1$, $\gamma_\nu = 2ab$, and has qualification $\bar{\nu} = \frac{1}{2}$ for all factorizations*

of ς *and* λ *as*

$$\varsigma = \sigma s \text{ and } \lambda = \lambda_v \lambda_u \quad \textit{with } 0 \leq \sigma, s \leq a\sqrt{\varsigma} \text{ and } 0 < \lambda_v, \lambda_u \leq b\sqrt{\lambda}\,, \tag{4.26}$$

where $1 \leq a, b < \infty$ *are constants that do not depend on* λ *or* ς*.*

Proof. Let us recollect the last condition in Definition 4.2:

$$\sup_{0<\varsigma\leq\kappa^2} \frac{\varsigma^\nu}{\lambda^\nu} |1 - \varsigma\varphi_\lambda(\varsigma)| \leq \gamma_\nu, \quad \text{for any } \nu \in]0, \bar{\nu}]\,,$$

where γ_ν does not depend on λ. In order to show this for all cases covered by Eq. (4.26), I rewrite the condition by taking the supremum with respect to the factorizations of ς and λ given the constants a and b:

$$\sup_{\substack{0<\varsigma\leq\kappa^2 \\ 0<\lambda_v,\lambda_u\leq b\sqrt{\lambda} \\ 0<\sigma,s\leq a\sqrt{\varsigma}}} \frac{\varsigma^\nu}{\lambda^\nu} \left(1 - \frac{\varsigma}{\varsigma + \lambda_v\sigma + \lambda_u s + \lambda}\right) \leq \gamma_\nu, \qquad \text{for any } \nu \in]0, \bar{\nu}]\,.$$

The left-hand side then becomes

$$\sup_{0<\varsigma\leq\kappa^2} \frac{\varsigma^\nu}{\lambda^\nu} \left(1 - \frac{\varsigma}{\varsigma + 2ab\sqrt{\lambda}\sqrt{\varsigma} + \lambda}\right) = \sup_{0<\varsigma\leq\kappa^2} \left(\frac{2ab\lambda^{\frac{1}{2}-\nu}\varsigma^{\nu+\frac{1}{2}} + \lambda^{1-\nu}\varsigma^\nu}{\varsigma + 2ab\sqrt{\lambda}\sqrt{\varsigma} + \lambda}\right).$$

By checking the extreme values of the latter expression with respect to $(\varsigma, \lambda, \nu)$ using standard differential calculus, we observe that it is bounded by $\gamma_\nu = 2ab$ if $\nu \in]0, \frac{1}{2}]$. With values of $\bar{\nu}$ larger than $\frac{1}{2}$, the term $2ab\lambda^{\frac{1}{2}-\nu}\varsigma^{\nu+\frac{1}{2}}$ in the numerator grows arbitrarily while $\lambda \to 0$, and hence the qualification is $\bar{\nu} = \frac{1}{2}$. The other conditions in Definition 4.2 can be checked by direct computation. □

Thus, Eq. (4.25) can be positioned within the spectral filtering regularization framework with separate regularization parameter values for instances and tasks. In contrast to Eq. (4.24), the filter of two-step KRR can be factorized into a component for the tasks and

instances separately:

$$\vartheta_\lambda(s,\sigma) = \varphi_{\lambda_u}(\sigma)\varphi_{\lambda_v}(s)\,. \tag{4.27}$$

This decomposition gives rise to some computational shortcuts for performing cross-validation, as will be discussed in Section 4.6.

Providing a different regularization for instances and tasks also makes sense from a learning point of view. It is easy to imagine a setting in which the instance has a much larger influence in determining the label compared to the task or vice versa. For example, consider a collaborative filtering setting with the goal of recommending books for customers. Suppose that the sales of a book are for a very large part determined simply by being a bestseller novel or not, and less by individual customer's taste. When building a predictive model, one would give more freedom to the part concerning the books (hence a lower regularization). Less degrees of freedom are given to the inference of the user's personal task, as this is harder to learn and explains less of the variance in the preferences. This can be extended even further, by choosing specific filter functions separately for the instances and tasks tuned to the application at hand. In a pairwise setting, the filter function to perform independent-task KRR arises as a special case with $\lambda_v = 0$:

$$\vartheta^{\mathrm{IT}}_{\lambda_v}(s,\sigma) = \frac{1}{(\sigma+\lambda_u)s}\,,$$

when the task kernel is full rank (see Theorem 4.1).

Next, I analyze the consistency properties of two-step KRR in setting D, given the above results about the universality of the pairwise Kronecker product kernel and the spectral filtering interpretation of the method. Let $R(\cdot)$ denote the expected prediction error of a hypothesis f with respect to some unknown probability measure $\rho(\mathbf{x}, y)$

on the joint space $\mathcal{X} \times \mathbb{R}$ of inputs and labels, that is,

$$R(f) = \int_{\mathcal{X}\times\mathbb{R}} (f(\mathbf{x}) - y)^2 \mathrm{d}\rho(\mathbf{x}, y)\,.$$

Given the input space $\mathcal{X}$, the minimizer of the error is the so-called regression function:

$$f_\rho(\mathbf{x}) = \int_{\mathbb{R}} y \ \mathrm{d}\rho(y \mid \mathbf{x})\,.$$

Following Bauer et al. (2007); Lo Gerfo et al. (2008); Baldassarre et al. (2012), I characterize the quality of a learning algorithm via its consistency properties. In particular, by considering whether the learning algorithm is consistent in the following sense:

Definition 4.4. *A learning algorithm is consistent if the following holds with high probability*

$$\lim_{n\to\infty} \int_{\mathcal{X}} \left(\hat{f}_n^\lambda(\mathbf{x}) - f_\rho(\mathbf{x})\right)^2 d\rho(\mathbf{x}) = 0\,,$$

where $\hat{f}_n^\lambda$ denotes the hypothesis inferred by the learning algorithm from a training set having n independently and identically drawn training examples.

The following result is assembled from the existing literature concerning spectral filtering based regularization methods and I present it here only in a rather abstract form. For the exact details and further elaboration, I refer to Bauer et al. (2007); Lo Gerfo et al. (2008); Baldassarre et al. (2012).

Theorem 4.6. *If the filter function is admissible and the kernel function is universal, then the learning algorithm is consistent in the sense of Def. 4.4. Furthermore, if the regularization parameter is set as $\lambda = \frac{1}{n^{2\nu+1}}$, where n denotes the number of independently and identically drawn training examples, then the following holds with high*

probability:

$$R(\hat{f}^{\lambda}) - R(f_{\rho}(\mathbf{x})) = \mathcal{O}\left(n^{-\frac{\bar{\nu}}{2\bar{\nu}+1}}\right) . \tag{4.28}$$

Intuitively put, the universality of the kernel ensures that the regression function belongs to the hypothesis space of the learning algorithm and the admissibility of the regularizer ensures that $R(\hat{f}^{\lambda})$ converges to it when the size of the training set approaches infinity and the rate of convergence is reasonable.

Corollary 4.6.1. *Two-step KRR is consistent and the hypothesis it infers from the training set of size $n = mq$ converges to the underlying regression function with a rate at least proportional to*

$$R(\hat{f}^{\lambda}) - R(f_{\rho}(u, v)) = \mathcal{O}\left(\min(m, q)^{-\frac{\bar{\nu}}{2\bar{\nu}+1}}\right) . \tag{4.29}$$

Proof. The result follows from the admissibility of the pairwise filter function, the universality of the pairwise Kronecker product kernel and the fact that the training set consists of at least $\min(m, q)$ independently and identically drawn training examples. □

Hence, it is proven that the two-step KRR is not only a universal method (can approximate any pairwise prediction function), but will also converge to the prediction function that generated the data when provided with enough training examples.

4.5.4 Learning relations using pairwise kernels

In Section 2.2.1 it was discussed how a monadic pairwise prediction function can be used to model graded relations between two objects of the same space. A pairwise function $f : \mathcal{U}^2 \to \mathbb{R}$ can be learned using a suitable pairwise kernel, defined over two monadic dyads (u, u') and $(\bar{u}, \bar{u}')$. Analogously to Eq. (4.6), the Kronecker kernel for the monadic

case is calculated as

$$\Gamma((u, u'), (\bar{u}, \bar{u}')) = k(u, u')k(\bar{u}, \bar{u}') .$$

It has been shown that this kernel is universal if $k(\cdot, \cdot)$ is universal, it can be used to approximate arbitrary relations (Waegeman et al., 2012). Since the above pairwise kernel is a special case of the more general pairwise kernel of Eq. (4.6), this result is a special case of Theorem 4.4.

For the various properties discussed in Section 2.2.1, special pairwise kernels can be constructed. Using the respective pairwise kernel, the learned pairwise prediction function will satisfy the respective property. The most important kernels are summarized in Table 4.2. The intuition why these kernels lead to pairwise prediction functions with the desired properties can easily be gained from the prediction function in primal form implied by these kernels. For details and proofs of universality of these kernels within their respective relation class I refer to Waegeman et al. (2012). For a discussion on some algebraic shortcuts to efficiently implement some of these kernels for Kronecker KRR and conditional ranking, see Pahikkala et al. (2013).

4.5.5 Kernels without object features

Setting A is surprisingly complex, as it is possible to also make accurate predictions without any features of the objects, evidenced by the success of techniques such as matrix factorization. The structure of the label matrix $\mathbf{Y}$, e.g. being low rank, often contains enough information to successfully make predictions for Setting A. As discussed in Section 2.4.3, in recommender systems, methods that do not take side-features into account are often categorized as collaborative filtering. In this section I will briefly elaborate on how such content-based filtering can be emulated using the kernel-based framework discussed in this work.

Table 4.2: Overview of some important pairwise kernels for relational learning. Some of the kernels can only be used to learn relations with certain properties, but can be shown to be universal in their respective class if $k(\cdot,\cdot)$ is a universal kernel.

Relation type	definition	kernel
Arbitrary relations	-	$k(u,u')k(\bar{u},\bar{u}')$
Symmetric	Def. 2.3 and 2.4	$k(u,u')k(\bar{u},\bar{u}') + k(u,\bar{u}')k(\bar{u}',u')$
Anti-symmetric & reciprocal	Def. 2.5 and 2.6	$k(u,u')k(\bar{u},\bar{u}') - k(u,\bar{u}')k(\bar{u},u')$
Transitive	Def. 2.7 and 2.8	$k(u,\bar{u})$
Ranking representable (reciprocal)	Def. 2.9	$k(u,u') + k(\bar{u},\bar{u}') - k(u,\bar{u}') - k(\bar{u},u')$
Ranking representable (symmetric)	Def. 2.9	$k(u,u') + k(\bar{u},\bar{u}') + k(u,\bar{u}') + k(\bar{u},u')$
Euclidean representable	Def. 2.10	$(k(u,\bar{u}) - k(u,\bar{u}') - k(u',\bar{u}) + k(u',\bar{u}'))^2$

The idea behind a featureless description of u or v is simple: an object u resp. v can be described by the observed labels of the dyads that contain the object. In the context of item recommendation, this seems reasonable: users are described by the ratings they have given to items and, likewise, items are described by users' ratings. For example, Basilico and Hofmann (2004b) use a kernel based on the Pearson correlation of rating vectors of users to obtain a kernel description of users for collaborative filtering. In bioinformatics, van Laarhoven et al. (2011) predict drug-target interactions using so-called Gaussian interaction profile (GIP) kernels, in the notation of this work given by

$$k^{\mathrm{GIP}}(u_i, u_k) = \exp(-\gamma_u ||\mathbf{Y}_{i.} - \mathbf{Y}_{i.}||^2)$$
$$g^{\mathrm{GIP}}(u_j, u_l) = \exp(-\gamma_v ||\mathbf{Y}_{.j} - \mathbf{Y}_{j.}||^2)\,,$$

with γ_u and γ_v two hyperparameters determining the bandwidth of the kernel. The above kernel is nothing more than the classical radial basis kernel applied to the corresponding row or column of the label matrix.

There is nothing inherently wrong with using the labels to construct feature descriptions or kernels for the object. One should only be cautious when taking a holdout set for model selection or model evaluation; the omitted labels should also be removed from the feature description to prevent overfitting. This makes these kernels incompatible with the holdout tricks presented in Section 4.6.1.

Kernels that take observed labels into account, such as the GIP kernel, are in theory quite powerful. As they can be used to learn nonlinear associations, they lead to more expressive models than matrix factorization. The advantage of using these kernels compared to other collaborative filtering techniques such as matrix factorization, k-nearest neighbors or restricted Boltzmann machines, is that side-features can elegantly be incorporated into the model. To this end, one only has to combine the collaborative and content-based kernel

matrices, for example by computing a weighted sum or element-wise multiplication.

As an alternative for constructing kernels based on the observed labels, I propose to use different kernels, 'agnostic' of the true objects:

$$k^{\text{smooth}}(u_i, u_k) = 1 + \theta_u \delta(u_i, u_k)$$
$$g^{\text{smooth}}(v_j, v_l) = 1 + \theta_v \delta(v_j, u_l)\,,$$

or as Gram matrices:

$$\mathbf{K} = \mathbb{J}_{m\times m} + \theta_u \mathbb{I}_m \qquad \text{and} \qquad \mathbf{G} = \mathbb{J}_{q\times q} + \theta_v \mathbb{I}_q\,. \tag{4.30}$$

Here, θ_u and θ_v are two hyperparameters of the kernels and $\mathbb{J}_{m\times m}$ is a $m \times m$ matrix filled with ones. I will call these kernels smoothing kernels for reasons that will follow. The rationale behind these kernels is quite simple: a kernel that is the identity matrix would imply that all objects are unique and independent; there is no similarity between them. Using the all-ones matrix on the other hand encodes all the objects as exactly the same; no distinction between two objects can be made. Hence, the kernels of (4.30) represent a trade-off between all objects being similar (first part) and all objects being unique (second part). This is controlled by the hyperparameters θ_u and θ_v.

Using these kernels in the Kronecker-based models has an interesting interpretation: the predictions can be written as a weighted sum of averages.

THEOREM 4.7 (Smoothing kernels). *Predictions using Kronecker KRR or two-step KRR for Setting A using the Gram matrices (4.30) are of the form:*

$$f(u_i, v_j) = \alpha_1 \mathbf{Y}_{ij} + \alpha_2 \frac{1}{q}\sum_{l=1}^{q} \mathbf{Y}_{il} + \alpha_3 \frac{1}{m}\sum_{k=1}^{m} \mathbf{Y}_{kj} + \alpha_4 \frac{1}{mq}\sum_{k=1}^{m}\sum_{l=1}^{q} \mathbf{Y}_{kl}\,,$$

with $(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \in \mathbb{R}^4$.

Proof. First, note that the hat matrix $\mathbf{H}^{\text{smooth}}$ (Eq. (4.9)) of Kronecker (and hence two-step) KRR methods is of the form

$$\mathbf{H}^{\text{smoother}} = a_1\mathbb{I}_q \otimes \mathbb{I}_m + a_2\mathbb{J}_q \otimes \mathbb{I}_m + a_3\mathbb{I}_q \otimes \mathbb{J}_m + a_4\mathbb{J}_q \otimes \mathbb{J}_m\,. \quad (4.31)$$

This can easily be seen because the pairwise Gram matrix will be of the form of (4.31) and Properties A.7 and A.8 state that multiplying or inverting matrices of the form of (4.31) results in a matrix of the same form. From these properties, it follows that the hat matrix will also be of this form.

The prediction for dyad (u_i, v_j) is given by $[\mathbf{H}^{\text{smooth}}\text{vec}(\mathbf{Y})]_{jq+i}$. Using the relation between the Kronecker product and the vectorization operation, each term of (4.31) can be rewritten using Property A.5 as follows

$$\begin{aligned}
a_1[(\mathbb{I}_q \otimes \mathbb{I}_m)\text{vec}(\mathbf{Y})]_{jm+i} &= a_1[\mathbb{I}_m\mathbf{Y}\mathbb{I}_q]_{ij} = a_1\mathbf{Y}_{ij} \\
a_2[(\mathbb{J}_{q\times q} \otimes \mathbb{I}_m)\text{vec}(\mathbf{Y})]_{jm+i} &= a_2[\mathbb{I}_m\mathbf{Y}\mathbb{J}_{q\times q}]_{ij} = a_2\sum_{l=1}^{q}\mathbf{Y}_{il} \\
a_3[(\mathbb{I}_q \otimes \mathbb{J}_{m\times m})\text{vec}(\mathbf{Y})]_{jm+i} &= a_3[\mathbb{J}_{m\times m}\mathbf{Y}\mathbb{I}_q]_{ij} = a_3\sum_{k=1}^{m}\mathbf{Y}_{kj} \\
a_4[(\mathbb{J}_{q\times q} \otimes \mathbb{J}_{m\times m})\text{vec}(\mathbf{Y})]_{jm+i} &= a_4[\mathbb{J}_{m\times m}\mathbf{Y}\mathbb{J}_{q\times q}]_{ij} = a_4\sum_{l=1}^{q}\sum_{k=1}^{m}\mathbf{Y}_{kl}\,,
\end{aligned}$$

which proves the theorem. □

The smoother kernel is thus quite restrictive in the type of models that can be learned. It can only exploit the fact that some rows or columns have a larger average value (e.g. in item recommendation, some items in collaborative filtering have a high average rating, independent for the user). Nevertheless, it can lead to a good baseline and in Chapter 7 I will show that is can provide a good performance for detecting missing interactions in species interaction matrices.

4.6 Efficient algorithms for Kronecker kernel-based pairwise learning

In this section I derive some computational shortcuts for Kronecker and two-step KRR. For Kronecker KRR, it is well known that the huge system of Eq. (4.5) can be solved efficiently because the Gram matrix can be decomposed. The two-step method takes this decomposition one step further, as the model can be seen as applying two consecutive regression steps. This allows to derive efficient algorithms for cross-validation for each of the four settings depicted in Figure 4.1, while the original Kronecker KRR only allowed for a shortcut for Setting A. The same linear algebra can also be used to implement a scheme for online updating of the model with new instances or tasks. These shortcuts for cross-validating for Settings B, C and D and online updating cannot be derived for Kronecker ridge regression in general. Furthermore, in Section 4.6.3 a simple iterative transductive algorithm is proposed to impute missing values in a label matrix. This algorithm is highly relevant in the context of pairwise learning, because it allows us to extend computational shortcuts to cases where the dataset is incomplete.

4.6.1 Efficient hold-out computation

As indicated in Figure 4.1 and Section 2.3.2, making predictions in a dyadic setting is much more complex compared to the classical case when there is only one task. This implies that the correct way to assess the performance and to do model selection is also more complicated. To this end I suggest cross-validation settings that take the structure of the label matrix into account. Depending on the prediction setting of interest, one should withhold individual elements, rows, columns or both of the label matrix, as shown in Figure 4.3. These schemes have been discussed in other works, often in the context of predict-

ing interactions in molecular biology, e.g. Park and Marcotte (2012); Pahikkala et al. (2015). Section 2.3 discusses this in detail. Here, I will only focus on deriving shortcuts for leave-one-out cross-validation for these settings. Extensions to general hold-out schemes, such as K-fold cross validation, can be obtained using similar reasonings as the one in this section.

It is well known that for independent-task KRR one can efficiently compute the values for leave-one-out cross-validation, provided one has stored the hat matrix (Rifkin and Lippert, 2007a). For the instances, using the kernel $k(\cdot, \cdot)$, the hat matrix is denoted as $\mathbf{H}^k = \mathbf{K}(\mathbf{K} + \lambda_u \mathbb{I})^{-1}$ (see Eq. (4.9)). As noted earlier, if this matrix is obtained using an eigenvalue decomposition of $\mathbf{K}$, $\mathbf{H}^k$ can be computed efficiently for any λ_u:

$$\begin{aligned}
\mathbf{H}^k &= \mathbf{K}(\mathbf{K} + \lambda_u \mathbb{I})^{-1} \\
&= \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top(\mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top + \lambda_u \mathbb{I})^{-1} \\
&= \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top\mathbf{U}(\mathbf{\Sigma} + \lambda_u \mathbb{I})^{-1}\mathbf{U}^\top \\
&= \mathbf{U}(\mathbf{\Sigma}(\mathbf{\Sigma} + \lambda_u \mathbb{I})^{-1})\mathbf{U}^\top .
\end{aligned}$$

First, I will provide the well-known shortcut for independent-task KRR for calculating the leave-one-out cross-validation values, using our notation. I start with the classical leave-one-out shortcut.

THEOREM 4.8. *A single row of the matrix $\mathbf{F}^{\mathrm{IT,LOO}}$ containing the labels re-estimated by leave-one-out cross-validation using independent-task KRR with associated hat matrix $\mathbf{H}^k$, can be calculated as*

$$\mathbf{F}^{\mathrm{IT,LOO}}_{i.} = \frac{\mathbf{H}^k_{i.}\mathbf{Y} - \mathbf{H}^k_{ii}\mathbf{Y}_{i.}}{1 - \mathbf{H}^k_{ii}} .$$

Proof. This is merely a multivariate version of the leave-one-out shortcut, proven in texts such as Wahba (1990); Pahikkala et al. (2006); Rifkin and Lippert (2007a). □

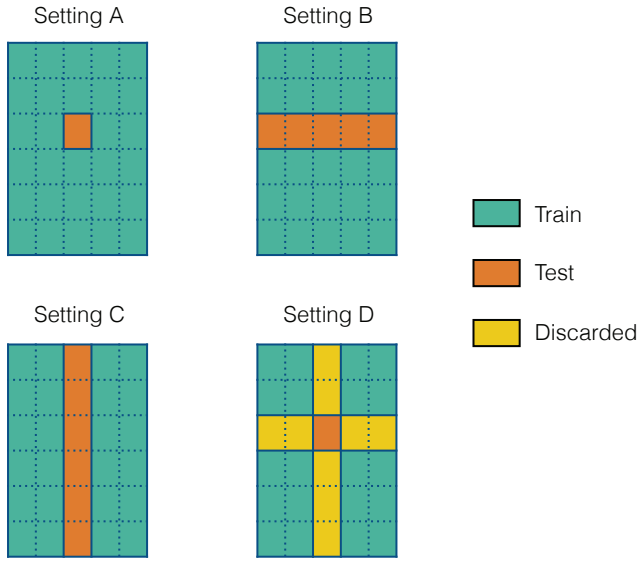


Figure 4.3: Overview of different versions of leave-one-out cross-validation settings discussed in Section 4.3 applied to a problem with six instances and four tasks.

The above theorem can be applied to ridge regression and related models. The shortcut is relevant for both Settings A and B, as it is used to estimate how well the model can generalize to new instances, though KRR does not use any information on the tasks. Starting from this general shortcut and using the connection between independent-task, two-step and Kronecker KRR, we can derive shortcuts for Setting A.

Corollary 4.8.1 (Setting A). *A single element of the matrix containing the labels re-estimated by leave-one-out cross-validation for Setting A, i.e. leaving out one dyad at a time, using Kronecker KRR, can be calculated as*

$$\mathbf{F}_{ij}^{\mathrm{KK,LOO,A}} = \frac{\mathbf{H}_{s.}^{\mathrm{KK}}\mathrm{vec}(\mathbf{Y}) - \mathbf{H}_{ss}^{\mathrm{KK}}\mathbf{Y}_{ij}}{1 - \mathbf{H}_{ss}^{\mathrm{KK}}},$$

with $\mathbf{H}^{\mathrm{KK}} = (\mathbf{G} \otimes \mathbf{K})(\mathbf{G} \otimes \mathbf{K} + \lambda \mathbb{I})^{-1}$ *and* $s = mj + i$.

Proof. It was noted earlier that Kronecker KRR is merely KRR using the Gram matrix $\mathbf{G} \otimes \mathbf{K}$ and $\mathrm{vec}(\mathbf{Y})$ as the single output label vector. Since $\mathbf{H}^{\mathrm{KK}}$ is the corresponding hat matrix, the proof follows directly from rephrasing Theorem 4.8 in this terminology. □

By using the eigenvalue decomposition of the kernel matrices, the diagonal elements of $\mathbf{H}^{\mathrm{KK}}$ and $\mathbf{H}^{\mathrm{KK}}\mathrm{vec}(\mathbf{Y})$ can be computed efficiently.

Corollary 4.8.2 (Setting A). *A single element of the matrix containing the labels re-estimated by leave-one-out cross-validation for Setting A, i.e. leaving out one dyad at a time, using two-step KRR can be calculated as*

$$\mathbf{F}_{ij}^{\mathrm{TS,LOO,A}} = \frac{\mathbf{H}_{s.}^{\mathrm{TS}}\mathrm{vec}(\mathbf{Y}) - \mathbf{H}_{ss}^{\mathrm{TS}}\mathbf{Y}_{ij}}{1 - \mathbf{H}_{ss}^{\mathrm{TS}}},$$

with $\mathbf{H}^{\mathrm{TS}} = (\mathbf{G} + \lambda_v \mathbb{I})^{-1}\mathbf{G} \otimes (\mathbf{K} + \lambda_u \mathbb{I})^{-1}\mathbf{K}$ *and* $s = mj + i$.

Proof. Theorem 4.5.2 in Section 4.5.1 states that two-step KRR is merely KRR with the Gram matrix $\boldsymbol{\Xi}$ and $\mathrm{vec}(\mathbf{Y})$ as the single output label vector. From Eq. (4.15) it follows that $\mathbf{H}^{\mathrm{TS}}$ is the hat matrix for two-step KRR in vectorized form:

$$\begin{aligned}
\mathbf{f} &= \mathrm{vec}(\mathbf{K}\mathbf{A}^{\mathrm{TS}}\mathbf{G}) \\
&= \mathrm{vec}(\mathbf{K}\left(\mathbf{K} + \lambda_u \mathbb{I}\right)^{-1}\mathbf{Y}\left(\mathbf{G} + \lambda_v \mathbb{I}\right)^{-1}\mathbf{G}) \\
&= [(\mathbf{G} + \lambda_v \mathbb{I})^{-1}\mathbf{G} \otimes (\mathbf{K} + \lambda_u \mathbb{I})^{-1}\mathbf{K}]\mathrm{vec}(\mathbf{Y}).
\end{aligned}$$

Hence, the proof follows directly from rephrasing Theorem 4.8 in this terminology. □

These two shortcuts are of interest when validating or tuning models for collaborative filtering or network inference. For example, recently a model for recipe completion was validated by iteratively with-

holding single elements from the label matrix (De Clercq et al., 2015). An interesting application would be in supervised biological network inference, e.g. predicting interactions between biomolecules, such as proteins, ligands or DNA or predicting interactions between species, such as plants and pollinators (see e.g. Rafferty and Ives (2013); Hadfield et al. (2014) for applications of pairwise learning in such a setting). Such datasets are often plagued with false negatives or false positives, which make them difficult to analyze, see Schrynemackers et al. (2013) for a discussion. Using the provided holdout tricks, it is possible to re-estimate each interaction extremely efficiently in order to screen for mislabeled observations.

Because two-step KRR can be decomposed into two 'steps', it is possible to derive shortcuts for leave-one-out cross-validation for Settings B, C and D. Below is a shortcut for Setting B, which allows for cross-validation of one row (instance) at a time.

Corollary 4.8.3 (Setting B). *A single row of the matrix containing the labels re-estimated by leave-one-out cross-validation for Setting B, i.e. leaving out one instance at a time, using two-step KRR, can be calculated as*

$$\mathbf{F}_{i.}^{\mathrm{TS,LOO,B}} = \frac{\left(\mathbf{H}_{i.}^{k}\mathbf{Y} - \mathbf{H}_{ii}^{k}\mathbf{Y}_{i.}\right)\mathbf{H}^{g}}{1 - \mathbf{H}_{ii}^{k}}.$$

Proof. Using two-step KRR for Setting B can be thought of as just using independent-task KRR with $\mathbf{H}^k$ as the hat matrix and $\mathbf{Y}\mathbf{H}^g$ as the label matrix. The proof follows directly from rephrasing Theorem 4.8 in this terminology. □

Similarly, using the following shortcut one can perform cross-validation for Setting C, leaving out one column (task) at a time.

Corollary 4.8.4 (Setting C). *A single column of the matrix containing the labels re-estimated by leave-one-out cross-validation for*

Setting C, i.e. leaving out one task at a time, using two-step KRR, can be calculated as

$$\mathbf{F}_{\cdot j}^{\mathrm{TS,LOO,C}} = \frac{\mathbf{H}^k \left(\mathbf{Y}\mathbf{H}_{\cdot j}^g - \mathbf{Y}_{\cdot j}\mathbf{H}_{jj}^g \right)}{1 - \mathbf{H}_{jj}^g} .$$

Proof. Similarly as for Corollary 4.8.3 but using a 'transposed' dataset, two-step KRR for Setting C can be thought of as just performing independent-task KRR with $\mathbf{H}^g$ as the hat matrix and $\mathbf{Y}^\top \mathbf{H}^k$ as the label matrix. We apply Theorem 4.8 using this terminology and transpose the obtained matrix. □

Finally, the theorem below gives the shortcut for Setting D, the zero-shot learning setting.

Corollary 4.8.5 (Setting D). *A single element of the matrix containing the labels re-estimated by leave-one-out cross-validation for Setting D, i.e. leaving out one dyad with the corresponding row and column and predicting for that dyad, using two-step KRR can be calculated as*

$$\mathbf{F}_{ij}^{\mathrm{TS,LOO,D}} = \frac{\mathbf{H}_{i\cdot}^k \left(\mathbf{Y}\mathbf{H}_{\cdot j}^g - \mathbf{Y}_{\cdot j}\mathbf{H}_{jj}^g \right) - \mathbf{H}_{ii}^k \left(\mathbf{Y}_{i\cdot}\mathbf{H}_{\cdot j}^g - \mathbf{Y}_{ij}\mathbf{H}_{jj}^g \right)}{(1 - \mathbf{H}_{ii}^k)(1 - \mathbf{H}_{jj}^g)} .$$

Proof. This can easily be proven by applying Theorem 4.8 twice. First, a column vector with the predictions for task j is generated, using a model trained only based on the $q-1$ remaining tasks. Adopting a similar reasoning as in Corollary 4.8.4, we obtain

$$\mathbf{F}_{\cdot j}^{\mathrm{STEP1}} = \frac{\mathbf{Y}\mathbf{H}_{\cdot j}^g - \mathbf{Y}_{\cdot j}\mathbf{H}_{jj}^g}{1 - \mathbf{H}_{jj}^g} .$$

This vector is subsequently used as a label vector for an 'unseen' task j. We apply Theorem 4.8 once more to obtain the leave-one-out

Table 4.3: Overview of algebraic shortcuts for performing leave-one-out prediction for the different settings using two-step KRR. $\mathrm{mat}_{m\times q}$ is an operator that reshapes a vector in an $m \times q$ matrix, diag_v takes the diagonal elements of a square matrix to a vector and diag_m takes the corresponding diagonal matrix of a given square matrix and $\mathbf{1}_n$ is a vector of ones with a length of n.

model	setting	leave-one-out values
IT	A & B	$\frac{\mathbf{H}^k\mathbf{Y} - \mathrm{diag}_m(\mathbf{H}^k)\mathbf{Y}}{\mathrm{diag}_v(\mathbb{I} - \mathbf{H}^k)\mathbf{1}_q^\top}$
KK	A	$\mathrm{mat}_{m\times q}\left(\frac{\mathbf{H}^{\mathrm{KK}}\mathrm{vec}(\mathbf{Y}) - \mathrm{diag}_m(\mathbf{H}^{\mathrm{KK}})\mathrm{vec}(\mathbf{Y})}{\mathrm{diag}_v(\mathbb{I} - \mathbf{H}^{\mathrm{KK}})}\right)$
TS	A	$\mathrm{mat}_{m\times q}\left(\frac{\mathbf{H}^{\mathrm{TS}}\mathrm{vec}(\mathbf{Y}) - \mathrm{diag}_m(\mathbf{H}^{\mathrm{TS}})\mathrm{vec}(\mathbf{Y})}{\mathrm{diag}_v(\mathbb{I} - \mathbf{H}^{\mathrm{TS}})}\right)$
TS	B	$\frac{\mathbf{H}^k\mathbf{Y}\mathbf{H}^g - \mathrm{diag}_m(\mathbf{H}^k)\mathbf{Y}\mathbf{H}^g}{\mathrm{diag}_v(\mathbb{I} - \mathbf{H}^k)\mathbf{1}_q^\top}$
TS	C	$\frac{\mathbf{H}^k\mathbf{Y}\mathbf{H}^g - \mathbf{H}^k\mathbf{Y}\mathrm{diag}_m(\mathbf{H}^g)}{\mathbf{1}_m^\top\mathrm{diag}_v(\mathbb{I} - \mathbf{H}^g)^\top}$
TS	D	$\frac{(\mathbf{H}^k - \mathrm{diag}_m(\mathbf{H}^k))\mathbf{Y}(\mathbf{H}^g - \mathrm{diag}_m(\mathbf{H}^g))}{\mathrm{diag}_v(\mathbb{I} - \mathbf{H}^k)\mathrm{diag}_v(\mathbb{I} - \mathbf{H}^g)^\top}$

value of instance (row) i:

$$\begin{aligned}
\mathbf{F}_{ij}^{\mathrm{TS,LOO,D}} &= \frac{\mathbf{H}^k_{i.}\mathbf{F}^{\mathrm{STEP1}}_{.j} - \mathbf{H}^k_{ii}\mathbf{F}^{\mathrm{STEP1}}_{ij}}{1 - \mathbf{H}^k_{ii}} \\
&= \frac{\mathbf{H}^k_{i.}\left(\frac{\mathbf{Y}\mathbf{H}^g_{.j} - \mathbf{Y}_{.j}\mathbf{H}^g_{jj}}{1-\mathbf{H}^g_{jj}}\right) - \mathbf{H}^k_{ii}\left(\frac{\mathbf{Y}_{i.}\mathbf{H}^g_{.j} - \mathbf{Y}_{ij}\mathbf{H}^g_{jj}}{1-\mathbf{H}^g_{jj}}\right)}{1 - \mathbf{H}^k_{ii}} \\
&= \frac{\mathbf{H}^k_{i.}\left(\mathbf{Y}\mathbf{H}^g_{.j} - \mathbf{Y}_{.j}\mathbf{H}^g_{jj}\right) - \mathbf{H}^k_{ii}\left(\mathbf{Y}_{i.}\mathbf{H}^g_{.j} - \mathbf{Y}_{ij}\mathbf{H}^g_{jj}\right)}{(1 - \mathbf{H}^k_{ii})(1 - \mathbf{H}^g_{jj})}.
\end{aligned}$$

□

All the derived shortcuts are summarized in a more compact matrix notation in Table 4.3. Each matrix of hold-out values is computed

by dividing the predicted labels of the complete training set by an appropriate linear transformation of the diagonal elements of the hat matrices. As noted earlier, using an eigenvalue decomposition with a time complexity of $\mathcal{O}(m^3 + q^3)$, these hat matrices and predictions can be obtained for any values of λ_u and λ_v by straightforward matrix manipulations. The shortcuts are in fact valid for all models with a pairwise spectral filter of the form in Eq. (4.27). Stated more boldly, after the eigenvalue decomposition, one can tune or validate the model for any of the four settings at essentially no computational cost, compared to the cost of the initial preprocessing. For Kronecker KRR, the only computational shortcut of Table 4.3 possible is for Setting A, as this makes no use of this property. Since many interesting learning problems relate to the other settings, I consider the shortcuts possible for two-step KRR a very strong merit of this method.

4.6.2 Learning with mini batches

In some cases it is desirable to train a model in an online fashion, as opposed to the standard batch training. For example, when new instances for the different tasks become available, it is more desirable to update the model parameters rather than completely retrain the model. Similarly, new tasks could be added to the model, which also influences the performance of the older tasks. Figure 4.4 illustrates how the initial label matrix can be updated using the proposed algorithms.

Such settings are prevalent in online applications, such as recommender systems, in which data is added dynamically so that the systems should be updated swiftly. In the context of dealing with large-scale data, the required matrices might simply be too large to fit in main memory as a whole, thus an iterative approach is needed.

In addition to gradient-based methods such as stochastic gradient descent and conjugated gradient descent, closed-form solutions can be derived for updating the model parameters for two-step KRR. Since

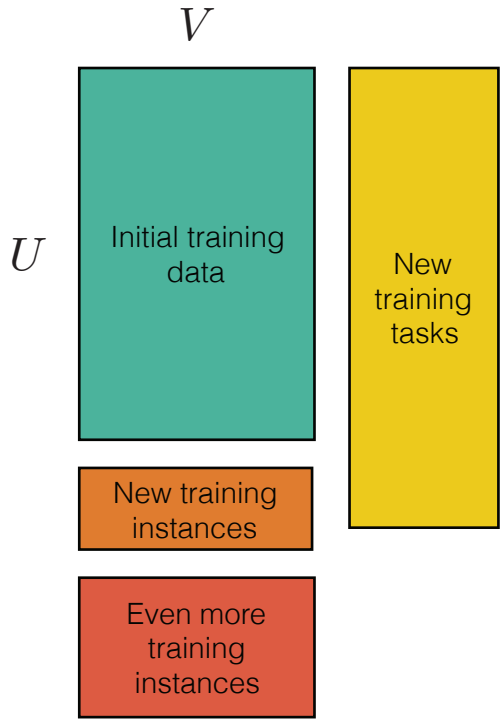


Figure 4.4: Illustration of new instances or tasks that can be added to the initial label matrix. The two-step KRR can be updated without needing to be retrained.

I assume that this is of particular interest for large-scale data applications, I will derive a shortcut for the primal form. Denote a feature vector for the instances by $\boldsymbol{\phi} \in \mathbb{R}^d$ and for the tasks by $\boldsymbol{\psi} \in \mathbb{R}^r$. Here I assume that $d \ll m$ and $r \ll q$. These can either be the primal features or be obtained from a decomposition of the kernel matrices, for example by means of the Nyström method (Drineas and Mahoney, 2005). This leads to the associated feature matrices $\boldsymbol{\Phi} \in \mathbb{R}^{m \times d}$ and $\boldsymbol{\Psi} \in \mathbb{R}^{q \times r}$. Hence, the primal notation boils down to:

$$\begin{aligned} \mathbf{K} &= \boldsymbol{\Phi}\boldsymbol{\Phi}^\top & \mathbf{k} &= \boldsymbol{\Phi}\boldsymbol{\phi} \\ \mathbf{G} &= \boldsymbol{\Psi}\boldsymbol{\Psi}^\top & \mathbf{g} &= \boldsymbol{\Psi}\boldsymbol{\psi}\,. \end{aligned}$$

Thus the model of Eq. (4.13) translates to

$$\begin{aligned} f^{\mathrm{TS}}(\mathbf{x}) &= \boldsymbol{\phi}^\top \boldsymbol{\Phi} \mathbf{A}^{\mathrm{TS}} \boldsymbol{\Psi}^\top \boldsymbol{\psi} \\ &= \boldsymbol{\phi}^\top \mathbf{W}^{\mathrm{TS}} \boldsymbol{\psi} \,, \end{aligned}$$

with $\mathbf{W}^{\mathrm{TS}}$ the primal parameter matrix. Suppose again that the dataset is complete, then, starting from Eq. (4.15), the parameters are given by

$$\begin{aligned} \mathbf{W}^{\mathrm{TS}} &= \boldsymbol{\Phi}(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \lambda_u \mathbb{I})^{-1} \mathbf{Y} (\boldsymbol{\Psi}\boldsymbol{\Psi}^\top + \lambda_v \mathbb{I})^{-1} \boldsymbol{\Psi}^\top \\ &= (\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda_u \mathbb{I})^{-1} \boldsymbol{\Phi}^\top \mathbf{Y} \boldsymbol{\Psi} (\boldsymbol{\Psi}^\top\boldsymbol{\Psi} + \lambda_v \mathbb{I})^{-1} \,, \end{aligned}$$

where I made use of the matrix inversion lemma (Property A.3). Suppose that in a first training phase, we use only the first $m-l$ instances to train the first model and later update the model with the remaining l instances. Without loss of generality, we divide the labels and instance features as follows in corresponding block matrices

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\Phi}_1 \\ \boldsymbol{\Phi}_2 \end{pmatrix} \quad \text{and} \quad \mathbf{Y} = \begin{pmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \end{pmatrix} \,,$$

with $\boldsymbol{\Phi}_1 \in \mathbb{R}^{(m-l)\times d}$, $\boldsymbol{\Phi}_2 \in \mathbb{R}^{l\times d}$, $\mathbf{Y}_1 \in \mathbb{R}^{(m-l)\times q}$ and $\mathbf{Y}_2 \in \mathbb{R}^{l\times q}$. The model parameters of the complete dataset can then be calculated as

$$\begin{aligned} \mathbf{W}^{\mathrm{TS}} &= (\boldsymbol{\Phi}_1^\top\boldsymbol{\Phi}_1 + \boldsymbol{\Phi}_2^\top\boldsymbol{\Phi}_2 + \lambda_u \mathbb{I})^{-1} (\boldsymbol{\Phi}_1^\top\mathbf{Y}_1 + \boldsymbol{\Phi}_2^\top\mathbf{Y}_2) \boldsymbol{\Psi} (\boldsymbol{\Psi}^\top\boldsymbol{\Psi} + \lambda_v \mathbb{I})^{-1} \\ &= (\mathbf{M}_1^{-1} + \boldsymbol{\Phi}_2^\top\boldsymbol{\Phi}_2)^{-1} (\boldsymbol{\Phi}_1^\top\mathbf{Y}_1 + \boldsymbol{\Phi}_2^\top\mathbf{Y}_2) \boldsymbol{\Psi} (\boldsymbol{\Psi}^\top\boldsymbol{\Psi} + \lambda_v \mathbb{I})^{-1} \\ &= [\mathbf{M}_1 - \mathbf{M}_1\boldsymbol{\Phi}_2^\top (\mathbb{I} + \boldsymbol{\Phi}_2\mathbf{M}_1\boldsymbol{\Phi}_2^\top)^{-1} \boldsymbol{\Phi}_2\mathbf{M}_1] \\ &\quad (\boldsymbol{\Phi}_1^\top\mathbf{Y}_1 + \boldsymbol{\Phi}_2^\top\mathbf{Y}_2) \boldsymbol{\Psi} (\boldsymbol{\Psi}^\top\boldsymbol{\Psi} + \lambda_v \mathbb{I})^{-1} \,, \end{aligned} \tag{4.32}$$

where $\mathbf{M}_1 = (\boldsymbol{\Phi}_1^\top\boldsymbol{\Phi}_1 + \lambda_u \mathbb{I})^{-1}$. In Eq. (4.32), I have made use of the Woodbury identity (Property A.1). Thus, in order to update the model parameters, the $l \times l$ matrix $(\mathbb{I} + \boldsymbol{\Phi}_2^\top\mathbf{M}_1\boldsymbol{\Phi}_2)$ has to be inverted. A practical implementation for this scheme is given in Algorithm 4.1.

INPUT: $\mathbf{\Phi}_{\text{new}}, \mathbf{Y}_{\text{new}}, \mathbf{W}_{\text{old}}, \mathbf{M}_{\text{old}}, \mathbf{B}^g, \mathbf{\Phi}_{\text{old}}^\top \mathbf{Y}_{\text{old}}$

1: $\mathbf{M}_{\text{new}} = \mathbf{M}_{\text{old}} - \mathbf{M}_{\text{old}}\mathbf{\Phi}_{\text{new}}^\top(\mathbf{\Phi}_{\text{new}}\mathbf{M}_{\text{old}}\mathbf{\Phi}_{\text{new}}^\top + \mathbb{I})^{-1}\mathbf{\Phi}_{\text{new}}\mathbf{M}_{\text{old}}$ /* $\mathcal{O}(l^3 + d^2 l)$ */

2: $\mathbf{W}_{\text{new}}^{\text{TS}} = \mathbf{M}_{\text{new}}(\mathbf{\Phi}_{\text{old}}^\top \mathbf{Y}_{\text{old}} + \mathbf{\Phi}_{\text{new}}^\top \mathbf{Y}_{\text{new}})\mathbf{B}^g$ /* $\mathcal{O}(d^3 + q^3)$ */

3: **return** $\mathbf{W}_{\text{new}}, \mathbf{M}_{\text{new}}, (\mathbf{\Phi}_{\text{old}}^\top \mathbf{Y}_{\text{old}} + \mathbf{\Phi}_{\text{new}}^\top \mathbf{Y}_{\text{new}})$

Alg. 4.1: Update primal parameters $\mathbf{W}_{\text{old}}^{\text{TS}}$ of two-step KRR using batches of l new instances (objects of type u), assuming $l < d$. In addition to the old weight matrix, new instance features and labels, the algorithm requires precomputed matrices $\mathbf{M}_{\text{old}} = (\mathbf{\Phi}_{\text{old}}^\top \mathbf{\Phi}_{\text{old}} + \lambda_u \mathbb{I})^{-1}$, $\mathbf{B}^g = \mathbf{\Psi}(\mathbf{\Psi}^\top \mathbf{\Psi} + \lambda_v \mathbb{I})^{-1}$ and $\mathbf{\Phi}_{\text{old}}^\top \mathbf{Y}_{\text{old}}$. The algorithm also updates these matrices using the new data.

If $l > d$, it is useful to make use of the matrix identity

$$(\mathbb{I} + \mathbf{\Phi}_2 \mathbf{M}_1 \mathbf{\Phi}_2^\top)^{-1} \mathbf{\Phi}_2^\top \mathbf{M}_1 = \mathbf{\Phi}_2^\top (\mathbf{\Phi}_2^\top \mathbf{\Phi}_2 + \mathbf{M}_1^{-1})^{-1}$$

in line 1 so that only a $d \times d$ matrix has to be inverted. To update for new tasks, a very similar algorithm can be derived. In this case I assume that the data is divided as follows:

$$\mathbf{\Psi} = \begin{pmatrix} \mathbf{\Psi}_1 \\ \mathbf{\Psi}_2 \end{pmatrix} \quad \text{and} \quad \mathbf{Y} = \begin{pmatrix} \mathbf{Y}_1 & \mathbf{Y}_2 \end{pmatrix},$$

with $\mathbf{\Phi}_2 \in \mathbb{R}^{l \times r}$, $\mathbf{\Psi}_1 \in \mathbb{R}^{(q-l) \times r}$, $\mathbf{Y}_1 \in \mathbb{R}^{m \times (q-l)}$ and $\mathbf{Y}_2 \in \mathbb{R}^{m \times l}$. Using this notation, Algorithm 4.2 updates the parameters for a set of l new tasks.

4.6.3 An iterative transductive algorithm for incomplete data

As discussed earlier, the computational shortcuts for efficiently training and tuning Kronecker kernel-based models, including for two-step KRR, can only be applied when the dataset is complete. If the dataset is not complete, System (4.5) has to be solved directly, using for example conjugated gradient descent. In this section I propose a very

INPUT: $\mathbf{\Psi}_{\text{new}}, \mathbf{Y}_{\text{new}}, \mathbf{W}_{\text{old}}, \mathbf{N}_{\text{old}}, \mathbf{B}^k, \mathbf{Y}_{\text{old}}\mathbf{\Psi}_{\text{old}}$

1: $\mathbf{N}_{\text{new}} = \mathbf{N}_{\text{old}} - \mathbf{N}_{\text{old}}\mathbf{\Psi}_{\text{new}}^\top(\mathbf{\Psi}_{\text{new}}\mathbf{N}_{\text{old}}\mathbf{\Psi}_{\text{new}}^\top + \mathbb{I})^{-1}\mathbf{\Psi}_{\text{new}}\mathbf{N}_{\text{old}}$ /* $\mathcal{O}(l^3 + r^2 l)$ */

2: $\mathbf{W}_{\text{new}}^{\text{TS}} = \mathbf{B}^k(\mathbf{Y}_{\text{old}}\mathbf{\Psi}_{\text{old}} + \mathbf{Y}_{\text{new}}\mathbf{\Psi}_{\text{new}})\mathbf{N}_{\text{new}}$ /* $\mathcal{O}(d^3 + r^3)$ */

3: **return** $\mathbf{W}_{\text{new}}, \mathbf{N}_{\text{new}}, (\mathbf{Y}_{\text{old}}\mathbf{\Psi}_{\text{old}} + \mathbf{Y}_{\text{new}}\mathbf{\Psi}_{\text{new}})$

Alg. 4.2: Update primal parameters $\mathbf{W}_{\text{old}}^{\text{TS}}$ of two-step KRR using batches of l new tasks (objects of type v), assuming $l < q$. In addition to the old weight matrix, new task features and labels, the algorithm requires pre-computed matrices $\mathbf{N}_{\text{old}} = (\mathbf{\Psi}_{\text{old}}^\top\mathbf{\Psi}_{\text{old}} + \lambda_v\mathbb{I})^{-1}$, $\mathbf{B}^k = (\mathbf{\Phi}^\top\mathbf{\Phi} + \lambda_u\mathbb{I})^{-1}\mathbf{\Phi}^\top$ and $\mathbf{Y}_{\text{old}}\mathbf{\Psi}_{\text{old}}$. The algorithm also updates these matrices using the new data.

simple iterative algorithm to deal with an incomplete dataset using models for which

$$\text{vec}(\mathbf{F}) = \mathbf{H}\text{vec}(\mathbf{Y})$$
$$\mathbf{f} = \mathbf{H}\mathbf{y}$$

can be computed efficiently. This is the case for all models discussed in Section 4.4, as well as any model implied by a pairwise filter.

For now, I will omit reference to the pairwise aspect of the data and focus on a general transductive setting. First, I will state some important properties of the hat matrix $\mathbf{H}$ for the KRR-based models discussed in this chapter.

THEOREM 4.9. *For KRR using a positive definite Gram matrix* $\mathbf{\Gamma}$ *and a regularization parameter* $\lambda \geq 0$*, the associated hat matrix* $\mathbf{H}$ *is symmetric and all its eigenvalues* $\sigma_i(\mathbf{H})$ *belong to* $]0, 1[$.

Proof. Recall that for KRR it holds that

$$\mathbf{H} = \mathbf{\Gamma}(\mathbf{\Gamma} + \lambda\mathbb{I})^{-1}$$

and using the eigenvalue decomposition $\mathbf{\Gamma} = \mathbf{W}\mathbf{\Lambda}\mathbf{W}^\top$, this boils down to

$$\mathbf{H} = \mathbf{W}\mathbf{\Lambda}(\mathbf{\Lambda} + \lambda\mathbb{I})^{-1}\mathbf{W}^\top .$$

Clearly, this matrix is symmetric and the eigenvalues are given by

$$\sigma_i(\mathbf{H}) = \frac{\sigma_i(\Lambda)}{\sigma_i(\Lambda) + \lambda} ,$$

with $\sigma_i(\cdot)$ indicating the i-th eigenvalue of a matrix. Given that eigenvalues are strictly positive when a matrix is positive definite, it should be clear that the theorem holds. □

As discussed at length in Section 4.5, since Kronecker KRR and two-step KRR can be seen as special cases of KRR, the above theorem also holds for these models.

Suppose the label vector $\mathbf{y}$ is incomplete, some of the instances (or dyads) are not labeled. These missing values will be imputed by choosing them such that the corresponding predictions using the model remain unchanged. Without loss of generality, this can be written as:

$$\begin{bmatrix} \mathbf{f}_{\text{reest}} \\ \mathbf{f}_{\text{imp}} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{12}^\top & \mathbf{H}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{y}_{\text{obs}} \\ \mathbf{f}_{\text{imp}} \end{bmatrix} ,$$

with $\mathbf{y}_{\text{obs}}$ the observed labels, $\mathbf{f}_{\text{reest}}$ the associated re-estimation of the observed labels and $\mathbf{f}_{\text{imp}}$ the imputed values of the missing labels. This rationale can be motivated from the transductive regularized empirical risk minimization problem of Eq. (4.17). For missing labels, the predicted value and the imputation should be the same. Hence, these corresponding terms drop in the quadratic loss function. The imputed values can be obtained by solving the following linear system

$$\mathbf{f}_{\text{imp}} = \mathbf{H}_{12}^\top\mathbf{y}_{\text{obs}} + \mathbf{H}_{22}\mathbf{f}_{\text{imp}} \Leftrightarrow \mathbf{H}_{12}^\top\mathbf{y}_{\text{obs}} = (\mathbb{I} - \mathbf{H}_{22})\mathbf{f}_{\text{imp}} . \qquad (4.33)$$

THEOREM 4.10. *For KRR using a semi-positive definite Gram matrix* $\mathbf{\Gamma}$ *and* $\lambda > 0$*, System (4.33) has a unique solution.*

Proof. Theorem 4.9 states that $\sigma_i(\mathbf{H}) \in]0, 1[$, for any i. Firstly, the matrix $\mathbf{H}_{22}$ is a principal submatrix of a positive definite matrix, hence its eigenvalues are all strictly positive. Secondly, the Poincaré Separation Theorem implies that the i-th eigenvalue of principal submatrix $\mathbf{H}_{22}$ is smaller or equal than the i-th eigenvalue of $\mathbf{H}$. Together, these statements imply that $\sigma_i(\mathbf{H}_{22}) \in]0, 1[$, for any i. This in turn implies that the eigenvalues of $(\mathbb{I} - \mathbf{H}_{22})$belong to the $]0, 1[$ interval. Since all eigenvalues of this matrix are nonzero, System (4.33) has a unique solution. □

Rather than solving (4.33) explicitly, I propose an iterative algorithm where $\mathbf{f}_{\text{imp}}$ is initialized with some sensible values (e.g. using zeros or the average of the observed values) and these values are updated until convergence. Pseudocode of this simple algorithm is given in Algorithm 4.3. It can easily be shown that this algorithm converges to the correct solution.

THEOREM 4.11. *Algorithm 4.3 converges to the unique solution of (4.33) for* $\mathbf{f}_{imp}$*. The mean squared error between the approximation and* $\mathbf{f}_{imp}$ *decays exponentially with the number of iterations.*

Proof. Let $\mathbf{f}_{\text{imp}}^{(t)}$ denote the approximation of $\mathbf{f}_{\text{imp}}$ at iteration t. The error of the approximation at time t is given by

$$\mathbf{e}^{(t)} = \mathbf{f}_{\text{imp}} - \mathbf{f}_{\text{imp}}^{(t)} .$$

Furthermore, it follows that

$$\begin{aligned} \mathbf{f}_{\text{imp}}^{(t+1)} &= \mathbf{H}_{12}^{\top}\mathbf{y}_{\text{obs}} + \mathbf{H}_{22}\mathbf{f}_{\text{imp}}^{(t)} \\ \mathbf{f}_{\text{imp}} + \mathbf{e}^{(t+1)} &= \mathbf{H}_{12}^{\top}\mathbf{y}_{\text{obs}} + \mathbf{H}_{22}(\mathbf{f}_{\text{imp}} + \mathbf{e}^{(t)}) \\ \mathbf{f}_{\text{imp}} + \mathbf{e}^{(t+1)} &= \mathbf{f}_{\text{imp}} + \mathbf{H}_{22}\mathbf{e}^{(t)} . \end{aligned}$$

Input: $\mathbf{y}_{\text{obs}}, \mathbf{H}_{12}, \mathbf{H}_{22}$
Output: $\mathbf{f}_{\text{imp}}$
Initalize $\mathbf{f}_{\text{imp}}$
while $\mathbf{f}_{\text{imp}}$ not converged **do**
    $\mathbf{f}_{\text{imp}} \leftarrow \mathbf{H}_{12}^{\top}\mathbf{y}_{\text{obs}} + \mathbf{H}_{22}\mathbf{f}_{\text{imp}}$
end while

Alg. 4.3: Iterative transductive prediction algorithm.

So the error changes as

$$\mathbf{e}^{(t+1)} = \mathbf{H}_{22}\mathbf{e}^{(t)}\,,$$

and since it was shown that $\sigma_i(\mathbf{H}_{22}) \in]0, 1[$, for any i, it follows that

$$||\mathbf{e}^{(t+1)}|| < ||\mathbf{e}^{(t)}||\,.$$

The magnitude of the error decreases every epoch, where the rate of convergence is determined by the largest eigenvalue of $\mathbf{H}_{22}$. □

In the context of pairwise learning, Algorithm 4.3 is particularly relevant. I have noted before that the $\mathcal{O}(m^3 + q^3)$ time complexity for pairwise learning using Kronecker-based models is usually only possible to attain when the label matrix $\mathbf{Y}$ is complete. For incomplete datasets the time complexity is $\mathcal{O}(n^3)$, the time complexity of performing KRR with n dyads. Theorem 4.11 suggests a different approach:

1. Initialize the missing values of the unlabelled dyads, making the label matrix complete. This can be done by taking the average of the observed labels.

2. Fit a model using this label matrix. This step has a very low computational cost if the eigenvalue decompositions of $\mathbf{K}$ and $\mathbf{G}$ were already computed.

3. Update the missing values using the model.

4. Repeat steps 2 and 3 until convergence.

The final model, trained on the imputed dataset, can then be used for predicting Settings B, C and D.

4.7 Experiments

In the experiments I will demonstrate the learning properties of two-step KRR, compared to independent-task and Kronecker KRR. Furthermore, the experiments also demonstrate the efficient algorithms for training and evaluating the two-step KRR model, in contrast to the more limited toolkit for the Kronecker KRR model. To be more specific:

- In Section 4.7.1 I study the performance of two-step and Kronecker KRR for the four different settings on four drug-targets benchmarks. I illustrate the use of the analytical shortcuts for cross-validation.
- In Section 4.7.2, I study the imputation problem of Setting A into somewhat more detail.
- In Section 4.7.3 I use a case study of drug-target interactions to study the differences between independent-task regression, multi-task learning and zero-shot learning. Some of the learning curves can only be made without resorting to slower gradient-based optimization for training the models with two-step KRR.
- Finally, in Section 4.7.4 I demonstrate learning in mini-batches on a large-scale hierarchical text classification problem.

I refer to Romera-Paredes and Torr (2015) for some experimental results that show that two-step KRR is a competitive method compared to established zero-shot learning methods, including DAP (Lampert et al., 2014) and ZSRwUA (Jayaraman and Grauman, 2014), for

some zero-shot learning benchmark datasets. That paper is referred to for a comparison with the state-of-the-art.

4.7.1 Study of regularization for the different settings

In this section I investigate the influence of the regularization parameters λ, λ_u and λ_v of two-step and Kronecker KRR for the different Settings A, B, C and D. I will also demonstrate the scalability of the different shortcuts for cross-validation, described in Section 4.6. To this end, I use four drug-target classification datasets collected by Yamanishi et al. (2008)[2]. Each dataset concerns a different class of protein targets: enzymes (e), G protein-coupled receptors (gpcr), ion channels (ic) and nuclear receptors (nc). The properties of these datasets are given in Table 4.4 and heat maps are shown in Figure 4.5. The interactions are given in the form of a binary adjacency matrix. Both the drugs and targets come along with a respective similarity matrix. For the drugs, common substructures are calculated using a graph alignment algorithm. The Jaccard similarity measure is used to obtain a drug similarity based on these substructures. The similarity matrix of the targets is a normalized version of the scores obtained by Smith-Waterman alignment (Smith and Waterman, 1981). I rescored the labels, such that positive interactions have a value of N/N^+, while negative interactions have a value of $-N/N^-$, with N the number of pairs and N^+ and N^- the number of positive and negative interactions, respectively. By using this relabeling, minimizing the squared loss becomes equivalent with Fisher discriminant analysis (Bishop, 2006), making the method suitable for classification. See Chapter 3 for a more extensive discussion.

For each of the models, I perform leave-one-out cross-validation for new pairs, new targets and both, as described in the introduction.

[2]Available at `http://web.kuicr.kyoto-u.ac.jp/supp/yoshi/drugtarget`

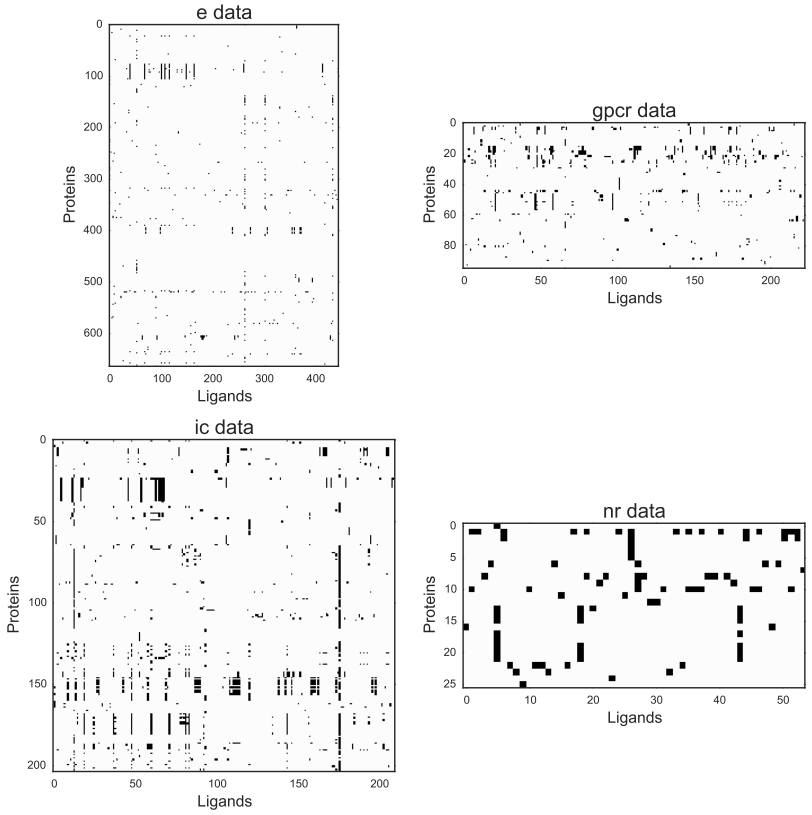


Figure 4.5: Heat maps of the four drug-target datasets of Yamanishi et al. (2008).

The cross-validated predictions are obtained using the computation short-cuts described in Section 4.6.1. For the four different cross-validation settings I use micro-wise AUC as a performance measure.

Table 4.4: Overview of the different drug-target datasets used in Section 4.7.1.

dataset	e	gpcr	ic	nr
# targets	664	95	204	26
# drugs	445	223	210	54
fraction of interactions (%)	0.99	3.00	3.45	6.41
median degree targets	2	3	5	3
median degree drugs	2	2	3	1

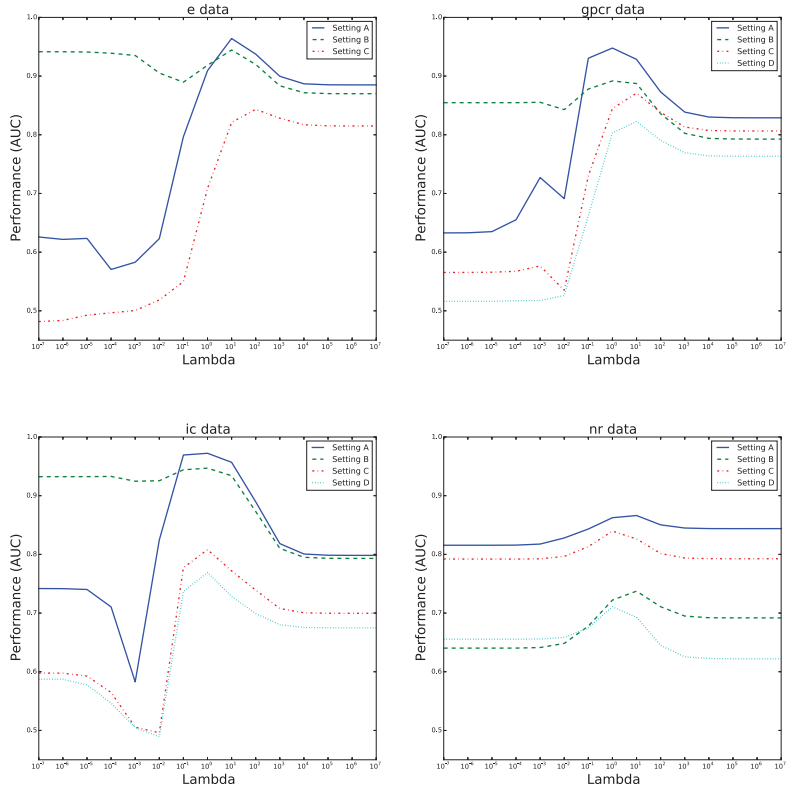


Figure 4.6: Performance for different evaluation schemes for the four drug-target datasets for different values of the regularization parameter using Kronecker KRR. The optimal regularization is dependent on the hyperparameter and validation setting.

Table 4.5: Overview of the performance and running time using Kronecker KRR (KK), two-step KRR (TS) and the two-step method with a single regularization parameter (TSS) for the different drug-target datasets and cross-validation settings. One experiment could not be completed in less than three days of running time. See main text for details. All experiments were performed using a basic Numpy implementation of the models and cross-validation shortcuts. Alle experiments were run on a AMD Opteron server (2500.159 MHz).

		best performance (AUC)				running time			
data		e	gpcr	ic	nr	e	gpcr	ic	nr
setting	method								
A	KK	0.9640	0.9478	0.9723	0.8662	7.64s	0.22s	0.54s	0.01s
A	TS	0.9644	0.9420	0.9705	0.8857	68.68s	3.14s	5.645s	0.215s
A	TSS	0.9641	0.9420	0.9611	0.8857	5.96s	0.22s	0.44s	0.02s
B	KK	0.9445	0.8918	0.9469	0.7373	1.57h	12.88s	91.14s	0.24s
B	TS	0.9476	0.9104	0.9480	0.7831	68.79s	3.17s	5.645s	0.205s
B	TSS	0.9476	0.8997	0.9480	0.7235	5.93s	0.22s	0.43s	0.01s
C	KK	0.8434	0.8709	0.8080	0.8396	0.69h	81.56s	97.13s	1.01s
C	TS	0.8506	0.8721	0.8078	0.8517	68.53s	3.12s	5.745s	0.24s
C	TSS	0.8456	0.8710	0.8029	0.8476	5.94s	0.22s	0.43s	0.02s
D	KK	-	0.8228	0.7691	0.7107	>3d	2.15h	5.48h	26.47s
D	TS	0.8277	0.8341	0.7701	0.7269	98.555s	3.855s	7.39s	0.245s
D	TSS	0.8135	0.8266	0.7701	0.7074	7.95s	0.27s	0.54s	0.01s

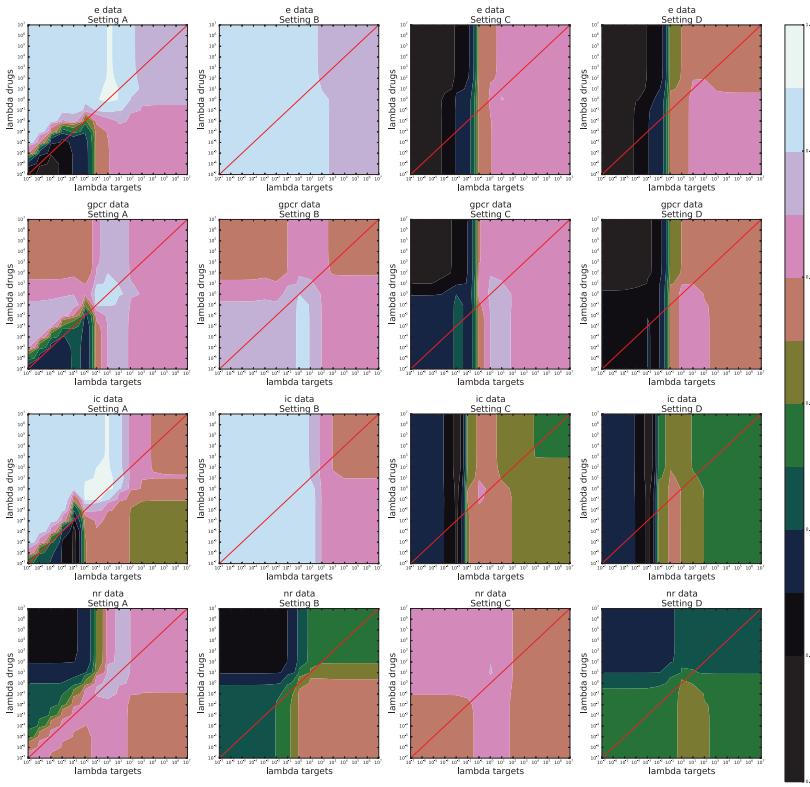


Figure 4.7: Performance for different evaluation schemes for the four drug-target datasets for different values of the regularization parameters using two-step KRR. The optimal regularization is heavily dependent of the hyperparameters and validation setting. The red diagonal indicates the search for two-step KRR with a single regularization parameter.

I trained Kronecker and two-step KRR with the regularization parameters λ, λ_u and λ_v each taken from the grid with values $10^{-7}, 10^{-6}$, $\ldots$, $10^5, 10^6$. For each (combination of) regularizer(s), the performance was calculated. For Setting A for both methods and the other settings for two-step KRR, I used the analytic shortcuts to calculate the holdout values. For the other settings using Kronecker KRR, I calculated the eigenvalue decomposition of a submatrix of the Gram matrix $\mathbf{K}$ ($\mathbf{G}$) for each row (column) for Setting B (Setting C). For

Setting D, I computed the eigenvalue decomposition of a submatrix of $\mathbf{K}$ for each row of $\mathbf{Y}$, while calculating the eigenvalue decomposition of a submatrix of $\mathbf{G}$ for each element in this row. This corresponds to performing the smallest number of computations, while only holding one eigenvalue decomposition of each Gram matrix in memory. A small modification of the two-step KRR is also explored where the two regularization parameters coincide such that $\lambda_u = \lambda_v = \lambda$. Hence, only a one-dimensional grid of hyperparameters has to be explored, resulting in a faster tuning at the cost of perhaps obtaining a slightly worse model. I will abbreviate this as two-step single hyperparameter (TSS) KRR. For every model, dataset and cross validation scheme, the best AUC is retained.

Table 4.5 shows the best performances for both methods, as well as the running times for performing the complete cross-validation and hyperparameter grid search. For the different settings and datasets, I observe that both methods have a similar performance, with two-step KRR often slightly outperforming Kronecker KRR. For both methods, Setting D is the hardest and Setting A the easiest, as expected.

When comparing the running times for model selection, I observe the computational advantage of two-step KRR. For Setting A, both methods have a holdout shortcut, hence both are fast. Kronecker KRR has to iterate over a grid of 15 regularization values, while two-step KRR has to search a grid of 15×15 regularization parameters, making it slower. Both methods are very fast in practice though. For Settings B and C, there is only an efficient algorithm for two-step KRR. For datasets larger than dataset nr, two-step KRR is much faster than Kronecker KRR. For Setting D two-step KRR is much faster compared to Kronecker KRR, where it was not even possible to do this cross-validation for the e dataset within three days.

In Figures 4.6, 4.7 and 4.8 it is visualized how the performance depends on the regularization for the different settings. The performance is quite sensitive to the value of the regularization parameter(s), a fact

well known in machine learning. The optimal regularization is also strongly determined by the cross-validation setting, especially for the two-step method. The contour plots of Figure 4.7 illustrate that for different settings a specific model has to be selected. Penalizing the instances and the tasks separately is natural for these types of learning problems, and the effect on the performance can be a valuable diagnostic tool to aid the model building. Two-step KRR allows for efficiently exploring this space for any setting, in contrast to Kronecker KRR.

From these figures it is also clear that the optimal regularization depends both on the type of dataset and on the type of predictions one is aiming at. For example, comparing the setting of new targets with new drugs for the datasets e, gpcr and ic, we see that for generalizing to new targets a low regularization for the drugs is needed, while the latter cases require the opposite. For these cases, it seems that the models can predict better for new targets compared to drugs, indicating that the feature description for the drugs is more adequate. Hence, finding a suitable model for predicting for new targets in this case is harder, indicated by smaller regions of high performance. We see that for the nr data, the regions are quite irregular and dependent on the setting, likely because this is the smallest dataset. Depending on the dataset, Setting B or Setting C is the harder one. Likely this is determined by both the number of targets and drugs in the training set, as well as the quality of the two kernel matrices. The two-step method with a single regularization parameter is even faster to tune than the two-step KRR with only a minor decrease in performance.

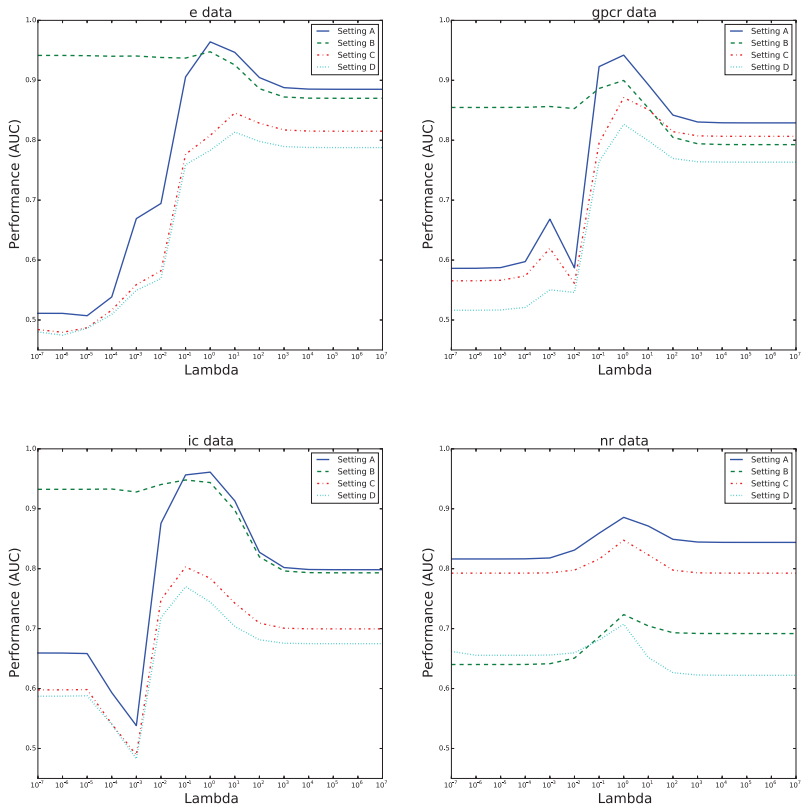


Figure 4.8: Performance for different evaluation schemes for the four drug-target datasets for different values of the regularization parameters using two-step KRR with a single regularization parameter. The optimal regularization is dependent of the hyperparameters and validation setting.

4.7.2 A study of missing value imputation

In the following series of experiments, I investigate Setting A in somewhat more detail. I will study how the Kronecker-based models discussed in this work can be used for imputing missing parts of data matrices. To be more specific, I will show how the performance deteriorates when larger parts of the label matrix are missing and how

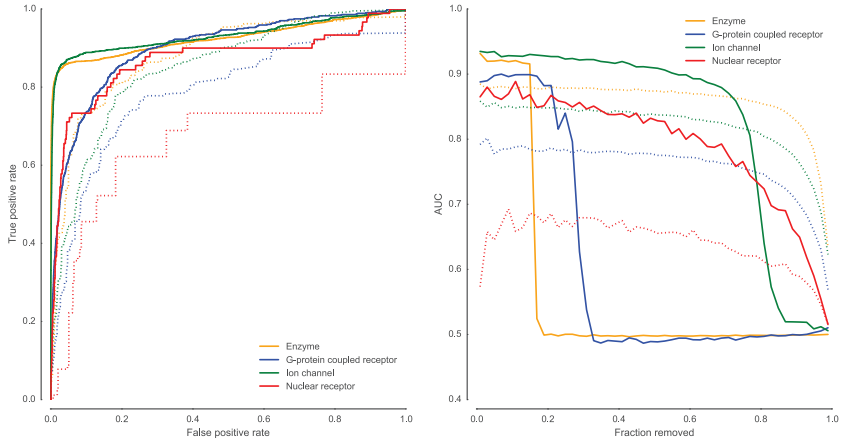


Figure 4.9: (left) ROC curves for LOO cross-validation for predicting interactions for the drug-targets datasets. (right) AUC as a function of the fractions of interaction that were removed. The performance was calculated of the held-out interactions. Solid lines indicate that feature kernels were used, dashed lines indicate that smoothing kernels were used.

imputation can be done using two-step or Kronecker KRR without any information about the objects.

The left part of Figure 4.9 depicts ROC curves for the four drug-target datasets using both feature and smoothing kernels (Eq. (4.30)) in two-step KRR when performing LOO cross-validation for Setting A. Using features for the targets and drugs results in a substantially higher performance than using only a smoothing kernel. While only exploiting the structure of the relation matrix, one still obtains a clearly better-than-random performance for the four datasets. The difference in performance between the full and dashed lines quantifies how much extra information the model extracts from the features compared to just exploiting the non-random distribution of the interactions.

In the right part of Figure 4.9 I study how the performance drops when larger fractions of the label matrix are missing, i.e. making the

label matrix incomplete. The horizontal axis indicates the fraction of the interactions that were removed (from 0.01 to 0.99 in twenty-five steps) and the vertical axis shows the AUC of the predictions for these missing values. The imputation was done using two-step KRR together with the iterative (using 20 iterations) algorithm of Section 4.6.3. I used LOO cross-validation for Setting A to determine the optimal regularization parameters. Each line is an average of 100 repetitions. When the label matrix is close to complete, the performance of the model using features is again higher than when only smoothing is used. For the enzyme and G protein-coupled receptor dataset it is interesting to note that the performance using features undergoes a transition after removing a certain fraction of interactions: the performance drops abruptly. The corresponding models using smoothing kernels do not exhibit this behaviour, on the contrary, the performance remains remarkably stable.

I investigated the behavior of the two-step KRR with larger fractions of missing values into more detail. In Figure 4.10 the 100 individual learning curves for the four datasets when using features are shown. This shows that the transition in the performance for the enzyme and G-protein coupled receptor data is rather stable. One possible explanation is that the value of the optimal regularization parameter is dependent on the number of observed interactions. In Figure 4.11, I plot the learning curves of how the micro-wise AUC deteriorates when more positive interactions are removed (i.e. the fraction of false negatives increases) for different values of $\lambda = \lambda_u = \lambda_v$. Note that here instead of removing observations as done in the previous experiment, I changed positive interactions into negative interactions. The performance is computed using LOO for Setting A and each curve is an average of 100 repetitions. It seems that the optimal regularization does not depend too much on the completeness of the dataset. Note that these curves do not exhibit the sharp decline

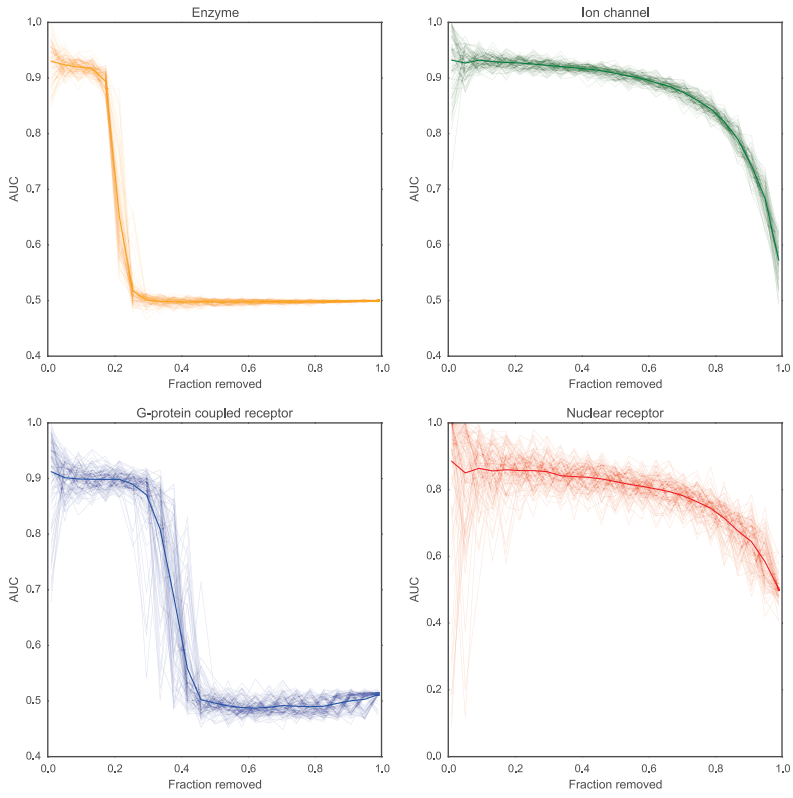


Figure 4.10: The 100 repetitions of the learning curves for the different datasets using two-step KRR with the features. The average curve (the same as in Figure 4.9) is the thick line. The variance of these curves is rather small.

in performance, this seems to be a property of having many missing values rather than false negatives.

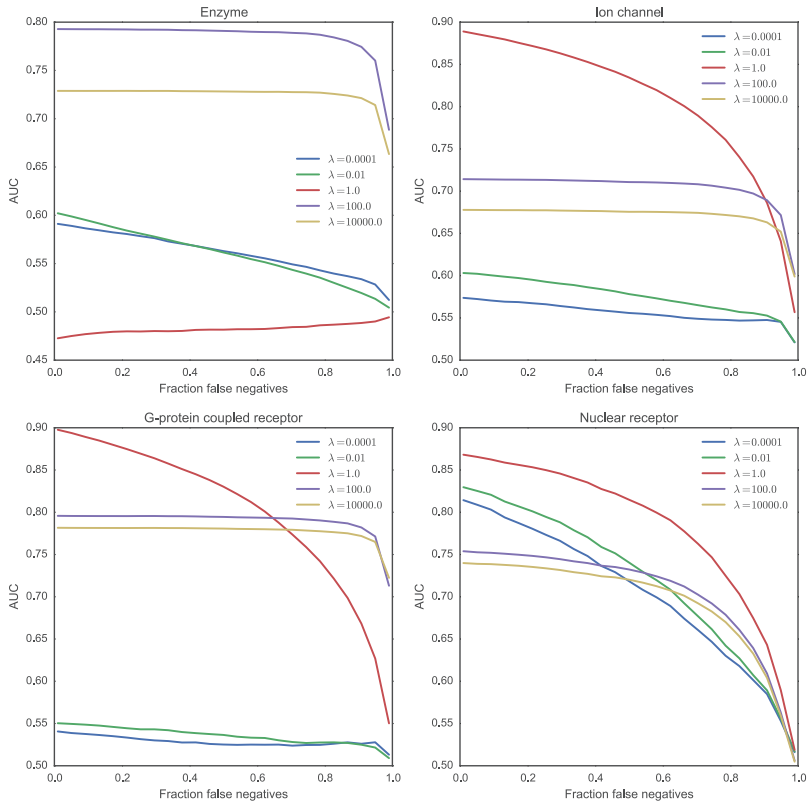


Figure 4.11: Learning curves using two-step KRR with different values for $\lambda = \lambda_u = \lambda_v$. Except for the G-protein coupled receptor dataset, the optimum value of regularization does not really change if the dataset contains a moderate to high number of false negative interactions.

The pairwise methods discussed in this work can also be applied to images. Using suitable kernels, the Kronecker-based methods can also be used as a linear image filter - see Gonzalez and Woods (2007) for an introduction. A black-and-white image is merely a matrix with intensity values for each pixel. Here, the only features for the rows and columns are the x- and y-coordinates of the pixels. For the rows (resp. columns) a kernel can be constructed that quantifies the distance between pixels in the vertical (resp. horizontal) direction. In the

experiments, I kernels which are a sum of standard radial basis kernels (Equation (3.4)) on the pixel coordinates for the rows and columns plugged in Kronecker KRR with a regularization parameter $\lambda = 0.1$. The squared bandwidths (hyperparameter γ^2 Equation (3.4)) of the two radial basis kernels are 10 and 10^5 respectively, i.e. modelling short- and long distance correlations between pixels. I illustrate the imputation algorithm on a benchmark image of a cup of coffee.

Figure 4.12 shows the example image of which parts were removed. I either randomly removed 1%, 10%, 50%, 90% or 99% of the pixels or removed a 100×400 pixels block from the image. Subsequently, the iterative imputation algorithm of Section 4.6.3 was used to impute the missing part of the image. The missing pixels were initialized with the average value of the remaining pixels in the image. The bottom of Figure 4.12 shows the mean squared error of the values of the imputed pixels as a function of the number of iterations of the algorithm. For reference purposes, the variance is also indicated, corresponding to the expected mean squared error of using the mean as imputation. In all cases, the algorithm could restore the image substantially better than the baseline. If the image is relatively complete, the imputation is quite fast; all imputations could be done in under a minute on a standard laptop. Figure 4.13 shows some of the image restorations. With 10% of the pixels missing, the restoration is visually indistinguishable of the original. Using only 10% of the pixels, a blurry image of the original can be produced. In the case where a block of the image is missing, a 'shadow' of the coffee cup can be seen, showing that the model can at least detect some high-level features of the image. Restoring an image with missing pixels is called the 'inpainting problem' (Bertalmio et al., 2000). This framework can also be linked to compressed sensing, a domain in the intersection of information theory, signal processing and machine learning with the goal of retrieving a signal from random measurements (Fornasier and Rauhut, 2011).

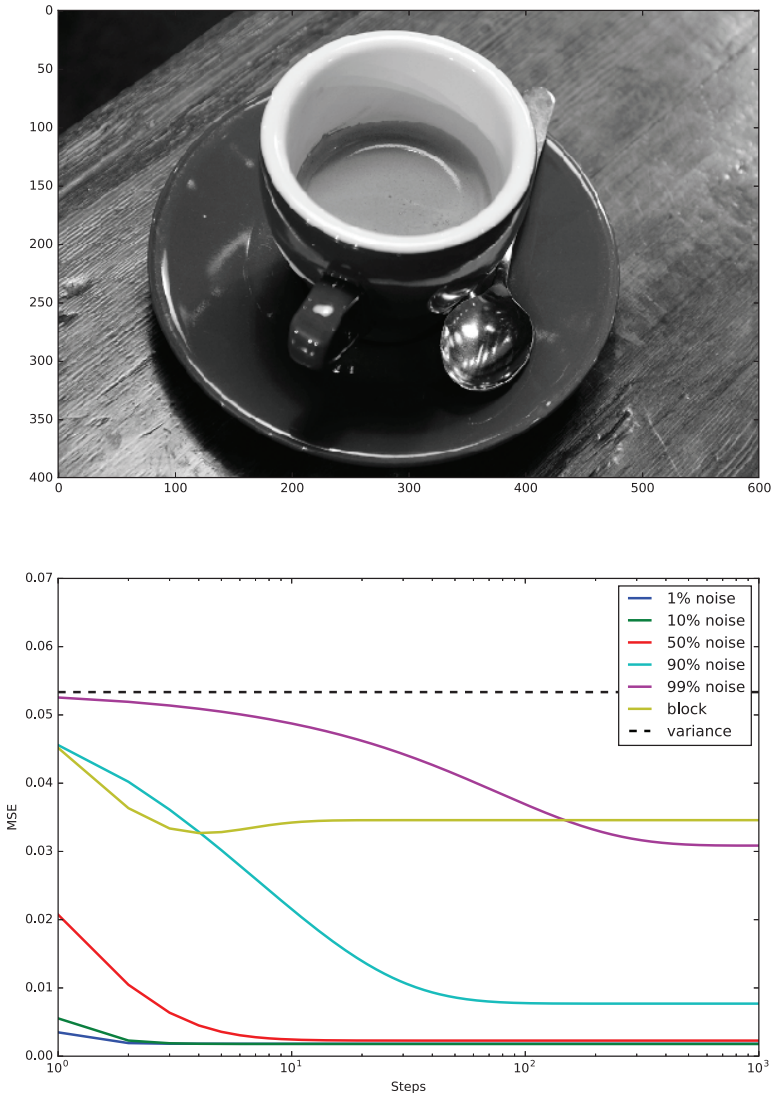


Figure 4.12: (top) An image of a cup of coffee. (bottom) Mean squared error of the imputed pixels as a function of the number of iterations of the imputation algorithm. Missing pixels are initiated with the average value of the observed pixels. The dotted line indicates the variance of the pixels, i.e. the approximate mean squared error of imputing with the average value of the imputed pixels.

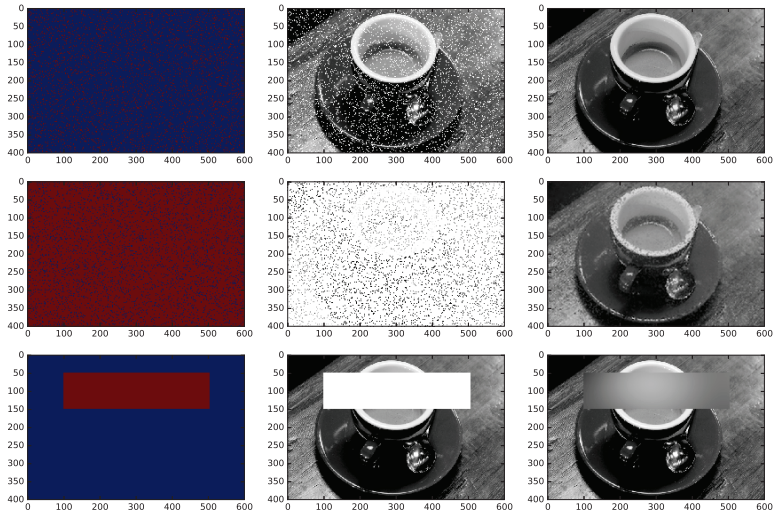


Figure 4.13: Examples of missing pixel imputation on the coffee image. (left) Mask indicating which pixels were removed, blue indicates available, red indicates missing. (middle) The coffee image with the corresponding missing pixels. (right) The restored image. (from top to bottom) 10% of the pixels randomly removed, 90% of the pixels randomly removed, a block of the image removed.

4.7.3 Comparison of different transfer learning settings

In this series of experiments, I compare different types of transfer learning settings in drug-target prediction. I simulate the zero-shot and almost zero-shot learning problem as follows. In each experiment, one drug is considered to be the target task in question, where the task is to predict the interactions of targets with respect to the drug. Further, other tasks formed in the same way are provided as auxiliary information, leading to a zero-shot learning or almost zero-shot learning setting. The experiments are performed 100 times with different training/test set splits.

The experiments were performed using two different datasets. In the first experiment, the enzyme dataset of the previous section was reused. The goal is to learn to predict for the given drug the binding with a set of targets that were not encountered in the training phase. The performances are always computed over a test set of 164 protein targets for a given task, i.e. I assess whether for a given target one can discriminate between targets with more or less affinity for this drug. The performance was measured by calculating the AUC for the test set of targets for each drug or task separately (i.e. macro AUC).

I also used a different drug-target interaction prediction dataset[3] (Davis et al., 2011; Pahikkala et al., 2015) consisting of 68 drug compounds and 442 protein targets. In contrast to the earlier drug-target datasets, this is a regression task with real-valued labels. The kernel between the drugs is based on the 3D Tanimoto similarity coefficient, and the sequence similarity between the protein targets was computed using the normalized version of the Smith-Waterman score. Furthermore, for each drug-target pair we have a real-valued label, the negative logarithm of the kinase disassociation constant K_d, which characterizes the interaction affinity between the drug and target in question. In each experiment, the task of interest corresponds to one of the drugs in the data set. The goal is to learn to predict for the given drug the K_d values for drug-target pairs unseen during the training phase. The performances are always computed over a test set of 192 protein targets for a given task, i.e. we assess whether for a given target we can discriminate between drug-target pairs with more or less affinity for this drug. The performances are averages over all repetitions and over all target tasks, and are measured using the C-index. Regularization parameter selection is performed using leave-one-out cross-validation on the training data using the shortcuts of Section 4.6.1.

For each task, I vary the number of available training drug-targets. For the enzyme dataset the number of training drug-targets is in-

[3] `http://users.utu.fi/aatapa/data/DrugTarget`

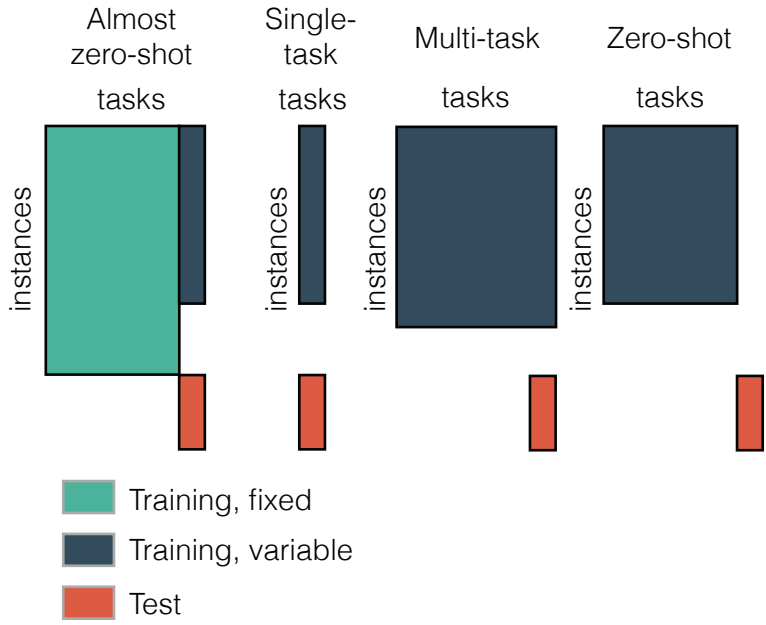


Figure 4.14: Overview of the approaches investigated in Section 4.7.3. Training data of which the size is constant in the experiments is colored in green. Training data of which the number of instances varies over different experiments is shown in blue and test data is indicated in red.

creased from 25 to 500 in steps of 25 and for the drug-target affinity the number of drug-targets is varied from 10 to 250 in steps of 5. In addition, I access to the training data for all training drug-targets for the auxiliary tasks. As summarized in Figure 4.14, I evaluate a number of different approaches:

- Independent-task: KRR trained with data from the target task only. The regularization parameter is selected based on leave-one-out cross-validation for Setting A.

- Multi-task: both the target and auxiliary tasks have the same training data available (multi-output learning leveraging task correlations, tackled with Kronecker and two-step KRR). The

regularization parameter is selected based on leave-one-out cross-validation for Setting A (Kronecker and two-step KRR) and Setting B (two-step KRR).

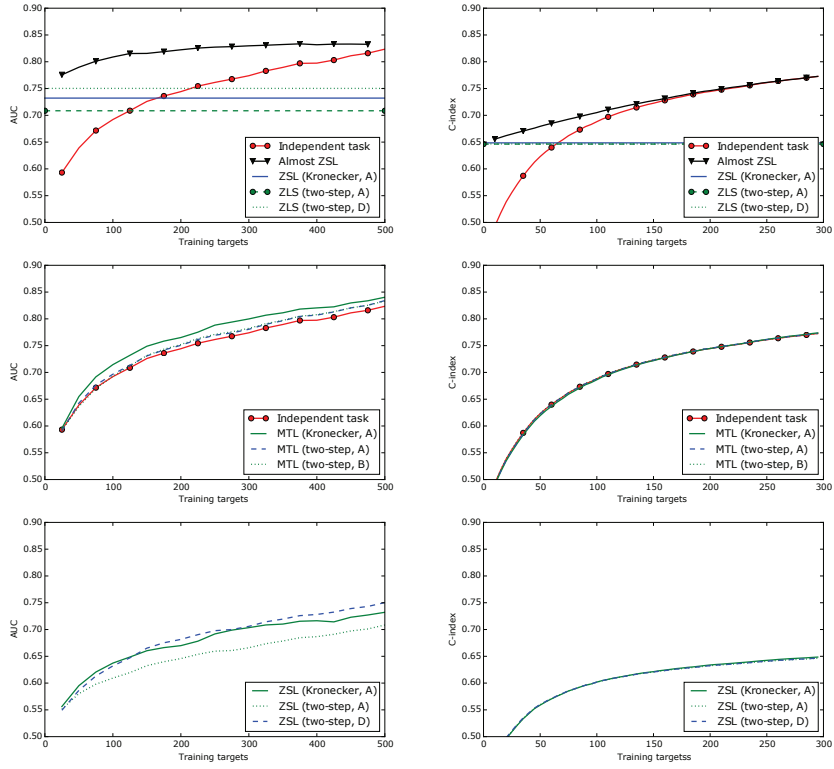
- Zero-shot learning: Kronecker KRR with no data for the target task using Kronecker and two-step KRR. The regularization parameter is selected based on leave-one-out cross-validation for Setting A (Kronecker and two-step KRR) and Setting D (two-step KRR).

- Almost zero-shot learning: using a varying amount of data from the target task, and all the available data from auxiliary tasks (tackled with two-step KRR). Here the missing labels for the target task are imputed in a first step and in a second step the 'completed' data is used to predict for the new drug-targets. For both steps the regularization parameter is selected based on leave-one-out cross-validation for Setting A.

I do not consider Kronecker KRR in the almost full cold start experiment due to computational considerations. Unlike for two-step KRR, no closed-form solution exists for the method in this setting, and the iterative conjugate gradient based method has rather poor scalability.

In Figure 4.15, I present the results for the experiments for the two datasets. The top two plots show the experiments where all the auxiliary tasks have the data for all training drug-targets, and the amount of data available for the target task is varied. It can be seen that learning is always possible even in the full zero-shot setting, where both two-step KRR and Kronecker KRR perform much better than random. For both datasets, the independent-task approach begins to outperform the full zero-shot setting after some point when one has access to enough training drug-targets. Combining these two sources of information leads to the best performance for the enzyme dataset and for the drug-target affinity dataset up until 150 training drug-targets. In both cases, using auxiliary tasks can greatly improve

performance when there are only a few labeled instances of the target task. However, once there is enough data available for the target task, there is no longer any positive transfer from the auxiliary tasks, though there also seems to be no harm from negative transfer.



(a) Enzyme dataset.

(b) Drug-target affinity dataset.

Figure 4.15: Learning curves for (a) the enzyme and (b) the drug-target affinity data. Top: target data increased, Middle: target and auxiliary data increased, Bottom: auxiliary data increased.

In the second row of Figure 4.15, I consider a setting in which there is the same amount of data available for both the auxiliary tasks and the target tasks. This setting corresponds closely to the traditional multi-output regression problem, the exception being that only the label for the target task is of interest during testing. For the enzyme dataset, Kronecker KRR slightly outperforms two-step KRR and both perform better than independent-task KRR. For the two-step method, it does not seem to matter whether the hyperparameters were selected for Setting A or Setting B. For the second dataset, we can see that the multi-task method that uses the task correlation information fails to outperform the simple independent-task method, suggesting that on this type of data one requires significantly more data in the auxiliary tasks compared to the target tasks in order for it to be helpful for learning.

In the bottom row of Figure 4.15 I consider the full zero-shot learning setting, while increasing the amount of data available for the auxiliary tasks. For the first dataset I observe that the two-step method slightly outperforms Kronecker KRR when the hyperparameters are optimized for Setting D. For the second dataset, there is no noteworthy difference between both methods. Both approaches generalize to the unknown target task, though the results are still much worse than when having a significant amount of data for the target task.

Here, two-step KRR shows itself to be competitive compared to Kronecker KRR. Previously, Schrynemackers et al. (2013) have, in their overview article on dyadic prediction in the biological domain, made the observation that in terms of prediction accuracy there does not seem to be a clear winner between the independent-task and multi-task type of learning approaches. Based on these experiments, a deciding factor whether one may expect positive transfer from related tasks seems to be the amount of data available for the target task. The two-step method performs well in the almost full zero-shot set-

tings with availability of a significant amount of auxiliary data and only very little data for the target task. But when there is enough data available for the target task, auxiliary data is no longer helpful.

4.7.4 Online-learning of hierarchical text classification

In a final experiment I study the online learning of a large-scale hierarchical text classification problem. I will demonstrate learning with mini-batches, showing that both independent-task and two-step KRR improve when iteratively adding more training data. I used the Wikipedia benchmark dataset (Partalas et al., 2015) of the Large Scale Hierarchical Text Classification Challenge[4]. I used the dataset provided for the third track, which is a subset of a larger, better curated set of another challenge. This dataset contains over 380,000 Wikipedia articles, for which the goal is to assign one of 12,633 labels, denoting the category of the article. The articles are described by a sparse bag-of-words vector with a length of 833,482. Each article can belong to only one class and the classes are part of a directed acyclic graph, representing the hierarchy of the categories (e.g. 'restricted Boltzmann machine' is a subcategory of 'artificial neural networks', which is a subcategory of 'machine learning'). For this subchallenge, the label space has been extended with 3000 novel labels. There are 6000 new articles that are labeled according to this new scheme. Here I will study the performance for both a test set of 10,000 articles with tasks seen in the training phase (Setting B or multi-task learning) and the performance on this dataset with novel labels (Setting D or zero-shot learning) as a function of the number of training articles.

The bag-of-words representation was compressed by using canonical correspondence analysis to obtain 1000 orthogonal components that are maximally correlated with the training labels. I also added

[4] http://lshtc.iit.demokritos.gr/

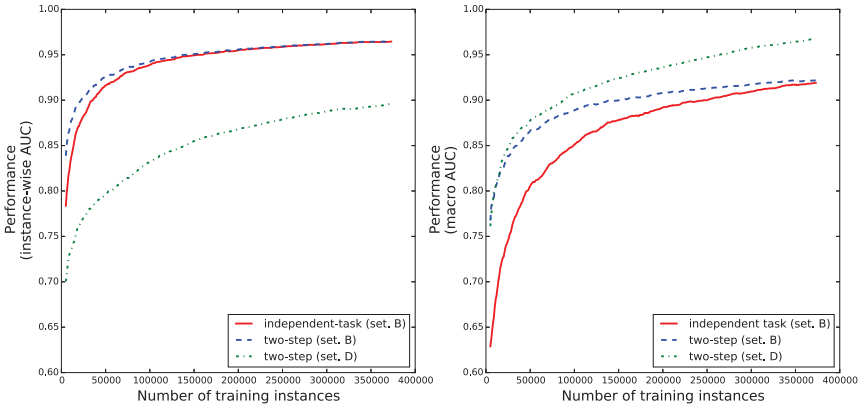


Figure 4.16: Learning curves for the different models for the Wikipedia dataset.

a constant feature so that the model could fit an intercept. For describing the tasks, I considered two approaches:

- Considering all the tasks independently, i.e. using independent-task KRR or, equivalently, $\mathbf{G} = \mathbb{I}$ for two-step KRR.
- Using features describing the hierarchy between the classes. I used the Dijkstra algorithm to generate all pairwise distances d_{ij} between the nodes of the label graph. A kernel was constructed by using a radial basis kernel with the distance in the exponent: $k_{ij} = e^{-\frac{d_{ij}}{10}}$.

The model is initially trained using 5000 instances. Subsequently, the model is iteratively updated with batches of 1000 instances using Algorithm 4.6.2. The regularization parameters λ_u and λ_u are chosen by minimizing the mean squared error[5] during leave-one-out cross-validation for Setting B (see Section 4.6.1).

The accuracy of the predictions was measured either as instance-wise AUC (i.e. the capacity of the model to discriminate between a relevant and an irrelevant label for a given article) and macro-wise

[5] I did model selection on mean squared error rather than area under the curve to speed up this procedure.

AUC (i.e. the capacity of the model to give, for a given label, a higher score to an article having the label than a an article that does not have the label). These learning curves are presented in Figure 4.16. For Setting B, for both evaluation schemes, using task features greatly improves the performance when the number of training instances is low. With more training data, both methods converge to a similar performance. Note that for the macro AUC evaluation, the learning rate is much slower compared to instance-wise AUC, implying that the former task is harder. Note that using two-step KRR, the performance is never worse than for ridge regression. It is thus advisable to start with this method when few data points are available and update using Algorithm 4.1 when more labeled instances become accessible.

For Setting D (Figure 4.16), a different pattern can be observed. First note that the AUC values cannot be directly compared with those from Setting B, since the test set contains less and different labels. Here, for both evaluations, the test performance has not converged yet, implying that more training data would be beneficial for the model. This makes sense, as assigning novel labels is much harder than known labels. Nevertheless, using two-step KRR one can both discriminate between relevant and irrelevant labels for a given article and between relevant and irrelevant articles for a given new label. For the macro AUC evaluation, the slope of the learning curve is again quite steep, indicating that for this problem more training data is required.

4.8 Conclusions and perspectives

In this chapter I studied a new two-step KRR method, and compared it to independent-task KRR and Kronecker KRR. I have shown that these methods are very related: Kronecker KRR is a special case of ordinary KRR, two-step KRR is a special case of (Kronecker) KRR, while independent-task KRR is again a special case of two-step KRR.

This unifying framework has allowed me to study the spectral interpretation for all these methods. Two-step KRR, which was found both theoretically and experimentally to work very well, has additional computational advantages. Because the model building can conceptually be decomposed into two independent steps, efficient novel holdout tricks and algorithms for online learning can be obtained. Using the shortcuts I was experimentally able to study the learning rate of our methods. All experiments illustrate that the use of task features can significantly improve performance, but careful tuning is required. An intriguing question for further research is whether it would be useful to combine other models than ridge regression or to even mix different types of spectral filters for the two steps.

Given the recent surge into fields such as zero-shot learning and extreme classification, two-step KRR has quite some potential to become a standard tool for many problems. I believe that two-step KRR, as a special case of Kronecker KRR, is particularly useful in two specific situations. Firstly, when dealing with rather small-scale interaction datasets (hundreds or thousands rows and columns) with a lot of domain knowledge. Such situations are often encountered in biological applications, e.g. molecular interaction prediction or the study of species-species interactions. In these domains, kernel-based methods are already well established for encoding prior knowledge. For such datasets, two-step KRR allows for fast and flexible model selection and validation, so that the researcher can easily explore what is possible with the data at hand. A second application would be in large-scale data applications. When dealing with huge output spaces, two-step KRR is a simple method to enforce prior knowledge on the outputs, while the suggested learning in mini-batches is an attractive alternative for gradient-based optimization.

Acknowledgements

The two-step KRR method was developed together with dr. Tapio Pahikkala and dr. Antti Airola during a research visit in the summer of 2013. They have greatly helped in constructing the theoretical framework and especially the theorems of Section 4.5 are largely their work.

5

Efficient exact top-K inference in pairwise prediction

Many complex multi-target prediction problems that concern large target spaces are characterised by a need for efficient prediction strategies that avoid the computation of predictions for all targets explicitly. Examples of such problems emerge in several subfields of machine learning, such as collaborative filtering, multi-label classification, dyadic prediction and biological network inference. Earlier, I discussed how these subfields relate to pairwise learning.

Section 2.3.1 of this PhD thesis introduced the conditional ranking problem. A subproblem of conditional ranking is top-K inference, finding the K most relevant objects based on a query and a pairwise prediction function. In this chapter I analyze efficient and exact algorithms for computing the top-K predictions in different problem settings, using a general class of models that are referred to as separable linear relational models. All the Kronecker kernel-based methods of Chapter 4 fall in this category. I show how to use those inference algorithms, which are modifications of well-known information retrieval methods, in a variety of machine learning settings. Furthermore, I study the possibility of scoring items incompletely, while still retaining an exact top-K retrieval. Experimental results in several application domains reveal that the so-called threshold algorithm is

very scalable, performing often many orders of magnitude more efficient than the naive approach.

5.1 Introduction and formal problem description

Subjected to both great expectations as well as great criticism, “Big Data" and “Big Data Analytics" are two notions that are without doubt highly popular these days. Not only does “Big Data" generate a number of interesting research questions for classical machine learning settings such as classification and regression problems, it also leads to novel challenges in less traditional learning settings. For example, though multi-target prediction is not a recent field (see e.g. Blockeel et al. (1998)), Big Data problems nowadays routinely deal with very large output spaces (e.g. tagging photos with thousands of labels, recognizing faces of millions of users). Roughly speaking, multi-target prediction can be seen as a term that intends to unify methods that are developed in several subfields of machine learning. Multi-target prediction has as general research theme the principles behind learning problems where predictions for multiple targets need to be generated simultaneously.

In this chapter, I will use the notation $f(u, v)$ to denote the predicted score of a multi-target prediction model, where v will denote an instance or query, and u a target, a label or an item. I give a couple of examples that are analyzed in more detail later in this chapter to make this point more clear:

- In collaborative filtering methods for recommender systems, $f(u, v)$ will represent a score that indicates the degree to which query v will be interested in item u. Using methods such as matrix factorization, users and items are here typically represented by

a set of latent variables – see e.g. Takács et al. (2008); Volinsky (2009).

- In multi-label classification and multivariate regression (often referred to as multi-output regression), $f(u, v)$ will represent the prediction for instance v and label u, where typically a feature representation is available for the user, whereas no additional information about u is known in the basic setting – see e.g. Agrawal et al. (2013); Dembczynski et al. (2012); Tsoumakas and Katakis (2007).

- In content-based filtering, dyadic prediction and network inference problems, feature vectors will be available for both instances v and targets u. For example, in content-based filtering, one would recommend items to users based on user profiles and side information about items (Basilico and Hofmann, 2004b; Chu and Park, 2009). In protein-ligand interaction modelling for drug design, a bio-informatics application of dyadic prediction, one would model a compatibility score $f(u, v)$ based on feature descriptions of proteins and ligands (Jacob and Vert, 2008; Wang et al., 2016).

Using a generic methodology, I analyze in this chapter efficient methods for finding the best scoring targets in the above application domains. Existing machine learning methods in those domains are often suffering from severe bottlenecks when predictions need to be generated and stored for a large number of couples (u, v). As a result, instead of computing the scores $f(u, v)$ explicitly for all couples, one could argue that it suffices to retrieve the objects u resulting in the highest scores for a given object v. For example, in computational drug design it suffices to retrieve the best-binding molecules for a given protein – see e.g. Jacob et al. (2008). Similarly, in marketing applications of recommender systems, one is often mainly interested in those products that are most relevant for a given user – see e.g. Su

and Khoshgoftaar (2009). More formally, rephrased as an information retrieval problem, we are often interested in finding the set that contains the K highest-scoring objects of this database with respect to a certain instance (or query) v.

Let me introduce some notation to make this problem statement a bit more precise. Using a generic notation, I consider two types of objects $v \in \mathcal{V}$ and $u \in U' \subset \mathcal{U}$. For simplicity, I assume that the database U' is finite with cardinality M. I will analyse bilinear models (Definition 2.11) that calculate for each couple (u, v) the following score as prediction:

$$f(u, v) = \mathbf{q}(v)^\top \mathbf{p}(u) = \sum_{r=1}^{R} q_r(v)\, p_r(u)\,. \tag{5.1}$$

As the representations need to be separated for v and u, I call the above class of models SEP-LR models. Each of the two objects is represented by an R-dimensional model vector:

$$\mathbf{q}(v) = (q_1(v), q_2(v), \ldots, q_R(v))^\top\,,$$
$$\mathbf{p}(u) = (p_1(u), p_2(u), \ldots, p_R(u))^\top\,.$$

The general applicability of SEP-LR models to the application domains that are mentioned above will be further discussed in Section 5.4. In matrix factorization methods, R will correspond to the rank of the low-rank decomposition. In multi-label classification problems, it will be the number of features.

As I am interested in computing only a subset of the predictions for couples (u, v), we define the set S_v^K, containing the K most relevant objects u for the query v. The problem that we intend to solve can then be formally written as:

$$S_v^K = \operatorname*{arg\,max}_{\mathcal{S} \in [U']^K} \sum_{u \in \mathcal{S}} f(u, v)\,, \tag{5.2}$$

with $[U']^K$ the set of all K-element subsets of U'. This is an extension to problem (2.4). Thus, for two objects $u \in S_v^K$ and $u' \notin S_v^K$, it holds that $f(u, v) \geq f(u', v)$. The set S_v^K is not necessarily unique, because ties may occur. Let us remark that problem statement (5.2) is related to nearest neighbor search, but finding the point with maximum inner product or maximum cosine similarity is not identical to finding the point that is closest w.r.t. Euclidean distance:

$$\begin{aligned}\arg\min_{u \in U'} ||\mathbf{q}(v) - \mathbf{p}(u)||^2 &= \arg\min_{u \in U'} ||\mathbf{p}(u)||^2 - 2\mathbf{q}(v)^\top \mathbf{p}(u) + ||\mathbf{q}(v)||^2 \\ &= \arg\min_{u \in U'} ||\mathbf{p}(u)||^2 - 2\mathbf{q}(v)^\top \mathbf{p}(u)\,.\end{aligned}$$

The term $||\mathbf{q}(v)||^2$ can be dropped as it remains constant as soon as v is fixed, but $||\mathbf{p}(u)||^2$ cannot be omitted. When the norm of $\mathbf{p}(u)$ is the same for each u, this expression is identical to (5.1). In many applications, the norm of the items has a clear meaning and cannot be omitted. Also, in contrast to distances, I study more general problem settings in which v and u belong to a different domain and a distance between them is less natural to define. Existing methods for speeding up nearest neighbor search, such as Elkan (2003), are hence not directly applicable to maximum inner product search.

One approach to compute the maximum inner product would be to partition the target space using efficient data structures such as k-d trees (Bentley, 1975), ball trees (Omohundro, 1989), cover trees (Beygelzimer et al., 2006) or branch-and-bound search techniques (Koenigstein et al., 2012). Methods of that kind are able to find the top-scoring predictions in an efficient way for low-dimensional Euclidean embeddings (in this case, when R is small), but they bring no improvement compared to a naive linear search when the dimensionality is larger than twenty. Space partitioning methods are hence inapplicable to the problem settings that are the focus of this chapter.

When time efficiency is more important than predictive power, one could opt for employing specialized approximate algorithms. Locality-

sensitive hashing techniques, which are popular for the related problem of nearest neighbor search, would be a good choice. Recently, a locality-sensing hashing method for maximum inner product search has been developed (Shrivastava and Li, 2014, 2015). However, in addition to delivering approximate predictions, methods of that kind are usually restricted to finding the top-1 set. Another approximate method would be to cluster the queries v in several groups, for which rankings of targets u can be precomputed by means of predictive indices and related data structures (Agarwal and Gurevich, 2012; Goel et al., 2009).

In this Chapter, I am interested in computing the set S_v^K as efficiently as possible in an exact manner. To this end, I depart from a pairwise prediction function $f(\cdot, \cdot)$ and I make a number of additional assumptions that are all quite natural for the applications that are considered. First, I assume that the top-K predictions should be returned, where $K \geq 1$. Second, I also assume that R can be bigger than twenty, so that space partitioning methods become inapplicable. Third, I assume that at prediction time queries v need to be processed one-by-one. If queries are arriving in large batches, further speed-ups could be obtained by using specialized libraries for matrix multiplication.

I analyse exact algorithms for inferring the top-K set by adopting existing methods from the database and information retrieval literature. Indeed, a strongly related problem as (5.2) is often observed in information retrieval (Ilyas et al., 2008). When queries and documents have sparse representations and relevance is defined by means of cosine similarity, one can reduce the computational complexity of retrieval by using data structures such as inverted indices (Zobel and Moffat, 2006). I will exploit a similar idea. Even though most of the specialized techniques in information retrieval put a strong emphasis on sparsity, which makes them inapplicable to multi-target prediction problems, I will show that certain techniques can be used to compute

the top-K set more efficiently compared to linear search. In particular, I am analyzing in this chapter Fagin's algorithm and extensions thereof (Fagin, 1999). Similar to inverted indices, those algorithms will score for a given query the items of several lists until one is guaranteed to have found the top-K scoring items.

The remainder of this chapter is structured as follows. I present and discuss in Section 5.2 three exact algorithms for obtaining the K highest scores for a given query. In Section 5.4 I will give an overview of machine learning problems that can be cast in our general framework, followed by Section 5.5, where I experimentally study the algorithms using datasets from several application domains. Finally, in Section 5.6, I conclude with some practical guidelines and some directions for future research.

5.2 Exact algorithms for top-K inference

The most straightforward way to solve problem (5.2) is by simply calculating the score $f(u, v)$ for each target u and retaining the K targets with the highest scores. We will call this algorithm the naive algorithm, as we have to calculate all M scores to gather the top-K highest scoring targets. Computing the score for one object has a time and space complexity of $\mathcal{O}(R)$, as this amounts to computing a weighted sum of R terms. Apart from some set operations that can be done with a time complexity of $\mathcal{O}(1)$, we also occasionally have to update the current top-K set when a new target is scored higher than the worst target in the current list. Using efficient data structures such as heaps, this can be done with a time complexity of $\mathcal{O}(\log K)$. Hence the time complexity for the naive algorithm is $\mathcal{O}((R + \log K)M)$.

I show that the problem can be solved more efficiently using exact methods that are well known in database research and information retrieval, namely Fagin's algorithm (Fagin, 1999) and the so-called threshold algorithm (Fagin et al., 2003). Both algorithms in

essence find in an efficient way the maximum of an aggregation operator $Q(z_1, ..., z_R)$. More specifically, if $z_1, ..., z_R$ are the R grades of an object, then $Q(z_1, ..., z_R)$ represents the overall grade of that object. Fagin's algorithm and the threshold algorithm both find in an exact but efficient manner the K objects in a database with highest $Q(z_1, ..., z_R)$. The algorithms assume that all variables $z_1, ..., z_R$ belong to the interval $[0, 1]$ and Q has to be an increasing aggregation operator, i.e. $Q(z_1, ..., z_R) \leq Q(z'_1, ..., z'_R)$ if $z_r \leq z'_r$ for every r.

Given a query v, problem statement (5.2) can be easily transformed into the original problem setting of Fagin by defining $z_r = q_r(v)p_r(u)$. The resulting values z_r do not necessarily belong to the interval $[0, 1]$, but they can be transformed accordingly for a fixed query v and set of targets U'. In what follows we therefore assume that Fagin's algorithm and the threshold algorithm are applicable, and I further adopt a machine learning notation in explaining the two algorithms.

Key to both algorithms is a set of R sorted lists $L_1, ..., L_R$ that contain pointers to all the targets, ordered according to each of the R model descriptors $p_r(u)$. The pseudo-code of Fagin's algorithm is given in Algorithm 5.1. It first scans the targets that are at the top of the selected lists until K targets are found that occur somewhere close to the top of each list (the random access phase). Those targets are stored in the set `targetsToCheck`. Subsequently, the score for all observed targets is calculated in a second phase, and the set S_v^K is constructed as the K targets with highest scores in this set (the sorted access phase).

It should be intuitively clear that the set S_v^K will be computed in a correct manner. The stopping criterion implies that for at least K targets the score $f(u, v)$ has been computed. The best targets in `targetsToCheck` will be returned as the set S_v^K, while the monotonicity of the scoring function guarantees that the scores for these K targets are at least as high as for any target not in the considered

Input: U', v, K, $L_1, \ldots, L_R$
Output: S_v^K

```
1: S_v^K ← ∅
2: bookkeeping[1..M] ← 0
3: targetsToCheck ← {}
4: numberOfTargetsInAllLists ← 0
5: while numberOfTargetsInAllLists < K do
6:     for r ← 1 to R do
7:         u ← get next item from L_r
8:         targetsToCheck ← targetsToCheck ∪ {u}
9:         bookkeeping[u] ← bookkeeping[u]+1
10:        if bookkeeping[u] = R then
11:            numberOfTargetsInAllLists ← numberOfTargetsInAll-
   Lists +1
12:        end if
13:    end for
14: end while
15: for u in targetsToCheck do
16:    score ← f(u, v) /* O(R) */
17:    if lowest score in S_v^K < score then
18:        update S_v^K with scored item u /* O(log K) */
19:    end if
20: end for
```

Alg. 5.1: Fagin's Algorithm.

top of the sorted lists. If $q_r(v)$ is negative, the corresponding list L_r is reverted in the first part of the algorithm. This is equivalent to transferring the sign of $q_r(v)$ to the corresponding features of $p_r(v)$, i.e. working with $|q_r(v)|$ and $-p_r(u)$. Without loss of generality, we can thus assume that the score $f(u, v)$ is increasing w.r.t. all $p_r(u)$.

The pseudo-code of the threshold algorithm is given in Algorithm 5.2. In contrast to Fagin's algorithm, this approach uses information of the query v to put emphasis on the dimensions in $\mathbf{p}(u)$ that are relevant for $\mathbf{q}(v)$. The algorithm is therefore not divided in a random and a sorted access phase. In iteration d, it scores the targets observed at depth d in the lists. It keeps popping elements from lists to obtain

Input: U', v, K, $L_1, \ldots, L_R$
Output: S_v^K

```
1: S_v^K ← ∅
2: calculated ← ∅
3: lowerBound ← −∞
4: upperBound ← +∞
5: while lowerBound < upperBound do
6:     upperBound ← 0
7:     for r ← 1 to R do
8:         u ← get next item from L_r
9:         if u ∉ calculated then
10:            score ← f(u, v) /* O(R) */
11:            calculated ← calculated ∪ {u}
12:            if lowerBound < score then
13:                update S_v^K with the new scored item /* O(log K) */
14:                if |S_v^K| = K then
15:                    lowerBound ← lowest score of a target in S_v^K /* O(1) */
16:                end if
17:            end if
18:        end if
19:        upperBound ← upperBound + q_r(v) p_r(u) /* O(1) */
20:    end for
21: end while
```

Alg. 5.2: The threshold Algorithm.

promising targets until a stopping criterion is reached, i.e. when the lowest score in the current top-K set, the lower bound, exceeds an upper bound on the values of the scores of targets that have not been investigated yet. This is summarized in the following theorem.

THEOREM 5.1. *Upon termination of the threshold algorithm, it has found the set* S_v^K.

Proof. In iteration d, let $u_{L_r(d)}$ be the target at position d of list L_r, then the upper bound is given by:

$$\text{upperBound}(d) = \sum_{r=1}^{R} q_r(u)\, p_r(u_{L_r(d)})\,. \tag{5.3}$$

For a given query v, if an item u has not occurred in any of the lists at depth d, its score $f(u, v)$ will not exceed upperBound(d). As the score can be calculated as an increasing function of the components of $p_r(u)$, it follows that upperBound(d) is greater than or equal to the score of any target not yet encountered. As a result, this is an upper bound on the scores of such targets, so it should be clear that the algorithm computes the set S_v^K in a correct manner. □

The algorithms are illustrated on a small toy example in Table 5.1. Here, a database of size $N = 10$ is queried for the top-1 set using a model with $R = 4$ components. The naive approach consists of scoring all ten items and withholding the one with the largest score. Fagin's algorithm scans the sorted lists for five steps, at that point item 5 has been encountered in each list, terminating the random access phase. All nine items seen in the random access phase are scored in the sorted access phase. The threshold algorithm terminates after two steps, when the lower bound has exceeded the upper bound. Five items are scored by this algorithm. All three methods return $\{6\}$ as the correct top-1 set, but they differ in the number of items that have been scored.

In what follows I compare the computational complexity of Fagin's algorithm and the threshold algorithm from a multi-target prediction perspective. To this end, we consider the cost associated with the number of targets that have to be scored, and derive a bound accordingly. In both algorithms, the R lists of the features for the M targets have to be sorted. Using conventional sorting algorithms, this can be done with a time complexity of $\mathcal{O}(RM \log M)$. If the targets remain unchanged, or are updated only slowly, this is an operation that has

Table 5.1: (top) Toy dataset for a given query v with $R = 4$, $K = 1$ and $\mathbf{q}(v) = (0.1, 2.5, 1, 0.5)^\top$. The best item is no. 6 and it is indicated in boldface. (left) Fagin's algorithm applied to the toy dataset for finding the top-1 set. Item no. 5 has appeared in each list at depth five, so Fagin's algorithm will stop after five iterations. We score all encountered items: $\{2, 3, 4, 5, 6, 7, 8, 9, 10\}$. The top is calculated by scoring nine items instead of ten and item 6 is returned as best scoring item. (right) Applying the threshold algorithm to the toy example. In the second step the lower bound already exceeds the upper bound and the algorithm terminates. The top-scoring target is found while only five of the ten targets are scored.

item	$p_1(u)$	$p_2(u)$	$p_3(u)$	$p_4(u)$	$f(u, v)$
1	-0.5	-1.4	-0.8	-1.0	-4.85
2	0.9	-1.9	-0.3	0.5	-4.71
3	-0.8	-0.4	-0.1	0.9	-0.73
4	-0.7	-1.7	0.2	-2.5	-5.37
5	0.8	0.2	0.0	0.7	0.93
6	1.0	1.6	0.9	-0.6	**4.7**
7	0.1	0.4	-0.6	-2.0	-0.59
8	-2.4	0.6	0.4	-0.4	1.46
9	-1.6	0.2	1.0	0.3	1.49
10	0.0	1.0	-0.6	1.4	2.6

step	L_1	L_2	L_3	L_4
1	6	6	9	10
2	2	10	6	3
3	**5**	8	8	**5**
4	7	7	4	2
5	10	**5**	**5**	9
-	1	9	3	8
-	4	3	2	6
-	3	1	7	1
-	9	4	10	7
-	8	2	1	4

step	to score	lowerBound	upperBound
1	$\{6, 9, 10\}$	4.7	5.8
2	$\{2, 3\}$	4.7	3.94
-			
-			
-			
-			
-			
-			
-			
-			

to be done only once. Consequently, the cost of sorting should not be included in the computational cost for computing S_v^K.

In Algorithm 5.1, lines 5 to 14 are concerned with finding the relevant targets to score. Suppose the lists have to be followed to a depth $D \leq M$, then this part has a time complexity of $\mathcal{O}(RD)$. The last part of Fagin's algorithm is identical to the naive algorithm, resulting in a time complexity of $\mathcal{O}(M_F R)$, because M_F targets are scored (with $M_F \leq M$). For independent lists (i.e. the position of a given target is independent for the different lists), the number of targets M_F to score is of the order $M^{\frac{R-1}{R}} K^{\frac{1}{R}}$. Following this simplifying assumption and again ignoring the cost of maintaining the current best top-K list, we obtain a time complexity of $\mathcal{O}(RM^{\frac{R-1}{R}} K^{\frac{1}{R}})$. The time complexity of Fagin's algorithm is thus less than the time complexity of the naive algorithm, but the improvement is in practice rather small. More specifically, Fagin's algorithm will calculate the fewest scores when all components of the representation $\mathbf{p}(y)$ have a very strong (positive or negative) correlation. However, if those components are highly correlated, they likely share information and perhaps some effort should be done to reduce the number of components, e.g. by means of feature selection techniques.

The complexity analysis of the threshold algorithm is rather simple: it only calculates M_T scores with $M_T \leq M$, so its computational complexity is $\mathcal{O}(M_T R)$. Fagin et al. (2003) made a distinction between the *sorted access cost*, the number of objects retrieved from the sorted lists, and the *random access cost*, the total number of objects obtained from the database under random access. The latter is related computing a score, as for this the full model vector of an object. The total cost is defined as follows.

Definition 5.1 (Middleware cost). *Let S be the sorted access cost, and let R be the random access cost. For a given algorithm $\mathcal{A}$, database*

U' and query object v the middleware cost is given by

$$\text{cost}(\mathcal{A}, U', v) = c_1 S + c_2 R\,,$$

with c_1 and c_2 two positive constants.

It has been shown in Fagin et al. (2003) that the threshold algorithm is instance-optimal, meaning that the algorithm cannot be outperformed by any other algorithm when wild guesses are not allowed.

Definition 5.2 (Instance optimality). *Let $\mathbb{A}$ be a class of algorithms, let $\mathbb{U}$ be a class of target sets and let $\text{cost}(\mathcal{A}, U', x)$ be the cost when running algorithm $\mathcal{A}$ on target set U' for query v. We say that an algorithm $\mathcal{B}$ is instance-optimal over $\mathbb{A}$ and $\mathbb{U}$ if $\mathcal{B} \in \mathbb{A}$ and if for every $\mathcal{A} \in \mathbb{A}$, every $U' \in \mathbb{U}$ and every v in $\mathcal{V}$ it holds that*

$$\text{cost}(\mathcal{B}, U', v) = \mathcal{O}(\text{cost}(\mathcal{A}, U', v))\,.$$

The above equation means that there exist constants c and c' such that $\text{cost}(\mathcal{B}, U', v) \leq c \times \text{cost}(\mathcal{A}, U', v) + c'$ for every choice of v, $U' \in \mathbb{U}$ and $\mathcal{A} \in \mathbb{A}$.

In other words, if an algorithm $\mathcal{B}$ is instance-optimal, then for any query v no algorithm that obtains a lower time complexity exists. However, it can still be the case that another instance-optimal algorithm will need less computations than $\mathcal{B}$, but the difference would then be attributed to a constant time factor. This type of optimality is in fact much stronger than optimality in the average case or worst case: it holds for every query. We can easily show that the threshold algorithm is instance-optimal for the problem that is the main interest of this paper. To this end, we have to make a restriction to algorithms that do not make wild guesses.

DEFINITION 5.3 (Wild guess). *An algorithm for solving (5.2) is said to make a wild guess if it computes for a given target u the score $f(u, v)$ before u has been observed in any of the lists $L_1, ..., L_R$.*

THEOREM 5.2. *Let $\mathbb{U}$ be the class of all possible target sets U'. Let $\mathbb{A}$ be the class of algorithms that solve for any query v problem (5.2) in an exact manner without making wild guesses. Then the threshold algorithm is instance-optimal over $\mathbb{A}$ and $\mathbb{U}$.*

This result follows immediately from Theorem 6.1 in Fagin et al. (2003) by transforming problem statement (5.2) to the original problem setting of Fagin, as discussed more formally above, and observing that this leads to searching for the maximum of a monotonic aggregation operator. Crucial in the above theorem is that we make a restriction to algorithms that do not make wild guesses. Due to lucky shots, algorithms that make wild guesses would be able to find the top-K set more rapidly, but they would not outperform a deterministic algorithm for general queries v. As I focus in this chapter (and thesis) on deterministic algorithms, I omit further details of the comparison with stochastic methods, but the analysis becomes much more delicate if one would like to extend the above theorem beyond the class of deterministic algorithms. I refer to Fagin et al. (2003) for more details. The next statement formally shows that Fagin's algorithm cannot be instance-optimal.

THEOREM 5.3. *Let $\mathbb{U}$ be the class of all possible target sets U'. Let $\mathbb{A}$ be the class of algorithms that solve for any query v problem (5.2) in an exact manner without making wild guesses. Then Fagin's algorithm is not instance-optimal over $\mathbb{A}$ and $\mathbb{U}$.*

Proof. To show that Fagin's algorithm is not instance-optimal, it suffices to prove that it has a larger time complexity than the threshold algorithm for one particular query v. Adopting the same notation as in Table 1, let us assume that $\mathbf{q}(v) = (1, 1)$ with $R = 2$ and let us

define the lists L_1 and L_2 as in Table 5.2. I consider in this example a dataset that contains M targets and we assume that the values of $p_1(u)$ increase with the indices $\{1, \dots M\}$ of the items, while the values of $p_2(u)$ decrease with the indices. With the construction of $p_1(u)$ and $p_2(u)$ as in the table, Fagin's algorithm needs $M/2$ steps to terminate, whereas the threshold algorithm only needs two steps, independent of M. So, the former has for this artificial dataset a time complexity of $\mathcal{O}(M)$, whereas the latter has a complexity of $\mathcal{O}(1)$. As a result, Fagin's algorithm cannot be instance-optimal. □

The above theorem confirms that datasets can be found where Fagin's algorithm will suffer from a higher time complexity than the threshold algorithm. Conversely, due to the instance-optimality of the threshold algorithm, one cannot find datasets where this threshold algorithm will exhibit a higher time complexity than Fagin's algorithm. The next theorem supports a related, but different claim.

THEOREM 5.4. *For any query v and for any possible target set U', the threshold algorithm never computes more scores $f(u, v)$ than Fagin's algorithm.*

Proof. For simplicity I give a proof by contradiction for the case $K = 1$ (top-1). An extension for the case $K > 1$ is then immediate. Let us assume that Fagin's algorithm terminates at depth d in the lists L_i. Suppose also that the threshold algorithm needs to compute more scores than Fagin's algorithm, implying that it terminates at a depth larger than d. At depth d an upper bound on the score of targets that have not been observed yet is given by Eq. (5.3). At depth d, a lower bound on the highest score is given by the target that has been observed in all lists L_i when Fagin's algorithm reaches this depth. We know that at depth d this lower bound exceeds the upper bound in Eq. (5.3). As a result, the threshold algorithm should also stop at this depth, so this is a contradiction. □

Table 5.2: Hypothetical dataset used to show that Fagin's algorithm is not instance-optimal for $R = 2$. (top) A dataset containing M targets and the scores when $\mathbf{q}(x) = (1, 1)$. (middle and bottom) An overview of the steps needed to find the top-1 set for Fagin's algorithm and the threshold algorithm. The former needs $M/2$ steps to terminate, whereas the latter only needs two steps, independent of M. I assume in this example that the values of $t_1(y)$ increase with the indices $\{1, \dots M\}$ of the items, while the values of $t_1(y)$ increase with the indices. For simplicity, ties occur among targets $2, 3, ..., M-1$. Ties can be easily removed by constructing a more complicated example.

item	$t_1(y)$	$t_2(y)$	$f(u, v)$
1	1.1	0.1	1.1
2	0.5	0.5	1.0
3	0.5	0.5	1.0
⋮	⋮	⋮	⋮
$M-1$	0.5	0.5	1.0
M	0.1	1.0	1.1

step	L_1	L_2
1	1	**M**
2	2	M-1
3	3	M-2
⋮	⋮	⋮
$M/2$	$M/2$	$M/2$
$M/2+1$	$M/2+1$	$M/2-1$
⋮	⋮	⋮
M	**M**	1

step	to score	lowerBound	upperBound
1	$\{1, M\}$	1.1	2.0
2	$\{2, M-1\}$	1.1	1.0
-			
⋮			
-			
-			
⋮			
-			

The threshold algorithm will never compute scores for more targets than Fagin's algorithm, but this does not imply that it will always need less computations. This is due to the fact that it is characterized by some additional overhead for computing the lower and upper bound in each iteration, and for verifying whether the first bound exceeds the latter. This additional overhead does not influence the time complexity of the threshold algorithm, but it explains why constants need to be considered in the definition of instance-optimality.

In contrast to Fagin's algorithm, the threshold algorithm does not require a large buffer size, as only the upper and lower bounds are needed to scan the lists. Consequently, the buffer size of the threshold algorithm is bounded, while this is not the case for Fagin's algorithm. The latter will hence suffer from a too large memory consumption, making it a practically less useful method, especially for high-dimensional problems.

Sorting the lists can be done offline and parallel extensions can be easily implemented (Fan and Huai, 2014). If speed predominates over accuracy, the threshold algorithm can be halted before the stopping criterion is reached, so that the top-K set is potentially not correct, but the running time is shorter. For example, if the threshold algorithm would be terminated in the first step of the example above, we would have obtained the correct top by scoring only three targets. This modification is known as the halted threshold algorithm (Fagin et al., 2003). Later on in the experiments I show that this heuristic could yield quite satisfactory results.

The threshold algorithm can also elegantly deal with sparse data. If $\mathbf{q}(v)$ and $\mathbf{p}(u)$ are large, sparse, non-negative feature vectors (e.g. in memory-based collaborative filtering or when dealing with bag-of-word features), only the non-zero elements and corresponding pointers have to be stored. By using sparse vector-vector multiplication implementations, the calculation of a single score can be reduced. The sorted lists L_r only need to contain the pointers for items where $p_r(u)$

is positive. Furthermore, one only has to consider lists corresponding to non-zero elements of $\mathbf{q}(v)$ (Algorithm 5.2, line 7). Thus, in addition to improving the computing time for calculating a single score (which can also be done using the naive algorithm), much fewer items have to be scored. I will demonstrate this in the experimental section.

Despite the strong theoretical result above, I show that it is still possible to slightly improve the threshold algorithm in some cases. The reason is that Theorem 5.2 does not take the dimensionality R of feature vectors into account. I propose a small modification so that not all scores have to be calculated completely. Let us assume that the following double inequality holds for the score of a newly-observed target u at depth d of the threshold algorithm:

$$f(u,v) \leq \sum_{r=1}^{l} q_r(v)\, p_r(u) + \sum_{r=l+1}^{R} q_r(v)\, p_r(u_{L_r(d)}) \leq \text{lowerBound}(d)\,, \tag{5.4}$$

with $1 \leq l < R$. For such a target it is not needed to calculate the score entirely. Algorithm 5.3 modifies the scoring functionality of the original threshold algorithm by applying this idea. I will refer to this modification as the partial threshold algorithm. It stops the calculation when it becomes clear that the score cannot improve the lower bound. We start from the upper bound and gradually update the score to either obtain the full score or to halt when the partially calculated score is lower than the lower bound. The partial threshold algorithm will return the same top-K set and is still an exact algorithm. The number of items that are considered is the same as with the threshold algorithm, only the calculation of some scores that do not improve the top will be halted early, resulting in a decrease in computations.

Input: $\mathbf{q}(v)$, $\mathbf{p}(u)$, upperBound, lowerBound, $L_1, \ldots, L_R$, d
Output: score or fail

score ← upperBound
for $r \leftarrow 1$ **to** R **do**
    score ← score - $q_r(v)\, p_r(u_{L_r(d)})$
    score ← score + $q_r(v)\, p_r(u)$
    if score $\leq$ lowerBound **then**
        break and return a fail /* *item u will not improve* S_v^K */
    end if
end for

Alg. 5.3: Calculating the scores for the Partial Threshold Algorithm

5.3 Recommending recipes: an illustrative example

Everyone likes to eat. It is deciding what to eat that is often the hard problem. In this section I will try to further clarify the Threshold Algorithm by developing a toy recipe recommendation engine. The idea is simple; we have a set of ingredients in our fridge that we would like to combine and we want to find a suitable recipe for this. Suppose our query is v= {egg, chocolate, bacon}. We have also a database of recipes from which to select a suitable recipe. Each recipe is only described by the presence or absence of a particular ingredient. The database of nine recipes is given below.

	egg	cream	sugar	chocolate	tomato	pasta	cheese	bacon
spaghetti Bolognese	0	0	0	0	1	1	1	0
scrambled egg	1	0	0	0	0	0	1	1
pasta carbonara	1	1	0	0	0	1	1	1
créme brûllée	1	1	1	0	0	0	0	0
cheesecake	1	0	1	0	0	0	1	0
chocolate bacon pie	1	0	1	1	0	0	0	1
chocolate mousse	1	1	1	1	0	0	0	0
meal salad	1	0	0	0	1	0	1	1
whipped cream	0	1	1	0	0	0	0	0

One metric that could be used to score the relevance of a recipe with respect to a query is the Jaccard similarity coefficient (i.e. the number of ingredients that are in both the query and the recipe divided by the total number of ingredients in the query or the recipe). However, we would like to incorporate some prior knowledge about what kind of recipes we would like to have returned. Firstly, not all ingredients have the same importance to the identity of a recipe. For example, a lot of the recipes contain egg and few contain bacon. Our metric should thus put a higher emphasis on rarer ingredients because, so is our hypothesis, these contribute more to the uniqueness of the recipe. Furthermore, we should also provide some sort of scaling, as not to only favour recipes with many ingredients that are more likely to have ingredients matching the query. For this reason, I do a rescaling of this data to obtain better representations $\mathbf{p}(u)$ and $\mathbf{q}(v)$ of the ingredients:

1. Each ingredient i is weighted according to $\sqrt{\log(1/p_i)}$, with p_i the probability of ingredient i occurring in a random recipe. This is inspired by Kim et al. (2015) and makes that the dot product will be a sum of the information entropies for each in-

gredient. Or, more simply, rarer ingredients receive a higher score.

2. The representations $\mathbf{p}(u)$ and $\mathbf{q}(v)$ are standardized by dividing the model vector by its vector norm, e.g. $\mathbf{p}'(u) = \frac{\mathbf{p}(u)}{\|\mathbf{p}(u)\|_2}$. So recipes with fewer ingredients get a larger weight per ingredient.

The rescaled database is shown below.

	egg	cream	sugar	chocolate	tomato	pasta	cheese	bacon
spaghetti Bolognese	0	0	0	0	.65	.65	.40	0
scrambled egg	.39	0	0	0	0	0	.60	.70
pasta carbonara	.25	.45	0	0	0	.62	.39	.45
créme brûllée	.39	.70	.60	0	0	0	0	0
cheesecake	.42	0	.64	0	0	0	.64	0
chocolate bacon pie	.28	0	.43	.69	0	0	0	.51
chocolate mousse	.28	.51	.43	.69	0	0	0	0
meal salad	.28	0	0	0	.69	0	.43	.51
whipped cream	0	.76	.65	0	0	0	0	0

Similarly, the model vector for the query is then

$$\mathbf{q}(v) = [0.16, 0, 0, 0.87, 0, 0, 0, 0.47]^\top .$$

The score for each recipe with respect to the query is then simply the dot product between the two description vectors (Eq. (5.1)).

In order to find the top-1 set (i.e. the most relevant recipe for this query), the Naive Algorithm would need to score every recipe in the database. As discussed in the previous section, the Threshold Algorithm can be used to speed up this procedure. First, we have to construct the R sorted lists, one list of recipe indices for each ingredient. Note that this is a sparse problem in the sense that most of

the ingredients do not appear in the query. Since missing ingredients do not contribute to the score, we can omit these from the query, database and sorted lists. Below are the relevant sorted lists (left) and the database with the scored items.

	L_1	L_4	L_8
1	5	6	2
2	2	7	6
3	4	1	8
4	6	2	3
5	7	3	1
6	8	4	4
7	3	5	5
8	1	8	7
9	9	9	9

	id	egg	chocolate	bacon	score
spaghetti Bolognese	1	0	0	0	
scrambled egg	2	.39	0	.70	.52
pasta carbonara	3	.25	0	.45	
créme brûllée	4	.39	0	0	
cheesecake	5	.42	0	0	.13
bacon chocolate cake	6	.28	.69	.51	.90
chocolate mousse	7	.28	.69	0	.62
meal salad	8	.28	0	.51	
whipped cream	9	0	0	0	

In the first step, highlighted in yellow, we have to score recipes $\{2, 5, 6\}$. This leaves us with the top-1 set $S_v^1 = \{6\}$. The score of recipe 6 was 0.90, hence the lower bound is also equal to 0.90 (remember that the lower bound is the lowest score in the current top-K set). The upper bound can be computed using Eq. (5.3):

$$\text{upper bound} = .39 \times .16 + .69 \times .87 + .51 \times .47 = 0.93\,.$$

Since the upper bound exceeds the lower bound, we are at the moment not certain that the current top-1 set is correct. In the next iteration, highlighted in orange, we pop new recipe indices from the sorted lists. Only recipe number 7 is new. After scoring this, we cannot improve the top-1 set and thus the lower bound also remains unchanged. The upper bound has dropped to $.39 \times .16 + 0 \times .87 + .51 \times .47 = 0.30$ and this indicates that the current top-1 set is correct. The most relevant

recipe to this query is the chocolate bacon pie, some kind of comfort food for depressed people. Yes, this really exists[1]. We have obtained this result by only needing to score four recipes, in contrast to nine required scorings for the Naive Algorithm.

5.4 Application domains and relationships with SEP-LR models

In this section I illustrate that many multi-target prediction methods are specific instantiations of SEP-LR models and the inference methods discussed in the previous sections. The methods that I consider in Section 5.4.1 are frequently encountered in the area of recommender systems (discussed in Section 2.4.3). The example on recipe recommendation developed in the previous section is an example of collaborative filtering. The methods that I discuss in Sections 5.4.2 and 5.4.3 are used in other domains as well.

For the application domains discussed below, many of the currently used techniques boil down to SEP-LR models. When that is the case, the algorithms of Section 5.2 can be applied to decrease the running time at no cost in predictive performance, as we can guarantee that the correct top-K predictions will be returned. When a non-SEP-LR model is used, it may be considered to switch to a SEP-LR model to perform queries potentially faster. This may lead to a decreased performance as one is restricted to linear models. Depending on the application this trade-off may be worth considering or not.

[1] `http://www.wineandglue.com/2013/05/chocolate-bacon-bomb-pie-gyco.html`

5.4.1 Memory-based and model-based collaborative filtering

In this section I discuss how our methodology can be applied to so-called user-based and item-based collaborative filtering. Since these methods directly process the available dataset, they are often denoted as memory-based collaborative filtering. To keep the discussion accessible, let us focus on the latter of the two settings, where recommendations are made by retrieving items that are similar to the items a user has seen before – see e.g. Sarwar et al. (2001). A popular similarity measure in this area is the cosine similarity:

$$\cos(u, v) = \frac{\mathbf{x}_u^\top \mathbf{x}_v}{\|\mathbf{x}_u\|_2 \|\mathbf{x}_v\|_2}\,, \tag{5.5}$$

in which v should be interpreted as an item that has been seen before, whereas u rather refers to an item that could be recommended. The vectors $\mathbf{x}_u$ and $\mathbf{x}_v$ here typically consist of ratings or zero-one purchase flags for different users. Using this notation, item-based collaborative filtering can be interpreted as a specific case of model (5.1) by defining

$$\mathbf{q}(v) = \frac{\mathbf{x}_v}{\|\mathbf{x}_v\|_2} \quad \text{and} \quad \mathbf{p}(u) = \frac{\mathbf{x}_u}{\|\mathbf{x}_u\|_2}\,. \tag{5.6}$$

A similar reasoning can be followed for user-based collaborative filtering methods and other similarity measures such as the Pearson correlation and the adjusted cosine similarity. Both measures can also be written as dot products, but sparsity will be lost due to centering.

In contrast to the memory-based approach, collaborative filtering can also be performed by building a model to explain the preference of a user for a particular target. This is referred to as model-based collaborative filtering and includes models such as Bayesian networks, clustering and matrix factorization methods (Lee et al., 2011). Here, each target and each user are associated with a vector of R latent variables of which the dot product represents the joint preference. A

matrix $\mathbf{Y}$ containing the ratings for each combination of objects v and u is approximated as a product of two low-rank matrices: $\mathbf{Y} \approx \mathbf{PQ}$. This means that the predicted scores in model (5.1) can be written in matrix form as $\mathbf{F} = \mathbf{PQ}$, such that the score $f(u_i, v_j)$ of the i-th user and j-th item is computed as a dot-product of the i-th row in $\mathbf{P}$ and the j-th column in $\mathbf{Q}$. As a result, the connection with SEP-LR models holds for basically all matrix decomposition algorithms, including non-negative matrix factorization, independent component analysis, sparse principal component analysis and singular value decomposition methods (Hastie et al., 2001).

5.4.2 Multi-label classification, multivariate regression and multi-task learning

In this section I discuss applications of SEP-LR models in three large subfields of machine learning that have a lot of commonalities, namely multi-label classification, multivariate regression and multi-task learning. The former two subfields consider the simultaneous prediction of multiple binary or real-valued targets, respectively (Dembczynski et al., 2012; Tsoumakas and Katakis, 2007). Multi-task learning then further unifies those subfields, and further extends them to problems where not all targets are observed (or relevant) for all instances (Ben-David and Schuller, 2003; Caruana, 1997). However, all multi-label classification and multivariate regression problems can also be solved with multi-task learning methods. Of course, finding the top-K set is only relevant when an ordering of the targets is appropriate. For example, consider a protein-ligand interaction prediction problem where the goal is to find, for a given ligand, a set of proteins from a large database that are likely to show affinity towards it. If the model that is used predicts binding energy, the problem is a multivariate regression problem. When the model returns a value that is related to the

probability that a given ligand will interact with a particular protein or not, we are in a multi-label classification setting.

Using the notation introduced before, this means that v represents an instance and u a target (a.k.a. label or output). As we will be computing the K most relevant targets for a given instance v, we end up with a so-called multi-label ranking setting. Furthermore, suppose we have M targets and a set of N_{tr} training instances for which a feature representation $\boldsymbol{\psi}(x)$ is available.

Many multi-label classification, multivariate regression and multi-task learning methods can be formulated as SEP-LR models. Due to lack of space, I only discuss a few representative cases. The first approach is a very simple approach: it defines independent models for different targets. This method is known as the binary relevance method in the multi-label classification community (Tsoumakas and Katakis, 2007). Using linear models or linear basis function models, this implies that the model for target u can be represented as

$$f(u, v) = \mathbf{w}_u^\top \boldsymbol{\psi}(v) \,.$$

Hence, this model is a specific case of the SEP-LR framework, with $\mathbf{q}(v) = \boldsymbol{\psi}(v)$ and $\mathbf{p}(v) = \mathbf{w}_u$.

Multivariate (kernel) ridge regression, or independent task KRR, is probably the most basic model of this kind. Ridge regression constructs a model for every target separately, ignoring potential dependencies between targets. Such dependencies can be modelled using more sophisticated methods. In multivariate regression, dependencies between targets are often modelled using (kernel) PCA, which transforms the problem into a set of uncorrelated tasks (see e.g. Weston et al. (2007); Song et al. (2013)). An alternative but related approach consists of using a specialised loss function or regularisation term that enforces models of different targets to behave more similar, e.g. Evgeniou (2005); Evgeniou and Pontil (2004); Jalali et al. (2010). An interesting example from this point of view is the partial least squares

(PLS) algorithm, which projects the features onto a lower-dimensional subspace (Shawe-Taylor and Cristianini, 2004). The inference algorithms discussed in this work are applicable to all these methods.

5.4.3 Pairwise learning, content-based filtering and supervised network inference

Finally, there are applications where feature representations are available for both v and u, resulting in separable linear relational models of the following type:

$$f(u, v) = (\boldsymbol{\psi}(v) \otimes \boldsymbol{\phi}(u))^\top \mathrm{vec}(\mathbf{W}) = \boldsymbol{\phi}(u)^\top \mathbf{W} \boldsymbol{\psi}(v)\,.$$

Here, vec is the vectorisation operator, which stacks the columns of a matrix into a vector and $\otimes$ is the Kronecker product. The second part of the equation is used to stress that this can be seen as a standard linear model to make predictions for a pair of objects. This model can be cast in the form of Eq. (5.1) by writing the terms explicitly:

$$f(u, v) = \sum_{i,j} W_{ij} \boldsymbol{\phi}(u)_i \boldsymbol{\psi}(v)_j\,.$$

Models of this type are frequently encountered in several subfields of machine learning, such as pairwise learning, content-based filtering, biological network inference, dyadic prediction and conditional ranking. They have been discussed at length in Chapter 4 and in Chapter 6 I will present a case study where pairwise ranking is used to infer the function of enzymes.

5.5 Experimental results

The goal of this experimental section is twofold. First, I want to show that the proposed methods can be applied to the different machine

learning settings described above. In addition to that, I also want to illustrate the scalability and practical use of the algorithms. By means of several case studies using different datasets I will elucidate the algorithms of Section 5.2. Fagin's algorithm is included in this manuscript for didactic interest, but I observed in initial experiments that it could not cope adequately with most of the higher-dimensional problems. The buffer needed to store targets that are encountered in the lists grows rather quickly when the dimensionality of the data increases. Hence, I have omitted experiments with this algorithm. Instead, for the experimental validation I focus on the practical use of the threshold and partial threshold algorithm and compare their performance with the naive algorithm.

5.5.1 Data dependencies in collaborative filtering

In this series of experiments I study the scalability of the threshold algorithm for a collection of collaborative filtering datasets. I use five sparse datasets for memory-based and model-based collaborative filtering using the cosine similarity and matrix factorization respectively. The datasets are listed in Table 5.3. I use the Movielens dataset with both 100K and 1M observations and the BookCrossing dataset, both from Grouplens[2], the Audioscrobbler dataset[3], and a recipe composition dataset[4]. I have removed empty rows and columns and taken the logarithm of the positive values of the Audioscrobbler dataset. In the Audioscrobber and Recipes datasets the scores are obtained by implicit feedback (i.e. a zero indicates no observation for a given item-

[2]http://grouplens.org/

[3]http://www-etud.iro.umontreal.ca/~bergstrj/audioscrobbler_data.html

[4]This dataset (Ahn et al., 2011) is strictly speaking not a collaborative filtering benchmark dataset, but it can be treated using a similar workflow. Here the goal would be to find related recipes for a particular ingredient combination. See (De Clercq et al., 2015) for a recommender system built using this dataset.

Table 5.3: Overview of the properties of the different datasets used for the collaborative filtering experiments using memory- and model-based methods.

Name	# rows	# colums	# non-zero elements	Feedback
Audioscrobbler	73,458	47,085	656,632	implicit
BookCrossing	105,283	340,538	1,149,780	explicit
Movielens100K	943	1,682	100,000	explicit
Movielens1M	6,040	3,952	1,000,000	explicit
Recipes	56,498	381	464,407	implicit

user pair), for the others by explicit feedback (i.e. a zero indicates no match for a given item-user pair).

For memory-based collaborative filtering, I used the cosine similarity to define a set of items with a high similarity to the query. As described above, if each item is normalized, such that the L_2-norm is equal to one, a dot product is equivalent to the cosine similarity. For model-based collaborative filtering, I applied matrix factorization by means of probabilistic PCA (Tipping and Bishop, 1997), having as advantage that a computationally efficient expectation maximization algorithm can be used. I used 5, 10, 50, 100 and 250 latent features in the decomposition of all datasets.

I performed a large series of queries to compare the performance of the different algorithms discussed in Section 5.2. For memory-based collaborative filtering, I used the naive and threshold algorithms. For matrix factorization, the partial threshold algorithm was also used for querying. For each experiment I randomly chose a dataset, the size of the top among the values 1, 5, 10, 50 and 100 and ten queries from the rows of the data matrix. To assess the effect of the size of the database, I randomly witheld 10%, 50% or all the items in the database. Each of the algorithms was applied to find the same top-K relevant targets for all these settings.

The number of scores calculated by the threshold algorithm relative to the number of scores calculated by the naive algorithm for

the different settings is given in Figure 5.1. As guaranteed by the design of the threshold algorithm, in the worst-case scenario the same number of items has to be scored as by the naive algorithm. Clearly, the average gain increases for both settings with database size, as expected. Furthermore, the larger the top-K that is sought, the smaller the gain. The difference between top-1 and top-50 seems to be roughly an order of magnitude for most cases. Comparing the memory-based with the model-based setting, the former shows larger improvements in computational costs, even for small datasets and large top sizes. This is because the threshold algorithm can deal well with sparseness, as discussed in Section 5.2. For the matrix factorization experiments, we see that the relative efficiency decreases when the number of latent features R increases. Large improvements in running time are mainly to be expected when the dimensionality of the data is modest.

5.5.2 Running time and partial threshold algorithm using Uniprot data

In this experiment I study the improvement in querying time and whether the computation time can be reduced by means of the partial threshold algorithm. To this end I use a large multi-label classification dataset related to protein function. I downloaded the complete Uniprot database[5], containing more than half a million of annotated protein sequences. I consider the problem of functionally annotating a protein based on its amino acid sequence. The function of the protein is represented by the gene ontology (GO) label. The GO annotation represents the molecular function, location and biological role of a protein. I ended up with a dataset containing 211,149 proteins with 21,274 distinct labels.

As a feature representation, I used weighted subsequence kernels (Shawe-Taylor and Cristianini, 2004), which define similarity be-

[5]http://www.uniprot.org

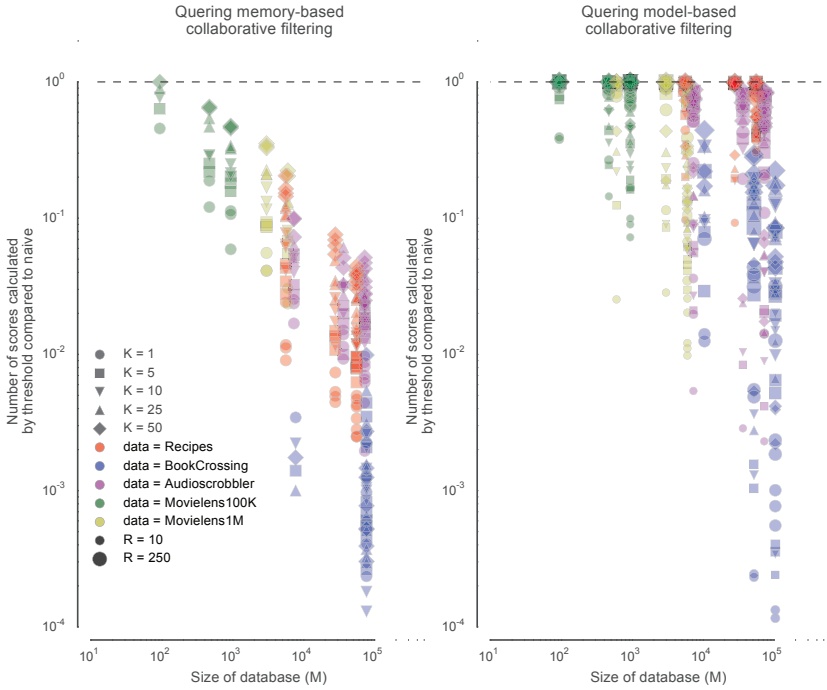


Figure 5.1: Performance of the threshold algorithm relative to the naive algorithm for the collaborative filtering models based on memory and matrix factorization. Marker shape, color and sizes indicates top size, dataset and number of latent features in the case of model-based collaborative filtering.

tween sequences based on subsequences of a fixed length. It is infeasible to compute a complete kernel matrix for hundreds of thousands of instances. I performed an approximation inspired by the Nyström method (Drineas and Mahoney, 2005), a method to approximate the decomposition of large kernel matrices. I arbitrarily selected 500 reference substrings with a length of 100 amino acids from the database, for which I calculated the different subsequence kernels with subsequence lengths from one to twenty with respect to the full protein sequences in the dataset. Thus, if I treat the subsequence length as a hyperparameter, each protein is described by 500 features. In order to find the best model, the models were trained using 80% of the data,

while 20% was withheld as a validation set. All models were evaluated by calculating the AUC over the different functional labels, for each protein separately, and averaging over different proteins. Apart from the subsequence length, each model has its own hyperparameter that had to be optimized:

- **Ridge**: the best combination of substring length and regularisation parameter from the grid $\{0.01, 0.1, 1, 10, 100\}$ was chosen,
- **PLS**: the dimension of the latent subspace was treated as a hyperparameter, and selected from the grid $\{10, 50, 100, 250\}$.

I measured the classification performance using the AUC over the labels, averaged over all proteins in the test set (i.e. instance-wise AUC). For ridge regression, I obtained an AUC of 0.982, while for the PLS model an AUC of 0.980 was attained. I used the naive, threshold and partial threshold algorithm to query for the top 1, 5, 10, 25 or 50 most likely labels on the complete label space or random 10% or 50% subsets. Each datapoint represents the average efficiency of a batch of ten protein queries. The results are represented in Figure 5.2.

In the first graph of Figure 5.2, I plotted the relative improvement in number of scores of the threshold algorithm plotted against the improvement in running time for a series of queries on the Uniprot dataset. It is clear that an improvement in the number of scores translates into a proportional improvement in running time ($R^2 = 0.964$). Similarly as for the collaborative filtering results, larger improvements occur for large databases and small top sizes. It is interesting to note that much larger improvements are recorded for the ridge regression model than for PLS. Most likely, this is because PLS orthogonalizes the variables before mapping to the output space, influencing the distribution of the features.

The second graph of Figure 5.2 compares the computational performance of the partial threshold algorithm described in Section 5.2 with the standard threshold algorithm. Recall that the number of

items the partial threshold algorithm tries to score is always the same as for the threshold algorithm. In most cases, the partial threshold algorithm will only calculate a fraction of a score. To compare with the threshold algorithm, I measure the average fraction of partial scores $p(u)_r q(v)_r$ that are added. Hence, each item that is fully scored by the partial threshold algorithm is given a value of one, while partially scored items receive the value of the proportion calculated in Algorithm 5.3. Here, I show that the partial threshold algorithm always has to calculate less full scores compared to the threshold algorithm. Since this algorithm has some extra overhead compared to the original threshold algorithm (the score is updated in two steps), this algorithm is only expected to improve in time when on average only a relatively small fraction of a score has to be calculated. Only for ridge regression with top size equal to one, we see an improvement noticeably larger than 200%. For these queries, a small improvement in running time is observed compared to the threshold algorithm, while for the other experiments a small increase in running time is detected. Likewise, for the model-based collaborative filtering datasets of Section 5.5.1, a similar pattern was observed. The partial threshold algorithm has to calculate only roughly half a score on average and in practice shows something between a minor increase or decrease in running time.

5.5.3 Behaviour of individual queries

To study the behaviour of the threshold algorithm in more detail, I analyzed the lower bound for 100 queries for the top-5 for the Audioscrobbler dataset (model-based collaborative filtering with 50 latent features) and the Uniprot dataset (using partial least squares). The results are represented in Figure 5.3. These plots demonstrate that even though it may take many iterations for the algorithm to terminate, the correct top is often found swiftly. The top graphs monitor the lower bound (i.e. the worst score in the current top-K set). In most cases this lower bound does not increase dramatically after a

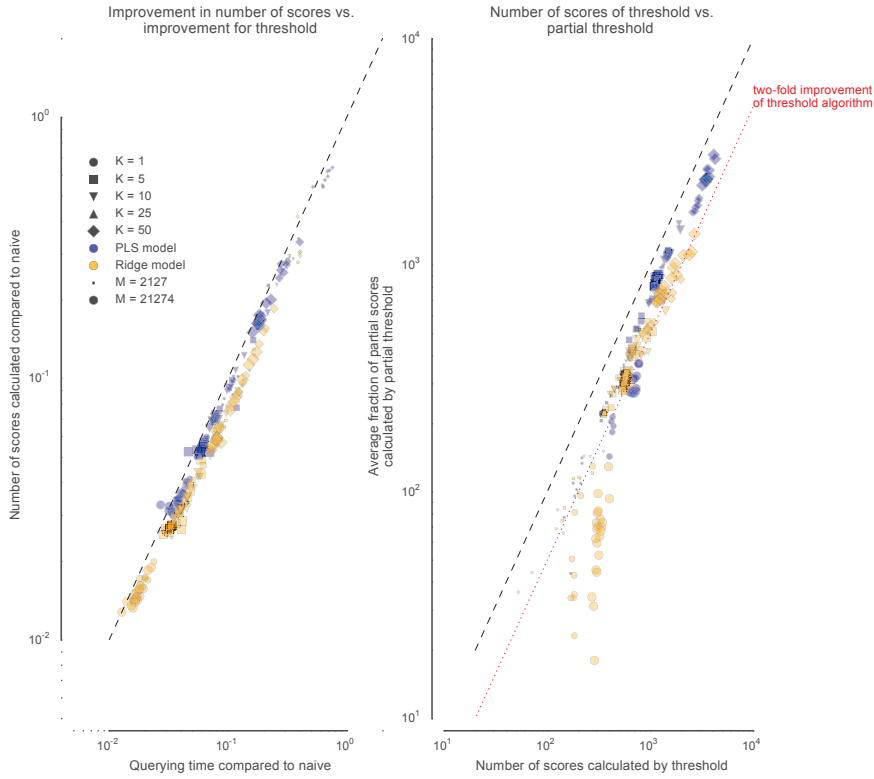


Figure 5.2: (left) Improvement in number of scores of the threshold algorithm plotted against the improvement in running time for a series of queries on the Uniprot dataset. (right) Number of scores that are calculated by the partial threshold algorithm versus the number of scores needed for the threshold algorithm to find the top labels for the Uniprot data.

few hundred iterations. Hence, if the desired top does only need to contain good candidates instead of the correct top-K set, one can decrease computation time by early halting. Furthermore, the lower plots demonstrate that, at least for these cases, the threshold algorithm often finds the correct top much sooner than it terminates. This can be seen in the lower plots of Figure 5.3 where there is a strong lag between the time the correct top is found and the time the algorithm returns this top. We can conclude that it could make sense in some cases to use the halted threshold algorithm (see Fagin et al.

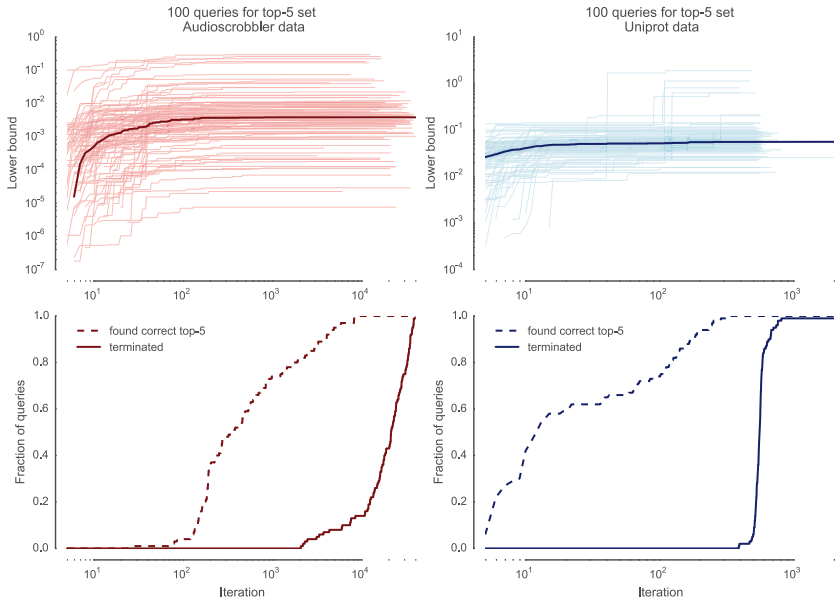


Figure 5.3: Detail of 100 queries for the top-5 for the Audioscrobbler dataset (model-based collaborative filtering with 50 latent features) and the Uniprot dataset (using partial least squares). The top graphs give the values of the lower bound for the individual queries throughout the run of the threshold algorithm. The median of the upper bound is given by the bold line. The bottom graphs show the fraction of queries that have found the correct top at a step in the algorithm and after how many steps the algorithm terminates.

(2003)) which stops when the computational resources have run out (e.g. when a specified maximum querying time has been exceeded).

5.5.4 Large-scale text classification

In this final experiment I demonstrate the scalability of the threshold algorithm on a huge text mining task with the goal of putting Wikipedia articles in 325,056 possible categories. This dataset was downloaded from the Kaggle 'Large Scale Hierarchical Text Classification' challenge[6] (Partalas et al., 2015). The winning submission

[6]https://www.kaggle.com/c/lshtc

to this competition was based on an ensemble of sparse generative models extending Multinomial Naive Bayes. The base-classifiers consisted of hierarchically smoothed models combining document, label, and hierarchy level Multinomials, with feature pre-processing using variants of term frequency-inverse document frequency and BM25.

The training data contained 2,365,436 articles that were labeled with the different categories they belong to. Each article had a feature description in the form of a sparse bag-of-words vector, with a length of 1,617,899. The word counts were transformed into term frequency-inverse document frequencies (Manning et al., 2008). I performed partial least squares regression to map the feature vectors to the output space. This approach is similar to that performed by Mineiro and Karampatziakis (2014) on this dataset. The models had 10, 50, 100, 500 and 1000 latent features. The threshold algorithm was used for the different models to retrieve the highest-scoring class (i.e. K=1). The models were evaluated using average AUC over the labels and precision-at-1 (the percentage of queries that returned the correct class at the top) on a random test set of 10,000 articles. As a baseline, I also included a popular recommender which always returns a score proportional to the probability of the label occurrence, independent of the word features. The results are represented in Table 5.4.

All the models have a better prediction accuracy than the baseline and the performance increases with the number of latent features used. As was the case for the previous experiments, the average number of scores needed to find the top increases sublinearly with R, it seems to scale well (when R is 1000, on average only 2.8% of the classes have to be considered).

5.6 Conclusions and future perspectives

In this chapter, I have discussed three generally applicable methods for finding efficiently the most relevant targets in a wide range of multi-

Table 5.4: Performance of the models for the text classification data. The partial least squares models were calculated with different numbers of latent features R. The performance is calculated for 10,000 test articles of the categories and averaged. A popular recommender is used as a baseline and has an AUC of 0.848 and a precision-at-1 of 0.165. The average number of scores calculated increases sublinear with R. The degree obtained by a least-squares fit (i.e. number of scores $\propto CR^a$) is $0.77 \pm 3.6 \times 10^{-3}$.

R	10	50	100	500	1000
AUC	0.918	0.955	0.962	0.975	0.980
Precision-at-1	0.182	0.191	0.197	0.218	0.237
Average number of scores calculated	28.3	179.4	441.7	3581.3	8995.7

target prediction problems. The experimental results clearly confirm that the threshold algorithm is quite scalable, and often many orders of magnitude more efficient than the naive approach. One can thus conclude that this algorithm is very suitable for querying in large-data settings using many popular machine learning techniques. I have shown experimentally that large improvements can be expected when the size M of the database is large and when only a small top-K set is requested. Furthermore, the number of scores that have to be calculated increases with the dimension R. When querying time is vital, it is thus recommended to consider the use of (supervised) dimension reduction techniques on the features. Empirically, I have also shown that the threshold algorithm can cope well with large sparse feature vectors.

I proposed a small modification of the threshold algorithm. By making use of the lower and upper bound, the calculation of the scores can be terminated early while still retaining an exact method. In the experiments, I provided a proof-of-concept, showing a modest reduction in the number of calculations. In practice, due to the extra computational overhead that is created, this is not always translated into a reduced running time. I am optimistic that this modification can be of use in future applications for two reasons. First, certain hardware implementations might be suited for this algorithm as it is now hard to beat optimized implementations for ordinary matrix multiplication. Second, I believe that our partial threshold algorithm may lead to very efficient inexact querying algorithms. In Section 5.5.3 I demonstrated that the threshold algorithm is often overly conservative to determine when the correct top is found. An interesting future research topic would be to study the trade-off between uncertainty in the top-K set and computational cost. This would likely be rather empirical, as this should be dependent on the distribution of the database and the query.

The framework that I presented works well for SEP-LR models corresponding to Eq. (5.1), which apply to a wide range of statistical models. However, in certain applications, one also encounters pairwise features that jointly describe information of a query and a target. For example in bioinformatics, if one wants to find miRNAs (small nucleic acid sequences with a regulating function) that can interact with messenger RNA, one sometimes considers the number of nucleotide matches between these sequences (e.g. De Paepe et al. (2016)). Similarly, when recommending products to users, one could consider as a feature the number of views a user had on a page where the product is mentioned. Features of that kind cannot be incorporated into SEP-LR models. One would need linear models of the following form:

$$f(u, v) = \sum_{r=1}^{R} z_r(u, v)\,,$$

with $\boldsymbol{z}(u, v) = (z_1(u, v), z_2(u, v), ..., z_R(u, v))^\top$ a pairwise representation of the query and target. This can be thought of as a generalisation of Eq. (5.1). Note that the feature description is never known in advance, because it depends upon the query v. Finding efficient algorithms for dealing with these types of models is also an interesting topic for future research.

6

Functional annotation of enzymes using conditional ranking

Enzyme sequences and structures are routinely used in the biological sciences as queries to search for functionally related enzymes in online databases. To this end, one usually departs from some notion of similarity, comparing two enzymes by looking for correspondences in their sequences, structures or surfaces. For a given query, the search operation results in a ranking of the enzymes in the database, from very similar to dissimilar enzymes, while information about the biological function of annotated database enzymes is ignored.

In this chapter I show that rankings of that kind can be substantially improved by applying the kernel-based learning algorithms discussed in Chapter 4. This approach enables the detection of statistical dependencies between similarities of the active cleft[1] and the biological function of annotated enzymes. This is in contrast to search-based approaches, which do not take annotated training data into account. Similarity measures based on the active cleft are known to outperform sequence-based or structure-based measures under certain conditions. I consider the Enzyme Commission (EC) classification hierarchy for obtaining annotated enzymes during the training phase. The results

[1] The active cleft is the part of an enzyme that directly binds to a substrate and carries the reaction.

of a set of sizeable experiments indicate a consistent and significant improvement for a set of similarity measures that exploit information about small cavities in the surface of enzymes.

6.1 Introduction

Modern high-throughput technologies in molecular biology are generating more and more protein sequences and tertiary structures, of which only a small fraction can ever be experimentally annotated w.r.t. functionality. Predicting the biological function of enzymes[2] remains extremely challenging, and especially novel functions are hard to detect, despite the large number of automated annotation methods that have been introduced in the last decade (Friedberg, 2006; Dunaway-Mariano, 2008).

Existing online services, such as BLAST or ReliBase, often provide tools to search in databases that contain collections of annotated enzymes. These systems rely on some notion of similarity when searching for related enzymes, but the definition of similarity differs from system to system. Indeed, a vast number of measures for expressing similarity between two enzymes exists in the literature, performing calculations on different levels of abstraction. One can make a major subdivision into approaches that solely use the sequence of amino acids, approaches that also take into account the tertiary structure, and approaches that consider local fold information by analyzing small cavities (hypothetical binding sites) at the surface of an enzyme.

Sequence-based measures such as BLAST (Altschul et al., 1990) and PSI-BLAST (Altschul et al., 1997) can be computed in an efficient manner and are able to find enzymes with related functions under certain conditions. In addition to these, several kernel-based methods

[2]Enzymes are biomolecules that catalyze chemical reactions. All the enzymes considered in this chapter are proteins and vice versa. Both notions will be used interchangeably.

have been developed to make predictions for proteins at the sequence level - see e.g. (Leslie et al., 2002, 2004). A high sequence similarity usually results in a high structural similarity, and proteins with a sequence identity (the number of matches in an alignment) above 40% are generally considered to share the same structure (Powers et al., 2006). However, this assumption becomes less reliable in the twilight zone, when the sequence identity is situated between 25 and 40%. Furthermore, enzymes with comparable functions can exhibit sequences with very low sequence identity (Rost, 2002; Chalk et al., 2004).

For these reasons, and because three-dimensional crystal structures are becoming more and more available in online databases, the comparison of proteins at the structural level has gained increasing attention. The secondary structure of an enzyme is known to highly influence its biological function (Kinoshita et al., 2007) and contains valuable information that is missing at the sequence level (Thornton et al., 2000; Kotera et al., 2004; Egelhofer et al., 2010). Many approaches that perform calculations on the overall fold of the protein have been developed - see e.g. Shatsky et al. (2004); Shatsky (2006); Harrison et al. (2003). Unfortunately, such approaches are also not optimal for determining the function of enzymes. They require knowledge of active site residues and usually lead to a quite coarse representation, especially for enzymes, where often only a few specific residues are responsible for the catalytic mechanism (Martin et al., 1998). For example, the vicinal-oxygen-chelate superfamily shows a large functional diversity while having only a limited sequence diversity (Babbitt and Gerlt, 1997; Gerlt and Babbitt, 2001). It has also been shown that some parts of the protein structure space have a high functional diversity (Osadchy and Kolodny, 2011), further limiting the use of global fold similarity. For these reasons, many methods consider local structural features, such as evolutionary conserved residues (Kristensen et al., 2008; Erdin et al., 2011).

The most appropriate similarity measures for the prediction of enzyme functions focus on surface regions in which ligands, co-factors and substrates can bind (Laskowski et al., 1996). Cavities in the surface are known to contain valuable information, and exploiting similarities between those cavities helps finding functionally related enzymes. By considering structural and physico-chemical information of binding sites, one can detect relationships that cannot be found using traditional sequence- and fold-based methods, making such similarities of particular interest for applications in drug discovery (Pérot et al., 2010; Andersson et al., 2010). In addition to providing a complementary notion of protein families (Weisel et al., 2010), these methods also allow for extracting relationships between cavities of unrelated proteins (Weber et al., 2004). Similarity measures that highlight cavities and binding sites can be further subdivided into graph-based approaches such as Huan et al. (1999); Borgwardt et al. (2005); Gärtner (2008); Shervashidze (2009); Vacic et al. (2010), geometric approaches such as (Shatsky et al., 2006; Fober et al., 2011) and feature-based approaches such as (Weill and Rognan, 2010; Fober et al., 2012). These measures will be discussed more thoroughly in Section 6.2.

This chapter aims to show that the search for functionally related enzymes can be substantially improved by applying *learning-to-rank* methods discussed in Section 4.4.3. These algorithms use training data to build a mathematical model for ranking objects, such as enzymes, that are not necessarily seen among the training data. While the methods can be applied to all types of data, as long as a meaningful similarity measure can be constructed, I only demonstrate its power using cavity-based measures, for the reasons explained above. Ranking-based machine learning algorithms are often used for applications in information retrieval (Hüllermeier and Fürnkranz, 2010). Due to their proven added value for search engines, ranking-based machine learning methods have gained some popularity in bioinformatics, for example in drug discovery (Rathke et al., 2010; Agarwal

et al., 2010) or to find similarities between proteins (Weston et al., 2004; Kuang et al., 2005). Despite this, many online services such as BLAST, PDB, Dali and CavBase solely rely on similarity measures to construct rankings, without utilizing annotated enzymes and learning algorithms to steer the search process during a training phase. However, due to the presence of annotated enzymes in online databases, improvements can be made by applying ranking-based machine learning algorithms. This amounts to a transition from an unsupervised to a supervised learning scenario.

Using four different cavity-based similarity measures and one based on sequence alignment as input for RankRLS (Pahikkala et al., 2009), a kernel-based ranking algorithm, I demonstrate a significant improvement for each of these measures. RankRLS works in a similar way as competitors such as RankSVM (Joachims, 2005), because it uses annotated training data to learn rankings during a training phase. The training data is annotated via the Enzyme Commission (EC) functional classification hierarchy, a commonly used way to subdivide enzymes into functional classes. EC numbers adopt a four-number hierarchical structure, representing different levels of catalytic detail. An overview of the enzymatic function encoded by the top digit of the EC number is given in Table 6.1. Importantly, this representation focuses on the chemical reactions that occur, and not on structure or homology. As explained more elaborately in Section 6.2, the EC numbers are used to construct a ground truth catalytic similarity measure, and subsequently to generate ground-truth rankings. In addition to obtaining annotated training data, this procedure also allows for a fair comparison with the more traditional approach, using conventional performance measures for rankings. This way of evaluating also characterizes the difference between our search engine approach and previous work in which supervised learning algorithms for EC number assignment have been considered – for a far from complete list see e.g. Borgwardt et al. (2005); Rousu et al. (2006); Sokolov and

Ben-Hur (2008); Arakaki et al. (2009). In this chapter I am unable to compare to such methods, because they do not return rankings as output. Nonetheless, similar to some of these approaches, I do take the hierarchical structure of the EC numbers into account. Instead of predicting one EC number, a ranking of functionally related enzymes is returned for a given query. In this scheme the top of the obtained ranking is expected to contain enzymes with functions similar to the query enzyme with an unknown EC number. A ranking provides end users with a generally well-known and easily understandable output, while still useful results can be retrieved when an enzyme with a new EC number is encountered.

6.2 Material and methods

6.2.1 Database

This work builds upon CavBase, a database that is made commercially available as part of ReliBase (Hendlich et al., 2003). CavBase is used for the automated detection, extraction, and storage of protein cavities from experimentally determined protein structures, available through the Protein Data Bank (PDB).

The geometrical arrangement of the pocket and its physico-chemical properties are first represented by predefined pseudocenters – spatial points that characterize the geometric center of a functional group specified by a particular property. The type and the spatial position of the pseudocenters depend on the amino acids that border the binding pocket and expose their functional groups. They are derived from the protein structure using a set of predefined rules (Schmitt et al., 2002). Hydrogen-bond donor, acceptor, mixed donor/acceptor, hydrophobic aliphatic, metal ion, pi (accounts for the ability to form π-π interactions) and aromatic properties are considered as possible types of pseudocenters. These pseudocenters can be regarded as

Table 6.1: Overview of the function of enzymes according to the top EC number (Voet and Voet, 2010).

EC class	Enzyme	Description	Typical reaction
EC 1	Oxidoreductases	Catalyses oxidation or reduction reactions, transfers an O or H atom or electrons from one molecule to the other.	$AH + B \longrightarrow A + BH$ (reduction) $A + O \longrightarrow AO$ (oxidation)
EC 2	Transferases	Transfers a functional group from one molecule to another.	$AB + C \longrightarrow A + BC$
EC 3	Hydrolases	Formation of two products from a substrate by hydrolysis.	$AB + H_2O \longrightarrow AOH + BH$
EC 4	Lyases	Non-hydrolytic addition or removal of groups from substrates.	$RCOCOOH \longrightarrow RCOH + CO_2$ or $[X{-}A{-}B{-}Y] \longrightarrow [A{=}B + X{-}Y]$
EC 5	Isomerases	Intramolecule rearrangement, i.e. isomerization changes within a single molecule.	$AABC \longrightarrow BCA$
EC 6	Ligases	Join together two molecules by synthesis of new C−O, C−S, C−N or C−C bonds with simultaneous breakdown of ATP.	$X + Y^{+}\,ATP \longrightarrow XY + ADP + Pi$

a compressed representation of surface areas where certain protein-ligand interactions are encountered. Consequently, a set of pseudo-centers is an approximate representation of a spatial distribution of physico-chemical properties.

To build and test the models we require an appropriate data set that contains sufficiently many proteins and EC classes. Based on the experience of local pharmaceutical experts, we chose the data set of EC classes depicted in Table 6.2. To generate the first data set (data set I), we retrieved all proteins from the PDB which got assigned one of these EC classes. Thus, we ended up with a set of 5,257 proteins. To ensure that only unique proteins were contained in our data set, we used the protein culling server[3] with its default parameterization. As such, all proteins that have high pairwise homology were filtered out. This procedure resulted in a data set of cardinality 1,714. To extract the active site of the protein we used the assumption that the largest binding site of a protein does contain its catalytic center (Laskowski et al., 1996). Hence, for each protein we took the binding site from the database CavBase which maximized the volume. From our data set, 158 proteins were not contained in the CavBase (e.g. because the structure was determined by NMR instead of X-ray). Therefore these proteins were removed from the data set, resulting in a final data set of size 1,556.

The first data set comes with two drawbacks. First of all, the binding site containing the catalytic centre was determined by a pure heuristic, namely by taking the largest binding site among all binding sites a protein exhibits. Moreover, sufficient resolution was not a criterion for selecting the cavities. This may lead to a data set of low quality. Therefore, relying on the expertise of pharmaceutical experts we compiled another data set referred to as data set II, containing the same EC classes. For this data set, all proteins from the PDB that have a resolution of at least 2.5 Å were considered. Moreover

[3] `http://www.bioinf.manchester.ac.uk/leaf/`

Table 6.2: List of the 21 EC numbers with their accepted name and the number of examples of each class for the two data sets.

EC number	accepted name	# set I	# set II
EC 1.1.1.1	alcohol dehydrogenase	23	15
EC 1.1.1.21	aldehyde reductase	35	30
EC 1.5.1.3	dihydrofolate reductase	110	6
EC 1.11.1.5	cytochrome-c peroxidase	92	31
EC 1.14.15.1	camphor 5-monooxygenase	30	36
EC 2.1.1.45	thymidylate synthase	63	22
EC 2.1.1.98	diphthine synthase	5	43
EC 2.4.1.1	phosphorylase	43	40
EC 2.4.2.29	tRNA-guanine transglycosylase	32	16
EC 2.7.11.1	non-specific serine/threonine enzyme kinase	304	24
EC 3.1.1.7	acetylcholinesterase	23	13
EC 3.1.3.48	enzyme-tyrosine-phosphatase	151	28
EC 3.4.21.4	trypsin	118	72
EC 3.4.21.5	thrombin	87	51
EC 3.5.2.6	β-lactamase	153	8
EC 4.1.2.13	fructose-bisphosphate aldolase	48	4
EC 4.2.1.1	carbonate dehydratase	186	76
EC 4.2.1.20	tryptophan synthase	13	7
EC 5.3.1.5	xylose isomerase	18	21
EC 5.3.3.1	steroid Δ-isomerase	14	10
EC 6.3.2.1	pantoate-β-alanine ligase	8	8

the binding site volume was required to range between 350 Å^3 and 3500 Å^3. Structures not meeting these conditions were eliminated since resolutions below 2.5 Å usually lead to a too coarse representation, while binding sites with volumes outside the above-mentioned range are usually artefacts produced by the algorithm used for their detection. From the resulting set of 24,102 proteins the active site was selected. This resulted in a data set of 1730 enzymes on which we applied the protein culling server (Wang and Dunbrack, 2003) to finally end up with a second data set of 561 enzymes.

6.2.2 Similarity measures for cavities

In the introduction I have motivated why our analysis is restricted to similarity measures for cavities, which are three-dimensional objects that can be represented in multiple ways. Some measures are graph-

based, transforming cavities into node-labeled and edge-weighted graphs. This allows to apply traditional techniques to compare graphs (e.g. Huan et al. (1999)). Unfortunately, techniques that construct such similarity measures, such as those based on graph isomorphisms, are not appropriate for comparing noisy and flexible protein structures. Computing the maximum common subgraph (Bunke and Shearer, 1998) can be considered as a more appropriate alternative, and this method will be used in this paper as a baseline (see below). The graph edit distance (Sanfeliu and Fu, 1983) is another measure to compare graphs, specifying the number of edit operations needed to transform a given graph into another graph. This distance can be calculated in different ways, e.g. by using a greedy heuristic (Weskamp et al., 2007) or quadratic programming (Neuhaus and Bunke, 2007). Unfortunately, the graph edit distance is very hard to parameterize and often quite inefficient. More efficient approaches belong to the class of graph kernels. They have gained a lot of attention in bioinformatics, as they allow for a sufficiently high degree of error tolerance. Different realizations are available, such as the shortest path kernel (Borgwardt et al., 2005), the random walk kernel (Gärtner, 2008) and the graphlet kernel (Shervashidze, 2009; Vacic et al., 2010). Graph kernels work particularly well for small molecules such as ligands, but they are less useful for larger molecules such as proteins. They gave rather poor results in Fober et al. (2012), which explains why I concentrated here on the maximum common subgraph as a representative for graph-based approaches.

As a second category of measures for cavities, geometric methods directly process the labeled spatial coordinates of the functional parts, denoted as point clouds, instead of transforming a protein cavity into a graph. Remarkably, only few approaches have been proposed that build on this representation. In Shatsky et al. (2006) geometric hashing is employed to calculate a superposition of protein cavities that can be used to derive an alignment and a similarity score. A similar

approach was used in Fober et al. (2011), in which an optimization problem was solved instead of applying geometric hashing. Beside these two approaches, several other methods exist for comparing two point clouds (Alt and Guibas, 1996). Unfortunately, the majority of these methods cannot cope with biological data, due to a very high complexity or error intolerance.

As a third family of approaches, one can also represent the protein cavity as a feature vector, taking both the geometry of the cavity and physico-chemical properties into account – see e.g. Fober et al. (2012); Weill and Rognan (2010). Subsequently, traditional or specialized measures can be applied to these vectors to obtain similarity scores between protein cavities (Mahé et al., 2005; Deza and Deza, 2009).

In the experiments I selected a representative method for each of the three groups: one graph-based measure, one geometric measure and one feature-based measure. I also considered the original CavBase measure and a measure obtained from the Smith-Waterman protein sequence alignment. This led to a comparison of five different measures, four based on cavities and one based on sequence alignment. Below, these measures are explained more in detail:

- **Labeled Point Cloud Superposition (LPCS)** (Fober et al., 2011). This value is obtained by processing labeled point clouds. Hence, the CavBase data can be used directly without a need for transforming it into another representation. Intuitively, two labeled point clouds are considered similar if they can be spatially superimposed. More specifically, an approximate superposition of the two structures is obtained by fixing the first point cloud and moving the second point cloud as a whole. Two point clouds are well superimposed when each point in the first cloud can be matched with a point in the second point cloud, while the distances of these points are small and their labels consistent. This concept is used to define a fitness function that is maximized using a direct search approach (Beyer and Schwefel, 2002). The

obtained maximal fitness is taken as the similarity between the two labeled point clouds. A similar measure was also proposed in Hoffmann et al. (2010), but a convolution kernel was suggested to obtain similarities between the point clouds.

- **Maximum Common Subgraph (MCS)** (Bunke and Shearer, 1998). Using the MCS, the original representation in the form of a labeled point cloud must be transformed into a node-labeled and edge-weighted graph. Each pseudocenter is becoming a node labeled with the corresponding physico-chemical property. To capture the geometry, a complete graph is considered, where each edge is weighted with the Euclidean distance between the two pseudocenters it is adjacent to. The problem of measuring similarity between protein cavities now boils down to the problem of measuring similarity between graphs. A well-known approach is to search for the maximum common subgraph of the two input graphs and to define similarity as the size of the maximum common subgraph relative to the size of the largest graph. In case of noisy data, a threshold ϵ is required, defining two edges to be equal if their weight differs at most by ϵ. In this paper, this parameter is set to 0.2 Å, as recommended by several authors (Weskamp et al., 2007; Fober et al., 2009a).

- **CavBase (CB)** (Schmitt et al., 2002). CavBase also makes use of an algorithm for the detection of common subgraphs. Instead of considering the largest common subgraph, as done in the case of MCS, the 100 largest common subgraphs are considered. Each common subgraph is used to determine a transformation rule by means of the Kabsch algorithm (Kabsch, 1976), which superimposes both proteins. In a post-processing step the surface points are also superimposed according to the transformation rule, and a similarity score is derived using these surface points. Eventually, a set of 100 similarity values is obtained,

from which the highest value is returned as similarity between the two protein cavities.

- **Fingerprints (FP)** Fingerprints are a well-known concept and have been used successfully in many domains. For the comparison of protein binding sites, the authors in (Fober et al., 2009b) transformed the protein binding site into a node-labeled and edge-weighted graph as described above. Moreover they defined generically a set of features, namely complete node-labeled and edge-weighted graphs of size three. For each such feature, a test is performed to decide whether or not the feature is contained in the graph representing the protein. This is done by subgraph isomorphism, to check whether the labels are identical. The nodes of the features are labeled by the set of physicochemical properties. Edges of the patterns are labeled by intervals or bins and instead of testing for equivalence. A test is performed whether edge weight of the graph representing the protein falls into the bin of the pattern. The thus generated fingerprints are compared by means of the Jaccard similarity measure, as proposed by Mahé et al. (2005).

- **Smith-Waterman (SW)** Beside using approaches based on structure to compare protein binding sites, I used also sequence alignment in our experimental study. To calculate sequence alignments I used the Smith-Waterman algorithm (Smith and Waterman, 1981) which was parameterized with the Blosum-62 matrix. From the sequence alignment I derived the sequence identity which was subsequently used to perform experiments.

6.2.3 Unsupervised ranking

In the introduction I have explained why existing online services such as BLAST, PDB, Dali and CavBase construct rankings in an unsupervised way. These systems create a ranking by means of a similarity

measure only, without training a model that uses annotated enzymes. The annotated enzymes in a database are simply ranked according to their similarity with an enzyme query with unknown function. In the case of CavBase, the enzymes having a high cavity-based similarity appear on top of the ranking, those exhibiting a low cavity-based similarity end up at the bottom. More formally, let us represent the similarity between a pair of enzymes by $k : \mathcal{U}^2 \to \mathbb{R}^+$, where $\mathcal{U}$ represents the set of all potential enzymes. Given the similarities $k(u, u')$ and $k(u, u'')$, we compose the ranking of u' and u'' conditioned on the query u as:

$$u' \succeq_u^k u'' \Leftrightarrow k(u, u') \geq k(u, u'') \,, \tag{6.1}$$

where $\succeq_u^k$ indicates the relation *is ranked higher than, for query u, based on the measure k*, as discussed in Section 2.3. Note that this is a relation between two enzymes *conditioned* on a third enzyme. In our context, there is no meaningful ranking possible between enzymes u' and u'' without referring to another enzyme u. This approach adopts the same methodology as a nearest neighbor classifier, but a ranking rather than a class label should be seen as the output of the algorithm.

The quality of such rankings can be evaluated when the database contains annotated enzymes and annotated queries. In an evaluation phase, I compare the obtained ranking with the *ground truth* ranking, which can be constructed from the EC numbers for annotated enzymes. This ground truth ranking can be deduced from the catalytic similarity (i.e. ground truth similarity) between the query and all database enzymes, by counting the number of successive matches in the EC label of the query and the database enzymes. Thus the catalytic similarity is a property of only a pair of enzymes. In contrast, in order to create the ground truth ranking of two enzymes, the catalytic similarity has to be calculated w.r.t. a third enzyme. For example, an enzyme with EC number EC 2.4.2.23 has a catalytic similarity of two compared to an enzyme labeled as EC 2.4.99.12, since both enzymes belong to the family of glycosyltransferases. Conversely, the

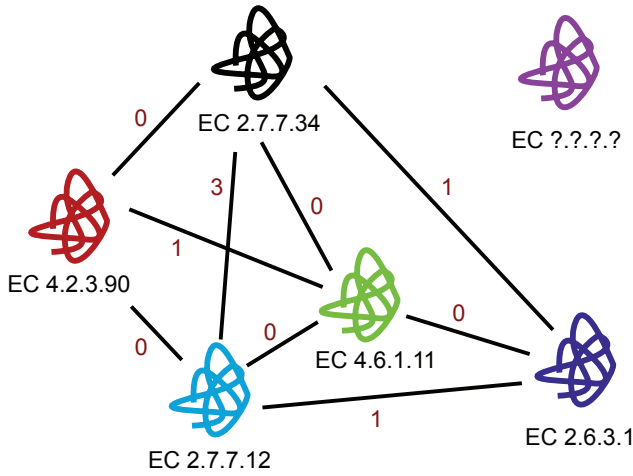


Figure 6.1: Six enzyme structures are shown, five of which correspond to a known EC number. The catalytic similarity Q is depicted on the edges of the graph. The algorithm that I present allows us to infer for the unannotated query (denoted as EC ?.?.?.?) a ranking of the annotated enzymes. To this end, the unsupervised approach solely uses cavity-based similarity measures, whereas the supervised approach also takes the EC numbers of annotated enzymes into account.

same enzyme manifests a similarity value of only one with an enzyme labeled as EC 2.8.2.23. Both are transferases in this case, but they show no further relevant similarity in the chemistry of the reactions to be catalyzed.

More formally, let us represent the catalytic similarity between two enzymes by a relation $Q : \mathcal{U}^2 \to \{0, 1, 2, 3, 4\}$. Q is defined by:

$$Q(u, u') = \sum_{i=1}^{4} \prod_{j=1}^{i} q_i q_j \,,$$

where q_i equals 1 if the i-th digit of the EC numbers of u and u' are the same and 0 otherwise. Figure 6.1 gives an example for six enzymes, five of which correspond to a known EC number. The catalytic similarity Q is depicted on the edges of the graph. The proposed algorithm allows us to infer, for an unannotated query, a ranking of

the annotated enzymes, some of which the algorithm may not have encountered among the training data.

Given the similarity values $Q(u, u')$ and $Q(u, u'')$, we compose similar to Eq. (6.1) the ground truth ranking of u' and u'' conditioned on the query u as:

$$u' \succeq_u^Q u'' \Leftrightarrow Q(u, u') \geq Q(u, u'') \,.$$

As a result, an entire ground truth ranking of database enzymes with known EC numbers can be constructed, given an annotated query enzyme.

6.2.4 Supervised ranking

In contrast to unsupervised ranking approaches, supervised algorithms do take ground truth information into account during a training phase. I perform experiments with so-called conditional ranking algorithms of Section 4.4.3 using the RankRLS implementation (Pahikkala et al., 2009). Let us introduce the short-hand notation $e = (u, u')$ to denote a couple consisting of an enzyme query u and a database enzyme u'. RankRLS produces a linear basis function model of the type:

$$f(e) = f(u, u') = \langle \mathbf{w}, \Phi(e) \rangle \,, \tag{6.2}$$

in which $\mathbf{w}$ denotes a vector of parameters and $\Phi(e)$ an implicit feature representation for the couple $e = (v, v')$.

RankRLS differs from more conventional kernel-based methods because it optimizes a convex and differentiable approximation of the rank loss in bipartite ranking (i.e. area under the ROC curve) instead of the zero-one loss. Together with the standard L2 regularization term on the parameter vector $\mathbf{w}$ and a regularization parameter λ,

the following loss is minimized:

$$\mathcal{L}(f, S) = \sum_{u \in \mathcal{U}} \sum_{e, \bar{e} \in E_v} (Q_e - Q_{\bar{e}} - f(e) + f(\bar{e}))^2 \,, \tag{6.3}$$

for a given training set $S = \{(e, Q_e) \mid e \in E\}$. Here $Q_e = Q(v, v')$ denotes the ground truth similarity as defined above, E the set of pairs of training queries and objects for which ground truth information is available and E_v the subset of E containing results for the query v. The outer sum in Eq. (6.3) takes all queries into account, and the inner sum analyzes all pairwise differences between the ranked results for a given query. This loss can be minimized in a computationally efficient manner, using analytic shortcuts and gradient-based methods, as discussed in Section 4.4.3.

According to the representer theorem (Schölkopf and Smola, 2002), one can rewrite Eq. (6.2) in the following dual form:

$$f(e) = \langle \mathbf{w}, \Phi(e) \rangle = \sum_{\bar{e} \in E} a_{\bar{e}} \Gamma(e, \bar{e}) \,.$$

with $\Gamma(e, \bar{e})$ a kernel function with four enzymes as input and $a_{\bar{e}}$ the weights in the dual space. I again adopt the Kronecker product feature mapping, containing information on couples of enzymes:

$$\Phi(e) = \Phi(u, u') = \phi(u) \otimes \phi(u') \,,$$

with $\phi(v)$ a feature mapping of an individual enzyme. One can easily show that this pairwise feature mapping yields the Kronecker product pairwise kernel in the dual representation:

$$\begin{aligned} \Gamma(e, \bar{e}) &= \langle \phi(u) \otimes \phi(u'), \phi(\bar{u}) \otimes \phi(\bar{u}') \rangle \\ &= \langle \phi(u), \phi(\bar{u}) \rangle \langle \phi(u'), \phi(\bar{u}') \rangle \\ &= \Gamma(u, u', \bar{u}, \bar{u}') = k(u, \bar{u}) k(u', \bar{u}') \,, \end{aligned}$$

with $k(\cdot,\cdot)$ a traditional kernel for enzymes. Specifying a universal kernel for k leads to a universal kernel for Γ (Waegeman et al., 2012), indicating that one can use the kernel to represent any arbitrary relation, provided that the learning algorithm has access to training data of sufficient quality. This kernel has been introduced in (Ben-Hur and Noble, 2005) for modelling protein-protein interactions. I consider this kernel because of its universal approximation property, but also other pairwise kernels exist, such as the cartesian pairwise kernel (Kashima et al., 2010), the metric learning pairwise kernel (Vert et al., 2007) and the transitive pairwise kernel (Herbrich et al., 2000; Pahikkala et al., 2010). Nonetheless, it is probably not very surprising that such kernels only yield an improvement if the concepts to be learned satisfy the restrictions that are imposed by the kernels (Waegeman et al., 2012).

With the exception of the FP measure, none of the similarity measures discussed in Section 6.2.2 are strictly speaking valid kernels. Using the above construction, all similarity measures can be converted into kernels of type K^{ϕ}, when they are made symmetric and positive definite. These attributes guarantee a numerically stable and unique solution of the learning algorithm. I simply enforced symmetry by averaging the similarity matrix with its transpose. Subsequently, I made the different similarity matrices positive definite by performing an eigenvalue decomposition and setting all eigenvalues smaller than 10^{-10} equal to zero. This method leads to a negligible loss of information compared to the numerical accuracy of our algorithms and data storage. Finally, each kernel matrix was normalized so that all diagonal elements have a value equal to one. Since these procedures were performed on the whole data set, one arrives at a so-called transductive learning setting (Chapelle et al., 2006). Minor adjustments result in a more traditional inductive learning setting. Note that overfitting is prevented when applying this procedure, since the EC numbers of the enzymes in the data set are not taken into account.

Since the catalytic similarity is a symmetric measure I also perform a post-processing to the output of our algorithm. The matrix with the predicted values used for ranking the enzymes is made symmetric by averaging it with its transpose.

6.2.5 Performance measures for ranking

The ranking obtained with unsupervised or supervised learning algorithms can be compared to the ground truth ranking by applying performance measures that are commonly used in information retrieval.

First of all, the C-index is considered, as discussed in Section 2.3.3. Our interest in this metric is motivated by two reasons. Firstly, unlike most other performance measures I consider, all levels of the EC hierarchy are taken into account to determine the performance of different algorithms. Predicted rankings can be interpreted as layered or multipartite rankings – see e.g. Waegeman et al. (2008); Fürnkranz et al. (2009). The ranking accuracy preserves this hierarchical structure by counting all pairwise comparisons. The second reason is the fact that the ranking accuracy is optimized by the RankRLS software, using the convex and differentiable approximation given in Eq. (6.3). This loss function characterizes the most important difference with more traditional kernel-based algorithms, such as support vector machines, resulting in an information retrieval setting instead of a more traditional classification or network inference setting.

Since the C-index is not generally known in bioinformatics, I also evaluated the algorithms using three more conventional performance measures that are commonly considered for bipartite rankings (i.e. rankings containing relevant versus irrelevant objects). These three measures are the AUC, mean average precision (MAP) and normalized discounted cumulative gain (nDCG) (Järvelin and Kekäläinen, 2002). For AUC and MAP all ground truth rankings had to be converted into bipartite rankings, leading to a decrease in granularity for performance estimation. I chose a cut-off threshold of three in ground

truth similarity: a retrieved enzyme is relevant to the enzyme query if at least the first three components of its EC number are identical to the query.

6.2.6 Experimental set-up

I selected two data sets of enzymes from CavBase, as described in Section 6.2.1. The ground truth catalytic similarity of all enzyme pairs was computed for each data set. Each data set was further randomized and split into four cross-validation folds of equal size. In the unsupervised case each subset was used individually to allow for a comparison with the supervised model. Of such a subset, each enzyme was used as a query to rank the remaining enzymes, as described in Section 6.2.3. The performance for each of these rankings was averaged to obtain the global performance over the folds.

In the supervised setting, each fold was withheld as a test set, while the other three folds of the data set were used for training and model selection. This process was repeated for each fold, so that every enzyme was used for training and testing (thus, four-fold outer cross-validation). Neither the query nor the database enzymes are thus used for building a model, which allows us to demonstrate that our methods can generalize to new enzymes. This corresponds to Setting D, the hardest setting. In addition, a 10-fold inner cross-validation loop was implemented for estimating the optimal regularization parameter λ, as recommended in Varma and Simon (2006). The value of this hyperparameter, which controls model complexity, was selected from a grid containing all powers of 10 from 10^{-4} to 10^{5}. The final model was trained using the whole training set and the median of the best hyperparameter values over the ten folds. I used the implementation RLScore[4] in Python to train the models. The data set and a modified version of the software can be found at `http://users.ugent.be/~mstock/Datasets/` .

[4]See `https://github.com/aatapa/RLScore` for this software.

6.3 Results and discussion

6.3.1 Differences between cavity-based similarities and data sets

Table 6.3 gives a global summary of the results obtained for the unsupervised and supervised ranking approaches for both data sets. One can note a sizeable difference between the performances for the different cavity-based similarities, data sets and performance measures used. Despite this variation, it is clear that data set I is considerably harder than data set II. This can easily be explained by the fact that data set II only contains enzymes with a certain resolution of the active site. Furthermore, for set II the active site is determined by an expert, while for set I the active site is resolved by heuristically choosing the largest cavity. It is likely that some mistakes are made in this annotation process. Consequently, inferring functional similarity of data set I will be harder.

The cavity-based similarity measure based on fingerprints usually results in the worst performance, except for the ranking error in the unsupervised setting. It seems that the performance of the FP does not improve as much in the supervised approach, compared to other cavity-based similarity measures. This is likely because the fingerprints cause a high loss of information, since even functionally dissimilar enzyme cavities can be considered similar according to this metric.

Comparing the two graph-based similarities (MCS and CB), we see some differences between the data sets. Though both perform relatively well, MCS performs better for data set I, while CB is the clear champion of data set II. The good performance of CavBase for data set II can be explained easily. CavBase computes the 100 largest common subgraphs, which could be used to construct a cavity-based similarity measure. However, a graph representation leads to a loss

Table 6.3: Summary of the results obtained for unsupervised and supervised ranking for both data sets. For each combination of similarity and performance measure, the performance is averaged over the different folds and queries, with the standard deviation between parentheses. CB=CavBase, FP=fingerprints, MCS=Maximum Common Subgraph, LPCS=Labeled Point Cloud Superposition, SW=Smith-Waterman.

		Set I				
		CB	FP	MCS	LPCS	SW
Unsupervised	C-index	0.6096 (0.1286)	0.6411 (0.1670)	0.5960 (0.1010)	0.6173 (0.1244)	0.6576 (0.1446)
	MAP	0.4883 (0.2912)	0.4478 (0.2692)	0.4967 (0.2659)	0.4826 (0.2572)	0.5201 (0.2964)
	AUC	0.7110 (0.1845)	0.7221 (0.1961)	0.6967 (0.1728)	0.7183 (0.1565)	0.7397 (0.1980)
	nDCG	0.7161 (0.3107)	0.6615 (0.2709)	0.7686 (0.3001)	0.7364 (0.276)	0.7401 (0.2597)
Supervised	C-index	0.7717 (0.1960)	0.7175 (0.2098)	0.7988 (0.1906)	0.7741 (0.1945)	0.7899 (0.1789)
	MAP	0.5659 (0.3384)	0.5966 (0.3219)	0.6854 (0.3001)	0.6324 (0.3064)	0.6963 (0.3103)
	AUC	0.8242 (0.1857)	0.8003 (0.2225)	0.8585 (0.1802)	0.8306 (0.1954)	0.8813 (0.1591)
	nDCG	0.6550 (0.4195)	0.6928 (0.3984)	0.7717 (0.3635)	0.7324 (0.3724)	0.7518 (0.3669)
		Set II				
		CB	FP	MCS	LPCS	SW
Unsupervised	C-index	0.7216 (0.1911)	0.7212 (0.1546)	0.8070 (0.1736)	0.7515 (0.1647)	0.7856 (0.1591)
	MAP	0.9156 (0.1662)	0.7156 (0.2478)	0.9094 (0.1514)	0.8303 (0.1768)	0.9167 (0.1634)
	AUC	0.9415 (0.1209)	0.8714 (0.1447)	0.9622 (0.0847)	0.8937 (0.1165)	0.9523 (0.1146)
	nDCG	0.9599 (0.0708)	0.8189 (0.2003)	0.9330 (0.0948)	0.8986 (0.1146)	0.9589 (0.0768)
Supervised	C-index	0.9997 (0.0011)	0.931 (0.1096)	0.9944 (0.0253)	0.9883 (0.0583)	0.9985 (0.0076)
	MAP	0.9995 (0.0037)	0.905 (0.2397)	0.9980 (0.0265)	0.9663 (0.1172)	0.9980 (0.0393)
	AUC	0.9999 (0.0005)	0.9844 (0.0502)	0.9997 (0.0036)	0.9926 (0.0354)	0.9998 (0.0042)
	nDCG	0.9983 (0.0185)	0.8887 (0.2243)	0.9744 (0.0773)	0.9467 (0.1183)	0.9968 (0.0344)

of information, since the coordinates of pseudocenters cannot be restored. Moreover, since the size of the (maximum) common subgraph is an integer that usually lies in the range of 4 to 12 nodes, there is a loss of resolution by mapping many different pairs of cavities to the same similarity score. In theory, MCS suffers from these drawbacks. Even though the resolution problem is to a certain extent solved if the size of the maximum common subgraph is divided by the size of the larger binding site, the graph representation could still lead to a slight loss of information.

On the other hand, the LPCS measure uses geometric information, hence, no loss of information is introduced by transforming the pseudocenter representation into a graph representation. Moreover, this transformation does not cause the resolution problem. Yet, the measure is computed via solving a multimodal optimization problem, so it is possible to get stuck in a local optimum, resulting in a similarity score that is too low. Similar to MCS, LPCS seems to perform relatively better for data set than for data set II, probably because the local optimum becomes less of an issue in the former case. This can be explained by the fact that data set I contains larger cavities, on average, hence making it harder to find the global optimum.

Finally, I consider the measure based on sequence alignment. For data set I, the SW similarity measure competes with the MCS as one of the best measures, depending on the performance measure. For data set II in the supervised case, it is only outperformed by the CB measure. It is clear that the SW measure is a powerful method for comparing cavities. It is also limited by a bad resolution of the cleft. Like MCS, SW seeks to quantify the largest similar region, here as a local alignment. As this contains, information about the common residues of the cavity, this is a simple, though powerful measure.

6.3.2 The benefits of supervised ranking

The ranking of both data set I and data set II showed a considerable improvement in the supervised approach. Three important reasons can be put forward to explain the improvement in performance. In this section I will illustrate this using data set II, as this set showed the most clear effects of learning. First of all, the traditional benefit of supervised learning plays an important role. One can expect that supervised ranking methods outperform unsupervised methods, because they take annotations into account during the training phase to guide the model towards retrieval of enzymes with a similar EC number. Conversely, unsupervised methods solely rely on a meaningful similarity measure between enzymes, while ignoring EC numbers.

Second, I also advocate that supervised ranking methods have the ability to preserve the hierarchical structure of EC numbers in their predicted rankings. Figure 6.2 supports this claim. It summarizes the values used for ranking one fold of the test set obtained by the different models. A higher value (indicated by a lighter color) in a row means that this enzyme is considered to have a higher catalytic similarity w.r.t. the query enzyme. So, for unsupervised ranking it visualizes $\Gamma(u, u')$, for supervised ranking the values $f(u, u')$ are shown. Each row of the heat map corresponds to one query. For the supervised models one notices a much better correspondence with the ground truth. Furthermore, the different levels of catalytic similarity can be better distinguished. In addition, an example of the distributions of the predicted values within one query is visualized in Figure 6.4 by means of box plots. The different populations within a plot correspond to the different levels of the catalytic similarity w.r.t. the query enzyme. This illustrates again that supervised models can make a better discrimination between enzymes that are functionally more similar and those that are dissimilar. For this example query, no quartiles are overlapping in any supervised model, unlike the unsuper-

vised approach, which only detects a good ranking for exact matches (i.e. enzymes having an EC number identical to the query).

A third reason for improvement by the supervised ranking method can be found in the exploitation of dependencies between different catalytic similarity values. Roughly speaking, if one is interested in the catalytic similarity between enzymes u and u', one can try to compute this catalytic similarity in a direct way based on mutual relationships in cavities, or derive it in an indirect way from the cavity-based similarity with a third enzyme u''. This division into a direct and an indirect approach shows a certain correspondence to similar discussions in the context of inferring protein-protein interaction and signal transduction networks – see e.g. Albert et al. (2007); Vert et al. (2007); Geurts et al. (2007). In some sense, unsupervised ranking boils down to a direct approach, while supervised ranking should be interpreted as indirect. Especially when the similarity matrix contains noisy values, one can expect that the indirect approach allows for detecting *back bone* entries and correcting the noisy ones.

6.3.3 Differences between performance measures

Table 6.3 indicates that the different performances are to some degree influenced by the similarity measure and data set used. This is especially clear for the supervised ranking approach. One can observe a clear distinction between the ranking accuracy and area under the ROC curve, which treat every position as equally important, and the other two measures, which concentrate on the top of the ranking. This should not come as a surprise, as an approximation of the ranking error is optimized in our algorithms. The AUC is related to the C-index, as they coincide for bipartite rankings. In the latter we only make a distinction between 'relevant' enzymes, which have three or more EC numbers in common, and enzymes which do not. Since

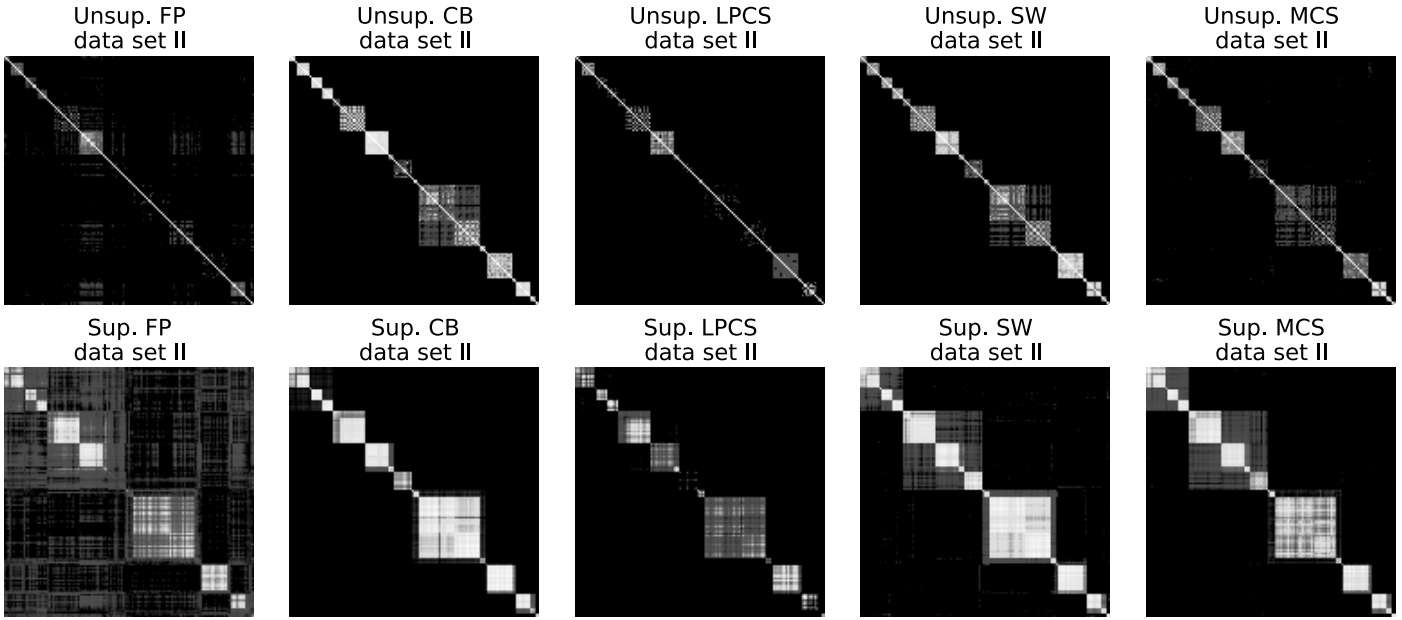


Figure 6.2: Heat maps of the values used for ranking data set II for one fold in the testing phase. Each row of the heat map corresponds to one query. The four figures on top visualize the cavity-based similarities $\Gamma(u, u')$ that are used to construct an unsupervised ranking. The four figures at the bottom visualize the model output $f(u, u')$, which is used to derive a supervised ranking. Lighter colors indicate higher values.

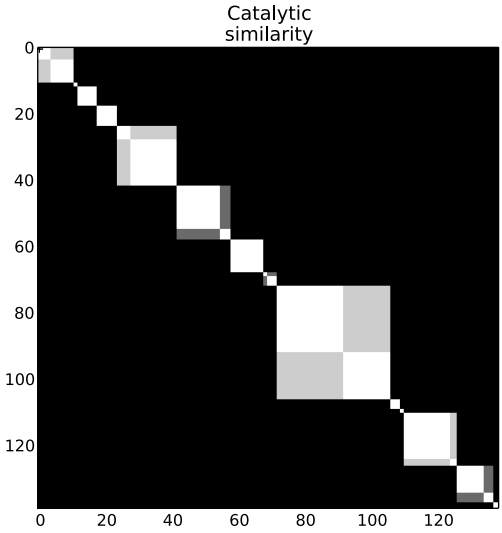


Figure 6.3: The ground truth catalytic similarity that had to be learned. Lighter colors indicate higher values.

the C-index uses a finer fragmentation of functional similarity, this is a more severe performance measure compared to the AUC. For data set II, AUC and C-index are both very close to the theoretical optimum for nearly all cavity-based similarity measures in the supervised case.

Figure 6.5 shows the ROC curves that are obtained by applying the cut-off threshold for data set I, which defines a database enzyme a hit if at least the first three digits of the EC number are correct. In contrast to the scalar performance measures of Table 6.3, the ROC curve gives information about the quality of the ranking at all positions. It is immediately clear that supervised ranking outcompetes unsupervised ranking, as the former's curves are closer to the upper left corner. Typically for these curves, at the left part, the line has a very high slope, showing that a certain fraction of relevant enzymes can be detected with very high sensitivity and specificity. This fraction that can be detected nearly without mistakes increases after the supervised learning step, as indicated by the higher 'offset' of these

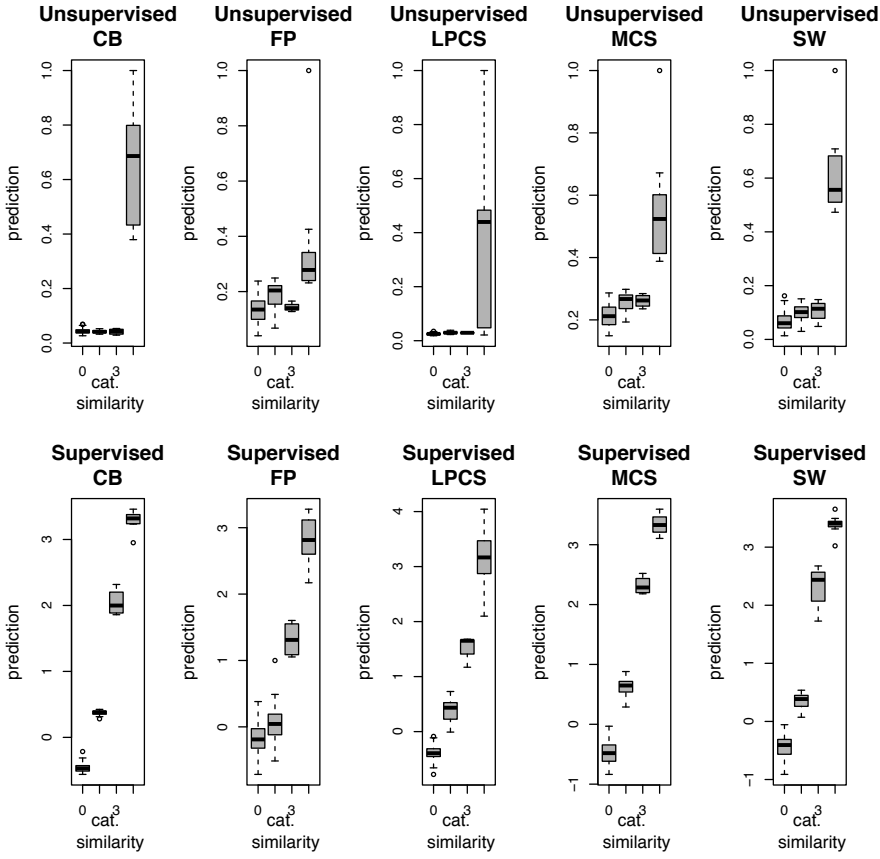


Figure 6.4: Box-and-whisker plots of the values used for ranking data set II for one randomly chosen query as an example. The different populations on the x-axis denote the groups that are formed by subdividing the database enzymes according to the number of EC number digits they share with the query. Given a query u and a database enzyme u', the y-axis shows the distribution of the values $f(u, u')$ and $k(u, u')$ for the supervised and the unsupervised approach, respectively. For nearly all cavity-based similarity measures, one can observe a much better separation of the groups for the supervised approach.

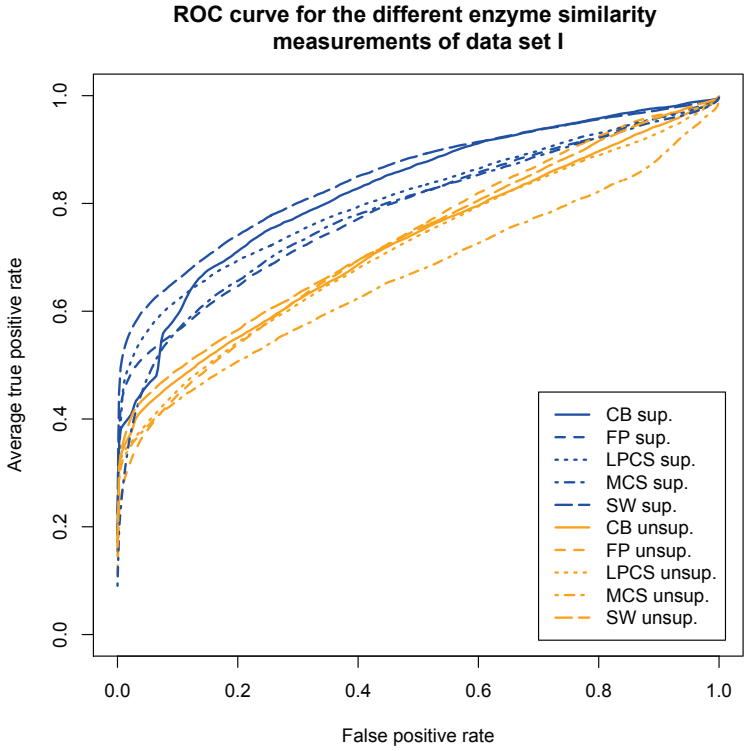


Figure 6.5: Receiver operating characteristic curves for the unsupervised and supervised ranking methods for data set I. An enzyme is considered functionally similar to the query if the first three digits of the EC number are identical to those of the query.

curves (most clear for SW, LPCS and FP in the ROC curve). The next section of the curve is usually a part with a lower average slope, indicating that at some point it becomes harder to detect relevant enzymes. For the unsupervised curves, this is nearly a straight line, which means that from that point the detection of catalytically similar enzymes is essentially random. The supervised curves still have a concave shape at their second part, which shows that relevant enzymes can still be detected in that part.

From Table 6.3 it is also clear that supervised ranking usually scores worse for MAP and nDCG compared to C-index and AUC. For

the nDCG the performance sometimes even decreases after learning a model! This can easily be explained by the fact that our model optimizes the quality of the complete ranking, in contrast to only the top which is assessed by MAP and nDGC. Note that the most functionally similar enzymes (i.e. the same EC number) can likely be detected based on the cavity-based similarity alone. Hence, training a model might not be required to perform well in this section. One can see this learning effect nicely in Figure 6.5 for the CB similarity measure. The quality of the supervised ranking of the top for this measure is worse than for all other measures (indicated by the low nDCG). The overall ranking (indicated by the AUC) is quite good in comparison, as the lower part compensates for the bad ranking at the top. Depending on the application, the top or the general ranking might be more of interest.

6.4 Related work

Since the comparison of enzymes has become an important task in functional bioinformatics, a vast number of similarity measures for proteins has been proposed so far. As mentioned in the introduction, a reliable method will focus on the geometry and the physico-chemical properties of certain regions of an enzyme. However, methods that are based on the sequence or the fold usually exhibit a much lower complexity and they can also lead to good results, especially when the sequence identity is above a certain threshold. *ProFunc* (Laskowski et al., 2005) is in this regard a very interesting tool, in which a bulk of different methods is applied, such as sequence alignment, motif and template search, and also a comparison of active sites. The biological function of enzymes is derived from the closest match in different databases such as PDB, UniProt and PROCAT, and finally returned by the program. Despite being very powerful, this approach becomes nevertheless very inefficient, with runtimes up to several hours for a

single protein. Since I considered a sizeable data set for which nearly 3,000,000 pairwise similarity scores had to be computed, it became impossible to compare our results to ProFunc.

In addition to focusing on individual enzymes, one can also take protein-protein interactions into account for inferring the function. Proteins that are close to each other in a protein-protein interaction network are expected to have similar functions, so one can try to infer the function of an unannotated protein by looking at its neighbors (Schwikowski et al., 2000). Similarly, one can also solve optimization problems over the global network, such as maximizing the number of edges that connect proteins sharing the same function (Vazquez et al., 2003). Other approaches make use of probabilistic graphical models such as Markov random fields (Deng et al., 2003; Letovsky and Kasif, 2003). Conceptually, these methods might also improve the predictions obtained by an unsupervised search-based approach, but they usually do not consider cavity and binding site information to predict the function of proteins.

6.5 Conclusion

In this chapter I have recast the EC annotation problem as a conditional ranking problem. I have shown that retrieval of enzymes w.r.t. functionality can be substantially improved by applying a supervised ranking method that takes advantage of ground truth EC numbers during the training phase. In contrast, more traditional methods rely heavily on a notion of similarity to search for functionally related enzymes in online databases. Such methods lead to an unsupervised approach in which annotations are not taken into account.

I focused specifically on cavity-based similarity measures, because their benefits compared to sequence-based and structure-based approaches have been demonstrated in previous works, although the method can work with any meaningful similarity measure defined on

enzymes. In the experiments I could demonstrate a considerable improvement of the quality of the overall ranking. The results were influenced by the type of data used and the way the ranking was evaluated, indicating that the most optimal method is highly dependent of the specific problem setting. Nevertheless, the supervised ranking algorithm outperformed the unsupervised ranking algorithm for all cavity-based similarities for most performance measures considered. While the unsupervised approach succeeded quite well in returning exact matches to a query, the hierarchical structure of EC numbers was better preserved in the rankings predicted by the supervised approach. As such, supervised ranking can be interpreted as a powerful alternative for retrieval methods that are more traditionally used in bioinformatics.

Acknowledgements

This work was done in collaboration with dr. Thomas Fober and dr. Serghei Glinca, who provided the data set and associated similarities.

7

Detecting false negatives in species interaction datasets

Species interaction datasets, often represented as sparse matrices, are usually collected through observation studies targeted at identifying species interactions. Due to the extensive required sampling effort, species interaction datasets usually contain many false negatives, often leading to bias in derived descriptors. I show that a simple linear filter can be used to detect false negatives by scoring interactions based on the structure of the interaction matrices. On 180 different datasets of various sizes, sparsities and ecological interaction types, I found that on average in about 75% of the cases, a false negative interaction got a higher score than a true negative interaction. Furthermore, I show that this filter is very robust, even when the interaction matrix contains a very large number of false negatives. The results demonstrate that unobserved interactions can be detected in species interaction datasets, even without resorting to information about the species involved.

The linear filter that is used in this chapter is very related to the kernel-based methods discussed in Chapter 4. As shown in Section 4.5.5, if one uses smoothing kernels to describe the objects, the resulting models can be written as a simple linear combination of averages of the labels. These models are used in tandem with a leave-

one-out shortcut, analogous to the ones developed in Section 4.6.1, to detect false negatives.

7.1 A filter for species interaction matrices

Biological data such as microscopy images, environmental sensor readings and species incidence counts are inherently noisy. Often a simple linear transformation can be applied to obtain a denoized reestimation of the data (MacKay, 2003). For instance, a noisy image can be rectified by applying a filter that exploits the fact that adjacent pixels in an image tend to have similar values (Gonzalez and Woods, 2007). Similarly, species interaction values are not randomly distributed, but exhibit structures such as nestedness (Bascompte et al., 2003; Bastolla et al., 2009), modularity (Olesen et al., 2007) or low-dimensional embedding (Eklöf et al., 2013). Since these interactions are largely determined by evolved traits of both partners (Junker et al., 2013; Hadfield et al., 2014; Shimizu et al., 2014), a filter for these types of data could take this information into account.

Machine learning methods, often based on kernels, have been applied with great success in similar cases, for example to predict interaction values between biomolecules based on sequence information (Ben-Hur and Noble, 2005; Vert et al., 2007; Pelossof et al., 2015), but seem to have remained absent from an ecological context. If no side information such as traits or phylogeny of the individual species is available, only the structure of the interaction dataset can be exploited. This can be realized by letting the filtered interaction values not only depend on the observed interaction, but also on the degree to which the two species in the interactions are involved in other interactions. Let $\mathbf{Y} = [Y_{ij}]$ be the sparse $n \times m$ matrix of interaction values, either a binary matrix or a matrix of positive real numbers expressing

interaction strength. I refer to the non-zero values, i.e. detected interactions, as positive interactions, and to the zero values, i.e. absent interactions, as negative interactions[1]. The filtered interaction matrix $\mathbf{F} = [\mathbf{F}_{ij}]$ can be obtained as the following weighted average of averages:

$$\mathbf{F}_{ij} = \alpha_1 \mathbf{Y}_{ij} + \alpha_2 \frac{1}{n} \sum_{k=1}^{n} Y_{kj} + \alpha_3 \frac{1}{m} \sum_{l=1}^{m} Y_{il} + \alpha_4 \frac{1}{nm} \sum_{k=1}^{n} \sum_{l=1}^{m} Y_{kl} \,, \quad (7.1)$$

where $(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \in [0,1]^4$ and $\sum_{i=1}^{4} \alpha_i = 1$. The first term is proportional to the interaction value, while the last term is proportional to the average of all interaction values in the matrix. The second (resp. third) term is proportional to the average of the values in the corresponding column (resp. row), i.e. relative to the promiscuity of the individual species. The parameters $\alpha_1, \alpha_2, \alpha_3$ and α_4 act as weighting coëfficiënts. This filter is illustrated on a toy dataset in Figures 7.1(a)(b)(c). See Section 4.5.5 for a more in-depth discussion of this filter and a connection to the other models used in this work.

Usually, interaction datasets are sampled by monitoring one of the species types and observing the number of interactions with the species of the other type (Goldwasser and Roughgarden, 1997) (e.g. studying the fecal matter of predators to assess their preys or keeping track of pollinators landing on plants). As a consequence, these interaction matrices are often undersampled and some zeros might be false negatives rather than true negative interactions (Blüthgen, 2010; Chacoff et al., 2012). This can lead to some serious biases in descriptors derived from such matrices (Goldwasser and Roughgarden, 1997; Banašek-Richter et al., 2004; Fründ et al., 2015; Jordano, 2016). To assess whether a particular interaction between species i and species

[1] In ecological literature, 'positive interaction' is often used to refer to an interaction in which both species benefit (e.g. symbiosis), while 'negative interaction' is used for an interaction where one of the species has a disadvantage (e.g. parasitism). In this work, I use the term positive (resp. negative) interactions to refer to an observed (resp. unobserved) interaction, regardless of the nature of the interaction. This is more consistent with standard statistical terminology.

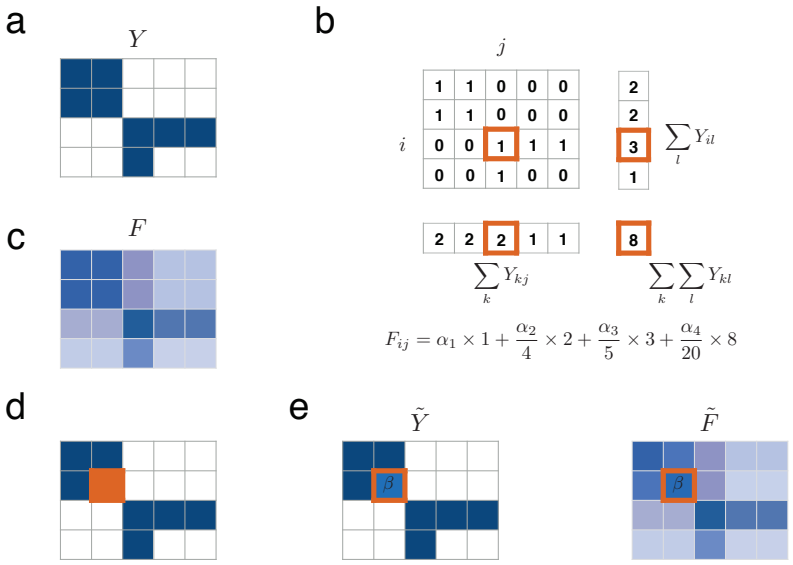


Figure 7.1: (a) A small binary interaction dataset $\mathbf{Y}$ of four by five species. (b) Example of how to calculate the filtered value for $\mathbf{Y}_{ij}$. The filter in Eq. (7.1) computes a weighted average of the observed value itself, the column and row averages and the average of all interaction values in the matrix. (c) The filtered dataset $\mathbf{F}$ corresponding to $\mathbf{Y}$. (d) The dataset $\mathbf{Y}$ where one value is to be imputed, indicated in orange. The imputation of this interaction value should be independent of the original, possibly wrong, value. (e) and (f) The value β of the imputed interaction does, by definition, not change when passing through the filter.

j is likely to occur in reality according to the dataset, one should ideally not make use of the observed interaction value $\mathbf{Y}_{ij}$. We therefore impute this interaction value, further on denoted as β, in such a way that when it is passed through the filter, it remains unchanged. This embodies the rationale that we want to impute the interaction value to closely match the rest of the data according to the filter. Consider Eq. (7.1) using a copy of $\mathbf{Y}$ where $\mathbf{Y}_{ij}$ is replaced by β, then it should

hold that:

$$\beta = \frac{\alpha_2}{n} \sum_{\substack{k \\ k \neq i}} Y_{kj} + \frac{\alpha_3}{m} \sum_{\substack{l \\ l \neq j}} Y_{il} + \frac{\alpha_4}{nm} \sum_{\substack{k,l \\ (k,l) \neq (i,j)}} Y_{kl} + \left(\alpha_1 + \frac{\alpha_2}{n} + \frac{\alpha_3}{m} + \frac{\alpha_4}{nm} \right) \beta \tag{7.2}$$

$$= \left[\mathbf{F}_{ij} - \left(\alpha_1 + \frac{\alpha_2}{n} + \frac{\alpha_3}{m} + \frac{\alpha_4}{nm} \right) \mathbf{Y}_{ij} \right] + \left(\alpha_1 + \frac{\alpha_2}{n} + \frac{\alpha_3}{m} + \frac{\alpha_4}{nm} \right) \beta \,. \tag{7.3}$$

This is illustrated in Figures 7.1(d)(e)(f) for the toy dataset. Solving for β, we obtain

$$\beta = \frac{\mathbf{F}_{ij} - \left(\alpha_1 + \frac{\alpha_2}{n} + \frac{\alpha_3}{m} + \frac{\alpha_4}{nm} \right) \mathbf{Y}_{ij}}{1 - \left(\alpha_1 + \frac{\alpha_2}{n} + \frac{\alpha_3}{m} + \frac{\alpha_4}{nm} \right)} \,. \tag{7.4}$$

This imputation does *not* depend on the original value of $\mathbf{Y}_{ij}$, as can be gleaned from Eq. (7.2). Only the other interaction values in the dataset contribute to the imputation. The process of imputing the interaction values one by one is known as LOO imputation. Equation (7.4) is a special case of the well-known LOO shortcut (Wahba, 1990) and provides a computationally efficient way of performing LOO imputation.

As a simple method to detect false negatives in interaction matrices, I suggest to score negative interactions in datasets using LOO imputation and rank the negative interactions according to this score[2]. Negative interactions that receive high scores during imputation are potential false negatives and should be examined more closely. In the experiments I will demonstrate, first, that imputations of positive interactions will on average result in higher scores than negative interactions and, second, that false negatives in turn receive higher scores than true negatives, making this a suitable method for false negative discovery. The proposed linear filter will be compared to the

[2]The last term in Eq. (7.1), i.e. the average interaction value, will not influence the ranking of interactions. However, if the goal is to impute the interaction value to some degree of accuracy, this term provides an essential contribution.

use of a low-rank approximation of the interaction matrix, obtained through singular value decomposition (SVD), a popular method to impute missing values in collaborative filtering (Zhang et al., 2005; Isinkaye et al., 2015). The re-estimation using SVD is obtained by retaining only the leading eigenvalues of the matrix $\mathbf{Y}$ after decomposition. Since the eigenvalue spectrum of the interaction dataset is related to the nestedness of the network (Staniczenko et al., 2013), it seems sensible that this method could work well for nested interaction networks. The filter works demonstratively better than SVD in most cases and remains performant even with very high rates of false negative interactions. Finally, I illustrate that when forbidden links (i.e. true negatives) are known, the performance can be increased slightly.

7.2 Material and methods

In the experiments I used a series of species interaction datasets obtained from the Interaction Web DataBase[3] and Web of Life database[4]. I only withheld datasets with at least ten rows and ten columns, leaving us with 180 datasets describing anemone-fish, host-parasite, plant-ant, plant-herbivore, plant-seed dispensers, plant-pollinator and predatory-prey interactions. I have chosen such a diverse catalogue of datasets to illustrate that the proposed method is broadly applicable. Some datasets contained only binary absence-presence information, others contained valued interactions, such as frequency of visits. The method can be applied regardless. All datasets were quite sparse, with an average positive interaction density ρ of 0.15 ± 0.12 (average value $\pm$ standard deviation calculated over the different datasets).

In this chapter I investigate whether the scores of imputed interaction values can be used to discriminate between unobserved positive

[3] https://www.nceas.ucsb.edu/interactionweb/resources.html
[4] http://www.web-of-life.es/

and negative interactions. As a performance metric, I will use the area under the ROC curve (AUC), calculated as

$$A = \frac{1}{|\mathcal{S}^+||\mathcal{S}^-|} \sum_{(i,j)\in\mathcal{S}^+} \sum_{(k,l)\in\mathcal{S}^-} H(\mathbf{F}_{ij} - \mathbf{F}_{kl}) \,, \tag{7.5}$$

with $\mathbf{F}_{ij}$ the imputed score, $\mathcal{S}^+$ (resp. $\mathcal{S}^-$) the set of the positive (resp. negative) interactions and $H(\cdot)$ the Heaviside step function. The AUC can be interpreted as the probability that a randomly chosen positive interaction receives a higher score than a randomly chosen negative interaction.

The LOO imputations of the interaction datasets were computed using Eq. (7.4). Since we use AUC to evaluate the imputations, we are not interested in the exact values. Rather, positive interactions should on average receive higher imputed values compared to negative interactions. A small explorative study on a couple of datasets has shown that the ranking-based evaluation using AUC is quite insensitive to the exact values of the parameters of the filter. Hence, I have set all parameters equal, i.e. $(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = (0.25, 0.25, 0.25, 0.25)$, meaning that each of the four averages in Eq. (7.1) has the same weight. The filter is thus reduced to a standard average. If the filter would be used to estimate the probability of interaction or the interaction strength, I recommend to do some tuning of the parameters to the dataset at hand, for example, using cross-validation to minimize squared loss.

7.3 Detecting false negative species interactions

First, I show that a positive interaction receives a higher score than a negative interaction. For each dataset, I calculated the LOO imputation and compared the scores of the positive and the negative interactions. The average AUC was found to be 0.77 ± 0.10, meaning that on average there is about 77% chance that a missing positive interaction will receive a higher score than a missing negative interaction. Intriguingly, I found that using the strength of the interactions tends to decrease the performance. When datasets containing strength of interactions were binarized by setting positive values to one, the performance increased on average with $3.5\% \pm 4.4\%$. A paired t-test showed that this increase in average AUC is significant at the 0.01 level ($p \ll 10^{-10}$, $n = 94$ datasets). This implies that in many cases the strength of interaction is too noisy to be exploited by the filter. This was to be expected, as quantitative interaction strength depends on local conditions (Wootton and Emmerson, 2005; Berlow et al., 2004), and is therefore more susceptible to noise. Hence, making the interaction matrix binary often leads to more robust filtering.

Four sizeable datasets representing different types of interactions were studied in more detail, see Figure 7.2. In Figure 7.3(a) the ROC curves illustrate that usually a large fraction of the positive interactions can easily be detected without obtaining many false positives. This is important for practical applications, as these high-scoring interactions should be used to decide which interactions are promising for validation in the field. The top-scoring interactions are strongly enriched with positives, as illustrated in Figure 7.3(b), which shows the precision (fraction of top-scoring positive interactions) as a function of the size of the top. Although the individual patterns vary with the density, distribution and sampling effort of the interaction datasets,

here one can observe also a clear trend that making the datasets binary results in higher precision. On average, for all datasets, the precision at the top-10 was 0.69 ± 0.27, which is substantially higher than the average density of 15%, the expected precision of a random scoring.

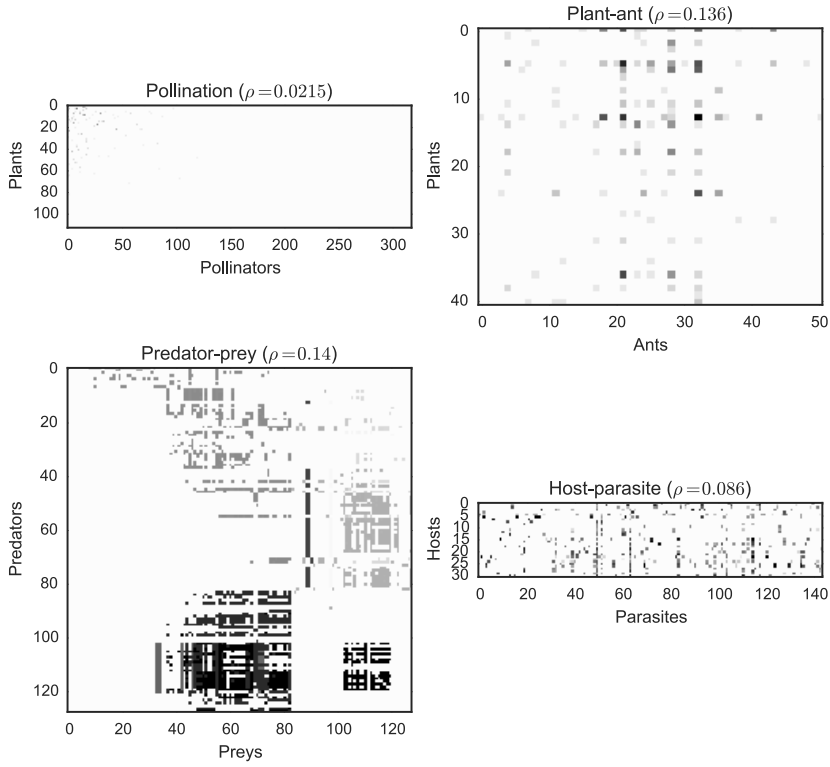


Figure 7.2: Heat maps of four valued species interaction datasets with the corresponding density ρ. These are interaction networks of, respectively, pollination (Kakutani et al., 1990), plant-ant (Blüthgen et al., 2004; Blüthgen and K. Fiedler, 2004), predator-prey (Lafferty et al., 2006) and host-parasite interactions (Dechtiar, 1972). The brightness of the color corresponds to the value of the interaction.

Since most species interaction datasets are obtained through observation studies, negative interactions may either indicate that the

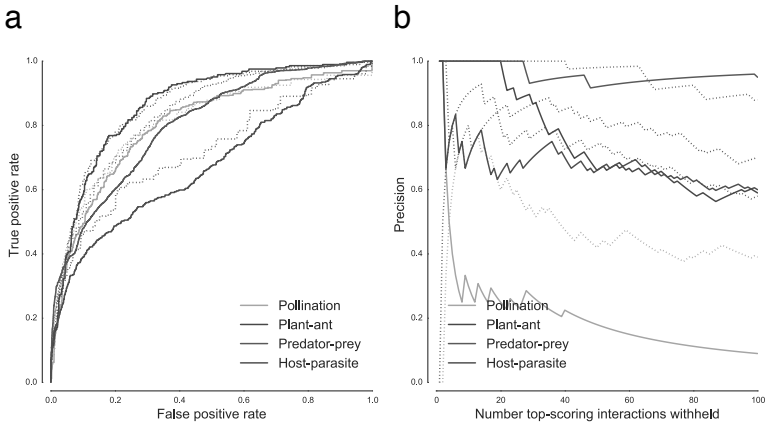


Figure 7.3: Results of the imputation experiments using the four datasets shown in Figure 7.2. (a) ROC curves for the scores of the LOO imputation. (b) The precision of detecting true interactions as a function of the size of top-scoring interactions. In both plots full lines represent experiments where the intensity of the interactions was used and broken lines represent experiments where the interaction dataset was binarized.

species do not interact in practice or that their interaction is not observed during the study. To show that linear filtering can reveal false negatives, I created variants of each dataset, each with exactly one positive interaction made negative, and I did this for every positive interaction. Subsequently, all negative interactions were scored using LOO imputation and the score of the false negative was compared with the scores of the true negatives (Figure 7.4). The average AUC for detecting these false negatives was 0.78 ± 0.098, averaged over all the 180 datasets. Again, when the interaction datasets containing strength of interaction were binarized, the performance increased with on average $4.0\% \pm 4.4\%$. Using a paired t-test, this increase in average AUC was also found to be significant at the 0.01 level ($p \ll 10^{-10}$, $n = 94$ datasets). Whereas the previous experiment showed that positive interactions receive higher scores than negative interactions, this experiment demonstrates that within the negative interactions, false

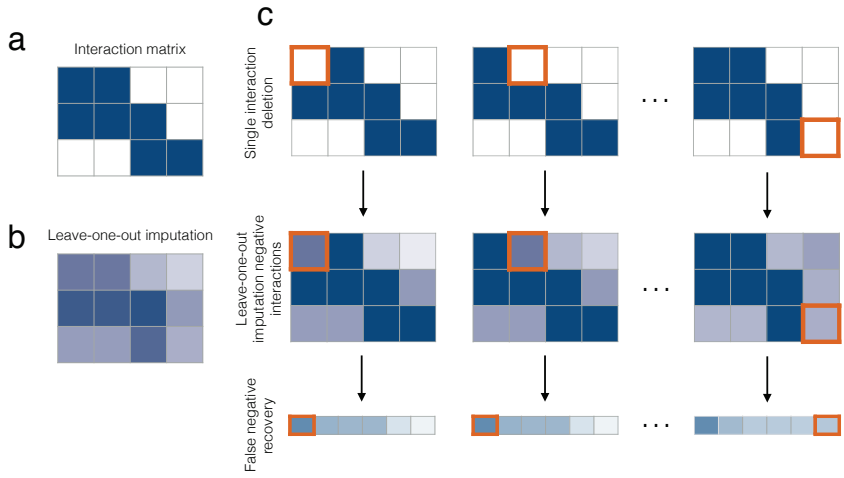


Figure 7.4: (a) A small binary interaction matrix $\mathbf{Y}$ of three by four species. (b) The corresponding matrix where each value is computed by LOO imputation. The score of each interaction is calculated only based on the values of all other interactions, without its original value. (c) To test if a false negative can be detected, each positive interaction is made negative one by one, indicated in orange. For each of these changed datasets, all negative interactions are scored using LOO imputation. The negative interactions are sorted by their scores and the position of the false negative, indicated by an orange frame, is determined.

negatives tend to receive higher scores than true negatives. Table 7.1 summarizes the AUC scores obtained for the two described experiments.

Even when many interactions are missing, the method remains performant. In an additional experiment, first, I illustrate how the performance of the linear filter changes with larger fractions of false negatives and, second, I compare the linear filter to the use of a low-rank approximation of the interaction matrix $\mathbf{Y}$ obtained by SVD. SVD can be used to obtain the closest approximation in terms of mean squared error of a matrix for a given rank. The rank was chosen as the lowest rank such that the approximated dataset retained at

Table 7.1: Average AUC, aggregated for different densities ρ and different total numbers of positive interactions in all the different datasets. The first part gives the results for the imputation experiments, the second part presents the results for the false negative recovery experiments. All datasets with interaction strengths were binarized for these experiments.

	imputation AUC			false negative recovery AUC		
density ρ	]0, 0.1]	]0.1, 0.25]	]0.25, 1]	]0, 0.1]	]0.1, 0.25]	]0.25, 1]
# interactions						
]0, 50]	0.8187	0.6119	0.7180	0.8443	0.6489	0.7426
]50, 100]	0.7561	0.7194	0.8017	0.7696	0.7341	0.8190
]100, 1000]	0.8259	0.7857	0.8002	0.8301	0.7925	0.8088
]1000, 10000]	0.8219	0.8423	-	0.8232	0.8429	-
]10000, $+\infty$[	0.8482	-	-	0.8486	-	-

least 75% of the variance of the original dataset. The re-estimated matrix was evaluated the same way as the matrix obtained by LOO imputation using the linear filter. Experiments using both the linear filter and the SVD approximation were performed on the four datasets in Figure 7.2, by randomly setting 5%, 10%, 20%, 50% or 90% of the positive interaction values to zero. Using AUC, I assessed how well the re-estimated interaction values could be used to discriminate between true and false negatives. Re-estimation was done using both the original interaction datasets and versions of the datasets where the interaction values were binarized. Each experiment was repeated 100 times. The performances are listed in Table 7.2. For three datasets, the linear filter clearly shows a better performance. Interestingly, SVD seems to work really well on the predator-prey dataset, a large dataset with visually a strong structural pattern. Nevertheless, using the linear filter usually leads to a good performance, especially since most interaction matrices are rather small. This filter also seems to be still able to detect false negative interactions even when the percentage of false negatives is very high, in contrast to using the low-rank approximation. This indicates that the method is quite robust, even when the datasets contain many missing values.

Finally, I performed a small experiment where true negatives or forbidden links are known. To this end, I use the 25-by-25 seed-dispersal network of Olesen and coauthors (Olesen et al., 2011). It consists of 156 observed positive interactions and 228 forbidden interactions due to phenological uncoupling or morphological constraints. I used the linear filter to perform LOO imputation on the interaction matrix. Figure 7.5 shows the distributions of the imputed values for the positive interactions, true negative interactions and negative interactions that are potential false positives. The AUC for discriminating between positive and negative interactions (both true negatives and false negatives) using LOO imputation was found to be 0.8270. When only trying to discriminate between true positives and true negatives,

Table 7.2: Comparison of the linear filter with SVD for an increasing fraction of randomly assigned false negatives (FN) for four datasets. The AUC is given for both the original dataset and a binarized version. Each performance is an average of 100 repetitions. In most cases the linear filter is better than SVD. The performance of the latter deteriorates quickly with an increasing number of false negatives. The performance of the linear filter remains relatively high, even with 90% of false negatives.

		Host-parasite		Plant-ant		Pollination		Predator-prey	
	binarized	No	Yes	No	Yes	No	Yes	No	Yes
FN fraction	method								
0.05	SVD	0.6139	0.5672	0.6129	0.4876	0.5672	0.4967	0.8642	0.9031
	filter	0.6801	0.7110	0.8596	0.8799	0.7988	0.8131	0.7997	0.8405
0.10	SVD	0.5863	0.5611	0.6029	0.4931	0.5671	0.4957	0.8202	0.8514
	filter	0.6729	0.7178	0.8611	0.8629	0.8019	0.8074	0.7996	0.8390
0.20	SVD	0.5825	0.5563	0.5833	0.4962	0.5542	0.4863	0.7618	0.7705
	filter	0.6720	0.7149	0.8653	0.8620	0.7965	0.8059	0.7974	0.8343
0.50	SVD	0.5493	0.5284	0.5520	0.4947	0.5274	0.4897	0.6525	0.6326
	filter	0.6615	0.7017	0.8476	0.8438	0.7881	0.7971	0.7937	0.8246
0.90	SVD	0.5026	0.5008	0.5025	0.5022	0.5015	0.4978	0.5159	0.5089
	filter	0.6168	0.6342	0.7511	0.7366	0.7215	0.7255	0.7532	0.7599

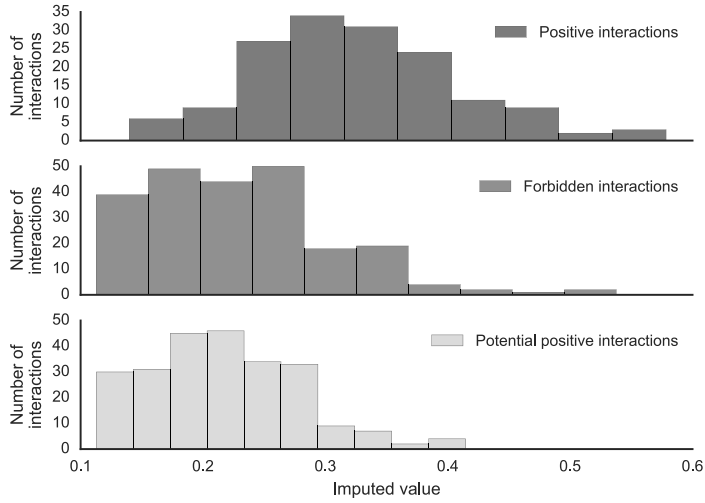


Figure 7.5: Histogram of the imputed values for the positive interactions, forbidden interactions and negative interactions, which are potential false positives. The positive interactions are on average imputed with a higher score than both kinds of negative interactions.

the AUC was 0.7981. Upon removing the true negatives, the AUC improved slightly to 0.8543. For this dataset, it seems that the true negatives are somewhat harder to identify than the negatives in general. When true negatives are known, it is best to only search for false negatives within the potentially positive interactions.

7.4 Discussion

Evidently, the latent information in the interaction matrices can be used to detect unobserved (false negative) interactions. I am convinced that techniques such as linear filtering may allow to either directly ameliorate an interaction dataset or can be used to suggest promising interactions that can subsequently be verified in the field. Making use of *in silico* predicted interaction scores to suggest ex-

periments *in vitro* is already commonplace in domains such as drug discovery (Jorgensen, 2004) and can be seen as part of the broader paradigm of recommender systems (Lü et al., 2012; Zeng et al., 2014). Negative interactions with high scores are natural targets for increased sampling effort, as they are most likely to occur in reality.

Standard algorithms for recommender systems make recommendations by exploiting structures in the data, e.g. low-rankness of the interaction matrix (Mazumder et al., 2010). This idea could be applied to predict the value of missing interactions. For example, it has been used successfully to predict the joint growth between heterotrophic and methanotrophic bacteria Stock et al., (2013). Other methods for filtering a network could be based on different principles, for example the stochastic block model (Guimerà and Sales-Pardo, 2009). In essence, the simple linear filter of Eq. (7.1) and the associated imputation formula (7.4) only use information on row and column counts to do an imputation. We can motivate the use of this filter in three ways. Firstly, it is a very simple first method to try to infer false negatives. Although despite having four parameters, their exact value is less important if one is only interested in ranking interactions, so not much tuning is required. Secondly, the filter is very robust and works demonstratively well on small datasets and with a very large fraction of false negatives. Finally, using the shortcut for LOO cross validation, it is very easy and computationally efficient to get a realistic estimate of the performance of the filter for a given dataset. More complex methods are expected to yield better performance, but require to be tuned more carefully to the dataset at hand.

Often, one has information about the individual species, such as geographical location, morphology or phylogeny, which can also be incorporated to predict interaction (Rafferty and Ives, 2013; Hadfield et al., 2014; Morales-Castilla et al., 2015). Using such side information, denoted as content-based filtering in recommender systems (Lü et al., 2012), can improve the accuracy of the prediction as well as ex-

plain the interactions based on species traits, if used in combination with model selection tools. As I have not incorporated such information in the method, the performances presented in this work can be seen as a lower bound for detecting missing interactions.

8

Conclusions and future perspectives

In this PhD dissertation, I have studied pairwise prediction functions of the form

$$f(u, v) = \boldsymbol{\phi}(u)^\top \mathbf{W} \boldsymbol{\psi}(v)\,. \tag{8.1}$$

Here $\boldsymbol{\phi}(\cdot)$ and $\boldsymbol{\psi}(\cdot)$ are the feature descriptions of the objects. Any *prior knowledge about the nature of the objects* should be incorporated in these descriptions. The weight matrix $\mathbf{W}$ is learned from data. Any *prior knowledge on the nature of the pairwise relationship* between u and v should be incorporated in the learning algorithm used to determine $\mathbf{W}$.

Though Eq. (8.1) is a linear model written in primal form, it can learn nonlinear patterns through the nonlinear feature descriptions. In this sense, kernel methods (and related methods) offer the best of both worlds. They combine the efficiency and theoretical guarantees of linear learning algorithms with the flexibility of nonlinear learning algorithms. The methods discussed in this dissertation are in a sense *especially linear*. Both the predictions and the parameters are linear combinations of the labels:

$$\mathrm{vec}(\mathbf{F}) = \mathbf{H}\mathrm{vec}(\mathbf{Y}) \tag{8.2}$$

$$\mathrm{vec}(\mathbf{W}) = \mathbf{B}\mathrm{vec}(\mathbf{Y})\,. \tag{8.3}$$

The structure of the matrices **H** and **B** depends on the feature descriptions and on which of the pairwise learning methods described in Section 4.4 is used. These equations arise because these methods optimize for a squared loss. If a hinge or a logistic loss is optimized, no such nice closed-form solutions are obtained. It is because of the linearity of the methods discussed in this work that the algorithmic shortcuts described in for example Section 4.6 can be derived.

Though systems (8.2) and (8.3) both solve an *inverse problem*, given the observed label matrix, one tries to find either the re-estimated labels **F** or the weights of the model **W** to generate these weights. This inverse problem is solved using the eigenvalue decompositions of the two Gram matrices **K** and **G** in conjunction with a pairwise filter function to efficiently generate the re-estimations or the model weights. All pairwise learning algorithms, independent task KRR, Kronecker KRR, ordinary KRR, two-step KRR and the linear filter of Eq. (7.1), efficiently combine the different feature descriptions of the objects to make predictions.

An important contribution of this PhD dissertation are the exact algorithms presented in Section 4.6. Firstly, the efficient leave-one-out cross validation shortcuts allow for a very fast model selection and validation for the different settings. I have also shown in Chapter 7 that these shortcuts can be used to develop a method to detect false negative interactions. Secondly, if new labeled objects are encountered, the model parameters can efficiently be updated using this new data. This extends the pairwise learning framework to large-scale data applications (in which the parameters are fitted using several batches of training data) or in a dynamical context, where new data is made available. Lastly, I have presented a simple iterative algorithm that can learn efficiently from incomplete data and have shown both theoretically as well as empirically that this rapidly converges to the exact solution. I consider this to be of particular importance, as the elegance and efficiency of Kronecker kernel-based ridge regres-

sion for pairwise learning is now available for incomplete data. This contribution greatly expanded the scope of the previous Kronecker kernel-based ridge regression models. In a similar spirit, I have shown that using the pairwise prediction models for top-K inference can also be done very efficiently if some precomputing is done in advance. Here, keeping a set of sorted lists of the database objects is the key to greatly reduce the computations for obtaining the correct top-K set.

Since pairwise learning is such a general problem, it is related to many other machine learning paradigms, as discussed in Chapter 2 and Sections 4.3 and 5.4. However, the framework discussed in this PhD thesis is intrinsically linked to a feature description of the objects. In Section 4.5.5 I show that this framework can even be used when no feature description (other than the identity of the object) is available of the objects. In a sense, this links collaborative filtering with content-based filtering. Chapter 7 shows that even without features of the objects (*in casu* species) a good performance can be achieved and Figure 4.9 shows that not using the feature description at all might even be beneficial in some cases.

Much emphasis has been on the different prediction settings discussed in Section 2.3.2. Of the four settings, two are subjectively the most interesting, Setting A and Setting D. Setting A, in this dissertation represented by Eq. (8.2), is the re-estimation problem. It links pairwise learning to the world of network analysis, matrix completion, recommender systems and even image analysis (as I have explored in the experiments of Section 4.7.2). Setting D, on the other hand, is the complete generalization towards new objects. For this, Eq. (8.3) is used to obtain parameters for the general pairwise prediction model. To me, an interesting extension to setting D would be the structured output prediction problem of Eq. (2.3). I will briefly elaborate on my vision for both.

8.1 A general system representation (Setting A)

For Setting A, let us look at a broader picture. The label matrix $\mathbf{Y}$ is not merely a collection of labels, it is a description of a system. For example, it can describe a molecular network or the interactions between a set of species. In many cases, one would represent such a system by a graph or network rather than a matrix, though this representation is mathematically equivalent. Many questions arise when analysing such a system:

- Since $\mathbf{Y}$ is obtained empirically, it likely contains errors, how do we *curate* this matrix?
- A system has *emergent properties*, for example the stability of a molecular interaction network for external perturbations. How can we learn these emergent properties?
- How do we *compare* different systems?
- How *complex* is the system of interest?
- How to learn the *dynamics* of the system, how does it evolve through space and time?

Of course, the first research question is explored in depth in this PhD dissertation. Generally speaking, I would argue that all these questions can be addressed by constructing a common representation $\Phi(\mathbf{Y})$ for these systems. Using a suitable transformation, the system will be *encoded* such that it can be used for either recovering a re-estimated version (*decoding*) of the system or as a basis to make predictions for the system.

Kernels could be an excellent starting point to represent ecological networks, though I would put emphasis on more modern approaches.

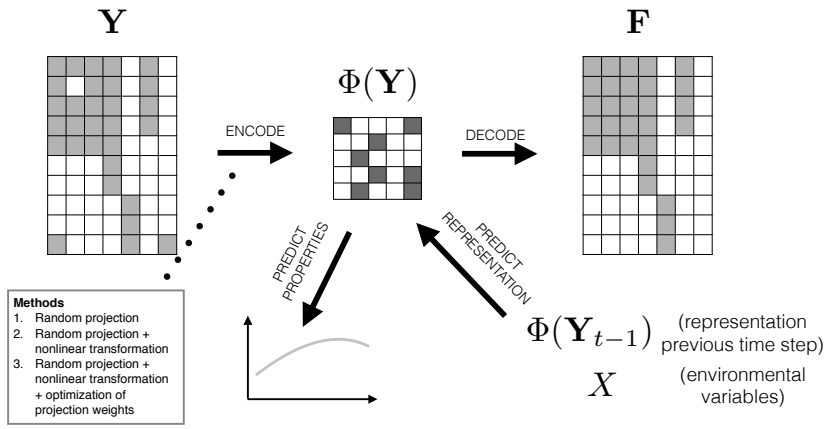


Figure 8.1: Concept for constructing a general representation for a system that can be described by a matrix.

I would explore representations in steps of increasing complexity and flexibility. Firstly, I would explore representations based on random numerical linear algebra (Drineas and Mahoney, 2016). Here, large datasets are approximated using random sampling or projections. These tools are particularly interesting for biological applications, as they allow to generate a sparse representation based on, for example, the few nodes of a network that are most informative. Secondly, in addition to the random projection, I would add a nonlinearity. This has been shown to emulate kernel methods, in the sense that nonlinear patterns in the data can be found, e.g. Lopez-Paz et al. (2014). As a final extension, the parameters of the random projection can be optimized for re-estimation or prediction. This essentially boils down to an artificial neural network, either an autoencoder (Baldi, 2012) (optimized for re-estimation) or a standard feed-forward neural network (optimized for prediction). See Figure 8.1 for a visualization of this conceptional framework.

This representation can be directly used to answer many research questions:

- Re-estimating the system matrix from the representation, so that, for example, noise in sampling the system is reduced.
- Use the representation as a feature basis for predicting the emergent properties of the system.
- Using the representation to analyze complexity. For example, if the representation can be given a probabilistic interpretation, we can borrow ideas from information theory to analyze the network. This has been explored in a project tutored by me (Dhiedt et al., 2016).
- Using the representation to compare similar systems from a different time or space.
- The representation itself can be predicted based on external features $(\Phi(Y) \mid X)$ or a representation of the system at a previous time step $(\Phi(Y_t) \mid \Phi(Y_{t-1}))$. From this predicted representation, a system under these conditions can be recovered using the decoder. This idea can be used to predict how systems will change.

8.2 Structured output prediction with labels (Setting D)

For Setting D, it is interesting to think of pairwise learning as an extension of structured output prediction. Recall Eq. (2.3) where we treated structured output prediction as an optimization problem:

$$u^* = \arg\max_{u \in \mathcal{U}} f(u, v)\,.$$

It is interesting to make the distinction between typical structured output prediction and pairwise learning. Structured output prediction

learns models to perform the mapping $\mathcal{V} \to \mathcal{U}$ (in the notation of this work). To this end, a training set of the form $\{(u_i, v_i) \mid i \in \{1, \ldots, n\}\} \subseteq \mathcal{U} \times \mathcal{V}$ is used, for example:

- A set of images, each with a sentence describing the image.
- A collection of sentences with the associated grammar trees.
- A set of proteins, each with a ligand that binds it.
- A set of RNA sequences with their observed secondary structure.

So, structured output prediction problems start from a set of examples of an input with the *correct* output attached to it. Pairwise learning models on the other hand learn the mapping $\mathcal{U} \times \mathcal{V} \to \mathcal{Y}$ based on a set of examples of the form $\{(u_i, v_i, y_i) \mid i \in \{1, \ldots, n\}\} \subseteq \mathcal{U} \times \mathcal{V} \times \mathcal{Y}$. This is actually a much richer source of information, as for each couple (u, v), we are provided with the *quality* of the combination of objects. As such, we can construct a model that can also learn from negative examples (in a classification context) or from gradations of matches (in a regression context). Using pairwise datasets, we can learn a model that provides *for a given v a suitable $u \in \mathcal{U}$ with an as high as possible predicted value y.*

Most pairwise models presented in this work are not optimal for structured output prediction using quality information, as they learn to predict a label conditional on a couple (u, v). For the type of prediction in mind, we should model the probability or energy of (y, u) conditional on v (a probable u with high score y for a given u). This would need a different approach of modelling. I believe a framework based on energy-modelling (see LeCun et al. (2006) for a tutorial) or kernel dependency estimation (see Song et al. (2013) for an overview) would provide us with the relevant tools.

All models in this work are simple (though powerful) and seldomly require more than a few lines of computer code to be implemented. In recent years, a wide variety of computer packages have become commonplace which reduce 'several pages of mathematical derivations and

months of work implementing the model' to 'something you can implement in an afternoon'. This includes probabilistic programming languages such as PyMC3 and STAN, that bring probabilistic and Bayesian modelling to the masses and packages such as Theano and TensorFlow, that make constructing a complex artificial neural network a breeze. It would be worthwhile to investigate how these tools can be used in pairwise learning.

Pairwise learning and the further research topics outlined above can be used for a variety of applications. As a bioscience engineer, I believe two research lines, each tied respectively to Setting A and Setting B, have a lot of potential. Firstly, there is the world of ecological networks, see Poisot et al. (2016), how to represent, understand, predict and control ecosystems. Secondly, at the molecular level, there is the world of synthetic biology Gibson (2014), designing new molecular systems. I believe that those two issues, understanding and shaping living systems at micro and macro scale, are at the heart of bioscience engineering. There is no doubt in my mind that computational techniques, machine learning, but also mathematical optimization, data management, simulation, probabilistic modelling, etc., will be vital towards this monumental endeavour.

Bibliography

Abernethy, J., Bach, F., Evgeniou, T., and Vert, J.-P. (2008). A new approach to collaborative filtering: operator estimation with spectral regularization. *Journal of Machine Learning Research*, 10:803–826.

Adams, R. P., Dahl, G. E., and Murray, I. (2010). Incorporating side information in probabilistic matrix factorization with Gaussian processes. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 1–9.

Agarwal, D. and Gurevich, M. (2012). Fast top-k retrieval for model based recommendation. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, pages 483–492.

Agarwal, S., Dugar, D., and Sengupta, S. (2010). Ranking chemical structures for drug discovery: a new machine learning approach. *Journal of Chemical Information and Modeling*, 50(5):716–31.

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.

Ahn, Y.-Y., Ahnert, S. E., Bagrow, J. P., and Barabási, A.-L. (2011). Flavor network and the principles of food pairing. *Scientific Reports*, 1(196).

Albert, R., DasGupta, B., Dondi, R., Kachalo, S., Sontag, E., Zelikovsky, A., and Westbrooks, K. (2007). 14:927–949.

Allesina, S. and Levine, J. M. (2011). A competitive network theory of species diversity. *Proceedings of the National Academy of Sciences of the United States of America*, 108(14):5638–5642.

Aloy, P. and Russell, R. B. (2006). Structural systems biology: modelling protein interactions. *Nature Reviews. Molecular Cell Biology*, 7(3):188–197.

Alt, H. and Guibas, L. J. (1996). Discrete geometric shapes: matching, interpolation, and approximation: a survey. In *Handbook of Computational Geometry*.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410.

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402.

Alvarez, M. A., Rosasco, L., and Lawrence, N. D. (2012). Kernels for vector-valued functions: a review. *Foundations and Trends in Machine Learning*, 4(3):195–266.

Anderson, C. (2006). *The Long Tail: Why the Future of Business is Selling Less of More*. Hyperion Books.

Andersson, C., Chen, B. Y., and Linusson, A. (2010). Mapping of ligand-binding cavities in proteins. *Proteins*, 78(6):1408–1422.

Antal, M. A., Böde, C., and Csermely, P. (2009). Perturbation waves in proteins and protein networks: applications of percolation and game theories in signaling and drug design. *Current Protein & Peptide Science*, 10(2):161–172.

Arakaki, A. K., Huang, Y., and Skolnick, J. (2009). EFICAz2: enzyme function inference by a combined approach enhanced by machine learning. *BMC Bioinformatics*, 10:107.

Argyriou, A., Micchelli, C. A., M. Pontil, and Y. Massimiliano (2010). On spectral learning. *Journal of Machine Learning Research*, 11:935–953.

Argyriou, A., Micchelli, C. A., Pontil, M., and Ying, Y. (2007). A spectral regularization framework for multi-task structure learning. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems*, pages 25–32.

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404.

Babbitt, P. C. and Gerlt, J. (1997). Understanding enzyme superfamilies. *Journal of Biological Chemistry*, 272(49):30591–30594.

Bakir, G. H., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., and Vishwanathan, S. V. N. (2007). *Predicting Structured Data*. The MIT Press.

Bakir, G. H., Weston, J., and Schölkopf, B. (2004). Learning to find pre-images. In *Advances in Neural Information Processing Systems 16*, number iii, pages 449–456.

Baldassarre, L., Rosasco, L., Barla, A., and Verri, A. (2012). Multi-output learning via spectral filtering. *Machine Learning*, 87(3):259–301.

Baldi, P. (2012). Autoencoders, Unsupervised Learning, and Deep Architectures. *ICML Unsupervised and Transfer Learning*, pages 37–50.

Banašek-Richter, C., Cattin, M.-F., and Bersier, L.-F. (2004). Sampling effects and the robustness of quantitative and qualitative food-web descriptors. *Journal of Theoretical Biology*, 226(1):23–32.

Barber, D. (2011). *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 04-2011 edition.

Bartlett, P. P. L. and Mendelson, S. (2002). Rademacher and Gaussian complexities: risk bounds and structural results. *Journal of Machine Learning Research*, 3(3):463–482.

Bartomeus, I. (2013). Understanding linkage rules in plant-pollinator networks by using hierarchical models that incorporate pollinator detectability and plant traits. *PLoS ONE*, 8(7):1–8.

Bartomeus, I., Gravel, D., Tylianakis, J., and Aizen, M. (2016). A common framework for identifying linkage rules across different types of interactions. *Functional Ecology*, 30(12):1894–1903.

Bartomeus, I., Gravel, D., Tylianakis, J. M., Aizen, M., Dickie, I., and Bernard-Verdier, M. (2015). A common framework for identifying rules across different types of interactions. *Functional Ecology*, 10:1–10.

Bascompte, J., Jordano, P., Melián, C. J., and Olesen, J. M. (2003). The nested assembly of plant-animal mutualistic networks. *Proceedings of the National Academy of Sciences of the United States of America*, 100(16):9383–9387.

Basilico, J. and Hofmann, T. (2004a). A joint framework for collaborative and content filtering. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 550–551. ACM Press.

Basilico, J. and Hofmann, T. (2004b). Unifying collaborative and content-based filtering. In *Proceedings of the 21st International Conference on Machine Learning*, pages 9–16.

Bastolla, U., Fortuna, M. A., Pascual-García, A., Ferrera, A., Luque, B., and Bascompte, J. (2009). The architecture of mutualistic networks minimizes competition and increases biodiversity. *Nature*, 458(7241):1018–1020.

Bauer, F., Pereverzev, S., and Rosasco, L. (2007). On regularization algorithms in learning theory. *Journal of Complexity*, 23(1):52–72.

Ben-David, S. and Schuller, R. (2003). Exploiting task relatedness for multiple task learning. In *Proceedings of the 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop*, pages 567–580.

Ben-Hur, A. and Noble, W. (2005). Kernel methods for predicting protein-protein interactions. *Bioinformatics*, 21 Suppl 1:38–46.

Ben-Hur, A. and Noble, W. S. (2005). Kernel methods for predicting protein-protein interactions. *Bioinformatics*, 21(Suppl 1):i38–46.

Bengio, Y. (2016). Springtime for AI: the rise of deep learning. *Scientific American*.

Bengio, Y., Courville, A., and Vincent, P. (2012). Representation learning: a review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798 – 1828.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18:509–517.

Berlow, E. L., Neutel, A.-M., Cohen, J. E., Ruiter, P. C. D., Ebenman, B., Emmerson, M., Fox, J. W., Jansen, V. A. A., Jones, J. I., Kokkoris, G. D., Logofet, D. O., Mckane, A. J., Montoya, J. M.,

and Petchey, O. (2004). Interaction strengths in food webs: issues and opportunities. *Journal of Animal Ecology*, 73(3):585–598.

Bertalmio, M., Sapiro, G., and Caselles, V. (2000). Image inpainting. In *Proceedings of the 27th SIGGRAPH*.

Bertero, M. and Boccacci, P. (1998). *Introduction to Inverse Problems in Imaging*. CRC Press.

Beyer, H.-G. and Schwefel, H.-P. (2002). Evolution strategies: A comprehensive introduction. *Natural Computing*, 1(1):3–52.

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.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc.

Bleakley, K. and Yamanishi, Y. (2009). Supervised prediction of drug-target interactions using bipartite local models. *Bioinformatics*, 25(18):2397–2403.

Blockeel, H., De Raedt, L., and Ramon, J. (1998). Top-down induction of clustering trees. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 55–63.

Blüthgen, N. (2010). Why network analysis is often disconnected from community ecology: A critique and an ecologist's guide. *Basic and Applied Ecology*, 11(3):185–195.

Blüthgen, N. and K. Fiedler (2004). Preferences for sugars and amino acids and their conditionality in a diverse nectar-feeding ant community. *Journal of Animal Ecology*, 73(1):155–166.

Blüthgen, N., Menzel, F., and Blüthgen, N. (2006). Measuring specialization in species interaction networks. *BMC ecology*, 6(1):1–9.

Blüthgen, N., Stork, N. E., and Fiedler, K. (2004). Bottom-up control and co-occurrence in complex communities: honeydew and nectar determine a rainforest ant mosaic. *Oikos*, 106(2):344–358.

Bock, J. and Gough, D. (2001). Predicting protein-protein interactions from primary structure. *Bioinformatics*, 17(5):455–460.

Bonilla, E. V., Agakov, F. V., and Williams, C. K. I. (2007). Kernel multi-task learning using task-specific features. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, pages 43–50.

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S. V. N., Smola, A. J., and Kriegel, H.-P. (2005). Protein function prediction via graph kernels. *Bioinformatics*, 21 Suppl 1:i47–56.

Brunner, C. and Fischer, A. (2012). Pairwise support vector machines and their application to large scale problems. *Journal of Machine Learning Research*, 13:2279–2292.

Bunke, H. and Shearer, K. (1998). A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3-4):255–259.

Candes, E. and Recht, B. (2008). Exact low-rank matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9:717–772.

Caruana, R. (1997). Multitask learning. *Machine learning*, 75:41–75.

Chacoff, N. P., Vázquez, D. P., Lomáscolo, S. B., Stevani, E. L., Dorado, J., and Padrón, B. (2012). Evaluating sampling completeness in a desert plant-pollinator network. *Journal of Animal Ecology*, 81(1):190–200.

Chalk, A. J., Worth, C. L., Overington, J. P., and Chan, A. W. E. (2004). PDBLIG: Classification of small molecular protein bind-

ing in the protein data bank. *Journal of Medical Chemistry*, 47(15):3807–3816.

Chang, D. T.-H., Syu, Y.-T., and Lin, P.-C. (2010). Predicting the protein-protein interactions using primary structures with predicted protein surface. *BMC bioinformatics*, 11 Suppl 1:S3.

Chapelle, O., Schölkopf, B., and Zien, A. (2006). *Semi-Supervised Learning*. MIT Press.

Chu, W. and Park, S.-T. (2009). Personalized recommendation on dynamic content using predictive bilinear models. In *Proceedings of the 18th International Conference on World Wide Web*, pages 691–700.

Costello, J., Heiser, L., Georgii, E., Gönen, M., Menden, M., Wang, N., Bansal, M., Ammad-ud din, M., Hintsanen, P., Khan, S., Mpindi, J.-P., Kallioniemi, O., Honkela, A., Aittokallio, T., Wennerberg, K., Abbuehl, J.-P., Allen, J., Altman, R., Balcome, S., Battle, A., Bender, A., Berger, B., Bernard, J., Bhattacharjee, M., Bhuvaneshwar, K., Bieberich, A., Boehm, F., Califano, A., Chan, C., Chen, B., Chen, T.-H., Choi, J., Coelho, L., Cokelaer, T., Creighton, C., Cui, J., Dampier, W., Davisson, V., De Baets, B., Deshpande, R., DiCamillo, B., Dundar, M., Duren, Z., Ertel, A., Fan, H., Fang, H., Gauba, R., Gottlieb, A., Grau, M., Gusev, Y., Ha, M., Han, L., Harris, M., Henderson, N., Hejase, H., Homicsko, K., Hou, J., Hwang, W., Ijzerman, A., Karacali, B., Keles, S., Kendziorski, C., Kim, J., Kim, M., Kim, Y., Knowles, D., Koller, D., Lee, J., Lee, J., Lenselink, E., Li, B., Li, B., Li, J., Liang, H., Ma, J., Madhavan, S., Mooney, S., Myers, C., Newton, M., Overington, J., Pal, R., Peng, J., Pestell, R., Prill, R., Qiu, P., Rajwa, B., Sadanandam, A., Sambo, F., Shin, H., Song, J., Song, L., Sridhar, A., **Stock, M.**, Sun, W., Ta, T., Tadesse, M., Tan, M., Tang, H., Theodorescu, D., Toffolo, G., Tozeren, A., Trepicchio, W., Varoquaux, N., Vert, J.-P., Waegeman, W., Walter, T., Wan,

Q., Wang, D., Wang, W., Wang, Y., Wang, Z., Wegner, J., Wu, T., Xia, T., Xiao, G., Xie, Y., Xu, Y., Yang, J., Yuan, Y., Zhang, S., Zhang, X.-S., Zhao, J., Zuo, C., van Vlijmen, H., van Westen, G., Collins, J., Gallahan, D., Singer, D., Saez-Rodriguez, J., Kaski, S., Stolovitzky, J. G., and Stolovitzky, G. (2014). A community effort to assess and improve drug sensitivity prediction algorithms. *Nature Biotechnology*, 32(12):1202–1212.

Daumé, H., Langford, J., and Marcu, D. (2009). Search-based structured prediction. *Machine Learning*, 75(3):297–325.

Davis, M. I., Hunt, J. P., Herrgard, S., Ciceri, P., Wodicka, L. M., Pallares, G., Hocker, M., Treiber, D. K., and Zarrinkar, P. P. (2011). Comprehensive analysis of kinase inhibitor selectivity. *Nature Biotechnology*, 29(11):1046–1051.

De Baets, B., De Meyer, H., De Schuymer, B., and Jenei, S. (2006). Cyclic evaluation of transitivity of reciprocal relations. *Social Choice and Welfare*, 26(2):217–238.

De Clercq, M., **Stock, M.**, De Baets, B., and Waegeman, W. (2015). Data-driven recipe completion using machine learning methods. *Trends in Food Science & Technology*, 49:1–13.

De Paepe, A., Van Peer, G., **Stock, M.**, Volders, P.-J., Vandesompele, J., De Baets, B., and Waegeman, W. (2016). miRNA target prediction through modeling quantitative and qualitative miRNA binding site information in a stacked model structure. *Nucleic Acid Research*, Accepted.

Dechtiar, A. O. (1972). Parasites of fish from Lake of the Woods, Ontario. *Journal of Fisheries Research Board of Canada*, 29:275–283.

Dembczynski, K., Waegeman, W., Cheng, W., and Hüllermeier, E. (2012). On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88(1-2):5–45.

Deng, M., Zhang, K., Mehta, S., Chen, T., and Sun, F. (2003). Prediction of protein function using protein-protein interaction data. *Journal of Computational Biology*, 10:947–960.

Deza, M. M. and Deza, E. (2009). *Encyclopedia of Distances*. Springer, Heidelberg, Germany.

Dhiedt, E., Hoebeke, L., Puynen, S., and Vanwyck, T. (2016). *Hoeveel informatie zit er in een ecologisch netwerk?* Bachelor project, Ghent University.

Ding, H., Takigawa, I., Mamitsuka, H., and Zhu, S. (2013). Similarity-based machine learning methods for predicting drug-target interactions: a brief review. *Briefings in Bioinformatics*, 14(5):734–747.

Drineas, P. and Mahoney, M. (2005). On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175.

Drineas, P. and Mahoney, M. W. (2016). RandNLA: randomized numerical linear algebra. *Communications of the ACM*, 59(6):80–90.

Dunaway-Mariano, D. (2008). Enzyme function discovery. *Structure*, 16(11):1599–1600.

Egelhofer, V., Schomburg, I., and Schomburg, D. (2010). Automatic assignment of EC numbers. *PLoS Computational Biology*, 6:e1000661.

Eklöf, A., Jacob, U., Kopp, J., Bosch, J., Castro-Urgal, R., Chacoff, N. P., Dalsgaard, B., de Sassi, C., Galetti, M., Guimarães, P. R., Lomáscolo, S. B., Martín González, A. M., Pizo, M. A., Rader, R.,

Rodrigo, A., Tylianakis, J. M., Vázquez, D. P., and Allesina, S. (2013). The dimensionality of ecological networks. *Ecology Letters*, 16(5):577–583.

Elkan, C. (2003). Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th International Conference on Machine Learning*, pages 147–153.

Erdin, S., Lisewski, A. M., and Lichtarge, O. (2011). Protein function prediction: towards integration of similarity metrics. *Current Opinion in Structural Biology*, 21(2):180–188.

Evgeniou, T. (2005). Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637.

Evgeniou, T. and Pontil, M. (2004). Regularized multi-task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117.

Fagin, R. (1999). Combining fuzzy information from multiple systems. *Journal of Computer and System Sciences*, 58(1):83–99.

Fagin, R., Lotem, A., and Naor, M. (2003). Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences*, 66(4):614–656.

Fan, W. and Huai, J.-P. (2014). Querying big data: bridging theory and practice. *Journal of Computer Science and Technology*, 29(5):849–869.

Fang, Y. and Si, L. (2011). Matrix co-factorization for recommendation with rich side information and implicit feedback. In *The 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, pages 65–69. ACM.

Faust, K. and Raes, J. (2012). Microbial interactions: from networks to models. *Nature reviews. Microbiology*, 10(8):538–50.

Fawcett, T. (2004). ROC graphs: notes and practical considerations for researchers. Technical report.

Fober, T., Glinca, S., Klebe, G., and Hüllermeier, E. (2011). Superposition and alignment of labeled point clouds. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(6):1653–1666.

Fober, T., Mernberger, M., Klebe, G., and Hüllermeier, E. (2009a). Evolutionary construction of multiple graph alignments for the structural analysis of biomolecules. *Bioinformatics*, 25(16):2110–2117.

Fober, T., Mernberger, M., Klebe, G., and Hüllermeier, E. (2012). Fingerprint kernels for protein structure comparison. *Molecular Informatics*, 31(6-7):443–452.

Fober, T., Mernberger, M., Moritz, R., and Hüllermeier, E. (2009b). Graph-kernels for the comparative analysis of protein active sites. In *German Conference on Bioinformatics*, pages 21 – 31, Halle (Saale), Germany.

Fornasier, M. and Rauhut, H. (2011). Compressive sensing. In *Handbook of Mathematical Methods in Imaging*, pages 187–228.

Friedberg, I. (2006). Automated protein function prediction–the genomic challenge. *Briefings in Bioinformatics*, 7(3):225–42.

Friedman, N., Linial, M., Nachman, I., and Pe'er, D. (2000). Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(1998):601–620.

Fründ, J., McCann, K. S., and Williams, N. M. (2015). Sampling bias is a challenge for quantifying specialization and network structure: lessons from a quantitative niche model. *Oikos*, 125(4):502–513.

Fürnkranz, J., Hüllermeier, E., and Vanderlooy, S. (2009). Binary decomposition methods for multipartite ranking. *Lecture Notes in Computer Science*, 5781:359–374.

Gao, J., Barzel, B., and Barabási, A.-L. (2016). Universal resilience patterns in complex networks. *Nature*, 530(7590):307–312.

Gärtner, T. (2008). *Kernels for Structured Data*. World Scientific, Singapore.

Genest, C. and Favre, A. (2007). Everything you always wanted to know about copula modeling but were afraid to ask. *Journal of Hydrologic Engineering*, 12(4):347–368.

Gerlt, J. A. J. and Babbitt, P. P. C. (2001). Divergent evolution of enzymatic function: mechanistically diverse superfamilies and functionally distinct suprafamilies. *Annual review of biochemistry*, 70:209–246.

Geurts, P., Touleimat, N., Dutreix, M., and D'Alché-Buc, F. (2007). Inferring biological networks with output kernel trees. *BMC Bioinformatics*, 8(2):S4.

Gibson, D. G. (2014). Programming biological operating systems: genome design, assembly and activation. *Nature methods*, 11(5):521–526.

Gibson, M. R., Richardson, D. M., and Pauw, A. (2012). Can floral traits predict an invasive plant's impact on native plant-pollinator communities? *Journal of Ecology*, 100(5):1216–1223.

Gittens, A. and Mahoney, M. W. (2013). Revisiting the Nyström method for improved large-scale machine learning. *Journal of Machine Learning Research*, 28(3):567–575.

Goel, S., Langford, J., and Strehl, A. (2009). Predictive indexing for fast search. In *Advances in Neural Information Processing Systems*, pages 505–512.

Goldwasser, L. and Roughgarden, J. (1997). Sampling effects and the estimation of food-web properties. *Ecology*, 78(1):41–54.

Gönen, M. and Heller, G. (2005). Concordance probability and discriminatory power in proportional hazards regression. *Biometrika*, 92(4):965–970.

Gonnelli, G., **Stock, M.**, Verwaeren, J., Maddelein, D., De Baets, B., Martens, L., and Degroeve, S. (2015). A decoy-free approach to the identification of peptides. *Journal of Proteome Research*, 14(4):1792–1798.

Gonzalez, R. C. and Woods, R. E. (2007). *Digital Image Processing*. Pearson.

Goulson, D. (2016). *A Buzz in the Meadow: The Natural History of a French Farm*. Picador.

Guimerà, R. and Sales-Pardo, M. (2009). Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences of the United States of America*, 106(52):22073–22078.

Hadfield, J. D., Krasnov, B. R., Poulin, R., and Nakagawa, S. (2014). A tale of two phylogenies: comparative analyses of ecological interactions. *The American naturalist*, 183(2):174–87.

Harrison, A., Pearl, F., Sillitoe, I., Slidel, T., Mott, R., Thornton, J., and Orengo, C. (2003). Recognizing the fold of a protein structure. *Bioinformatics*, 19(14):1748–1759.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

Hayashi, K., Takenouchi, T., Tomioka, R., and Kashima, H. (2012). Self-measuring similarity for multi-task Gaussian processes. *Trans-*

actions of the Japanese Society for Artificial Intelligence, 27(3):103–110.

Hendlich, M., Bergner, A., Günther, J., and Klebe, G. (2003). Relibase: design and development of a database for comprehensive analysis of protein-ligand interactions. *Journal of Molecular Biology*, 326(2):607–620.

Herbrich, R., Graepel, T., and Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. In Smola, A., Bartlett, P., Schölkopf, B., and Schuurmans, D., editors, *Advances in Large Margin Classifiers*, pages 115–132. MIT Press.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53.

Hoffmann, B., Zaslavskiy, M., Vert, J.-P., and Stoven, V. (2010). A new protein binding pocket similarity measure based on comparison of clouds of atoms in 3D: application to ligand prediction. *BMC Bioinformatics*, 11(99):99.

Hu, S., Dilcher, D. L., Jarzen, D. M., and Winship Taylor, D. (2008). Early steps of angiosperm pollinator coevolution. *Proceedings of the National Academy of Sciences of the United States of America*, 105(1):240–245.

Huan, J., Bandyopadhyay, D., Wang, W., Snoeyink, J., Prins, J., and Tropsha, A. (1999). Comparing graph representations of protein structure for mining family-specific residue-based packing motifs. *Journal of Computational Biology : a Journal of Computational Molecular Cell Biology*, 12(6):657–671.

Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. (2006). Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501.

Hüllermeier, E. and Fürnkranz, J. (2010). *Preference Learning*. Springer.

Huynh-Thu, V. A., Irrthum, A., Wehenkel, L., and Geurts, P. (2010). Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE*, 5(9):1–10.

Ilyas, I. F., Beskales, G., and Soliman, M. A. (2008). A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys*, 40(4).

Isinkaye, F., Folajimi, Y., and Ojokoh, B. (2015). Recommendation systems: principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273.

Jacob, L., Hoffmann, B., Stoven, V., and Vert, J.-P. (2008). Virtual screening of GPCRs: an in silico chemogenomics approach. *BMC Bioinformatics*, 9(1):1–16.

Jacob, L. and Vert, J.-P. (2008). Protein-ligand interaction prediction: an improved chemogenomics approach. *Bioinformatics*, 24(19):2149–2156.

Jalali, A., Sanghavi, S., Ravikumar, P., and Ruan, C. (2010). A dirty model for multi-task learning. In *Neural Information Processing Symposium*, pages 964–972.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446.

Jayaraman, D. and Grauman, K. (2014). Zero shot recognition with unreliable attributes. In *Advances in Neural Information Processing Systems*, pages 3464–3472.

Joachims, T. (2005). A support vector method for multivariate performance measures. In *Proceedings of the International Conference on Machine Learning, Bonn, Germany*, pages 377–384.

Johnson, R. and Zhang, T. (2008). Graph-based semi-supervised learning and spectral kernel design. *IEEE Transactions on Information Theory*, 54(1):275–288.

Jordano, P. (2016). Sampling networks of ecological interactions. *Functional Ecology*, 30(12):1883–1893.

Jorgensen, W. L. (2004). The many roles of computation in drug discovery. *Science*, 303(5665):1813–1818.

Junker, R. R., Blüthgen, N., Brehm, T., Binkenstein, J., Paulus, J., Martin Schaefer, H., and Stang, M. (2013). Specialization on traits as basis for the niche-breadth of flower visitors and as structuring mechanism of ecological networks. *Functional Ecology*, 27(2):329–341.

Kabsch, W. (1976). A solution of the best rotation to relate two sets of vectors. *Acta Crystallographica*, 32:922–923.

Kadri, H., Ghavamzadeh, M., and Preux, P. (2013). A generalized kernel approach to structured output learning. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1–9.

Kahneman, D. (2011). *Thinking Fast and Slow*. Farrar, Straus and Giroux.

Kakutani, T., Inoue, T., Kato, M., and Ichihashi, H. (1990). Insect-flower relationship in the campus of Kyoto University, Kyoto: An overview of the flowering phenology and the seasonal pattern of insect visits. *Contribution from the Biological Laboratory, Kyoto University*, 27:465–521.

Karaman, M. M. W., Herrgard, S., Treiber, D. K. D., Gallant, P., Atteridge, C. E. C., Campbell, B. B. T., Chan, K. K. W., Ciceri, P., Davis, M. M. I., Edeen, P. P. T., Faraoni, R., Floyd, M., Hunt,

J. J. P., Lockhart, D. J. D., Milanov, Z. Z. V., Morrison, M. M. J., Pallares, G., Patel, H. H. K., Pritchard, S., Wodicka, L. M. L., and Zarrinkar, P. P. (2008). A quantitative analysis of kinase inhibitor selectivity. *Nature Biotechnology*, 26(1):127–132.

Károlyi, G., Neufeld, Z., and Scheuring, I. (2005). Rock-scissors-paper game in a chaotic flow: the effect of dispersion on the cyclic competition of microorganisms. *Journal of Theoretical Biology*, 236(1):12–20.

Kashima, H., Kato, T., Yamanishi, Y., Sugiyama, M., and Tsuda, K. (2009). Link propagation: a fast semi-supervised learning algorithm for link prediction. In *Proceedings of the SIAM International Conference on Data Mining*, pages 1100–1111.

Kashima, H., Oyama, S., Yamanishi, Y., and Tsuda, K. (2010). Cartesian kernel: an efficient alternative to the pairwise kernel. *IEICE Transactions on Information and Systems*, 93(10):2672–2679.

Kerr, B., Riley, M. A., Feldman, M. W., and Bohannan, B. J. M. (2002). Local dispersal promotes biodiversity in a real-life game of rock-paper-scissors. *Nature*, 418(6894):171–4.

Kim, S.-D., Lee, Y.-J., Kang, S. H., Cho, H.-G., and Yoon, S.-M. (2015). Constructing cookery network based on ingredient entropy measure. *Indian Journal of Science and Technology*, 8(23):1–9.

Kinoshita, K., Murakami, Y., and Nakamura, H. (2007). eF-seek: prediction of the functional sites of proteins by searching for similar electrostatic potential and molecular surface shape. *Nucleic Acid Research*, 35(suppl 2):W398–W402.

Kluser, S., Neumann, P., Chauzat, M.-P., Pettis, J. S., Peduzzi, P., Witt, R., Fernandez, N., and Theuri, M. (2010). Global honey bee colony disorders and other threats to insect pollinators. Technical report.

Koenigstein, N., Ram, P., and Shavitt, Y. (2012). Efficient retrieval of recommendations in a matrix factorization framework. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 535–544.

Kotera, M., Okuno, Y., Hattori, M., Goto, S., and Kanehisa, M. (2004). Computational assignment of the EC numbers for genomic-scale analysis of enzymatic reactions. *Journal of the American Chemical Society*, 126(50):16487–98.

Krasnov, B. R., Fortuna, M. A., Mouillot, D., Khokhlova, I. S., Shenbrot, G. I., and Poulin, R. (2012). Phylogenetic signal in module composition and species connectivity in compartmentalized host-parasite networks. *The American Naturalist*, 179(4):501–511.

Kristensen, D. M., Ward, R. M., Lisewski, A. M., Erdin, S., Chen, B. Y., Fofanov, V. Y., Kimmel, M., Kavraki, L. E., and Lichtarge, O. (2008). Prediction of enzyme function based on 3D templates of evolutionarily important amino acids. *BMC Bioinformatics*, 9(17):1–19.

Kuang, R., Weston, J., Noble, W. S., and Leslie, C. (2005). Motif-based protein ranking by network propagation. *Bioinformatics*, 21(19):3711–8.

Lafferty, K. D., Dobson, A. P., and Kuris, A. M. (2006). Parasites dominate food web links. *Proceedings of the National Academy of Sciences*, 103(30):11211–6.

Laird, R. a. and Schamp, B. S. (2006). Competitive intransitivity promotes species coexistence. *The American naturalist*, 168(2):182–193.

Lampert, C. H., Nickisch, H., and Harmeling, S. (2014). Attribute-based classification for zero-shot visual object categorization.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(3):453–465.

Larochelle, H., Erhan, D., and Bengio, Y. (2008). Zero-data learning of new tasks. In *Proceedings of the 23rd National Conference on Artificial Intelligence*, pages 646–651.

Lasko, T. A., Bhagwat, J. G., Zou, K. H., and Ohno-Machado, L. (2005). The use of receiver operating characteristic curves in biomedical informatics. *Journal of Biomedical Informatics*, 38(5):404–415.

Laskowski, R., Watson, J., and Thornton, J. (2005). Profunc: a server for predicting protein function from 3d stucture. *NAR*, 33:W89–W93.

Laskowski, R. A., Luscombe, N. M., Swindells, M. B., and Thornton, J. M. (1996). Protein clefts in molecular recognition and function. *Protein Science*, 5(12):2438–2452.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. J. (2006). A tutorial on energy-based learning. In *Predicting Structured Data*, pages 191–246. The MIT Press.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Lee, J., Sun, M., and Lebanon, G. (2011). A comparative study of collaborative filtering algorithms. *ACM Transactions on the Web*, 5(1):1–27.

Lei, B. R. and Olival, K. J. (2014). Contrasting patterns in mammal-bacteria coevolution: Bartonella and Leptospira in bats and rodents. *PLoS Neglected Tropical Diseases*, 8(3):1–11.

Leslie, C., Eskin, E., and Noble, W. S. (2002). The spectrum kernel: a string kernel for SVM protein classification. In *Pacific Symposium on Biocomputing*, pages 566–575.

Leslie, C. S., Eskin, E., Cohen, A., Weston, J., and Noble, W. S. (2004). Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–76.

Letovsky, S. and Kasif, S. (2003). Predicting protein function from protein/protein interaction data: a probabilistic approach. *Bioinformatics*, 19:i197–i204.

Lin, H. W. and Tegmark, M. (2016). Why does deep and cheap learning work so well?

Liu, H. and Yang, Y. (2015). Bipartite edge prediction via transductive learning over product graphs. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 1880–1888.

Lo Gerfo, L., Rosasco, L., Odone, F., De Vito, E., and Verri, A. (2008). Spectral algorithms for supervised learning. *Neural Computation*, 20(7):1873–1897.

Lopez-Paz, D., Sra, S., Smola, A. J., Ghahramani, Z., and Schölkopf, B. (2014). Randomized nonlinear component analysis. In *Proceedings of the 31th International Conference on Machine Learning*, volume 31, pages 1359–1367.

Lü, L., Medo, M., Yeung, C. H., Zhang, Y.-C., Zhang, Z.-K., and Zhou, T. (2012). Recommender systems. *Physics Reports*, 519(1):1–49.

Luce, R. and Suppes, P. (1965). *Handbook of Mathematical Psychology*. Wiley.

Macarron, R., Banks, M. N., Bojanic, D., Burns, D. J., Cirovic, D. A., Garyantes, T., Green, D. V. S., Hertzberg, R. P., Janzen, W. P., Paslay, J. W., and Schopfer, U. (2011). Impact of high-throughput screening. *Nature Reviews Drug Discovery*, 10(3):188–195.

MacKay, D. J. (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

Maetschke, S. R., Madhamshettiwar, P. B., Davis, M. J., and Ragan, M. A. (2014). Supervised, semi-supervised and unsupervised inference of gene regulatory networks. *Briefings in Bioinformatics*, 15(2):195–211.

Mahé, P., Ueda, N., Akutsu, T., Perret, J.-L., and Vert, J.-P. (2005). Graph kernels for molecular structure-activity relationship analysis with support vector machines. *Journal of Chemical Information and Modeling*, 45(4):939–951.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Marbach, D., Costello, J., Küffner, R., and Vega, N. (2012). Wisdom of crowds for robust gene network inference. *Nature Methods*, 9(8):796–807.

Marcotte, E. M. (1999). Detecting protein function and protein-protein interactions from genome sequences. *Science*, 285(5428):751–753.

Martin, A. C., Orengo, C. A., Hutchinson, E. G., Jones, S., Karmirantzou, M., Laskowski, R. A., Mitchell, J. B., Taroni, C., and Thornton, J. M. (1998). Protein folds and functions. *Structure*, 6(7):875–84.

Martin, C. D. and Van Loan, C. F. (2006). Shifted Kronecker product systems. *SIAM Journal on Matrix Analysis and Applications*, 29(1):184–198.

Mazumder, R., Hastie, T., and Tibshirani, R. (2010). Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287–2322.

Menon, A. and Elkan, C. (2011). Link prediction via matrix factorization. *Machine Learning and Knowledge Discovery in Databases*, 6912:437–452.

Menon, A. K. and Elkan, C. (2010). A log-linear model with latent features for dyadic prediction. In *Proceedings of the 10th International Conference on Data Mining*, pages 364–373.

Mineiro, P. and Karampatziakis, N. (2014). Fast label embeddings for extremely large output spaces. *CoRR*, abs/1412.6.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. a., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Morales-Castilla, I., Matias, M. G., Gravel, D., and Araújo, M. B. (2015). Inferring biotic interactions from proxies. *Trends in Ecology and Evolution*, 30(6):347–356.

Stock, M., Dembczynski, K., De Baets, B., and Waegeman, W. (2016). Exact and efficient top-K inference for multi-target prediction by querying separable linear relational models. *Data Mining and Knowledge Discovery*, 30(5):1370–1394.

Stock, M., Fober, T., Hüllermeier, E., Glinca, S., Klebe, G., Pahikkala, T., Airola, A., De Baets, B., and Waegeman, W. (2014). Identification of functionally related enzymes by learning-to-rank methods. *IEEE Transactions on Computational Biology and Bioinformatics*, 11(6):1157–1169.

Stock, M., Hoefman, S., Kerckhof, F.-M., Boon, N., De Vos, P., De Baets, B., Heylen, K., and Waegeman, W. (2013). Exploration and

prediction of interactions between methanotrophs and heterotrophs. *Research in Microbiology*, 164(10):1045–1054.

Stock, M., Poisot, T., Waegeman, W., and De Baets, B. (2017). Linear filtering reveals false negatives in species interaction data. *Scientific Reports*, 7(45908):1–8.

Neuhaus, M. and Bunke, H. (2007). *Bridging the Gap between Graph Edit Distance and Kernel Machines*. World Scientific, New Jersey.

Nicolaï, N., De Leersnyder, F., Copot, D., **Stock, M.**, Ionescu, C., Gernaey, K., Nopens, I., and De Beer, T. (2017). Development of a PAT-based advanced process control for continuous twin-screw wet granulation. *AIChE Journal*, Under revi.

Olesen, J. M., Bascompte, J., Dupont, Y. L., Elberling, H., Rasmussen, C., and Jordano, P. (2011). Missing and forbidden links in mutualistic networks. *Proceedings. Biological sciences / The Royal Society*, 278(1706):725–732.

Olesen, J. M., Bascompte, J., Dupont, Y. L., and Jordano, P. (2007). The modularity of pollination networks. *Proceedings of the National Academy of Sciences of the United States of America*, 104(50):19891–19896.

Oliver, T. H., Heard, M. S., Isaac, N. J., Roy, D. B., Procter, D., Eigenbrod, F., Freckleton, R., Hector, A., Orme, C. D. L., Petchey, O. L., Proença, V., Raffaelli, D., Suttle, K. B., Mace, G. M., Martín-López, B., Woodcock, B. A., and Bullock, J. M. (2015). Biodiversity and resilience of ecosystem functions. *Trends in Ecology & Evolution*, 30(11):673–684.

Omohundro, S. M. (1989). Five balltree construction algorithms. *Science*, 51:1–22.

Osadchy, M. and Kolodny, R. (2011). Maps of protein structure space reveal a fundamental relationship between protein structure and

function. *Proceedings of the National Academy of Sciences of the United States of America*, 108(30):12301–12306.

Oyama, S. and Manning, C. (2004). Using feature conjunctions across examples for learning pairwise classifiers. In Boulicaut, J.-F., Esposito, F., Giannotti, F., and Pedreschi, D., editors, *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, volume 3201 of *Lecture Notes in Computer Science*, pages 322–333. Springer.

Pahikkala, T., Airola, A., **Stock, M.**, De Baets, B., and Waegeman, W. (2013). Efficient regularized least-squares algorithms for conditional ranking on relational data. *Machine Learning*, 93(2-3):321–356.

Pahikkala, T., Airola, A., Pietila, S., Shakyawar, S., Szwajda, A., Tang, J., and Aittokallio, T. (2015). Toward more realistic drug-target interaction predictions. *Briefings in Bioinformatics*, 16(2):325–337.

Pahikkala, T., Boberg, J., and Salakoski, T. (2006). Fast n-fold cross-validation for regularized least-squares. In *Proceedings of the Nineth Scandinavian Conference on Artificial Intelligence*, pages 83–90.

Pahikkala, T., **Stock, M.**, Airola, A., Aittokallio, T., De Baets, B., and Waegeman, W. (2014). A two-step learning approach for solving full and almost full cold start problems in dyadic prediction. *Lecture Notes in Computer Science*, 8725:517–532.

Pahikkala, T., Tsivtsivadze, E., Airola, A., Järvinen, J., and Boberg, J. (2009). An efficient algorithm for learning to rank from preference graphs. *Machine Learning*, 75(1):129–165.

Pahikkala, T., Waegeman, W., Tsivtsivadze, E., Salakoski, T., and De Baets, B. (2010). Learning intransitive reciprocal relations

with kernel methods. *European Journal of Operational Research*, 206(3):676–685.

Palatucci, M., Hinton, G., Pomerleau, D., and Mitchell, T. M. (2009). Zero-shot learning with semantic output codes. In *Advances in Neural Information Processing Systems*, pages 1410–1418.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Park, S.-T. and Chu, W. (2009). Pairwise preference regression for cold-start recommendation. In *Proceedings of the Third ACM Conference on Recommender Systems*, pages 21–28. ACM.

Park, Y. and Marcotte, E. M. (2012). Flaws in evaluation schemes for pair-input computational predictions. *Nature Methods*, 9(12):1134–1136.

Partalas, I., Kosmopoulos, A., Baskiotis, N., Artieres, T., Paliouras, G., Gaussier, E., Androutsopoulos, I., Amini, M.-R., and Galinari, P. (2015). LSHTC: a benchmark for large-scale text classification. *rXivpreprint arXiv:1503.08581*.

Pazos, F. and Valencia, A. (2001). Similarity of phylogenetic trees as indicator of protein-protein interaction. *Protein Engineering*, 14(9):609–614.

Pazos, F. and Valencia, A. (2002). In silico two-hybrid system for the selection of physically interacting protein pairs. *Proteins: Structure, Function, and Genetics*, 47(2):219–227.

Pelossof, R., Singh, I., Yang, J. L., Weirauch, M. T., Hughes, T. R., and Leslie, C. S. (2015). Affinity regression predicts the recognition code of nucleic acid-binding proteins. *Nature Biotechnology*, 33(12):1242–1249.

Peralta, G. (2016). Merging evolutionary history into species interaction networks. *Functional Ecology*, 30(12):1917–1925.

Pérot, S., Sperandio, O., Miteva, M. A., Camproux, A.-C., and Villoutreix, B. O. (2010). Druggable pockets and binding site centric chemical space: a paradigm shift in drug discovery. *Drug Discovery Today*, 15(15-16):656–667.

Pitre, S., North, C., Alamgir, M., Jessulat, M., Chan, A., Luo, X., Green, J. R., Dumontier, M., Dehne, F., and Golshani, A. (2008). Global investigation of protein-protein interactions in yeast Saccharomyces cerevisiae using re-occurring short polypeptide sequences. *Nucleic acids research*, 36(13):4286–4294.

Poisot, T., Baiser, B., Dunne, J. A., Kéfi, S., Massol, F., Mouquet, N., Romanuk, T. N., Stouffer, D. B., Wood, S. A., and Gravel, D. (2015). Mangal - making ecological network analysis simple. *Ecography*, 38:1–7.

Poisot, T., Stouffer, D. B., and Kéfi, S. (2016). Describe, understand and predict: why do we need networks in ecology? *Functional Ecology*, 30(12):1878–1882.

Poulin, R., Krasnov, B. R., Mouillot, D., and Thieltges, D. W. (2011). The comparative ecology and biogeography of parasites. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 366(1576):2379–2390.

Powers, R., Copeland, J. C., Germer, K., Mercier, K. A., Ramanathan, V., and Revesz, P. (2006). Comparison of protein active site structures for functional annotation of proteins and drug design. *Proteins: Structure, Function, and Bioinformatics*, 65:124–135.

Prinz, F., Schlange, T., and Asadullah, K. (2011). Believe it or not: how much can we rely on published data on potential drug targets? *Nature reviews. Drug discovery*, 10(9):712–713.

Rafferty, N. E. and Ives, A. R. (2013). Phylogenetic trait-based analyses of ecological networks. *Ecology*, 94(10):2321–2333.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Rathke, F., Hansen, K., Brefeld, U., and Müller, K.-R. (2010). StructRank: a new approach for ligand-based virtual screening. *Journal of Chemical Information and Modeling*, 51(1):83–92.

Raymond, R. and Kashima, H. (2010). Fast and scalable algorithms for semi-supervised link prediction on static and dynamic graphs. In Balcázar, J. L., Bonchi, F., Gionis, A., and Sebag, M., editors, *Proceedings of the European Conference on Machine learning and Knowledge Discovery in Databases*, volume 6323 of *Lecture Notes in Computer Science*, pages 131–147. Springer.

Ricci, E., De Bie, T., and Cristianini, N. (2008). Magic moments for structured output prediction. *Journal of Machine Learning Research*, 9:2803–2846.

Rifkin, R. and Lippert, R. (2007a). Notes on regularized least squares. Technical Report MIT-CSAIL-TR-2007-025, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA.

Rifkin, R. and Lippert, R. (2007b). Value regularization and Fenchel duality. *Journal of Machine Learning Research*, 8:441–479.

Rohrbach, M., Stark, M., and Schiele, B. (2011). Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1641–1648.

Romera-Paredes, B. and Torr, P. (2015). An embarrassingly simple approach to zero-shot learning. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 2152–2161.

Rost, B. (2002). Enzyme function less conserved than anticipated. *Journal of Molecular Biology*, 318(2):595–608.

Rousu, J., Saunders, C., Szedmak, S., and Shawe-Taylor, J. (2006). Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7:1601–1626.

Sanfeliu, A. and Fu, K. (1983). A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 13(3):353–362.

Sarwar, B., Karypis, G., Konstan, J., and Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International Conference on World Wide Web - WWW '01*, pages 285–295, New York, New York, USA. ACM Press.

Schmitt, S., Kuhn, D., and Klebe, G. (2002). A new method to detect related function among proteins independent of sequence and fold homology. *Journal of Molecular Biology*, 323(2):387–406.

Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels*. The MIT Press.

Schölkopf, B., Tsuda, K., and Vert, J.-P. (2004). *Kernel Methods in Computational Biology*. The MIT Press.

Schrynemackers, M., Küffner, R., and Geurts, P. (2013). On protocols and measures for the validation of supervised methods for the inference of biological networks. *Frontiers in Genetics*, 4(262):1–16.

Schrynemackers, M., Wehenkel, L., Babu, M. M., and Geurts, P. (2015). Classifying pairs with trees for supervised biological network inference. *Molecular Biosystems*, 11(8):2116–2125.

Schwikowski, B., Uetz, P., and Fields, S. (2000). A network of protein-protein interactions in yeast. *Nature Biotechnology*, 18:3548–3557.

Shan, H. and Banerjee, A. (2010). Probabilistic matrix factorization for collaborative filtering. In *Proceedings of the 10th IEEE International Conference on Data Mining*, pages 1025–1030.

Shatsky, M. (2006). *The Common Point Set Problem with Applications to Protein Structure Analysis*. PhD thesis, Tel Aviv University, Tel Aviv, Israel.

Shatsky, M., Niussinov, R., and Wolfson, H. J. (2004). A method for simultaneous alignment of multiple protein structures. *Proteins: Structure, Function, and Bioinformatics*, 56:143–156.

Shatsky, M., Shulman-Peleg, A., Nussinov, R., and Wolfson, H. J. (2006). The multiple common point set problem and its application to molecule binding pattern detection. *Journal of Computational Biology*, 13(2):407–428.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Shervashidze, N. (2009). Efficient graphlet kernels for large graph comparison. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 5, pages 488–495.

Shimizu, A., Dohzono, I., Nakaji, M., Roff, D. A., Miller, D. G., Osato, S., Yajima, T., Niitsu, S., Utsugi, N., Sugawara, T., and Yoshimura, J. (2014). Fine-tuned bee-flower coevolutionary state hidden within multiple pollination interactions. *Scientific Reports*, 4(3988):1–9.

Shrivastava, A. and Li, P. (2014). Asymmetric lsh (ALSH) for sublinear time maximum inner product search (MIPS). In *Advances in Neural Information Processing Systems*, pages 2321–2329.

Shrivastava, A. and Li, P. (2015). Improved asymmetric locality sensitive hashing (ALSH) for maximum inner product search (MIPS).

In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Sinervo, B. and Lively, C. M. (1996). The rock-paper-scissors game and the evolution of alternative male strategies. *Nature*, 380(6571):240–243.

Singhal, A. (2001). Modern information retrieval: a brief overview. *Bulletin of the IEEE Computer Society Technical Committee of Data Engineering*, 24:1–9.

Smith, T. and Waterman, M. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197.

Smola, A., Gretton, A., Song, L., and Schölkopf, B. (2007). A Hilbert space embedding for distributions. In *The 18th International Conference of Algorithmic Learning Theory*, pages 13–31.

Snoek, J., Larochelle, H., and Adams, R. (2012). Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951—-2959.

Sokolov, A. and Ben-Hur, A. (2008). A structured-outputs method for prediction of protein function. In *Proceedings of the third International Workshop on Machine Learning in Systems Biology*.

Song, L., Fukumizu, K., and Gretton, A. (2013). Kernel embeddings of conditional distributions. *IEEE Signal Processing Magazine*, 30(4):98–111.

Staniczenko, P. P. P. A., Kopp, J. C. J., and Allesina, S. (2013). The ghost of nestedness in ecological networks. *Nature Communications*, 4:1391.

Stein, R. R., Marks, D. S., and Sander, C. (2015). Inferring pairwise interactions from biological data using maximum-entropy probability models. *PLoS Computational Biology*, 11(7):1–22.

Steinwart, I. (2002). On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93.

Strogatz, S. H. (2001). Exploring complex networks. *Nature*, 410(6825):268–276.

Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:1–19.

Sun, M., Wang, X., Zou, C., He, Z., Liu, W., and Li, H. (2016). Accurate prediction of RNA-binding protein residues with two discriminative structural descriptors. *BMC Bioinformatics*, 17(231):1–14.

Swamidass, S. J., Azencott, C.-A., Daily, K., and Baldi, P. (2010). A CROC stronger than ROC: measuring, visualizing and optimizing early retrieval. *Bioinformatics*, 26(10):1348–1356.

Takács, G., Pilászy, I., Németh, B., and Tikk, D. (2008). Matrix factorization and neighbor based algorithms for the netflix prize problem. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 267–274, New York, New York, USA. ACM Press.

Thompson, J. (1999). The evolution of species interactions. *Science*, 284(5423):2116–2118.

Thornton, J., Todd, A., Milburn, D., Borkakoti, N., and Orengo, C. (2000). From structure to function: approaches and limitations. *Nature Structural Biology*, 7 Suppl:991–994.

Tipping, M. and Bishop, C. (1997). Probabilistic principal component analysis. *Journal of the Royal Statistical Society*, 3:611–622.

Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: an overview. *International Journal of Data Warehousing & Mining*, 3(3):1–13.

Vacic, V., ans S. Lonardi, L. I., and Radivojac, P. (2010). Graphlet kernels for prediction of functional residues in protein structures. *Journal of Computational Biology*, 17(1):55–72.

van Laarhoven, T., Nabuurs, S. B., and Marchiori, E. (2011). Gaussian interaction profile kernels for predicting drug-target interaction. *Bioinformatics*, 27(21):3036–3043.

Van Loan, C. F. (2000). The ubiquitous Kronecker product. *Journal of Computational and Applied Mathematics*, 123(1-2):85–100.

Vapnik, V. (2000). *The Nature of Statistical Learning Theory*. Springer.

Varma, S. and Simon, R. (2006). Bias in error estimation when using cross-validation for model selection. *BMC Bioinformatics*, 7(91).

Vazquez, A., Flammini, A., Maritan, A., and Vespignani, A. (2003). Global protein function prediction from protein-protein interaction networks. *Bioinformatics*, 21:ii59–ii65.

Vert, J.-P. (2008). Reconstruction of biological networks by supervised machine learning approaches. In *Elements of Computional Systems Biology*, pages 165–188.

Vert, J.-P., Qiu, J., and Noble, W. S. (2007). A new pairwise kernel for biological network inference with support vector machines. *BMC Bioinformatics*, 8(S-10):1–10.

Voet, D. and Voet, J. G. (2010). *Biochemistry*. Wiley, 4th edition.

Volinsky, C. (2009). Matrix factorization techniques for recommender systems. volume 42, pages 30–37.

Waegeman, W., De Baets, B., and Boullart, L. (2008). Learning layered ranking functions with structured support vector machines. *Neural Networks*, 21(10):1511–1523.

Waegeman, W., Pahikkala, T., Airola, A., Salakoski, T., **Stock, M.**, and De Baets, B. (2012). A kernel-based framework for learning graded relations from data. *IEEE Transactions on Fuzzy Systems*, 20(6):1090–1101.

Wahba, G. (1990). *Spline Models for Observational Data*. SIAM.

Wang, G. and Dunbrack, R. L. (2003). PISCES: a protein sequence culling server. *Bioinformatics*, 19(12):1589–1591.

Wang, S., Sun, S., and Xu, J. (2016). AUC-maximized deep convolutional neural fields for sequence Labeling. In *Proceedings of the European Conference, ECML PKDD*, pages 1–17.

Weber, A., Casini, A., Heine, A., Kuhn, D., Supuran, C. T., Scozzafava, A., and Klebe, G. (2004). Unexpected nanomolar inhibition of carbonic anhydrase by COX-2-selective celecoxib: new pharmacological opportunities due to related binding site recognition. *Journal of Medical Chemistry*, 47(3):550–557.

Weill, N. and Rognan, D. (2010). Alignment-free ultra-high-throughput comparison of druggable protein-ligand binding sites. *Journal of Chemical Information and Modeling*, 50(1):123–135.

Weisel, M., Kriegl, J. M., and Schneider, G. (2010). Architectural repertoire of ligand-binding pockets on protein surfaces. *Chembiochem*, 11(4):556–563.

Weskamp, N., Hüllermeier, E., Kuhn, D., and Klebe, G. (2007). Multiple graph alignment for the structural analysis of protein active sites. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2):310–20.

Weston, J., Eliseeff, A., Zhou, D., Leslie, C., and Noble, W. S. (2004). Protein ranking: from local to global structure in the protein similarity network. *Proceedings of the National Academy of Science*, 101:6559–6563.

Weston, J., Schölkopf, B., and Bousquet, O. (2007). Joint Kernel Maps. In *Predicting Structured Data*, pages 176–191. The MIT Press.

Wootton, J. T. and Emmerson, M. (2005). Measurement of interaction strength in nature. *Annual Review of Ecology, Evolution, and Systematics*, 36(1):419–444.

wyffels, F. and Schrauwen, B. (2010). A comparative study of reservoir computing strategies for monthly time series prediction. *Neurocomputing*, 73(10-12):1958–1964.

Xia, Z., Wu, L.-Y., Zhou, X., and Wong, S. T. C. (2010). Semi-supervised drug-protein interaction prediction from heterogeneous biological spaces. *BMC Systems Biology*, 4(S6):1–12.

Yamanishi, Y., Araki, M., Gutteridge, A., Honda, W., and Kanehisa, M. (2008). Prediction of drug-target interaction networks from the integration of chemical and genomic spaces. *Bioinformatics*, 24(13):i232–i240.

Yamanishi, Y., Vert, J.-P., and Kanehisa, M. (2004). Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, 20(Suppl 1):i363–370.

Zachariah, D. and Sundin, M. (2012). Alternating least-squares for low-rank matrix reconstruction. *IEEE Signal Processing Letters*, 19(4):231–234.

Zeng, W., Zeng, A., Liu, H., Shang, M., and Zhou, T. (2014). Uncovering the information core in recommender systems. *Scientific Reports*, 4(6140):1–14.

Zhang, S., Wang, W., Ford, J., Makedon, F., and Pearlman, J. (2005). Using singular value decomposition approximation for collaborative filtering. In *Proceedings of the Seventh IEEE International Conference on E-Commerce Technology*.

Zhou, T., Kuscsik, Z., Liu, J.-G., Medo, M., Wakeling, J. R., Zhang, Y.-C., and Settings, M. (2010). Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences of the United States of America*, 107(10):4511–4515.

Zhou, T., Shan, H., Banerjee, A., and Sapiro, G. (2012). Kernelized probabilistic matrix factorization: exploiting graphs and side information. In *Proceedings of the 12th SIAM International Conference on Data Mining*, pages 403–414. SIAM.

Zobel, J. and Moffat, A. (2006). Inverted files for text search engines. *ACM Computing Surveys*, 38(2).

Zou, H., Hastie, T., Tibshirani, R., Johnstone, I., and Lu, A. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):262–286.

A

Properties of matrices

Matrices are among the most important mathematical structures. A good grasp on matrix algebra is indispensable for understanding machine learning methods (or, arguably, any quantitative discipline). Since this PhD dissertation is specifically concerned with learning matrix-valued data (as opposed to vector-valued data in more traditional settings), I reserved some space here to clarify some essential notations and properties of matrices.

Throughout this dissertation, matrices are denoted as an upper case boldface letter, e.g. $\mathbf{A} = [a_{ij}]$. The element at the i-th row and the j-th column is given by $\mathbf{A}_{ij}$, while the i-th row vector, resp. j-th column vector, is given by $\mathbf{A}_{i.}$, resp. $\mathbf{A}_{.j}$. Note that in some texts a matrix is often denoted by a non-boldface upper case letter. I chose to represent all multi-dimensional objects in boldface, freeing the non-boldface upper case letters for scalars.

I use $\mathbb{I}_n$ as the symbol for the $n \times n$ identity matrix, though the subscript will be dropped if clear from context. Similarly, the symbol $\mathbb{J}_{nm}$ is used to denote an $n \times m$ matrix filled with ones. The transpose of a matrix $\mathbf{A}$ is indicated by $\mathbf{A}^\top$. A symmetric matrix is a matrix for which $\mathbf{A}^\top = \mathbf{A}$.

In machine learning and mathematical optimization, one often works with positive-definite matrices, defined below.

DEFINITION A.1 (Positive-definite matrix). *A positive-definite matrix* $\mathbf{P}$ *is a symmetric* $n \times n$ *real matrix* $\mathbf{P}$ *for which it holds that*

$$\mathbf{z}^\top \mathbf{P}\mathbf{z} > 0\,,$$

for any non-zero column vector $\mathbf{z}$ *of real numbers.*

The inverse of a square matrix $\mathbf{A}$, denoted $\mathbf{A}^{-1}$, satisfies

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbb{I}\,.$$

Furthermore, because $\mathbf{A}\mathbf{B}^{-1}\mathbf{B}^{-1}\mathbf{B} = \mathbb{I}$, it holds that

$$\mathbf{A}\mathbf{B}^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}\,.$$

Also we have

$$(\mathbf{A}^\top)^{-1} = (\mathbf{A}^{-1})^\top\,.$$

The inversion of matrices is in general a computationally expensive operation, typical with a time complexity of $\mathcal{O}(n^3)$. There are some useful properties for cheaply computing the inverses of matrices with a special structure. Below is the Woodbury matrix identity.

PROPERTY A.1 (Woodbury matrix identity). *For conformable matrices, it holds that*

$$(\mathbf{A} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} + \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1}\,.$$

This is useful in the common case when $\mathbf{A}^{-1}$ is known or easy to compute (e.g. $\mathbf{A}$ is diagonal) and $\mathbf{B}$ and $\mathbf{C}$ have much more rows than columns. A special case of the Woodbury matrix identity is the Sherman-Morrison formula.

PROPERTY A.2 (Sherman-Morrison formula). *It holds that*

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^\top)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^\top\mathbf{A}^{-1}}{1 + \mathbf{v}^\top\mathbf{A}^{-1}\mathbf{u}}\,,$$

for any invertible matrix $\mathbf{A}$ *and any two conformable column vectors* $\mathbf{u}$ *and* $\mathbf{v}$*.*

Another useful property is the matrix inversion lemma, given below.

PROPERTY A.3 (Matrix inversion lemma). *For conformable matrices, it holds that*

$$(\mathbf{P}^{-1} + \mathbf{B}^\top\mathbf{R}^{-1}\mathbf{B})^{-1}\mathbf{B}^\top\mathbf{R}^{-1} = \mathbf{P}\mathbf{B}^\top(\mathbf{B}\mathbf{P}\mathbf{B}^\top + \mathbf{R})^{-1}\,.$$

Matrices are often analyzed by computing their eigenvalues and eigenvectors. For an $n \times n$ matrix $\mathbf{A}$, the eigenvector equation is given by

$$\mathbf{A}\mathbf{v}_i = \sigma_i\mathbf{v}_i\,,$$

for $i = 1, \ldots, n$, where $\mathbf{v}_i$ is an eigenvector and σ_i the corresponding eigenvalue. It is easy to show that if a matrix is symmetric, the eigenvalues are all real-valued and if the matrix is symmetric and positive-definite, all the eigenvalues are real-valued and positive. Furthermore, a symmetric $n \times n$ matrix $\mathbf{A}$ can be written as

$$\begin{aligned}\mathbf{A} &= \sum_{i=1}^{n} \sigma_i\mathbf{v}_i^\top\mathbf{v}_i \\ &= \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top\,,\end{aligned} \tag{A.1}$$

with $\mathbf{U}$ an $n \times n$ matrix with all the eigenvectors as column vectors and $\mathbf{\Lambda}$ an $n \times n$ diagonal matrix with the corresponding eigen values on the diagonal. Eq. (A.1) is called the eigenvalue decomposition.

An important result about eigenvalues of matrices is given by the Poincaré separation theorem.

THEOREM A.1 (Poincaré separation theorem). *Let* $\mathbf{A}$ *be an* $n \times n$ *matrix and let* $\mathbf{U}$ *be an* $n \times k$ *matrix, with* $k \leq n$*), such that* $\mathbf{U}^\top \mathbf{U} = \mathbb{I}$*. Then,*

$$\sigma_j(\mathbf{U}^\top \mathbf{A}\mathbf{U}) \leq \sigma_j(\mathbf{A})\,,$$

with equality if the columns of $\mathbf{U}$ *are the first* k *eigenvectors of* $\mathbf{A}$*.*

COROLLARY A.1.1. *Let* $\mathbf{A}$ *be an* $n \times n$ *matrix and let* $\mathbf{B}$ *be an* $k \times k$ *principal submatrix of* $\mathbf{A}$*, obtained by deleting* $n - k$ *rows and the corresponding* $n - k$ *columns. Then,*

$$\sigma_j(\mathbf{B}) \leq \sigma_j(\mathbf{A})\,,$$

Proof. This follows directly from Theorem A.1, by choosing an appropriate $\mathbf{U}$ that indexes the selected columns and rows. □

Furthermore, we can also show that the product of two symmetric positive definite matrices has real-valued positive eigenvalues.

PROPERTY A.4. *Consider two symmetric positive-definite matrices* $\mathbf{A}$ *and* $\mathbf{B}$*. The product* $\mathbf{AB}$ *has real-values positive eigenvalues and is diagonalizable.*

Proof. We can show this easily by using the square root of $\mathbf{B}$ (i.e. this means that $\mathbf{B}^{1/2}\mathbf{B}^{1/2} = \mathbf{B}$). Since $\mathbf{B}$ is positive-definite, $\mathbf{B}^{1/2}$ is positive-definite as well. We have

$$\begin{aligned}\mathbf{AB} &= \mathbf{A}\mathbf{B}^{1/2}\mathbf{B}^{1/2} \\ &= \mathbf{B}^{-1/2}(\mathbf{B}^{1/2}\mathbf{A}\mathbf{B}^{1/2})\mathbf{B}^{1/2}\,.\end{aligned}$$

It follows that $\mathbf{AB}$ has the same eigenvalues as $\mathbf{B}^{1/2}\mathbf{AB}^{1/2}$; these eigenvalues are all real-valued and positive. Since $\mathbf{B}^{1/2}\mathbf{AB}^{1/2}$ is diagonalizable, $\mathbf{AB}$ is as well. □

Note that $\mathbf{AB}$ is not necessarily symmetric.

The trick of pairwise learning is transforming a matrix in a vector. This can be done by the vectorization operation.

DEFINITION A.2 (Vectorization). *The vectorization operator* $\text{vec}(\cdot)$ *is a linear operator that transforms an* $n \times m$ *matrix* $\mathbf{A}$ *in a column vector of length* nm *by stacking each of the rows of* $\mathbf{A}$ *on top of each other.*

Furthermore, the Kronecker product is defined as follows.

DEFINITION A.3 (Kronecker product). *If* $\mathbf{A} = [a_{ij}]$ *is an* $n \times m$ *matrix and* $\mathbf{B} = [ij]$ *is an* $p \times q$ *matrix, then the Kronecker product* $\mathbf{A} \otimes \mathbf{B}$ *is the* $mp \times nq$ *block matrix:*

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \dots & a_{1m}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{n1}\mathbf{B} & \dots & a_{nm}\mathbf{B} \end{bmatrix} .$$

For example, if

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \qquad \text{and} \qquad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} ,$$

then

$$\text{vec}(\mathbf{A}) = \begin{bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \end{bmatrix} \qquad \text{and} \qquad \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{bmatrix}$$

The relation between vectorization and the Kronecker product is given by the following property.

PROPERTY A.5. *For any conformable matrices* $\mathbf{N}, \mathbf{M}$ *and* $\mathbf{X}$*, it holds that*

$$(\mathbf{N}^\top \otimes \mathbf{M})\mathrm{vec}(\mathbf{X}) = \mathrm{vec}(\mathbf{MXN})\,.$$

Computing the Kronecker product between two reasonably large matrices results in a huge matrix, often too large to fit in computer memory. If the Kronecker product is only needed in an intermediary step, the above identity can be used to dramatically reduce computation time and memory requirement.

Using the eigenvalue decomposition of matrices, a large system of equations using the Kronecker product can be solved efficiently.

PROPERTY A.6. *(Pahikkala et al., 2013) Let* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ *be diagonalizable matrices, that is, the matrices can be eigen decomposed as*

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1} \text{ and } \mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{-1}\,,$$

where $\mathbf{V}, \mathbf{U} \in \mathbb{R}^{n\times n}$ *contain the eigenvectors and the diagonal matrices* $\mathbf{\Lambda}, \mathbf{\Sigma} \in \mathbb{R}^{n\times n}$ *contain the corresponding eigenvalues of* $\mathbf{A}$ *and* $\mathbf{B}$*. Then, the following type of shifted Kronecker product system*

$$(\mathbf{A} \otimes \mathbf{B} + \lambda\mathbb{I})\mathbf{a} = \mathrm{vec}(\mathbf{Y})\,, \tag{A.2}$$

where $\lambda > 0$ *and* $\mathbf{Y} \in \mathbb{R}^{n\times n}$*, can be solved with respect to* $\mathbf{a}$ *in* $\mathcal{O}(n^3)$ *time if the inverse of* $(\mathbf{A} \otimes \mathbf{B} + \lambda\mathbb{I})$ *exists.*

Proof. By multiplying both sides of Eq. (A.2) by $(\mathbf{A} \otimes \mathbf{B} + \lambda\mathbb{I})^{-1}$, it is relatively straightforward to show that

$$\mathbf{a} = \mathrm{vec}(\mathbf{V}(\mathbf{C} \odot \mathbf{E})(\mathbf{U}^\top)^{-1})\,, \tag{A.3}$$

with $\odot$ the Hadamard product (element-wise matrix multiplication),

$$\mathbf{E} = \mathbf{U}^{-1}\mathbf{Y}(\mathbf{V}^{-1})^{\top}$$

and

$$\mathrm{diag}_m(\mathrm{vec}(\mathbf{C})) = (\mathbf{\Lambda} \otimes \mathbf{\Sigma} + \lambda\mathbb{I})^{-1}\,.$$

The eigen decompositions of $\mathbf{A}$ and $\mathbf{B}$ as well as all matrix multiplications in Eq. (A.3) can be computed in $\mathcal{O}(n^3)$ time. □

Lastly, I present two matrix identities that are useful in deriving the linear filter of Chaper 4. Consider two matrices of the form

$$\mathbf{A} = a_1\mathbb{I}_m \otimes \mathbb{I}_q + a_2\mathbb{J}_m \otimes \mathbb{I}_q + a_3\mathbb{I}_m \otimes \mathbb{J}_q + a_4\mathbb{J}_m \otimes \mathbb{J}_q$$

and

$$\mathbf{B} = b_1\mathbb{I}_m \otimes \mathbb{I}_q + b_2\mathbb{J}_m \otimes \mathbb{I}_q + b_3\mathbb{I}_m \otimes \mathbb{J}_q + b_4\mathbb{J}_m \otimes \mathbb{J}_q\,.$$

Two properties can easily be deduced.

PROPERTY A.7. $\mathbf{C} = \mathbf{AB}$ *is given by*

$$\mathbf{C} = c_1\mathbb{I}_m \otimes \mathbb{I}_q + c_2\mathbb{J}_m \otimes \mathbb{I}_q + c_3\mathbb{I}_m \otimes \mathbb{J}_q + c_4\mathbb{J}_m \otimes \mathbb{J}_q\,,$$

with

$$\begin{aligned}
c_1 &= a_1b_1\\
c_2 &= a_1b_2 + a_2b_1 + a_2b_2m\\
c_3 &= a_1b_3 + a_3b_1 + a_3b_3q\\
c_4 &= a_1b_4 + a_2b_3 + a_2b_4m + a_3b_2 + a_3b_4q + a_4b_1 + a_4b_2m + a_4b_3q + a_4b_4mq\,.
\end{aligned}$$

PROPERTY A.8. $\mathbf{D} = \mathbf{A}^{-1}$ *is given by*

$$\mathbf{D} = d_1\mathbb{I}_m \otimes \mathbb{I}_q + d_2\mathbb{J}_m \otimes \mathbb{I}_q + d_3\mathbb{I}_m \otimes \mathbb{J}_q + d_4\mathbb{J}_m \otimes \mathbb{J}_q\,,$$

with

$$
\begin{aligned}
d_1 &= \frac{1}{a_1} \\
d_2 &= \frac{-a_2}{a_1(a_1 + a_2 m)} \\
d_3 &= \frac{-a_3}{a_1(a_1 + a_3 q)} \\
d_4 &= (a_2(a_1 + a_3 q)(a_3 + a_4 m) + a_3(a_1 + a_2 m)(a_2 + a_3 q + a_4 q) \\
&\quad - a_4(a_1 + a_2 m)(a_1 + a_3 q))(a_2(a_1 + a_2 m)(a_1 + a_3 q)(a_1 + a_2 m \\
&\quad + a_3 q + a_4 m q))^{-1} .
\end{aligned}
$$

B

Curriculum vitae



Michiel Stock studied bioscience engineering: cell and gene biotechnology with a major in computational biology. In 2012 he joined the KERMIT research unit of the department of mathematical modelling, statistics and bioinformatics as a teaching assistant. He taught (among others) statistics, machine learning, computational modelling and mathematical optimization. As a tutor of various projects and theses, he helped students discover their inner computer scientist. Michiel's research interests lies in the intersection of machine learning, mathematics and computational biology. His work has led to several theoretical and application papers and was presented at multiple international congresses.

Contact

email	**michielfmstockgmail.com**
phone	**+32 475 71 51 73**
Github	**MichielStock**
Twitter	**@michielstock**

Education

2001–2007 **Science-Mathematics**, *Sint-Pietercollege en Sint-Jozefshandelsschool*, Blankenberge

2007–2010 **Bachelor BioScience Engineering**, *Ghent University*, Ghent, *cum laude*.

2010–2012 **Master BioScience Engineering: Cell- and Gene Biotechnology**, *Ghent University*, Ghent, *magna cum laude*, major: computational biology.

Master thesis

title	*Learning Pairwise Relations in Bioinformatics: Three Case Studies*
supervisors	Bernard De Baets & Willem Waegeman
description	In my thesis I investigated how Kronecker kernel-based methods for pairwise learning can be applied for solving problems in drug-target interaction prediction, enzyme annotation and microbiology.
link	`goo.gl/7wzPGa`

Experience

Academic

2012–present **Research and teaching assistant**, *Ghent University*, Ghent At the departement of Mathematical Modelling at the faculty of BioScience Engineering I was heavily involved in both teaching and researching statistical modelling and computational intelligence.

- Research topic: *Exact and Efficient Algorithms for Pairwise Learning*;
- Teaching responsibilities:
  - Statistical Data Analysis
  - Applied Statistics
  - Predictive Modelling
  - Computational Modelling
  - Selected Topics in Mathematical Optimization
  - Probabilistic Models
  - Quality Control and Risk Assessment
- Tutor of six bachelor projects and five master theses.

Foreign experiences

Summer 2011 **Visiting student**, *Turku University*, Turku, Finland. Performed a guest stay at the department of computer sciences

Summer 2013 **Visiting researcher**, *Turku University*, Turku, Finland. Performed a second guest stay at the department of computer sciences

Summer jobs

August 2010 **Biotech laboratory assistant**, *CropDesign*, Ghent. Laboratory assistant plant transformation

August 2011 **Biotech laboratory assistant**, *Devgen*, Ghent. Laboratory assistant plant transformation

Languages

Dutch **native speaker**

English **fluent**

French **working knowledge**

Swedish **Duolingo enthousiast**

Skills

Teaching	several years of experience teaching mathematics
Science comm.	oral presentation, designing posters, writing papers, etc.
Progr. languages	Python, R, Matlab, Perl, etc.
Modelling	machine learning, statistics, mathematical optimization, bioinformatics, etc.
Biotech	educated as a bioscience engineer with a speciality in biotechnology

Awards

Best paper	2015 IEEE Computer Intelligence Society Outstanding Transactions on Fuzzy Systems Outstanding Paper award for the paper "A kernel-based framework for learning graded relations from data".
iGEM	gold medal and nomination for the best measurement award for the international Genetically Engineered Machine (iGEM) competition held annually in Boston for our project dewpal, a biotechnological water collector.

Theses

2012-2013 **De Paepe, A.**, The prediction of interaction between mRNA and miRNA using machine learning techniques.,
Supervisors: Waegeman, W. and De Baets, B.
Tutor: **Stock, M.**

2013-2014 **De Clercq, M.**, Prediction of ingredient combinations using machine learning techniques.,
Supervisors: Waegeman, W. and De Baets, B.
Tutor: **Stock, M.**

2014-2015 **Tsang, W. K.**, Automation and in-silico prediction of wet lab-experiments on microbial networks.,
Supervisors: Waegeman, W. and Boon, N.
Tutors: **Stock, M.** and Vilchez, R.

2015-2016 **Tilleman, L.**, In silico engineering van cytochroom P450 via machine learning technieken.,
Supervisors: Waegeman, W. and De Baets, B.
Tutor: **Stock, M.**

2016-2017 **Bodyn, L.**, Exploration of deep autoencoders on cooking recipes.,
Supervisors: Ley, C. and Waegeman, W.
Tutor: **Stock, M.**

Bachelor projects

2013-2014 **De Jaegher, B., De Kesel, J., Depicker, A. and Verbanck, T.**, Innovatieve toepassingen van artificiële intelligentie in de levensmiddelensector.,
Supervisors: Waegeman, W. and De Baets, B.
Tutor: **Stock, M.**

2013-2014 **Clauwaert, J., Couvreur, K., Demol, M. and Ramon, F.**, Chemo-informatica: Kan de computer de chemicus vervangen?,
Supervisors: Waegeman, W. and De Baets, B.
Tutors: **Stock, M.** and Van der Weeën, P.

2014-2015 **Heyse, J., Mauroo, E., Minne, M. and Roels, K.**, Bepaling van de meest representatieve meetplaats van het grondwaterpeil bij infrastructuurwerken.,
Supervisors: Waegeman, W. and Nopens, I.
Tutor: **Stock, M.** and Van Hoey, S.

2014-2015 **Clicque, H., Maes, K., Marcoen, V. and Stalmans, L.**, De computer achter het fornuis.,
Supervisors: Waegeman, W. and De Baets, B.
Tutors: De Clercq, M. and **Stock, M.**

2014-2015 **Backx, S., Desmedt, W., Ghysels, S. and Vanavermaete, D.**, Data-gedreven synthetische biologie: ontwerp van nieuwe eiwitten.,
Supervisors: Waegeman, W. and De Baets, B.
Tutor: **Stock, M.**

2015-2016 **Dhiedt, E., Hoebeke, L., Puynen, S. and Vanwyck, T.** , Hoeveel informatie zit er in een ecologisch netwerk?,
Supervisors: Waegeman, W. and De Baets, B.
Tutor: **Stock, M.**

Interests

reading	avid reader of both fiction and non-fiction, see my Goodreads account: `goo.gl/rXyFz9`
games	fan of board games, Dungeons & Dragons
sports	running, bouldering, badminton, ...
science	my passion for science, mathematics and programming is not contained within the working hours

Publications

1. Waegeman, W., Pahikkala, T., Airola, A., Salakoski, T., **Stock, M.**, and De Baets, B. (2012). A kernel-based framework for learning graded relations from data. *IEEE Transactions on Fuzzy Systems*, 20(6):1090–1101

2. Pahikkala, T., Airola, A., **Stock, M.**, De Baets, B., and Waegeman, W. (2013). Efficient regularized least-squares algorithms for conditional ranking on relational data. *Machine Learning*, 93(2-3):321–356

3. **Stock, M.**, Hoefman, S., Kerckhof, F.-M., Boon, N., De Vos, P., De Baets, B., Heylen, K., and Waegeman, W. (2013). Ex-

ploration and prediction of interactions between methanotrophs and heterotrophs. *Research in Microbiology*, 164(10):1045–1054

4. Costello, J., Heiser, L., Georgii, E., Gönen, M., Menden, M., Wang, N., Bansal, M., Ammad-ud din, M., Hintsanen, P., Khan, S., Mpindi, J.-P., Kallioniemi, O., Honkela, A., Aittokallio, T., Wennerberg, K., Abbuehl, J.-P., Allen, J., Altman, R., Balcome, S., Battle, A., Bender, A., Berger, B., Bernard, J., Bhattacharjee, M., Bhuvaneshwar, K., Bieberich, A., Boehm, F., Califano, A., Chan, C., Chen, B., Chen, T.-H., Choi, J., Coelho, L., Cokelaer, T., Creighton, C., Cui, J., Dampier, W., Davisson, V., De Baets, B., Deshpande, R., DiCamillo, B., Dundar, M., Duren, Z., Ertel, A., Fan, H., Fang, H., Gauba, R., Gottlieb, A., Grau, M., Gusev, Y., Ha, M., Han, L., Harris, M., Henderson, N., Hejase, H., Homicsko, K., Hou, J., Hwang, W., Ijzerman, A., Karacali, B., Keles, S., Kendziorski, C., Kim, J., Kim, M., Kim, Y., Knowles, D., Koller, D., Lee, J., Lee, J., Lenselink, E., Li, B., Li, B., Li, J., Liang, H., Ma, J., Madhavan, S., Mooney, S., Myers, C., Newton, M., Overington, J., Pal, R., Peng, J., Pestell, R., Prill, R., Qiu, P., Rajwa, B., Sadanandam, A., Sambo, F., Shin, H., Song, J., Song, L., Sridhar, A., **Stock, M.**, Sun, W., Ta, T., Tadesse, M., Tan, M., Tang, H., Theodorescu, D., Toffolo, G., Tozeren, A., Trepicchio, W., Varoquaux, N., Vert, J.-P., Waegeman, W., Walter, T., Wan, Q., Wang, D., Wang, W., Wang, Y., Wang, Z., Wegner, J., Wu, T., Xia, T., Xiao, G., Xie, Y., Xu, Y., Yang, J., Yuan, Y., Zhang, S., Zhang, X.-S., Zhao, J., Zuo, C., van Vlijmen, H., van Westen, G., Collins, J., Gallahan, D., Singer, D., Saez-Rodriguez, J., Kaski, S., Stolovitzky, J. G., and Stolovitzky, G. (2014). A community effort to assess and improve drug sensitivity prediction algorithms. *Nature Biotechnology*, 32(12):1202–1212

5. Pahikkala, T., **Stock, M.**, Airola, A., Aittokallio, T., De Baets, B., and Waegeman, W. (2014). A two-step learning approach for solving full and almost full cold start problems in dyadic prediction. *Lecture Notes in Computer Science*, 8725:517–532

6. **Stock, M.**, Fober, T., Hüllermeier, E., Glinca, S., Klebe, G., Pahikkala, T., Airola, A., De Baets, B., and Waegeman, W. (2014). Identification of functionally related enzymes by learning-to-rank methods. *IEEE Transactions on Computational Biology and Bioinformatics*, 11(6):1157–1169

7. De Clercq, M., **Stock, M.**, De Baets, B., and Waegeman, W. (2015). Data-driven recipe completion using machine learning methods. *Trends in Food Science & Technology*, 49:1–13

8. De Paepe, A., Van Peer, G., **Stock, M.**, Volders, P.-J., Vandesompele, J., De Baets, B., and Waegeman, W. (2016). miRNA target prediction through modeling quantitative and qualitative miRNA binding site information in a stacked model structure. *Nucleic Acid Research*, Accepted

9. Gonnelli, G., **Stock, M.**, Verwaeren, J., Maddelein, D., De Baets, B., Martens, L., and Degroeve, S. (2015). A decoy-free approach to the identification of peptides. *Journal of Proteome Research*, 14(4):1792–1798

10. **Stock, M.**, Dembczynski, K., De Baets, B., and Waegeman, W. (2016). Exact and efficient top-K inference for multi-target prediction by querying separable linear relational models. *Data Mining and Knowledge Discovery*, 30(5):1370–1394

11. Nicolaï, N., De Leersnyder, F., Copot, D., **Stock, M.**, Ionescu, C., Gernaey, K., Nopens, I., and De Beer, T. (2017). Development of a PAT-based advanced process control for continuous twin-screw wet granulation. *AIChE Journal*, Under revi

12. **Stock, M.**, Poisot, T., Waegeman, W., and De Baets, B. (2017). Linear filtering reveals false negatives in species interaction data. *Scientific Reports*, 7(45908):1–8

insert blanco page at the end