Accelerating sparse restricted Boltzmann machine training using non-Gaussianity measures

Sander Dieleman
Electronics and Information Systems (ELIS)
Ghent University, Ghent, Belgium
sander.dieleman@ugent.be

Benjamin Schrauwen
Electronics and Information Systems (ELIS)
Ghent University, Ghent, Belgium
benjamin.schrauwen@ugent.be

Abstract

In recent years, sparse restricted Boltzmann machines have gained popularity as unsupervised feature extractors. Starting from the observation that their training process is biphasic, we investigate how it can be accelerated: by determining when it can be stopped based on the non-Gaussianity of the distribution of the model parameters, and by increasing the learning rate when the learnt filters have locked on to their preferred configurations. We evaluated our approach on the CIFAR-10, NORB and GTZAN datasets.

1 Introduction

Since the advent of *deep learning* in 2006 with Hinton et al. [9]'s seminal paper, restricted Boltzmann machines (RBMs) have become very popular, not just as building blocks for deep architectures, but also as feature extractors and generative models in their own right. A few years later Lee et al. [13] introduced the sparse RBM, which has since then formed the basis of a lot of work in unsupervised feature learning [3, 4, 14, 15].

In this paper, we investigate how the training of sparse Gaussian RBMs intended for feature extraction can be monitored and accelerated, starting from the observation that the training process seems to be biphasic in nature: a relatively short exploration phase is followed by a much longer finetuning phase. We propose to measure the non-Gaussianity of the distribution of the model parameters during training, and we show how this information can be used to speed up training.

The paper is structured as follows: Section 2 introduces sparse Gaussian RBMs. Section 3 describes the challenges of measuring training progress for these models, and the advantages of measuring the non-Gaussianity of the distribution of the model parameters during training. Our experiments are described and results are reported in Section 4. Finally, we conclude in Section 5 and point out some directions for future research.

2 Restricted Boltzmann machines

2.1 Overview

A restricted Boltzmann machine is a probabilistic model consisting of a set of *visible units* $\mathbf{v}$ and a set of *hidden units* $\mathbf{h}$ which form a bipartite graph; there are no connections between pairs of visible units or pairs of hidden units, but every visible unit is connected to every hidden unit. An RBM can be characterized by an *energy function* $E(\mathbf{v}, \mathbf{h})$. The joint probability distribution over all units defined by the model is then given by $P(\mathbf{v}, \mathbf{h}) = Z^{-1} \exp(-E(\mathbf{v}, \mathbf{h}))$, where $Z = \int_{\mathbf{v},\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))$ is the normalizing constant. The basic RBM model has the following energy function (assuming $\mathbf{v}$ and $\mathbf{h}$ are column vectors):

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{n_v} b_i v_i - \sum_{j=1}^{n_h} c_j h_j - \sum_{i=1}^{n_v} \sum_{j=1}^{n_h} v_i W_{ij} h_j, \quad \mathbf{v} \in \{0,1\}^{n_v}, \mathbf{h} \in \{0,1\}^{n_h},$$

where n_v and n_h are the number of visible and hidden units respectively. The matrix W (*weights*) and the vectors $\mathbf{b}$ and $\mathbf{c}$ (*biases*) are the parameters of the model. The set of weights that connect a single hidden unit to all visible units is called a *filter*. This energy function gives rise to an RBM with *Bernoulli units* (often called *binary units*).

The visible units of an RBM correspond to the input variables of the data that is to be modelled. The hidden units capture correlations between visible units. To use an RBM as a feature extractor, the values of the hidden units are inferred from the visible units[1]. This yields a feature representation of the input. Unfortunately, maximum likelihood learning is intractable in RBMs. Instead, *contrastive divergence*, which is an approximation to maximum likelihood learning, can be used [7].

2.2 Gaussian RBMs

The RBM described earlier can only be used to model binary data. It is possible to construct an RBM for continuous data, with Gaussian visible units, which are normally distributed conditioned on the hidden units. The following energy function gives rise to a *Gaussian RBM*:

$$E(\mathbf{v}, \mathbf{h}) = \sum_{i=1}^{n_v} \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{j=1}^{n_h} c_j h_j - \sum_{i=1}^{n_v} \sum_{j=1}^{n_h} \frac{v_i}{\sigma_i} W_{ij} h_j, \quad \mathbf{v} \in \mathbb{R}^{n_v}, \mathbf{h} \in \{0,1\}^{n_h}.$$

In this formulation, each visible unit has a fixed variance σ_i, and only the mean depends on the hidden units. The variances of the visible units can be either learnt, or they can be set to a sensible value on beforehand. Because of its limited capacity to model variance, this type of RBM is rarely as a density model. The variances are often fixed at 1 for convenience.

2.3 Sparse RBMs

It has been demonstrated in the past that sparsity is a desirable property for feature representations: it increases interpretability and sometimes classification performance [8]. We can encourage an RBM to produce sparse features with a form of data-dependent regularization. To do this, we add a penalty term to the objective function which is proportional to the difference between the probability of activation of the hidden units and a small target value. For details, we refer to Goh et al. [5].

3 Measuring training progress

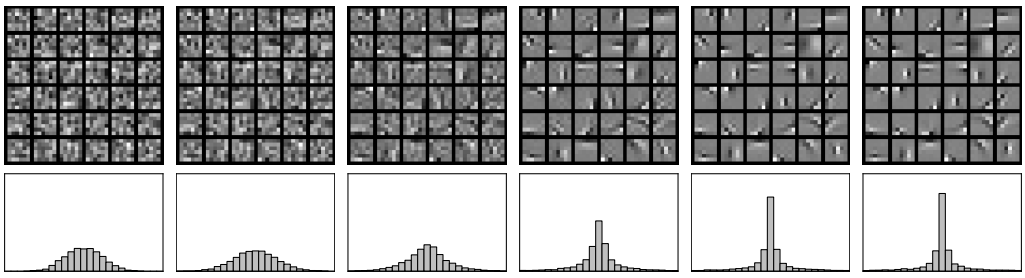
Monitoring RBM training progress is a challenging problem. The objective function itself cannot be used for this purpose because it is intractable to compute. *Annealed importance sampling* [16] is sometimes used to estimate this quantity, but it is not always reliable and computationally intensive.

A frequently used alternative is the **one-step reconstruction error**: the mean squared error between a set of data points and the reconstruction from their inferred feature representations. Unfortunately this measure can be very misleading, as it correlates poorly with training progress: it strongly depends on the mixing rate of the alternating Gibbs chain that is used to draw samples from the model distribution (see Section 5 of Hinton [8] for details).

When training an RBM to use as a feature extractor, there is another option: we could **evaluate classification performance** during training, and stop when the classification accuracy reaches a plateau. However, this means that we would have to extract features and train a classifier at regular intervals, which is a very expensive operation. As a result, this approach is rarely used in practice.

For differentiable convex optimization problems, a common approach to measuring progress is to monitor the **magnitude of the gradient**; it is expected to decrease as the optimization progresses, and it is zero at the optimal solution. Although this approach may be applicable to some non-convex

[1] With Bernoulli hidden units, the probabilities of activation are often used instead of samples, to yield a deterministic feature representation.



(a) 10 epochs (b) 50 epochs (c) 100 epochs (d) 200 epochs (e) 400 epochs (f) 1000 epochs

Figure 1: Evolution of the weights during sparse RBM training on the modified CIFAR-10 dataset, with initial parameter values drawn from a Gaussian distribution. A subset of the filters (top) as well as the weight histograms (bottom) are shown. After 50 epochs, the filters have found their preferred configurations. After 400 epochs, they have been finetuned. See Figure 2(a) for the evolution of the non-Gaussianity measures during this experiment.

problems, this is not the case for RBM training. The magnitude of the gradient reaches a minimal value fairly quickly, after which it stays at that level.

3.1 Parameter histograms

Several authors have suggested to monitor statistics and histograms of the model parameters and the activations of the hidden units, as well as their gradients, during training. Bengio [1] suggests that visualizing these statistics and histograms can be useful to monitor training progress and to identify under- or overfitting. Yosinski and Lipson [19] also note that these visualizations can be instrumental in identifying inadequately tuned hyperparameters and implementation bugs.

3.2 Measuring non-Gaussianity of the parameters

Inspecting a histogram of the weight matrix W, which connect the visible and hidden units, reveals an interesting progression: during the first few epochs, the histogram has a Gaussian shape. Afterwards, it becomes progressively more peaked around zero, which indicates that the weight matrix is becoming sparser. This is despite the fact that the sparsity penalty applies to the activations of the hidden units, and not directly to the parameters. This is shown in Figure 1.

This implies that the training process consists of two phases: the first phase could be interpreted as an *exploration* phase, where good configurations for the filters are found. Once each filter has 'locked on' to a good configuration, it becomes progressively more pronounced and any remaining noise disappears: the filters are *finetuned*.

A similar observation was made by Yosinski and Lipson [19], who found in their RBM training experiments that the probabilities of activation of the hidden units had almost converged to their final values after only a few epochs, but the reconstruction error had not yet decreased significantly at that point. From this, they concluded that the units quickly decide on their preferred stimulus, but further finetuning takes much longer.

This indicates that measuring the non-Gaussianity of the histogram of the parameters could tell us which phase the training is currently in, and perhaps also how far training has progressed. Measures of non-Gaussianity have been studied in the context of Independent Component Analysis (ICA) [10]. Two such measures that are commonly used are the *negentropy* and the *excess kurtosis*.

The **negentropy** $J(X)$ of a random variable X is the difference between the differential entropy of a Gaussian-distributed random variable with the same variance as X (which we will denote σ_X^2), and the differential entropy of X itself:

$$J(X) = H(\mathcal{N}(0, \sigma_X^2)) - H(X) = \frac{1}{2} \ln(2\pi e \sigma_X^2) - H(X).$$

For a given variance, the Gaussian distribution is the *maximum entropy distribution*, so the negentropy $J(X)$ is always nonnegative, and it is zero if and only if X has a Gaussian distribution. A histogram-based estimation can be used for the differential entropy of X.

The **excess kurtosis** $\mathrm{Kurt}[X]$ of a random variable X, henceforth simply called 'kurtosis', measures the *peakedness* or *tail heaviness* of its distribution and is defined as follows:

$$\mathrm{Kurt}[X] = \frac{\mathbb{E}[(X - \mathbb{E}[X])^4]}{(\mathrm{Var}[X])^2} - 3.$$

It is zero when X is Gaussian-distributed. Unlike the negentropy, it allows for distinguishing sub-Gaussian (flatter) and super-Gaussian (more peaked) distributions. The biphasic nature of the training process is demonstrated using these non-Gaussianity measures in the experiment described in Section 4.3.1.

3.3 Determining convergence

There is an inherent limit to the peakedness of the distribution of the weights: at some point it cannot become any more peaked without changing the configurations of the filters. At the end of the finetuning phase of training, the peakedness will reach its maximum, and as a result the kurtosis and the negentropy of the distribution of the weights will both reach a saturation value. This value depends on the model parameters and the nature of the dataset. As it turns out, the quality of the features will not increase past this point. This is demonstrated in Section 4.3.2.

3.4 Faster training with a non-decreasing learning rate schedule

A key disadvantage of Gaussian RBMs is that they typically require a lower learning rate to avoid divergence, compared to their binary counterparts. As previously discussed, the parameters will reach their preferred configurations after only a few epochs of training, at which point the magnitude of the gradient will have decreased significantly. This suggests that we could try increasing the learning rate at that point. This is investigated in Section 4.3.3.

4 Experiments and results

4.1 Datasets

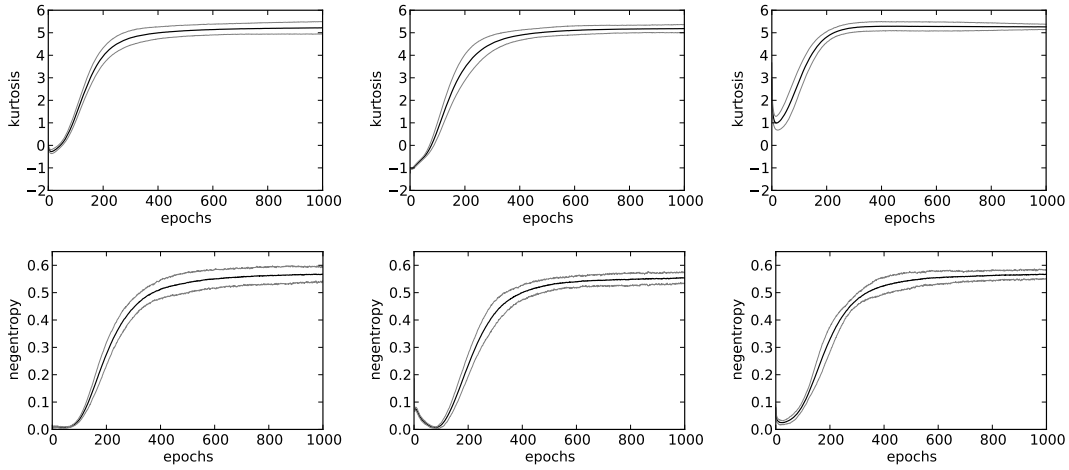
We used modified versions of the NORB [12] and CIFAR-10 [11] datasets for our experiments. The resulting modified datasets are the same as those used by Saxe et al. [17]. We also used the GTZAN genre classification dataset [18].

The normalized-uniform subset of the 'small NORB dataset' contains 48,600 stereo image pairs of toys (24,300 for training, 24,300 for testing), imaged under different angles and lighting conditions and divided into 5 classes. We retained only the first image of each stereo pair and downscaled the images from 96x96 to 32x32 pixels. The CIFAR-10 dataset contains 60,000 32x32 color images (50,000 for training, 10,000 for testing), divided into 10 classes. We converted them to grayscale. The GTZAN dataset contains 1,000 30-second music fragments, divided into 10 genres. We used 500 fragments for training, 200 for validation and 300 for testing. Like Hamel et al. [6], we extracted mel-spectrograms with 128 components from the audio data.

4.2 Classification pipeline

We trained a number of RBMs for feature extraction. To assess the usefulness of non-Gaussianity measures for tracking training progress, we evaluated classification performance at regular intervals during training, which allowed us to study the relation between both.

Our classification pipeline for NORB and CIFAR-10 is largely inspired by Coates et al. [3]. To train the RBMs, we first extracted 100,000 random 6x6 patches from training examples. Each of the patches was then preprocessed. We applied *contrast normalization* by subtracting the mean of the entire patch from all the pixels, and dividing them by the standard deviation, followed by *ZCA whitening* [11] and a *zero mean unit variance* transform.



(a) Gaussian (b) uniform (sub-Gaussian) (c) Laplacian (super-Gaussian)

Figure 2: Evolution of kurtosis (top) and negentropy (bottom) of the parameters W during sparse RBM training on the modified CIFAR-10 dataset, with initial parameter values drawn from different distributions. The mean (thick line) and 99% confidence intervals (thin lines), computed over 10 training runs with different random initializations are shown.

To extract features from the data, we first extracted all possible patches from each image. The patches were then preprocessed as above, and 200 features were extracted from each patch with a trained RBM. These features were then sum-pooled within each quadrant of the image, yielding a feature vector with 800 components.

For GTZAN, we extracted 100,000 random windows from the mel-spectrogram representations of the fragments to use for RBM training. The windows were ZCA whitened and a zero mean unit variance transform was applied. To extract features from the data, we preprocessed all of the mel-spectrogram windows for each fragment. We then pooled the features by computing the mean, variance, minimum and maximum of each component for every 128 windows (no overlap), as in Hamel et al. [6]. This yielded 10 coarse timescale feature vectors with 800 components per fragment. Each of the feature vectors was used as a separate training example.

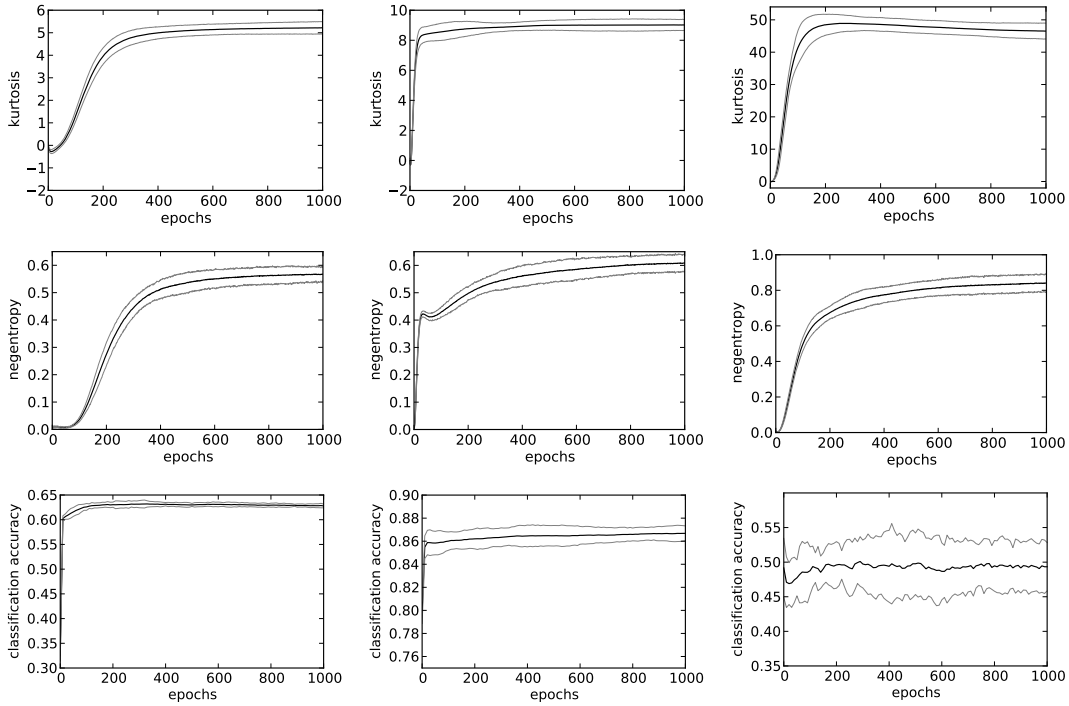
For all datasets, each RBM had 200 hidden units, and the training procedure was run for 1,000 epochs. Although Coates et al. [3] tried much larger models, we decided to keep them relatively small to reduce computation time. As a result, the classification performance attained by the models we trained is far below the state of the art on all datasets. Since we are only looking to measure training progress, this should not be an issue.

The feature vectors were then used to train logistic regression classifiers with minibatch gradient descent for 400 epochs, after applying another zero mean unit variance transform. The learning rate was set to 0.1. To obtain a validation dataset for classifier training for NORB and CIFAR-10, we split off 10% of the training sets. We used Theano [2] to enable GPU acceleration for both classifier and RBM training, as well as data preprocessing.

4.3 Experiments

4.3.1 Different initializations

To verify that the distribution of the weights becomes Gaussian at first, and then progressively more peaked, we varied the distribution from which the initial weight values are drawn. We tried Gaussian, uniform and Laplacian distributions, with the parameters chosen such that the mean is 0 and the standard deviation is 0.05. The Laplacian distribution is super-Gaussian and has a kurtosis of 3. The uniform distribution is sub-Gaussian with a kurtosis of -1.2.



(a) CIFAR-10 (b) NORB (c) GTZAN

Figure 3: Evolution of kurtosis (top), negentropy (middle) and classification accuracy (bottom) during sparse RBM training on (a) CIFAR-10, (b) NORB and (c) GTZAN.

For each possible distribution, we trained 10 RBMs on the modified CIFAR-10 dataset with CD-1 (one step contrastive divergence). We used minibatch gradient descent with a learning rate of 0.01, a relative sparsity cost of 0.1, a sparsity target of 0.1 and a minibatch size of 100. The evolution of the non-Gaussianity measures during training is visualized in Figure 2.

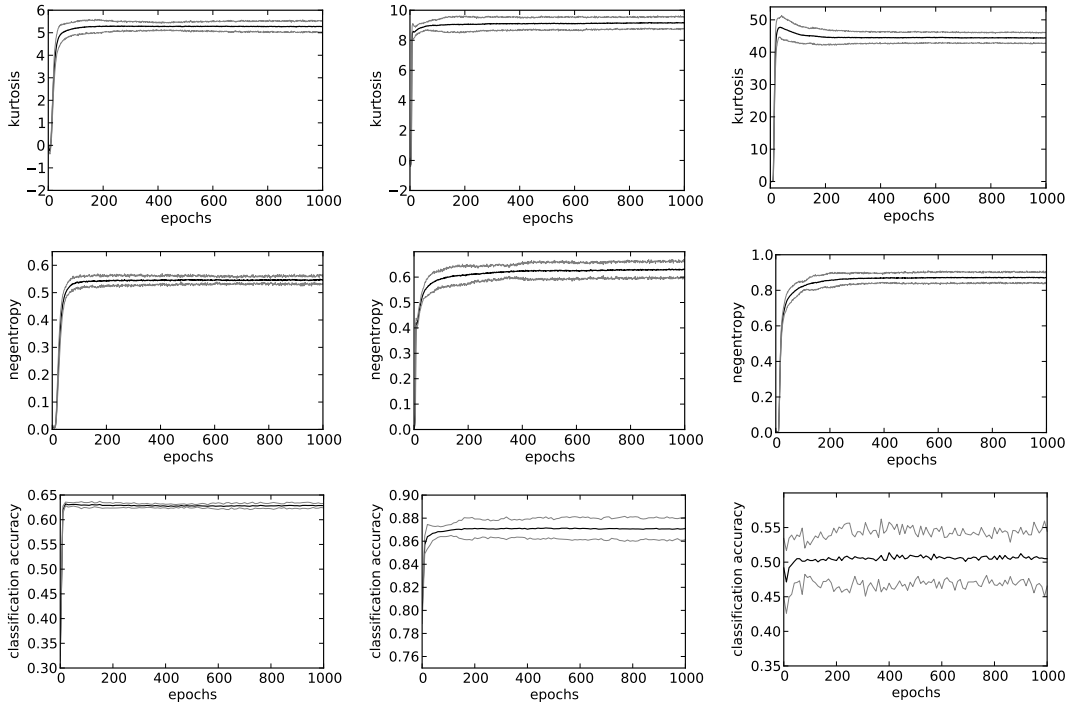
The results corroborate our qualitative observations in Section 3.2 regarding the biphasic nature of the training process. The negentropy decreases (Laplacian and uniform initialization) or stays at zero (Gaussian initialization) for a while, before increasing toward a certain data-dependent level, and then saturating. In the Laplacian case, the kurtosis never reaches zero, indicating that the distribution never becomes fully Gaussian.

4.3.2 Relation with classification performance

Next, we investigated the relation between the non-Gaussianity of the distribution of the weights, and the classification performance attained by extracting features using these weights. We stored the parameters every epoch, and then trained a classifier every 10 epochs. We trained 10 RBMs on each dataset. For this and all following experiments, we initialized the parameter values from a Gaussian distribution. All hyperparameters were the same as in the previous experiment. The evolution of the non-Gaussianity measures and the classification accuracy are visualized in Figure 3.

We can see that the classification accuracy on CIFAR-10 reaches its maximal value after about 150 epochs. The kurtosis and negentropy are increasing at that point, but they have not reached their saturation values. This indicates that a certain degree of noise in the filters will not affect classification performance. A possible explanation for this is that the classification pipeline involves a feature pooling step, which reduces variance.

For NORB, things are a bit more complicated, as shown in Figure 3(b): the negentropy increases quickly almost immediately, but then it dips down, only to increase again at a much slower rate. The kurtosis shows no such behavior; it increases very quickly initially, and then much more slowly



(a) CIFAR-10 (b) NORB (c) GTZAN

Figure 4: Evolution of kurtosis (top), negentropy (middle) and classification accuracy (bottom) during sparse RBM training on (a) CIFAR-10, (b) NORB and (c) GTZAN, where the learning rate is increased tenfold after a fixed number of epochs.

until it saturates roughly around epoch 500. The classification accuracy shows a similar pattern, saturating slightly earlier at around epoch 400.

For GTZAN, the kurtosis initially increases beyond its saturation value. The classification performance does not significantly improve during training, so unfortunately we cannot draw any conclusions. Presumably, the model is too small for this dataset.

4.3.3 Effect of increasing the learning rate

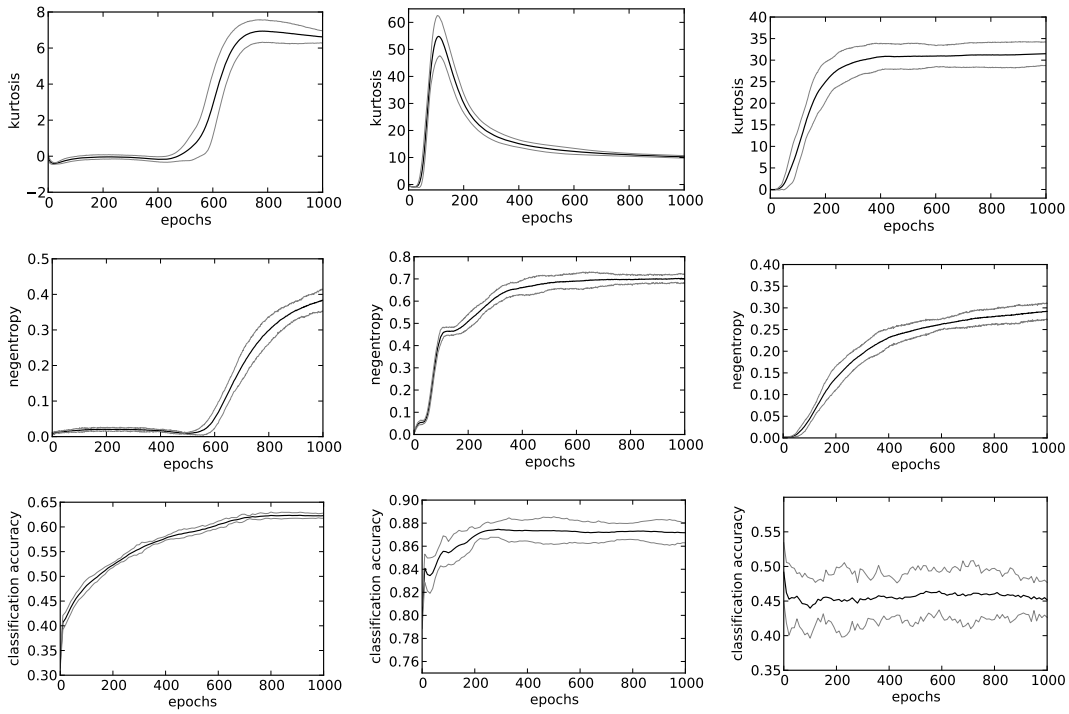
To examine the effect of increasing the learning rate once the filters have locked on to their preferred configurations, we repeated the previous experiment, but we increased the learning rate tenfold (to 0.1) after five epochs for CIFAR-10 and NORB, and after ten epochs for GTZAN. For convenience, we decided to increase the learning rate after a fixed number of epochs. It is important to note here that using this higher learning rate from the start would lead to immediate divergence.

The results are shown in Figure 4. This experiment demonstrates that increasing the learning rate does not seem to affect classification performance negatively, so it is a simple and efficient method to get good features faster.

4.3.4 RBMs without sparsity regularization

Up until now we have been exclusively using sparse RBMs. To establish the importance of the sparsity penalty, we also repeated the experiment described in Section 4.3.2 without it (i.e. with a sparsity cost of zero). The results are shown in Figure 5. We also tried to repeat the experiment from Section 4.3.3 without sparsity, but this consistently lead to divergence.

Without sparsity, there is no clear biphasic structure to be discerned, and it seems to take longer to get good features. Although comparable performance is eventually reached on CIFAR-10 and



(a) CIFAR-10 (b) NORB (c) GTZAN

Figure 5: Evolution of kurtosis (top), negentropy (middle) and classification accuracy (bottom) during RBM training on (a) CIFAR-10, (b) NORB and (c) GTZAN, when no sparsity penalty is used.

NORB, this is not the case for GTZAN. Overall, this experiment shows that the sparsity penalty has an additional benefit on top of increasing interpretability of the features: it reduces the training time needed to obtain good features.

5 Conclusion and future work

Measures of non-Gaussianity are a useful tool for monitoring training progress. For sparse RBMs, the evolution of these measures during training will follow a specific pattern revealing a biphasic structure, which we can take advantage of. Clearly it is pointless to continue training once the non-Gaussianity measures saturate, and typically we can stop training even before that happens; the learnt filters will be somewhat noisy, but this turns out not to affect classification performance.

Furthermore, we have shown that the learning rate can safely be increased after the filters have locked on to their preferred configurations. This allows for good features to be obtained much faster, at no additional cost: classification performance is not affected. It may seem odd to advocate an increase of the learning rate during training, when monotonically decreasing learning rate schedules are commonplace. However, both approaches are not mutually exclusive: one could imagine using a decreasing learning schedule with a one-time increase.

In summary, we have demonstrated two ways to accelerate sparse RBM training: by stopping the training process once the non-Gaussianity of the parameters saturates, and by increasing the learning rate once the filters have locked on to their preferred configurations.

In future work, we would like to investigate more rigorously when and by how much the learning rate can safely be increased during sparse RBM training, without it negatively affecting performance. We would also like to study the connection with more sophisticated adaptive learning rate methods, which typically make use of second order information.

References

[1] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. *CoRR*, abs/1206.5533, 2012.

[2] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.

[3] Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. *Journal of Machine Learning Research - Proceedings Track*, 15:215–223, 2011.

[4] Sander Dieleman, Philémon Brakel, and Benjamin Schrauwen. Audio-based music classification with a pretrained convolutional network. In *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, 2011.

[5] Hanlin Goh, Nicolas Thome, and Matthieu Cord. Biasing restricted boltzmann machines to manipulate latent selectivity and sparsity. In *Deep Learning and Unsupervised Feature Learning Workshop — NIPS*, 2010.

[6] Philippe Hamel, Simon Lemieux, Yoshua Bengio, and Douglas Eck. Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011.

[7] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:2002, 2000.

[8] Geoffrey E. Hinton. A practical guide to training restricted boltzmann machines. Technical report, University of Toronto, 2010.

[9] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. ISSN 0899-7667.

[10] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Netw.*, 13(4-5):411–430, May 2000. ISSN 0893-6080.

[11] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Master's thesis, 2009.

[12] Yann LeCun, Fu Jie Huang, and Léon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the 2004 IEEE computer society conference on Computer vision and pattern recognition*, CVPR'04, pages 97–104, Washington, DC, USA, 2004. IEEE Computer Society.

[13] Honglak Lee, Chaitanya Ekanadham, and Andrew Y. Ng. Sparse deep belief net model for visual area v2. In *Advances in Neural Information Processing Systems 20*. MIT Press, 2008.

[14] Juhan Nam, Jorge Herrera, Malcolm Slaney, and Julius O. Smith. Learning sparse feature representations for music annotation and retrieval. In *Proceedings of the 13th International Conference for Music Information Retrieval (ISMIR) Conference*, 2012.

[15] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. Multimodal deep learning. In *International Conference on Machine Learning (ICML)*, Bellevue, USA, June 2011.

[16] Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of Deep Belief Networks. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 872–879. Omnipress, 2008.

[17] Andrew Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Ng. On random weights and unsupervised feature learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1089–1096, New York, NY, USA, June 2011. ISBN 978-1-4503-0619-5.

[18] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10:293–302, 2002. ISSN 1063-6676.

[19] Jason Yosinski and Hod Lipson. Visually debugging restricted boltzmann machine training with a 3d example. In *Representation Learning Workshop, 29th International Conference on Machine Learning*, 2012.