Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient

Tijmen Tieleman TIJMEN@CS.TORONTO.EDU
Department of Computer Science, University of Toronto, Toronto, Ontario M5S 3G4, Canada

Abstract

A new algorithm for training Restricted Boltzmann Machines is introduced. The algorithm, named Persistent Contrastive Divergence, is different from the standard Contrastive Divergence algorithms in that it aims to draw samples from almost exactly the model distribution. It is compared to some standard Contrastive Divergence and Pseudo-Likelihood algorithms on the tasks of modeling and classifying various types of data. The Persistent Contrastive Divergence algorithm outperforms the other algorithms, and is equally fast and simple.

1. Introduction

Restricted Boltzmann Machines (RBMs) (Hinton et al., 2006; Smolensky, 1986) are neural network models for unsupervised learning, but have recently seen a lot of application as feature extraction methods for supervised learning algorithms (Salakhutdinov et al., 2007; Larochelle et al., 2007; Bengio et al., 2007; Gehler et al., 2006; Hinton et al., 2006; Hinton & Salakhutdinov, 2006). The success of these models raises the issue of how best to train them.

Most training algorithms are based on gradient descent, but the standard objective function (training data likelihood) is intractable, so the algorithms differ in their choice of approximation to the gradient of the objective function. At present, the most popular gradient approximation is the Contrastive Divergence (CD) approximation (Hinton et al., 2006; Hinton, 2002; Bengio & Delalleau, 2007); more specifically the CD-1 approximation. However, it is not obvious whether it is the best. For example, the CD algorithm has a parameter specifying the number of Markov Chain transitions performed, and although the most commonly chosen value is 1, other choices are possible and reasonable, too (Carreira-Perpinan & Hinton, 2005).

In this paper, a new gradient approximation algorithm is presented and compared to a variety of CD-based algorithms. The quantitative measures of test data likelihood (for unsupervised learning) and classification error rate (for supervised learning) are investigated, and the type of feature detectors that are developed are also shown. We find that the new algorithm produces more meaningful feature detectors, and outperforms the other algorithms.

The RBMs on which these experiments were done all had binary units. However, this special case can easily be generalized to other *harmoniums* (Smolensky, 1986; Welling et al., 2005) in which the units have Gaussian, Poisson, multinomial, or other distributions in the exponential family, and the training algorithms described here require only minor modifications to work in most of those models.

In Section 2, the RBM model and CD gradient estimator are discussed. In Section 3, the Persistent Contrastive Divergence algorithm is introduced. In Sections 4 and 5, the experiments and results are described, and Section 6 concludes with a discussion and some plans for future work.

2. RBMs and the CD Gradient Approximation

2.1. Restricted Boltzmann Machines

An RBM is an energy-based model for unsupervised learning (Hinton, 2002; Smolensky, 1986). It consists of two layers of binary units: one *visible*, to represent the data, and one *hidden*, to increase learning capacity. Standard notation is to use i for indices of visible units, j for indices of hidden units, and w_{ij} for the strength of the connection between the i^{th} visible unit and the j^{th} hidden unit. If v_i denotes the state of the

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

i^{th} visible unit, and h_j denotes the state of the j^{th} hidden unit, an *energy* function is defined on states: $E(v,h) = -\sum_{i,j} v_i h_j w_{ij} - \sum_i v_i b_i - \sum_j h_j b_j$, where b stands for the biases. Through these energies, probabilities are defined as $P(v,h) = \frac{e^{-E(v,h)}}{Z}$ where Z is the normalizing constant $Z = \sum_{x,y} e^{-E(x,y)}$. The probability of a data point (represented by the state v of the visible layer) is defined as the marginal: $P(v) = \sum_h P(v,h) = \frac{\sum_h e^{-E(v,h)}}{Z}$. Thus, the training data likelihood, using just one training point for simplicity, is $\phi = \log P(v) = \phi^+ - \phi^-$ where $\phi^+ = \log \sum_h e^{-E(v,h)}$ and $\phi^- = \log Z = \log \sum_{x,y} e^{-E(x,y)}$. The *positive* gradient $\frac{\partial \phi^+}{\partial w_{ij}}$ is simple: $\frac{\partial \phi^+}{\partial w_{ij}} = v_i \cdot P(h_j = 1|v)$. The *negative* gradient $\frac{\partial \phi^-}{\partial w_{ij}} = P(v_i = 1, h_j = 1)$, however, is intractable. If we could get samples from the model, we could Monte Carlo approximate it, but even getting those samples is intractable.

2.2. The Contrastive Divergence Gradient Approximation

To get a tractable approximation of $\frac{\partial \phi^-}{\partial w_{ij}}$, one uses some algorithm to *approximately* sample from the model. The Contrastive Divergence (CD) algorithm is one way to do this. It is designed in such a way that at least the *direction* of the gradient estimate is somewhat accurate, even when the size is not. CD-1 is, at present, the most commonly used algorithm for training RBMs. One of the algorithms we compare is regular CD-1; another is CD-10, which is generally considered to be better if the required computer time is available.

A variation on CD is *mean field* CD (Welling & Hinton, 2002), abbreviated MF CD. This has the advantage of being a deterministic gradient estimate, which means that larger learning rates can be used. We include mean field CD-1 in the comparison.

3. The Persistent Contrastive Divergence Algorithm

CD-1 is fast, has low variance, and is a reasonable approximation to the likelihood gradient, but it is still significantly different from the likelihood gradient when the mixing rate is low. This can be seen by drawing samples from the distribution that it learns (see Figure 4). Generally speaking, CD-n for greater n is preferred over CD-1, if enough running time is available. In Neal's 1992 paper about Sigmoid Belief Networks (1992), a solution is suggested for such situations. In the context of RBMs, the idea is as follows (see also (Yuille, 2004)).

What we need for approximating $\frac{\partial \phi^-}{\partial w_{ij}}$ is a sample from the model distribution. The standard way to get it is by using a Markov Chain, but running a chain for many steps is too time-consuming. However, between parameter updates, the model changes only slightly. We can take advantage of that by initializing a Markov Chain at the state in which it ended for the previous model. This initialization is often fairly close to the model distribution, even though the model has changed a bit in the parameter update. Neal uses this approach with Sigmoid Belief Networks to approximately sample from the posterior distribution over hidden layer states given the visible layer state. For RBMs, the situation is a bit simpler: there is only one distribution from which we need samples, as opposed to one distribution per training data point. Thus, the algorithm can be used to produce gradient estimates *online* or using mini-batches, using only a few training data points for the positive part of each gradient estimate, and only a few 'fantasy' points for the negative part. The fantasy points are updated by one full step of the Markov Chain each time a mini-batch is processed.

Of course this still is an approximation, because the model does change slightly with each parameter update. With infinitesimally small learning rate it becomes exact, and in general it seems to work best with small learning rates.

We call this algorithm *Persistent Contrastive Divergence* (PCD), to emphasize that the Markov Chain is not reset between parameter updates.

4. Experiments

We did a variety of experiments, using different data sets (digit images, emails, artificial data, horse image segmentations, digit image patches), different models (RBMs, classification RBMs, fully visible Markov Random Fields), different training procedures (PCD, CD-1, CD-10, MF CD, pseudo likelihood), and different tasks (unsupervised vs. supervised learning).

4.1. Data Sets

The first data set that we used was the MNIST dataset of handwritten digit images (LeCun & Cortes,). The images are 28 by 28 pixels, and the data set consists of 60,000 training cases and 10,000 test cases. To have a validation set, we split the official training set of 60,000 cases into a training set of 50,000 cases and a validation set of 10,000 cases. To have binary data, we treat the pixel intensities as probabilities. Each time a binary data point is required, a real-valued MNIST image is binarized by sampling from the given Bernoulli distribution for each pixel. Thus, in effect, our data set is a mixture of 70,000 factorial distributions: one for each of the data points in the MNIST data set.

Another data set was obtained by taking small patches of 5 by 5 pixels, from the MNIST images. To have somewhat smooth-looking data, we binarized by thresholding at 1/2. The 70,000 MNIST data points were thus turned into 70,000 times $(28 - 5 + 1)^2$ is 4,032,000 patches. This data set was split into training (60%), validation (20%), and test (20%) sets.

A data set consisting of descriptions of e-mails was made available by Sam Roweis. It describes 5,000 e-mails using a variety of binary features - mostly word presence vs. absence features. The e-mails are labeled as spam or non-spam.

An artificial data set was created by combining the outlines of rectangles and triangles. Because this data set is artificially generated, there is an infinite amount of it, which helps shed some light on the reasons for using weight decay regularization.

Lastly, we used a data set of image segmentations: in pictures of horses, the segmentation indicates which pixels are part of the horse and which are background (Borenstein et al., 2004). By using only the segmentation, we have a binary data set.

4.2. Models

The first model we used is an RBM, exactly as described above. For the MNIST and horse segmentation data sets, we used 500 hidden units; for the artificial data set we used 100.

One of the evaluations is how well the learned RBM models the test data, i.e. log likelihood. This is intractable for regular size RBMs, because the time complexity of that computation is exponential in the size of the smallest layer (visible or hidden). One experiment, therefore, was done using only 25 hidden units, so that log likelihood could be calculated exactly in about two hours. Another experiment uses an approximate assessment of the normalization constant Z, that was developed recently in our group (Salakhutdinov & Murray, 2008). This algorithm works for any number of hidden units, but its reliability has not been researched extensively. Nonetheless, it seems to give a reasonable indication, and can be used to complement other results.

RBMs, however, are models for unsupervised learning, so for classification we used a slightly different model, described in more detail in (Hinton et al., 2006). We used an RBM with one added visible unit, which represented the label. The training data points are then combinations of inputs with their labels, and testing is done by choosing the most likely label given the input, under the learned model. This model we call a 'classification RBM'. Note that the label unit is not necessarily binary (although in the spam classification task it is). In the MNIST classification task it is multinomial: it can have 10 different values. This, however, does not significantly change the algorithms (Hinton, 2002). For MNIST classification we used 500 hidden units; for spam classification we used 100.

The third model we tested is significantly different: a fully visible, fully connected Markov Random Field (MRF) (see for example (Wainwright & Jordan, 2003)). One can use the PCD algorithm on it, although it looks a bit different in this case. We compared its performance to the more commonly used Pseudo-Likelihood optimization algorithm (Besag, 1986). To have exact test data log likelihood measurements, we used small models, with only 25 units.

4.3. The Mini-batch Optimization Procedure

We used the *mini-batch* learning procedure: we only used a small number of training points for each gradient estimate. We used 100 training points in each mini-batch for most data sets.

4.4. Algorithm Details

The PCD algorithm can be implemented in various ways. One could, for example, choose to randomly reset some of the Markov Chains at regular intervals. Initial tests showed that the best implementation is as follows: no Markov Chains get reset; one full Gibbs update is done on each of the Markov Chains for each gradient estimate; and the number of Markov Chains is equal to the number of training data points in a mini-batch.

PCD for fully visible MRFs is a bit different from PCD for RBMs. A pleasant difference is that $\frac{\partial \phi^+}{\partial \theta}$ is constant, so it can be precomputed for the entire training set. Thus, no variance results from the use of mini-batches, and the training set can be discarded after $\frac{\partial \phi^+}{\partial \theta}$ is computed over it. An unpleasant difference is that the Markov Chain defined by Gibbs sampling has slower mixing: MRFs with connections between the visible units lack the pleasant property of RBMs that all visible units can be updated at the same time.

A Pseudo-Likelihood (PL) gradient computation requires more work than a PCD gradient computation, because it requires a logistic regression gradient estimate for each of the units. As a result, we found that using mini-batches of 50 training points instead of 100 took only a little bit more time per training point, and did allow updating the model parameters almost twice as often, which is preferable in the mini-batch optimization procedure.

4.5. Other Technical Details

The learning rates used in the experiments are not constant. In practice, decaying learning rates often work better. In these experiments, the learning rate was linearly decayed from some initial learning rate to zero, over the duration of the learning. Preliminary experiments showed that this works better than the $\frac{1}{t}$ schedule suggested in theoretical work by (Robbins & Monro, 1951), which is preferable when infinitely much time is available for the optimization.

Some experiment parameters, such as the number of hidden units, and the size of the mini-batches, were fixed. However, the initial learning rate was chosen using a validation set, as was weight decay for the (shorter) experiments on the spam, horses, MNIST patches, and artificial data sets. For each algorithm, each task, and each training duration, 30 runs were performed with evaluation on validation data, trying to find the settings that worked best. Then a choice of initial learning rate and, for the shorter experiments, weight decay, were made, and with those chosen settings, 10 more runs were performed, evaluating on test data. This provided 10 test performance numbers, which were summarized by their average and standard deviation (shown as error bars).

5. Results

5.1. The three MNIST Tasks

The results on the three MNIST tasks are shown in Figures 1, 2, and 3.

It is clear that PCD outperforms the other algorithms. PCD, CD-1, and MF CD all take approximately the same amount of time per gradient estimate, with MF CD being a little bit faster because it does not have to create random numbers. CD-10 takes about four times as long as PCD, CD-1, and MF CD, but it is indeed better than CD-1.

While CD-1 is good for some purposes, it is substantially different from the true likelihood gradient. This can be seen by drawing samples from an RBM that was trained with CD-1. Figure 4 shows those next to samples drawn from an RBM that was trained using PCD. It is clear that PCD is a better approximation

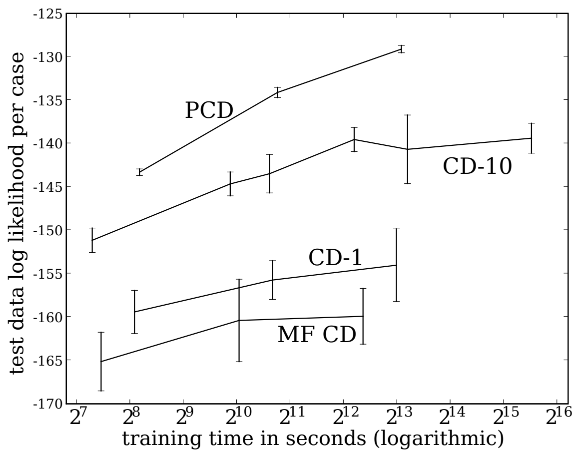

Figure 1. Modeling MNIST data with 25 hidden units (exact log likelihood)

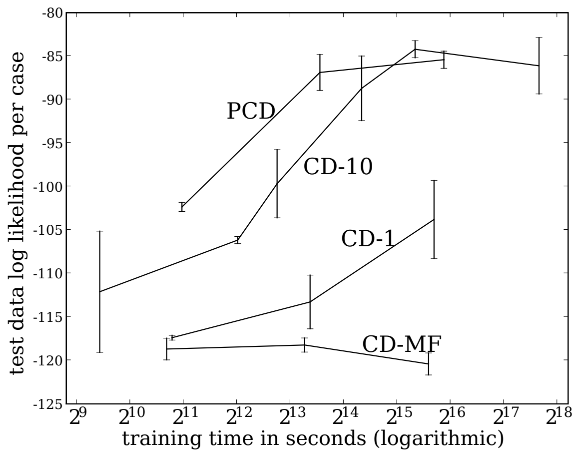

Figure 2. Modeling MNIST data with 500 hidden units (approximate log likelihood)

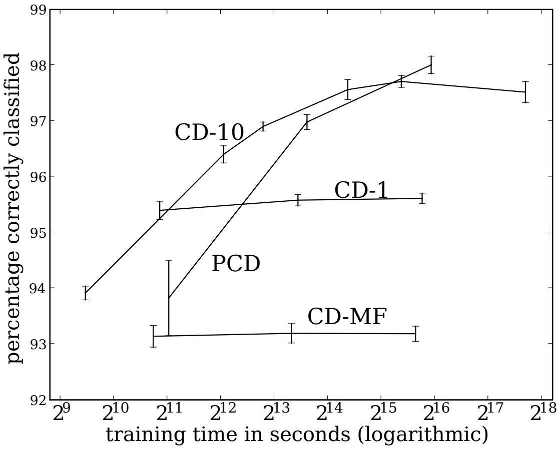

Figure 3. Classification of MNIST data

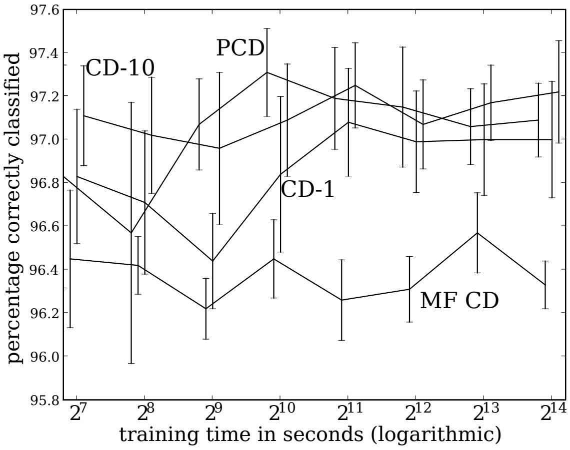

Figure 6. Classifying e-mail as spam versus non-spam

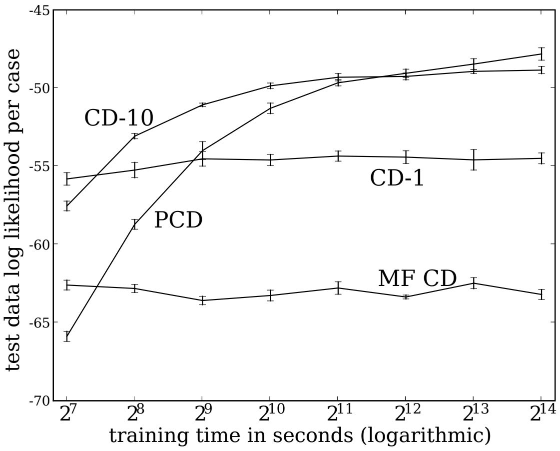

Figure 5. Modeling artificial data

to the likelihood gradient.

Classification is a particularly interesting task because it gives an indication of how well the model can extract relevant features from the input. RBMs are most often used as feature detectors, and this finding suggests that PCD creates feature detectors that give better classification than CD-1.

5.2. Modeling Artificial Data

In Figure 5 we see essentially the same as what happened on the MNIST tasks. MF CD is clearly the worst of the algorithms, CD-1 works better, and CD-10 and PCD work best, with CD-10 being preferable when little time is available and PCD being better if more time is available.

This data set was artificially generated, so there was an infinite amount of data available. Thus, one might think that the use of weight decay serves no purpose. However, all four algorithms did work best with some weight decay. The explanation for this is that CD algorithms are quite dependent on the mixing rate of the Markov Chain defined by the Gibbs sampler, and that mixing rate is higher when the parameters of the model are smaller. Thus, weight decay keeps the model mixing reasonably well, and makes CD algorithms work better. The effect is strongest for MF CD, which performs only one Gibbs update and does so without introducing noise. MF CD worked best with a weight decay strength of 10^{-3}. CD-1 does introduce some noise in the update procedure, and required less weight decay: $3 \cdot 10^{-4}$. CD-10 performs more updates, and is less dependent on the mixing rate. The best weight decay value for CD-10 turned out to be approximately $1.3 \cdot 10^{-4}$. Finally, the mixing mechanism used by PCD is even better, but it is still based on the Gibbs sampler, so it, too, works better with some weight decay. The best weight decay strength for PCD was approximately $2.5 \cdot 10^{-5}$.

5.3. Classifying E-mail Data

In Figure 6 the results on the e-mail classification task are shown. Because this is a small data set (5,000 data points in total, i.e. only 1000 test data points), we see that the error bars on the performace are quite large. Thus, we cannot carefully compare the performance of CD-1, CD-10, and PCD. We only see that MF CD is, again, not the best method.

However, we can conclude that RBMs can be used for this task, too, with acceptable performance, and that

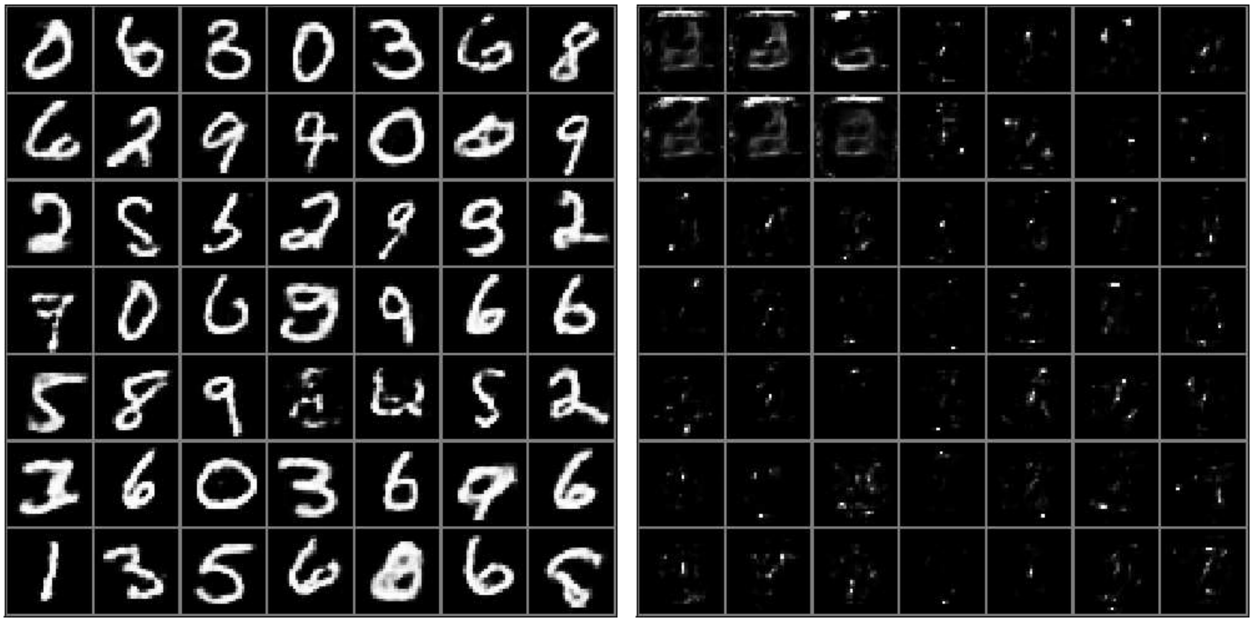

Figure 4. Samples from an RBM that was trained using PCD (left) and an RBM that was trained using CD-1 (right). Clearly, CD-1 did not produce an accurate model of the MNIST digits. Notice, however, that some of the CD-1 samples vaguely resemble a three.

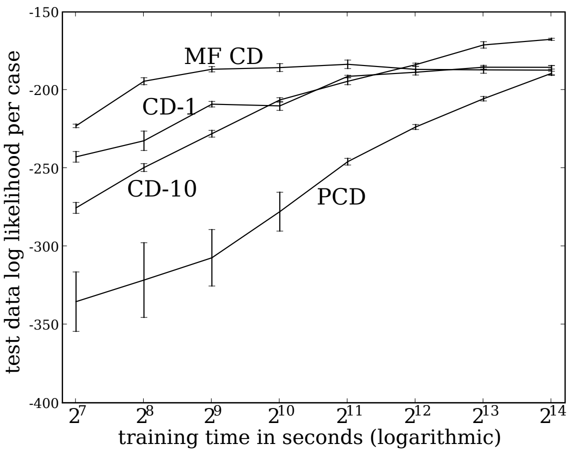

Figure 7. Modeling horse segmentation data

PCD is a reasonable choice of training algorithm.

5.4. Modeling Horse Contours

In Figure 7 we see a different picture: PCD is not the best algorithm here. The most plausible explanation is that although the same amount of training time was used, the data is much bigger: 1024 visible units, and 500 hidden units. Thus, there were 20 times as many connections in the RBM to be learned, which also means processing one mini-batch took more than 10 times as long as for the artificial data. Thus, we are essentially looking at a short optimization. Above, we already saw that CD-10 is better than PCD when little time is available, and that is confirmed here. We conjecture that, given significantly more training time, PCD would perform better than the other algorithms.

5.5. PCD on Fully Visible MRFs

To verify that PCD also works well with other models, we did some experiments with fully visible, fully connected MRFs. To be able to have exact test data likelihood evaluation, we made the MRFs small, and modeled 5 by 5 pixel patches from the MNIST digit images.

Pseudo-Likelihood (PL) training works reasonably well on this data set, but it does not produce the best probability models. Presumably this is simply because PL optimizes a different objective function. As a result, PL needed early stopping to prevent diverging too much from the data likelihood objective function, and the optimal learning rates are more or less inversely proportional to the duration of the optimization. Even with only a few seconds training time, the best test data likelihood is already achieved: -5.35.

PCD training does go more in the direction of the data likelihood function - asymptotically it gives its exact gradient. Thus, PCD did profit from having more time to run. Figure 8 shows the performance. The asymptotic value of approximately -5.15 does seem to be the best possible model: we also used exact gradient

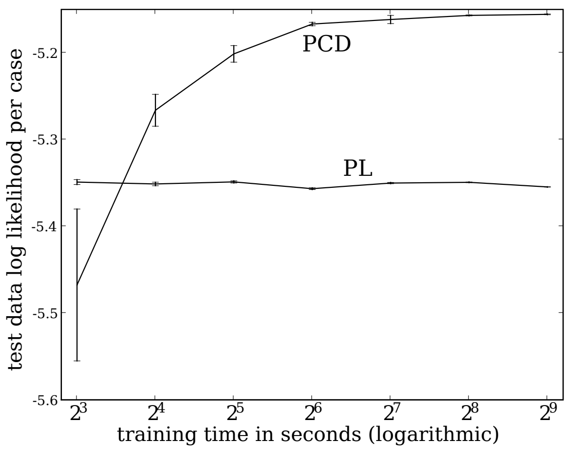

Figure 8. Training a fully visible MRF

optimization (which is slow, but possible), and this equally ended up with test data log likelihood of -5.15. However, the entropy of the training data distribution is significantly less than 5.15 'nats': it is 4.78 nats. This difference is probably due to the fact that the model has insufficient complexity to completely learn the training data distribution.

Incidentally, the training data log likelihood is only 0.004 better than the test data log likelihood - presumably because this data set is quite large and the model is quite small.

6. Discussion and Future Work

One issue not investigated is the use of weight decay. It is quite possible that the more approximate algorithms (such as CD-1 and MF CD) would benefit more from weight decay than CD-10 and PCD. In an RBM with zero weights, CD-1 and MF CD give exactly the likelihood gradient, and in general, in RBMs with small weights those algorithms give better approximations to the likelihood gradient than in RBMs with large weights. Weight decay keeps the weights small, and thus enables gradient estimates that approximate the likelihood gradient more closely. For many tasks, however, large weights may be required for good performance, so strong weight decay is undesirable if it can be avoided.

Also, the amount of training time used in these experiments is insufficient to find the asymptotic performance. In Figure 3 one can see, for example, that PCD clearly profits from more training time. To find out what its performance would be with more training time is future work, but we have seen runs (with more training time and more hidden units) where as few as 104 out of the 10,000 test cases were misclassified. Clearly, this is worth investigating further.

Another issue suggesting future work is that the classification RBMs in these experiments were not trained to maximize classification performance. They were trained to accurately model the joint distribution over images and labels. It is possible to train classification RBMs directly for classification performance; the gradient is fairly simple and certainly tractable. A natural way to use this classification error gradient is *after* training the RBM for joint density modeling. However, in preliminary experiments we found that this procedure begins to overfit very quickly (often after improving performance by less than 0.1%), so we did not include it in this paper. It is, however, still possible that *combining* the classification gradient with the density modeling gradient is a method that could yield more improvements. This is future work.

The main limitation of PCD is that it appears to require a low learning rate in order to allow the "fantasy" points to be sampled from a distribution that is close to the stationary distribution for the current weights. A theoretical analysis of this requirement can be found in (Yuille, 2004) and (Younes, 1999). Some preliminary experiments, however, suggest that PCD can be made to work well even when the learning rate is much larger than the one suggested by the asymptotic justification of PCD and we are currently exploring variations that allow much larger learning rates.

Acknowledgements

I thank Geoffrey Hinton and Ruslan Salakhutdinov for many useful discussions and helpful suggestions. Nikola Karamanov and Alex Levinshtein helped by providing data sets. The anonymous reviewers also provided many useful suggestions. This research was supported by NSERC and Microsoft.

References

Bengio, Y., & Delalleau, O. (2007). *Justifying and generalizing contrastive divergence* (Technical Report 1311). Université de Montréal.

Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., & Montreal, Q. (2007). Greedy Layer-Wise Training of Deep Networks. *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference.*

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B*, *48*, 259–302.

Borenstein, E., Sharon, E., & Ullman, S. (2004). Combining Top-Down and Bottom-Up Segmentation. *Computer Vision and Pattern Recognition Workshop, 2004 Conference on*, 46–46.

Carreira-Perpinan, M., & Hinton, G. (2005). On contrastive divergence learning. *Artificial Intelligence and Statistics, 2005*.

Gehler, P., Holub, A., & Welling, M. (2006). The rate adapting poisson model for information retrieval and object recognition. *Proceedings of the 23rd international conference on Machine learning*, 337–344.

Hinton, G. (2002). Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, *14*, 1771–1800.

Hinton, G., & Salakhutdinov, R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, *313*, 504–507.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. *Proceedings of the 24th international conference on Machine learning*, 473–480.

LeCun, Y., & Cortes, C. The MNIST database of handwritten digits.

Neal, R. (1992). Connectionist learning of belief networks. *Artificial Intelligence*, *56*, 71–113.

Robbins, H., & Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, *22*, 400–407.

Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. *Proceedings of the 24th international conference on Machine learning*, 791–798.

Salakhutdinov, R., & Murray, I. (2008). On the quantitative analysis of deep belief networks. *Proceedings of the International Conference on Machine Learning*.

Smolensky, P. (1986). *Information processing in dynamical systems: foundations of harmony theory.* MIT Press Cambridge, MA, USA.

Wainwright, M., & Jordan, M. (2003). Graphical models, exponential families, and variational inference. *UC Berkeley, Dept. of Statistics, Technical Report*, *649*.

Welling, M., & Hinton, G. (2002). A New Learning Algorithm for Mean Field Boltzmann Machines. *Artificial Neural Networks-Icann 2002: International Conference, Madrid, Spain, August 28-30, 2002: Proceedings*.

Welling, M., Rosen-Zvi, M., & Hinton, G. (2005). Exponential family harmoniums with an application to information retrieval. *Advances in Neural Information Processing Systems*, *17*, 1481–1488.

Younes, L. (1999). On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics An International Journal of Probability and Stochastic Processes*, *65*, 177–228.

Yuille, A. (2004). The Convergence of Contrastive Divergences. *Advances in Neural Information Processing Systems*, *3*, 4.