NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning, Whistler, Canada, December 2010

Investigating Convergence of Restricted Boltzmann Machine Learning

Hannes Schulz **Andreas Müller**[*] **Sven Behnke**

Computer Science VI, Autonomous Intelligent Systems Group
University of Bonn
Römerstraße 164, 53117 Bonn, Germany
{schulz,amueller,behnke}@ais.uni-bonn.de

Abstract

Restricted Boltzmann Machines are increasingly popular tools for unsupervised learning. They are very general, can cope with missing data and are used to pretrain deep learning machines. RBMs learn a generative model of the data distribution. As exact gradient ascent on the data likelihood is infeasible, typically Markov Chain Monte Carlo approximations to the gradient such as Contrastive Divergence (CD) are used. Even though there are some theoretical insights into this algorithm, it is not guaranteed to converge. Recently it has been observed that after an initial increase in likelihood, the training degrades, if no additional regularization is used. The parameters for regularization however cannot be determined even for medium-sized RBMs. In this work, we investigate the learning behavior of training algorithms by varying minimal set of parameters and show that with relatively simple variants of CD, it is possible to obtain good results even without further regularization. Furthermore, we show that it is not necessary to tune many hyperparameters to obtain a good model – finding a suitable learning rate is sufficient. Fast learning, however, comes with a higher risk of divergence and therefore requires a stopping criterion. For this purpose, we investigate the commonly used Annealed Importance Sampling, an approximation to the true log likelihood of the data and find that it completely fails to discover divergence in certain cases.

1 Introduction

Restricted Boltzmann Machines (RBMs, [1]) have been widely used as generative models, for unsupervised feature extraction and as building blocks of deep belief networks [2, 3]. Applications range from image processing [4] and classification [5] to collaborative filtering [6]. Despite this success RBM training remains a problematic task. For even medium-sized RBMs likelihood maximization is not possible because the true gradient of the likelihood is not tractable.

Most applications instead rely on a fast Markov chain Monte Carlo (MCMC) approximation to the gradient, called Contrastive Divergence (CD), proposed by Hinton [5]. CD was shown to work well in practice in a number of tasks, even though it is not a good approximation to the likelihood gradient [7].

There are a number of variants of CD, notably Persistent CD [8] (PCD), Fast Persistent CD [9], Tempered Transitions [10], and Parallel Tempering [11, 12]. Most of these come with

[*] This work was supported in part by NRW State within the B-IT Research School.

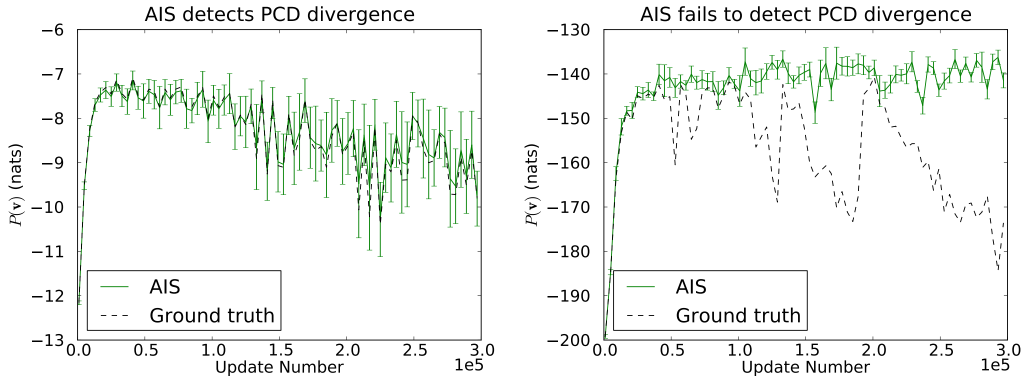

Figure 1: AIS (solid green with average uncertainty) vs. ground truth (black, dashed). **Left.** AIS follows true likelihood even during unstable learning. **Right.** AIS fails to detect PCD learning divergence.

a variety of hyperparameters in addition to the more common heuristics of weight-decay, momentum, and learning rate schedules. Since exact evaluation of the objective function is infeasible for interesting datasets, it is not clear which heuristic to choose and how to set the hyperparameters. Empirical evaluations exist [11, 9] but are scarce on realistic datasets.

In Fischer et al. [13], Dejardins et al. [11] and others, it was even observed that CD (and PCD) training diverges on the training set after an initial increase in likelihood. In [13], it was concluded that the right choice of hyperparameters can solve this problem. Due to the problem of calculating the partition function, however, it is not clear how the best training method and hyperparameters could be chosen.

Please also note that the effects discussed in this paper are not related to the common "overfitting" phenomenon. This can be dealt with by choosing a validation set and monitoring the ratio of the unnormalized probabilities, causing the partition functions to cancel. What we are looking at in this paper is whether the training method actually follows the gradient on the training set as opposed to generalization on a validation set.

To our knowledge, there are two main methods in use today to evaluate the learning progress of RBMs. One is the so-called "reconstruction error", the other is Annealed Importance Sampling (AIS, [14]). In the practical guide to training RBMs [15], Hinton both refers researchers to the reconstruction error but also warns them to rely on it. The "reconstruction error" is the difference between a data point and the "reconstruction", i.e. the expected value of the visible nodes, given the expected value of the hidden nodes, given the data point. It was found in [13], that this measure is truly dangerous, since it does not correlate with the objective function of RBM training and in particular does not detect the divergence of likelihood. In this paper, we can confirm this observation for more realistically sized RBMs.

The second commonly used method to evaluate RBMs is Annealed Importance Sampling (AIS [14, 7]). AIS is an MCMC method that can be used to approximate the partition function of an RBM with the help of a baseline model. We investigate the use of AIS, not only to judge the final result of learning but also to find good hyperparameters and as an indicator when to stop learning to prevent divergence.

The main observations in this paper can be summarized as follows: By analyzing detailed learning curves on medium-sized RBMs, we find that using PCD and a simple update rule suffices to produce high likelihood values. In particular, it is not necessary to tune many hyperparameters or to find the right learning schedule. Prevention of divergence remains difficult, however, since results on AIS approximations are rather mixed. In most cases, the behaviour of the true likelihood was reproduced accurately but in other cases serious divergence was not detected at all. Theoretical work by Yuille [16] shows that CD is guaranteed to converge to a local maximum when an appropriate learning rate schedule is used. Whether this can be used in practice is not clear, as too conservative learning rate schedules result in convergence to low likelihood values [13]. It was also shown that it is NP-hard to approximate the likelihood of a given RBM to a certain precision [17]. We therefore suggest more research in the direction of early stopping and detection of divergence.

2 Background on Restricted Boltzmann Machines

A Restricted Boltzmann Machine (RBM) is an undirected graphical model with binary observed variables $\mathbf{v} \in \{0,1\}^n$ (visible nodes) and binary latent variables $\mathbf{h} \in \{0,1\}^m$ (hidden nodes). The energy function of an RBM is given by

$$E(\mathbf{v}, \mathbf{h}, \theta) = -\mathbf{v}^T W \mathbf{h} - \mathbf{b}^T \mathbf{v} - \mathbf{a}^T \mathbf{h}, \tag{1}$$

where $\theta = (W, \mathbf{b}, \mathbf{a})$ are the model parameters, namely pairwise visible-hidden interaction weights and biases of visible and hidden activation potentials, respectively. This yields a probability distribution

$$p(\mathbf{v};\theta) = \frac{1}{Z(\theta)} p^*(\mathbf{v};\theta) = \frac{1}{Z(\theta)} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h},\theta)},$$

where $Z(\theta)$ is the normalizing constant (partition function) and $p^*(\cdot)$ denotes unnormalized probability. The conditional distributions $p(\mathbf{v}|\mathbf{h})$ and $p(\mathbf{h}|\mathbf{v})$ factorize completely, making exact inference of the respective posteriors possible. Their expected values are given by

$$\langle \mathbf{v} \rangle_p = \sigma(W\mathbf{h} + \mathbf{b}) \qquad \text{and} \qquad \langle \mathbf{h} \rangle_p = \sigma(W\mathbf{v} + \mathbf{b}). \tag{2}$$

Here, σ denotes element-wise application of the logistic sigmoid: $\sigma(x) = (1 + \exp(-x))^{-1}$. In practice, Contrastive Divergence (CD, [5]) or one of its variants is used to approximate the true parameter gradient

$$\frac{\partial \ln p(\mathbf{v})}{\partial W} = \langle \mathbf{v}\mathbf{h}^T \rangle_+ - \langle \mathbf{v}\mathbf{h}^T \rangle_-$$

by a MCMC algorithm. Here, $\langle\cdot\rangle_+$ and $\langle\cdot\rangle_-$ refer to the expected values with respect to the data distribution and model distribution, respectively. The expected value of the data distribution is approximated in the “positive phase”, while the expected values of the model distribution are approximated in the “negative phase”. For CD in RBMs, $\langle\cdot\rangle_+$ can be calculated in closed form, while $\langle\cdot\rangle_-$ is estimated using k steps of a Markov chain started at the training data.

Recently, Tieleman [8] proposed a faster alternative to CD, called Persistent Contrastive Divergence (PCD), which employs a persistent Markov chain to approximate $\langle\cdot\rangle_-$. This is done by maintaining a set of “fantasy particles” $\mathbf{v}_-$, $\mathbf{h}_-$ during the whole training. The chains are also governed by the transition operator in Equation (2) and are used to calculate $\langle \mathbf{v}\mathbf{h}^T \rangle_-$ as the expected value with respect to the Markov chains $\langle \mathbf{v}_- \mathbf{h}_-^T \rangle$.

If the learning rate is small enough, the chains $\mathbf{v}_-$ and $\mathbf{h}_-$ should mix faster than the model changes. Therefore they form a better estimate of the model distribution than a k-step Gibbs sampling as performed by CD. As a side effect, PCD removes k from the set of hyperparameters to be adjusted and again emphasizes the importance of the learning rate.

RBMs can be stacked to build hierarchical models. The training of stacked models proceeds greedily layer-wise. After training an RBM, one calculates the expected values $\langle \mathbf{h} \rangle_{p(\mathbf{h}|\mathbf{v})}$ of its hidden variables given the training data. Keeping the parameters of the first RBM fixed, we can then train another RBM using $\langle \mathbf{h} \rangle_{p(\mathbf{h}|\mathbf{v})}$ as its input. We do not directly investigate stacking but concentrate on the learning of a single layer, as the results can be directly applied to the stacked setting.

Annealed Importance Sampling (AIS) can be used to obtain an approximation of the partition function of an RBM. It is an algorithm to estimate the ratio of two normalization constants, and builds upon the following fact. Let $p_A(v) = p_A^*(v)/Z_A$ and $p_B(v) = p_B^*(v)/Z_B$ be two distributions such that $p_A(v) \neq 0$ if $p_B(v) \neq 0$. Then:

$$\frac{Z_B}{Z_A} = \frac{\int p_B^*(\mathbf{v})d\mathbf{v}}{Z_A} = \int \frac{p_B^*(\mathbf{v})}{p_A^*(\mathbf{v})} p_a(\mathbf{v}) d\mathbf{v} = \left\langle \frac{p_B^*(\mathbf{v})}{p_A^*(\mathbf{v})} \right\rangle_{p_A}$$

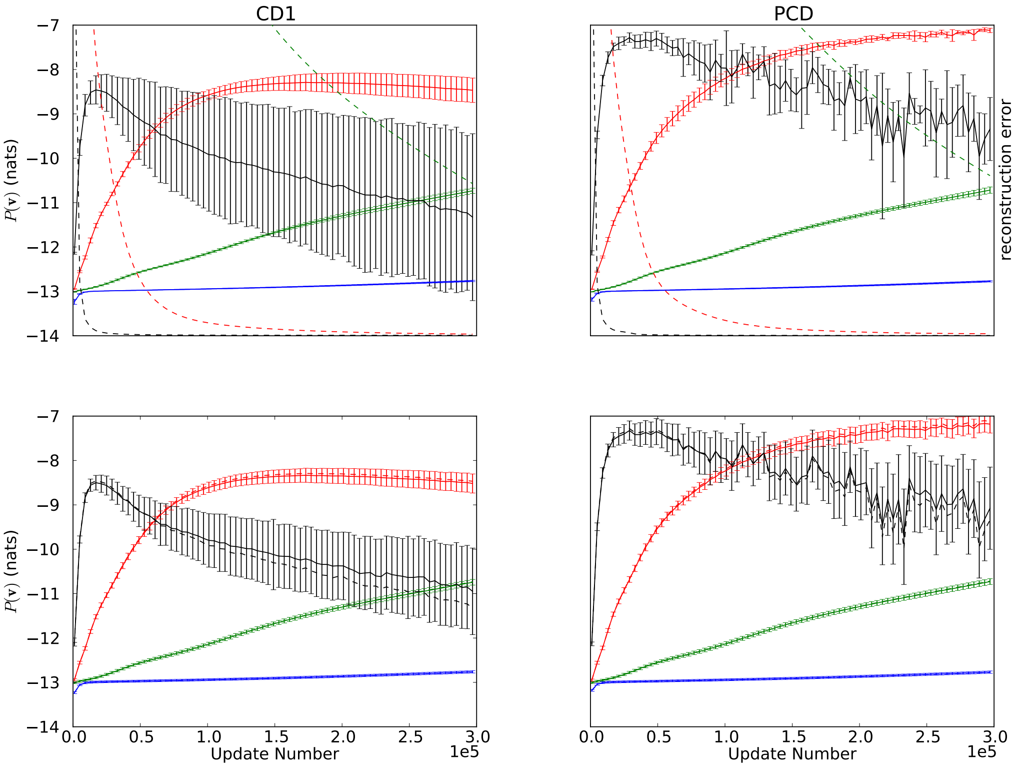

Figure 2: Learning curves for Shifter dataset: **Top row.** Exact likelihood during training, with standard deviation for different random initializations. Dashed lines show "reconstruction error". **Bottom row.** AIS approximations to likelihood with error bars showing mean uncertainty. Colors indicate different learning rates η: black $\eta = 0.1$, red $\eta = 0.01$, green $\eta = 0.001$, blue $\eta = 0.0001$. See main text for a detailed discussion.

If is is possible to draw independent samples from p_A, then this expected value can be approximated using a Monte Carlo approach. This only gives good approximations if p_A is very close to p_B.

AIS overcomes this weakness by introducing an annealing chain of distributions p_n such that $p_0 = p_A$, $p_N = p_B$ and p_k is very close to p_{k+1}. Calculating the ratios of all intermediate normalization functions – which can be done efficiently using an MCMC algorithm – then yields the desired ratio Z_B/Z_A.

As detailed in [7], AIS can be applied to calculating the partition function of an RBM by setting p_B to a distribution for which the normalization constant can be computed effiently and setting p_A to the distribution modeled by the RBM.

3 Experimental Setup

We use three datasets in this paper, which we chose for comparability with the literature.

Shifter. Labeled Shifter Ensemble [13] is a 19-dimensional data set containing 768 samples. The samples are generated in the following way: The states of the first eight visible units are set uniformly at random. The states of the following eight units are cyclically shifted copies of the first eight. The shift can be zero, one unit to the left, or one to the right and is indicated by the last three units. The average log-likelihood is $\log \frac{1}{768} \approx -6.64$ if the distribution of the data set is modeled perfectly.

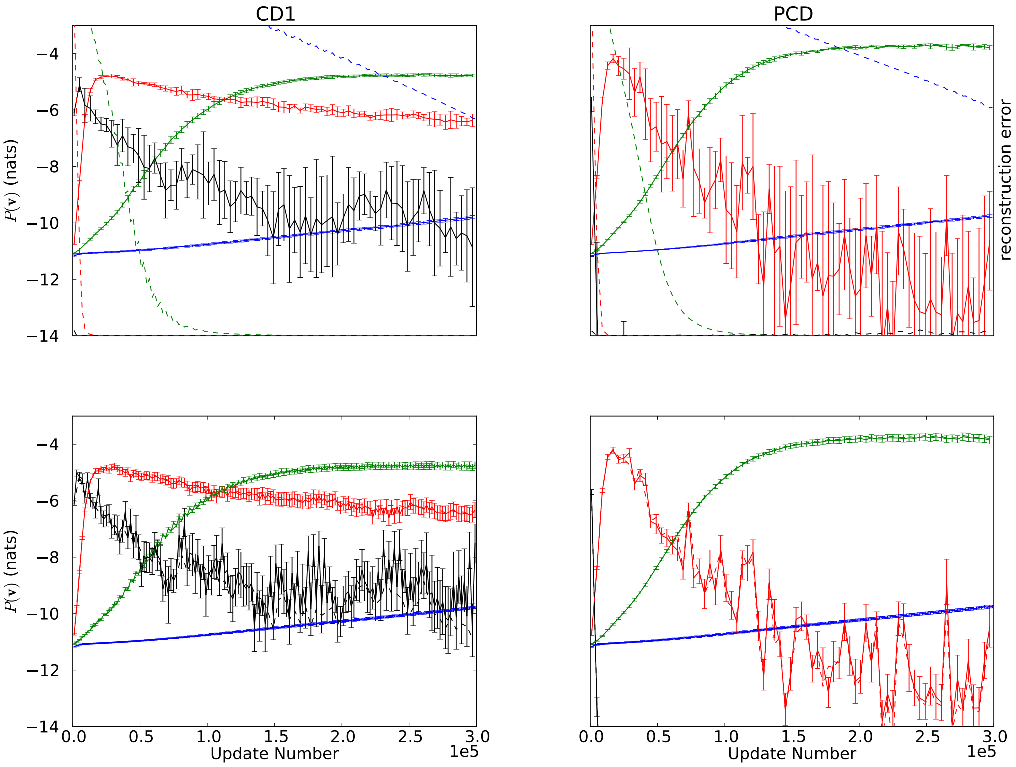

Figure 3: Learning curves for Bars and Stripes. Color coding as in Figure 2.

Bars and Stripes. This dataset also stems from [13] and has 16 visible units. Each pattern corresponds to a square of 4 × 4 units and is generated by first randomly choosing an orientation, vertical or horizontal with equal probability, and then picking the state for all units of every row or column uniformly at random. Since each of the two completely uniform patterns can be generated in two ways, the upper bound of the average log-likelihood is -3.21.

MNIST. Finally, we use the MNIST database of handwritten digits[1], which is more realistic dataset than the first two. If not by itself, it certainly has gained relevance through heavy use for evaluation of new learning algorithms.

Determining the log-likelihood. Due to the symmetric structure of the RBM energy function with respect to $\mathbf{v}$ and $\mathbf{h}$, the likelihood can be factored in two different ways:

$$\begin{aligned}\log p(\mathbf{v};\theta) &= -Z(\theta) + \log \sum_{\mathbf{h}} \exp\left(-E(\mathbf{v},\mathbf{h};\theta)\right) \\ &= -Z(\theta) + \left(\mathbf{b}^T\mathbf{v}\right) + \sum_{j=1}^{n} \log\left(1+\exp\left(a_j + \sum_{i=1}^{m} w_{ij}v_j\right)\right) \qquad (3) \\ &= -Z(\theta) + \left(\mathbf{a}^T\mathbf{h}\right) + \sum_{j=1}^{m} \log\left(1+\exp\left(b_j + \sum_{i=1}^{n} w_{ji}h_j\right)\right) \qquad (4)\end{aligned}$$

Depending on the dimensions of W, the larger of $\mathbf{h}$ and $\mathbf{v}$ is summed out (Equations (3) or (4), respectively). For $Z(\theta) = \sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp\left(-E(\mathbf{v},\mathbf{h};\theta)\right)$, all possible values of $\mathbf{v} \in \{0,1\}^n$ or $\mathbf{h} \in \{0,1\}^m$ must be considered.

[1] `http://yann.lecun.com/exdb/mnist/`

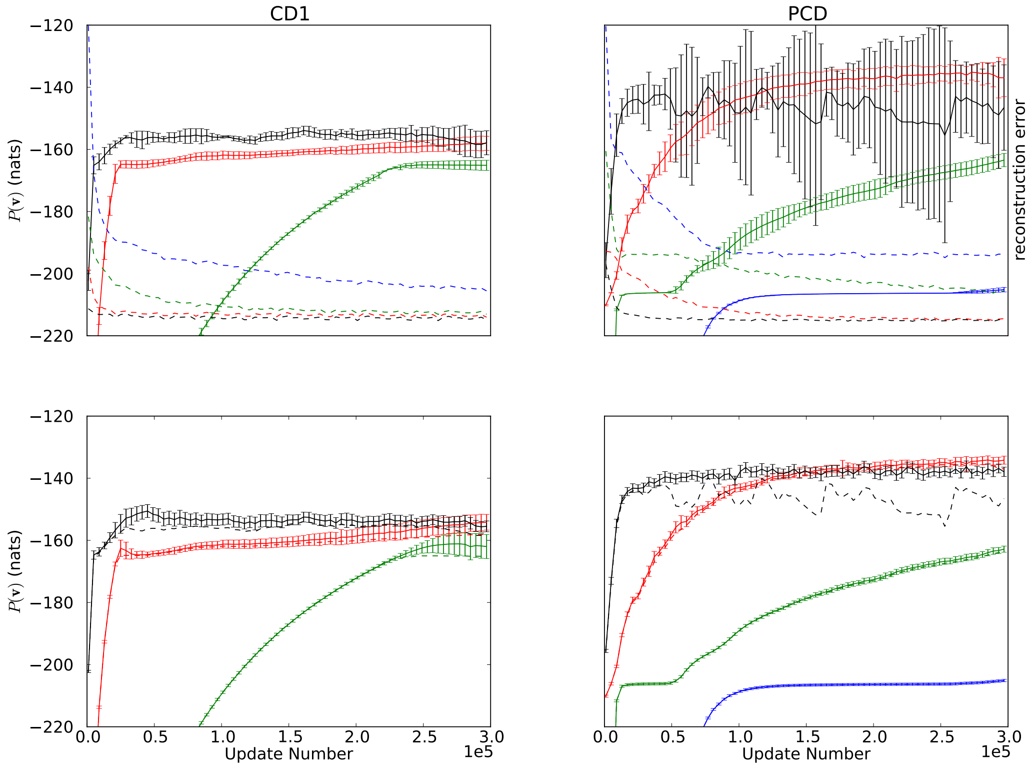

Figure 4: Learning curves for MNIST for varying learning rates. Color coding as in Figure 2. Note especially the divergence of AIS and the ground truth likelihood in the plot on the lower right.

Details on Learning Procedure. Since the Shifter and Bars and Stripes datasets are quite small, we use true batch learning. Minibatch learning gave similar results which are not shown here. For the MNIST dataset we use minibatches of size 400. Other batch sizes gave similar results, small batch sizes just result in less stable learning curves. No weight-decay, learning rate schedule, momentum or sparsity bias was used in either case.

For PCD, we used as many chains as there were samples in a batch. This was mainly done for convenient implementation. Since varying the batch size did not have significantly influence on our main observations, PCD does not seem to be very sensitive to the particular number of persistent chains used.

Details on Annealed Importance Sampling. As a base model for AIS, we used the standard procedure of modeling each visible unit as independent. This corresponds to a energy model consisting only of the bias term $E(\mathbf{v}, \theta) = -\mathbf{b}^T\mathbf{v}$. The maximum likelihood solution for $\mathbf{b}$ is then given by: $\mathbf{b} = \log(\bar{\mathbf{v}}) - \log(1 - \bar{\mathbf{v}})$, where $\bar{\mathbf{v}}$ is the mean of $\mathbf{v}$ over the dataset.

We initialized $\mathbf{b}$ to a smoothed version of the maximum likelihood solution, by using $\bar{\mathbf{v}}' = \bar{\mathbf{v}} + 0.1$. This heuristic was chosen on MNIST and kept unchanged for the other datasets. The schedule employed for the parameter β was proposed in [7], where β was taken uniformly spaced in three intervals. Specifically, we use 500 $\beta \in [0, 0.5[$, then 4000 $\beta \in [0.5, 0.9[$ and 10.000 $\beta \in [0.9, 1.0]$ for a total of 14.500 intermediate distributions. Using β that were uniformly distributed in $[0, 1]$ did not produce significantly different results. We used 512 parallel Markov chains for a stable approximation.

Empirical Verification of Implementation. Since the calculations for training and evaluation of the true and approximate (AIS) log-likelihood are quite demanding, we parallelize them using the NVIDIA CUDA framework and the CUV python library [18]. Training and evaluation jobs are further distributed over a cluster of 25 GPUs on eight computers. A complete learning curve for MNIST with 25 hidden units can thereby be generated in about ten minutes. Two adjustments were made to ensure numerical stability. Firstly, the large sums in Equations (3) and (4) are calculated using the Kahan algorithm on CPU and logarithmic summing on GPU. Secondly, we set $\log(1 + \exp(x)) := x$ if $x > -\log(\epsilon)$, where ϵ is machine precision for double (CPU) or float (GPU). We calculated the true average log-likelihood of MNIST for a trained RBM both on GPU and on CPU and consistently find that the results differ only after the fifth decimal place.

4 Results

During training, we save the weights every 4000 weight updates and determine the ground truth log-likelihood, the estimated log-likelihood using AIS, norm of weight matrix and the reconstruction error. All data shown is for RBMs with 25 (24) hidden units for MNIST (Bars and Stripes and Shifter). We repeated each experiment with five different random initializations except the special case for MNIST where AIS does not follow the ground truth, which was repeated 10 times to ensure that this is a repeatable observation.

Likelihood development over training. In both, PCD and CD1, learning curves were strongly dependent on the learning rate. Further parameters (learning rate schedules, weight decay) are hard to set and may lead to convergence to suboptimal results [13]. We find that we can obtain results which compare favorably with the literature without the use of these methods.

With regards to learning rates, we observe that for all datasets (top rows of Figures 2, 3 and 4) learning speed correlates with learning rate. In contrast to, for example, neural networks, large learning rates in RBM training do not result in instable learning and bad optima but in divergence of the log-likelihood function after achieving good likelihood values. This divergence of RBM training has been observed before (e. g. [13, 11]) and we could reproduce this effect in larger RBMs on MNIST (Figure 4, top row). The divergence effect is more pronounced for high learning rates. This is expected for PCD, since PCD requires “small” learning rates, so that the persistent chains stay close to the current model distribution. Still, large learning rates reached good solutions quickly before training diverged. We also observed divergence for smaller learning rates when training was carried on long enough (data not shown).

In general, learning with higher learning rates is more dependent on the seed (see e. g. the upper left plot of Figure 2).

The optimum in PCD learning is consistently larger than the optimum in CD1 learning, which is conforming to the literature (e. g. [8]).

AIS approximation of log-likelihood. Generalizing the above observations and accepting the divergence of RBM training algorithms, we would like to choose a learning rate which is large and stop before the likelihood starts to diverge. As can be observed in all plots, the “reconstruction error” (dashed lines in top row) mentioned before is of no help at all. The value always decreases. We therefore consider AIS as an evaluation method of the learning process.

For the toy datasets (Bars and Stripes, Shifter) and also for most cases in the MNIST dataset, AIS approximates the ground truth likelihood accurately. This is even the case when learning is very unstable (e. g. left plot in Figure 1). We also measured the uncertainty of AIS. The error bars in the bottom rows of Figures 2, 3 and 4 show the uncertainty of AIS averaged over learning trials with varying random seeds. As long as learning is stable, the uncertainty is very small. Surprisingly, however, for various trials ground truth likelihood dropped dramatically, while AIS completely failed to either capture the change in likelihood or to increase uncertainty (see right of Figure 1 for single run and bottom right of Figure 4 for the average over trials). We also observed this behavior for smaller learning rates (which diverge later) on MNIST, but not at all on the toy datasets. Therefore generalizations from small models to larger ones should be taken with a grain of salt. To investigate the reason for the drop in likelihood, we examined the persistent chains at the problematic update steps, but found no obvious deficiency in the mixing. Therefore, AIS, while it models easy cases perfectly and some hard cases very well should be used cautiously for such purposes as stopping learning, finding hyper-parameters and evaluating new learning algorithms.

5 Conclusions

While RBMs are successful learning machines, their training remains a tricky task. We evaluated training methods with minimal parameter sets on small to medium-sized problems to analyze the behavior of CD1 and PCD training with respect to parameter selection and divergence. Our findings suggest that often a simple setup provides good results, provided one finds a suitable learning rate. We can confirm divergence of CD1 and PCD learning algorithms and therefore investigated AIS as a stopping and evaluation method. The presented results suggest that AIS is often, but not always, a good measure of the training progress and we suggest further investigation into alternative criteria.

Acknowledgements We thank Asja Fischer for valuable suggestions and discussions.

References

[1] P. Smolensky, Information processing in dynamical systems: Foundations of harmony theory, in: D. Rumelhart, J. McClelland (Eds.), Parallel Distributed Processing, Vol. 1, MIT Press, Cambridge, 1986, Ch. 6, pp. 194–281.

[2] Y. Bengio, Learning Deep Architectures for AI, Foundations and Trends in Machine Learning 2 (2009) 1–127.

[3] R. Salakhutdinov, G. Hinton, Deep Boltzmann Machines, in: Converence on Artificial Intelligence and Statistics (AIStats), Vol. 5, 2009, pp. 448–455.

[4] M. Ranzato, G. Hinton, Modeling Pixel Means and Covariances using Factorized Third-order Boltzmann Machines, in: Proceedings of Computer Vision and Pattern Recognition (CVPR), 2010, pp. 2551–2558.

[5] G. Hinton, S. Osindero, W. Teh, A fast learning algorithm for deep belief nets, Neural Computation 18 (7) (2006) 1527–1554.

[6] R. Salakhutdinov, A. Mnih, G. Hinton, Restricted Boltzmann Machines for Collaborative Filtering, in: Proceedings of the 24th international conference on Machine learning, 2007, pp. 791–798.

[7] R. Salakhutdinov, I. Murray, On the quantitative analysis of deep belief networks, in: Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 872–879.

[8] T. Tieleman, Training restricted Boltzmann machines using approximations to the likelihood gradient, in: International Conference on Machine Learning (ICML), 2008, pp. 1064–1071.

[9] T. Tieleman, G. Hinton, Using fast weights to improve persistent contrastive divergence, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 1033–1040.

[10] R. Salakhutdinov, Learning in markov random fields using tempered transitions, in: Advances in Neural Information Processing Systems (NIPS) 22, 2009, pp. 1598–1606.

[11] G. Desjardins, A. Courville, Y. Bengio, P. Vincent, O. Dellaleau, Parallel tempering for training of restricted Boltzmann machines, in: Journal of Machine Learning Research Workshop and Conference Proceedings, Vol. 9, 2010, pp. 145–152.

[12] K. Cho, T. Raiko, A. Ilin, Parallel Tempering is Efficient for Learning Restricted Boltzmann Machines, in: Proceedings of International Joint Conference on Neural Networks (IJCNN), 2010.

[13] A. Fischer, C. Igel, Empirical Analysis of the Divergence of Gibbs Sampling Based Learning Algorithms for Restricted Boltzmann Machines, Proceedings of the International Conference on Artificial Neural Networks (ICANN) (2010) 208–217.

[14] R. Neal, Annealed importance sampling, Statistics and Computing 11 (2) (2001) 125–139.

[15] G. Hinton, A Practical Guide to Training Restricted Boltzmann Machines, Tech. rep., University of Toronto (2010).

[16] A. Yuille, The Convergence of Contrastive Divergences, in: L. K. Saul, Y. Weiss, L. Bottou (Eds.), Advances in Neural Information Processing Systems (NIPS) 17, 2005, pp. 1593–1600.

[17] P. Long, R. Servedio, Restricted Boltzmann Machines are Hard to Approximately Evaluate or Simulate, in: Proceedings of the 27th International Conference on Machine Learning, 2010, pp. 703–710.

[18] A. Müller, H. Schulz, S. Behnke, Topological Features in Locally Connected RBMs, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN), 2010.