Tempered Markov Chain Monte Carlo for training of Restricted Boltzmann Machines

Desjardins, G., Courville, A., Bengio, Y., Vincent, P. and Delalleau, O.

Technical Report 1345, Dept. IRO, Université de Montréal

Abstract

Alternating Gibbs sampling is the most common scheme used for sampling from Restricted Boltzmann Machines (RBM), a crucial component in deep architectures such as Deep Belief Networks. However, we find that it often does a very poor job of rendering the diversity of modes captured by the trained model. We suspect that this hinders the advantage that could in principle be brought by training algorithms relying on Gibbs sampling for uncovering spurious modes, such as the Persistent Contrastive Divergence algorithm. To alleviate this problem, we explore the use of tempered Markov Chain Monte-Carlo for sampling in RBMs. We find both through visualization of samples and measures of likelihood that it helps both sampling and learning.

1 Introduction and Motivation

Restricted Boltzmann Machines Smolensky [1986], Freund and Haussler [1994], Hinton [2002], Welling et al. [2005] have attracted much attention in recent years because of their power of expression Le Roux and Bengio [2008], because inference (of hidden variables h given visible variables x) is tractable and easy, and because they have been used very successfully as components in deep architectures Bengio [2009] such as the Deep Belief Network Hinton et al. [2006]. Both generating samples and learning in most of the literature on Restricted Boltzmann Machines (RBMs) rely on variations on alternating Gibbs sampling, which exploits the bipartite structure of the graphical model (there are links only between visible and hidden variables, but not between visible or between hidden variables). RBMs and other Markov Random Fields and Boltzmann machines are energy-based models in which we can write $p(x) \propto e^{-\mathrm{Energy}(x)}$. The log-likelihood gradient of such models contains two main terms: the so-called positive phase contribution tells the model to decrease the energy of training example x and the so-called negative phase contribution tells the model to increase the energy of all other points, in proportion to their probability according to the model. The negative phase term can be estimated by Monte-Carlo if one can sample from the model, but exact unbiased sampling is intractable, so different algorithms use different approximations.

The first and most common learning algorithm for RBMs is the Contrastive Divergence (CD) algorithm Hinton [1999, 2002]. It relies on a short alternating Gibbs Markov chain starting from the observed training example. This short chain yields a biased but low variance sample from the model, used to push up the energy of the most likely values (under the current model) near the current training example. It carves the energy landscape so as to have low values at the training points and higher values around them, but does not attempt to increase the energy (decrease the probability) of far-away probability modes, thereby possibly losing probability mass (and likelihood) there. After training an RBM with CD, in order to obtain good-looking samples, it is customary to start the Gibbs chain at a training example. As we find here, a single chain often does not mix well, visiting some probability modes often and others very rarely, so another common practice is to consider in parallel many chains all started from a different training example. However, this only sidesteps the problem of poor mixing.

The Persistent Contrastive Divergence (PCD) algorithm Tieleman [2008] was proposed to improve upon's CD limitation (pushing up only the energy of points near training examples). The idea is to keep a Markov chain (in practice several chains in parallel, for the reasons outlined above) to obtain the negative samples from an alternating Gibbs chain. Although the model is changing while we learn, we do not wait for these chains to converge after each update, with the reasoning that the parameter change is minor and the states which had high probability previously are still likely to have high probability after the parameter update.

Fast PCD Tieleman and Hinton [2009] was later proposed to improve upon PCD's ability to visit spurious modes. Two sets of weights are maintained. The "slow weights" w_m represent the standard generative model. They are used in the positive phase of learning and to draw samples of the model using a regular alternating Gibbs chain. The negative phase however, uses an additional set of "fast weights". Negative particles are samples from a persistent Markov chain with weights $w_m + v_m$, which creates a dynamic overlay on the energy surface defined by the model. Mixing is facilitated by using a large learning rate for parameters v_m, which is independent from that used for w_m (slow weights can therefore be fine-tuned using a decreasing learning rate with no impact on mixing). The fast weights v_m are pushed strongly towards zero with an L2 penalty ($0.05||v||^2$ in the pseudo-code provided in Tieleman and Hinton [2009]), ensuring that their effect is only temporary. Both w_m and v_m are updated according to the sampling approximation of the log-likelihood gradient. Tieleman and Hinton [2009] report substantial improvements in log-likelihood with FPCD in comparison to PCD and CD.

2 RBM Log-Likelihood Gradient and Contrastive Divergence

We formalize here the notation for some of the above discussion regarding RBMs and negative phase samples. Consider an energy-based model $p(s) \propto e^{-\text{Energy}(s)}$, with $s = (x, h)$, and marginal likelihood $p(x) = \sum_h e^{-\text{Energy}(x,h)} / \sum_{x,h} e^{-\text{Energy}(x,h)}$. The marginal log-likelihood gradient with respect to some model parameter w_m has two terms

$$\frac{\partial \log p(x)}{\partial w_m} = -\sum_h P(h|x) \frac{\partial \text{Energy}(s)}{\partial w_m} + \sum_s P(s) \frac{\partial \text{Energy}(s)}{\partial w_m} \tag{1}$$

which are respectively called the positive phase and negative phase contributions[1]. Consider a Markov random field defined by statistics g_m, i.e.

$$p(s) \propto e^{-\sum_m w_m g_m(s)}.$$

Then the gradients in Eq. 1 are easily computed by

$$\frac{\partial \text{Energy}(s)}{\partial w_m} = g_m(s).$$

In the following we consider RBMs with binary units x_j and h_i, and energy function $E(s) = -h'Wx - h'b - x'c$. Here, the statistics of interest are $h_i x_j$, h_i and x_j, and we associate them with the parameters W_{ij}, b_i, and c_j respectively.

The Contrastive Divergence (CD) algorithm Hinton [1999, 2002] consists in approximating the sums in Eq. 1 by stochastic samples, i.e. in updating parameter w_m by

$$w_m \leftarrow w_m - \epsilon \left(\frac{\partial \text{Energy}(x, \tilde{h}_1)}{\partial w_m} - \frac{\partial \text{Energy}(\tilde{x}_k, \tilde{h}_{k+1})}{\partial w_m} \right)$$

where $\tilde{h}_{t+1}$ is sampled from the model conditional distribution $P(h|\tilde{x}_t)$ (denoting by $\tilde{x}_0$ the training sample x used to initialize the Gibbs chain), $\tilde{x}_t$ is sampled from $P(x|\tilde{h}_{t-1})$, and ϵ is the learning rate of the update. Here, k is the number of alternating steps performed in the Gibbs chain: typically one uses $k = 1$ for efficiency reasons. This works well in practice, even though it may not be a good approximation of the log-likelihood gradient Carreira-Perpiñan and Hinton [2005], Bengio and Delalleau [2009].

[1] One should be careful that the signs of these terms in Eq. 1 do not match their name.

3 Tempered MCMC

Markov Chain Monte Carlo methods provide a way of sampling from otherwise unmanageable distributions by means of sampling from a sequence of simpler *local* distributions. Despite the correlations induced between neighboring samples in the sequence, provided some very general conditions of the resulting Markov chain (such as ergodicity) are satisfied, samples are assured to converge to the target distribution.

However, they suffer from one very important drawback. Because these methods are based on local steps over the sample space, they are subject to becoming "stuck" in local maximum of probability density, over-representing certain modes of the distribution while under-representing others.

Parallel Tempering MCMC is one of a collection of methods (collectively referred to as Extended Ensemble Monte Carlo methods Iba [2001]) designed to overcome this shortcoming of standard MCMC methods. The strategy is simple: promote mixing between multiple modes of the distribution by drawing samples from smoothed versions of the target distribution. Provided the topology of the distribution is sufficiently smoothed, the local steps of standard MCMC methods are then able to leave the local regions of high probability density to more fully explore the sampling space.

Consider the target distribution from which we wish to draw well mixing samples, given by:

$$p(x) = \frac{\exp(-E(x))}{Z}. \tag{2}$$

We create an extended system by augmenting the target distribution with an indexed temperature parameter:

$$p_{t_i}(x) = \frac{\exp(-E(x)/t_i)}{Z(t_i)} \tag{3}$$

At high temperatures ($t_i >> 1$), the effect of the temperature parameter is to smooth the distribution, as the effective energies become more uniform (uniformly zero) over the sampling space.

In the case of Parallel tempering, the strategy is to simulate from multiple MCMC chains, each at one of an ordered sequence of temperatures t_i from temperature $t_0 = 1$ that samples from the distribution of interest (the target distribution) to a high temperature $t_T = \tau$, ie.

$$t_0 = 1 < t_1 < \cdots < t_i < \cdots < t_{T-1} < t_T = \tau.$$

At the high temperatures the chain mixes well but is not the distribution in which we are interested, so the following question remains: how do we make use of the well mixing chains running at high temperatures to improve sampling efficiency from our target distribution at $t_0 = 1$? In parallel tempering this question is addressed via the introduction of cross temperature state swaps. At each time-step, two neighbouring chains running at temperature t_k and t_{k+1} may exchange their particles x_k and x_{k+1} with an exchange probability given by:

$$r = \frac{p_k(x_{k+1})p_{k+1}(x_k)}{p_k(x_k)p_{k+1}(x_{k+1})} \tag{4}$$

For the family of Gibbs distribution (in which we are particularly interested in), this boils down to:

$$r = \exp((\beta_k - \beta k + 1) \cdot (E(x_k) - E(x_{k+1}))) \tag{5}$$

It is straightforward to see how this algorithm can be applied to the training of RBMs. Instead of running a single persistent Markov Chain as in PCD, multiple chains are run in parallel, each at their own temperature t_i. For each gradient update, all chains perform one step of Gibbs sampling after which state swaps are proposed between all neighbouring chains in a sequential manner. In this paper, swaps were proposed from high to low temperatures (between temperatures t_i and t_{i-1}), as a way to encourage the discovery of new modes. Empirical evidence seems to suggest that this ordering is not crucial to performance however. The negative particle used in the gradient of Eq. 1 is then the particle at temperature t_0.

4 Related Work

Alternatively, one may also forego parallel chains altogether. In work which was done in parallel to ours, Salakhutdinov [2010] uses the method of tempered transitions to help mixing in Markov random fields. In this setting, transition operators are applied to a single Markov chain in sequence, such that the temperature of the chain is gradually increased to a maximum value before being decreased back to the nominal temperature. The resulting particle is then accepted based on the likelihood of the entire trajectory.

5 Experimental Observations

In Tieleman and Hinton [2009], Tieleman and Hinton [2009] highlight the importance of good sampling during the negative phase of RBM training. Without good mixing, negative particles can adversely affect the energy landscape by getting trapped in regions of high-probability and raising the energy level at that mode. In the extreme setting of a distribution containing few but well defined modes, combined with a high-learning rate, negative particles have the potential to pool together and cohesively undo the learning process. In this section, we will investigate the relation between mixing and learning for the most popular RBM training algorithms. The conclusions drawn from this analysis will serve as justification for the tempered MCMC method we propose.

5.1 CD-k: local learning ?

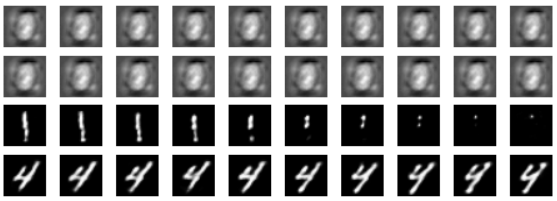
Because CD only runs the Markov Chain for a small number of steps, it is very susceptible to the mixing rate of the chain. Early in training, mixing is good since weights are initialized to small random values. As training progresses however, weights become larger and the energy landscape more peaked. For a fixed value of k, mixing thus degrades over time and leads to negative samples being increasingly correlated with training data. This can lead to a degeneracy where the energy of training examples is lowered but increased in the immediate proximity, in effect forming an energy barrier around the wells formed by the data. When sampling from the resulting model, a Markov Chain initialized with a random state will thus fail to find the high-probability modes as the energy barrier defines a boundary of low-probability. This can be observed in Fig. 1(a). In this figure, each row represent samples from separate chains, with samples shown every 50 steps of Gibbs sampling. The top two chains are initialized randomly while the bottom two chains were initialized with data from the test set. The top chains never converge to a digit while the bottom chains exhibit fairly poor mixing.

This phenomenon can also be confirmed by using the tempered MCMC procedure to sample from an RBM trained with CD. Fig.1(b) shows a mixing plot of tempered MCMC sampling using 50 chains and a maximum temperature of 10. The first line of the image shows the state of the chains at the start of training, with each color representing a single particle x_i native to temperature t_i. Low temperatures are shown on the left and high temperatures on the right. Each subsequent line tracks the movement of the original particle through time. High acceptance ratios would cause the colors to become entangled after a certain number of iterations. For an RBM trained with CD however, there is a clear bottleneck in the lower temperature ranges, through which high energy particles do not go through. This gives more credibility to our theory of how CD-1 learning modifies the energy landscape. Positive training data pushes down hard on the energy landscape but only locally, while sharp ridges are formed around the wells by the negative phase. With enough time and tempered MCMC chains, one should theoretically converge on these wells. Our experience suggests that this does not happen in practice, which speaks to the sharpness of the energy landscape formed by CD.

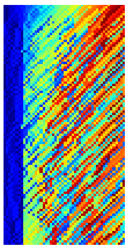
5.2 Limitations of PCD

Tieleman [2008] introduced persistent CD and recently PCD with fast-weights as ways to address the issue of bad mixing during the negative phase of learning. Maintaining a persistent chain has the benefit that particles explore the energy landscape more globally, pulling up the energy as they go along.

In Figure 2(a), we confirm that samples drawn during learning mix fairly well early on in training. The samples shown were collected midway through the learning procedure (epoch 5 of 10), from



(a)

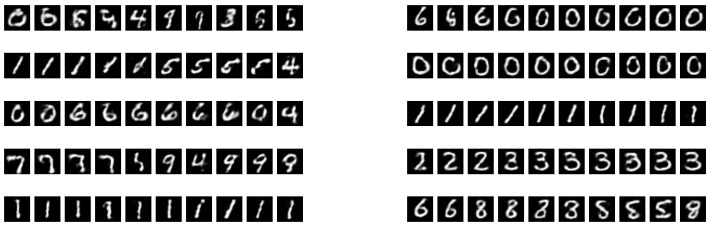


(b)

Figure 1: (a) Gibbs sampling from an RBM trained with CD, starting with a random initialization (2 top chains) vs. initializing with test images (b) Cross-temperature state swaps during MCMC sampling of an RBM trained with CD. Each color represents a particle originating at temperature t_k at time $t = 0$, as it jumps from one temperature to another. Temperatures in the range $[t_0, t_T]$ are shown from left to right. A bottleneck is clearly visible in the lower-temperature range.

an RBM with 500 units trained with persistent CD. Fig. 2(b) tells a different story however. These samples were obtained after epoch 10, at which point learning was effectively stopped (learning rate of 0). We can see that mixing has degraded significantly.

We believe two factors are responsible for this. As mentioned previously, early on in training the energy landscape is smooth and mixing is good. With each weight update however, the positive phase creates wells which become progressively deeper. Eventually, negative samples become trapped in low-probability states and are unable to explore the energy landscape. This results in parameter updates which deviate from the true-likelihood gradient. This is very problematic since there is no early-stopping heuristics which can be used to stop learning in time to avoid this bad mixing.



(a) (b)

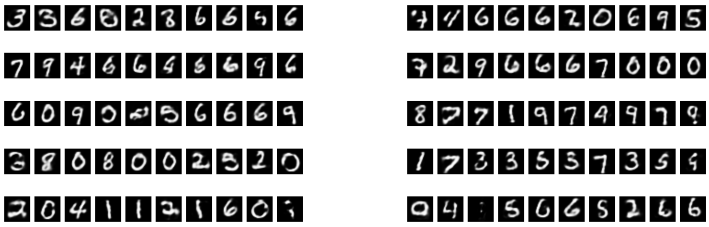
Figure 2: Negative samples drawn from an RBM trained with PCD, at (a) epoch 5 of 10 (b) after learning is stopped. Notice that once the learning procedure is stopped, the mixing rate of the chains drops dramatically and samples become trapped in a minima of the energy landscape. Each row shows samples drawn from a single Gibbs chain, with 50 steps of Gibbs sampling between consecutive images.

The other factor to take into account is the effect of "fast weights" explored in Tieleman and Hinton [2009]. The sampling procedure used during learning encourages the particles to mix, since each update renders the state of the negative particles less probable. Negative particles are therefore "encouraged" to move around in input space. The fast-weights algorithm (FPCD) exploits this idea by perturbing the model, but in a way which only affects the negative phase. Unfortunately, FPCD generates spurious samples as shown in Figure 3. These in effect, represent the paths which a negative particle must take to jump from one mode to the next. Not being true samples of the model however, they undoubtedly hurt the learning process since the parameter updates will not follow the true gradient.

Figure 3: Samples obtained using the fast-weight sampling procedure.

5.3 Tempered MCMC for training and sampling RBMs

We now use the same experimental protocol to show that our RBM trained with tempered MCMC addresses the above problems in a straight-forward and principled manner. Figure 4 clearly shows that samples obtained from the fully trained model (once learning is stopped) mix extremely well.



(a) (b)

Figure 4: Negative samples drawn from an RBM with 500 hidden units, trained with tempered MCMC, (a) at epoch 5 of 10 (b) after learning is stopped. It is clear that the resulting model exhibits much better mixing than in Fig. 2(b). Again, each row shows samples drawn from a single Gibbs chain, with 50 steps of Gibbs sampling between consecutive images.

The maximum temperature t_T and number of parallel chains to use were chosen somewhat arbitrarily. The maximum temperature was chosen by visualizing the samples from the top-most chain and making sure that the chain exhibited good mixing. The number of chains was chosen to be large enough such that the mixing plot of Figure 1(b) showed (i) a large number of cross-temperature state swaps and (ii) that a single particle x_i, on average, visited temperatures in the range $[t_0, t_T]$ with equal proportions.

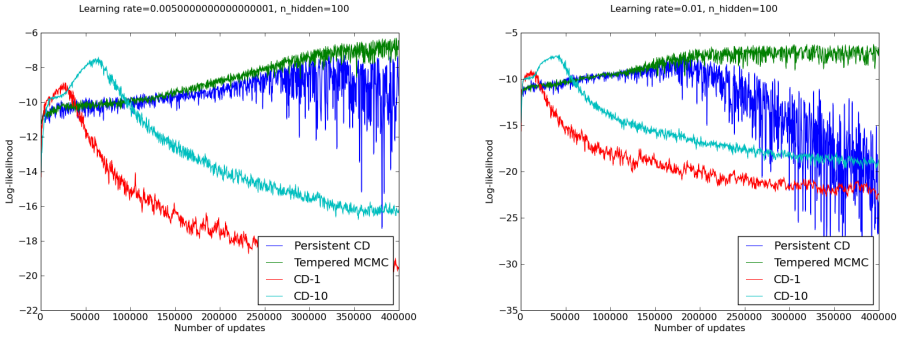
With regards to the learning rate, we found that a decreasing learning rate schedule was necessary for the model to learn a good generative model. This should not be surprising, as it seems to echo the theoretical findings of Younes [1998], which outlines a proof of convergence for profiles of the type a/t with small enough a. Empirically, we found that decreasing the learning rate linearly towards zero also worked well.

5.4 Tempered MCMC vs. CD and PCD

Here we consider a more quantitative assessment of the improvements offered by using tempered MCMC, compared to regular CD and PCD. In order to compute the exact log-likelihood of our models, we use a toy dataset of 4x4 binary pixel images. Half of the training examples are images which represent two black lines over a white background, such that pixels of each line are not adjacent to the other line's pixels. The lines are made of two pixels and the image is assumed to have a torus structure. The second half of the training examples are the same samples where black and white have been swapped. In total, this yields 320 valid images (out of the 2^{16} possible images) which are all used as training data.

For all three algorithms, we performed 400,000 weight updates using mini-batches of size 8. This means that each training example is presented 10,000 times to each model. Learning rates were either held constant throughout learning or decreased linearly towards zero. No weight decay was used in any of the models. The other hyperparameters were varied as follows:

- number of hidden units in $\{10, 20, 30, 40, 50, 100, 200, 300\}$
- initial learning rates in $\{0.1, 0.05, 0.01, 0.005, 0.001\}$
- for CD-k, a number k of steps in $\{1, 3, 5, 10, 25\}$
- for tempered MCMC, 50 chains spaced uniformly between $t_0 = 1$ and $t_T = 2$.



(a) (b)

Figure 5: Evolution of the exact log-likelihood of the training data as a function of epochs. (a)100 hidden units and learning rate 0.005 (b)100 hidden units and learning rate 0.01. The CD-k and PCD algorithms become unstable after a while, whereas the tempered MCMC method exhibits a much more reliable behaviour.

Figure 5 shows typical examples of the behaviour of the three algorithms. What usually happens is that after some time, the poor mixing of the chains used in CD-k and PCD leads to updates which actually *hurt* the modeling ability of the RBM. We observed that this phenomenon became worse as the learning rate increased. In contrast, the tempered MCMC version is much more stable and typically increases the log-likelihood in a pseudo-monotonic manner.

5.5 Estimating Likelihood on MNIST

Next we wanted to obtain a similar quantitative measure, that would provide for a more objective comparison of the algorithms considered, on a real-world problem.

An important characteristic of RBMs is that, while computing the exact likelihood under the learnt model is intractable, it is however *relatively* easy to generate samples from it. We thus set to compute a *quantitative* measure that could reflect the "quality" of the sample generation that the various trained models were capable of. More specifically, we want a numerical measure of how close the sample generation by a particular trained RBM is to the "true" but unknown distribution which produced the training samples $\mathcal{D}$. This will allow us to evaluate more objectively to what degree the considered training procedures are able to capture the distribution they were presented.

Notice that by formulating the question in this manner we link a trained model and a sampling procedure. What we evaluate is a (model + sampling-procedure) combination, i.e. the resulting measure depends not only on the model's parameters but also on the particular sampling procedure used after training to generate new samples. While it is natural here to use the same sampling procedure during post-training generation as was used during training, it is also possible to use after training a different procedure than what was used during training, e.g. train using simple CD but then generate using tempered MCMC.

The measure we consider is the average log probability of the samples in a held out test set $\mathcal{D}_{\text{test}}$ under a non-parametric Parzen Windows density estimation $\hat{p}_{\sigma,\mathcal{D}_{\text{s}}}$ based on the samples $\mathcal{D}_{\text{s}}$ generated by a given model that was trained on a train set $\mathcal{D}$. The test set $\mathcal{D}_{\text{test}}$ has n samples $x^{(1)}, \ldots, x^{(n)}$ that originate from the same unknown distribution as the data $\mathcal{D}$ used for training. Similarly $\mathcal{D}_{\text{s}}$ has n' samples $s^{(0)}, \ldots, s^{(n')}$ generated using a given (model + sampling-procedure).

Formally our sample generation quality measure is:

$$\ell(\mathcal{D}_{\text{test}}, \mathcal{D}_{\text{s}}) = \frac{1}{n} \sum_{k=1}^{n} \log \hat{p}_{\sigma,\mathcal{D}_{\text{s}}}(x^{(k)}) \tag{6}$$

where $\hat{p}_{\sigma,\mathcal{D}_{\text{s}}}(x^{(k)})$ is the density evaluated at point $x^{(k)}$ obtained with a non-parametric kernel density estimator based on $\mathcal{D}_{\text{s}}$ with hyperparameter σ. i.e. $\hat{p}_{\sigma,\mathcal{D}_{\text{s}}}(\mathbf{x}) = \frac{1}{n'} \sum_{k=1}^{n'} K(\mathbf{x}; s^{(j)})$. We will use, as customary, a simple isotropic Gaussian kernel with standard deviation σ, i.e. $K = \mathcal{N}_{s^{(j)},\sigma}$.

In practice we proceeded as follows:

- Search for a good value of the kernel bandwidth σ. We perform a grid search trying to maximize the log probability of test samples $\mathcal{D}_{\text{test}}$ under $\hat{p}_{\sigma,\mathcal{D}_{\text{s}}}$ i.e. $\sigma \simeq \arg\max_{\sigma'} \ell(\mathcal{D}_{\text{test}}, \mathcal{D})$. We then keep this bandwidth hyperparameter fixed.
- Generate $\mathcal{D}_{\text{s}}$ made of n' samples from the considered RBM with the desired sampling procedure. For convenience[2] we choose $n' = 2n$ which is 20,000 for the standard MNIST test set.
- Compute generation quality $\ell(\mathcal{D}_{\text{test}}, \mathcal{D}_{\text{s}})$ as defined in Eq. 6.
- Repeat these last two steps for all models and sampling procedures under consideration.

Table 1 reports the generation quality measure obtained for different combinations of training procedure (yielding a trained model) and post-training sample generation procedure. Model selection was performed by selecting, for each combination of (training procedure, sampling algorithm), the hyperparameters leading to the best likelihood. Including the sampling procedure in the model selection process allows for a fair comparison of training algorithms. Hyperparameters are thus selected such that the trained model is compatible with the sampling procedure.

As we can see, the tempered models have a significantly higher likelihood than all the other training algorithms, regardless of sampling procedure. It is interesting to note that in this case, sampling with tempered MCMC did not result in a higher likelihood. This may indicate that the parameter σ should be optimized independently for each model $\hat{p}_{\sigma,\mathcal{D}_{\text{s}}}(\mathbf{x})$. We leave this as future work. Also interesting to note: sampling CD or PCD-trained models with tempered MCMC results in a worse performance than with standard Gibbs sampling. This is definitely more pronounced in the case of CD and highlights the issues raised in section 5.1. As for PCD, this again confirms that mixing eventually breaks down during learning, after which negative particles fail to explore the energy landscape properly. Using tempered MCMC during training seems to avoid all these pitfalls.

6 Conclusion

We presented a new learning algorithm to train Restricted Boltzmann Machines, relying on the strengths of a more advanced sampling scheme than the ones typically used until now. The tempered MCMC sampling technique allows for a better mixing of the underlying chain used to generate samples from the model. We have shown that this results in better generative models, from

[2] using the same sizes $n' = n$ allows us to compute the reverse $\ell(\mathcal{D}_{\text{s}}, \mathcal{D}_{\text{test}})$ without needing to search for a different hyperparameter σ.

Table 1: Log probability of test set samples under a non-parametric Parzen window estimator based on generated samples from models obtained with different training procedures. For reference the measure obtained on the training set is 239.88 $\pm$ 2.30 ($\pm$ indicated standard error). As can be seen the TMCMC trained model largely dominates.

	Sample generation procedure		
Training procedure	TMCMC	Gibbs (random start)	Gibbs (test start)
TMCMC	208.26	**210.72**	209.83
FPCD	180.41	174.87	175.92
PCD	80.06	127.95	139.36
CD	-1978.67	-854.08	37.18

qualitative and quantitative observations on real and simulated datasets. We have also shown that the use of tempering affords a higher reliability and increased robustness to learning rates and number of unsupervised training epochs. More experiments are still required however to assess the impact of this method on classification, especially in the context of deep architectures, where one typically stacks trained RBMs before using a global supervised criterion. Amongst other questions, we would like to determine if and under which conditions a better generative model translates to an increase in classification performance.

References

Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1), 2009.

Yoshua Bengio and Olivier Delalleau. Justifying and generalizing contrastive divergence. *Neural Computation*, 21(6):1601–1621, 2009.

Miguel A. Carreira-Perpiñan and Geoffrey E. Hinton. On contrastive divergence learning. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS'05)*, pages 33–40. Society for Artificial Intelligence and Statistics, 2005.

Yoav Freund and David Haussler. Unsupervised learning of distributions on binary vectors using two layer networks. Technical Report UCSC-CRL-94-25, University of California, Santa Cruz, 1994.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.

Geoffrey E. Hinton. Products of experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN)*, volume 1, pages 1–6, Edinburgh, Scotland, 1999. IEE.

Goeffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

Yukito Iba. Extended ensemble monte carlo. *International Journal of Modern Physics*, C12:623–656, 2001.

Nicolas Le Roux and Yoshua Bengio. Representational power of restricted boltzmann machines and deep belief networks. *Neural Computation*, 20(6):1631–1649, 2008.

Ruslan Salakhutdinov. Learning in markov random fields using tempered transitions. In Dale Schuurmans, Yoshua Bengio, Christopher Williams, John Lafferty, and Aron Culotta, editors, *Advances in Neural Information Processing Systems 22 (NIPS'09)*, 2010. To appear.

Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 6, pages 194–281. MIT Press, Cambridge, 1986.

Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08)*, pages 1064–1071. ACM, 2008.

Tijmen Tieleman and Geoffrey Hinton. Using fast weights to improve persistent contrastive divergence. In Léon Bottou and Michael Littman, editors, *Proceedings of the Twenty-sixth International Conference on Machine Learning (ICML'09)*, pages 1033–1040, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: http://doi.acm.org/10.1145/1553374.1553506.

Max Welling, Michal Rosen-Zvi, and Geoffrey E. Hinton. Exponential family harmoniums with an application to information retrieval. In L.K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17 (NIPS'04)*, pages 1481–1488, Cambridge, MA, 2005. MIT Press.

Laurent Younes. On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. In *Stochastics and Stochastics Models*, pages 177–228, 1998.