

---

# Parallel Tempering for Training of Restricted Boltzmann Machines

---

Guillaume Desjardins, Aaron Courville, Yoshua Bengio, Pascal Vincent, Olivier Delalleau

Dept. IRO, Université de Montréal

P.O. Box 6128, Succ. Centre-Ville, Montreal, H3C 3J7, Qc, Canada

## Abstract

Alternating Gibbs sampling between visible and latent units is the most common scheme used for sampling from Restricted Boltzmann Machines (RBM), a crucial component in deep architectures such as Deep Belief Networks (DBN). 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 property hinders RBM training methods such as the Contrastive Divergence and Persistent Contrastive Divergence algorithm that rely on Gibbs sampling to approximate the likelihood gradient. 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 on a toy dataset that it helps both sampling and learning.

## 1 INTRODUCTION

Restricted Boltzmann Machines (Smolensky, 1986; Freund & Haussler, 1994; Hinton, 2002; Welling et al., 2005) have attracted much attention in recent years because of their power of expression (Le Roux & Bengio, 2008), because inference (of hidden variables  $\mathbf{h}$  given visible variables  $\mathbf{v}$ ) 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). Most of the literature on Restricted Boltzmann Machines (RBMs) relies on variations of alternating Gibbs sampling, which exploits the bipartite structure of the graphical model (there are links only between visible

and hidden variables), in order to learn and sample from these models. RBMs and other Markov Random Fields are energy-based models in which we can write  $p(\mathbf{v}, \mathbf{h}) \propto e^{-E(\mathbf{v}, \mathbf{h})}$ . The log-likelihood gradient of such models contains two main terms: the so-called positive phase contribution tells the model to decrease the energies associated with training example  $\mathbf{v}$  and the so-called negative phase contribution tells the model to increase the energy of all other configurations of  $(\mathbf{v}, \mathbf{h})$ , in proportion to their probability according to the model. The negative phase term can be estimated via a Monte-Carlo scheme if one can sample from the model, but exact unbiased sampling is intractable, so different algorithms use different approximations.

The most popular learning algorithm for RBMs is the Contrastive Divergence (CD) algorithm (Hinton, 1999; Hinton, 2002). It relies on approximating the negative phase contribution to the gradient with samples drawn from a short alternating Gibbs Markov chain starting from the observed training example. Using these short chains yields a low variance, but biased estimates of the gradient. One can envision the action of the CD negative phase as pushing up the energy of the most likely values (under the model) *near training examples*. This strategy of focusing training effort near the training data appears to be effective in learning representations or features of data. However, as our experiments suggest, the data-centric focus of CD training can result in spurious probability modes far from the training data – making CD less ideal as a method of training full-fledged generative models of data (which is what one needs for the top level of a DBN).

The Persistent Contrastive Divergence (PCD) algorithm (Tieleman, 2008) was proposed to improve upon CD’s limitations (of pushing up only the energy of points near training examples). Similar to CD, PCD approximates the negative phase of the gradient with samples drawn from a short alternating Gibbs chain. However in PCD, rather than starting this chain from training data, the state of the Markov chain *persists* from the previous iteration of the gradient calcula-

---

Appearing in Proceedings of the 13<sup>th</sup> International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9. Copyright 2010 by the authors.

tion. Although the model is changing while we learn, we do not wait for the chain to converge after each update, with the reasoning that since (with a sufficiently small learning rate) the change in both the parameters and the invariant distribution of the model are small, just a few Gibbs iterations are sufficient to track these changes and maintain samples from the evolving invariant distribution of the model. However, as noted in (Tieleman & Hinton, 2009), reliance on a single persisting Markov chain often leads to degenerative training. When faced with the kind of multimodal target distributions that are common in machine learning applications, MCMC methods, such as the Gibbs sampling employed in the PCD negative phase estimation, are subject to becoming “stuck” in local maxima of probability density. This is because these methods are based on local steps over the sample space. The result is a chain that mixes slowly, over-representing certain modes of the distribution while under-representing others. It produces a distorted estimate of the negative phase contribution of the gradient that undermines training. To help mitigate these problems, common practice is to consider many chains in parallel as a kind of mini-batch. However, this practice only sidesteps the problem of poor mixing because it would require us to maintain potential many more parallel chains than modes of the distribution. We suggest that this is not a strategy that will scale to large-scale problems with truly complicated distributions.

Fast PCD (Tieleman & Hinton, 2009) (FPCD) was later proposed to improve upon PCD’s mixing properties. Through an exaggerated (high learning rate) version of the dynamics of PCD training, FPCD works by actively discouraging the negative phase Markov chain from dwelling in any single mode of the distribution by quickly “unlearning” that mode. A potential disadvantage of FPCD is that the negative phase samples are drawn from a distribution that can diverge from the invariant distribution of the model. Relative to PCD, FPCD trades sampling fidelity for superior mixing. In some cases this trade-off appears to be a good one, (Tieleman & Hinton, 2009) report substantial improvements in log-likelihood with FPCD in comparison to PCD and CD.

In this paper, we propose an alternate strategy for ameliorating the mixing of the PCD negative phase Markov chain while still drawing samples from a process with the correct invariant distribution. Our method replaces the single Gibbs chain used in PCD with a series of chains implementing a parallel tempering scheme. Parallel Tempering MCMC is one of a collection of methods (collectively referred to as Extended Ensemble Monte Carlo methods (Iba, 2001)) designed

to overcome the inability of standard MCMC methods to handle multimodal distributions. The strategy is simple: promote mixing between multiple modes of the distribution by drawing samples from smoothed (higher temperature) 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. Using a toy training set and the MNIST digits dataset, we explore the properties of the proposed parallel tempering training strategy and compare it to CD, PCD and FPCD. We find both through visualization of samples and measures of likelihood that the use of parallel tempering in the negative phase of PCD improves both sampling and learning.

## 2 APPROXIMATING LOG-LIKELIHOOD GRADIENTS

We consider log linear models parameterized by a vector of parameters  $\theta$ , for which the probability for a configuration  $\mathbf{x}$  will be given by:

$$p(\mathbf{x}) = \frac{1}{Z(\theta)} \exp(-E(\mathbf{x}))$$

where energy function  $E$  has the form  $E(\mathbf{x}) = -\theta^T \phi(\mathbf{x}) = -\sum_k \theta_k \phi_k(\mathbf{x})$ .  $Z(\theta) = \sum_{\mathbf{x}} \exp(-E(\mathbf{x}))$  is the partition function whose computation is usually intractable.

Vector  $\mathbf{x} = (\mathbf{v}, \mathbf{h})$  is decomposed into observed (visible) variables  $\mathbf{v}$  and latent (hidden) variables  $\mathbf{h}$ . The marginal likelihood of a visible configuration  $\mathbf{v}$  can thus be written as

$$p(\mathbf{v}) = \frac{\sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))}{\sum_{\mathbf{x}} \exp(-E(\mathbf{x}))}$$

This yields a marginal log-likelihood gradient with respect to a model parameters  $\theta$  that can be decomposed into two terms:

$$\begin{aligned} \frac{\partial \log p(\mathbf{v})}{\partial \theta} &= -\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}) \frac{\partial E(\mathbf{v}, \mathbf{h})}{\partial \theta} + \sum_{\mathbf{x}} p(\mathbf{x}) \frac{\partial E(\mathbf{x})}{\partial \theta} \\ &= \mathbb{E}_{p(\mathbf{h}|\mathbf{v})}[\phi(\mathbf{v}, \mathbf{h})] - \mathbb{E}_{p(\mathbf{x})}[\phi(\mathbf{x})] \end{aligned}$$

where  $\mathbb{E}_p[\dots]$  denotes expectation with respect to distribution  $p$ .

This decomposition suggests computing the gradient by estimating these two expectations through sampling. The first term corresponds to sampling hidden configurations when the visibles are clamped to the training input; it will be called the positive phase contribution. The second term corresponds to obtaining joint hidden and visible samples from the current

model; it will be called the negative phase. We will be particularly interested in RBMs which correspond to parameterization  $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{c}\}$ , and  $E(\mathbf{v}, \mathbf{h}) = -(\mathbf{h}^T \mathbf{W} \mathbf{v} + \mathbf{h}^T \mathbf{b} + \mathbf{v}^T \mathbf{c})$ , i.e.  $\phi(\mathbf{v}, \mathbf{h}) = (\mathbf{h} \mathbf{v}^T, \mathbf{h}, \mathbf{v})$ . Parameter  $\mathbf{b}_i$ ,  $\mathbf{c}_j$  and  $\mathbf{W}_{ij}$  are associated respectively with  $\mathbf{h}_i$ ,  $\mathbf{v}_j$  and  $\mathbf{h}_i \mathbf{v}_j$ . In RBMs, it is straightforward to sample  $\mathbf{h}|\mathbf{v}$  and  $\mathbf{v}|\mathbf{h}$ , so obtaining samples for the positive phase is easy. However obtaining samples from the model for the unclamped negative phase is not, as it would require running the alternating Gibbs chain (alternating between sampling  $\mathbf{h}|\mathbf{v}$  and  $\mathbf{v}|\mathbf{h}$ ) to equilibrium.

**Contrastive Divergence:** The Contrastive Divergence (CD) algorithm (Hinton, 1999; Hinton, 2002) approximates the gradient by running the chain for only a few steps to get a sample for the negative phase. Parameters are then learnt by stochastic gradient descent using a learning rate  $\eta$  and the approximated log likelihood gradient computation, i.e.  $\theta \leftarrow \theta + \eta \left( \phi(\mathbf{v}, \tilde{\mathbf{h}}_0) - \phi(\tilde{\mathbf{v}}_k, \tilde{\mathbf{h}}_k) \right)$ , where we initialize the Gibbs chain at  $\tilde{\mathbf{v}}_0 = \mathbf{v}$ , sampling  $\tilde{\mathbf{h}}_0 \sim p(\mathbf{h}|\mathbf{v} = \tilde{\mathbf{v}}_0)$  and where  $(\tilde{\mathbf{v}}_k, \tilde{\mathbf{h}}_k)$  is obtained after  $k$  alternating steps performed in the Gibbs chain. Typically one uses  $k = 1$  for efficiency reasons.

Despite CD’s popularity, it does not yield the best approximation of the log-likelihood gradient (Carreira-Perpiñan & Hinton, 2005; Bengio & Delalleau, 2009). Empirically, we find that at the beginning of training it behaves well. But as the training progresses and the magnitude of the parameter increases, the ergodicity of the Markov chain decreases and, consequently, the quality of approximation to the gradient degrades.

**Persistent CD:** Persistent CD (PCD) on the other hand, approximates the gradient by drawing negative phase samples from a persistent Markov chain, which attempts to track changes in the model. If we denote the state of this chain at timestep  $t$  as  $v_t^{(-)}$ , then the gradient update follows  $\left( \phi(\mathbf{v}, \tilde{\mathbf{h}}_0) - \phi(\tilde{\mathbf{v}}_{t+k}^{(-)}, \tilde{\mathbf{h}}_{t+k}^{(-)}) \right)$ , where  $(\tilde{\mathbf{v}}_{t+k}^{(-)}, \tilde{\mathbf{h}}_{t+k}^{(-)})$  are again obtained after  $k$  alternating Gibbs steps starting from state  $v_t^{(-)}$ . This estimator of the likelihood gradient can be seen as the exactly gradient of the following cost function (Tieleman & Hinton, 2009):  $KL(p||p_\theta) - KL(p_{\theta,t}||p_\theta)$ , where  $p$  is the training distribution,  $p_\theta$  the model distribution and  $p_{\theta,t}$  is the distribution formed by the Markov chain at time  $t$ . The second term is, in essence, an error term, incurred by performing only  $k$  steps of Gibbs sampling instead of running the chain to equilibrium. When the mixing rate of the chain is good, this error term can be all but ignored. As the ergodicity of the Markov chain decreases however (due to large learning

rates or long training time), it becomes the dominant factor leading to unstable behavior (Tieleman, 2008).

**Fast PCD:** Fast PCD (Tieleman & Hinton, 2009) (FPCD) was later proposed to improve upon this limitation by introducing a separate mixing mechanism, which does not deteriorate as training progresses. To do this, FPCD maintains two sets of weights. The “slow weights”  $w_m$  represent the standard parameters of the model and are used in both the positive and negative phases. An additional set of “fast weights” is used in the negative phase however, where samples are drawn from a persistent Markov chain with weights  $w_m + v_m$ . The fast-weights create a dynamic overlay on top of the energy surface defined by the model. It encourages mixing by temporarily reducing the probability of recently visited modes. Good mixing can thus be achieved by using a large learning rate for the fast parameters  $v_m$ . Since this learning rate is independent from that used for  $w_m$ , the model can still be fine-tuned using a decreasing learning rate with no impact on mixing. Both  $w_m$  and  $v_m$  are updated according to the standard PCD gradient, although changes in the weights  $v_m$  are decayed exponentially fast to 0 to ensure that their effect is only temporary.

### 3 TEMPERED MCMC

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

$$p(\mathbf{x}) = \frac{\exp(-E(\mathbf{x}))}{Z}. \quad (1)$$

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

$$p_{t_i}(\mathbf{x}) = \frac{\exp(-E(\mathbf{x})/t_i)}{Z(t_i)} \quad (2)$$

At high temperatures ( $t_i \gg 1$ ), 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  $t_T = \tau$ , i.e.

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

At high temperatures, the chain mixes well but is not sampling from 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  $\mathbf{x}_k$  and  $\mathbf{x}_{k+1}$  with an exchange probability given by:

$$r = \frac{p_{t_k}(\mathbf{x}_{k+1})p_{t_{k+1}}(\mathbf{x}_k)}{p_{t_k}(\mathbf{x}_k)p_{t_{k+1}}(\mathbf{x}_{k+1})} \quad (3)$$

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

$$r = \exp((\beta_k - \beta_{k+1}) \cdot (E(\mathbf{x}_k) - E(\mathbf{x}_{k+1}))), \quad (4)$$

where  $\beta_k$  is the inverse temperature parameter.

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 randomly selected neighbouring chains. The negative particle used in the gradient of Eq. 1 is then the particle at temperature  $t_0$ .

While there are numerous ways to promote mixing in MCMC simulations, Parallel Tempering MCMC is particularly well suited to being paired with a PCD-style learning algorithm. The multiple parallel chains constitute a pool of samples (replicas) whose relative probability with respect to the target distribution is reflected in their relative dwell time in the nominal temperature chain. This offers a significant advantage over the standard practice with PCD of maintaining a batch of independent chains, since there is no way of weighing the relative likelihood of the samples within these batches. This relative weighting mechanism permits rapid adaptation of the sampling scheme to small changes in the relative probability of existing modes of the distribution. For example, consider the MNIST dataset, if by chance there were a disproportionate number of twos drawn in the negative phase at learning iteration  $k$ , then through the action of the gradient, the chain swapping mechanism would respond quickly to reduce the number of twos in the negative phase samples drawn at iteration  $k + 1$ . As a result, learning with parallel tempering is naturally more robust to instabilities that often plague PCD (Younes, 1998).

**Related Work** Alternatively, one may also forego parallel chains altogether. In work which was done concurrently but independently of ours, (Salakhutdinov, 2010) uses the method of tempered transitions

(Neal, 1994) to promote mixing and fine-tuning the parameters of 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. While this is similar in spirit to the method we propose, maintaining the extended set of particles has the advantage that it does not suffer from potentially high accept/reject ratios. This makes the parallel implementation better suited for use throughout training, as opposed to being limited to a fine-tuning stage.

## 4 EXPERIMENTAL OBSERVATIONS

In Section 2, we saw how CD and other "persistent" learning algorithms differ in their approximations to the true likelihood gradient. While PCD is inherently brittle to the problem of reduced ergodicity, CD and FPCD both offer alternative ways of dealing with this issue. We thus begin our experimental section with qualitative observations of negative phase samples obtained with each method. We do this in order to illustrate the failure modes of each algorithm and show that the tempered approach is not subject to the same issues. We perform these observations by training 500 hidden units RBMs on the very familiar MNIST dataset.

### 4.1 QUALITATIVE OBSERVATIONS

**CD-k** Since CD initializes its Markov chain with a positive training example and mixing rate is known to decrease with learning, negative phase samples obtained with CD are bound to become increasingly autocorrelated and correlated with training data.

Without some form of early-stopping, this can lead to a degeneracy where the energy of training examples is lowered but increased in the immediate proximity, in effect creating 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. In this figure, each row represents samples from independent chains, with samples shown every 50 steps of Gibbs sampling. The top two chains were initialized randomly while the bottom two chains were initialized from the test set. The chains at the top never converge to a digit, while the bottom chains exhibit very poor mixing.

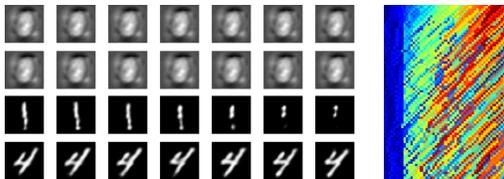


Figure 1: (left) Gibbs sampling from an RBM trained with CD, initializing the chain with random samples (top 2 chains) vs. test images (bottom 2). (right) Cross-temperature state swaps obtained by sampling from the CD-trained model with parallel tempering.

This phenomenon can be confirmed by using parallel tempering to sample from an RBM trained with CD. Fig.1 shows a mixing plot obtained using 50 chains and a maximum temperature of 10. The first line of the image shows the initial state of the system, with each color representing a single particle  $\mathbf{x}_k$  native to temperature  $t_k$ , with temperatures ordered from left to right. Each subsequent line tracks the movement of the original particle through time. High acceptance ratios should cause the colors to become entangled after a certain number of iterations. However, the energy barrier formed by CD around positive training examples can create a bottleneck, through which high energy particles do not go through. This effect will only be worse for a Markov chain operating at a temperature of 1.

**PCD and FPCD** In Figure 2(a), we confirm that samples drawn during PCD learning mix fairly well *early on in training*. The samples shown were collected midway through the learning procedure (epoch 5 of 10), from an RBM with 500 hidden units trained with persistent CD. Fig. 2(c) shows samples obtained after epoch 10. At this point learning was effectively stopped (learning rate of 0). Consistent with our understanding of PCD, we can see that with more training, mixing degraded significantly. As shown in Figure 3, while FPCD samples mix much more readily than those of PCD, they also generates “spurious” samples that, in effect, represent the paths which negative phase samples must take in jumping from one mode to another. These inter-mode samples are clearly overrepresented in the FPCD negative phase gradient contribution relative to the true model distribution.

**Parallel Tempering** We now show that RBMs trained with tempered MCMC address the above problems in a straightforward and principled manner. Figure 2 shows that samples obtained from the fully trained model (once learning is stopped) mix well. The number of chains was chosen to be large enough such that the mixing plot of Figure 1 showed (i) a large



Figure 3: Samples obtained using the fast-weight sampling procedure: these represent the path taken by the negative particles to move from one mode to another.

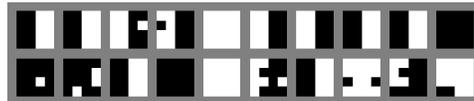


Figure 4: Examples from our toy dataset with  $p = 0.01$  (top) and  $p = 0.1$  bottom.  $p$  controls the relative distance between the four basic modes.

number of cross-temperature state swaps and (ii) that a single particle  $\mathbf{x}_i$ , on average, visited temperatures in the range  $[t_0, t_T]$  with equal proportions.

## 4.2 TOY DATASET

Here we consider a more quantitative assessment of the improvements offered by using tempered MCMC, compared to regular CD-1, PCD-1 and FPCD-1. In order to compute the exact log-likelihood of our models, we use a toy dataset of 4x4 binary pixel images. The dataset is composed of 10,000 images, which are variations of four basic modes, chosen to be maximally distant from one another. For each basic mode, we generate 2,500 near replicas, by considering a probability  $p$  of permuting each pixel independently. Note that from the point of view of a Gibbs sampler, the probability  $p$  directly controls the effective distance between each mode. As such, this dataset is well suited to study the issue of mixing in learning algorithms.

Figure 4 shows 10 training samples with  $p = 0.01$  (top) and 10 examples with  $p = 0.1$  (bottom).

In all experiments, we performed 500,000 weight updates for both positive and negative phases. Each training example is thus presented 50 times to each model. Learning rates were either held constant throughout learning or decreased to zero, either linearly (Tieleman, 2008) or with the standard  $1/t$  schedule. No weight decay was used in any of the models. As for the other hyperparameters, we tested number of hidden units in  $\{2,5,10\}$  and initial learning rates in  $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ . The tempered models were trained with  $\{2,5,10\}$  parallel chains with minimal inverse temperature of  $\beta = 0$ . To offset the computational cost of having parallel chains, CD, PCD and FPCD performed  $k = \{2, 5, 10\}$  steps of Gibbs sampling between consecutive parameter updates, accumulating negative phase samples so as to form a nega-

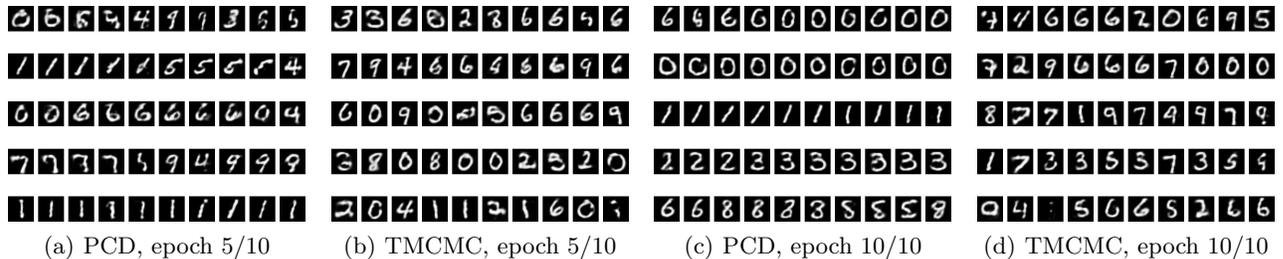


Figure 2: Negative samples drawn from a 500 hidden unit RBM trained with PCD and tempered MCMC (TMCMC) during and after training. Each row shows samples drawn from a single Gibbs chain, with 50 steps of Gibbs sampling between consecutive images. PCD and tempered MCMC both yield good samples during training, fueled in part by the “fast weights” effect. Once the learning rate becomes null, the mixing rate of PCD degrades significantly. In contrast, the tempered MCMC approach maintains exceptionally good mixing.

tive phase mini-batch. Tempered models were limited to a single Gibbs steps.

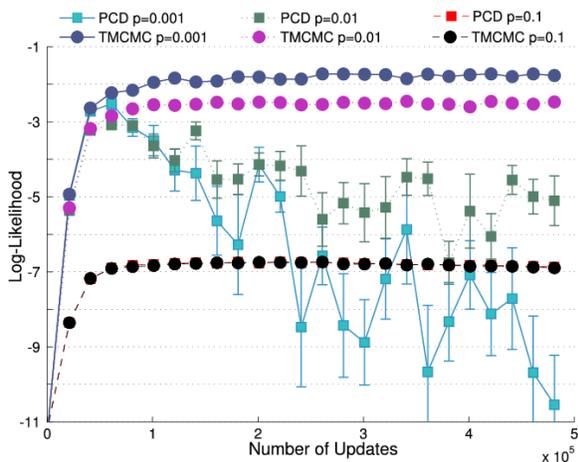


Figure 5: Log-likelihood of the training data under a model trained with PCD and TMCMC, as a function of the parameter  $p$ . Learning rate is fixed to  $10^{-3}$  and number of hidden units to 10. TMCMC significantly outperforms PCD, as the modes become further apart.

In Figure 5, we show typical results obtained using PCD with  $k = 10$  and TMCMC with 10 parallel chains. When the modes are close to one another ( $p = 0.1$ ), mixing is trivial and both algorithms perform equally well (learning rate of 0.001). As the modes become further apart however, good mixing becomes crucial. Negative particles must move from one mode to another to adequately model the distribution. This is hampered by learning however, which decreases the mixing rate of the Markov chain (Younes, 1998). Around 60,000 updates, mixing degrades to the point where the PCD algorithm starts to diverge: the Markov chain can no longer keep up with the parameter updates, resulting in a sudden drop in likelihood. Similar results are obtained for CD and FPCD. With 10 chains and a learning rate of 0.001, TMCMC seems immune to this issue.

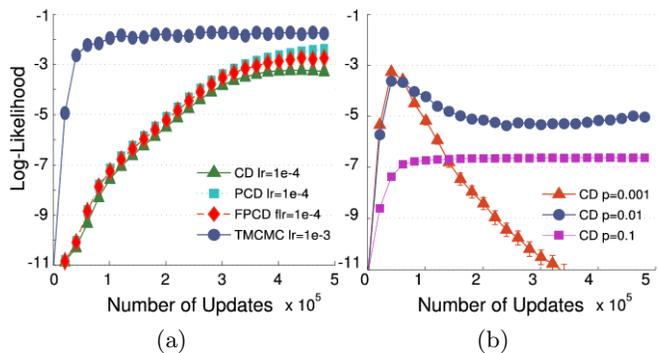


Figure 6: (a) Log-Likelihood of each algorithm for  $p = 10^{-3}$ . Learning rate is optimized based on performance after 500,000 updates. Bad mixing causes all three models to settle on small learning rates, resulting in slower convergence than TMCMC. All models use 10 hidden units. FPCD’s fast learning rate was optimized to 0.01. (b) Log-Likelihood of a model trained with CD, with a learning rate of  $10^{-3}$ . Bad mixing also causes learning to break down when modes are further apart, however it does so somewhat more gracefully than with PCD.

We then optimize the learning rates of each algorithm, such that the likelihood is optimal after 500,000 updates. Results are shown in Figure 6(a). Algorithms with poor mixing settle on a smaller learning rate of  $10^{-4}$ , in order to operate within their stable regime. This results in much slower convergence, when compared to TMCMC with a learning rate of 0.001. With enough training time (and potentially smaller learning rate), PCD should theoretically converge to the maximal value. Our results seem to reflect this for PCD and FPCD, although the same cannot be said for CD.

It is somewhat surprising that FPCD would behave so poorly on this dataset. Since it was designed to improve mixing one would expect it to behave similarly to TMCMC. It would appear that for this dataset, FPCD’s over-representation of “spurious” inter-mode

samples has a significant impact on learning performance. Observations of section 4.1 may help to shed light on the issue however.

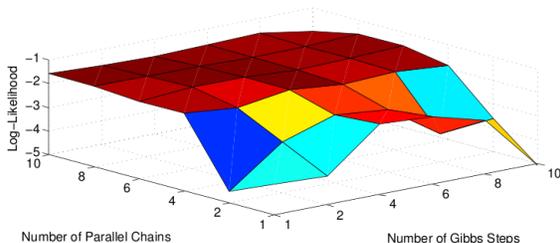


Figure 7: Likelihood of model trained on toy dataset, using the tempered algorithm. We vary both the number of chains and number of Gibbs steps in  $\{1, 2, 4, 6, 8, 10\}$ . Experiments were subject to a wall-time of 1 hour.

In our last toy experiment, we attempt to answer the following question. Given a fixed amount of resources, should one increase the number of parallel chains or increase the number of intermediate Gibbs steps? Using a tempered model, we optimized training likelihood for a variety of configurations, given a wall-time of 1 hour. The results of Fig. 7, show that while increasing  $k$  is initially advantageous, the gain in statistical performance is eventually offset by its high computational cost. On the other hand, tempering only requires four chains to achieve maximal performance.

### 4.3 ESTIMATING LIKELIHOOD ON MNIST

While computing the exact likelihood under the trained model is intractable for RBMs, it is however *relatively* easy to generate samples from it. We thus set out to compute a *quantitative* measure based on the “quality” of the samples generated by each model. 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 allows us to evaluate how well each training procedure is able to capture the training distribution.

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

Formally our sample generation quality measure is:

$$\ell(\mathcal{D}_{\text{test}}, \mathcal{D}_s) = \frac{1}{n} \sum_{k=1}^n \log \hat{p}_{\sigma, \mathcal{D}_s}(x^{(k)}) \quad (5)$$

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

The procedure is as follows: (1) we find an optimal  $\sigma$  by performing a grid search trying to maximize the log probability of test samples  $\mathcal{D}_{\text{test}}$  under  $\hat{p}_{\sigma, \mathcal{D}}$ . This value is then kept fixed. (2) we generate  $\mathcal{D}_s$  using the desired sampling procedure and a trained model, with  $n' = 10,000$ . (3) we then compute  $\ell(\mathcal{D}_{\text{test}}, \mathcal{D}_s)$  as defined in Eq. 5. We repeat steps one and two for all models and sampling procedures under consideration.

Table 1 reports the generation quality measure obtained for CD-1, PCD-1, FPCD-1 and TMCMC. The tempered model was trained using 50 parallel chains and a single Gibbs steps. Each weight update is thus more expensive than with the other methods. We consider three sampling algorithms: a standard Gibbs sampler whose chain is initialized either from random or from a test example and parallel tempering. Model selection was performed by optimizing the hyperparameters based on the likelihood score. Hyperparameters are thus selected such that the trained model is compatible with the sampling procedure.

As shown in Table 1, 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  $\sigma$  should be optimized independently for each model  $\hat{p}_{\sigma, \mathcal{D}_s}(\mathbf{x})$ . 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 4.1. 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.

## 5 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

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.

Training procedure	Sampling procedure		
	TMCMC	GIBBS (RANDOM)	GIBBS (TEST)
TMCMC	208.26	<b>210.72</b>	209.83
FPCD	180.41	174.87	175.92
PCD	80.06	127.95	139.36
CD	-1978.67	-854.08	37.18

typically used until now. The tempered MCMC sampling technique allows for better mixing of the underlying chain used to generate samples from the model. We have shown that this results in better generative models, from 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. Future work should investigate how TMCMC can compete with its competitors, on real world datasets, when equal computation time is allocated to each procedure. More experiments are also required to assess the impact of this method on classification, especially in the context of deep architectures. 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

- Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2, 1–127. Also published as a book. Now Publishers, 2009.
- Bengio, Y., & Delalleau, O. (2009). Justifying and generalizing contrastive divergence. *Neural Computation*, 21, 1601–1621.
- Carreira-Perpiñan, M. A., & Hinton, G. E. (2005). On contrastive divergence learning. *AISTATS'2005* (pp. 33–40). Savannah Hotel, Barbados: Society for Artificial Intelligence and Statistics.
- Freund, Y., & Haussler, D. (1994). *Unsupervised learning of distributions on binary vectors using two layer networks* (Technical Report UCSC-CRL-94-25). University of California, Santa Cruz.
- Hinton, G. E. (1999). Products of experts. *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN)* (pp. 1–6). Edinburgh, Scotland: IEE.
- Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14, 1771–1800.
- Hinton, G. E., Osindero, S., & Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18, 1527–1554.
- Iba, Y. (2001). Extended ensemble monte carlo. *International Journal of Modern Physics, C12*, 623–656.
- Le Roux, N., & Bengio, Y. (2008). Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation*, 20, 1631–1649.
- Neal, R. M. (1994). *Sampling from multimodal distributions using tempered transitions* (Technical Report 9421). Dept. of Statistics, University of Toronto.
- Salakhutdinov, R. (2010). Learning in Markov random fields using tempered transitions. *Advances in Neural Information Processing Systems 22 (NIPS'09)*.
- Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland (Eds.), *Parallel distributed processing*, vol. 1, chapter 6, 194–281. Cambridge: MIT Press.
- Tieleman, T. (2008). Training restricted boltzmann machines using approximations to the likelihood gradient. *ICML 2008* (pp. 1064–1071). Helsinki, Finland: ACM.
- Tieleman, T., & Hinton, G. (2009). Using fast weights to improve persistent contrastive divergence. *ICML 2009* (pp. 1033–1040). ACM.
- Welling, M., Rosen-Zvi, M., & Hinton, G. E. (2005). Exponential family harmoniums with an application to information retrieval. *NIPS 17*. MIT Press.
- Younes, L. (1998). On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics and Stochastics Models* (pp. 177–228).