Infinite Swapping Algorithm for Training Restricted Boltzmann Machines

Henrik Hult, Pierre Nyquist, and Carl Ringqvist

Abstract Given the important role latent variable models play, for example in statistical learning, there is currently a growing need for efficient Monte Carlo methods for conducting inference on the latent variables given data. Recently, Desjardins et al. (2010) explored the use of the parallel tempering algorithm for training restricted Boltzmann machines, showing considerable improvement over the previous state-of-the-art. In this paper we continue their efforts by introducing the infinite swapping algorithm, an MCMC method first conceived when attempting to optimise performance of parallel tempering (Dupuis et al. (2012)). We implement a Gibbs-sampling version of infinite swapping and evaluate its performance on a number of test cases, concluding that the algorithm enjoys better mixing properties than both persistent contrastive divergence and parallel tempering for complex energy landscapes associated with restricted Boltzmann machines.

1 Introduction

Consider a latent variable model with probability density of the form

$$p(\mathbf{v}) = \frac{1}{Z(\theta)} \sum_{\mathbf{h}} \exp\{-E_\theta(\mathbf{v},\mathbf{h})\}, \tag{1}$$

where $\mathbf{v}$ represent visible units and $\mathbf{h}$ hidden units, $Z(\theta)$ is an unknown normalising constant, and θ an unknown parameter. The function E_θ is referred to as the *energy* function. Training such models by maximum likelihood through, e.g., Stochastic Gradient Descent (SGD) using a training set of independent samples of visible units often requires Markov chain Monte Carlo (MCMC) methods. When the energy

KTH Royal Institute of Technology, Lindstedtsvagen 25, 100 44, Stockholm, Sweden. e-mail: hult@kth.se, e-mail: pierren@kth.se, e-mail: carrin@kth.se

landscape is complex, convergence to the desired stationary distribution may be slow because the Markov chain tends to get stuck near local minima. In the context of training in machine learning, this phenomenon might result in significant gradient estimation error.

A simple, yet interesting model of the form (1) is the Restricted Boltzmann Machine (see Section 2), which is prominent in statistical learning and used in various deep architectures. It is particularly successful in collaborative filtering, for instance in assigning ratings of movies to users, see [28]. Training of restricted Boltzmann machines have been gradually improved from contrastive divergence [16, 17], to persistent contrastive divergence [32] and most recently parallel tempering [3]. In a sense this paper continues the effort initiated in [3] by proposing the infinite swapping (INS) algorithm, designed to overcome rare-event sampling issues, for training restricted Boltzmann machines. Moreover, we investigate via an empirical study the impact the choice of training algorithm has on classification. This partially answers a question posed in [3], namely what impact the use of parallel tempering, and more generally extended ensemble Monte Carlo methods, may have on classification tasks.

Parallel tempering (PT) [9, 12, 31] has become a standard tool for molecular dynamics simulations, see for example [9, 13, 19, 22, 26, 30] and the references therein. The idea is that for models where there is a parameter acting like a temperature - the canonical case is a Gibbs measure and the associated (inverse) temperature, similar to (1) - one runs multiple Markov chains, each with a different "temperature", and couple them via swaps of the particle locations at random times according to a given intensity, see Section 3 for further details.

The infinite swapping algorithm was introduced in [8] as an improvement of parallel tempering, with documented success in a variety of chemical and biological physics settings. Consequently, it serves as a natural candidate for potentially improving training of machine learning models. It can be viewed as the limit of PT when the swap rate is sent to infinity; in [8] the corresponding sampling scheme is shown to be optimal from a large deviations perspective and in [5] a more in-depth analysis of PT and INS is carried out in the setting of continuous-time jump Markov processes. Recently [23] studied the ergodicity properties of INS at low temperature, deriving Eyring-Kramers formulae for the spectral gap and the log-Sobolev constant, showing superiority of infinite swapping over overdamped Langevin dynamics.

In addition to the theoretical results of [8, 5, 23] on the properties of PT and INS, recent empirical studies show superior performance of INS compared to PT and other Monte Carlo methods for a range of common performance measures [24, 7, 4]. So far the main application area for INS has been chemical and biological physics and adjacent areas, see for example [6, 21, 34, 25]. However, as extended ensemble methods, such as PT, are becoming increasingly popular for MCMC simulations in a wide range of areas, it is natural to consider INS in the same settings. Statistical learning and latent variable models is one such example where metastability is often a hindrance in the training phase.

In this paper we propose to use the INS algorithm in the training phase of a restricted Boltzmann machine to improve mixing of the underlying Markov chain

and to facilitate accurate estimation of gradients. The contributions of this paper include

- an implementation of a Gibbs sampling version of the infinite swapping algorithm for latent variable models,
- details on the infinite swapping algorithm for training restricted Boltzmann machines,
- empirical comparison of the performance of training restricted Boltzmann machines using infinite swapping, parallel tempering, and persistent contrastive divergence.

So far infinite swapping has mainly been considered for Langevin and Glauber dynamics [8, 21, 5] in continuous time. The paper [8] also contains discrete-time large deviations results and discusses the corresponding sampling schemes. For the application to restricted Boltzmann machines, the large size of the state space under consideration, although discrete, renders Glauber dynamics unsuitable because of the need to compute the full transition matrix. To the best of our knowledge this paper is the first to implement a Gibbs sampling version of INS that circumvents this computational issue. Although the present study is limited to restricted Boltzmann machines, it is plausible that this Gibbs sampling version of the infinite swapping algorithm can be further generalised to more complex latent variable models. In [2] the authors introduce Hamiltonian Monte Carlo in the setting of variational auto-encoders, another prominent latent variable model, to obtain unbiased estimators of gradients with low variance.

The remainder of the paper is organised as follows: In Section 2 the restricted Boltzmann machine is introduced. The parallel tempering and infinite swapping algorithms are presented in Section 3. An empirical study of INS performance is provided in Section 4 and the conclusions are summarised in Section 5.

2 Restricted Boltzmann machines

The Restricted Boltzmann Machine (RBM) [29, 11, 17, 33] is a probability distribution over an N-dimensional boolean space $\{0,1\}^N$. Let $\mathbf{v} \in \{0,1\}^N$, $\mathbf{h} \in \{0,1\}^M$ be row vectors and set $\mathbf{x} = (\mathbf{v},\mathbf{h}) \in \{0,1\}^{N+M}$. Let $\mathbf{W}$ be a real-valued parameter matrix of dimension $N \times M$, and let $\mathbf{b} \in \mathbb{R}^N$, $\mathbf{c} \in \mathbb{R}^M$ be real-valued bias parameter row vectors. The RBM probability function is defined as

$$p(\mathbf{v}) = Z^{-1} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}, \tag{2}$$

$$E(\mathbf{v},\mathbf{h}) = -\mathbf{v}\mathbf{W}\mathbf{h}^T - \mathbf{v}\mathbf{b}^T - \mathbf{h}\mathbf{c}^T, \tag{3}$$

An observed data point enters the RBM model as a vector of visible units $\mathbf{v} \in \{0,1\}^N$, while $\mathbf{h} \in \{0,1\}^M$ denotes accompanying hidden units, i.e., the latent variables. The combined vector $\mathbf{x} = (\mathbf{v},\mathbf{h}) \in \{0,1\}^{N+M}$ is referred to as a *particle*.

The latent structure facilitates simulation from the joint distribution through block Gibbs sampling, as block conditional probabilities $p(\mathbf{v}|\mathbf{h})$ and $p(\mathbf{h}|\mathbf{v})$ are available in explicit form and are easy to sample from. Indeed, letting $\mathbf{e}^{(n)}$ denote the n:th coordinate of any vector $\mathbf{e}$, it holds that

$$p(\mathbf{v}|\mathbf{h}) \propto \prod_{n=1}^{N} \exp\{\mathbf{v}^{(n)}(\mathbf{W}\mathbf{h}^T + \mathbf{b}^T)^{(n)}\}.$$

Let *sigm* be the sigmoid function $sigm(x) = (1+e^{-x})^{-1}$. The probability function factorises and straightforward algebra gives

$$p(\mathbf{v}^{(n)} = 1|\mathbf{h}) = sigm[(\mathbf{W}\mathbf{h}^T + \mathbf{b}^T)^{(n)}], \tag{4}$$

$$p(\mathbf{h}^{(m)} = 1|\mathbf{v}) = sigm[(\mathbf{v}\mathbf{W} + \mathbf{c})^{(m)}]. \tag{5}$$

Hence, sampling from the conditional distributions $p(\mathbf{v}|\mathbf{h})$, $p(\mathbf{h}|\mathbf{v})$ amounts to sampling independent Bernoulli variables, with probabilities extracted from the sigmoid forms (4) and (5). Samples from the joint distribution $p(\mathbf{v},\mathbf{h}) \propto \exp\{-E(\mathbf{v},\mathbf{h})\}$ can thus be obtained by running a block Gibbs Markov chain [18].

Inference for the parameters in a RBM, for a particular data point $\mathbf{v}$, is often conducted via maximum likelihood, by minimising the negative log-likelihood, $-\log p(\mathbf{v})$, with respect to the parameters $\mathbf{W},\mathbf{b},\mathbf{c}$. This is usually achieved with gradient descent methods, for which an estimate of the gradient $\nabla p(\mathbf{v})$ is needed.

A calculation of the gradient coordinate corresponding to the partial derivative w.r.t the parameter $w_{n,m}$ at row n and column m in the matrix $\mathbf{W}$ yields (see [10] for details)

$$\nabla_{w_{n,m}}(-\log p(\mathbf{v})) = \ldots = -\mathbf{v}^{(n)} p(\mathbf{h}^{(m)} = 1|\mathbf{v}) + \mathrm{E}[\mathbf{v}^{(n)} p(\mathbf{h}^{(m)} = 1|\mathbf{v})]. \tag{6}$$

The first and second term on the right-hand side of (6) are often referred to as the *positive phase* and the *negative phase*, respectively. The negative phase is the problematic term as it amounts to taking expectation under the joint distribution of $(\mathbf{v},\mathbf{h})$. However estimates of this part of the gradient can be obtained via the Gibbs sampling procedure.

When the number of data points $|\mathbf{D}| = d$ is large, the standard technique for minimising the average negative log-likelihood is (mini-batch) stochastic gradient descent (SGD). In the SGD method, a subset $\mathbf{D}' \subset \mathbf{D}$ of size $|\mathbf{D}'| = d' < d$ is chosen at random and the gradient coordinate w.r.t. $w_{n,m}$ at current parameter state is estimated by

$$\frac{1}{d'} \sum_{\mathbf{v} \in \mathbf{D}'} -\mathbf{v}^{(n)} p(\mathbf{h}^{(m)} = 1|\mathbf{v}) + \mathrm{E}[\mathbf{v}^{(n)} p(\mathbf{h}^{(m)} = 1|\mathbf{v})]. \tag{7}$$

Here the expectation is taken w.r.t to $p(\mathbf{v})$. Similar to standard gradient descent, the estimate of the gradient is used to update the parameter vector in step $n+1$ through

$$(\mathbf{W},\mathbf{b},\mathbf{c})_{n+1} = (\mathbf{W},\mathbf{b},\mathbf{c})_n - \eta(\widetilde{\nabla\mathbf{W}},\widetilde{\nabla\mathbf{b}},\widetilde{\nabla\mathbf{c}})$$

where η is a scalar learning rate, and where $\widetilde{\nabla\mathbf{x}}$ denotes the estimated gradient coordinates for the matrix/vector $\mathbf{x}$.

For RBMs, where the negative phase is estimated using Gibbs sampling, SGD requires simulations to be run for each training step. Usually, a Gibbs chain of size d' is run for a fixed number of κ steps before an average is formed with the end samples. Starting the Gibbs chain anew *at sampled data points* in each gradient step is referred to as the contrastive divergence (CD-κ) training method. Since long burn-in periods might be expected with this approach, the Gibbs chain for a certain training step is typically started at the last samples of the previous training step. This method is referred to as persistent contrastive divergence (PCD-κ). Current state-of-the-art method for training RBMs is arguably a combination of PCD-1 and PT.

3 Parallel tempering and infinite swapping for Gibbs samplers

Consider the setting of Section 2, that is a RBM trained with SGD with batch size d'. Parallel tempering amounts to multiple Gibbs chains of size d' being run at different *temperatures* and particles being exchanged according to a Metropolis-Hastings rule. In the case of two temperatures, in addition to the original model (2) with temperature $\tau_1 = 1$, an additional RBM with a higher temperature $\tau_2 > 1$ is introduced:

$$p_{\tau_2}(\mathbf{v}) = Z_{\tau_2}^{-1} \sum_{\mathbf{h}} e^{-\frac{1}{\tau_2}E(\mathbf{v},\mathbf{h})}.$$

It is easy to check that all calculations above for $\tau_1 = 1$ carry over in a straightforward manner to a RBM with $\tau_2 > 1$. The PT method proceeds by running two Gibbs chains C_1, C_2, at respective temperature, each of size d'. Let $\mathbf{x}_{1,1},...,\mathbf{x}_{1,d'}$, $\mathbf{x}_{2,1},...,\mathbf{x}_{2,d'}$ denote the particles in C_1, C_2 respectively. After κ Gibbs steps, particles $\mathbf{x}_{1,i}, \mathbf{x}_{2,i}$ $i = 1,...,d'$ are swapped with probability

$$1 \wedge \frac{\exp\{-\frac{1}{\tau_1}E(\mathbf{x}_{2,i}) - \frac{1}{\tau_2}E(\mathbf{x}_{1,i}\}}{\exp\{-\frac{1}{\tau_1}E(\mathbf{x}_{1,i}) - \frac{1}{\tau_2}E(\mathbf{x}_{2,i})\}}.$$

Swaps of this kind are attempted according to the so-called *swap rate* (the jump intensity) of the algorithm. The process then starts anew with running C_1, C_2, at their respective temperatures for another κ Gibbs steps. The resulting process is ergodic with the product measure $p_{\tau_1} \otimes p_{\tau_2}$ as stationary distribution. Thus, the chain C_1 converges in distribution to p_{τ_1}, the distribution of interest.

There are several ways of extending PT to additional temperatures. Here, we follow the common approach of only attempting swaps between neighbouring particles, see [20]. For K chains with respective temperatures $\tau_1 < ..., < \tau_K$, all swaps of the

form $[\mathbf{x}_{k,i}, \mathbf{x}_{k+1,i}] \to [\mathbf{x}_{k+1,i}, \mathbf{x}_{k,i}]$ are attempted after every κ Gibbs step, starting at $k = 1$ and working upward to $k = K - 1$. The swapping probabilities used are thus

$$1 \wedge \frac{\exp\{-\frac{1}{\tau_k}E(\mathbf{x}_{k+1,i}) - \frac{1}{\tau_{k+1}}E(\mathbf{x}_{k,i}\}}{\exp\{-\frac{1}{\tau_k}E(\mathbf{x}_{k,i}) - \frac{1}{\tau_{k+1}}E(\mathbf{x}_{k+1,i})\}}.$$

The swapping mechanism limits the degree of dependency between samples and forces quicker mixing of samples, thus speeding up the convergence of C_1.

Compared to parallel tempering, the infinite swapping algorithm proposes a different mechanism for exchanging information between the tempered Gibbs chains. In PT, the particle exchange probabilities are given but the proposed changes (only neighbouring particles, etc.) can be chosen. In INS the full mechanism for exchange is used; in a sense, *all* possible swaps are attempted.

Consider K chains $C_1, ..., C_K$ with temperatures $\tau_1 < \tau_2 < ... < \tau_K$ and assume each chain contains only one particle (extending to the case of d' particles is straightforward). Denote by $\mathbf{x}_k = (\mathbf{v}_k, \mathbf{h}_k)$ the particle in chain k and let $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_K)$ denote the vector of particles. Let $\sigma_j, j = 1, ..., K!$ denote a permutation of the temperature indices $[1, ..., K]$, for some ordering of the $K!$ permutations. Write the RBM probability function of temperature τ_k as $p_k(\mathbf{x}) = Z_k^{-1} \exp\{-E(\mathbf{x})/\tau_k\}$. Define the *symmetrised distribution* $\bar{p}$ and the joint probability distribution p_{σ_j} across the chains for permutation σ_j as

$$\bar{p}(\mathbf{X}) = \frac{1}{K!}\sum_{j=1}^{K!} p_{\sigma_j}(\mathbf{X}), \quad p_{\sigma_j}(\mathbf{X}) = \prod_{k=1}^{K} p_{\sigma_j^k}(\mathbf{x}_k),$$

where σ_j^k denotes the k:th component of the permutation σ_j. The INS algorithm consists of first running the Gibbs chains independently for κ Gibbs steps; each Gibbs-step amounts to first sampling a point $\mathbf{h}$ from $p(\mathbf{h}|\mathbf{v})$ and then a point $\mathbf{v}$ from $p(\mathbf{v}|\mathbf{h})$. In the next step *temperatures* are swapped between the chains according to permutation σ_j with probability

$$\rho_{\sigma_j}(\mathbf{X}) = \frac{p_{\sigma_j}(\mathbf{X})}{\sum_{j=1}^{K!} p_{\sigma_j}(\mathbf{X})} = \frac{p_{\sigma_j}(\mathbf{X})}{K!\bar{p}(\mathbf{X})}.$$

The procedure is then repeated, resulting in a Markov chain sample generation scheme. The collective Markov chain $(C_1, ..., C_K)$ with the INS temperature swapping mechanism is referred to as the *INS-Gibbs Markov chain*. The isolated κ Gibbs steps together with one swapping operation will be referred to as an *INS-Gibbs step* of the INS-Gibbs Markov chain. Note that for large K, since the number of permutations is $K!$, the computational cost of INS can be too high for practical purposes. One can then use so-called *partial INS* (PINS) [8], in which temperatures are arranged into subgroups, swaps are attempted within one such subgroup at a time and with a handoff-rule for changing between subgroups. This reduces the computational cost of INS significantly. For example, in [4] $K = 30$ temperatures are used for a Lennard-

Jones model of 55-atoms argon cluster. Similarly, in [8] a collection of $K = 45$ temperatures are used for a Lennard-Jones cluster of 38 atoms. In these complex potential energy landscape the partial infinite swapping approach is appreciably more effective than conventional tempering approaches; see [7] for an extensive numerical study.

To obtain an estimate of $\mathrm{E}_{p_1}[f(\mathbf{x})]$ for any real-valued function f, a weighted average of the particles of each chain is formed:

$$\sum_{k=1}^{K} f(\mathbf{x}_k)\Big(\sum_{\sigma;\sigma^k=1} \rho_\sigma(\mathbf{X})\Big), \tag{8}$$

where $\{\sigma : \sigma^k = 1\}$ is the subset of permutations that have 1 as their k:th component (that is, that assign the k:th chain temperature τ_1). Proofs that the INS-Gibbs Markov kernel has the symmetrised distribution $\bar{p}$ as invariant distribution, and that the estimate (8) has the desired expected value $\mathrm{E}_{p_1}[f(\mathbf{x})]$ if samples are generated from $\bar{p}$, are included in the Appendix as Proposition 1 and 2, respectively. The proofs are similar to existing results and are included for completeness.

Because the sample space is finite and the Markov chains are irreducible, Proposition 1 ensures that the empirical measures of the full chain converges to the symmetrised distribution. Consequently, if the INS-Gibbs Markov chain is run long enough, its empirical measure will approximate the symmetrised distribution.

Algorithm 1 is an outline of the INS algorithm for obtaining an estimate of $E_{p_1}[f(\mathbf{x})]$ in the setting of Gibbs-sampling. For training a RBM, the last step of

Algorithm 1 INS-Gibbs Algorithm

1. Set number of chains K, temperature values $[\tau_1, ..., \tau_K]$, number of Gibbs steps κ between swap attempts, number of swap attempts q and initial data points.
2. Start chains with initial data points.
3. **for** i in $1 : q$
   a. Run each chain for κ Gibbs steps.
   b. Draw permutation σ with probability ρ_σ
   c. Permute the temperatures of the chains according to permutation σ
4. Form an estimate as

$$y = \sum_{k=1}^{K} f(\mathbf{x_k})\Big(\sum_{\sigma;\sigma^k=1} \rho_\sigma(\mathbf{X})\Big)$$

the algorithm corresponds to using the gradient negative phase estimate as f and the tempered RBM joint distributions for forming the weights. While obtaining an estimate as described in the algorithm, one can use the so-called particle/temperature-associations as a diagnostic of non-convergence. These are quantities than can be computed while running the algorithm and used to indicate whether it is possible for the empirical measure to have converged; see [4, 5] for details and an analysis of this diagnostic.

4 Numerical experiments

In order to evaluate the performance of INS for training RBMs a series of numerical experiments are conducted. Two types of data sets, described in Section 4.3, are considered. For the smaller data sets the exact likelihood and exact gradient can both be computed, which enables comparison of the training algorithms. For larger data sets neither the exact likelihood nor the exact gradient is tractable. Instead a classification Boltzmann machine will be used to evaluate the training algorithms; the quantity used for comparison is referred to as *prediction accuracy*.

4.1 Prediction accuracy

To compare training algorithms using a classification Boltzmann machine, each data point in the data set is concatenated with a vector $\mathbf{c} \in C$ representing its class, where C denotes the subspace of $\{0,1\}^{dim(\mathbf{c})}$ such that exactly one coordinate is nonzero, and $dim(\mathbf{c})$ is the number of classes. Such a vector is called a *one-hot* vector in the machine learning literature. For a RBM defined on an extended visible state space $\tilde{\mathbf{v}} = (\mathbf{v}, \mathbf{c})$, the conditional probability for class type $\mathbf{c}$ given $\mathbf{v}$ can be computed explicitly: it holds that

$$p(\mathbf{c}|\mathbf{v}) = \frac{\hat{p}(\mathbf{c},\mathbf{v})/Z}{\hat{p}(\mathbf{v})/Z} = \frac{\hat{p}(\mathbf{c},\mathbf{v})}{\hat{p}(\mathbf{v})} = \frac{\hat{p}(\mathbf{c},\mathbf{v})}{\sum_{\mathbf{c}\in C}\hat{p}(\mathbf{c},\mathbf{v})},$$

and the terms on the right-hand side can be computed with the *marginalisation trick* (see the Appendix for a description). The above expression can be used for classification, and for calculating classification error through comparing with the actual class type for data points. The efficiency of the algorithms can be evaluated by measuring the classification capabilities of each parameter state during training on the extended data sets. More specifically, for each parameter state the *prediction accuracy*,

$$A = \sum_{(\mathbf{v}_i,\mathbf{c}_i)\in\mathbf{D}'} \log\Big(\frac{\hat{p}(\mathbf{c}_i,\mathbf{v}_i)}{\sum_{\mathbf{c}\in C}\hat{p}(\mathbf{c},\mathbf{v}_i)}\Big),$$

is calculated, where $\mathbf{D}'$ is a randomised subset of data, for each state during training.

4.2 Parameters and settings

This section describes the parameters involved in the training of RBMs and the choices made for this work. The main objective of the numerical experiments is a comparison with [3], where performance results for PT in the RBM setting are presented. Parameters have therefore been selected accordingly, and no attempts have been made to find optimal parameter settings for the sampling and training tasks under consideration. Where possible, external recommendations have been taken into account, as have empirical observations from experiments on the impact of parameter changes.

4.2.1 Learning rate

The learning rate η can be chosen as a fixed constant or as a function of, e.g., the number of updates in SGD. A large learning rate increases the speed of training but may result in inaccurate optima or degenerate behaviour, while a small learning rate allows for greater accuracy at the cost of training speed. Numerical experiments indicate that a learning rate of 0.1 yields satisfactory performance, and this value is chosen throughout. This is also consistent with experiments in [3].

4.2.2 Initialization

Throughout the experiments, the initial weights and biases are drawn randomly from normal distributions of small variance (order of magnitude 0.01) and zero expected value, in line with the recommendations in [15]. Empirically, different draws do not yield different results, nor smaller changes in the variance parameter.

4.2.3 Training steps

The number of training steps are set to 10000 in all the numerical experiments, as all the relevant effects seems to appear within this range. Furthermore, in the examples of lower dimension, empirical observations suggest the likelihood is maximal after this number of steps. If the training rate is decreased or if the data dimension is significantly increased, more training steps would likely be needed.

4.2.4 Batch size

We set the batch size to be 10 thoroughout. Empirical studies show that this batch size is sufficient for non-degenerate behaviour of the likelihood. A larger batch size yields a more precise gradient estimate, but causes longer computational time. Therefore, for computational efficiency, a small batch size is preferred.

4.2.5 Temperatures

Both PT and INS run on temperatures $\{1, 2, 3, 4, 5\}$. Empirical studies show no significant difference when adding more temperatures for the examples under consideration. However, for certain parameter setups, fewer temperatures can result in degenerate behaviour similar to what is observed for a single chain, see Section 4.4. At five temperatures, no notable difference in running time between INS and PT is present, allowing for a fair comparison of the algorithms. For a large number of temperatures, the PT algorithm is considerably faster and a fair comparison would then be with PINS rather than INS. Moreover, mixing properties are satisfactory for the temperatures selection. This is also in line with demonstrations in [3], where 5 temperatures are considered (of roughly the same magnitude, however the exact temperature values are not disclosed).

4.2.6 Number of swaps

For PT, one swap is attempted for every step in the Markov chain. This choice makes the experiments consistent with [3]. Moreover, it is experimentally observed that an increase in the number of swap attempts per training step does not seem to have any significant effect on the results.

4.3 Data sets

We use two types of data sets for empirical evaluation of the INS algorithm for training a RBM. The first is a collection of toy data sets similar to that used in [3]; by changing the size of the toy data sets we can move from cases where the gradient can be computed exactly to those where this is not possible for either gradients or likelihoods. The second type of data set we consider is the well-known MNIST data (described in detail in a following subsection).

Toy data

The toy data sets are generated according to a generalisation of the "Toy Data" generating mechanism in [3]. The procedure involves choosing the number of modes μ, the distance between modes δ, the number of samples per modes ν and a permutation probability π; we require the number of modes to be a power of 2. The data is generated as follows:

1. Compute $i = \log_2(\mu)$, the number of binaries needed to encode the modes.
2. Create a list of the binary encoding of each mode.
3. Create a list of expanded mode encodings by expanding each encoding in 2. with δ copies.

4. Generate ν copies of each expanded mode encoding from 3.
5. Flip every binary variable in 4. independently with probability π, to obtain the *explanatory data*.
6. Add a one-hot vector representing mode type to get the joint *explanatory and response data*.

The following example is for (μ, δ, ν, π) = (4, 2, 2, 0.2) :

1. Let $i = \log_2(4) = 2$
2. $(1,1)$, $(1,0)$, $(0,1)$, $(0,0)$
3. $(1,1,1,1)$, $(1,1,0,0)$, $(0,0,1,1)$, $(0,0,0,0)$
4. $(1,1,1,1)$, $(1,1,1,1)$, $(1,1,0,0)$, $(1,1,0,0)$, $(0,0,1,1)$, $(0,0,1,1)$, $(0,0,0,0)$, $(0,0,0,0)$
5. $(1,1,1,0)$, $(1,1,1,1)$, $(1,1,0,1)$, $(1,1,1,0)$, $(0,0,1,0)$, $(0,0,1,1)$, $(0,0,0,0)$, $(1,0,0,0)$
6. $(1,1,1,0,1,0,0,0)$, $(1,1,1,1,1,0,0,0)$, $(1,1,0,1,0,1,0,0)$, $(1,1,1,0,0,1,0,0)$, $(0,0,1,0,0,0,1,0)$, $(0,0,1,1,0,0,1,0)$, $(0,0,0,0,0,0,0,1)$, $(1,0,0,0,0,0,0,1)$

In [3], π is varied over a toy data set without class type attached, with $\mu = 4, \delta = 8$, and $\nu = 2500$. This example is treated in Section 4.4.

MNIST

The MNIST data set is used as a benchmark for training and evaluating machine learning algorithms[1]. It consists of 55000 pictures of handwritten images of numbers $0, \ldots, 9$. One data point is a 28x28 matrix populated with grayscale pixel numbers between 0 and 1; Figure 1 shows two examples from the data set. In this work we round each pixel to 0 or 1 in order for the data to fit the binary RBM as described. Attached to each image is also a one-hot vector of dimension 10 representing number type.

Fig. 1 Two examples from the MNIST data set

[1] The data set, and more information about it, is available on Yann LeCun's webpage: http://yann.lecun.com/exdb/mnist/.

4.4 Evaluating the INS algorithm for small toy data sets

Consider a small toy data set for which the likelihood can be computed exactly and can be trained with exact SGD (exact gradient computed for a data subsample in each step). We compare four algorithms:

- exact SGD
- SGD PCD-1
- SGD PCD-1 with Parallel Tempering
- SGD PCD-1 with INS-Gibbs

The exact likelihood and the prediction accuracy is computed for each parameter state during each training algorithm. In addition, for every step in the respective Markov chain for PT and INS, the Euclidean distance between the true gradient and the gradient estimate is computed at two different fixed parameter states (early and late in the training). This empirical evaluation provides insight into the effectiveness of the INS-Gibbs algorithm. Moreover, it allows us to compare the non-standard performance measure prediction accuracy with the likelihood.

The first toy data set was generated using the following parameters: $(\mu, \delta, \nu, \pi) = (4, 4, 2500, 0.2)$. That is, the number of visible units is 12 (including 4 dimensions for class type one-hot vectors), and the number of data points is 10000 (see Section 4.3). For the RBM model, the number of hidden units was set to $M = 4$, the starting points of the Markov chains were drawn uniformly and the training procedures were repeated 20 times for each of the algorithms [2]. The result of the experiment is illustrated in Figures 2 and 3.

[2] All stochastic behaviour (except the data generation) were run on updated seeds in every iteration.

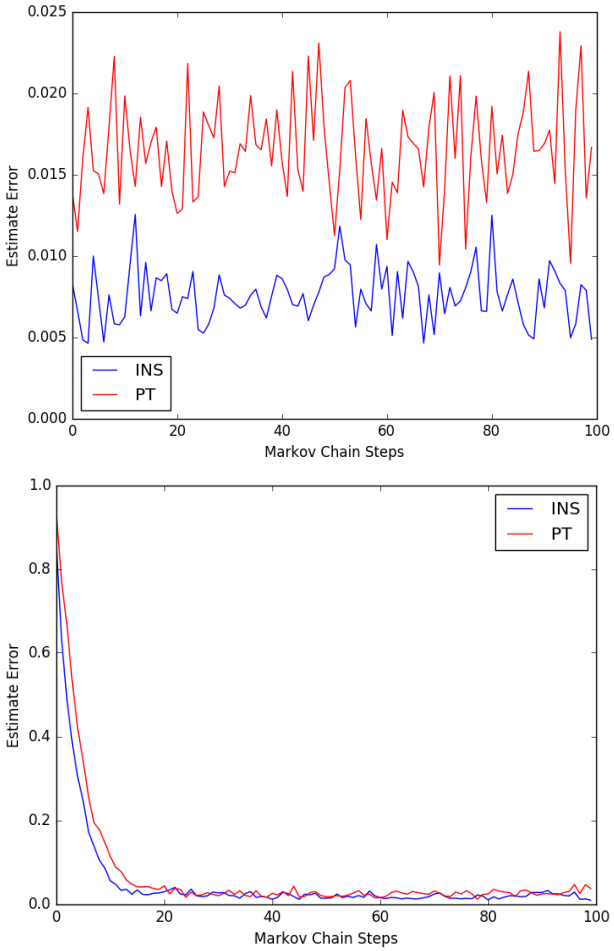

Fig. 2 Evaluation of different training algorithms on a small toy dataset. **Upper**: Euclidean distance from the gradient estimate to the true gradient, for the initial parameter state, as function over Markov chain steps. **Lower**: Euclidean distance from the gradient estimate to the true gradient, for the final parameter state, as function over Markov chain steps.

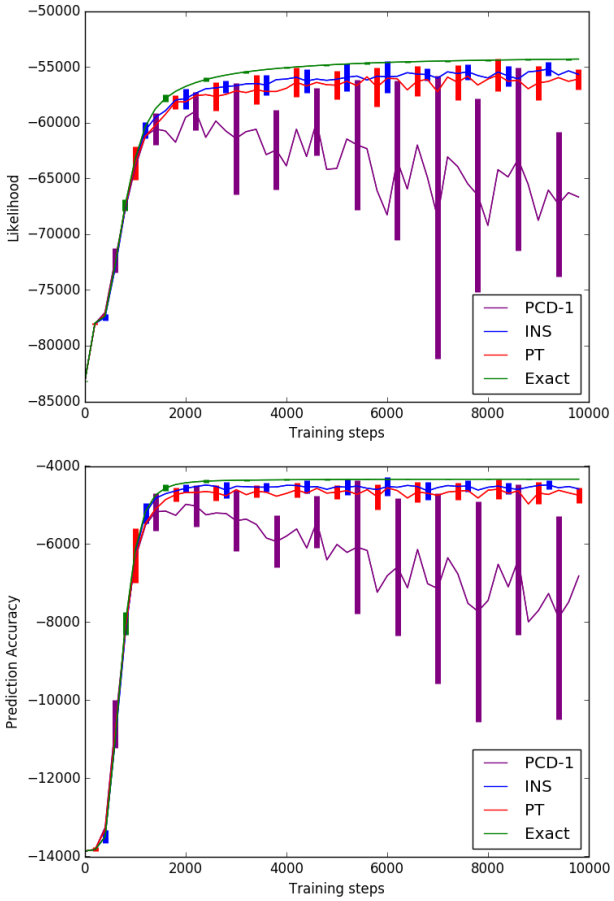

Fig. 3 Evaluation of different training algorithms on a small toy dataset. **Upper**: Average likelihood trajectory and variation (parameter state likelihood variance estimate). **Lower**: Average prediction accuracy trajectory and variation (parameter state prediction accuracy variance estimate).

Next, the experiments of [3] were recreated by using $(\mu, \delta, \nu, \pi) = (4, 8, 2500, 0.2)$. The number of hidden units was set to $M = 6$, all other parameters as in the previous experiment. The results are illustrated in Figures 4 and 5.

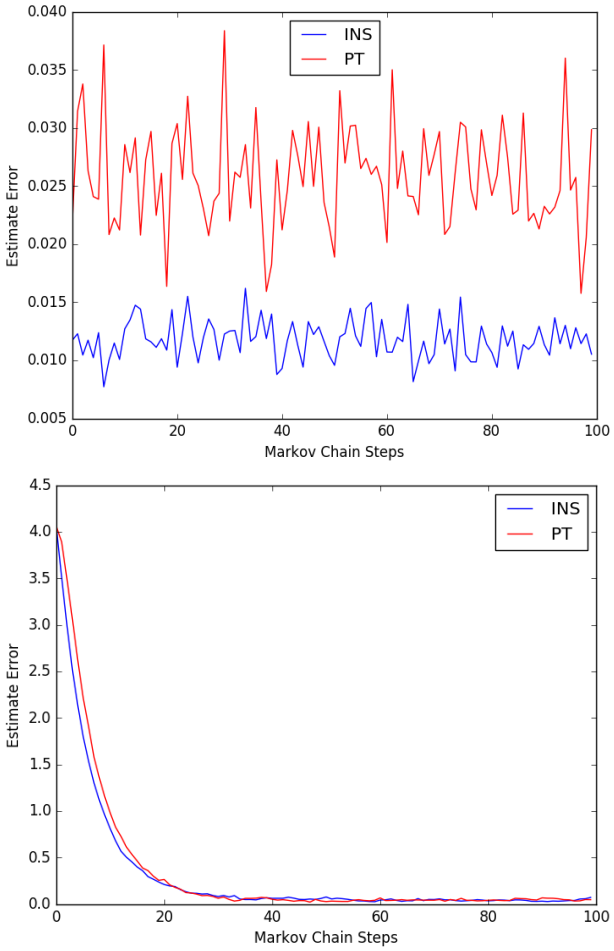

Fig. 4 Evaluation of different training algorithms on a data set generated according to [3]. See Figure 1 and 2 for subgraph description.

Figures 1-4 show the INS algorithm generally outperforming PCD-1 and being slightly superior to PT, consistent with previous studies in different contexts. Early in the training phase gradient estimation is relatively easy and deviations from the true gradient (as a function over Markov steps) is small. However, INS seems to produce smaller estimation error and variance, resulting in a more stable behaviour closer to the true gradient. Estimating the gradient becomes harder later in the training phase and both algorithms needs to be run for a considerable number of steps in order to converge. Again, we note that INS outperforms the other algorithms, converging slightly faster towards 0 distance to the true gradient than PT. The likelihood and

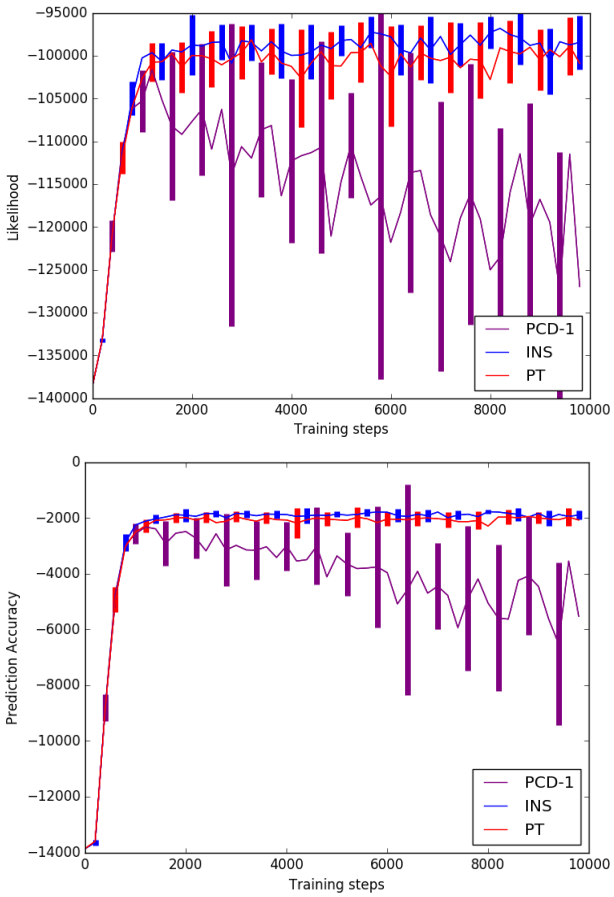

Fig. 5 Evaluation of different training algorithms on a data set generated according to [3]. See Figure 1 and 2 for subgraph description.

prediction accuracy graphs paint a similar picture: both show INS performing better, in terms of average behaviour as well as estimated variance.

Remark 1. The distinction between algorithms regarding the gradient estimate quality can be expected to increase with decreasing π, as the modes in the data become more separated; this should increase the importance of good mixing. However, for the prediction accuracy to be able to distinguish between algorithms, the classification problem must be sufficiently difficult, motivating the choice of $\pi = 0.2$. Indeed, if π is too small, mixing becomes harder but classification simpler and the algorithms will all exhibit similar prediction accuracy.

4.5 Evaluating the INS algorithm for a larger toy data set and MNIST

The performance of the training algorithm will now be evaluated on the MNIST data and a larger toy data set. Neither the exact training gradients nor the exact likelihood computations are now available. However, we can still use the prediction accuracy to evaluate the training algorithms. For the toy data generation the parameters were set to $(\mu, \delta, \nu, \pi) = (128, 150, 1000, 0.2)$, i.e the number of visible units is 1050, and the number of data points is 128000. Moreover, the number of hidden units were set to $M = 600$ for the toy data set and $M = 500$ for MNIST, remaining parameters were set as in the previous experiments. The outcome is illustrated in Figure 6.

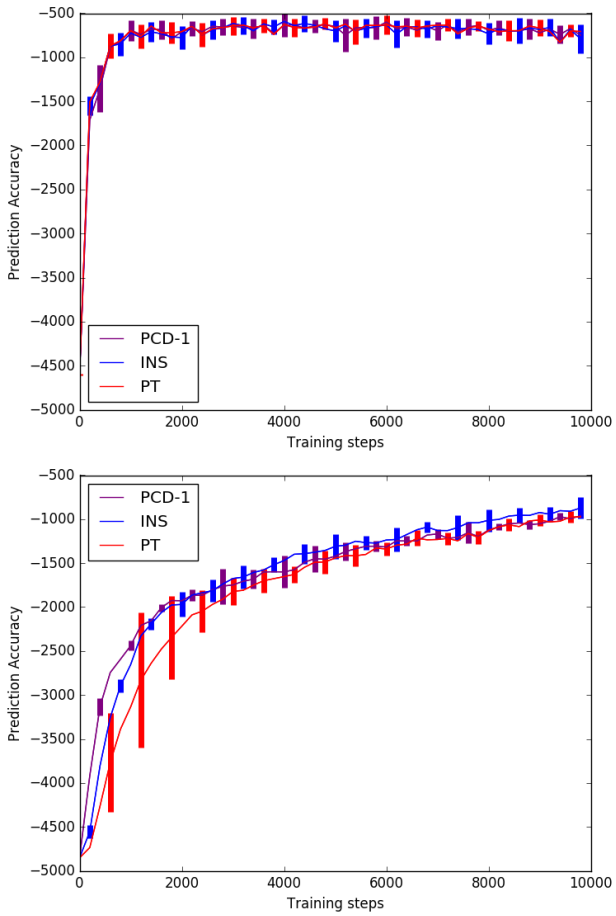

Fig. 6 Average prediction accuracy trajectory and variation for each training algorithm on the MNIST data set (upper) and the large toy dataset (lower).

From Figure 6, for the MNIST data INS enjoys the smallest variance initially but as the number of steps increase the differences between the algorithms have all but disappeared. Similarly for the toy data set; interesting to note for the toy data set is that PCD-1 does not display degenerate behaviour as in [3].

Remark 2. The results for MNIST, in particular when compared to those for likelihood estimation in [3], may be explained by the classification task being too simple, the modes being too few and too distinguished. In the large toy data set however, modes are greater in number and more similar. Here, we again observe a small difference in performance, both in terms of average behavior and variance. A reason might be that energy landscapes in higher dimensions tend to be less equipped with poor (in terms of training) local minima, putting less demand on gradient estimate quality [27].

5 Conclusions

We have presented the INS algorithm in a Gibbs-sampling setting for training RBMs, and conducted an empirical study of the performance of INS compared to persistent contrastive divergence and parallel tempering. The INS algorithm performs superior or equal to all other training algorithms, for the cases investigated; the difference is most notable for smaller data sets. One possible explanation is that the gradient estimate becomes hard late in training as the energy landscape becomes increasingly complex. A complex energy landscape prevents mixing, resulting different performances between the algorithms, due to their different mixing capabilities. As PT was developed to improve mixing over the PCD-1 method, and INS has been shown to be superior to PT in several models, the results of the empirical study are in line with expectations. The PCD-1 method even exhibits a degenerate behavior due to its poor mixing, as was also previously observed in [3].

For MNIST and the larger toy data set the modes of the distribution are further apart, which should prevent mixing to a stronger degree than for the other data sets. Therefore, it is a first surprising that the different algorithms perform more similar here than for the toy data sets in Section 4.4. One possible explanation is provided by [27]: even though the energy landscape is complex, the collection of local minima that one is likely to end up in tends to promote good performance. However this line of reasoning does not take into account *how* SGD moves around in the energy landscape but instead looks at a static picture and "counts" the number of critical points of different indices. Recent works suggest that this is too simplistic a view and that dynamics should be considered as well, see for example [1].

Another potential explanation for the observations for MNIST and the larger toy data set is that the performance measure, the prediction accuracy defined in Section 4.1, does not reflect the algorithms ability to mix at a fine enough level. This is combined with the fact that the classification task might be too simple, rendering the prediction accuracy incapable to distinguish between the different algorithms for training; see the remarks in Sections 4.4-4.5. Indeed, the empirical study in [3]

suggest significant improvements of PT over PCD-1 also for MNIST when likelihood is used to measure performance, whereas this is not observed when considering the prediction accuracy for the Boltzmann classifiers. Still, also for the toy data set in Section 4.5 and using the prediction accuracy, although the PCD-1 method seems of best quality early in training, INS again has slightly better classification capability later in training compared to the other methods.

Future work includes extending INS to variational auto-encoders, with an aim similar to [2], together with more extensive empirical studies, including both other data sets and comparing different performance measures (prediction accuracy, likelihood). These studies will also consider the impact of different hyperparameters in PT and INS (number of temperatures, choice of temperatures, swap rate for PT etc.), and performance when equal computational time is allotted to the different algorithms.

6 Appendix

6.1 The marginalisation trick

Let $\hat{p}(\mathbf{v},\mathbf{h})$ denote the unnormalised joint probability function for $(\mathbf{v},\mathbf{h})$ and $\hat{p}(\mathbf{v})$ denote the unnormalised probability function for $\mathbf{v}$ (suppressing parameter dependence):

$$\hat{p}(\mathbf{v},\mathbf{h}) = \exp\{-E(\mathbf{v},\mathbf{h})\}, \;\; \hat{p}(\mathbf{v}) = \sum_{\mathbf{h}} \hat{p}(\mathbf{v},\mathbf{h}).$$

For the unnormalised joint distribution, by Baye's rule it holds that, for any $\mathbf{h}$,

$$\hat{p}(\mathbf{v}) = \frac{\hat{p}(\mathbf{v},\mathbf{h})}{\hat{p}(\mathbf{v},\mathbf{h})/\hat{p}(\mathbf{v})} = \frac{\hat{p}(\mathbf{v},\mathbf{h})}{p(\mathbf{h}|\mathbf{v})}.$$

The left-hand side is independent of $\mathbf{h}$ and can thus be computed by choosing $\mathbf{h}$ arbitrarily, and inserting it in the computable operation on the right hand side. Taking $\mathbf{h} \equiv 1$ yields

$$\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} = \frac{e^{-E(\mathbf{v},\mathbf{1})}}{p(\mathbf{1}|\mathbf{v})} = \frac{e^{-E(\mathbf{v},\mathbf{1})}}{\prod_{m=1}^{M} sigm[(\mathbf{v}\mathbf{W}+\mathbf{c})^{(m)}]}.$$

In practice, $\mathbf{h}$ must be chosen with care in order to avoid numerical division by zero. For the numerical experiments in this paper

$$\mathbf{h} = \max_{\mathbf{h}} \; p(\mathbf{h}|\mathbf{v}) = (round(p(\mathbf{h}^{(1)} = 1|\mathbf{v})), ..., round(p(\mathbf{h}^{(M)} = 1|\mathbf{v}))),$$

is used where *round* denotes the rounding operator.

6.2 Propositions

Proposition 1. *The INS-Gibbs Markov kernel has the symmetrised distribution $\bar{p}$ as invariant distribution.*

Proof. Let $G_{\kappa,\sigma_j}(\mathbf{X}|\mathbf{X}')$ be the probability distribution for $\mathbf{X}$ after κ Gibbs steps when starting in $\mathbf{X}'$ and temperatures are assigned according to σ_j. Given values $\mathbf{X}'$, the following probability distribution holds for sample values $\mathbf{X}$ obtained after one full INS-Gibbs step:

$$\sum_{j=1}^{K!} \rho_{\sigma_j}(\mathbf{X}')G_{\kappa,\sigma_j}(\mathbf{X}|\mathbf{X}').$$

Integration w.r.t the symmetrised distribution yields

$$\begin{aligned}
&\sum_{\mathbf{X}'}\sum_{j=1}^{K!} \rho_{\sigma_j}(\mathbf{X}')G_{\kappa,\sigma_j}(\mathbf{X}|\mathbf{X}')\bar{p}(\mathbf{X}') \\
&\quad = \frac{1}{K!}\sum_{j=1}^{K!}\sum_{\mathbf{X}'} p_{\sigma_j}(\mathbf{X}')G_{\kappa,\sigma_j}(\mathbf{X}|\mathbf{X}') \\
&\quad = \frac{1}{K!}\sum_{j=1}^{K!} p_{\sigma_j}(\mathbf{X}) = \bar{p}(\mathbf{X}).
\end{aligned}$$

In the first step the definition of ρ_{σ_j} is used and in the second last step the fact that the Gibbs kernel G_{κ,σ_j} has the joint distribution p_{σ_j} as its invariant distribution. □

Proposition 2. *Let $\mathrm{E}_{\bar{p}}$ and E_{p_1} denote expectation with respect to $\bar{p}$ and p_1, respectively. Then,*

$$\mathrm{E}_{\bar{p}}\Big[\sum_{k=1}^{K} f(\mathbf{x_k})\Big(\sum_{\sigma;\sigma^k=1}\rho_\sigma(\mathbf{X})\Big)\Big] = \mathrm{E}_{p_1}[f(\mathbf{x_1})].$$

Proof.

For any $k = 1,...,K$, it holds that

$$
\begin{aligned}
\mathrm{E}_{\bar{p}}\Big[f(\mathbf{x_k})\Big(\sum_{\sigma;\sigma^k=1}\rho_\sigma(\mathbf{X})\Big)\Big] &= \mathrm{E}_{\bar{p}}\Big[f(\mathbf{x_k})\Big(\sum_{\sigma;\sigma^k=1}\rho_\sigma(\mathbf{X})\Big)\Big] \\
&= \sum_{\mathbf{X}} f(\mathbf{x_k})\Big(\sum_{\sigma;\sigma^k=1}\rho_\sigma(\mathbf{X})\Big)\bar{p}(\mathbf{X}) \\
&= \sum_{\mathbf{X}} f(\mathbf{x_k})\Big(\sum_{\sigma;\sigma^k=1}\frac{p_\sigma(\mathbf{X})}{K!}\Big) \\
&= \frac{1}{K!}\sum_{\mathbf{X}} f(\mathbf{x_k})\Big(\sum_{\sigma;\sigma^k=1}\prod_{i=1}^{K} p_{\sigma^i}(\mathbf{x}_i)\Big) \\
&= \frac{1}{K!}\sum_{\mathbf{X}} f(\mathbf{x}_k)\Big(\sum_{\sigma;\sigma^k=1} p_{\sigma^k}(\mathbf{x}_k)\prod_{i\neq k}^{K} p_{\sigma^i}(\mathbf{x}_i)\Big) \\
&= \frac{1}{K!}\sum_{\mathbf{x}_k} f(\mathbf{x}_k)p_1(\mathbf{x}_k)\sum_{\mathbf{x}_i, i\neq k}\Big(\sum_{\sigma;\sigma^k=1}\prod_{i\neq k}^{K} p_{\sigma^i}(\mathbf{x}_i)\Big) \\
&= \frac{1}{K!}\sum_{\mathbf{x}_k} f(\mathbf{x}_k)p_1(\mathbf{x}_k)(K-1)! \\
&= \frac{1}{K}\mathrm{E}_{p_1}[f(\mathbf{x}_1)].
\end{aligned}
$$

Summing over $k = 1, ..., K$ provs the claim. □

References

1. S. Arora, N. Cohen, N. Golowich and W. Hu. A Convergence Analysis of Gradient Descent for Deep Linear Neural Networks. *Preprint*, arXiv:1810.0228, 2018.
2. A. L. Caterini, A. Doucet and D. Sejdinovic. Hamiltonian Variational Auto-Encoder. *32nd Conference on Neural Information Processing Systems (NeurIPS)*, 2018.
3. G. Desjardins, A. Courville, Y. Bengio, P. Vincent O. and Dellaleau. Parallel tempering for training restricted Boltzmann machines. In *JMLR Workshop and Conference Proceedings: AISTATS 2010*, volume 9, pp. 145–152. 2010.
4. J. D. Doll and P. Dupuis. On performance measures for infinite swapping Monte Carlo methods. *Journal of Chemical Physics*, 143, 2015.
5. J. D. Doll, P. Dupuis and P. Nyquist. A large deviations analysis of certain qualitative properties of parallel tempering and infinite swapping algorithms. *Applied Mathematics and Optimization*, 78(1): 103–144, 2018.
6. J. D. Doll, P. Dupuis, and P. Nyquist. Thermodynamic integration methods, infinite swapping and the calculation of generalized averages. *Journal of Chemical Physics*, 146, 134111, 2017).
7. J. D. Doll, N. Plattner, D. L. Freeman, Y. Liu and P. Dupuis. Rare-event sampling: occupation-based performance measures for parallel tempering and infinite swapping Monte Carlo methods. *Journal of Chemical Physics*, 137, 2012.
8. P. Dupuis, Y. Liu, N. Plattner and J. D. Doll. On the infinite swapping limit for parallel tempering. *Multiscale Modeling and Simulation*, 10(3): 986–1022, 2012.
9. D. J. Earl and M. W. Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7:3910–3916, 2005.
10. A. Fischer and C. Igel. An introduction to restricted Boltzmann machines. In Alvarez, L., Mejail, M., Gomez, L., and Jacobo, J. (eds.), *Progress in Pattern Recognition, Image Analysis,*

Computer Vision, and Applications, volume 7441 of *Lecture Notes in Computer Science*. Springer, 2012.
11. Y. Freund and D. Haussler. Unsupervised learning of distributions on binary vectors using two layer networks. Technical report, University of California, Santa Cruz, 1994.
12. C. J. Geyer. Markov chain Monte Carlo maximum likelihood. In *Interface Foundation of North America*. Retrieved from the University of Minnesota Digital Conservancy, 1991. URL http://hdl.handle.net/11299/58440.
13. C. J. Geyer and E. A. Thompson. Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 90(431):909–920, 1995.
14. M. Hardt, B. Recht and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, volume 48, 2016.
15. G. E Hinton. A practical guide to training restricted restricted Boltzmann machines. , version 1, pp. 10, 2010.
16. G. E Hinton. Products of experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN)*, volume 1, pp. 1–6. 1999.
17. G. E Hinton. Training products of experts by minizing constrastive divergence. *Neural Computation*, 14:1771–1800, 2002.
18. C. S. Jensen, A. Kong, U. Kjaerulff. Blocking-Gibbs sampling in very large probabilistic expert systems *International Journal of Human-Computer Studies*, 647–666, 1995.
19. D. A. Kofke. On the acceptance probability of replica-exchange Monte Carlo trials. *Journal of Chemical Physics*, 117(15):6911–6914, 2002.
20. J. S. Liu. *Monte Carlo strategies in scientific computing*. Springer Series in Statistics. Springer, New York, 2008.
21. J. Lu and E. Vanden-Eijnden. Infinite swapping replica exchange molecular dynamics leads to a simple simulation patch using mixture potentials. *Journal of Chemical Physics*, 138, 2013.
22. E. Marinari, G. Parisi and J. J. Ruiz-Lorenzo. Numerical simulations of spin glass systems. *Spin Glasses and Random Fields*, 12, 1997.
23. G. Menz, A. Schlichting and W. Tang. Ergodicity of the infinite swapping algorithm at low temperature. *Preprint*, arXiv:1811.10174, 2018.
24. N. Plattner, J. D. Doll, P. Dupuis, H. Wang, Y. Liu and J. E. Gubernatis. An infinite swapping approach to the rare-event sampling problem. *Journal of Chemical Physics*, 135, 2011.
25. N. Plattner, J. D. Doll and M. Meuwly. Overcoming the rare event sampling problem in biological systems with infinite swapping. *Journal of Chemical Theory and Computation*, 9(9): 4215–4224, 2013.
26. F. Rao and A. Caflisch. Replica exchange molecular dynamics simulations of reversible folding. *The Journal of Chemical Physics*, 119: 4035, 2003.
27. Sagun, L., Ugur Guney, V, and Lecun, Y. Explorations on high dimensional landscapes. ICLR. 2015.
28. Salakhutdinov, R., Mnih, A, and Hinton, G.E. Restricted Boltzmann machines for collaborative filtering. ICML 2007.
29. P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In Rumelhart, D. E. and McClelland, J. L. (eds.), *Parallel Distributed Processing*, volume 1, chapter 6, pp. 194–281. Cambridge: MIT Press, 1986.
30. Y. Sugita and Y. Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chemical Physics Letters*, 314:141–151, 1999.
31. R. H. Swendsen and J. S. Wang. Replica Monte Carlo simulation of spin glasses. *Physical Review Letters*, 57:2607–2609, 1986.
32. T. Tieleman Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 1064–1071. Helsinki, Finland, 2008.
33. M. Welling, M. Rosen-Zvi and G. E. Hinton. Exponential family harmoniums with an application to information retrieval. In *NIPS 17*, volume 17. MIT Press, 2005.
34. T. Q. Yu, J. Lu, C. F. Abrams and E. Vanden-Eijnden. A multiscale implementation of infinite-swap replica exchange molecular dynamics. *Proceedings of the National Academy of Sciences USA*, 113(42):11744–11749, 2016.