Towards Maximum Likelihood: Learning Undirected Graphical Models using Persistent Sequential Monte Carlo

Hanchen Xiong HANCHEN.XIONG@UIBK.AC.AT
Sandor Szedmak SANDOR.SZEDMAK@UIBK.AC.AT
Justus Piater JUSTUS.PIATER@UIBK.AC.AT
Institute of Computer Science, University of Innsbruck
Technikerstr. 21a, A-6020 Innsbruck, Austria

Abstract

Along with the emergence of algorithms such as persistent contrastive divergence (PCD), tempered transition and parallel tempering, the past decade has witnessed a revival of learning undirected graphical models (UGMs) with sampling-based approximations. In this paper, based upon the analogy between Robbins-Monro's stochastic approximation procedure and sequential Monte Carlo (SMC), we analyze the strengths and limitations of state-of-the-art learning algorithms from an SMC point of view. Moreover, we apply the rationale further in sampling at each iteration, and propose to learn UGMs using *persistent sequential Monte Carlo* (PSMC). The whole learning procedure is based on the samples from a long, persistent sequence of distributions which are actively constructed. Compared to the above-mentioned algorithms, one critical strength of PSMC-based learning is that it can explore the sampling space more effectively. In particular, it is robust when learning rates are large or model distributions are high-dimensional and thus multi-modal, which often causes other algorithms to deteriorate. We tested PSMC learning, also with other related methods, on carefully-designed experiments with both synthetic and real-world data, and our empirical results demonstrate that PSMC compares favorably with the state of the art.

Keywords: Sequential Monte Carlo, maximum likelihood learning, undirected graphical models.

1. Introduction

Learning undirected graphical models (UGMs), or Markov random fields (MRF), has been an important yet challenging machine learning task. On the one hand, thanks to its flexible and powerful capability in modeling complicated dependencies, UGMs are prevalently used in many domains such as computer vision, natural language processing and social analysis. Undoubtedly, it is of great significance to enable UGMs' parameters to be automatically adjusted to fit empiric data, *e.g.* maximum likelihood (ML) learning. A fortunate property of the likelihood function is that it is concave with respect to its parameters (Koller and Friedman, 2009), and therefore gradient ascent can be applied to find the unique maximum. On the other hand, learning UGMs via ML in general remains intractable due to the presence of the partition function. Monte Carlo estimation is a principal solution to the problem. For example, one can employ Markov chain Monte Carlo (MCMC) to obtain samples from the model distribution, and approximate the partition function with the samples. However, the sampling procedure of MCMC is very inefficient because it usually requires a large number of steps for the Markov chain to reach equilibrium. Even though in some cases where efficiency can be ignored, another weakness of MCMC estimation is that it yields large estimation variances. A more practically feasible alternative is MCMC maximum likelihood (MCMCML;

Geyer 1991); see section 2.1. MCMCML approximates the gradient of the partition function with importance sampling, in which a proposal distribution is initialized to generate a fixed set of MCMC samples. Although MCMCML increases efficiency by avoiding MCMC sampling at every iteration, it also suffers from high variances (with different initial proposal distributions). Hinton (2002) studied *contrastive divergence* (CD) to replace the objective function of ML learning. This turned out to be an efficient approximation of the likelihood gradient by running only a few steps of Gibbs sampling, which greatly reduces variance as well as the computational burden. However, it was pointed out that CD is a biased estimation of ML (Carreira-Perpinan and Hinton, 2005), which prevents it from being widely employed (Tieleman, 2008; Tieleman and Hinton, 2009; Desjardins et al., 2010). Later, a *persistent* version of CD (PCD) was put forward as a closer approximation of the likelihood gradient (Tieleman, 2008). Instead of running a few steps of Gibbs sampling from training data in CD, PCD maintains an almost persistent Markov chain throughout iterations by preserving samples from the previous iteration, and using them as the initializations of Gibbs samplers in the current iteration. When the learning rate is sufficiently small, samples can be roughly considered as being generated from the stationary state of the Markov chain. However, one critical drawback in PCD is that Gibbs sampling will generate highly correlated samples between consecutive weight updates, so mixing will be poor before the model distribution gets updated at each iteration. The limitations of PCD sparked many recent studies of more sophisticated sampling strategies for effective exploration within data space (section 3). For instance, Salakhutdinov (2010) studied *tempered transition* (Neal, 1994) for learning UGMs. The strength of tempered transition is that it can make potentially big transitions by going through a trajectory of intermediary Gibbs samplers which are smoothed with different temperatures. At the same time, *parallel tempering*, which can be considered a parallel version of tempered transition, was developed by Desjardins et al. (2010) for training restricted Boltzmann machines (RBMs). Contrary to a single Markov chain in PCD and tempered transition, parallel tempering maintains a pool of Markov chains governed by different temperatures. Multiple tempered chains progress in parallel and are mixed at each iteration by randomly swapping the states of neighbouring chains.

The contributions of this paper are twofold. The first is theoretic. By linking Robbins-Monro's stochastic approximation procedure (SAP; Robbins and Monro 1951) and sequential Monte Carlo (SMC), we cast PCD and other state-of-the-art learning algorithms into a SMC-based interpretation framework. Moreover, within the SMC-based interpretation, two key factors which affect the performance of learning algorithms are disclosed: *learning rate* and *model complexity* (section 4). Based on this rationale, the strengths and limitations of different learning algorithms can be analyzed and understood in a new light. The second contribution is practical. Inspired by the understanding of learning UGMs from a SMC perspective, and the successes of global tempering used in parallel tempering and tempered transition, we put forward a novel approximation-based algorithm, *persistent SMC* (PSMC), to approach the ML solution in learning UGMs. The basic idea is to construct a long, persistent distribution sequence by inserting many tempered intermediary distributions between two successively updated distributions (section 5). According to our empirical results on learning two discrete UGMs (section 6), the proposed PSMC outperforms other learning algorithms in challenging circumstances, *i.e.* large learning rates or large-scale models.

2. Learning Undirected Graphical Models

In general, we can define undirected graphical models (UGMs) in an energy-based form:

$$p(\mathbf{x};\boldsymbol{\theta}) = \frac{\exp\left(-E(\mathbf{x};\boldsymbol{\theta})\right)}{\mathbf{Z}(\boldsymbol{\theta})} \tag{1}$$

$$\textit{Energy function:}\quad E(\mathbf{x};\boldsymbol{\theta}) = -\boldsymbol{\theta}^{\top}\phi(\mathbf{x}) \tag{2}$$

with random variables $\mathbf{x} = [x_1, x_2, \ldots, x_D] \in \mathcal{X}^D$ where x_d can take N_d discrete values, $\phi(\mathbf{x})$ is a K-dimensional vector of sufficient statistics, and parameter $\boldsymbol{\theta} \in \mathbb{R}^K$. $\mathbf{Z}(\boldsymbol{\theta}) = \sum_{\mathbf{x}} \exp(\boldsymbol{\theta}^{\top}\phi(\mathbf{x}))$ is the partition function for global normalization. Learning UGMs is usually done via maximum likelihood (ML). A critical observation of UGMs' likelihood functions is that they are concave with respect to $\boldsymbol{\theta}$, therefore any local maximum is also global maximum (Koller and Friedman, 2009), and gradient ascent can be employed to find the optimal $\boldsymbol{\theta}^*$. Given training data $\mathcal{D} = \{\mathbf{x}^{(m)}\}_{m=1}^{M}$, we can compute the derivative of average log-likelihood $\mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = \frac{1}{M}\sum_{m=1}^{M} \log p(\mathbf{x}^{(m)};\boldsymbol{\theta})$ as

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta}|\mathcal{D})}{\partial \boldsymbol{\theta}} = \underbrace{\mathbb{E}_{\mathcal{D}}(\phi(\mathbf{x}))}_{\psi^+} - \underbrace{\mathbb{E}_{\boldsymbol{\theta}}(\phi(\mathbf{x}))}_{\psi^-}, \tag{3}$$

where $\mathbb{E}_{\mathcal{D}}(\xi)$ is the expectation of ξ under the empirical data distribution $p_{\mathcal{D}} = \frac{1}{M}\sum_{m=1}^{M} \delta(\mathbf{x}^{(m)})$, while $\mathbb{E}_{\boldsymbol{\theta}}(\xi)$ is the expectation of ξ under the model probability with parameter $\boldsymbol{\theta}$. The first term in (3), which is often referred to as *positive phase* ψ^+, can be easily computed as the average of $\phi(\mathbf{x}^{(m)}), \mathbf{x}^{(m)} \in \mathcal{D}$. The second term in (3), also known as *negative phase* ψ^-, however, is not trivial because it is a sum of $\prod_{d=1}^{D} N_d$ terms, which is only computationally feasible for UGMs of very small size. Markov chain Monte Carlo (MCMC) can be employed to approximate ψ^-, although it is usually expensive and leads to large estimation variances. The underlying procedure of ML learning with gradient ascent, according to (3), can be envisioned as a behavior that iteratively pulls down the energy of the data space occupied by $\mathcal{D}$ (positive phase), but raises the energy over all data space $\mathcal{X}^D$ (negative phase), until it reaches a balance ($\psi^+ = \psi^-$).

2.1. Markov Chain Monte Carlo Maximum Likelihood

A practically feasible approximation of (3) is Markov chain Monte Carlo maximum likelihood (MCMCML; Geyer 1991). In MCMCML, a proposal distribution $p(\mathbf{x};\boldsymbol{\theta}_{\mathbf{0}})$ is set up in the same form as (1) and (2), and we have

$$\frac{\mathbf{Z}(\boldsymbol{\theta})}{\mathbf{Z}(\boldsymbol{\theta}_0)} = \frac{\sum_{\mathbf{x}} \exp(\boldsymbol{\theta}^{\top}\phi(\mathbf{x}))}{\sum_{\mathbf{x}} \exp(\boldsymbol{\theta}_0^{\top}\phi(\mathbf{x}))} \tag{4}$$

$$= \frac{\sum_{\mathbf{x}} \exp(\boldsymbol{\theta}^{\top}\phi(\mathbf{x}))}{\exp(\boldsymbol{\theta}_0^{\top}\phi(\mathbf{x}))} \times \frac{\exp(\boldsymbol{\theta}_0^{\top}\phi(\mathbf{x}))}{\sum_{\mathbf{x}} \exp(\boldsymbol{\theta}_0^{\top}\phi(\mathbf{x}))} \tag{5}$$

$$= \sum_{\mathbf{x}} \exp\Big((\boldsymbol{\theta} - \boldsymbol{\theta}_0)^{\top}\phi(\mathbf{x})\Big)\, p(\mathbf{x};\boldsymbol{\theta}_0) \tag{6}$$

$$\approx \frac{1}{S}\sum_{s=1}^{S} w^{(s)} \tag{7}$$

Algorithm 1 MCMCML Learning Algorithm

```
Input:  training data $\mathcal{D} = \{\mathbf{x}^{(m)}\}_{m=1}^{M}$; learning rate $\eta$; gap $L$ between two successive proposal
        distribution resets
 1: $t \leftarrow 0$, initialize the proposal distribution $p(\mathbf{x}; \boldsymbol{\theta}_0)$
 2: while ! stop criterion do
 3:   if $(t \mod L) == 0$ then
 4:     (Re)set the proposal distribution as $p(\mathbf{x}; \boldsymbol{\theta}_t)$
 5:     Sample $\{\bar{\mathbf{x}}^{(s)}\}$ from $p(\mathbf{x}; \boldsymbol{\theta}_t)$
 6:   end if
 7:   Calculate $w^{(s)}$ using (8)
 8:   Calculate gradient $\frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta}|\mathcal{D})}{\partial \boldsymbol{\theta}}$ using (9)
 9:   update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta \frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta}|\mathcal{D})}{\partial \boldsymbol{\theta}}$
10:   $t \leftarrow t + 1$
11: end while
Output:  estimated parameters $\boldsymbol{\theta}^* = \boldsymbol{\theta}_t$
```

where $w^{(s)}$ is

$$w^{(s)} = \exp\left((\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \phi(\bar{\mathbf{x}}^{(s)})\right), \tag{8}$$

and the $\bar{\mathbf{x}}^{(s)}$ are sampled from the proposal distribution $p(\mathbf{x}; \boldsymbol{\theta}_0)$. By substituting $\mathbf{Z}(\boldsymbol{\theta}) = \mathbf{Z}(\boldsymbol{\theta}_0)\frac{1}{S}\sum_{s=1}^{S} w^{(s)}$ into (1) and average log-likelihood, we can compute corresponding gradient as (note $\mathbf{Z}(\boldsymbol{\theta}_0)$ will be eliminated since it corresponds to a constant in the logarithm)

$$\frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta}|\mathcal{D})}{\partial \boldsymbol{\theta}} = \mathbb{E}_{\mathcal{D}}(\phi(\mathbf{x})) - \mathbb{E}_{\boldsymbol{\theta}_0}(\phi(\mathbf{x})), \tag{9}$$

where $\mathbb{E}_{\boldsymbol{\theta}_0}(\xi)$ is the expectation of ξ under a *weighted* empirical data distribution $p_{\boldsymbol{\theta}_0} = \sum_{s=1}^{S} w^{(s)}\delta(\bar{\mathbf{x}}^{(s)})/\sum_{s=1}^{S} w^{(s)}$ with data sampled from $p(\mathbf{x}; \boldsymbol{\theta}_0)$. From (9), it can be seen that MCMCML does nothing more than an importance sampling estimation of ψ^- in (3). MCMCML has the nice asymptotic convergence property (Salakhutdinov, 2010) that it will converge to the exact ML solution when the number of samples S goes to infinity. However, as an inherent weakness of importance sampling, the performance of MCMCML in practice highly depends on the choice of the proposal distribution, which results in large estimation variances. The phenomenon gets worse when it scales up to high-dimensional models. One engineering trick to alleviate this pain is to reset the proposal distribution, after a certain number of iterations, to the recently updated estimation $p(\mathbf{x}; \boldsymbol{\theta}^{estim})$ (Handcock et al., 2007). Pseudocode of the MCMCML learning algorithm is presented in Algorithm 1.

3. State-of-the-art Learning Algorithms

Contrastive Divergence (CD) is an alternative objective function of likelihood (Hinton, 2002), and turned out to be de facto a cheap and low-variance approximation of the maximum likelihood (ML) solution. CD tries to minimize the discrepancy between two Kullback-Leibler (KL) divergences, $KL(p^0|p^\infty_{\boldsymbol{\theta}})$ and $KL(p^n_{\boldsymbol{\theta}}|p^\infty_{\boldsymbol{\theta}})$, where $p^0 = p(\mathcal{D}; \boldsymbol{\theta})$, $p^n_{\boldsymbol{\theta}} = p(\bar{\mathcal{D}}_n; \boldsymbol{\theta})$ with $\bar{\mathcal{D}}_n$ denoting the data sampled after n steps of Gibbs sampling with parameter $\boldsymbol{\theta}$, and $p^\infty_{\boldsymbol{\theta}} = p(\bar{\mathcal{D}}_\infty; \boldsymbol{\theta})$ with $\bar{\mathcal{D}}_\infty$ denoting the data sampled from the equilibrium of a Markov chain. Usually $n = 1$ is used, and correspondingly it is referred to as the CD-1 algorithm. The negative gradient of CD-1 is

$$-\frac{\partial\big(CD_1(\mathcal{D};\boldsymbol{\theta})\big)}{\partial\boldsymbol{\theta}} = \mathbb{E}_{\mathcal{D}}(\phi(\mathbf{x})) - \mathbb{E}_{\bar{\mathcal{D}}_1}(\phi(\mathbf{x})) \tag{10}$$

where $\mathbb{E}_{\bar{\mathcal{D}}_1}(\xi)$ is the expectation of ξ under the distribution $p^1_{\boldsymbol{\theta}}$. The key advantage of CD-1 is that it efficiently approximates ψ^- in the likelihood gradient (3) by running only one step Gibbs sampling. While this local exploration of sampling space can avoid large variances, CD-1 was theoretically (Carreira-Perpinan and Hinton, 2005) and empirically (Tieleman, 2008; Tieleman and Hinton, 2009; Desjardins et al., 2010) proved to be a biased estimation of ML .

Persistent Contrastive Divergence (PCD) is an extension of CD by running a nearly persistent Markov chain. For approximating ψ^- in likelihood gradient (3), the samples at each iteration are retained as the initialization of Gibbs sampling in the next iteration. The mechanism of PCD was usually interpreted as a case of Robbins-Monro's stochastic approximation procedure (SAP; Robbins and Monro 1951) with Gibbs sampling as transitions. In general SAP, if the learning rate η is sufficiently small compared to the mixing rate of the Markov chain, the chain can be roughly considered as staying close to the equilibrium distribution (i.e. PCD→ML when $\eta \to 0$). Nevertheless, Gibbs sampling as used in PCD heavily hinders the exploration of data space by generating highly correlated samples along successive model updates. This hindrance becomes more severe when the model distribution is highly multi-modal. Although multiple chains (mini-batch learning) used in PCD can mitigate the problem, we cannot generally expect the number of chains to exceed the number of modes. Therefore, at the late stage of learning, PCD usually gets stuck in a local optimum, and in practice, small and linearly-decayed learning rates can improve the performance (Tieleman, 2008).

Tempered Transition was originally developed by Neal (1994) to generate relatively big jumps in Markov chains while keeping reasonably high acceptance rates. Instead of standard Gibbs sampling used in PCD, tempered transition constructs a sequence of Gibbs samplers based on the model distribution specified with different temperatures:

$$p_h(\mathbf{x};\boldsymbol{\theta}) = \frac{\exp(-E(\mathbf{x};\boldsymbol{\theta})\beta_h)}{\mathbf{Z}(h)} \tag{11}$$

where h indexes temperatures $h \in [0, H]$ and β_H are inverse temperatures $0 \le \beta_H < \beta_{H-1} < \cdots \beta_0 = 1$. In particular, β_0 corresponds to the original complex distribution. When h increases, the distribution gets more flat, where Gibbs samplers can more adequately explore. In tempered transition, a sample is generated with a Gibbs sampler starting from the original distribution. It then goes through a trajectory of Gibbs sampling through sequentially tempered distributions (11). A backward trajectory is then run until the sample reaches the original distribution. The acceptance of the final sample is determined by the probability of the whole forward-and-backward trajectory. If the trajectory is rejected, the sample does not move at all, which is even worse than local movements of Gibbs sampling, so β_H is set relatively high (0.9 in Salakhutdinov 2010) to ensure high acceptance rates.

Parallel Tempering, on the other hand, is a "parallel" version of Tempered Transition, in which smoothed distributions (11) are run with one step of Gibbs sampling in parallel at each iteration. Thus, samples native to more uniform chains will move with larger transitions, while samples native to the original distribution still move locally. All chains are mixed by swapping samples of randomly selected neighbouring chains. The probability of the swap is

$$r = \exp\left((\beta_h - \beta_{h+1})(E(\mathbf{x}_h) - E(\mathbf{x}_{h+1}))\right) \tag{12}$$

Although multiple Markov chains are maintained, only samples at the original distribution are used. In the worst case (there is no swap between β_0 and β_1), parallel tempering degrades to PCD-1. β_H can be set arbitrarily low (0 was used by Desjardins et al. 2010).

4. Learning as Sequential Monte Carlo

Before we delve into the analysis of the underlying mechanism in different learning algorithms, it is better to find a unified interpretation framework, within which the behaviors of all algorithms can be more apparently viewed and compared in a consistent way. In most previous work, PCD, tempered transition and parallel tempering were studied as special cases of Robbins-Monro's stochastic approximation procedure (SAP; Tieleman and Hinton 2009; Desjardins et al. 2010; Salakhutdinov 2010). These studies focus on the interactions between the mixing of Markov chains and distribution updates. However, we found that, since the model changes at each iteration, the Markov chain is actually not subject to an invariant distribution, the concept of the mixing of Markov chains is fairly subtle and difficult to capture based on SAP.

Alternatively, Asuncion et al. (2010) exposed that PCD can be interpreted as a sequential Monte Carlo procedure by extending MCMCML to a particle filtered version. To have an quick overview of sequential Monte Carlo More and how it is related to learning UGMs, we first go back to Markov chain Monte Carlo maximum likelihood (MCMCML; section 2.1) and examine it in an extreme case. When the proposal distribution in MCMCML is reset at every iteration as the previously updated estimation, *i.e.* $L = 1$ in Algorithm 1 and the proposal distribution is left as $p(\mathbf{x}; \boldsymbol{\theta}_{t-1})$ at the tth iteration, the weights will be computed as $w^{(s)} = \exp(\boldsymbol{\theta_t} - \boldsymbol{\theta_{t-1}})^\top \phi(\bar{\mathbf{x}}^{(s)})$. Since the parameters $\boldsymbol{\theta}$ do not change very much along iterations, it is not necessary to generate particles[1] from proposal distributions at each iteration. Instead, a set of particles are initially generated and reweighted sequentially for approximating the negative phase. However, if the gap between two successive $\boldsymbol{\theta}$ is relatively large, particles will degenerate. Usually, the effective sampling size (ESS) can be computed to measure the degeneracy of particles, so if ESS is smaller than a pre-defined threshold, resampling and MCMC transition are necessary to recover from it. The description above notably leads to *particle filtered MCMCML* (Asuncion et al., 2010), which greatly outperforms MCMCML with small amount of extra computation.

More interestingly, it was pointed out that PCD also fits the above sequential Monte Carlo procedure (*i.e.* importance reweighting + resampling + MCMC transition) with uniform weighting for all particles and Gibbs sampling as MCMC transition. Here we extend this analogy further to general Robbins-Monro's SAP, into which tempered transition and parallel tempering are also categorized, and write out a uniform interpretation framework of all learning algorithms from SMC perspective (see Algorithm 2). Note that all particle weights are uniformly assigned; resampling has no effect and can be ignored. In addition, the MCMC transition step is forced to take place at every iteration, believing that the particle set is always degenerated.

It is also worth noting that when we are applying algorithms in Algorithm 2, we are not interested in particles from any individual target distribution (which is usually the purpose of SMC).

1. From now on, we use "particles" to fit SMC terminology, it is equivalent to "samples" unless mentioned otherwise.

Algorithm 2 Interpreting Learning as SMC

```
Input:   training data $\mathcal{D} = \{\mathbf{x}^{(m)}\}_{m=1}^{M}$; learning rate $\eta$
1: Initialize $p(\mathbf{x}; \boldsymbol{\theta}_0), t \leftarrow 0$
2: Sample particles $\{\bar{\mathbf{x}}_0^{(s)}\}_{s=1}^{S} \sim p(\mathbf{x}; \boldsymbol{\theta}_0)$
3: while ! stop criterion do
4:   // importance reweighting
     Assign $w^{(s)} \leftarrow \frac{1}{S}, \forall s \in S$
5:   // resampling is ignored because it has no effect
6:   // MCMC transition
7:   switch (algorithmic choice)
8:   case CD:
9:     generate a brand new particle set $\{\bar{\mathbf{x}}_{t+1}^{(s)}\}_{s=1}^{S}$ with one step Gibbs sampling from $\mathcal{D}$
10:  case PCD:
11:    evolve particle set $\{\bar{\mathbf{x}}_t^{(s)}\}_{s=1}^{S}$ to $\{\bar{\mathbf{x}}_{t+1}^{(s)}\}_{s=1}^{S}$ with one step Gibbs sampling
12:  case Tempered Transition:
13:    evolve particle set $\{\bar{\mathbf{x}}_t^{(s)}\}_{s=1}^{S}$ to $\{\bar{\mathbf{x}}_{t+1}^{(s)}\}_{s=1}^{S}$ with tempered transition
14:  case Parallel Tempering:
15:    evolve particle set $\{\bar{\mathbf{x}}_t^{(s)}\}_{s=1}^{S}$ to $\{\bar{\mathbf{x}}_{t+1}^{(s)}\}_{s=1}^{S}$ with parallel tempering
16:  end switch
17:  // update distribution
     Compute the gradient $\Delta\boldsymbol{\theta}_t$ according to (3)
18:  $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta\Delta\boldsymbol{\theta}_t$
19:  $t \leftarrow t + 1$
20: end while
Output:   estimated parameters $\boldsymbol{\theta}^* = \boldsymbol{\theta}_t$
```

Instead, we want to obtain particles faithfully sampled from all sequence distributions. It can be easily imagined that one badly sampled particle set at tth iteration will lead to a biased incremental update $\Delta\boldsymbol{\theta}_t$. Consequently, the learning will go to a wrong direction even though the later sampling is perfectly good. In other words, we are considering all sequentially updated distributions $p(\mathbf{x}; \boldsymbol{\theta}_t)$ as our target distributions.

In practice, the performance of SMC highly depends on the construction of sequential distributions. In our learning case, sequential distributions are learned by iterative updates, therefore, learning and sampling are somehow entangled. As we mentioned earlier, particles will degenerate when the gap between successive sequential distributions is large. Checking ESS followed by resampling and MCMC transition can help to some extent. However, in many practical cases where real-world distributions are extremely complex, more consideration on MCMC transition is due. In our case of learning UGMs, the gap can be intuitively understood as the product of *learning rate* η and *model complexity* $\mathcal{O}(\boldsymbol{\theta})$. Therefore, we believe that learning rate and model complexity[2] are two key factors to challenge learning algorithms.

2. Here we consider the multimodality of a distribution as its complexity, *i.e.* smooth distributions are less complex than multi-modal distributions.

Within this SMC-based interpretation, we can see that four algorithms differ from each other at MCMC transitions, which is an important component in SMC (Schäfer and Chopin, 2013). In PCD, a one-step Gibbs sampler is used as MCMC transition. As for tempered transition, a Metropolis-Hasting (MH) move based on forward-and-backward sequence of Gibbs samplers of different temperatures is employed. Likewise, parallel tempering also uses a MH move. This move is generated by swapping particles native to the distributions of different temperatures. By contrast, in CD, a brand new particle set is generated by running one-step Gibbs sampling from training data, which is actually not a MCMC transition. When the learning rate is small and two successive distributions are smooth (*e.g.* at the early stage of learning or when the model is of low dimension), PCD, tempered transition and parallel tempering can traverse sampling space sufficiently well. However, when the learning rate is large or two sequential distributions exhibit multiple modes (*e.g.* at the late stage of learning or when the model is high-dimensional), highly correlated particles from the one-step Gibbs sampler's local movement cannot go through the gap between two distributions. Tempered transition and parallel tempering, instead, are more robust to the large gap since it moves closer to the later distribution by making use of many globally-tempered intermediary distributions. The worst case is CD, which always samples particles within the vicinity of training data $\mathcal{D}$. So it will eventually drop $\mathcal{D}$ down into an energy well surrounded by barriers set up by their proximities.

5. Persistent Sequential Monte Carlo

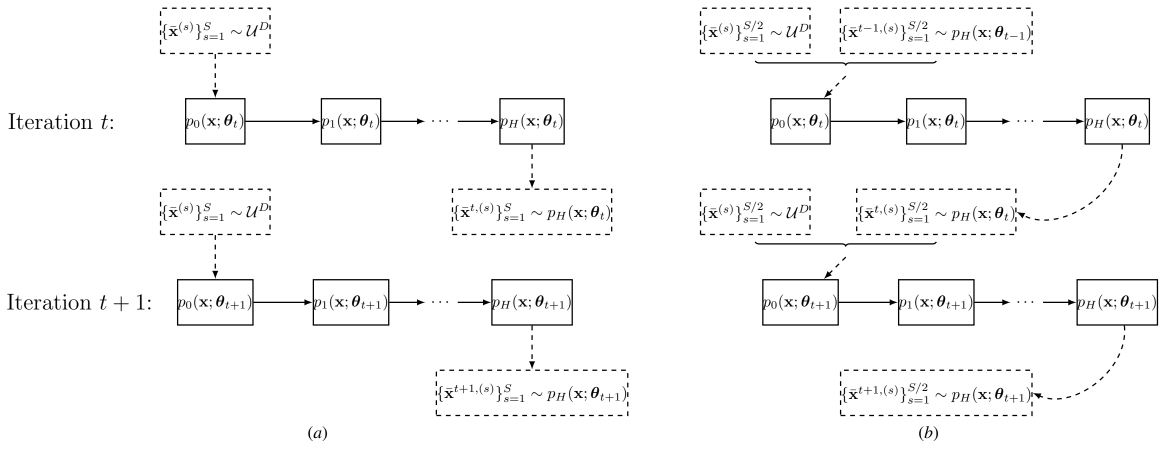
It was explained that learning UGMs can be interpreted as a SMC procedure. Here we propose to apply this rationale further in learning UGMs with a deeper construction of sequential distributions. The basic idea is very simple; given particles from $p(\mathbf{x};\boldsymbol{\theta}_t)$, many sub-sequential distributions are inserted to construct a sub-SMC for obtaining particles from $p(\mathbf{x};\boldsymbol{\theta}_{t+1})$. Inspired by global tempering used in parallel tempering and tempered transition, we build sub-sequential distributions $\{p_h(\mathbf{x};\boldsymbol{\theta}_{t+1})\}_{h=0}^{H}$ between $p(\mathbf{x};\boldsymbol{\theta}_t)$ and $p(\mathbf{x};\boldsymbol{\theta}_{t+1})$ as follows:

$$p_h(\mathbf{x};\boldsymbol{\theta}_{t+1}) \propto p(\mathbf{x};\boldsymbol{\theta}_t)^{1-\beta_h} p(\mathbf{x};\boldsymbol{\theta}_{t+1})^{\beta_h} \tag{13}$$

where $0 \leq \beta_H \leq \beta_{H-1} \leq \cdots \beta_0 = 1$. In this way, the length of the distribution sequence will be extended in SMC. In addition, obviously, $p_H(\mathbf{x};\boldsymbol{\theta}_{t+1}) = p(\mathbf{x};\boldsymbol{\theta}_t)$ while $p_0(\mathbf{x};\boldsymbol{\theta}_{t+1}) = p(\mathbf{x};\boldsymbol{\theta}_{t+1})$. Therefore, the whole learning can be considered to be based on a long, persistent sequence of distributions, and therefore the proposed algorithm is referred to as *persistent SMC* (PSMC). An alternative understanding of PSMC can be based on using standard SMC for sampling $p(\mathbf{x};\boldsymbol{\theta}_t)$ at each iteration. In standard SMC case, the sub-sequential distributions are:

$$p_h(\mathbf{x};\boldsymbol{\theta}_{t+1}) \propto p(\mathbf{x};\boldsymbol{\theta}_{t+1})^{\beta_h} \tag{14}$$

where $0 \leq \beta_H \leq \beta_{H-1} \leq \cdots \beta_0 = 1$. The schematic figures of standard SMC and PSMC are presented in Figure 1 where we can see a prominent difference between them, the continuity from $p_0(\mathbf{x};\boldsymbol{\theta}_t)$ to $p_H(\mathbf{x};\boldsymbol{\theta}_{t+1})$. Intuitively, PSMC can be seen as a linked version of SMC by connecting $p_0(\mathbf{x};\boldsymbol{\theta}_t)$ and $p_H(\mathbf{x};\boldsymbol{\theta}_{t+1})$. In addition, in our implementation of PSMC, to ensure adequate exploration, only half of the particles from $p_0(\mathbf{x};\boldsymbol{\theta}_t)$ are preserved to the next iteration; the other half particles are randomly initialized with a uniform distribution $\mathcal{U}^D$ (Figure 1(*b*)).



Iteration t:
Iteration $t+1$:
$\{\bar{\mathbf{x}}^{(s)}\}_{s=1}^{S} \sim \mathcal{U}^D$
$p_0(\mathbf{x};\boldsymbol{\theta}_t)$
$p_1(\mathbf{x};\boldsymbol{\theta}_t)$
$p_H(\mathbf{x};\boldsymbol{\theta}_t)$
$\{\bar{\mathbf{x}}^{t,(s)}\}_{s=1}^{S} \sim p_H(\mathbf{x};\boldsymbol{\theta}_t)$
$p_0(\mathbf{x};\boldsymbol{\theta}_{t+1})$
$p_1(\mathbf{x};\boldsymbol{\theta}_{t+1})$
$p_H(\mathbf{x};\boldsymbol{\theta}_{t+1})$
$\{\bar{\mathbf{x}}^{t+1,(s)}\}_{s=1}^{S} \sim p_H(\mathbf{x};\boldsymbol{\theta}_{t+1})$
(a)
$\{\bar{\mathbf{x}}^{(s)}\}_{s=1}^{S/2} \sim \mathcal{U}^D$
$\{\bar{\mathbf{x}}^{t-1,(s)}\}_{s=1}^{S/2} \sim p_H(\mathbf{x};\boldsymbol{\theta}_{t-1})$
$\{\bar{\mathbf{x}}^{t,(s)}\}_{s=1}^{S/2} \sim p_H(\mathbf{x};\boldsymbol{\theta}_t)$
$\{\bar{\mathbf{x}}^{t+1,(s)}\}_{s=1}^{S/2} \sim p_H(\mathbf{x};\boldsymbol{\theta}_{t+1})$
(b)

Figure 1: The schematic figures of (a) standard sequential Monte Carlo and (b) persistent sequential Monte Carlo for learning UGMs. Solid boxes denote sequential distributions and solid arrows represent the move (resampling and MCMC transition) between successive distributions. Dashed boxes are particle sets and dashed arrows mean feeding particles into a SMC or sampling particles out of a distribution.

One issue arising in PSMC is the number of β_h, *i.e.* H, which is also a problem in parallel tempering and tempered transition[3]. Here, we employed the bidirectional searching method (Jasra et al., 2011). When we construct sub-sequential distributions as (13), the importance weighting for each particle is

$$\begin{aligned} w^{(s)} &= \frac{p_h(\bar{\mathbf{x}}^{(s)};\boldsymbol{\theta}_{t+1})}{p_{h-1}(\bar{\mathbf{x}}^{(s)};\boldsymbol{\theta}_{t+1})} \\ &= \exp\left(E(\bar{\mathbf{x}}^{(s)};\boldsymbol{\theta}_t)\right)^{\Delta\beta_h} \exp\left(E(\bar{\mathbf{x}}^{(s)};\boldsymbol{\theta}_{t+1})\right)^{-\Delta\beta_h} \end{aligned} \tag{15}$$

where $\Delta\beta_h$ is the step length from β_{h-1} to β_h, *i.e.* $\Delta\beta_h = \beta_h - \beta_{h-1}$. We can also compute the ESS of a particle set as (Kong et al., 1994)

$$\sigma = \frac{(\sum_{s=1}^{S} w^{(s)})^2}{S\sum_{s=1}^{S} w^{(s)2}} \in \left[\frac{1}{S}, 1\right] \tag{16}$$

Based on (15) and (16), we can see that, when a particle set is given, ESS σ is actually a function of $\Delta\beta_h$. Therefore, assuming that we set the threshold of ESS as σ^*, we can then find the biggest $\Delta\beta_h$ by using bidirectional search (see Algorithm 3) . Usually a small particle set is used in learning (mini-patch scheme), so it will be quick to compute ESS. Therefore, with a small amount of extra computation, the gap between two successive βs and the length of the distribution sequence in PSMC can be actively determined, which is a great advantage over the manual tunning in parallel tempering and tempered transition. By integrating all pieces together, we can write out a pseudo code of PSMC as in Algorithm 4.

3. Usually, there is no systematic way to determine the number of β_h in parallel tempering and tempered transition, and it is selected empirically.

Algorithm 3 Finding $\Delta\beta_h$

```
Input:    a particle set $\{\bar{\mathbf{x}}^{(s)}\}_{s=1}^{S}$, $\beta_{h-1}$
 1: $l \leftarrow 0$, $u \leftarrow \beta_{h-1}$, $\alpha \leftarrow 0.05$
 2: while $|u - l| \geq 0.005$ and $l \leq \beta_{h-1}$ do
 3:     compute ESS $\sigma$ by replacing $\Delta\beta_h$ with $-\alpha$ according to (16)
 4:     if $\sigma < \sigma*$ then
 5:         $u \leftarrow \alpha$, $\alpha \leftarrow (l + \alpha)/2$
 6:     else
 7:         $l \leftarrow \alpha$, $\alpha \leftarrow (\alpha + u)/2$
 8:     end if
 9: end while
Output:    Return $\Delta\beta_h = \max(-\alpha, -\beta_{h-1})$
```

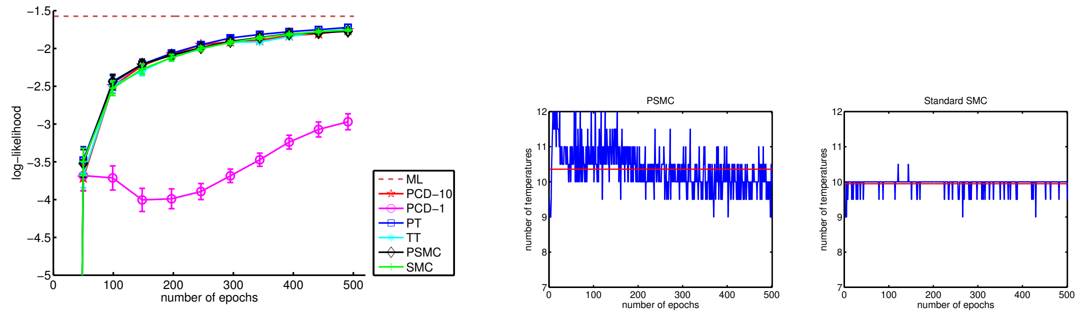
Algorithm 4 Learning with PSMC

```
Input:    a particle set $\{\mathbf{x}^{(m)}\}_{m=1}^{M}$, learning rate $\eta$
 1: Initialize $p(\mathbf{x}; \boldsymbol{\theta}_0), t \leftarrow 0$
 2: Sample particles $\{\bar{\mathbf{x}}_0^{(s)}\}_{s=1}^{S} \sim p(\mathbf{x}; \boldsymbol{\theta}_0)$
 3: while ! stop criterion do
 4:     $h \leftarrow 0$, $\beta_0 \leftarrow 1$
 5:     while $\beta_h < 1$ do
 6:         assign importance weights $\{w^{(s)}\}_{s=1}^{S}$ to particles according to (15)
 7:         resample particles based on $\{w^{(s)}\}_{s=1}^{S}$
 8:         compute the step length $\Delta\beta_h$ according to Algorithm 3
 9:         $\beta_{h+1} = \beta_h + \delta\beta$
10:         $h \leftarrow h + 1$
11:     end while
12:     Compute the gradient $\Delta\boldsymbol{\theta}_t$ according to (3)
13:     $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta\Delta\boldsymbol{\theta}_t$
14:     $t \leftarrow t + 1$
15: end while
Output:    estimated parameters $\boldsymbol{\theta}^* = \boldsymbol{\theta}_t$
```

6. Experiments

In our experiments, PCD, parallel tempering (PT), tempered transition (TT), standard SMC and PSCM were empirically compared on 2 different discrete UGMs, *i.e.* fully visible Boltzmann machines (VBMs) and restricted Boltzmann machines (RBMs). As we analyzed in section 4, large learning rate and high model complexity are two main challenges for learning UGMs. Therefore, two experiments were constructed to test the robustness of algorithms to different learning rates and model complexities separately. On one hand, one VBM was constructed with small size and tested with synthetic data. The purpose of the small-scale VBM is to reduce the effect of model complexity. In addition, the exact log-likelihood can be computed in this model. On the other hand, two RMBs were used in our second experiment, one is medium-scale and the other is large-scale.

learning rate $\eta_t = \frac{1}{100+t}$



(a) (b) (c)

Figure 2: The performance of algorithms with the first learning rate scheme. (a): log-likelihood *vs.* number of epochs; (b) and (c): the number of βs in PSMC and SMC at each iteration (blue) and their mean values (red).

They were applied on a real-world database MNIST[4]. In this experiment, the learning rate was set to be small to avoid its effect. In both experiments, mini-patch of 200 data instances were used. When PSMC and SMC were run, $\sigma^* = 0.9$ was used as the threshold of ESS. We recorded the number of βs at each iteration in PSMC, and computed the average value H. In order to ensure the fairness of the comparison, we offset the computation of different algorithms. In PT, $H\beta$s were uniformly assigned between 0 and 1. In TT, similarly, H βs were uniformly distributed in the range $[0.9, 1]$[5]. Two PCD algorithms were implemented, one is with one-step Gibbs sampling (PCD-1) and the other is with H-step Gibbs sampling (PCD-H). In the second experiment, the computation of log-likelihoods is intractable, so here we employed an annealing importance sampling (AIS)-based estimation proposed by Salakhutdinov and Murray (2008). All methods were run on the same hardware and experimental conditions unless otherwise mentioned.

6.1. Experiments with Different Learning Rates

A Boltzmann machine is a kind of stochastic recurrent neural network with fully connected variables. Each variable takes binary value $\mathbf{x} \in \{-1, +1\}^D$. Using the energy representation (2), parameters $\boldsymbol{\theta}$ correspond to $\{\mathbf{W} \in \mathbb{R}^{D\times D}, \mathbf{b} \in \mathbb{R}^{D\times 1}\}$ and $\phi(\mathbf{x}) = \{\mathbf{x}\mathbf{x}^\top, \mathbf{x}\}$. Here we used a fully visible Boltzmann machine (VBM), and computed the log-likelihood to quantify performances. In this experiment, a small-size VBM with only 10 variables is used to avoid the effect of model complexity. For simplicity, $W_{ij_{i,j\in[1,10]}}$ were randomly generated from an identical distribution $\mathcal{N}(0, 1)$, and 200 training data instances were sampled. Here we tested all learning algorithms with 3 different learning rate schemes: (1) $\eta_t = \frac{1}{100+t}$, (2) $\eta_t = \frac{1}{20+0.5\times t}$, (3) $\eta_t = \frac{1}{10+0.1\times t}$. The learning rates in the three schemes were at different magnitude levels. The first one is smallest, the second is intermediate and the last one is relative large. For the first scheme, 500 epochs were run, and the log-likelihood *vs.* number of epochs plots of different learning algorithms are presented in

4. http://yann.lecun.com/exdb/mnist/index.html
5. In our experiment, we used a TT similar to that used by Salakhutdinov (2010) by alternating between one Gibbs sampling and one tempered transition.

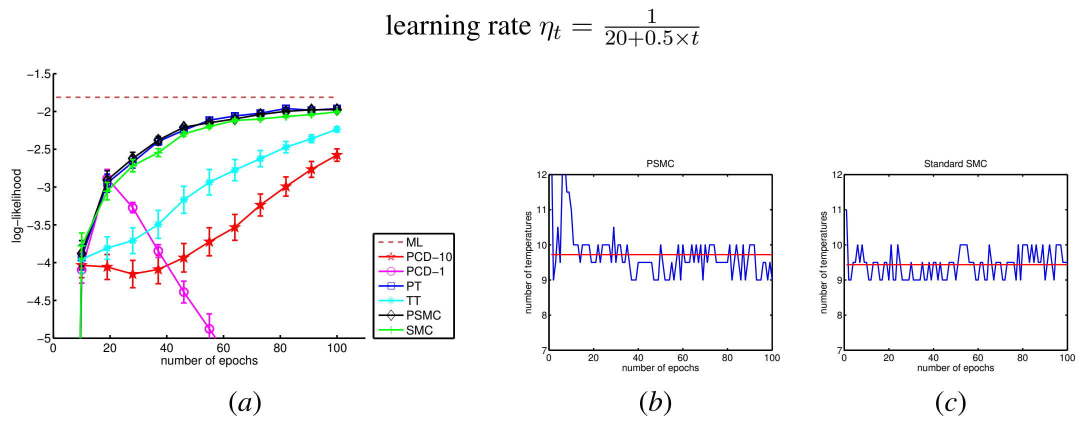


Figure 3: The performance of algorithms with the second learning rate scheme. (a): log-likelihood *vs.* number of epochs; (b) and (c): the number of βs in PSMC and SMC at each iteration (blue) and their mean values (red).

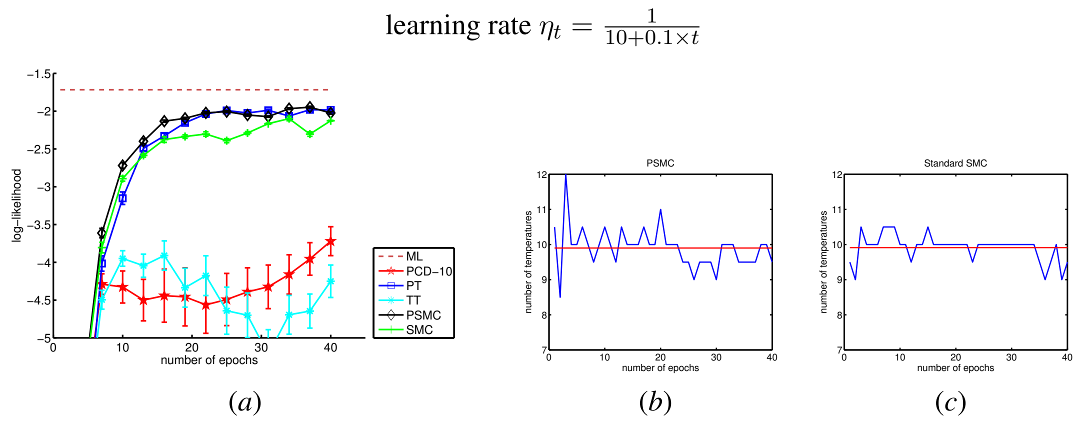


Figure 4: The performance of algorithms with the third learning rate scheme. (a): log-likelihood *vs.* number of epochs; (b) and (c): the number of βs in PSMC and SMC at each iteration (blue) and their mean values (red).

Figure 2(*a*). The number of βs in PSMC and SMC are also plotted in Figures 2(*b*) and 2(*c*) respectively. We can see that the mean value H in PSMC is around 10, which is slightly higher than the one in SMC. For the second and third learning rate schemes, we ran 100 and 40 epochs respectively. All algorithms' performances are shown in Figure 3(*a*) and 4(*a*). We found that the number of βs in PSMC and SMC are very similar to those of the first scheme (Figures 3(*b*), 3(*c*), 4(*b*) and 4(*c*)). For all three schemes, 5 trials were run with different initial parameters, and the results are presented with mean values (curves) and standard deviations (error bars). In addition, maximum likelihood (ML) solutions were obtained by computing exact gradients (3). For better quantitative comparison, the average log-likelihoods based on the parameters learned from six algorithms and three learn-

Models		(Avg.) Log-Likelihoods					
(Size)	Learning rate schemes	PCD-1	PCD-H	PT	TT	SMC	PSMC
VBM	$\eta_t = \frac{1}{100+t}$	-1.693	-1.691	$\mathbf{-1.689}$	-1.692	-1.692	-1.691
(15)	$\eta_t = \frac{1}{20+0.5\times t}$	-7.046	-2.612	-1.995	-2.227	-2.069	$\mathbf{-1.891}$
	$\eta_t = \frac{1}{10+0.1\times t}$	-25.179	-3.714	-2.118	-4.329	-2.224	$\mathbf{-1.976}$
	MNIST						
RBM	training data	-206.3846	-203.5884	206.2819	-206.9033	-203.3672	$\mathbf{-199.9089}$
(784 × 10)	testing data	-207.7464	-204.6717	206.2819	-208.2452	-204.4852	$\mathbf{-201.0794}$
RBM	training data	-176.3767	-173.0064	-165.2149	-170.9312	-678.6464	$\mathbf{-161.6231}$
(784 × 500)	testing data	-177.0584	-173.4998	-166.1645	-171.6008	-678.7835	$\mathbf{-162.1705}$

Table 1: Comparison of Avg.log-likelihoods with parameters learned from different learning algorithms and conditions.

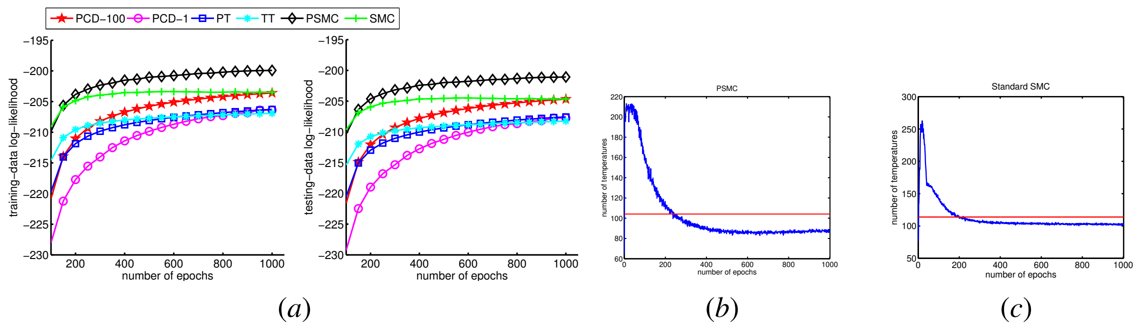


Figure 5: The performance of algorithms on the medium-scale RBM. (a): log-likelihood *vs.* number of epochs for both training images (left) and testing images (right) in the MNIST database; (b) and (c): the number of βs in PSMC and SMC at each iteration (blue) and their mean values (red).

ing rate schemes are listed in the upper part of Table 1. The results of the first experiment can be summarized as follows:

1. When the learning rate was small, PT, TT, SMC, PSMC and PCD-10 worked similarly well, outperforming PCD-1 by a large margin.

2. When the learning rate was intermediate, PT and PSMC still worked successfully, which were closely followed by SMC. TT and PCD-10 deteriorated, while PCD-1 absolutely failed.

3. When the learning rate went to relatively large, the fluctuation patterns were obvious in all algorithms. Meanwhile, the performance gaps between PSMC and other algorithms was larger. In particular, TT and PCD-10 deteriorated very much. Since PCD-1 failed even worse in this case, its results are not plotted in Figure 4(a).

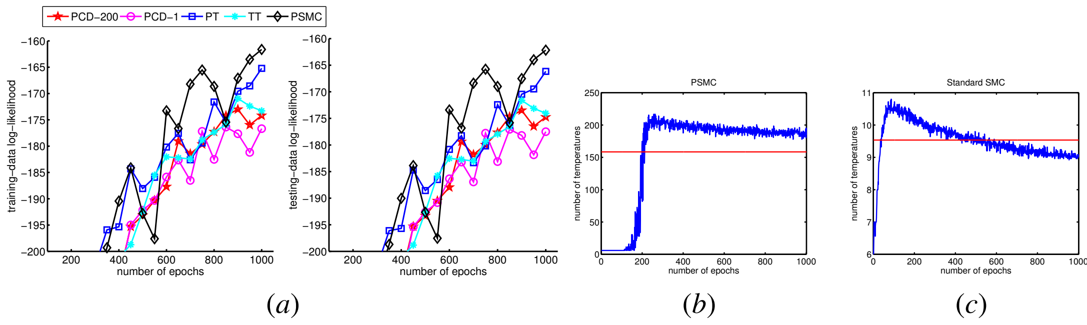


Figure 6: The performance of algorithms on the large-scale RBM. (a): log-likelihood *vs.* number of epochs for both training images (left) and testing images (right) in the MNIST database; (b) and (c): the number of βs in PSMC and SMC at each iteration (blue) and their mean values (red).

6.2. Experiments with Models of Different Complexities

In our second experiment, we used the popular restricted Boltzmann machine to model handwritten digit images (with the MNIST database). RBM is a bipartite Markov network consisting of a visible layer and a hidden layer, it is a “restricted” version of Boltzmann machine with only interconnections between the hidden layer and the visible layer. Assuming that the input data are binary and N_v-dimensional, each data point is fed into the N_v units of the visible layer $\mathbf{v}$, and N_h units in hidden layer $\mathbf{h}$ are also stochastically binary variables (latent features). Usually, $\{0, 1\}$ is used to represent binary values in RBMs to indicate the activations of units.The energy function $E(\mathbf{v}, \mathbf{h})$ is defined as $E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^\top \mathbf{W}\mathbf{h} - \mathbf{h}^\top \mathbf{b} - \mathbf{v}^\top \mathbf{c}$, where $\mathbf{W} \in \mathbb{R}^{N_v \times N_h}$, $\mathbf{b} \in \mathbb{R}^{N_v \times 1}$ and $\mathbf{c} \in \mathbb{R}^{N_h \times 1}$. Although there are hidden variables in the energy function, the gradient of likelihood function can be written out in a form similar to (3) (Hinton, 2002). Images in the MNIST database are 28×28 handwritten digits, *i.e.* N_v=784. To avoid the effect of learning rate, in this experiment, a small learning rate scheme $\eta_t = \frac{1}{100+t}$ was used and 1000 epochs were run in all learning algorithms. Two RBMs were constructed for testing the robustness of learning algorithms to model complexity, one medium-scale with 10 hidden variables (*i.e.* $\mathbf{W} \in \mathbb{R}^{784 \times 10}$), the other large-scale with 500 hidden variables (*i.e.* $\mathbf{W} \in \mathbb{R}^{784 \times 500}$)[6]. Similarly to the first experiment, we first ran PSMC and SMC, and recorded the number of triggered βs at each iteration and their mean values (Figure 5(*b*), 5(*c*), 6(*b*) and 6(*c*)). For the medium-scale model, the number of βs in PSMC and SMC are similar (around 100). However, for the large-scale model, the mean value of $|\{\beta_0, \beta_1, \cdots\}|$ is 9.6 in SMC while 159 in PSMC. The reason for this dramatic change in SMC is that all 200 particles initialized from the uniform distribution were depleted when the distribution gets extremely complex. For other learning algorithms, H was set 100 and 200 in the medium- and large-scale cases, respectively. Since there are 60000 training images and 10000 testing images in the MNIST database, we plotted both training-data log-likelihoods and testing-data log-likelihoods as learning progressed (see Figure 5(*a*) and 6(*a*)). More detailed quantitative comparison can be seen in the lower part of Table 1. Similarly, we conclude the results of the second experiments as follows:

6. Since a small-scale model was already tested in the first experiment, we did not repeat it here.

1. When the scale of RBM was medium, PSMC worked best by reaching the highest training-data and testing-data log-likelihoods. SMC and PCD-100 arrived the second highest log-likelihoods, although SMC converged much faster than PCD-100. PT, TT and PCD-1 led to the lowest log-likelihoods although PT and TT raised log-likelihoods more quickly than PCD-1.

2. When the scale of RBM was large, all algorithms displayed fluctuation patterns. Meanwhile, PSMC still worked better than others by obtaining the highest log-likelihoods. PT ranked second, and TT ranked third, which was slightly better than PCD-200. PCD-1 ranked last. SMC failed in learning the large-scale RBM, so its results are not presented in Figure 6(*a*).

7. Conclusion

A SMC interpretation framework of learning UGMs was presented, within which two main challenges of the learning task were disclosed as well. Then, a persistent SMC (PSMC) learning algorithm was developed by applying SMC more deeply in learning. According to our experimental results, the proposed PSMC algorithm demonstrates promising stability and robustness in various challenging circumstances with comparison to state-of-the-art methods. Meanwhile, there still exist much room for improvement of PSMC, *e.g.* using adaptive MCMC transition (Schäfer and Chopin, 2013; Jasra et al., 2011), which suggests many possible directions for future work. Besides, although PSMC is expected to approach the maximum likelihood solution in learning UGMs, sometimes maximizing the posterior function is more desirable (*e.g.* when the prior is available), so it is also interesting to extend PSMC for maximum a posteriori learning.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

References

Arthur U. Asuncion, Qiang Liu, Alexander T. Ihler, and Padhraic Smyth. Particle filtered MCMC-MLE with connections to contrastive divergence. In *ICML*, 2010.

Miguel A. Carreira-Perpinan and Geoffrey E. Hinton. On Contrastive Divergence Learning. In *AISTATS*, 2005.

Guillaume Desjardins, Aaron Courville, Yoshua Bengio, Pascal Vincent, and Olivier Delalleau. Tempered Markov Chain Monte Carlo for training of restricted Boltzmann machines. In *AISTATS*, 2010.

Charles J. Geyer. *Markov Chain Monte Carlo Maximum Likelihood.* Compuuting Science and Statistics:Proceedings of the 23rd Symposium on the Interface, 1991.

Mark S. Handcock, David R. Hunter, Carter T. Butts, Steven M. Goodreau, and Martina Morris. statnet: Software tools for the representation, visualization, analysis and simulation of network data. *Journal of Statistical Software*, pages 1548–7660, 2007.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

Ajay Jasra, David A. Stephens, Arnaud Doucet, and Theodoros Tsagaris. Inference for lévy-driven stochastic volatility models via adaptive sequential monte carlo. *Scandinavian Journal of Statistics*, 38:1–22, 2011.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Augustine Kong, Jun S. Liu, and Wing H. Wong. Sequential Imputations and Bayesian Missing Data Problems. *Journal of the American Statistical Association*, 89(425):278–288, March 1994.

Radford Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and Computing*, 6:353–366, 1994.

H. Robbins and S. Monro. A Stochastic Approximation Method. *Ann.Math.Stat.*, 22:400–407, 1951.

Ruslan Salakhutdinov. Learning in markov random fields using tempered transitions. In *NIPS*, 2010.

Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *ICML*, 2008.

Christian Schäfer and Nicolas Chopin. Sequential monte carlo on large binary sampling spaces. *Statistics and Computing*, 23(2):163–184, 2013.

T. Tieleman. Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient. In *ICML*, pages 1064–1071, 2008.

T. Tieleman and G.E. Hinton. Using Fast Weights to Improve Persistent Contrastive Divergence. In *ICML*, pages 1033–1040. ACM New York, NY, USA, 2009.