Mach Learn (2016) 103:239–260
DOI 10.1007/s10994-016-5564-x

Learning undirected graphical models using persistent sequential Monte Carlo

Hanchen Xiong[1] · **Sandor Szedmak**[2] · **Justus Piater**[3]

Received: 8 April 2015 / Accepted: 22 February 2016 / Published online: 1 April 2016

Abstract Along with the popular use of algorithms such as persistent contrastive divergence, 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, comparing it with related methods, on carefully designed experiments with both synthetic and real-world data. Our empirical results demonstrate that PSMC compares favorably with the state of the art by consistently yielding the highest (or among the highest) likelihoods. We also evaluated PSMC on two practical

Hanchen Xiong and Sandor Szedmak contributed to most of this work while they were at University of Innsbruck.

Editors: Hang Li, Dinh Phung, Tru Cao, Tu-Bao Ho, and Zhi-Hua Zhou.

✉ Hanchen Xiong
hanchen.xiong@zalando.de

Sandor Szedmak
sandor.szedmak@aalto.fi

Justus Piater
justus.piater@uibk.ac.at

[1] Zalando SE, Berlin, Germany

[2] Department of Computer Science, Aalto University, Espoo, Finland

[3] Institute of Comptuer Science, University of Innsbruck, Innsbruck, Austria

tasks, multi-label classification and image segmentation, in which PSMC displays promising applicability by outperforming others.

Keywords Sequential Monte Carlo · Maximum likelihood learning · Undirected graphical models

1 Introduction

Learning undirected graphical models (UGMs), e.g. Markov random fields (MRFs) or conditional random fields (CRFs), 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. by 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 Sect. 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 (Sect. 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 used 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 neighboring 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; Younes 1988) 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* (Sect. 4). Based on this rationale, the strengths and limitations of different learning algorithms can be analyzed and understood in a new light. This to some extent can be considered as an extension of the work from Asuncion et al. (2010) with wider generalization and deeper exploitation of the SMC interpretation of learning UGMs. 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 (Sect. 5). According to our empirical results on learning two discrete UGMs (Sect. 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:

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

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

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 the 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 the $\phi(\mathbf{x}^{(m)})$, $\mathbf{x}^{(m)} \in \mathcal{D}$. 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 sizes. 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 the entire 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 $p(\mathbf{x}; \boldsymbol{\theta_0})$ is set up in the same form as (1) and (2), and we have

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

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

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

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

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 the average log-likelihood, we can compute the corresponding gradient (noting that $\mathbf{Z}(\boldsymbol{\theta}_0)$ will be eliminated since it corresponds to a constant in the logarithm) as

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

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. 2008). Pseudocode of the MCMCML learning algorithm is presented in Algorithm 1.

Algorithm 1 MCMCML Learning Algorithm

Input: training data $\mathcal{D} = \{\mathbf{x}^{(m)}\}_{m=1}^{M}$; learning rate η; gap L between two successive proposal distribution resets

$t \leftarrow 0$, initialize the proposal distribution $p(\mathbf{x}; \boldsymbol{\theta}_0)$
while ! stop criterion **do**
  if $(t \mod L) == 0$ **then**
    (Re)set the proposal distribution to $p(\mathbf{x}; \boldsymbol{\theta}_t)$
    Sample $\{\bar{\mathbf{x}}^{(s)}\}$ from $p(\mathbf{x}; \boldsymbol{\theta}_t)$
  end if
  Calculate $w^{(s)}$ using (8)
  Calculate gradient $\frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta}|\mathcal{D})}{\partial \boldsymbol{\theta}}$ using (9)
  update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta \frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta}|\mathcal{D})}{\partial \boldsymbol{\theta}}$
  $t \leftarrow t + 1$
end while

Output: estimated parameters $\boldsymbol{\theta}^* = \boldsymbol{\theta}_t$

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, $KL(p^0|p_{\boldsymbol{\theta}}^{\infty})$ and $KL(p_{\boldsymbol{\theta}}^{n}|p_{\boldsymbol{\theta}}^{\infty})$, where $p^0 = p(\mathcal{D}; \boldsymbol{\theta})$, $p_{\boldsymbol{\theta}}^{n} = p(\bar{\mathcal{D}}_n; \boldsymbol{\theta})$ with $\bar{\mathcal{D}}_n$ denoting the data sampled after n steps of Gibbs sampling with parameter $\boldsymbol{\theta}$, and $p_{\boldsymbol{\theta}}^{\infty} = p(\bar{\mathcal{D}}_{\infty}; \boldsymbol{\theta})$ 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 $p_{\boldsymbol{\theta}}^{1}$. The key advantage of CD-1 is that it efficiently approximates ψ^- in the likelihood gradient (3) by running only one step of 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 the 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; Younes 1988) 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$\rightarrow$ML when $\eta \rightarrow 0$). 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 later stages of learning, PCD often 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 $0 \leq \beta_H < \beta_{H-1} < \cdots < \beta_0 = 1$. In particular, β_0 corresponds to the original complex distribution. When h increases, the distribution becomes increasingly 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 neighboring 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 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

Algorithm 2 SAP for learning UGMs

Input: training data $\mathcal{D} = \{\mathbf{x}^{(m)}\}_{m=1}^{M}$.
1: $t \leftarrow 0$, initialize the proposal distribution $p(\mathbf{x};\boldsymbol{\theta}_0)$.
2: Randomly initialize S sample particles $\{\tilde{\mathbf{x}}_0^{(s)}\}_{s=1}^{S}$
3: **while** ! stop criterion **do**
4: **for** s=1:S **do**
5: evolve particle $\tilde{\mathbf{x}}_t^{(s)}$ to $\tilde{\mathbf{x}}_{t+1}^{(s)}$ with a transition operator which leaves $p(\mathbf{x};\boldsymbol{\theta}_t)$ invariant
6: **end for**
7: Calculate gradient $\frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta}|\mathcal{D})}{\partial \boldsymbol{\theta}}$ using (3)
8: update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta_t \frac{\partial \tilde{\mathcal{L}}(\boldsymbol{\theta}|\mathcal{D})}{\partial \boldsymbol{\theta}}$
9: $t \leftarrow t + 1$, decrease learning rate η_t
10: **end while**
Output: estimated parameters $\boldsymbol{\theta}^* = \boldsymbol{\theta}_t$

approximation procedure (SAP; Younes 1988; Tieleman and Hinton 2009; Desjardins et al. 2010; Salakhutdinov 2010). A pseudocode of SAP is presented in Algorithm 2. 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 based on SAP due to the time inhomogeneity.

Asuncion et al. (2010) considered that PCD can be interpreted as a sequential Monte Carlo procedure by extending MCMCML to a particle filtered version. To give 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; Sect. 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 $w^{(s)} = \exp(\boldsymbol{\theta_t - \theta_{t-1}})^\top \phi(\bar{\mathbf{x}}^{(s)})$. Since the parameters $\boldsymbol{\theta}$ do not change very much across 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 a small amount of extra computation.

More interestingly, it was pointed out that PCD also fits the above sequential Monte Carlo procedure: importance reweighting + resampling + MCMC transition (Chopin 2002; Del Moral et al. 2006). One property worth noting is that PCD uses uniform weights for all particles and enforce a Gibbs sampling as the 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 3). Note that all particle weights are uniformly assigned; resampling has no effect and can be omitted. In addition, the MCMC transition step is forced to take place at every iteration.

It is also worth noting that when applying algorithms in Algorithm 3, we are not interested in particles from any individual target distribution (which is usually the purpose of SMC). Instead, we want to obtain particles faithfully sampled from all sequential distributions. In our case of learning UGMs, sequential distributions are learned by iterative updates. Therefore, learning and sampling are intertwined. It can be easily imagined that one badly sampled particle set at the 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.

At the first sight of Algorithm 3, the SMC interpretation of learning UGMs seems ad hoc and far-fetched since all particles are uniformly reweighted and therefore no resampling. However, it can be argued that Algorithm 3 is a perfectly valid SMC procedure when $|\eta\Delta\boldsymbol{\theta}_t| \to 0$ since all weights $w^{(s)} = \lim_{|\eta\Delta\boldsymbol{\theta}_t|\to 0} \frac{p(\bar{\mathbf{x}}_t^{(s)};\boldsymbol{\theta}_t)}{p(\bar{\mathbf{x}}_t^{(s)};\boldsymbol{\theta}_t+\eta\Delta\boldsymbol{\theta}_t)} = 1$. Therefore, the SMC interpretation scheme holds when the gaps between successive distributions are relatively small. Or, in other words, it is inappropriate to use uniform weights when the

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

Algorithm 3 Interpreting Learning as SMC

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

gaps are large. By using uniform reweighting, the larger gaps exist between successive distributions, the more badly the SMC scheme will be violated and therefore the performance will be harmed. Obviously, one straightforward way to ensure good performance is to reduce gaps. This is consistent with SAP learning framework where small learning rates are preferred. Meanwhile, by following SMC thinking, another possible remedy is to use real importance reweighting instead of the uniform one. Actually, this is the key advantageous property of the SMC interpretation beyond the SAP one. A new avenue of approximate learning for UGMs is revealed, where many possible improvements can be achieved by bring in state-of-the-art outcomes of SMC research. Particle filtered MCMCML (Asuncion et al. 2010) and the novel method introduced later are two successful examples under this consideration.

In addition, the gaps between successive distributions also matter in MCMC transition. Within the SMC-based interpretation, we can see that the 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–Hastings (MH) move based on a 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 the sampling space sufficiently well. However, when the learning rate is large or two sequential distributions exhibt multiple modes (e.g. at a 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.

Above all, since the update at each iteration is conducted as $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta \Delta \boldsymbol{\theta}_t$, the gap between $p(\mathbf{x}; \boldsymbol{\theta}_t)$ and $p(\mathbf{x}; \boldsymbol{\theta}_{t+1})$ can be intuitively understood as the product of *learning rate* η and *model complexity* $\mathcal{O}(\boldsymbol{\theta})$. Therefore, we consider learning rate and model complexity[2] are two key factors that challenge learning algorithms. We can verify this argument by checking the Kullback-Leibler divergence between $p(\mathbf{x}; \boldsymbol{\theta}_t)$ and $p(\mathbf{x}; \boldsymbol{\theta}_{t+1})$:

$$KL(p(\mathbf{x}; \boldsymbol{\theta}_t) || p(\mathbf{x}; \boldsymbol{\theta}_{t+1})) \tag{13}$$

$$= \sum_{\mathbf{x}} p(\mathbf{x}; \boldsymbol{\theta}_t) \log \frac{p(\mathbf{x}; \boldsymbol{\theta}_t)}{p(\mathbf{x}; \boldsymbol{\theta}_{t+1})} \tag{14}$$

$$= \mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta}_t)} \left[\log \left(\frac{\exp(\boldsymbol{\theta}_t^\top \phi(\mathbf{x}))}{\exp(\boldsymbol{\theta}_t^\top \phi(\mathbf{x}) + \eta \Delta \boldsymbol{\theta}_t^\top \phi(\mathbf{x}))} \cdot \frac{\sum_{\mathbf{x}} \exp(\boldsymbol{\theta}_t^\top \phi(\mathbf{x}) + \eta \Delta \boldsymbol{\theta}_t^\top \phi(\mathbf{x}))}{\sum_{\mathbf{x}} \exp(\boldsymbol{\theta}_t^\top \phi(\mathbf{x}))} \right) \right] \tag{15}$$

$$= \mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta}_t)} \left[\log \left(\exp(-\eta \Delta \boldsymbol{\theta}_t^\top \phi(\mathbf{x}) \right) + \log \left(\frac{\sum_{\mathbf{x}} \left(\exp(\boldsymbol{\theta}_t^\top \phi(\mathbf{x})) \cdot \exp(\eta \Delta \boldsymbol{\theta}_t^\top \phi(\mathbf{x})) \right)}{\sum_{\mathbf{x}} \exp(\boldsymbol{\theta}_t^\top \phi(\mathbf{x}))} \right) \right] \tag{16}$$

$$= \mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta}_t)} \left[-\eta \Delta \boldsymbol{\theta}_t^\top \phi(\mathbf{x}) + \log \left(\sum_{\mathbf{x}} \left(\frac{\exp(\boldsymbol{\theta}_t^\top \phi(\mathbf{x}))}{\sum_{\mathbf{x}} \exp(\boldsymbol{\theta}_t^\top \phi(\mathbf{x}))} \exp(\eta \Delta \boldsymbol{\theta}_t^\top \phi(\mathbf{x})) \right) \right) \right] \tag{17}$$

$$= \mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta}_t)} \left[-\eta \Delta \boldsymbol{\theta}_t^\top \phi(\mathbf{x}) \right] + \mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta}_t)} \left[\log \left(\sum_{\mathbf{x}} p(\mathbf{x}; \boldsymbol{\theta}_t) \exp(\eta \Delta \boldsymbol{\theta}_t^\top \phi(\mathbf{x})) \right) \right] \tag{18}$$

$$= \mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta}_t)} \left[\log \left(\mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta}_t)} \left[\exp(\eta \Delta \boldsymbol{\theta}_t^\top \phi(\mathbf{x})) \right] \right) \right] - \mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta}_t)} \left[\eta \Delta \boldsymbol{\theta}_t^\top \phi(\mathbf{x}) \right] \tag{19}$$

$$= \log \left(\mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta}_t)} \left[\exp(\eta \Delta \boldsymbol{\theta}_t^\top \phi(\mathbf{x})) \right] \right) - \mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta}_t)} \left[\eta \Delta \boldsymbol{\theta}_t^\top \phi(\mathbf{x}) \right] \tag{20}$$

Based on Jensen's inequality and concavity of log function:

$$\begin{aligned} \log \left(\mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta}_t)} \left[\exp(\eta \Delta \boldsymbol{\theta}_t^\top \phi(\mathbf{x})) \right] \right) &\geq \mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta}_t)} \left[\log \left(\exp(\eta \Delta \boldsymbol{\theta}_t^\top \phi(\mathbf{x})) \right) \right] \\ &= \mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta}_t)} \left[\eta \Delta \boldsymbol{\theta}_t^\top \phi(\mathbf{x}) \right] \end{aligned} \tag{21}$$

which leads to a trivial result from (20), i.e. non-negativity of Kullback-Leibler divergence. Here, to investigate how learning rate and model complexity can affect the Kullback-Leibler divergence, we will exploit an upper bound of Jensen's inequality (Dragomir 1999–2000).

Theorem 1 *(Dragomir 1999–2000) $X = \{x_i\}$ is a finite sequence of real numbers belonging to a fixed closed interval $I = [a, b], a < b$, and $P = \{p_i\}, \sum_i p_i = 1$ is a sequence of positive weights associated with X. If f is a differentiable convex function on I, then we have:*

$$\sum_i p_i f(x_i) - f\left(\sum_i p_i x_i\right) \leq \frac{1}{4}(b-a)(f'(b) - f'(a)) \tag{22}$$

[2] Here we consider the dimensionality of a distribution as its complexity, since high-dimensional distributions can more easily establish multiple modes than low dimensional ones.

Now we can write our result.

Corollary 1 *In Algorithm 3, the upper bound of the Kullback-Leibler divergence between successive distributions, i.e.* $p(\mathbf{x};\boldsymbol{\theta}_t)$ *and* $p(\mathbf{x};\boldsymbol{\theta}_{t+1})$ *is monotonically increasing with respect to learning rate* η *and model complexity* $\mathcal{O}(\boldsymbol{\theta})$.

Proof Let $U = \max_{\mathbf{x}}\{\exp(\eta\Delta\boldsymbol{\theta}_t^\top\phi(\mathbf{x}))\}$, $L = \min_{\mathbf{x}}\{\exp(\eta\Delta\boldsymbol{\theta}_t^\top\phi(\mathbf{x}))\}$, then substitute f, b, a in (22) with $-log, U, L$ respectively:

$$\begin{aligned}&\log\left(\mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta}_t)}\left[\exp(\eta\Delta\boldsymbol{\theta}_t^\top\phi(\mathbf{x}))\right]\right) - \mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta}_t)}\left[\eta\Delta\boldsymbol{\theta}_t^\top\phi(\mathbf{x})\right]\\ &\leq \tfrac{1}{4}(U-L)\left(\tfrac{1}{L}-\tfrac{1}{U}\right) = \tfrac{1}{4}\left(\tfrac{U}{L}+\tfrac{L}{U}-2\right)\end{aligned} \tag{23}$$

By combing (23) and (20),we have:

$$^{\sqcap}\{KL(p(\mathbf{x};\boldsymbol{\theta}_t)||p(\mathbf{x};\boldsymbol{\theta}_{t+1}))\} = \frac{1}{4}g\left(\frac{U}{L}\right) - \frac{1}{2} \tag{24}$$

where $^{\sqcap}\{\cdot\}$ denotes upper bound, $g(z) = z + \frac{1}{z}$ and it is monotonically increasing when $z \geq 1$. We can further denote $\Delta\boldsymbol{\theta}$ as

$$\Delta\boldsymbol{\theta} = |\Delta\boldsymbol{\theta}|\mathbf{e}_{\Delta\boldsymbol{\theta}} = \mathbf{e}_{\Delta\boldsymbol{\theta}}\left(\sum_{i=1}^{\mathcal{O}(\boldsymbol{\theta})}|\Delta\theta_i|^2\right)^{1/2} \tag{25}$$

where $\mathbf{e}_{\Delta\boldsymbol{\theta}}$ is the unit vector of the same direction as $\Delta\boldsymbol{\theta}$, $\Delta\theta_i$ is the magnitude of $i-$th dimensional gradient. Then (24) can be rewritten as:

$$\begin{aligned}^{\sqcap}\{KL(p(\mathbf{x};\boldsymbol{\theta}_t)||p(\mathbf{x};\boldsymbol{\theta}_{t+1}))\} = \frac{1}{4}g\left[\left(\frac{\max_{\mathbf{x}}\{\exp(\mathbf{e}_{\Delta\boldsymbol{\theta}}^\top\phi(\mathbf{x}))\}}{\min_{\mathbf{x}}\{\exp(\mathbf{e}_{\Delta\boldsymbol{\theta}}^\top\phi(\mathbf{x}))\}}\right)^{\eta\left(\sum_{i=1}^{\mathcal{O}(\boldsymbol{\theta})}|\Delta\theta_i|^2\right)^{1/2}}\right]\\ -\frac{1}{2}\end{aligned} \tag{26}$$

Since $h(z) = \omega^z$ is a monotonically increasing function when $\omega > 1$. Therefore, given fixed gradient magnitudes, it is obvious that $^{\sqcap}\{KL(p(\mathbf{x};\boldsymbol{\theta}_t)||p(\mathbf{x};\boldsymbol{\theta}_{t+1}))\}$ is monotonically increasing with respect to η and $\mathcal{O}(\boldsymbol{\theta})$. □

5 Persistent sequential Monte Carlo

It was explained that learning UGMs can be interpreted as a SMC procedure. Within the SMC interpretation, it is quite clear that most existing methods will deteriorate when learning rates or models' complexities are high due to uniform reweighing. A trivial yet novel cure is to employ importance weighting to make it a real SMC. In addition, 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

$$p_h(\mathbf{x};\boldsymbol{\theta}_{t+1}) \propto p(\mathbf{x};\boldsymbol{\theta}_t)^{1-\beta_h}p(\mathbf{x};\boldsymbol{\theta}_{t+1})^{\beta_h}, \tag{27}$$

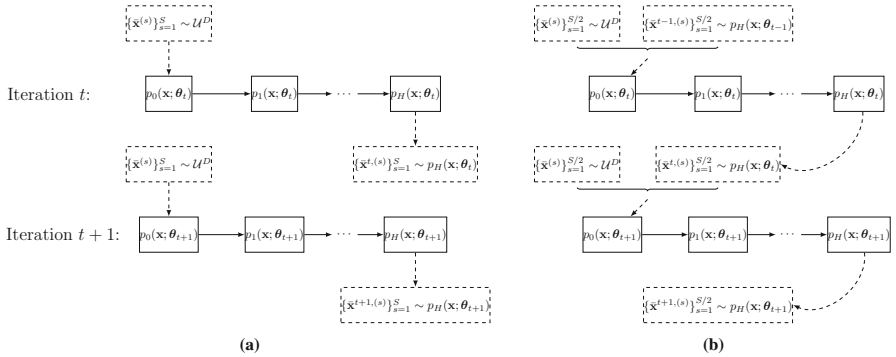

Fig. 1 Schematics of **a** standard sequential Monte Carlo and **b** persistent sequential Monte Carlo for learning UGMs. *Solid boxes* denote sequential distributions and *solid arrows* represent the move (resampling and MCMC transition) between successive distributions. *Dashed boxes* are particle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

where $0 \leq \beta_0 \leq \beta_1 \leq \cdots \leq \beta_H = 1$. In this way, the length of the distribution sequence will be extended in SMC. In addition, obviously, $p_H(\mathbf{x}; \boldsymbol{\theta}_{t+1}) = p(\mathbf{x}; \boldsymbol{\theta}_{t+1})$ while $p_0(\mathbf{x}; \boldsymbol{\theta}_{t+1}) = p(\mathbf{x}; \boldsymbol{\theta}_t)$. 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, the sub-sequential distributions are

$$p_h(\mathbf{x}; \boldsymbol{\theta}_{t+1}) \propto p(\mathbf{x}; \boldsymbol{\theta}_{t+1})^{\beta_h}, \tag{28}$$

where $0 \leq \beta_0 \leq \beta_1 \leq \cdots \leq \beta_H = 1$. The schematic figures of standard SMC and PSMC are presented in Fig. 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 of the particles are randomly initialized with a uniform distribution $\mathcal{U}^D$ (Fig. 1b). These extra, uniform samples balance particle degeneration and particle impoverishment, which is important in particular when the distribution has many modes (Li et al. 2014).

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 (27), the importance weighting for each particle is

$$\begin{aligned} w^{(s)} &= \frac{p_h(\bar{\mathbf{x}}^{(s)}; \boldsymbol{\theta}_{t+1})}{p_{h-1}(\bar{\mathbf{x}}^{(s)}; \boldsymbol{\theta}_{t+1})} \\ &= \exp\left(E(\bar{\mathbf{x}}^{(s)}; \boldsymbol{\theta}_t)\right)^{-\Delta\beta_h} \exp\left(E(\bar{\mathbf{x}}^{(s)}; \boldsymbol{\theta}_{t+1})\right)^{\Delta\beta_h} \qquad (29) \\ &= \exp\left((E(\bar{\mathbf{x}}^{(s)}; \boldsymbol{\theta}_{t+1}) - E(\bar{\mathbf{x}}^{(s)}; \boldsymbol{\theta}_t))\Delta\beta_h\right) \qquad (30) \end{aligned}$$

[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.

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

$$\sigma = \frac{\left(\sum_{s=1}^{S} w^{(s)}\right)^2}{S \sum_{s=1}^{S} w^{(s)2}} \in \left[\frac{1}{S}, 1\right] \tag{31}$$

Based on (30) and (31), 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 on ESS as σ^*, we can then find the biggest $\Delta\beta_h$ by using bidirectional search (see Algorithm 4). Usually a small particle set is used in learning (mini-batch 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. It is worth reminding that usually uniform temperatures are used in parallel tempering and tempered transition since the criterion for active tempering in them is lacking. By integrating all pieces together, we can write out a pseudo code of PSMC as in Algorithm 5.

Algorithm 4 Finding $\Delta\beta_h$

Input: a particle set $\{\bar{\mathbf{x}}^{(s)}\}_{s=1}^{S}$, β_h
$l \leftarrow 0, u \leftarrow 1, \alpha \leftarrow 0.05$
while $|u - l| \geq 0.005$ and $1 \geq u$ **do**
  compute ESS σ by replacing $\Delta\beta_h$ with α according to (31)
  if $\sigma < \sigma*$ **then**
    $u \leftarrow \alpha, \alpha \leftarrow (l + \alpha)/2$
  else
    $l \leftarrow \alpha, \alpha \leftarrow (\alpha + u)/2$
  end if
end while
Output: Return $\Delta\beta_h = \min(\alpha, 1 - \beta_h)$

Algorithm 5 Learning with PSMC

Input: a particle set $\{\mathbf{x}^{(m)}\}_{m=1}^{M}$, learning rate η
Initialize $p(\mathbf{x}; \boldsymbol{\theta}_0)$, $t \leftarrow 0$
Sample particles $\{\bar{\mathbf{x}}_0^{(s)}\}_{s=1}^{S} \sim p(\mathbf{x}; \boldsymbol{\theta}_0)$
while ! stop criterion **do**
  $h \leftarrow 0, \beta_0 \leftarrow 1$
  while $\beta_h < 1$ **do**
    assign importance weights $\{w^{(s)}\}_{s=1}^{S}$ to particles according to (30)
    resample particles based on $\{w^{(s)}\}_{s=1}^{S}$
    compute the step length $\Delta\beta_h$ according to Algorithm 4
    $\beta_{h+1} = \beta_h + \Delta\beta$
    $h \leftarrow h + 1$
  end while
  Compute the gradient $\Delta\boldsymbol{\theta}_t$ according to (3)
  $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta\Delta\boldsymbol{\theta}_t$
  $t \leftarrow t + 1$
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 Sect. 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 medium-scale and the other large-scale. 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-batches 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 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 a 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 performance. 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 three 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 Fig. 2a. The number of βs in PSMC and SMC are also plotted in Fig. 2b, 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 Figs. 3a and 4a. We found that the number of βs in PSMC and SMC are very similar to those of the first scheme (Figs. 3b, c, 4b, c). For all three schemes, 5 trials were

[4] http://yann.lecun.com/exdb/mnist/index.html.

[5] In our experiment, we used a TT similar to what used by Salakhutdinov (2010) by alternating between one Gibbs sampling and one tempered transition.

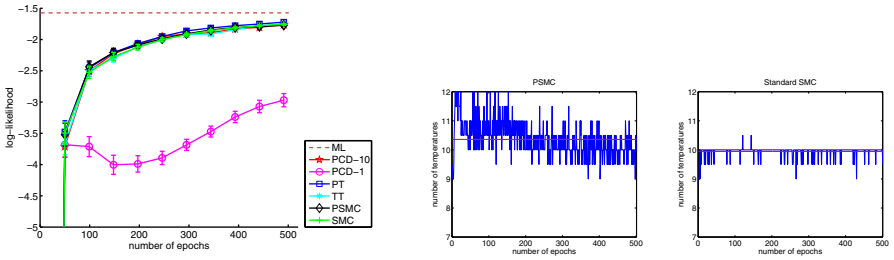

Fig. 2 performance of the algorithms with small learning rates. **a** log-likelihood versus number of epochs, **b**, **c** the number of βs in PSMC and SMC at each iteration (*blue*) and their mean values (*red*) (Color figure online)

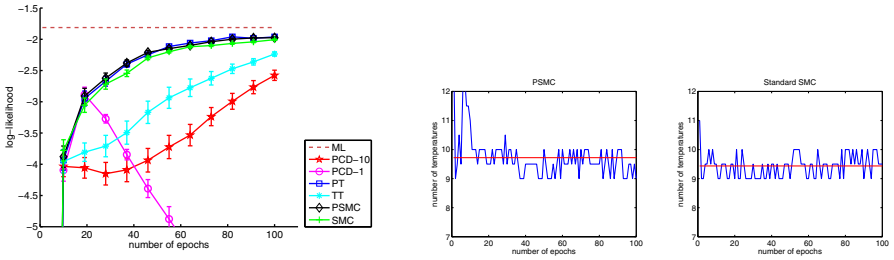

Fig. 3 Performance of the algorithms with intermediate learning rates. **a** log-likelihood versus number of epochs, **b**, **c** the number of βs in PSMC and SMC at each iteration (*blue*) and their mean values (*red*) (Color figure online)

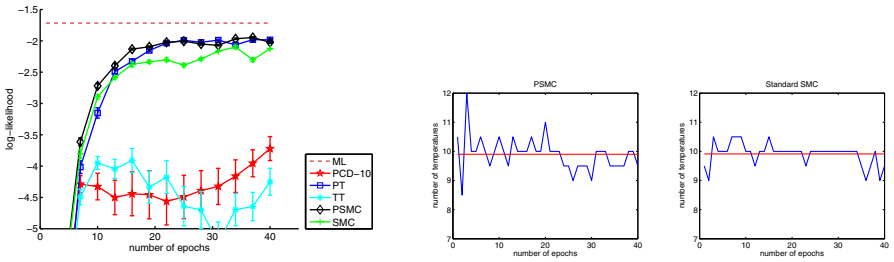

Fig. 4 Performance of the algorithms with large learning rates. **a** log-likelihood versus number of epochs, **b**, **c** the number of βs in PSMC and SMC at each iteration (*blue*) and their mean values (*red*) (Color figure online)

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 learning 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.

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

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	**−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	**−1.891**
	$\eta_t = \frac{1}{10+0.1\times t}$	−25.179	−3.714	−2.118	−4.329	−2.224	**−1.976**
	MNIST						
RBM	Training data	−206.3846	−203.5884	206.2819	−206.9033	−203.3672	**−199.9089**
(784 × 10)	Testing data	−207.7464	−204.6717	206.2819	−208.2452	−204.4852	**−201.0794**
RBM	Training data	−176.3767	−173.0064	−165.2149	−170.9312	−678.6464	**−161.6231**
(784 × 500)	Testing data	−177.0584	−173.4998	−166.1645	−171.6008	−678.7835	**−162.1705**

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 grew 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 Fig. 4a.

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 interconnections only 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 the likelihood function can be written 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 (Figs. 5b, c, 6b, 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 and 159 in PSMC. The reason for this dramatic change in SMC is that all 200 particles initialized from the

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

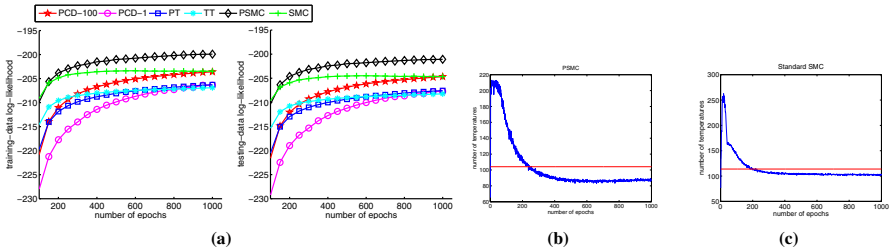

Fig. 5 Performance of the algorithms on the medium-scale RBM. **a** log-likelihood versus number of epochs for both training images (*left*) and testing images (*right*) in the MNIST database, **b**, **c** the number of βs in PSMC and SMC at each iteration (*blue*) and their mean values (*red*) (Color figure online)

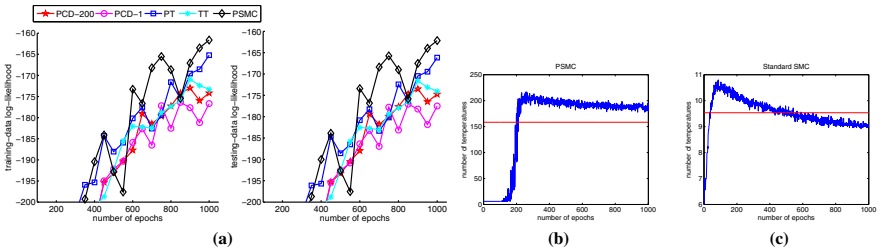

Fig. 6 Performance of the algorithms on the large-scale RBM. **a** log-likelihood versus number of epochs for both training images (*left*) and testing images (*right*) in the MNIST database, **b**, **c** the number of βs in PSMC and SMC at each iteration (*blue*) and their mean values (*red*) (Color figure online)

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 Figs. 5a, 6a). 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:

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 Fig. 6a.

7 Real-world applications

In this section we present some evaluations and comparisons of different learning algorithms on two practical tasks: *multi-label classification* and *image segmentation*. Different from previous experiments where generative models were learned, here we trained discrimina-

tive models. Therefore, two conditional random fields (CRFs) were employed. Generally speaking, let us denote $\mathbf{x}$ as input and $\mathbf{y} \in \mathcal{Y}$ as output, and our target is to learn an UGM:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp(\boldsymbol{\theta}^\top \phi(\mathbf{y}, \mathbf{x}))}{\mathbf{Z}} \tag{32}$$

where the partition function $\mathbf{Z}$ is

$$\mathbf{Z} = \sum_{\mathbf{y} \in \mathcal{Y}} \exp(\boldsymbol{\theta}^\top \phi(\mathbf{y}, \mathbf{x})) \tag{33}$$

where $\phi(\mathbf{y}, \mathbf{x})$ is defined based on task-oriented dependency structure. Note that the partition function $\mathbf{Z}$ is computed by marginalizing out only $\mathbf{y}$ because our interest is a conditional distribution. Six algorithms were implemented: PCD-H, PCD-1, PT, TT, SMC and PSMC. Similar setups were used for all algorithms as the previous section. Learning rate $\eta_t = \frac{1}{10+0.1*t}$ was used and 100 iterations were run. Different from generative models, learning CRFs needs to compute gradient on individual input-output-pair. For each input $\mathbf{x}$, the size of particle set $\{\hat{\mathbf{y}}^{(s)}\}$ is 200. Similar to other supervised learning schemes, a regularization $\frac{1}{2}||\boldsymbol{\theta}||^2$ was added and a trade-off parameters was tuned via k-fold cross-validation ($k = 4$).

It is worth mentioning that better results can be expected in both experiments by running more iterations, using better learning rates or exploiting feature engineering. However, our purpose here is to compare different learning algorithms under the same conditions rather than improving state-of-the-art results in multi-label classification and image segmentation respectively.

7.1 Multi-label classification

In multi-label classification, inter-label dependency is rather critical. Assume that input $\mathbf{x} \in \mathbb{R}^d$ and there are L labels (i.e. $\mathbf{y} \in \{-1, +1\}^L$), here we modeled all pairwise dependencies among L labels, and therefore the constructed conditional random field is

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp\left(\mathbf{y}^\top \mathbf{W}_E \mathbf{y} + \mathbf{y}^\top \mathbf{W}_v \mathbf{x}\right)}{\mathbf{Z}}, \tag{34}$$

where $\mathbf{W}_E \in \mathbb{R}^{L \times L}$ captures pairwise dependencies among L labels while $\mathbf{W}_v \in \mathbb{R}^{L \times d}$ reflects the dependencies between input $\mathbf{x}$ and all individual labels. In the test phase, with learned $\mathbf{W}_E$ and $\mathbf{W}_v$, for a test input $\mathbf{x}^\dagger$, we predict the corresponding $\mathbf{y}^\dagger$ with 100 rounds of Gibbs sampling based on (34).

In our experiment, we used the popular *scene* database (Boutell et al. 2004), where scene images are associated with a few semantic labels. In the database, there are 1121 training instances and 1196 test instances. In total there are 6 labels ($L = 6$) and a 294-dimensional feature vector was extracted from each image ($\mathbf{x} \in \mathbb{R}^{294}$). Readers are referred to Boutell et al. (2004) for more details about the database and feature extraction.

We evaluated the performance of multi-label classification using *precision* (P), *recall* (R), and the *F1* measure (F). For each label, the precision is computed as the ratio of the number of images assigned the label correctly over the total number of images predicted to have the label, while the recall is the number of images assigned the label correctly divided by the number of images that truly have the label. Then precision and recall are averaged across all labels. Finally, the F1 measure is calculated as $F = 2\frac{P \times R}{P+R}$. The results of all six algorithms are presented in Table 2. The average number of temperatures in PSMC is around 5, so PCD-5 was implemented. Also 5 temperatures were used in PT and TT. We can see that PSMC yields

Table 2 A comparison of six learning algorithms on the multi-label classification task

	Precision (%)	Recall (%)	F1 (%)
PCD-1	57.7	59.3	58.5
PCD-5	70.3	72.6	71.4
TT	70.0	67.5	68.7
PT	**72.2**	77.1	74.6
SMC	71.7	75.1	73.4
PSMC	71.9	**78.5**	**75.1**

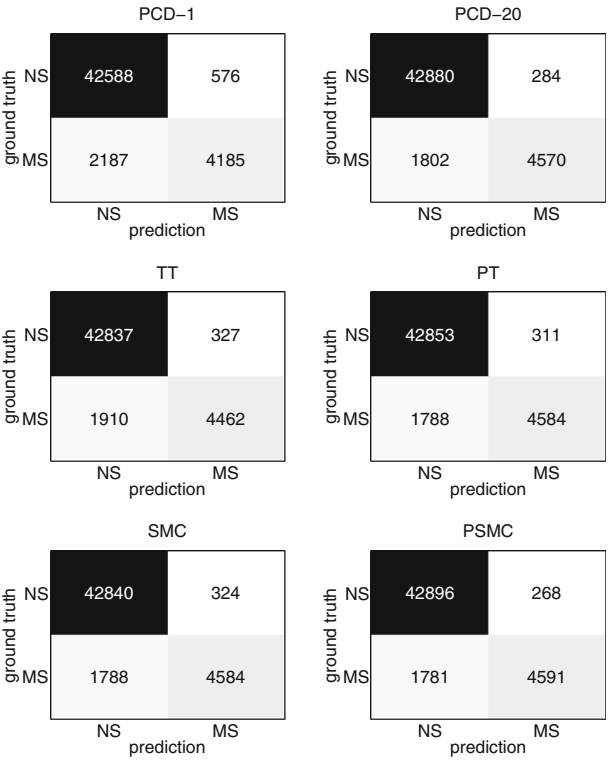

Fig. 7 Confusion matrices of binary segmentation by six algorithms

the best F1 measure of 75.1, followed by PT and SMC with 74.6 and 73.4 respectively. The results of PCD-5 and TT are relative worse, while PCD-1 is the worst.

7.2 Image segmentation

Image segmentation essentially is a task to predict the semantic labels of all image pixels or blocks. Inter-label dependencies within a neighborhood are usually exploited in image segmentation. For instance, by dividing an image into equal-size and non-overlapping blocks, the label of a block depends not only on the appearance of the block, but also on the labels of its neighboring blocks. For simplicity, here we only consider binary labels. In addition,

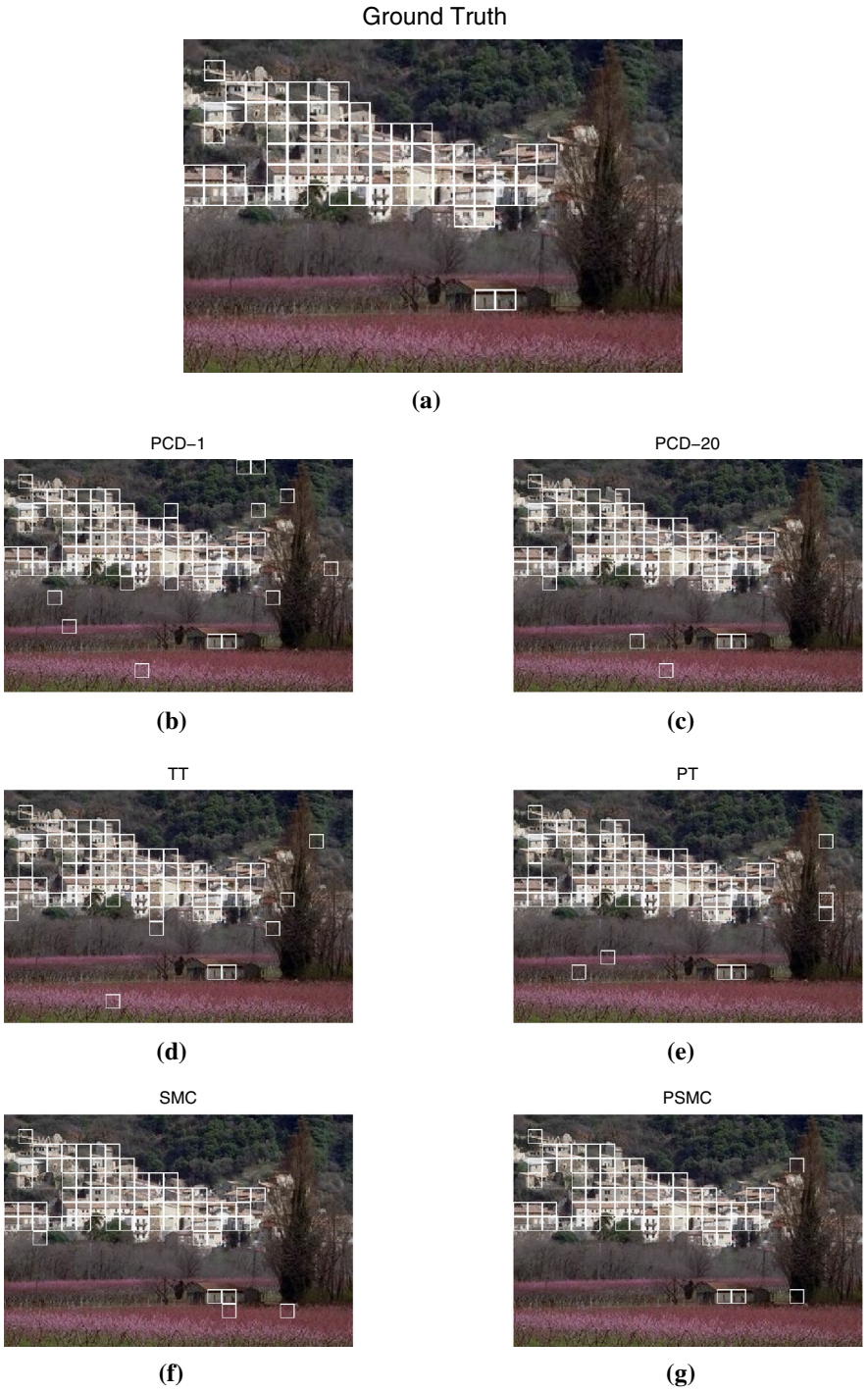

Fig. 8 An example image and corresponding segmentations by six algorithms. Regions within *white boxes* are predicted as "man-made structures" while the remaining are "nature structures"

we assume that blocks and inter-label dependencies are position invariant. Therefore, a conditional random filed can be constructed as

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp\left(\sum_{u,v\in E} y_u W_e y_v + \sum_{v\in V} y_v \mathbf{w}_v^\top \mathbf{x}_v\right)}{\mathbf{Z}}, \tag{35}$$

where $y_v \in \{-1, +1\}$, E denotes the set of all edges connecting neighboring blocks, $W_e \in \mathbb{R}$ encodes the dependency between neighboring labels, V denotes the set of all block's labels, and $\mathbf{w}_v \in \mathbb{R}^{d\times 1}$ encodes the dependency between block label and its appearance which is represented by a d-dimensional feature $\mathbf{x}_v \in \mathbb{R}^d$. Similarly to the multi-label classification experiment, desired labels are predicted via 100 rounds of Gibbs sampling in the test phase.

In our experiment, we used the binary segmentation database from Kumar and Hebert (2003), where each image is divided into non-overlapping blocks of size 16×16 and each block is annotated with either "man-made structure" (MS) or "nature structure" (NS). Overall, there are 108 training images and 129 test images. The training set contains 3004 MS blocks and 36,269 NS blocks, while the test set contains 6372 MS blocks and 43,164 NS blocks. Each block's appearance is represented by a 3-dimensional feature which includes the mean of gradient magnitude, the 'spikeness' of the count-weighted histogram of gradient orientations, and the angle between the most frequent orientation and the second most frequent orientation. The feature was designed to fit this specific application. More explanation of the database and its feature design can be found in Kumar and Hebert (2003).

We found that the average number of temperatures in PSMC is 20; therefore PCD-20 was run and 20 temperatures were used in TT and PT. We quantify the segmentation performance of six algorithms with confusion matrices, which are presented in Fig. 7. We can see that PSMC outperforms all others (by checking the diagonal entries of confusion matrices). For qualitative comparison, an example image and corresponding segmentations are shown in Fig. 8. It can be seen that the segmentation by PSMC is closer to the ground truth compared to the others.

8 Conclusion

A SMC interpretation framework of learning UGMs was presented, within which two main challenges of the learning task were disclosed as well: large learning rate and high model complexity. 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). In addition, as we pointed out earlier, bring learning UGMs under SMC thinking makes it possible to leverage research outcomes from SMC community, which suggests many possible directions for future work.

Another research brunch of approximately learning UGMs or estimating partition function is variational principle (Wainwright and Jordan 2008). Different from sampling-based method, variational methods approximate log partition function with *mean filed theory free energy* or *Bethe free enegry*. These methods are usually preferred to sampling-based methods because of better efficiency. In particular, dual decomposition techniques (Schwing et al. 2011) can make computation parallel. However, the applicabilities of variational methods

are rather limited, *e.g.* they are only guaranteed to work well for tree-structured graphs. Even though some progresses were made to handle densely-connected graphs, they are still restricted to the higher-order potentials of certain specific form, *e.g.* a function of the cardinality (Li et al. 2013) or piece-wise linear potentials (Kohli et al. 2009). By contrast, sampling-based methods are more general since they can work on arbitrary UGMs given enough computation resources. In addition, with the development of parallelization of sampling methods (Vergé et al. 2015), it is also possible to employ distributed computing to boost the efficiency of sampling-based methods. After all, as two research streams of approximate learning or inference, so far there can be no quick conclusion on which one is superior to the other. Instead, it will be more interesting and meaningful to investigate the connection between these two directions, which was lightly touched in Yuille (2011).

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

Asuncion, A.U., Liu, Q., Ihler, A.T., & Smyth, P. (2010). Particle filtered MCMC-MLE with connections to contrastive divergence. In *International conference on machine learning (ICML)*.

Boutell, M. R., Luo, J., Shen, X., & Brown, C. M. (2004). Learning multi-label scene classification. *Pattern Recognition*, *37*(9), 1757–1771.

Carreira-Perpinan, M.A., & Hinton, G.E. (2005). On contrastive divergence learning. In *International conference on artificial intelligence and statistics*.

Chopin, N. (2002). A sequential particle filter method for static models. *Biometrika*, *89*(3), 539–552.

Del Moral, P., Doucet, A., & Jasra, A. (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *68*(3), 411–436.

Desjardins, G., Courville, A., Bengio, Y., Vincent, P., & Delalleau, O. (2010). Tempered Markov Chain Monte Carlo for training of restricted Boltzmann machines. In *AISTATS*.

Dragomir, S.S. (1999–2000). A converse result for jensens discrete inequality via gruss inequality and applications in information theory. *Analele Univ. Oradea. Fasc. Math.*, (7), 179–189.

Geyer, C.J. (1991). *Markov Chain Monte Carlo maximum likelihood.* In Compuuting science and statistics: Proceedings of the 23rd symposium on the interface.

Handcock, M. S., Hunter, D. R., Butts, C. T., Goodreau, S. M., & Morris, M. (2008). statnet: Software tools for the representation, visualization, analysis and simulation of network data. *Journal of Statistical Software*, *24*(1), 1548–7660.

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

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

Kohli, P., Ladický, L., & Torr, P. H. (2009). Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, *82*(3), 302–324.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. Cambridge: MIT Press.

Kong, A., Liu, J. S., & Wong, W. H. (1994). Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, *89*(425), 278–288.

Kumar, S., & Hebert, M. (2003). Man-made structure detection in natural images using a causal multiscale random field. In *IEEE international conference on computer vision and pattern recognition (CVPR)*, pp. 119–126.

Li, T., Sun, S., Sattar, T. P., & Corchado, J. M. (2014). Fight sample degeneracy and impoverishment in particle filters: A review of intelligent approaches. *Expert Systems with Applications*, *41*(8), 3944–3954.

Li, Y., Tarlow, D., & Zemel, R. (2013). Exploring compositional high order pattern potentials for structured output learning. In *Proceedings of the 2013 IEEE conference on computer vision and pattern recognition*, pp. 49–56.

Neal, R. (1994). Sampling from multimodal distributions using tempered transitions. *Statistics and Computing*, *6*, 353–366.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, *22*, 400–407.

Salakhutdinov, R. (2010). Learning in markov random fields using tempered transitions. In *Advances in neural information processing systems (NIPS)*.

Salakhutdinov, R., & Murray, I. (2008). On the quantitative analysis of deep belief networks. In *International conference on machine learning (ICML)*.

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

Schwing, A., Hazan, T., Pollefeys, M., Urtasun, R. (2011). Distributed message passing for large scale graphical models. In *Computer vision and pattern recognition (CVPR), 2011 IEEE conference on*, pp. 1833–1840, June 2011.

Tieleman, T. (2008). Training restricted Boltzmann machines using approximations to the likelihood gradient. In *International conference on machine learning (ICML)*, pp. 1064–1071.

Tieleman, T., & Hinton, G.E. (2009). Using fast weights to improve persistent contrastive divergence. In *International conference on machine learning (ICML)* (pp. 1033–1040). NY, USA: ACM New York.

Vergé, Christelle, Dubarry, Cyrille, Del Moral, Pierre, & Moulines, Eric. (2015). On parallel implementation of sequential monte carlo methods: The island particle model. *Statistics and Computing*, *25*(2), 243–260.

Wainwright, M.J., & Jordan, M.I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, *1*(1–2), 1–305.

Younes, L. (1988). Estimation and annealing for gibbsian fields. *Annales de l'institut Henri Poincar (B) Probabilits et Statistiques*, *24*(2), 269–294.

Yuille, A. L. (2011). Belief propagation, mean field, and bethe approximations. In P. Kohli, A. Blake, & C. Rother (Eds.), *Advances in Markov random fields for vision and image processing*. Cambridge: MIT Press.