Learning Sigmoid Belief Networks via Monte Carlo Expectation Maximization

Zhao Song[†] **Ricardo Henao**[†] **David Carlson**[‡] **Lawrence Carin**[†]
[†]Department of Electrical and Computer Engineering, Duke University
[‡]Department of Statistics and Grossman Center for Statistics of Mind, Columbia University

Abstract

Belief networks are commonly used generative models of data, but require expensive posterior estimation to train and test the model. Learning typically proceeds by posterior sampling, variational approximations, or recognition networks, combined with stochastic optimization. We propose using an online Monte Carlo expectation-maximization (MCEM) algorithm to learn the maximum *a posteriori* (MAP) estimator of the generative model or optimize the variational lower bound of a recognition network. The E-step in this algorithm requires posterior samples, which are already generated in current learning schema. For the M-step, we augment with Pólya-Gamma (PG) random variables to give an analytic updating scheme. We show relationships to standard learning approaches by deriving stochastic gradient ascent in the MCEM framework. We apply the proposed methods to both binary and count data. Experimental results show that MCEM improves the convergence speed and often improves hold-out performance over existing learning methods. Our approach is readily generalized to other recognition networks.

1 Introduction

The sigmoid belief network (SBN) (Neal, 1992) has drawn increasing attention in recent years as an essential component of deep belief networks (DBN) (Hinton et al., 2006) and for its use in recognition networks (Mnih and Gregor, 2014). In contrast to undirected graphical models, such as the restricted Boltzmann machine (RBM), the SBN is a directed graphical model, and efficiently generates data in a top-down manner. Unfortunately, the conditional distribution on the latent variables in the SBN lacks a closed form (Pearl, 1988; Hinton, 2010). This results in a computationally expensive inference procedures, so training an SBN is a difficult task (Gan et al., 2015b).

In Neal (1992), a gradient-ascent algorithm was proposed to learn the SBN parameters. The gradient was estimated via Monte Carlo integration. Alternatively, Saul et al. (1996) proposed a mean-field variational scheme for the hidden variables to approximately learn the model. These methods are not scalable to large datasets (Neal, 1992) nor easy to derive and implement (Saul et al., 1996). Recently, Mnih and Gregor (2014) proposed a neural variational inference and learning (NVIL) algorithm, which developed a parametric *recognition* model that transforms observed data into a variational posterior approximation. NVIL admits fast sampling of the hidden variables, and proposes several techniques for variance reduction on noisy gradient estimates. Learning proceeds through stochastic gradient descent (SGD). Gan et al. (2015b) proposed a new Bayesian framework for SBN inference, with shrinkage priors on the weight parameters. Inference proceeds by Gibbs sampling or variational Bayesian (VB) techniques, but these methods suffer from the same computational difficulties as traditional learning methods.

In this paper we develop an MCEM algorithm (Wei and Tanner, 1990) to learn the parameters of the SBN. MCEM approximates the E-step of the classical expectation-maximization (EM) algorithm (Dempster et al., 1977) by Monte Carlo samples of the *missing data*, and has been shown to achieve good empirical performance (Chan and Ledolter, 1995; Levine and Casella, 2001). We introduce auxiliary Pólya-Gamma variables (Polson et al., 2013) as additional missing data to efficiently optimize the subsequent M-step. Given the auxiliary variables, the *expected log complete-data log-likelihood*, *i.e.*, the $\mathcal{Q}$ function

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 41.

(Dempster et al., 1977), has a block-quadratic structure, and is efficiently updated. This quadratic structure was previously observed in logistic regression models (Polson and Scott, 2011; Scott and Sun, 2013).

Compared with previous learning methods, our MCEM algorithm has the following advantages: (*i*) Unlike mean-field VB approaches (Gan et al., 2015b), our approach optimizes the true objective function. This allows the model to more accurately capture dependencies between hidden variables and give better hold-out performance. (*ii*) Unlike gradient-based approaches (Neal, 1992; Mnih and Gregor, 2014), MCEM does not require step size tuning and has analytical updates for the M-step. Empirically, MCEM converges to a local optima faster than SGD. (*iii*) The analytic updates can readily admit many priors on the weights through variable augmentation.

Furthermore, we propose Rec-MCEM to combine NVIL and MCEM. Rec-MCEM utilizes the efficient samples from NVIL and replaces the gradient ascent step on the generative model with an EM algorithm to estimate model parameters. Although EM has higher per-iteration computational cost than gradient ascent, each iteration is more effective. We empirically show faster convergence and improved local maxima.

2 Sigmoid Belief Network

The SBN has been successfully employed as a building block in deep belief networks and in applications with binary (Neal, 1992), counts (Gan et al., 2015a; Mnih and Gregor, 2014), and real-valued (Frey, 1997) data. In this paper, we focus on binary and count data.

Notationally, we refer to vectors in bold lower-case letters (*i.e.,* $\boldsymbol{x}$), and matrices as upper-case letters (*i.e.,* W). $W_{i\cdot}$ represents the i^{th} row of the matrix W.

For an SBN, the probabilistic relationship between adjacent layers has a bi-partite structure[1]. For an SBN with M nodes in the visible layer and J_l nodes in the lth hidden layer, where $l = 1, \ldots, L$, and L is the number of hidden layers, the generative model is

$$p(\boldsymbol{h}^{(L)}|\boldsymbol{b}) = \prod_{i=1}^{J_L} \left[\exp(b_i)\right]^{h_i^{(L)}} / \left[1+\exp(b_i)\right]$$

$$p(\boldsymbol{h}^{(l)}|\boldsymbol{h}^{(l+1)}) = \prod_{i=1}^{J_l} \left[\exp(\psi_i^{(l+1)})\right]^{h_i^{(l)}} / \left[1+\exp(\psi_i^{(l+1)})\right]$$

$$p(\boldsymbol{v}|\boldsymbol{h}^{(1)}) = \prod_{i=1}^{M} \left[\exp(\psi_i^{(1)})\right]^{v_i} / \left[1+\exp(\psi_i^{(1)})\right]^{u_i} \quad (1)$$

where $\psi_i^{(l)} = W_{i\cdot}^{(l)}\boldsymbol{h}^{(l)} + c_i^{(l)}$, $h_i^{(l)} \in \{0,1\}$ denotes the ith binary unit in the lth hidden layer, $v_i \in \mathbb{Z}^+ \cup \{0\}$ is the ith input in the visible layer, $W^{(l)} \in \mathbb{R}^{J_{(l-1)} \times J_l}$, $\forall\, l = 1, \ldots, L$, represents the weight matrix between the $(l-1)$th layer and the lth layer. We set $M = J_0$ for notational simplicity. $\boldsymbol{b} \in \mathbb{R}^{J_L}$ is the bias in the top layer, and $\boldsymbol{c}^{(l)} \in \mathbb{R}^{J_l}$, $\forall l = 0, \ldots, L-1$ corresponds to the biases in the remaining layers.

The likelihood, $p(\boldsymbol{v}|\boldsymbol{h}^{(1)})$, in (1) can be adapted to different types of data (Scott and Sun, 2013): for binary data, a Bernoulli likelihood is employed, thus $u_i = 1$; for count data, a negative binomial (NB) likelihood is employed, thus $u_i = v_i + r$, where r is the dispersion parameter of the NB distribution. The NB distribution is equivalent to a gamma-Poisson distribution (Zhou et al., 2012), a generalization of the Poisson distribution that models over-dispersed data by separately controlling both the mean and variance of counts.

Let $\boldsymbol{\theta} = \{W^{(l)}, \boldsymbol{b}, \boldsymbol{c}^{(l)}\}$, $\forall\, l = 1, \ldots, L$ be the parameters we aim to learn given the observed data $\{\boldsymbol{v}_n\}_{n=1}^N$, our goal is to derive an MAP estimator[2],

$$\begin{aligned}\widehat{\boldsymbol{\theta}}_{\text{MAP}} &= \arg\max_{\boldsymbol{\theta}} \textstyle\sum_{n=1}^N \ln p\big(\boldsymbol{\theta}|\boldsymbol{v}_n\big) \\ &= \arg\max_{\boldsymbol{\theta}} \textstyle\sum_{n=1}^N \ln \sum_{\boldsymbol{h}} p\big(\boldsymbol{\theta}, \boldsymbol{h}|\boldsymbol{v}_n\big)\,.\end{aligned}$$

Marginalizing out $\boldsymbol{h}$ from $p\big(\boldsymbol{\theta}, \boldsymbol{h}|\boldsymbol{v}_n\big)$ has exponential complexity in the tree-width, $\max_\ell J_\ell$. Instead, we propose the MCEM algorithm for MAP estimation.

3 Monte Carlo Expectation Maximization

The EM algorithm is a well-known method for MAP estimation. We start by providing the standard update rule (Dempster et al., 1977). For clarity, we omit the bias terms, and we begin by examining only a single hidden layer. We drop the layer superscript for simplicity. We initially ignore prior terms.

$$\text{E-step}: \quad \mathcal{Q}(W|W^{(t)}) = \sum_{n=1}^{N} \mathbb{E}_{p(\boldsymbol{h}_n|\boldsymbol{v}_n, W^{(t)})}[\log p(\boldsymbol{v}_n, \boldsymbol{h}_n|W)] \quad (2a)$$

$$\text{M-step}: \quad W^{(t+1)} = \arg\max_{W} \mathcal{Q}(W|W^{(t)}). \quad (2b)$$

Evaluating the expectation in (2a) is exponentially expensive, $\mathcal{O}(NJ_0 2^{J_1})$, so this is infeasible for all but the smallest models. An alternative approach to this problem is to use the MCEM framework, where the E-step

[1] This does not hold for the autoregressive network in Gregor et al. (2014); Gan et al. (2015b), which is beyond the scope of this paper.

[2] The maximum-likelihood (ML) estimator can be obtained as a special case, by letting the prior be constant over the domain of the parameters, $\boldsymbol{\theta}$.

is approximated by collecting K posterior samples per data point from $\boldsymbol{h}_n^{(k)} \sim p(\boldsymbol{h}_n|\boldsymbol{v}_n, W^{(t)})$. The M-step is then equivalent to

$$W^{(t+1)} = \arg\max_W \sum_{n=1}^N \mathcal{Q}_n(W|W^{(t)}) \tag{3a}$$

with

$$\mathcal{Q}_n(W|W^{(t)}) = \frac{1}{K}\sum_{k=1}^K \Big[\boldsymbol{v}_n^T W \boldsymbol{h}_n^{(k)} - \sum_m \log\big(1+\exp([W\boldsymbol{h}_n^{(k)}]_m)\big)\Big], \tag{3b}$$

where $[\boldsymbol{x}]_m$ denotes the mth element of $\boldsymbol{x}$.

Note that (3b) does not have an analytic solution, but is a concave function. We can use standard gradient ascent to maximize $\tilde{\mathcal{Q}}(W|W^{(t)}) = \sum_{n=1}^N \mathcal{Q}_n(W|W^{(t)})$, with gradients

$$\nabla_W \mathcal{Q}_n(W|W^{(t)}) = \frac{1}{K}\sum_{k=1}^K \Big(\boldsymbol{v}_n - \sigma(W\boldsymbol{h}_n^{(k)})\Big)(\boldsymbol{h}_n^{(k)})^T, \tag{4}$$

where $\sigma(x) = 1/(1+\exp(-x))$ is the sigmoid function. We show in Section 3.1 that we can easily relate this view of MCEM to the gradient ascent approach of Neal (1992). In Section 3.2 we demonstrate that augmenting $\mathcal{Q}$ with Pólya-Gamma (PG) random variables leads to an analytic updating scheme that vastly improves the convergence speed of the M-step. We empirically demonstrate that the PG is a better optimization scheme compared to gradient ascent, and roughly equivalent to the spectral ascent approach of Carlson et al. (2015a).

3.1 Connection between MCEM and SGD

SGD has been frequently employed to learn an SBN (Neal, 1992; Mnih and Gregor, 2014; Saul et al., 1996)[3]. This gradient ascent procedure requires

$$\nabla \log p(\boldsymbol{v}_n|W) = \mathbb{E}_{p(\boldsymbol{h}_n|\boldsymbol{v}_n,W)}\left[(\boldsymbol{v}_n - \sigma(W\boldsymbol{h}_n))\,\boldsymbol{h}_n^T\right]. \tag{5}$$

Neal (1992) proposed to estimate this via Monte Carlo integration. Given K posterior samples per data point from $\boldsymbol{h}_n^{(k)} \sim p(\boldsymbol{h}_n|\boldsymbol{v}_n, W^{(t)})$, (5) is approximated as

$$\nabla \log p(\boldsymbol{v}_n|W) \simeq \frac{1}{K}\sum_{k=1}^K \Big(\boldsymbol{v}_n - \sigma(W\boldsymbol{h}_n^{(k)})\Big)(\boldsymbol{h}_n^{(k)})^T. \tag{6}$$

Note that (4) and (6) are equivalent for the same samples, $\boldsymbol{h}_n^{(k)}$. Thus, we can view the gradient ascent learning scheme for SBNs is the MCEM algorithm with a *single* gradient update on the M-step.

Drawing posterior samples from $\boldsymbol{h}_n^{(k)} \sim p(\boldsymbol{h}_n|\boldsymbol{v}_n, W^{(t)})$ is computationally expensive, $\mathcal{O}(NKJ_0J_1^2)$ via Gibbs sampling. The gradient formulation and update is $\mathcal{O}(NKJ_0J_1)$, or $\simeq J_1$ times faster. This motivates examining using multiple gradient steps, or other optimization procedures that more effectively optimize the $\mathcal{Q}$ function.

3.2 Augmenting Pólya-Gamma Random Variables

We develop an auxiliary-variable scheme to optimize the M-step. The idea is that by augmenting the $\tilde{\mathcal{Q}}$ function, we can use an *inner loop* of a distinct EM algorithm to optimize $\tilde{\mathcal{Q}}$.

We begin by introducing the following identity (Polson et al., 2013)

$$\frac{\big[\exp(\psi)\big]^v}{\big[1+\exp(\psi)\big]^u} = 2^{-u}\exp(\kappa\psi)\int_0^\infty \exp(-\frac{\omega\psi^2}{2})p(\omega)\,d\omega, \tag{7}$$

where $\kappa = v - \frac{u}{2}$ and $\omega \sim \text{PG}\ (u, 0)$. In our model, ψ represents an entry of $W\boldsymbol{h}_n^{(k)}$. The identity in (7) has been successfully employed to construct data augmentation schemes for Gibbs samplers (Polson et al., 2013), variational Bayes (Gan et al., 2015b), and EM algorithms (Scott and Sun, 2013).

The key property of (7) is that ψ has a Gaussian distribution conditioned on ω. Gaussian distributions naturally lend themselves to learning linear models. In our SBN model, we define auxiliary random variable matrix, $\boldsymbol{\omega} \in \mathbb{R}^{N\times J_0}$.

Often, we would like to obtain the MAP solution, or a penalized likelihood solution. We choose to use a Laplace prior on W to match Gan et al. (2015b). This is defined as $p(W_{i,j} \mid \lambda) = \frac{\sqrt{\lambda}}{2}\exp(-\sqrt{\lambda}W_{i,j})$. The Laplace prior is decomposed into a Gaussian scale-mixture (Figueiredo, 2003; Polson and Scott, 2011):

$$\begin{aligned} W_{i,j}|\tau_{i,j} &\sim \mathcal{N}\big(W_{i,j}\,|\,0,\tau_{i,j}\big), \\ \tau_{i,j}|\lambda &\sim \frac{\lambda}{2}\exp\big(-\frac{\lambda\tau_{i,j}}{2}\big), \end{aligned}$$

Both the PG and Laplace augmentation schemes have a closed-form representation on the conditional posterior expectation of the augmented variables, *i.e.*, $\mathbb{E}[\omega_{n,i}|-]$ and $\mathbb{E}[\tau_{i,J_l}^{-1}|-]$, where $\mathbb{E}[x|-]$ represents the conditional expectation of random variable x given all the other variables. Consequently, the E-step in this

[3] Saul et al. (1996) and Mnih and Gregor (2014) use a variational distribution on the posterior, but sample to estimate the gradient over W.

inner EM algorithm is analytic. The corresponding $\mathcal{Q}$ function is quadratic with respect to parameters, $\boldsymbol{\theta}$.

We provide the detailed derivation and forms for the intermediate variables in the Supplemental Materials, and summarize this inner EM algorithm as follows.

E-step: Compute

$$\widehat{\omega}_{n,i}^{(t+1)} = \mathbb{E}[\omega_{n,i}|-] = \frac{u_{n,i}}{2\psi_{n,i}^{(t)}} \tanh\left(\frac{\psi_{n,i}^{(t)}}{2}\right),$$
$$\widehat{\tau}_{i,J_0}^{(t+1)} = \mathbb{E}[\tau_{i,J_0}^{-1}|-] = \lambda \,/\, \left|W_{i,J_0}^{(t)}\right|.$$

M-step: Update W as

$$\left[W_{i\cdot}^{(t+1)}\right]^T = \left[X_i^{(t+1)} + \Phi_i^{(t+1)}\right]^{-1} \boldsymbol{\eta}_i^{(t)},$$

where $X_i^{(t+1)} = \sum_{n=1}^N \widehat{\omega}_{n,i}^{(t+1)} \boldsymbol{h}_n \boldsymbol{h}_n^T$, $\boldsymbol{\eta}_i^{(t)} = \sum_{n=1}^N \kappa_{n,i}^{(t)} \boldsymbol{h}_n$, and $\Phi_i^{(t+1)}$ is a diagonal matrix with elements $\left(\widehat{\tau}_{i,1}^{(t+1)}, \ldots, \widehat{\tau}_{i,J_0}^{(t+1)}\right)$.

This M-step admits a closed-form solution. In contrast, gradient ascent approaches must tune the step size. In terms of computational complexity, the update here is $\mathcal{O}(NKJ_0J_1 + J_0J_1^3)$ while a gradient update is is $\mathcal{O}(NKJ_0J_1)$. When $J_0J_1^3 < NKJ_0J_1$, this M-update will add only a small amount of overhead compared to gradient estimation.

3.3 Empirical Convergence Comparison

Both gradient ascent and PG are guaranteed to converge to the global optimum because $-\tilde{\mathcal{Q}}$ is convex. As well, on this objective function, both algorithms have an $\mathcal{O}(1/R)$ convergence rate, with R being the number of iterations (Boyd and Vandenberghe, 2004)[4]. Although the PG procedure adds an additional computational cost, it empirically improves optimization.

To provide evidence that the PG procedure is an effective optimization procedure that captures second-order information, we present empirical results on timing in Figure 1 for a simple SBN with $J_0 = 784$, $J_1 = 50$, and $N = 300$ on the MNIST dataset (Salakhutdinov and Murray, 2008). We show approaches using gradient ascent, spectral ascent (Carlson et al., 2015a), and our PG-update scheme, given the same fixed hidden variable, $\boldsymbol{h}$. The step-sizes for gradient ascent and spectral ascent are both set to $4/(NJ)$, which gives theoretically optimal worst-case convergence, as shown in Carlson et al. (2016). For further comparison, we add the curve of gradient ascent with a hand-tuned step-size. This tuned step-size improves performance, but the optimal setting changes with different network shapes and data, and is not feasible in practice. To emphasize the difference in the methods, we mark the first iteration for each method. While the PG update takes considerably longer, its first iteration yields much higher improvement than the corresponding first iteration of gradient ascent or spectral ascent. Both spectral ascent and the PG offer similar performance per time step, and both offer significant improvements over gradient descent. However, the PG method does not require step-size tuning. We also show in the Supplemental Materials that when given enough time, all methods achieve the same final maxima, which is consistent with the fact that $\tilde{\mathcal{Q}}$ is a concave function.

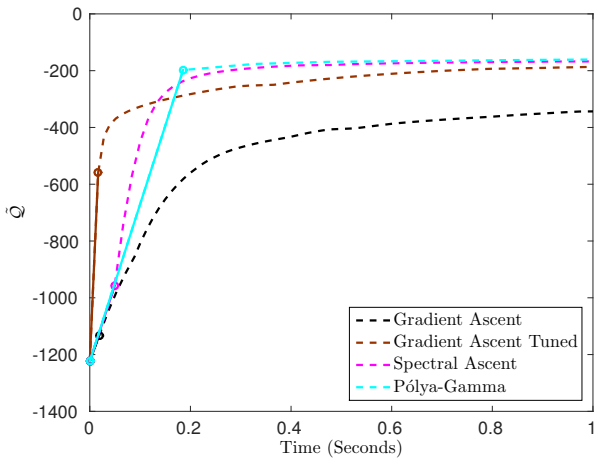


Figure 1: The value of $\tilde{\mathcal{Q}}$ as a function of running time for different optimization schemes.

The improved performance of PG over gradient ascent is supported by second-order information from the PG augmentation, while SGD is a first-order method. The trade-off between computation and efficacy of iterations is explored by Scott and Sun (2013). Comparing PG with spectral ascent is difficult, because spectral ascent is based on a non-Euclidean first order method (Carlson et al., 2015a).

There is a trade-off between how many iterations we use to optimize the $\mathcal{Q}$ function versus how often we refresh the Gibbs sampler. We discuss this in Section 5. We reiterate that a major advantage of the MCEM framework is that the SBN now has a sub-problem of a concave maximization problem. While we advocate for the PG or spectral assent approach, there are numerous convex minimization algorithms that can be straightforwardly applied. Furthermore, we note that approximations to the spectral ascent steps exist (Carlson et al., 2015b), and it may be possible to speed the PG steps.

[4]The PG is a Euclidean majorization function on $-\mathcal{Q}$ (Polson et al., 2013). The PG procedure is a majorization-minimization with a Euclidean norm, a standard case discussed in Section 9.4 of Boyd and Vandenberghe (2004).

3.4 Online MCEM

For large N, the presented *batch* MCEM algorithm becomes computationally intractable. We propose adapting Online EM (Liang and Klein, 2009) to derive an Online MCEM algorithm to handle large datasets.

The online EM algorithm stores sufficient statistics, and updates them via interpolation when new data arrives. This scheme can be described as follows. Let N_{mini} be the size of the minibatch. After sampling the mth minibatch, as in Liang and Klein (2009), each sufficient statistic is updated with $\boldsymbol{\phi}_m = (1 - \gamma_m)\boldsymbol{\phi}_{m-1} + \gamma_m \hat{\boldsymbol{\phi}}_m$, where $\hat{\boldsymbol{\phi}}_m$ are the sufficient statistics estimated on this minibatch. We set the step-size for the mth mini-batch to $\gamma_m = (m + 2)^{-\alpha}$, as suggested by Liang and Klein (2009) to guarantee convergence. With these settings, we derive an online MCEM algorithm for MAP estimates. Details are provided in the Supplemental Materials.

4 The Rec-MCEM Algorithm

Gibbs sampling hidden units in an SBN is computationally expensive and does not scale to large datasets or networks. A recent, popular approach to addressing this problem is to use a recognition model, or a variational auto-encoder, for inference in a directed graphical model (Kingma and Welling, 2013). The NVIL algorithm (Mnih and Gregor, 2014) extends the variational auto-encoder to discrete variables, and uses the SBN for the generative model. NVIL also introduces variance reduction techniques to obtain more stable gradients.

As in standard variational approaches (Saul et al., 1996; Gan et al., 2015b), the true posterior distribution $p(\boldsymbol{h}_n|\boldsymbol{v}_n, W)$ is replaced by a variational distribution $q(\boldsymbol{h}_n|\boldsymbol{v}_n, W) = \prod_{j=1}^{J_1} \text{Bern}(h_{nj}|\eta_{nj})$. In traditional variational approaches, $\boldsymbol{\eta}_n$ is learned independently for each data point, which is the same computational complexity as the Gibbs sampler. The key idea of variational autoencoder is to use a parametric, *recognition* model to estimate $\boldsymbol{\eta}_n$. A simple example is the mapping $\boldsymbol{\eta}_n = \sigma(A\boldsymbol{v}_n)$. This creates a variational distribution that is quick to estimate and sample from, but adds additional variables that need to be estimated in the learning phase.

NVIL uses stochastic gradient ascent to learn the model. This approach draws samples, $h_n^{(k)} \sim q(\boldsymbol{h}_n|\boldsymbol{v}_n, W)$, to approximate the gradients on the generative model (SBN), and the recognition model parameters. The generative model gradient is the same as (6), where samples come from the variational distribution. This sampling procedure to estimate gradients is identical to both Saul et al. (1996) and Mnih and Gregor (2014).

Algorithm 1 Online Rec-MCEM algorithm for MAP estimation.

Input: initialized parameters $A^{(0)}$ and $\boldsymbol{\theta}^{(0)}$, total number of mini-batches *nbatch*.
repeat
  for $k = 1$ to *nbatch* **do**
    Read the kth mini-batch data, $[\boldsymbol{v}_{(k-1)*N_{mini}+1}, \ldots, \boldsymbol{v}_{k*N_{mini}}]$.
    Sample hidden units from the recognition model, $[\boldsymbol{h}_{(k-1)*N_{mini}+1}, \ldots, \boldsymbol{h}_{k*N_{mini}}]$
    Update parameters of recognition model, A, by gradient ascent with using variance reduction techniques in (Mnih and Gregor, 2014).
    Update parameters of generative model, $\boldsymbol{\theta}$, by MCEM and update its sufficient statistics.
  end for
until Convergence.

Replacing the posterior distribution in the EM algorithm with a variational distribution is well-established (*i.e.,* Variational EM (Beal and Ghahramani, 2003)). We can construct a Variational MCEM algorithm and perform improved optimization on the M-step as in Section 3. Because sampling is significantly cheaper from the recognition model, $\mathcal{O}(NJ_1(J_0 + K))$, using the PG optimization scheme may non-trivially increase the per-iteration cost. However, as demonstrated in Figure 1, PG steps improves convergence of the M-step over gradient ascent, despite the slower per-iteration time. This is supported empirically in Section 5. Improving the optimization will improve the convergence of the algorithm.

The MCEM approach does not address recognition model learning, which is updated as in Mnih and Gregor (2014) by using variance reduction techniques to improve gradient estimation. We refer to our new method as Recognition-MCEM (Rec-MCEM), to combine the advantages of both NVIL and MCEM algorithms: fast sampling as inherited in the recognition model of NVIL algorithm and improved convergence of MCEM algorithm.

The outline of the online Rec-MCEM algorithm is provided in Algorithm 1. In contrast to NVIL, Rec-MCEM learns the parameters of the generative model via the EM algorithm introduced in Section 3. This has been shown in Section 3 to improve $\mathcal{Q}$ function convergence. We note that the extension of the MCEM algorithm to the recognition model is very efficient to implement, because the sampling procedure *already exists in the current recognition model learning methods*. This implies that MCEM is a flexible

Table 1: Average test log-likelihood computed via AIS (in nats), on two datasets. "Dim" represents the number of units in each hidden layer.

Method	Dim	MNIST	OCR
SGD	200	-118.85	-49.67
Gibbs		-111.79	-49.45
VB		-106.37	-45.20
NVIL		-104.76	-40.35
MCEM-ML		-99.08	-37.58
MCEM-MAP		**−98.88**	**−37.34**
VB	200-200	-103.78	-43.43
MCEM-ML		-98.90	-35.17
MCEM-MAP		-98.37	**−34.89**
NVIL		**−95.15**	-35.38

framework to combine with other methods, and is easily applied to other recognition models.

5 Experiments

We test our proposed MCEM and Rec-MCEM algorithms on binary and count data. Both ML and MAP versions are implemented. In their online variants, we set the learning rate to $\alpha = 0.501$, which is consistent with Liang and Klein (2009); Scott and Sun (2013), to guarantee convergence. We set the number of samples in the Monte Carlo E-step to $K = 10$, which is observed to work well in our experiments. To choose $\lambda^{(l)}$ in MCEM-MAP, we test over the grid: $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ with a validation set, and select the best one. Elements of $\boldsymbol{\theta}$ are initialized with $\mathcal{N}(0, 0.01)$, for both MCEM and Rec-MCEM. All code is written in MATLAB and tested on a Linux machine with 3.1GHz CPU and 8GB RAM.

We used different numbers of inner EM updates. The results suggest that increasing the number from 1 does not bring obvious improvement for the likelihood in terms of time. Note that a single PG update is equivalent to many GD updates. Therefore, we fix the number of inner EM updates to be 1. In this case, our MCEM reduces to a generalized EM (GEM) algorithm (McLachlan and Krishnan, 2008), since its M-step, corresponding to the inner EM updates, is guaranteed to monotonically improve, rather than maximize the $\mathcal{Q}$ function.

5.1 Modeling Binary Images

Our benchmark tests use the two publicly available datasets: MNIST and OCR. We implement SGD (Neal, 1992) and NVIL (Mnih and Gregor, 2014), and tuned their parameters for the best performance. The performance metric used is the log-likelihood of the test set, *i.e.*, $\log p(\boldsymbol{v}_{\text{test}} \mid \boldsymbol{\theta})$. Two approaches are employed to approximate this log-likelihood: AIS (Salakhutdinov and Murray, 2008; Grosse, 2014) and the variational lower bound. AIS typically provides a more accurate estimate but at a higher computational cost than the lower bound. To fairly compare the learning models on the log-likelihood, we use AIS to estimate the lower bound on the variational methods as well as the variational lower bound.

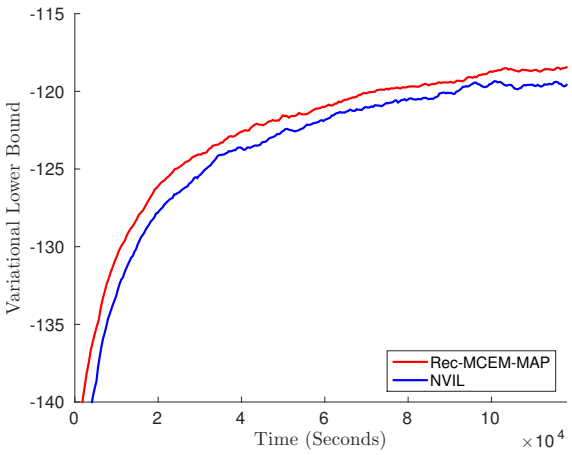
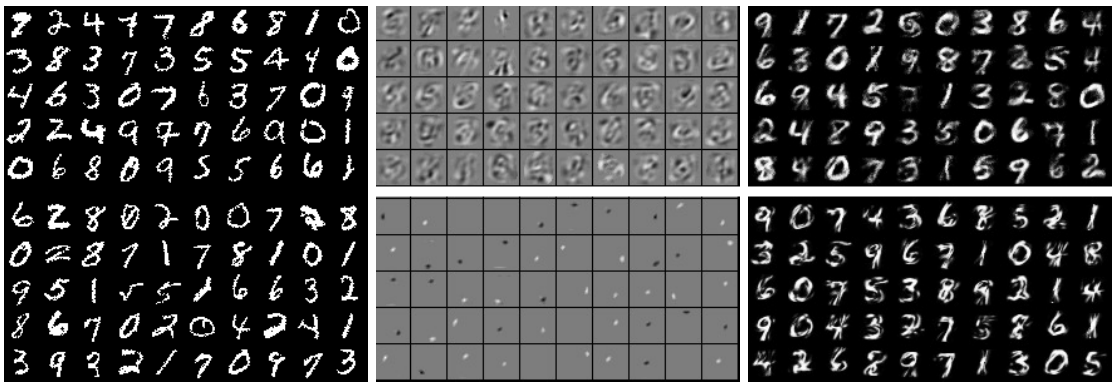


Figure 2: Variational lower bound as a function of time on the MNIST dataset, for a 200-hidden node SBN.

MNIST The MNIST dataset contains 60,000 training and 10,000 test images of size 28×28. We transform them into binary images using the same approach as (Salakhutdinov and Murray, 2008). From Table 1, we observe that MCEM-ML and MCEM-MAP give the best log-likelihood for a single hidden layer. We note that a newly proposed algorithm, SSD (Carlson et al., 2016) achieves a slightly higher log-likelihood of −98.5. For a two-layer model, NVIL achieves the best performance among all the methods. We then test the performance of the Rec-MCEM introduced in Section 4 by computing the variational lower bound. Figure 2 shows the learning curves for NVIL and Rec-MCEM-MAP. We observe that Rec-MCEM-MAP converges faster than NVIL. This is consistent with the analysis in Section 3.3, showing that PG updates move to the optimum point faster than the gradient ascent approach. These curves match the results in Table 2, which Rec-MCEM-MAP has an improved test lower bound over NVIL. We note that the two-layer models for Rec-MCEM and NVIL are trained with ADAM (Kingma and Ba, 2015), which empirically showed improved test performances on the MNIST dataset.

We generated synthesized images with the two-layer model with the parameters learned by MCEM-MAP. This uses the same approach as Gan et al. (2015b),



(a) (b) (c)

Figure 3: (a) Samples from training images; (b) Learned dictionary. (c) Synthesized images by MCEM-MAP with $\lambda = 0.01$ (top) and $\lambda = 1$ (bottom). Dictionary sparsity for $\lambda = \{0.01, 1\}$ is 0.669 and 0.037, respectively.

where label information is employed for training. As depicted in Figure 3, most of the synthesized images look similar to the training images. The dictionary generated by setting $\lambda = 1$ is more sparse and spatially localized than that for $\lambda = 0.01$, as expected. Sparsity is defined here as the proportion of elements of $W^{(1)}$ whose absolute values are greater than 10^{-9}. The synthetic digits generated with the sparse dictionary have good visual quality. Since SBN is a directed model, images in Figure 3(c) are generated more efficiently than the RBM. It avoids the need to sample visible and hidden units alternatively.

OCR The OCR dataset contains 42,152 training and 10,000 test images of size 16×8. Table 1 shows that similar to the MNIST case, both MCEM and NVIL outperform other methods. We observe that MCEM-MAP outperforms NVIL on a two-layer model. For Rec-MCEM, both ML and MAP variants have the largest lower bound for the one-layer case, as shown in Table 2. We show the learning curves in Figure 4. Similar to the MNIST case, Rec-MCEM-ML converges to a better optimum at a faster speed, NVIL still has higher lower bound than Rec-MCEM in the two-layer case.

5.2 Topic Modeling

We employ a negative binomial distribution with dispersion parameter r as the likelihood to model count data. The two datasets used to benchmark our model are 20 Newsgroups and Reuters Corpus Volume I (RCV1-v2); this is the same as Srivastava et al. (2013); Gan et al. (2015a). To compute the predictive perplexities on the test set, we follow the 80/20% split described in Gan et al. (2015a) (details in the Supplemental Materials). The results for Over-RSM and DPFA-SBN are taken from Srivastava et al. (2013) and Gan et al. (2015a), respectively. The results for NVIL are from our implementation. The dispersion parameter is tested at $r = \{0.1, 0.2, \ldots, 1.0\}$ over a validation set and the best one is calculated on the test set. Since RCV1-v2 is much larger than 20 Newsgroup, we use the online MCEM algorithm, with a mini batch size

Table 2: Average test variational lower bound (in nats) on two datasets. "Dim" is the number of units in each hidden layer.

Method	Dim	MNIST	OCR
VB	200	-117.04	-47.70
NVIL		-117.48	-43.65
Rec-MCEM-ML		-116.70	-42.10
Rec-MCEM-MAP		**−116.58**	**−41.83**
VB	200-200	-113.93	-45.98
Rec-MCEM-ML		-106.54	-41.18
Rec-MCEM-MAP		-106.42	-40.84
NVIL		**−105.50**	**−39.20**

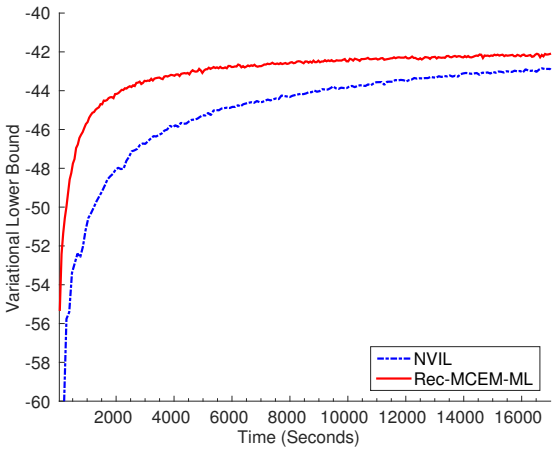


Figure 4: Variational lower bound as a function of time on the OCR dataset, for a 200 hidden nodes SBN.

Table 3: Five Nearest Neighbours from the word representation space learned by MCEM-MAP.

weapons	god	baseball	images	patients	car	clinton	computer
weapon	jesus	pitching	jpeg	disease	cars	administration	computers
guns	christ	season	image	treatment	engine	bush	engineering
gun	bible	players	formats	cancer	miles	secretary	science
armed	christians	teams	gif	medicine	rear	president	dept
assault	worship	winning	mov	doctor	honda	senate	operations

to $1,000$.

Table 4: Test perplexities for topic modeling. "Dim" is the number of hidden units in a layer. "N/A" indicates results were not available in corresponding references.

	Dim	20 News	Reuters
NVIL	128	1142	1260
Over-RSM		958	1060
Rec-MCEM-ML		927	1190
Rec-MCEM-MAP		916	1186
MCEM-ML		**839**	1031
MCEM-MAP		843	**1023**
DPFA-SBN	128-64-32	846	1143
DPFA-SBN	1024-512-256	N/A	964

Table 4 shows the test perplexities for different algorithms. For both 20 Newsgroups and Reuters datasets, MCEM-ML and MCEM-MAP outperforms other methods with a single layer. Encouragingly, MCEM-ML and MCEM-MAP achieve smaller perplexities than DPFA-SBN, the state-of-the-art deep models for topic modeling, given the same number of hidden nodes in the bottom layer. With a large network with dimension 1024-512-256, DPFA-SBN has the smallest perplexity on the Reuters dataset. This implies that adding more layers may improve the performance; we leave this extension as future work. Moreover, we notice that Rec-MCEM achieves smaller perplexities than NVIL, which is consistent with what we observed for a one-layer model in Section 5.1. The NVIL we implemented does not perform identically to Mnih and Gregor (2014), which may be explained by differences in pre-processing schemes.

Figure 5 shows the learning curves for both NVIL and Rec-MCEM-MAP. We notice that for the first few iterations, the perplexity of Rec-MCEM-MAP is much higher than that of NVIL. After that, Rec-MCEM-MAP converges at a faster speed and achieves smaller perplexity than NVIL when the curves stabilize.

Following Larochelle and Lauly (2012), we compare two words by calculating their Euclidean distance in the representation space, *i.e.*, $\|W_{i\cdot} - W_{j\cdot}\|_2$. Table 3 shows the five nearest neighbors for eight specific words from MCEM-MAP. All of the neighbours appear close in meaning, implying the algorithm learns a reasonable representation.

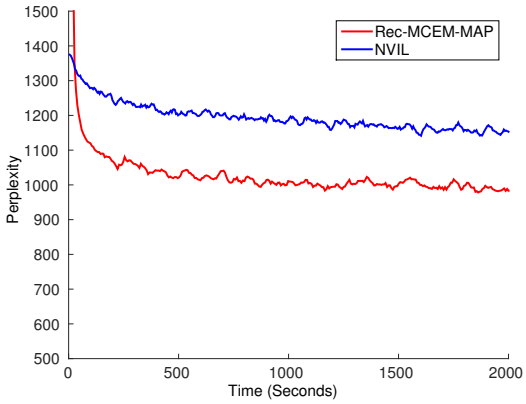


Figure 5: Perplexity as a function of time on the 20 Newsgroups for a 128-hidden node SBN.

6 Conclusion

We develop an MCEM algorithm for MAP estimation in SBNs. The connection between MCEM and the SGD formulation (Neal, 1992) is illustrated by presenting SGD as a special case of the MCEM algorithm. We optimize the maximization (M) step in the MCEM algorithm via an inner EM algorithm with analytic updates based on the PG and Laplace data augmentations. We empirically show that the PG update has faster convergence compared with the first-order gradient ascent method. We also show that MCEM can be combined with the recognition model for improved scalability.

Our future work will be focused on the extension of MCEM and Rec-MCEM to deep networks and large datasets with parallel computing. Another direction is to employ alternative sampling schemes, such as Pakman and Paninski (2013) in the Monte Carlo E-step to improve the sampling method's efficacy and efficiency.

Acknowledgements

This research was supported in part by ARO, DARPA, DOE, NGA, ONR and NSF.

References

Beal, M. J. and Ghahramani, Z. (2003). The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. In *Bayesian Statistics*, volume 7, pages 453–464. Oxford University Press.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Carlson, D., Cevher, V., and Carin, L. (2015a). Stochastic spectral descent for restricted Boltzmann machines. In *AISTATS*.

Carlson, D., Hsieh, Y.-P., Collins, E., Carin, L., and Cevher, V. (2016). Stochastic spectral descent for discrete graphical models. *IEEE J. Sel. Topics Signal Process.*

Carlson, D. E., Collins, E., Hsieh, Y.-P., Carin, L., and Cevher, V. (2015b). Preconditioned spectral descent for deep learning. In *NIPS*, pages 2953–2961.

Chan, K. and Ledolter, J. (1995). Monte Carlo EM estimation for time series models involving counts. *J. Am. Statistical Association*.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. Ser. B.* with discussion.

Figueiredo, M. A. (2003). Adaptive sparseness for supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.*

Frey, B. J. (1997). Continuous sigmoidal belief networks trained using slice sampling. In *NIPS*.

Gan, Z., Chen, C., Henao, R., Carlson, D., and Carin, L. (2015a). Scalable deep Poisson factor analysis for topic modeling. In *ICML*.

Gan, Z., Henao, R., Carlson, D., and Carin, L. (2015b). Learning deep sigmoid belief networks with data augmentation. In *AISTATS*.

Gregor, K., Danihelka, I., Mnih, A., Blundell, C., and Wierstra, D. (2014). Deep autoregressive networks. In *ICML*.

Grosse, R. B. (2014). *Model selection in compositional spaces*. PhD thesis, MIT.

Hinton, G., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*.

Hinton, G. E. (2010). Learning to represent visual input. *Philosophical Transactions of the Royal Society B: Biological Sciences*.

Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. In *ICLR*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*.

Larochelle, H. and Lauly, S. (2012). A neural autoregressive topic model. In *NIPS*.

Levine, R. A. and Casella, G. (2001). Implementations of the Monte Carlo EM algorithm. *Journal of Computational and Graphical Statistics*.

Liang, P. and Klein, D. (2009). Online EM for unsupervised models. In *NAACL*.

McLachlan, G. J. and Krishnan, T. (2008). *The EM Algorithm and Extensions*. Wiley, New York.

Mnih, A. and Gregor, K. (2014). Neural variational inference and learning in belief networks. In *ICML*.

Neal, R. M. (1992). Connectionist learning of belief networks. *Artificial intelligence*.

Pakman, A. and Paninski, L. (2013). Auxiliary-variable exact Hamiltonian Monte Carlo samplers for binary distributions. In *NIPS*.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.

Polson, N. G., Scott, J. G., and Windle, J. (2013). Bayesian inference for logistic models using Pólya–Gamma latent variables. *J. Am. Statistical Association*.

Polson, N. G. and Scott, S. L. (2011). Data augmentation for support vector machines. *Bayesian Analysis*.

Salakhutdinov, R. and Murray, I. (2008). On the quantitative analysis of deep belief networks. In *ICML*.

Saul, L. K., Jaakkola, T., and Jordan, M. I. (1996). Mean field theory for sigmoid belief networks. *Journal of artificial intelligence research*.

Scott, J. G. and Sun, L. (2013). Expectation-maximization for logistic regression. *arXiv preprint arXiv:1306.0040*.

Srivastava, N., Salakhutdinov, R. R., and Hinton, G. E. (2013). Modeling documents with deep Boltzmann machines. In *UAI*.

Wei, G. C. and Tanner, M. A. (1990). A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *J. Am. Statistical Association*.

Zhou, M., Hannah, L. A., Dunson, D. B., and Carin, L. (2012). Beta-negative binomial process and Poisson factor analysis. In *AISTATS*.