Proceedings of 2012 4th International Conference on Machine Learning and Computing
IPCSIT vol. 25 (2012) © (2012) IACSIT Press, Singapore

Learning Restricted Boltzmann Machines using Mode-Hopping MCMC

Yang Hu[+] and Yong Yu

Shanghai Jiao Tong University, Shanghai, 200240, China

Abstract. When Learning Restricted Boltzmann Machines (RBMs) using Stochastic Maximum Likelihood (SML), the mixing rate of the persistent Markov chain in SML plays an important role in learning better generative models. In this paper, we propose an algorithm to improve mixing by using long-range hopping on the energy landscape, which uses and collects the knowledge of model distribution along with learning. The experimental results show that our approach achieves competitive likelihood, and is 5 to 10 times faster than previously reported methods. Moreover, our approach keeps reasonable acceptance ratio even for high-dimensional models, and has the ability to directly discover and eliminate spurious modes which is a main cause of low likelihood in RBM learning.

Keywords: RBM, SML, MCMC, Mode-hopping

1. Introduction

Restricted Boltzmann Machines (RBMs) [1] is a simple but powerful graphical model which is attracting more and more research interests in recent years. It has been successfully applied in typical machine learning applications such as classification and collaborative filtering, etc. Specifically, it's also the building block of the much more powerful Deep Belief Networks. However, Maximum Likelihood learning in RBMs is typically very hard due to the presence of the partition function which requires an exponential time to compute. Alternative methods such as Contrastive Divergence [2] are developed to overcome this difficulty.

Recently, a newly introduced learning algorithm, called Stochastic Maximum Likelihood (SML) or Persistent Contrastive Divergence (PCD) [3], has achieved promising performance on both computational complexity and likelihood quantity [4]. The SML algorithm follows an iterative framework of first sampling from the current model and then using the samples to improve it. As a general approach, Markov chain Monte Carlo (MCMC) is used to do the sampling task. Recent studies on SML found that, if the MCMC sampler is frequently trapped in some modes on the energy landscape, the learned model shows a very bad generation ability, which also implies low likelihood and bad conditional sampling [5]. This leads to an interesting question: How to design a MCMC sampler in SML that can avoid mode trapping?

As for this problem, currently the major solution is using tempered MCMC in SML learning [6-8], in which several auxiliary "tempered" distributions are introduced to facilitate mixing. However, for steep distributions in high-dimensional space, a great number of tempered distributions may be typically required to achieve reasonable acceptance ratio, which makes the approach being costly in practice. In this paper, we propose a new method based on mode-hopping MCMC. By constantly collecting the location information of modes in the distribution, our method can perform hopping between modes very effectively. To the best of our knowledge, such learning-specific mode-hopping algorithm has never been discussed before, so this idea may bring a new direction for future research.

[+] Corresponding author. Tel.: +8613611742683.
E-mail address: gavinhu119@gmail.com.

2. Restricted Boltzmann Machines

Restricted Boltzmann Machines (RBMs) is a specific graphic model which has a bipartite structure. It contains two types of unit: *visible* and *hidden*, each resides in a layer. Visible units are used to represent the data, and hidden units are used to represent latent factors. We often use v to denote the state vector of the visible units, and h to denote the state vector of the hidden units. The state of units can either be continuous, discrete or binary [x]. In this paper, we focus on binary case for both layers.

For an RBM containing $|v|$ visible units and $|h|$ hidden units, let v_i denotes the state of the i-th visible unit, and h_j denotes the state of the j-th hidden unit. Then for a given state v,h, the energy of the RBM is defined as

$$E(v,h) = -\sum_{i=1}^{|v|}\sum_{j=1}^{|h|} v_i W_{ij} h_j - \sum_{i=1}^{|v|} v_i b_i - \sum_{j=1}^{|h|} h_j c_j \tag{1}$$

where W_{ij}, b_i, c_j are model parameters. The probability of state v,h is

$$P(v,h) = \frac{1}{Z}\exp(-E(v,h)), Z = \sum_{v,h}\exp(-E(v,h)) \tag{2}$$

3. Stochastic Maximum Likelihood Learning for RBMs

To learn an RBM, we are given a dataset $D = \{v^{(1)}, v^{(2)}, ..., v^{(d)}\}$, and the learning objective is to maximize the log-likelihood function as follows

$$L = \sum_{k=1}^{d} \log P(v^{(k)}) \equiv \sum_{k=1}^{d} \log \sum_{h} P(v^{(k)}, h) \tag{3}$$

In Stochastic Maximum Likelihood learning, we follow the gradient ascending procedure. For each iteration, we compute the gradient as (for length limitation we omitted the bias parameters b_i, c_j)

$$\frac{\partial L}{\partial W_{ij}} \propto \frac{1}{d}\sum_{k=1}^{d} v_i^{(k)} P(h_j \mid v^{(k)}) - \left\langle v_i h_j \right\rangle_{P(v,h)} \tag{4}$$

where $\left\langle \cdot \right\rangle_{P(v,h)}$ denotes an expectation under the current model distribution, which cannot be precisely computed in a reasonable time. To quickly approximate this term, in SML, a MCMC sampler is introduced to take samples from the distribution. Since the state of the Markov chain is not reset after the model is updated, the Markov chain is also referred to as a *Persistent Markov Chain* [3].

4. Mode-Hopping in SML

Intuitively, a mode is a deep well on the energy landscape such that a MCMC sampler based on local transition can hardly move out when it enters in. From this point of view, a mode can usually be represented by the state with minimal energy in its well, we refer to such state as a *mode state*. To locate modes, we introduce a deterministic function $\mu(x)$ that returns a nearby *mode state* of x, which is usually implemented with a local optimization algorithm.

Throughout this paper, we use a mode set $L = \{m_i : m_i = \mu(m_i), i = 1,2,3,...\}$ to keep the knowledge of mode locations required in our algorithm, in which each mode i is represented by the inner *mode state* m_i. Let $|L|$ denotes the number of modes in L. In practice, $|L|$ can be huge and we can't keep all the modes, thus we restrict the size of L under a predefined upperbound K. This leads to a problem of how to select modes into L, which will be discussed in section 4.2.

4.1. Proposing Mode-Hoppings in RBMs

Due to RBMs' bipartite structure, we can perform mode-hopping in marginal distribution of either v or h rather than in their joint distribution. In particular, consider a transition kernel $T(v,.)$ for the visible states.

Suppose the current state is (v,h), to propose a new state, we can first propose a new v using $T(v,.)$, followed by a new h using the Gibbs transition $P(.|v)$. Let (v',h') denotes the new state, then according to the Metropolis-Hastings rule, the acceptance probability of which is

$$r((v,h)\to(v',h'))=\min\left\{1,\frac{P(v',h')T(v',v)P(h|v)}{P(v,h)T(v,v')P(h'|v')}\right\}=\min\left\{1,\frac{P(v')T(v',v)}{P(v)T(v,v')}\right\} \tag{5}$$

To propose a mode-hopping in marginal distribution $P(v)$, we introduce the following transition kernels:

- $\hat{g}(v,i)$, moving from state v to a nearby mode i $(1\le i\le|L|)$.
- $H(i,j)$, hopping from mode i to j $(1\le i,j\le|L|,i\ne j)$.
- $\breve{g}(i,v)$, proposing a state v via mode i $(1\le i\le|L|)$.

where i,j denote mode indices in L. The mode-hopping from state v can be performed as follows: first move to a nearby mode i, next perform a hopping from i to j, then propose a new state v' via mode j, finally accept v' with probability

$$r(v,v')=\min\left\{1,\frac{P(v')\hat{g}(u,j)H(j,i)\breve{g}(i,v)}{P(v)\hat{g}(v,i)H(i,j)\breve{g}(j,u)}\right\} \tag{6}$$

To implement $\breve{g}(i,.)$ in RBMs, we first compute h that maximize $P(h|m_i)$, then we draw $v\sim P(.|h)$. For $\hat{g}(v,i)$, assume we have a mixture distribution with each $\breve{g}(i,.)$ as a component, we can take $\hat{g}(v,i)$ as the *a posteriori* probability of component i under the observation of v. According to Bayes rule, we have

$$\hat{g}(v,i)\equiv p(i|v)=\frac{p(v|i)p(i)}{\sum_{j=1}^{|L|}p(v|j)p(j)}\equiv\frac{\breve{g}(i,v)p(i)}{\sum_{j=1}^{|L|}\breve{g}(j,v)p(j)} \tag{7}$$

We can simply set $p(i)=1/|L|$ for each i, which yields $\hat{g}(v,i)=\breve{g}(i,v)/\sum_{j=1}^{|L|}\breve{g}(j,v)$.

Consider the mode-hopping probability, $H(i,j)$. If we can estimate the weight w_{ij} associated with hopping from mode i to mode j, then $H(i,j)$ can be defined as $H(i,j)=w_{ij}/\sum_{k\ne i}w_{ik}$.

We propose three general strategies for estimating the weights: 1. $w_{ij}=1$ for all i,j, where hopping to any other mode is uniformly chosen; 2. $w_{ij}\propto P(m_j)$ where a hopping to the mode with lowers energy is preferred; 3. $w_{ij}\propto\|m_i-m_j\|$ where $\|.\|$ denotes a distance measure, in this case a hopping with longer distance is preferred.

4.2. Maintaining the Mode Set

Using prior location information of the modes to facilitate mode-hopping is not a new idea [9]. In AIN Ibrahim's method [10], an initial exploration on the energy landscape is performed to locate modes before sampling. However, this approach can't be applied in learning directly, since (1) the energy landscape is almost flat before learning and we can't find any useful mode; (2) the energy landscape is changing during learning and the modes information may be out of date. Using training data to locate modes, as proposed in [11], could be a feasible solution since those modes are always valid with the model being correctly learned. But this approach cannot handle any spurious mode, which is occasionally learned and will hurt the SML learning if it's not eliminated in the end.

Rather than keeping a static modes information, our method starts with $L=\{\}$, then dynamically collects and adds modes into L along with learning. We use search algorithms to locate modes as AIN Ibrahim did. However, instead of performing an initial exploration, we rebuild L periodically to keep it up-to-date as the model changes.

In particular, since the size of L is limited by K, we define a total order for the mode states to determine which modes should be selected into L. We simply use the probability of mode states to rank the modes, and dynamically find and add modes into L. If the size of L exceeds its capacity K, we just drop the modes with lowest probabilities in L to guarantee that we always have $|L|\le K$. Besides, a predefined radius R is also introduced to prevent modes in L from aggregating by assuring that every two mode states m_i,m_j in L

have a hamming distance greater than or equal to R. If two states in L violate this restriction, we drop the one with lower probability. To find modes, in this work, we first use a random walk to pick a non-mode state v, then we use a local optimization $\mu(v)$ to locate its nearby mode state.

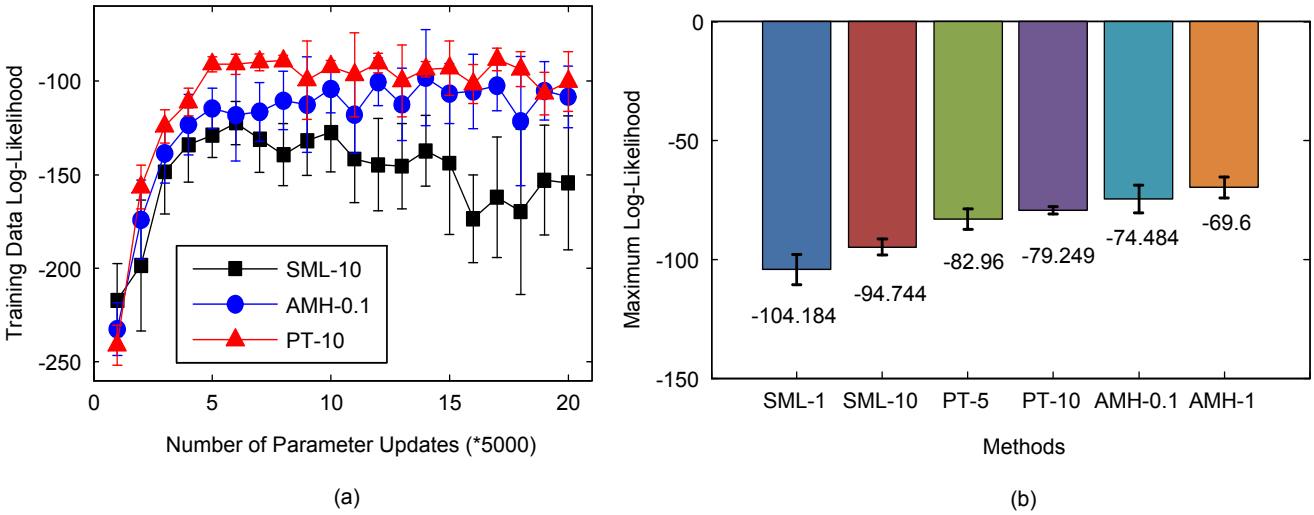

Fig. 1: a) Log-likelihood of training data under a model trained with SML-10, AMH-0.1 and PT-10. b) The maximum log-likelihood obtained with 100,000 updates for various algorithms.
Both SML-*n* and PT-*n* are approximately *n* times slower than SML-1. Besides, in our implementation, AMH-0.1 is just slightly slower than SML-1, and AMH-1 is 2 or 3 times slower than SML-1.

5. Experimental Results

In this section, we conduct various experiments on the familiar MNIST handwritten dataset to validate our algorithm. We first compare our algorithm with previous methods, and test its scalability on RBMs with various dimensionalities, and then we show visually that our algorithm can find spurious modes during learning.

5.1. Toy Experiments on the Log-Likelihood

We first conduct a series of toy experiments, in which we select 10 datavectors of digits 0 to 9 from MNIST to learn a small RBM with 10 hidden units, thus we can compute the exact log-likelihood of the model. In this setting we have 10 modes in total while the marginal distribution of h contains $2^{10}=1024$ different states. Since the number of states are much more than modes we believe there are many high energy states that may form barriers blocking the transition between certain modes, which can cause trouble for samplers.

For comparison, we train the model using different methods including SML, tempered SML and our mode-hopping SML. For tempered SML, we choose Parallel Tempering [8] as a representative since its parameters are relatively easy to handle. We use SML-*n* to denote regular SML learning with *n* Gibbs updates per parameter change, PT-*n* to denote Parallel Tempering SML with *n* tempered chains, and AMH-*p* to denote our method with a probability *p* of proposing a mode-hopping in each learning iteration. For PT and AMH, we always perform 1 Gibbs update per learning iteration, and the inverse temperatures of PT are spaced uniformly between 1 and 0.9. In the toy experiments, we set K to 10 and R to 80. We run each algorithm for 10 times and summarize their performance by their average log-likelihood and corresponding standard deviation (Fig. 1).

5.2. On the Acceptance Ratio

In our toy experiments, AMH shows an excellent performance. Because of the tiny dimensionality of the model and the small number of modes, the distribution won't be very steep, and AMH can always keep a reasonable acceptance ratio. Next we consider using more data to train higher dimensional models.

We take the first 10,000 datavectors from MNIST as training data. Fig. 2 shows the evolving of acceptance ratio as learning proceeds when h is set to various dimensionalities. Clearly, without using mode set, the acceptance ratio drops rapidly as the dimensionality raises (Fig. 2a). In our method (Fig. 2b), the acceptance ratio always stays in a reasonable range regardless of high dimensionality, and it does not drop drastically along with learning. Moreover, we can collect more modes in L by increasing K and decreasing R, which will provide more comprehensive information about the mode location. This can further improve the acceptance ratio of mode-hopping as shown in Fig. 2c.

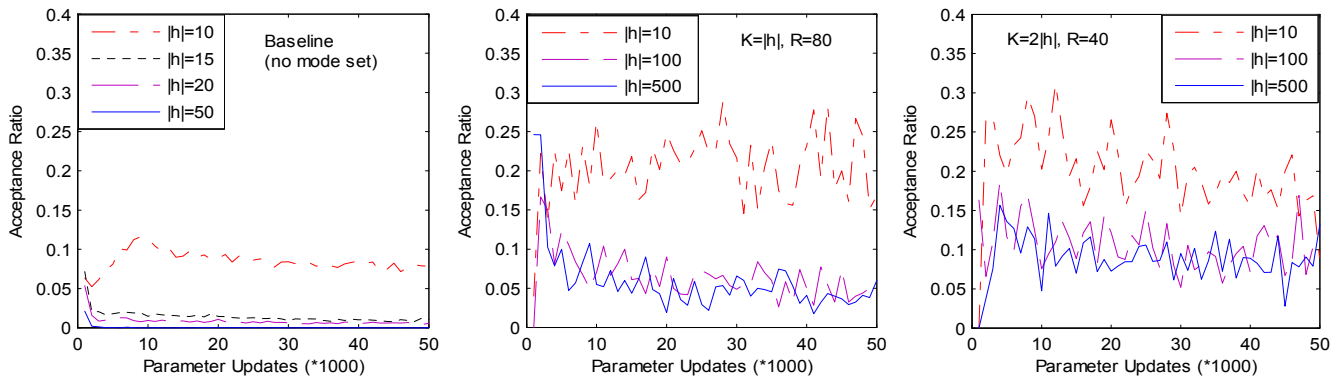

Fig. 2: Acceptance ratio along with learning: a) Baseline; b) K = $|h|$, R = 80; c) K = $2|h|$, R = 40. Our baseline is a random walk in the marginal distribution of h without using mode set.

5.3. On the Mode Set

Fig. 3 demonstrates some modes collected when learning an RBM with 500 hidden units. It is clear that there are many spurious modes beside the ones close to the training data, which will reduce the likelihood of the model. Our method successfully locates such modes and raises the chance to eliminate them by learning. At this point, our method is much better than other methods based on static mode information [11].

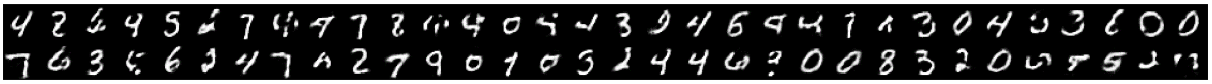

Fig. 3: Some modes collected when learning an RBM with 500 hidden units.

6. Conclusion

In this paper, a novel RBM learning algorithm of SML type is presented, in which we use the dynamically maintained location information of the modes to improve the mixing of the persistent chain. Since we don't use any intermediate distribution, and the mode location information can be incrementally updated during learning, our method is much more effective than the previous tempered methods. We test the algorithm under RBMs with various dimensionalities, indeed, it achieves superior performance at a relatively cheap computational cost.

7. References

[1] P Smolensky. Information processing in dynamical systems: Foundations of harmony theory. D. E. Rumelhart and J. L. McClelland (Eds.), Parallel distributed processing, 1:194–281, 1986.

[2] Geoffrey Hinton. Training products of experts by minimizing contrastive divergence. Neural Computation, 4:1771–1800, 2000.

[3] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In Proceedings of the 25th international conference on Machine learning, pages 1064–1071, 2008.

[4] Benjamin M. Marlin, Kevin Swersky, Bo Chen, and O De Freitas. Inductive principles for restricted boltzmann machine learning. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010

[5] Tijmen Tieleman and Geoffrey Hinton. Using fast weights to improve persistent contrastive divergence. In Proceedings of the 26th International Conference on Machine Learning, pages 1033–1040, 2009.

[6] Ruslan Salakhutdinov. Learning in markov random fields using tempered transitions. In Advances in Neural Information Processing Systems, 2010.

[7] Ruslan Salakhutdinov. Learning deep boltzmann machines using adaptive mcmc. In Proceedings of the 27th International Conference on Machine Learning, 2010.

[8] G. Desjardins, A. Courville, Y. Bengio, P. Vincent, and O. Delalleau. Tempered markov chain monte carlo for training of restricted boltzmann machine. In Proceedings of AISTATS, pages 145–152, 2010

[9] C. Sminchisescu, M. Welling, and G. Hinton. A mode-hopping mcmc sampler. Technical report, University of Toronto, 2003

[10] AIN Ibrahim. New Methods for Mode Jumping in Markov Chain Monte Carlo Algorithms. PhD thesis, University of Bath, 2009.

[11] Geoffrey Hinton and Max Welling. Wormholes improve contrastive divergence. In Advances in Neural Information Processing Systems, 2003