Markov Chain Monte Carlo

Introduction, Comparison & Analysis

Tzu-Heng Lin, 2014011054, W42
Department of Electronic Engineering, Tsinghua University, Beijing, China
lzhbrian@gmail.com

ABSTRACT

Markov Chain Monte Carlo (MCMC) is a technique to make an estimation of a statistic by simulation in a complex model. Restricted Bolztmann Machine(RBM) is a crucial model in the field of Machine Learning. However, training a large RBM model will include intractable computation of the partition functions, i.e.Z(θ). This problem has aroused interest in the work of estimation using a MCMC methods. In this paper, we first conduct Metropolis-Hastings Algorithm, one of the most prevalent sampling methods, and analyze its correctness & performance, along with the choice of the accepting rate. We then implement three algorithms: TAP, AIS, RTS, to estimate partition functions of an RBM model. Our work not only give an introduction about the available algorithms, but systematically compare the performance & difference between them. We seek to provide an overall view in the field of MCMC.

1. INTRODUCTION

Markov Chain A Markov Chain is a special stochastic process in which the current state only depends on its previous state, we call this property the Markov property or memorylessness. i.e.

$$P(X_{t+1} = x | X_t, X_{t-1}, \cdots) = P(X_{t+1} = x | X_t) \qquad (1)$$

Given a Markov Chain, if a vector π, has the property:

$$\pi = \pi P \qquad (2)$$

then we call π the stationary distribution, it denotes the final state distribution of the stochastic process in a Markov Chain.

Markov Chain has a wide application in many fields in the real world, such as social science[1], econmics & finance[4] and of course, computer science[11], etc.

Markov Chain Monte Carlo If we are given a probability distribution $p(x)$, it would be a great thing if we could generate some samples of it by a simple method, so here comes the Markov Chain Monte Carlo(MCMC) method. If we could construct a Markov Chain which its stationary distribution π just equals to $p(x)$, then we could use this Markov Chain to sample from this distribution $p(x)$. And this is the main idea of MCMC.

In this paper, we implement the Metropolis-Hastings Algorithm, which is one of the most widely used sampling method. We also deal with the accepting rate, which I will introduce later, for previous work[13] have shown that the accepting rate may influence the result of the experiment and there is theoretical support in choosing an optimal accepting rate.

Restricted Boltzmann Machine A Restricted Bolztmann Machine (RBM)[8] is a significant work bringing hypothesis in statistical physics to computer science. By stacking several layers of RBM, we will get a fundamental model, Deep Belief Network[6], in the field of Deep Learning, which is nowadays the hottest class of algorithms used in Machine Learning.

Estimating Partition functions In the process of training an RBM, however, will include incontractable computation of the partition function. When the model grows large, the complexity of this work will be incompletable. The good news is, several researches have shown that there are ways to avoid this by using an MCMC approach instead, to estimate it.

In this paper, we implement three prevalent MCMC methods of estimating a partition function, Thouless-Anderson-Palmer Sampling(TAP)[3], Annealed Importance Sampling(AIS)[10, 14], Rao-Blackwellized Tempered Sampling(RTS)[2], respectively, and give an overall comparison on the theory & performance between them.

2. RELATED WORK

Monte Carlo Sampling Method

Partition Function Estimation

...

3. METROPOLIS-HASTINGS

In the Introduction, we have shown the main idea of an MCMC sampling method. In this section, we will introduce the Metropolis Hastings Algorithm and conduct an experiment.

3.1 Algorithm[1]

3.1.1 Detailed Balance Condition

Before stepping further into the MH Algorithm, We would first introduce a theorem called the Detailed Balance Condition.

In the introduction, we said that we want to construct a Markov Chain which its stationary distribution $\pi(x)$ just equals to the required probability distribution $p(x)$.

At first, a theorem is needed.

THEOREM 3.1 (DETAIL BALANCE CONDITION). *Given a non periodic Markov Chain, if*

$$\pi(i)P_{ij} = \pi(j)P_{ji} \quad for\ all\ i,j \tag{3}$$

then $\pi(x)$ is the stationary distribution of this Markov Chain.

3.1.2 MCMC sampling method

Suppose we already have a transition matrix Q for a Markov Chain, $q(i,j)$ denote the probabilty of transition from state i to state j. For the general case,

$$p(i)q(i,j) \neq p(j)q(j,i)$$

That is to say, we do not have the detailed balance condition (Theorem 3.1) So we introduce an $\alpha(i,j)$ s.t.

$$p(i)q(i,j)\alpha(i,j) = p(j)q(j,i)\alpha(j,i) \tag{4}$$

By sysmetrical characteristic, we choose:

$$\alpha(i,j) = p(j)q(j,i) \quad \alpha(j,i) = p(i)q(i,j) \tag{5}$$

So the new Markov Chain Q' would have the property of which its stationary distribution is $p(x)$

$$p(i)\underbrace{q(i,j)\alpha(i,j)}_{Q'(i,j)} = p(j)\underbrace{q(j,i)\alpha(j,i)}_{Q'(j,i)} \tag{6}$$

We call the $\alpha(i,j)$ we introduced, accepting ratio. It means that, in the original Markov Chain Q, when state i transits to state j with a probability of $q(i,j)$, we accept this transition with a probabilty of $\alpha(i,j)$

Now, we have derived the MCMC sampling method.

3.1.3 Metropolis-Hastings Algorithm

The MCMC sampling method is a marvellous work. However, it has a critical drawback that if $\alpha(i,j)$ & $\alpha(j,i)$ are too small, we would seldom accept the transition.

A solution is that we multiply both $\alpha(i,j)$ & $\alpha(j,i)$ with a constant to make sure that the larger one between them equals 1. By doing so, we change the accepting ratio to

$$\alpha(i,j) = \min\left\{\frac{p(j)q(j,i)}{p(i)q(i,j)}, 1\right\} \tag{7}$$

and now, we get Metropolis-Hastings Algorithm[5].

I would like to further introduce one more concept called accepting rate(not accepting ratio), which denotes the statistic ratio of accepting the transition. i.e. If we request 10 transition and we accept 8 times, then the accepting rate would be 0.8. This concept is crucial when we are dealing with a continual Markov Chain to use the MH algorithm.

3.1.4 Symmetric Case

In a Markov Chain whose transition matrix is symmetric, we have

$$q(i,j) = q(j,i)$$

so the accepting ratio could be simplified to

$$\alpha(i,j) = \min\left\{\frac{p(j)}{p(i)}, 1\right\} \tag{8}$$

which is also known as the Metropolis Algorithm[9].

3.1.5 Continual Case

In a continual Markov Chain, such as the experiment we are going to do in the next subsection, we have a vague definition of transition matrix Q. So we introduce a concept called the proposal jump size, $sd.T$.

The method we get x_{k+1} from x_k is to add a sampled point of a normal distribution with a variance of the jump size and $\mu = 0$. For a two dimension example, we have:

$$x_{k+1} = x_k + sd.T \begin{pmatrix} norm_1 \\ norm_2 \end{pmatrix} \tag{9}$$

3.2 Sampling Experiment

For our experiment, we use an example of a bivariate Normal distribution, with

$$\mu = \begin{pmatrix} 5 \\ 10 \end{pmatrix}, \Sigma = \begin{pmatrix} 1 & 1 \\ 1 & 4 \end{pmatrix}$$

By theoretical computation, we can easily compute the the pearson correlation between the two dimensional value is 0.5.

$$\rho = 0.5$$

We then generate 10,000 samples using the MH algorithm and take the second half (i.e. the last 5,000 points), setting the standard deviation of proposal to 3.0. We can see from the result (Figure 1) that we have derived 5,000 sampled points whose pearson correlation value $\rho = 0.50009$, which matches the theoretical value.

3.3 Performance Analysis

3.3.1 Choice of proposal jump size

MH algorithm is an effective MCMC method for many diverse problems. However, its efficiency depends crucially on the selection of the proposal density. With the proposal jump size being small, the accepting rate would be very low and eventually stick to only one point(eg. the initial point); When the proposal jump size is too big, the accepting rate would be too high.

Roberts et al. have shown in previous work[13] that the optimal accepting rate of the MH algorithm should approximately be at 0.234 for the case of an N-dimensional Gaussian target distribution. We test the accepting rates in different proposal jump size(Figure 2) and find that the optimal value

[1] Available at https://github.com/lzhbrian/MCMC/blob/master/metropolis_hastings/metropolis_hasting.R in R[12]

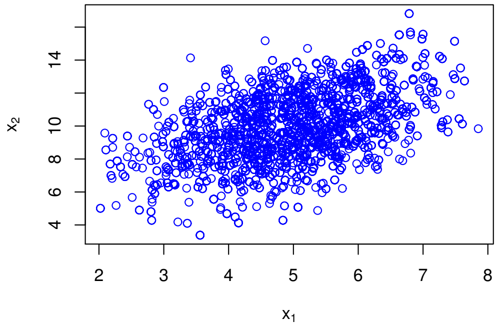

Figure 1: Sampling result of 5,000 points correlation = 0.50009 , set sd.T = 3.0

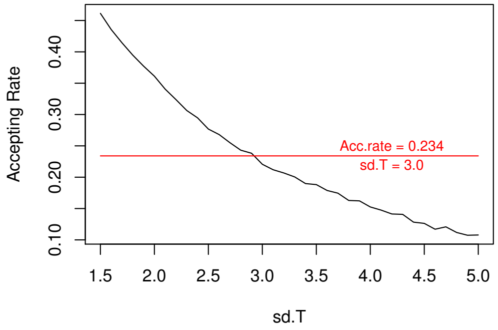

Figure 2: Accepting rate on different proposal jump size.

should be at appoximately 3.0 to acquire a model with accepting rate being close to 0.234. That is the reason why we choose 3.0 as our proposal jump size.

3.3.2 Efficiency

Due to the limit of the accepting rate, for a high dimensional condition, using the MH sampling methods may spend more time in traverse all of the possible states, which could sometimes be be less satisfying. Thus, many would switch to Gibbs Sampling Algorithm.

3.4 Gibbs Sampling

Gibbs Sampling is a special case of Metropolis Hastings Algorithm, by letting the accepting rate = 1, we will get a Gibbs Sampler. As the length & time limit, we will not specify more here.

4. PARTITION FUNCTION ESTIMATION

4.1 Restricted Bolztmann Machine

Co-invented and enhanced largely[6] by Geoff Hinton, a Restricted Bolzmann Machine(RBM)[8] is a model which brings the idea of a physics concept to the field of computer science.

4.1.1 Introduction

An RBM is a two-layer undirected model(Figure 3). The first layer of the RBM is called visible layer, and the second is called the hidden layer. In the model, every visible units are connected to all hidden units and vice versa. For every given value of visible layer **v** & hidden layer **h**, we can define an energy of this state.

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\mathbf{v}^T\mathbf{W}\mathbf{h} - \mathbf{b}^T\mathbf{v} - \mathbf{a}^T\mathbf{h} \tag{10}$$

where $\theta = \{W, \mathbf{b}, \mathbf{a}\}$ are the model configurations. W_{ij} represents the weight between visible unit v_i and hidden unit h_j. **b** & **a** are biases for visible and hidden layer, respectively.

4.1.2 Training an RBM

On training an RBM, we want our RBM model to have a lowest scale of energy. By doing so, we have to calculate the joint distribution over the visible and hidden units, which is defined by:

$$p(\mathbf{v}, \mathbf{h}; \theta) = \frac{e^{-E(\mathbf{v},\mathbf{h};\theta)}}{Z(\theta)} \tag{11}$$

where

$$Z(\theta) = \sum_{\mathbf{v}} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\theta)} \tag{12}$$

is the partition function.

However, calculating partition functions has always been an intractable work since we have to traverse all the possible state of **v** & **h**. When the model grows large, this process will be very time & rescouces consuming and thus become unrealistic for the real practice.

So, we have to introduce methods to estimate the partition functions instead of just calculating it in brute force. Although some deviation may include in the estimation, but the efficiency along with them make them preferable. In fact, studies have shown that only few deviation is included that we could just ignore it since it does petty influence on our training.

In the next subsection, we will discuss about three methods available, which each have their pros and cons in doing this complex estimation.

4.2 Algorithms

4.2.1 Thouless-Anderson-Palmer Sampling[2]

Thouless-Anderson-Palmer Sampling(TAP)[3] is a very efficient and easy-to-practice iterative procedure based on an improved mean field method from statistical physics called Thouless-Anderson-Palmer approach.

The main idea of this method is to iteratively compute the magnetization vector m^v,m^h, and then input the values into

[2] Available at https://github.com/lzhbrian/MCMC/blob/master/rbm/TAP.m in Matlab

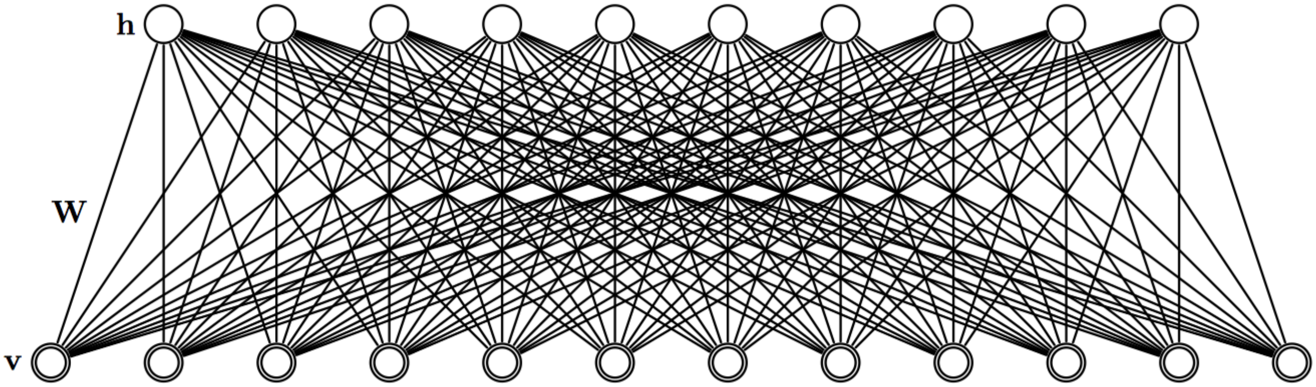

Figure 3: A Restricted Boltzmann Machine

the Legendre transform of the free energy $F = log(Z(\theta))$ to compute it.

The Legendre transform of F to the second order is:

$$\begin{aligned}\Gamma(\mathbf{m}^v, \mathbf{m}^h) \approx -S(\mathbf{m}^v, \mathbf{m}^h) - \sum_i a_i m_i^v - \sum_j b_j m_j^h \\ - \sum_{i,j} \Bigg(W_{i,j} m_i^v m_j^h \\ - 0.5 W_{ij} \Big(m_i^v - (m_i^v)^2 \Big) \Big(m_j^h - (m_j^h)^2 \Big) \Bigg)\end{aligned} \tag{13}$$

where $S(\mathbf{m}^v, \mathbf{m}^h)$ indicates the entropy:

$$\begin{aligned}S(\mathbf{m}^v, \mathbf{m}^h) = -\sum_i \Bigg(m_i^v log m_i^v + (1 - m_i^v) log(1 - m_i^v) \Bigg) \\ - \sum_j \Bigg(m_j^h log m_j^h + (1 - m_j^h) log(1 - m_j^h) \Bigg)\end{aligned} \tag{14}$$

The pseudo code of this algorithm is shown below:

4.2.2 Annealed Importance Sampling[3]

Annealed Importance Sampling(AIS)[10, 14] is probably one of the most preferable estimating methods avaible.

Previous work [7] have shown that if P_A and P_B in the SIS method is not close enough, the estimator would be very poor.

Based on SIS, the main idea of this algorithm is to gradually alter the value from an known Z_A to our required Z_B (or Z_K), by the following identity:

$$\frac{Z_K}{Z_0} = \frac{Z_1}{Z_0} \frac{Z_2}{Z_1} ... \frac{Z_K}{Z_{K-1}} \tag{15}$$

where

$$\frac{Z_K}{Z_{k+1}} = \frac{1}{M} \sum_{i=1}^{M} \frac{P_{k+1}^*(\mathbf{x}^{(i)})}{P_k^*(\mathbf{x}^{(i)})} \quad where\ x^{(i)} \sim P_k \tag{16}$$

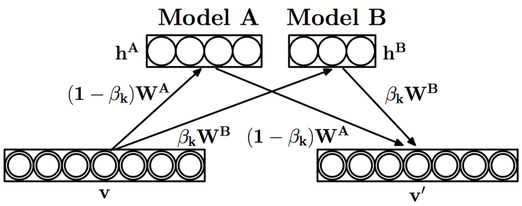
Model A
Model B
$\mathbf{h^A}$
$\mathbf{h^B}$
$(\mathbf{1} - \beta_\mathbf{k})\mathbf{W^A}$
$\beta_\mathbf{k}\mathbf{W^B}$
$\beta_\mathbf{k}\mathbf{W^B}$
$(\mathbf{1} - \beta_\mathbf{k})\mathbf{W^A}$
$\mathbf{v}$
$\mathbf{v'}$

Figure 4: The transition process from x_k to x_{k+1} which leaves $P_k(\mathbf{v})$ invariant.

in which we can get x_{k+1} from:

$$\begin{aligned}p(h_j^A = 1|\mathbf{v}) = sigmoid\Bigg((1 - \beta_k) \Bigg(\sum_i W_{ij}^A v_i + a_j^A \Bigg) \Bigg) \\ p(h_j^B = 1|\mathbf{v}) = sigmoid\Bigg(\beta_k \Bigg(\sum_i W_{ij}^B v_i + a_j^B \Bigg) \Bigg) \\ p(v_i' = 1|\mathbf{h}) = sigmoid\Bigg((1 - \beta_k) \Bigg(\sum_j W_{ij}^A h_i^A + b_i^A \Bigg) \\ + \beta_k \Bigg(\sum_j W_{ij}^B h_j^B + b_i^B \Bigg) \Bigg)\end{aligned} \tag{17}$$

this procedure is shown in Figure 4.

Note that model A indicates an initial model which we can easily compute all its configurations.

β in the above equations is defined by users as a set of inverse temperatures $\{0 = \beta_1 < \beta_2 < ... < \beta_K = 1\}$, which can define a sequence of

$$P_k(\mathbf{x}) \propto P_A^*(\mathbf{x})^{1-\beta_k} P_B^*(\mathbf{x})^{\beta_k} \tag{18}$$

where

$$P_k^*(\mathbf{v}) = \sum_{h^A h^B} e^{(1-\beta_k)E(\mathbf{v},\mathbf{h}^A;\theta_A) + \beta_k E(\mathbf{v},\mathbf{h}^B;\theta_B)} \tag{19}$$

The pseudo code of this algorithm is shown below:

4.2.3 Rao-Blackwellized Tempered Sampling[4]

[3] Available at https://github.com/lzhbrian/MCMC/blob/master/rbm/AIS.m in Matlab

[4] Available at https://github.com/lzhbrian/MCMC/blob/master/rbm/RTS.m in Matlab

Similar to AIS, Rao-Blackwellized Tempered(RTS)[2] Sampling also has a set of inverse temperatures $\{0 = \beta_1 < \beta_2 < ... < \beta_K = 1\}$, which can define a sequence of

$$f_k(\mathbf{x}) \propto f(\mathbf{x})^{\beta_k} p_1(\mathbf{x})^{1-\beta_k} \tag{20}$$

The pseudo code of this algorithm is shown below:

4.2.4 Other method

There are many other methods which can also estimate the partition functions. Such as Self-adjusted mixture sampling(SAMS)[15] proposed a method to estimate multiple partition functions together to improve the efficiency. As the length & time limit, we only implement 3 methods here in this paper.

4.3 Estimating Results

4.4 Performance Analysis

4.4.1

5. CONCLUSION

In this paper, we discuss about the Markov Chain Monte Carlo method which are now undoubtedly one of the most important sampling methods.

We comprehensively introduce the concept of Metropolis-Hastings Algorithm and conduct an experiment to verify its correctness. We also make some analysis about how accepting rate would interfere the sampling result.

We systematically compare three methods of partition function estimationm which are crucial works in training a Restricted Bolztmann Machine or a Deep Belief Network.

As future work, we would like to join more methods to the comparison and if could, propose some improvement to the algorithms available.

6. ACKNOWLEDGEMENT

I would like to thank Yubo Chen, Liren Yu, Yuanxin Zhang, XueChao Wang, Changran Hu, for the discussion with me on the algorithms. Without them, I wouldn't have the possibility to accomplish this work in such a short time. This paper is a project of Stochastic Process Course in Tsinghua University, taught by Prof. Zhijian Ou.

7. REFERENCES

[1] D. Acemoglu, G. Egorov, and K. Sonin. Political model of social evolution. *Proceedings of the National Academy of Sciences*, 108(Supplement 4):21292–21296, 2011.

[2] D. Carlson, P. Stinson, A. Pakman, and L. Paninski. Partition functions from rao-blackwellized tempered sampling. *arXiv preprint arXiv:1603.01912*, 2016.

[3] M. Gabrié, E. W. Tramel, and F. Krzakala. Training restricted boltzmann machine via the thouless-anderson-palmer free energy. In *Advances in Neural Information Processing Systems*, pages 640–648, 2015.

[4] J. D. Hamilton. A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica: Journal of the Econometric Society*, pages 357–384, 1989.

[5] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[6] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[7] D. J. MacKay. *Information theory, inference and learning algorithms.* Cambridge university press, 2003.

[8] J. L. McClelland, D. E. Rumelhart, P. R. Group, et al. Parallel distributed processing, 1987.

[9] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[10] R. M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

[11] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[12] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2016.

[13] G. O. Roberts, A. Gelman, W. R. Gilks, et al. Weak convergence and optimal scaling of random walk metropolis algorithms. *The annals of applied probability*, 7(1):110–120, 1997.

[14] R. Salakhutdinov. *Learning deep generative models*. PhD thesis, University of Toronto, 2009.

[15] Z. Tan. Optimally adjusted mixture sampling and locally weighted histogram analysis. *Journal of Computational and Graphical Statistics*, (just-accepted), 2015.