Markov Chain Monte Carlo

Introduction, Comparison & Analysis

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

ABSTRACT

[1]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.$Z(\theta)$. 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.

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

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

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

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[7] and of course, computer science[17], 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[15, 8], 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[19] 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)[14] is a significant work bringing theory in statistical physics to computer science. By stacking several layers of RBM, we will get a fundamental model, Deep Belief Network[9], 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, 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)[5], Annealed Importance Sampling (AIS)[16, 20], Rao-Blackwellized Tempered Sampling(RTS)[3], respectively, and give an overall comparison on the theory & performance between them.

2. RELATED WORK

In this paper, we do not dig very deep into a specific field of the algorithms, but provide an overall view of the MCMC method. There are brilliant seniors who have done many marvellous works in different specific field.

Metropolis-Hastings algorithm Gilks[6] and Chib et al.[4] did a significant job in giving a clear introduction and deeper explanation to the Metropolis-Hastings Algorithm. Beskos et al.[2] made a great work in analysing the complexity of the Algorithm.

Partition function estimation Hubbard[10] is probably the earliest reseacher managing to reduce the complexity of calculating a partition function. Jarzynski[11] systemetically analyzed the bias between the estimation and the real value of the partition functions in an sampling process.

Due to my limited knowledge, there might be some mistakes and flaws in this paper, please don't hesitate to contact

[1]Tzu-Heng Lin is currently an undergraduate student in Dept.of Electronic Engineering, Tsinghua University. His research interests include Big Data Mining, Machine Learning, etc. For more information about him, please see www.linkedin.com/in/lzhbrian. The code in this paper can be found in www.github.com/lzhbrian/MCMC.
Feel free to contact him at any time via lzhbrian@gmail.com or linzh14@mails.tsinghua.edu.cn

and correct me.

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[2]

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.

So, the key question will be how to construct a Markov Chain which satisfy this Detail Balance Condition.

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

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

[2]Available at https://github.com/lzhbrian/MCMC/blob/master/metropolis_hastings/metropolis_hasting.R in R[18]

and now, we get Metropolis-Hastings Algorithm[8].

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[15].

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:

$$x_{k+1} = x_k + sd.T\begin{pmatrix} n_1 \\ n_2 \end{pmatrix} \tag{9}$$

Algorithm 1 Metropolis-Hastings

Require: Required distribution p, Transfer Matrix Q
  Initialize x_1
  for $t = 1 \rightarrow$ inf **do**
    Sample $y \sim q(x|x_t)$
    Sample $u \sim U[0, 1]$
    if $u < \alpha(x_t, y) = min\{\frac{p(y)q(x_t|y)}{p(x_t)q(y|x_t)}, 1\}$ **then**
      $x_{t+1} \leftarrow y$
    else
      $x_{t+1} \leftarrow x_t$
    end if
  end for

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.

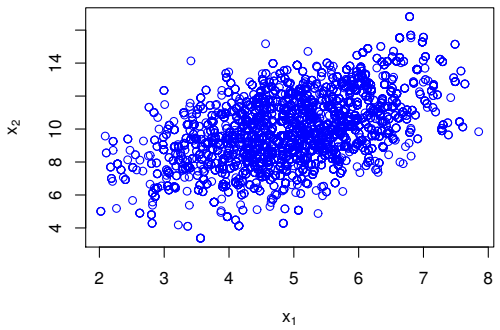

Figure 1: Sampling result of 5,000 points correlation = 0.50009 , set sd.T = 3.0

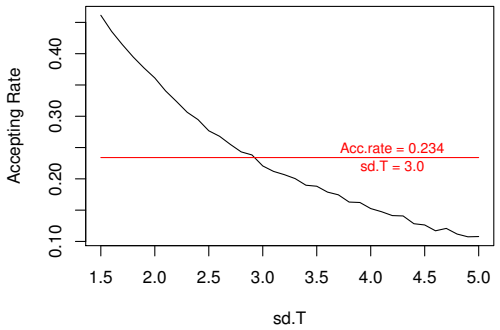

Figure 2: Accepting rate on different proposal jump size.

3.3 Performance Analysis

3.3.1 Choice of proposal jump size

MH algorithm is an effective MCMC method for many diverse problems. However, for a continual case in MH algorithm, its performance somewhat depends 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[19] 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 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. But it is worthy to notice that Gibbs sampling method is used more often than Metropolis-Hastings method in the real practice, probably because it has a slightly simpler process.

4. PARTITION FUNCTION ESTIMATION

4.1 Restricted Bolztmann Machine

Co-invented and enhanced largely[9] by Geoff Hinton, a Restricted Bolzmann Machine(RBM)[14] 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 $\mathbf{v}$ & hidden layer $\mathbf{h}$, 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. $\mathbf{b}$ & $\mathbf{a}$ 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 $\mathbf{v}$ & $\mathbf{h}$. 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[3]

Algorithm Thouless-Anderson-Palmer Sampling(TAP)[5] is a very efficient and easy-to-practice iterative procedure

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

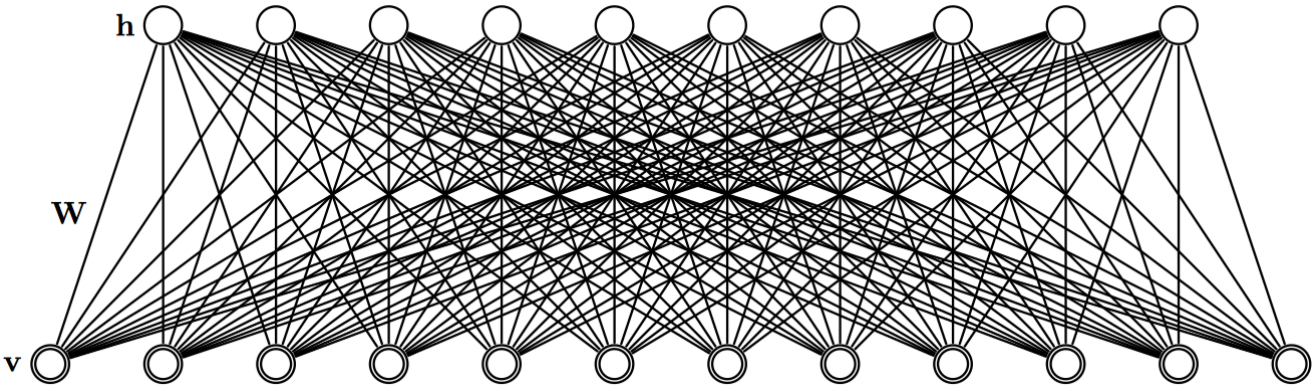

Figure 3: A Restricted Boltzmann Machine

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

The iteration of m^v and m^h follows:

$$\begin{aligned} m_j^h[t+1] &= sigmoid\Bigg[b_j + \sum_i W_{ij} m_i^v[t] - \\ &\qquad W_{ij}^2\left(m_j^h[t] - \frac{1}{2}\right)\Big(m_i^v[t] - (m_i^v[t])^2\Big)\Bigg] \\ m_i^h[t+1] &= sigmoid\Bigg[a_i + \sum_j W_{ij} m_j^h[t+1] - \\ &\qquad W_{ij}^2\left(m_i^v[t] - \frac{1}{2}\right)\Big(m_j^h[t+1] - (m_j^h[t+1])^2\Big)\Bigg] \end{aligned} \tag{13}$$

The Legendre transform of F to the second order is:

$$\begin{aligned} \Gamma(\mathbf{m}^v, \mathbf{m}^h) \approx & -S(\mathbf{m}^v, \mathbf{m}^h) - \sum_i a_i m_i^v - \sum_j b_j m_j^h \\ & - \sum_{i,j}\Bigg(W_{i,j} m_i^v m_j^h \\ & - 0.5 W_{ij}\Big(m_i^v - (m_i^v)^2\Big)\Big(m_j^h - (m_j^h)^2\Big)\Bigg) \end{aligned} \tag{14}$$

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

$$\begin{aligned} S(\mathbf{m}^v, \mathbf{m}^h) = & -\sum_i \Big(m_i^v log m_i^v + (1 - m_i^v) log(1 - m_i^v)\Big) \\ & - \sum_j \Big(m_j^h log m_j^h + (1 - m_j^h) log(1 - m_j^h)\Big) \end{aligned} \tag{15}$$

Practice For real practice, in the next subsection, we see TAP method can obtain a converged result in a very short time, but has less accuracy. And sometimes, the converged results are periodic, which is not what we want by us.

Algorithm 2 Thouless-Anderson-Palmer Sampling

Require: Required Iteration time N,
  Initialize m^v, m^h
  for $t = 1 \to N$ **do**
    $m^h[t+1] \leftarrow f(m^h[t])$
    $m^v[t+1] \leftarrow f(m^v[t], m^v[t+1])$
  end for
  $logZ(\theta) \leftarrow \Gamma(\mathbf{m}^v, \mathbf{m}^h)$

4.2.2 *Annealed Importance Sampling*[4]

Algorithm Annealed Importance Sampling(AIS)[16, 20] is probably one of the most preferable estimating methods avaible.

Previous work [13] 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:

$$\frac{Z_K}{Z_0} = \frac{Z_1}{Z_0}\frac{Z_2}{Z_1}\cdots\frac{Z_K}{Z_{K-1}} \tag{16}$$

where

$$\frac{Z_K}{Z_{k+1}} = \frac{1}{M}\sum_{i=1}^{M}\frac{P_{k+1}^*(\mathbf{x}^{(i)})}{P_k^*(\mathbf{x}^{(i)})} \quad where\ x^{(i)} \sim P_k \tag{17}$$

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

$$\begin{aligned} p(h_j^A = 1|\mathbf{v}) &= sigmoid\Bigg((1-\beta_k)\Big(\sum_i W_{ij}^A v_i + a_j^A\Big)\Bigg) \\ p(h_j^B = 1|\mathbf{v}) &= sigmoid\Bigg(\beta_k\Big(\sum_i W_{ij}^B v_i + a_j^B\Big)\Bigg) \\ p(v_i' = 1|\mathbf{h}) &= sigmoid\Bigg((1-\beta_k)\Big(\sum_j W_{ij}^A h_i^A + b_i^A\Big) \\ &\qquad + \beta_k\Big(\sum_j W_{ij}^B h_j^B + b_i^B\Big)\Bigg) \end{aligned} \tag{18}$$

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

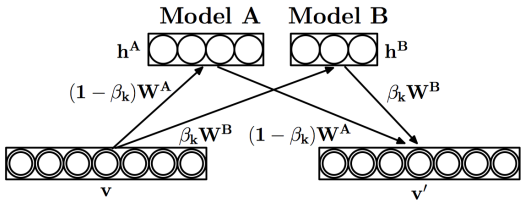

Figure 4: The transition process from x_k to x_{k+1} which leaves $P_k(\mathbf{v})$ invariant.

this procedure is shown in Figure 4.

Note that model A indicates an initial model which we can easily compute all its configurations. Commonly, we choose an RBM model with $\theta = \{0, 0, 0\}$

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

$$P_k(\mathbf{x}) \propto P_A^*(\mathbf{x})^{1-\beta_k} P_B^*(\mathbf{x})^{\beta_k} \tag{19}$$

where

$$P_k^*(\mathbf{v}) = \sum_{h^A h^B} e^{(1-\beta_k)E(\mathbf{v},\mathbf{h}^A;\theta_A)+\beta_k E(\mathbf{v},\mathbf{h}^B;\theta_B)} \tag{20}$$

Initialize Z_A with dataset In [20], Ruslan also notice a method to make Z_A near Z_B. As the length & time limit, we will not specify the process here.

Originally, we initialize model A with a configuration of $\theta = \{0, 0, 0\}$. This method can use the training data to initialize the visible bias $\mathbf{b}$ to a desired value s.t. we can get a better outcome of the estimation.

In our real practice, we find that this method take less than 0.005 second to initialize $\mathbf{b}$ even for a very big model (i.e.784 visible and 500 hidden units), but have strongly improved the result as we will mention in the next subsection.

Algorithm 3 Annealed Importance Sampling

Require: Required β_k s.t. $0 = \beta_0 < \beta_1 < ... < \beta_K = 1$
  Initialize b_A by dataset
  Sample $\mathbf{x}_1$ from $P_A = P_0$
  for $k = 1 \to K - 1$ **do**
    Sample $\mathbf{x}_{k+1}$ given $\mathbf{x}_k$ using $T_k(\mathbf{x}_{k+1} \longleftarrow \mathbf{x}_{k+1})$
  end for
  Set $\omega_{AIS} = \prod_{k=1}^{K} P_k^*(\mathbf{x}_k)/P_{k-1}^*(\mathbf{x}_k)$

4.2.3 Rao-Blackwellized Tempered Sampling[5]

Algorithm Similar to AIS, Rao-Blackwellized Tempered (RTS)[3] Sampling also has a set of inverse temperatures $\{0 = \beta_1 < \beta_2 < ... < \beta_K = 1\}$, which can define a sequence of

$$f_k(\mathbf{x}) \propto f(\mathbf{x})^{\beta_k} p_1(\mathbf{x})^{1-\beta_k} \tag{21}$$

Different from AIS, we do not traverse β. Instead we sample a β^* every loop, from the β set with the distribution $(\beta|x)$.

Subsequently, we sample from x_k to x_{k+1} by the probability of $q(x|\beta^*)$ just like what we did in AIS, shown in Figure 4. However, what also different from AIS is that, we have to iterate from x_k to x_{k+1} many times(i.e. 50 times in[3]) for the sake of getting a better x_{k+1}.

At the last of every loop, we update the lower variance estimator $\hat{\mathbf{c}}$ by

$$\hat{c}_k = \hat{c}_k + \frac{1}{N} q(\beta_k|x) \tag{22}$$

Finally, we get Z_k by

$$\hat{Z}_k^{RTS} = \hat{Z}_k \frac{r_1 \hat{c}_k}{r_k \hat{c}_1}, \quad k = 2, ..., K \tag{23}$$

in which what we do care is $Z_B \approx \hat{Z}_K^{RTS}$.

The posterier distribution $q(\beta_k|x)$ in the above equations is defined by:

$$q(\beta_k|x) = \frac{f_k(x) r_k / \hat{Z}_k}{\sum_{k'=1}^{K} f_{k'}(x) r_{k'} / \hat{Z}_{k'}} \tag{24}$$

Practice In the paper[3], Carlson et al. note an initializing method to initialize Z_k, whose procedure is just like the above process. The only difference is that they sampled β_k by uniform distribution in every loop, not by the distribution $(\beta|x)$. They claim that after doing such initializing work, then we conduct the algorithms above would acquire a better result.

In our real practice, we directly use the initializing method mentioned above by selecting β_k with a uniform distribution in every loop. We also initialize the value of Z_A by the method we have mentioned in the AIS section using the dataset. And we have found that the result is already satisfying, there is no need to conduct more loops with β_k sampled by $(\beta|x)$.

Also, we found that we have to conduct the procedure above for several times s.t. we can acquire our desired partition function value.(i.e. We did it for 100 times, that is to say we update $\mathbf{Z}$ for 100 times).

Algorithm 4 Rao-Blackwellized Tempered Sampling

Require: $\{\beta_k, r_k\}_{k=1,...,K}$
  Initialize b_A by dataset
  Initialize $log\hat{Z}_k, k = 2, ..., K$
  for $n = 1 \to Runtime$ **do**
    Initialize $\beta \in \{\beta_1 ... \beta_K\}$
    Initialize $\hat{c}_k = 0, k = 1, ..., K$
    for $t = 1 \to N$ **do**
      for $t = 1 \to Transitiontime$ **do**
        Sample $\mathbf{x}_{k+1}$ given $\mathbf{x}_k$ using $q(x|\beta^*)$
      end for
      $\mathbf{x}^* \leftarrow \mathbf{x}_{k+1}$
      Sample $\beta^* \sim (\beta|\mathbf{x}^*)$ or $\beta^* \in \{\beta_1 ... \beta_K\}$
      Update $\hat{c}_k \leftarrow \hat{c}_k + \frac{1}{N} q(\beta_k|\mathbf{x}^*)$
    end for
    Update $\hat{Z}_k^{RTS} \leftarrow \hat{Z}_k \frac{r_1 \hat{c}_k}{r_k \hat{c}_1}, k = 2, ..., K$
  end for

4.2.4 Other methods

There are many other methods which can also estimate the partition functions. Such as Self-adjusted mixture sampling(SAMS)[21] 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.

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

Hid Units	Mean $logZ(\theta)$	+/-3 sd	Real Value
10	226.05	0.1762	226.11
20	221.11	0.2329	N/A
100	348.45	0.2460	N/A
500	463.25	0.3790	N/A

Table 1: AIS result (with **b** init) with $+/-3$ standard deviations

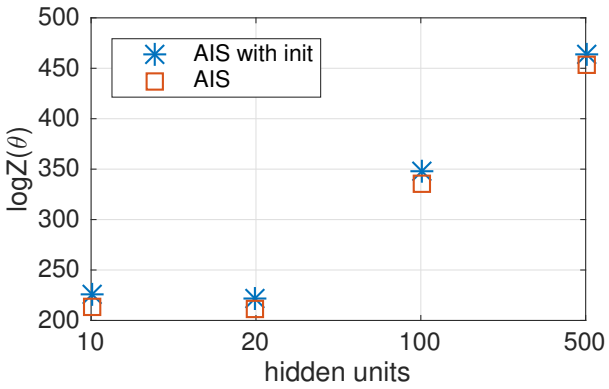

Figure 5: Comparison of AIS method, with or without an initialization of **b**

4.3 Estimating Results

We estimate the results of the three algorithms using 4 models with 10, 20, 100, 500 hidden units respectively, all 4 models have 784 visible units. The models are trained by the MNIST handwritten digit dataset[12].

Note that we only calculate the real value of the partition function in the model with 10 hidden units ($logZ(\theta) = 226.11$), due to my poor laptop has broken down several times when calculating the model with 20 hidden units and maltab doesn't even allow to calculate the other two models because they require unimaginable quantity of memories.

AIS By setting β uniformly sampling from 0 to 1 for 10,000 points, we run 100 times of AIS, and get a result with $+/-3$ standard deviations (Table 1). In a model with 10 hidden units. We see AIS with init almost obtain the real value (226.11) without error.

From Figure 5, we can see that the estimation results will differ a lot if we use an initialization method to initialize the visible bias **b**, as mentioned in the previous subsection. And from the validation of the 10 hidden units case, we can infer that with a initialization of **b** would largely improve the result.

RTS When doing the practice of RTS, as mentioned in the last subsection, we directly use the initializing method mentioned above by selecting β_k with a uniform distribution in every loop. Also, we use the same initialization method as in AIS to initialize **b** in model A.

We make 50 transitions from x_k to x_{k+1} in every loop, we take 100 loops (i.e.$N = 100$) every time, and we conduct the whole procedure for 100 times (i.e.we update **Z** for 100 times). Also, we select 100 point of β uniformly sampled from 0 to 1.

By conducting the whole above procedure for 20 times, we can obtain a result which is as good as AIS (Table 2). However, the efficiency of RTS show weaker performance than that of AIS (Table 3).

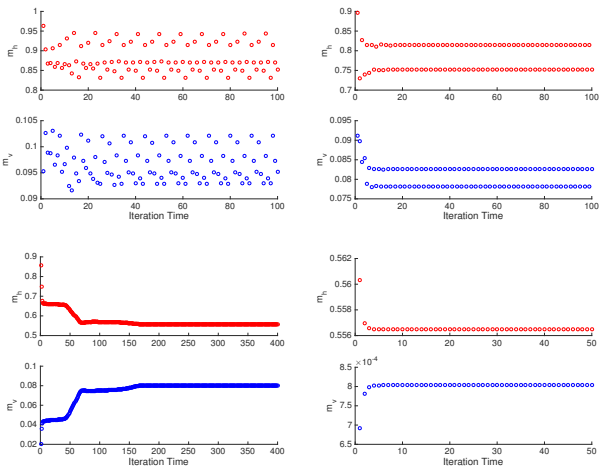

Figure 6: m_h, m_v convergence condition using TAP method. From left to right, up to down, the figure indicates an RBM model with 10, 20, 100, 500 hidden units. The convergence time is approximately 40, 20, 175, 5. By such few iteration time, all results can be obtained in less than 1 second.

Hid Units	TAP	AIS	RTS	Real Value
10	212.32	226.05	226.2707	226.11
20	214.85	221.11	218.8993	N/A
100	342.02	348.45	347.3684	N/A
500	450.31	463.25	459.7372	N/A

Table 2: $Z(\theta)$ estimation result of TAP, AIS, RTS (AIS & RTS with **b** init)

TAP When running a TAP, we see we can get converged m_v & m_h in very few iteration time (Figure 6, convergence time is approximately 40, 20, 175, 5 for the model with 10, 20, 100, 500 hidden units respectively). Given this observation, the computing time of TAP is negligeable (we obtain all results in less than a second). And also, its meaningless to discuss standard deviation here.

However, we do notice that the converged results are sometimes not consistent but periodic. (eg. Figure 6, when there are 10, 20 hidden units), this is not a good news because even if we have more resources to compute the iterations, we would not have a better result.

And disappointingly, compared to the result of other algorithms (Table 2), TAP usually have a lower estimation value, which is not preferable.

4.4 Theoretical Anaylsis

RTS From a theretical perspective, we have proven that the bias & the variance of the RTS method are to be:

$$E[log\hat{Z}_k^{RTS}] - E[Z_k] \approx \frac{1}{2}[\frac{\sigma_1^2}{\hat{c}_1^2} - \frac{\sigma_k^2}{\hat{c}_k^2}] \tag{25}$$

$$Var[log\hat{Z}_k^{RTS}] \approx \frac{\sigma_1^2}{\hat{c}_1^2} + \frac{\sigma_k^2}{\hat{c}_k^2} - \frac{2\sigma_{1k}}{\hat{c}_k\hat{c}_1} \tag{26}$$

where $\sigma_k^2 = Var[\hat{c}_k]$ and $\sigma_{1k} = Cov[\hat{c}_1, \hat{c}_k]$

This has shown that the bias of RTS has no definite sign.

AIS However, in AIS, Neal and Jarzynski et al.[16, 11] have shown that if we want the result to be unbiased, we would have to let $M = 1$ in the iteration, which by doing

Hid Units	TAP	AIS	RTS
10	< 1	77.38	68.28
20	< 1	63.03	87.42
100	< 1	92.87	164.34
500	< 1	154.63	603.59

Table 3: Efficiency of TAP, AIS, RTS (Unit: seconds), with a Intel Core i5 CPU Turbo Boost to 2.7GHz

so have lost the advantage of AIS. That is to say, on the other hand, if $M > 1$, we would have a negative bias due to Jenson Inequality.

From the result of our practice, we see when AIS and RTS show same good performance in estimation, RTS's efficiency is a little bit lower than that of AIS's.

TAP Although TAP shows the best efficiency, its results are the most disapointing. Apparantly TAP has underestimate the value of the partition function.

We did not analyze deeply on the reasons why it failed to perform a satisfying result, but our intuition tells that maybe it is because the Legendre transform. In our practice, we only took the Legendre transform to the 2nd order, which might result in the underestimation.

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 estimation 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] A. Beskos and A. Stuart. Computational complexity of metropolis-hastings methods in high dimensions. In *Monte Carlo and Quasi-Monte Carlo Methods 2008*, pages 61–71. Springer, 2009.

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

[4] S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335, 1995.

[5] 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.

[6] W. R. Gilks. *Markov chain monte carlo*. Wiley Online Library, 2005.

[7] 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.

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

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

[10] J. Hubbard. Calculation of partition functions. *Physical Review Letters*, 3(2):77, 1959.

[11] C. Jarzynski. Nonequilibrium equality for free energy differences. *Physical Review Letters*, 78(14):2690, 1997.

[12] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits, 1998.

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

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

[15] 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.

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

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

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

[19] 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.

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

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