马尔科夫链-蒙特卡洛方法在受限玻尔兹曼机中的实现与性能比较

电子工程系 无 43

丁文浩（2014011079）

dwh14@mails.tsinghua.edu.cn

Abstract

本文首先探讨了马尔科夫链-蒙特卡洛（Markov Chain Monte Carlo，MCMC）方法在二维高斯分布下的采样效率与准确性。进而将四种不同的 MCMC 算法应用于受限玻尔兹曼机（Restricted Boltzmann Machines，RBM）训练过程中的分割函数（partition function）求解过程。

通过比较似然值结果得出了不同采样算法在 MNIST 数据集下的性能与效率差异，并对于各个算法的优点与缺陷做出了分析和比较，最终总结出求解该问题的相对高效的方法与思路。

1.Introduction

许多基本的科学问题都可以使用模拟的思想来解决，蒙特卡洛算法（Monte Carlo）是使用随机的方式来产生大量样本模拟一个计算量巨大的过程，但是 MC 本身的模拟过程是随机的，为了能够得到更好的采样过程，一般使用 MCMC，沿着一条马尔科夫链进行采样，最终达到一个平稳的分布实现精确模拟（[1] Persi Diaconis，2010）。

MCMC 方法最早由 Metropolis 提出，后来由 Hastings 改进，合称为 M-H 算法（[2] Hastings, W.K. 1970），是 MCMC 的基础方法。

随着机器学习的发展，MCMC 在其中的大量数据训练过程中起着重要的作用（[3] Christophe Andrieu, 2003），并发展出了越来越多的优化的 MCMC 算法，比如：AIS（[4] Ruslan Salakhutdinov, 2009）、RTS（[5] David E. Carlson, 2016）、TAP（[6] Marylou Gabrie Eric W. Tramel Florent Krzakala, 2015）、SAMS（[7] Zhiqiang Tan, 2014）。

RBM 是在 1986 年由 Paul Smolensky 提出，这是一种无向二部图模型，一共分为两层，分别是可见层（visible）和隐含层（hidden），且不同于一般的玻尔兹曼机，RBM 的两个层的内部是独立的。

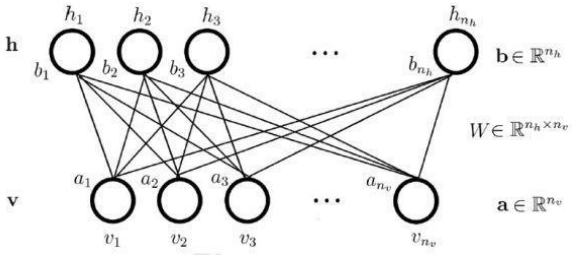

图 1 RBM 图模型

该模型刚被提出时，由于训练过程的复杂性该模型一直没有发展。直到 Hinton（[8] Geoffrey Hinton , 2010）等人提出对比收敛算法（Contrastive Divergence，CD）之后，RBM 才被广泛应用。

上述的几种 MCMC 方法都在 RBM 的分割函数估计问题上有着很好的效果。其中 AIS、RTS 与 SAMS 算法都是基于标准的模拟退火思想（[9] Radford M. Neal, 2008），在高温时寻找全局最优解，随着温度的降低使最优解稳定，但是这三种方法的问题是需要大量的采样过程来逐步逼近一个稳定的分布，所以在时间开销上比较大。TAP 算法采用的是一种交

叉迭代的方式，虽然能够快速的收敛，但是从结果上来看可能会产生周期振荡的现象，有时不能得到稳定解。SAMS 算法是 AIS 算法和 RTS 算法的一种总结与改进，避免了 AIS 这种在全局范围（global-jump）内的采样方式，同时相比 RTS 而言提高了样本点迭代更新的效率，能够在保证准确率的同时快速地收敛到最终结果。在文章的结果分析部分可以看到各个算法的时间效率和准确率的分析比较。

本文的结构如下：首先抽象出了 RBM 的物理模型，并在该模型下介绍了上述四种较为高效的 MCMC 分割函数计算方法，通过对于每种算法的编程实现给出最终的结果，最后对于得到的似然值结果进行比较和算法分析。

2.Model

RBM 是一种无监督学习方法，其目的是最大可能的拟合输入数据，但是对于一组输入数据来说，在不知道分布的情况下是非常难以学习的。根据统计力学的结论表明，任何概率分布都可以转变成基于能量的模型，并且可以利用能量模型的特有的性质和学习过程。

RBM 就采用了这种基于能量的物理模型，能量函数是描述整个系统状态的一种测度：概率分布越集中，系统的能量越小；系概率分布越趋于均匀分布，则系统的能量越大。因此，能量函数的全局最小值对应的是系统的最稳定状态。

RBM 的能量函数公式如下：

$$E(v,h) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i$$

而在统计热力学上，当系统和它周围的环境处于热平衡时，状态 i 发生的概率遵循下面的公式：

$$p_i = \frac{1}{Z} \times e^{-\frac{E_i}{k_b \times T}}$$

其中的 Z 是与状态无关的常数，基于这一点我们可以在能量函数的基础上就定义可视节点和隐藏节点的联合概率分布：

$$p(v,h) = \frac{e^{-E(v,h)}}{\sum_{v,h} e^{-E(v,h)}}$$

其中分母称为归一化常数，在热力学中也叫作分割函数，常表示为 Z(v, h, θ)。该求解过程的问题在于，当样本量比较大的时候 Z 的计算量巨大，不能采用普通的方式，所以在这里引入了 MCMC 方法，通过采样的方式来近似获得这个常数。

RBM 模型的用途主要是四种：

1）对数据进行编码，然后交给监督学习方法去进行分类或回归，当做一个降维的方法来使用。

2）得到了权重矩阵和偏移量，供 BP 神经网络初始化训练。实际应用结果表明，直接把 RBM 训练得到的权重矩阵和偏移量作为 BP 神经网络初始值，可以得到非常好的结果。

3）估计联合概率 p(v,h)，如果把 v 当做训练样本，h 当成类别标签，就可以利用贝叶斯公式求解 p(h |v)，进而进行分类，类似朴素贝叶斯、LDA、HMM，作为一个生成模型（Generative model）使用。

4）接计算条件概率 p(h | v)，如果把 v 当做训练样本，h 当成类别标签，就可以作为一个判别模型（Discriminative model）使用。

本文中在 MNIST 手写数据集中的应用就是 RBM 的判别模型的用法。

3.Basic Theory and Method

首先为了对 MCMC 的采样过程有一个充分地认识，采用 Metropolis-Hastings 算法对于二维高斯分布进行了采样。

M-H 算法是 MCMC 的基础方法。由 M-H 算法演化出了许多新的抽样方法，包括目前在 MCMC 中最常用的 Gibbs 抽样也可以看做 M-H 算法的一个特例。M-H 算法的过程如下，其中的转移函数 q(·) 使用了参数如下的二维高斯分布：

var_X = 1; var_Y = 1; corr_XY = 0.5;

所以条件采样的实现是在确定均值的情况下利用 q(·) 进行采样。

Algorithm 1 Metropolits-Hastings 采样算法

1）初始化马氏链的初始状态 $X_0 = x_0$

2）对 t = 0,1,2,… 循环以下过程进行采样

- 第 t 个时刻马氏链状态为$X_t = x_t$，采样 $y \sim q(x \mid x_t)$
- 从均匀分布采样，$u \sim \text{Uniform}[0,1]$
- 如果$u < \alpha(x_t, y) = \min\left\{\frac{p(y)q(x_t \mid y)}{p(x_t)q(y \mid x_t)}, 1\right\}$，则接受转移$y \leftarrow x_t$，即$X_{t+1} = y$；否则不接受转移，即$X_{t+1} = X_t$

根据该算法对一个参数如下的二维高斯分布进行 40000 次模拟采样：

$$\mathcal{N}\left\{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \middle| \begin{pmatrix} 5 \\ 10 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 1 & 4 \end{pmatrix}\right\}$$

最终得到的采样结果如下图所示：

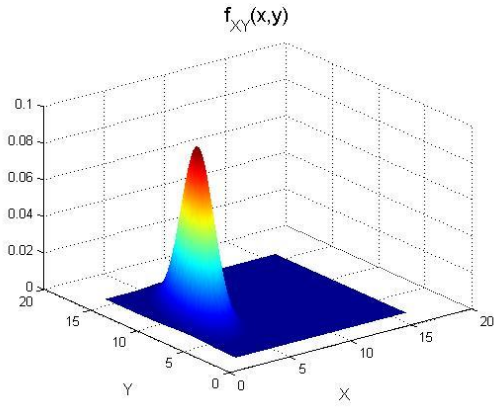

图 2 M-H 算法采样结果

采样过程是一个马氏链不断趋于平稳的过程，所以使用不同的颜色标出状态转移的过程如下图（**四种颜色代表四个阶段，顺序依次是黄、绿、红、蓝**）：

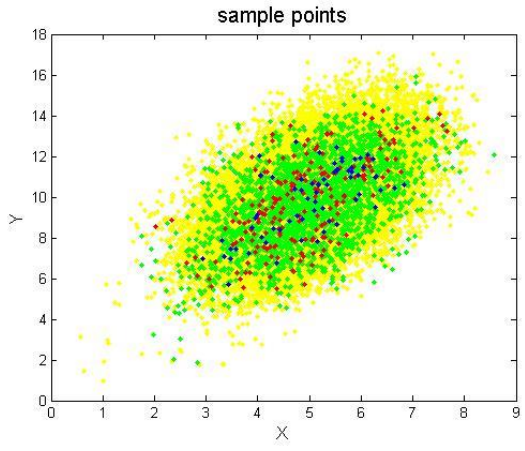

图 3 M-H 算法状态转移过程

从图中可以看出，当采样次数足够多时，最终的状态已经能够很好的收敛到一个有限的区域内，也就是说能够达到一个平稳分布。

最终的模拟相关系数为 0.511533，与理论值 0.5 基本一致。以此作为基础下面介绍我所采样的四种算法的基本理论：

1）AIS

AIS（Annealed Importance Sampling）算法的全称是退火重要性采样，采用一种基于模拟退火的思想。

其基本思想是使用马氏链构造一个逐渐趋近于平稳分布的序列，然后根据公式得到每两个状态之间的 Z 的比值：

$$\frac{\mathcal{Z}_B}{\mathcal{Z}_A} \approx \frac{1}{M}\sum_{i=1}^{M}\frac{P_B^*(\mathbf{x}^{(i)})}{P_A^*(\mathbf{x}^{(i)})} \equiv \frac{1}{M}\sum_{i=1}^{M} w^{(i)}$$

$$\frac{\mathcal{Z}_K}{\mathcal{Z}_0} = \frac{\mathcal{Z}_1}{\mathcal{Z}_0}\frac{\mathcal{Z}_2}{\mathcal{Z}_1}\cdots\frac{\mathcal{Z}_K}{\mathcal{Z}_{K-1}}.$$

由于状态 A 使我们自己构造的，所以根据过程中的ω的平均值就可以得到最终的状态 B 的归一化常数Z_B。整个算法的核心在于使用温度的倒数 β 作为迭代的步长，这是典型的模拟退火的思想：

$$\beta = \frac{1}{Temperature}$$

β从 0 到 1 的过程代表着温度从无穷大下降到 0，这个退火的过程可以有效地避免陷入局部最优解，从而在迭代结束的时候可以得到全局最优解。

2）RTS

RTS（Rao-Blackwellized Tempered Sampling） 算法与上面讨论的 AIS 算法有很多相似之处，同样是利用了模拟退火的采样方式，但是不同的是，在 AIS 方法当中我们每一次采样都是没有限制的，只是在给定了x_k的条件下通过采样得到x_{k+1}，这种称为 global-jump 的方式的缺点就是我们需要大量的采样点才能够缓慢收敛到我们最后想要的平稳分布。

为了能够使下一个采样状态能够尽可能地接近我们需要的分布，我们同时对所有的温度进行更新，最终选取最后一个温度（也就是最低温度）时的更新结果就是我们需要的稳定解，而整个 Z 的更新过程依赖于如下的公式：

$$Z_k = Z_k' \frac{r_1}{r_k}\frac{c_k}{c_1}$$

这里的 k 指的是整个向量的每一个分量，k = 1,2,3…K。上式中的 ck 称作 Rao-Blackwellized 形式的估计器（estimator），其用途就是判断将Z_k迭代至一个稳定的结果，其更新过程为：

$$\hat{c}_k = \frac{1}{N}\sum_{i=1}^{N} q(\beta_k|x^{(i)})$$

其中的β与 x 即为我们需要采样的结果。q(·)的表达式如下：

$$q(\beta_k) = \frac{r_k Z_k/\hat{Z}_k}{\sum_{k'=1}^{K} r_{k'} Z_{k'}/\hat{Z}_{k'}}$$

从该式子中可以看出，当我们迭代得到的Z_k与真实值存在误差时，还式子可以得到一个偏差值，根据这个偏差值可以修正 ck 的取值，直到迭代到 ck 接近于 rk 的稳定分布时则可以认为Z_k的取值是接近于真实值的。

3）TAP

TAP（Thouless-Anderson-Palmer）算法的主要思想是采用一种统计物理学的方法，通过求解确定性方程的方式通过交叉迭代得到自由能量，而不是采用传统的采样的方式。其具体实现过程是使用了一种拓展的高阶均值场方法（extened mean filed），从而弥补了以往采用一阶的 mean filed 算法的不足（website[2]）。

该方法采用的是三阶的求解过程，自由能表达式为：

$$F^{EMF} = S + \beta F^{MF} + \frac{1}{2}\beta^2 F^{Onsager} + \frac{2\beta^3}{3}\sum_{(i,j)} W_{ij}^3 (m_i - m_i^2)(\frac{1}{2} - m_i)(m_j - m_j^2)(\frac{1}{2} - m_j)$$

其中一阶项即为普通 MF 方法的计算结果，二阶项即为 TAP 方法的核心内容，是 TAP 方程中的 Onsager reaction 修正项，三阶项修正项是为了减少迭代结果的振荡幅度。

式中的m_i、m_j 分别表示更新过程中的 v 和 h 向量，更新的过程利用了稳定方程，也就是自由能的导数为零：

$$\frac{d\Gamma}{dm} = 0$$

下面列出一阶和二阶形式的 TAP 方程：

$$A(0, m^v) = \sum_i m^v \, ln \, m^v + (1 - m^v)\ln(1 - m^v)$$

$$A(0, m^h) = \sum_i m^h \, ln \, m^h + (1 - m^h)\ln(1 - m^h)$$

$$\frac{\partial}{\partial\beta} A(\beta, m^v, m^h) = \sum_i a_i m_i^v + \sum_i b_i m_i^h + \sum_{i,j} m_i^v \, w_{i,j} m_j^h$$

$$\frac{\partial^2}{\partial\beta^2} A(\beta, m^v, m^h) = \frac{1}{2}\sum_{i,j} (m_i^v - (m^v)_i^2) w_{i,j}^2 (m_j^h - (m^h)_j^2)$$

在迭代过程中其实还是使用了一个退火的热力学模型，不过所选取的 β 恒等于 1。

4）SAMS

SAMS（self - adjusted mixture sampling）算法属于一类算法，包括两种转移采样方法三种迭代更新的方法。

其中的两种转移核函数算法分别是 global-jump 类型和 local-jump 类型

$$K_\theta(y_{t-1}, y_t) = p_{\mathrm{LJ}}(l_t|l_{t-1}, x_{t-1}; \zeta)\, p(x_t|l_t, x_{t-1})$$

$$K_\theta(y_{t-1}, y_t) = p_{\mathrm{GJ}}(l_t|x_{t-1}; \zeta)\, p(x_t|l_t, x_{t-1})$$

从公式中可以看出其区别在于求解概率密度函数时是否受限于上一次的结果，典型的 global-jump 方法是 AIS，local-jump 算法是 RTS。

三种更新迭代的方法统一的形式是：

$$\zeta^{(t-\frac{1}{2})} = \zeta^{(t-1)} + \gamma_t(\delta^{(t)} - \pi)$$

$$\zeta^{(t)} = \zeta^{(t-\frac{1}{2})} - \zeta_1^{(t-\frac{1}{2})}$$

其中的 $\zeta_1^{(t-\frac{1}{2})}$ 指的是 $\zeta^{(t-\frac{1}{2})}$ 的第一个元素，这样可以保证第一个元素在每一次更新之后为零。

其主要思想与 RTS 基本相同，即为了尽快地得到收敛值，不采用简单的 global-jump 的采样方式，而是采用 local-jump 的条件采样方式，在给定平稳分布 π_j 的条件下进行更新和状态转移，转移公式如下：

- $\zeta^{(t-\frac{1}{2})} = \zeta^{(t-1)} + t^{-1}[\frac{\delta_1(L_t)}{\pi_1}, \dots, \frac{\delta_m(L_t)}{\pi_m}]^T$
- $\zeta^{(t-\frac{1}{2})} = \zeta^{(t-1)} + t^{-1}[\frac{u_1(L_t, X_t; \zeta^{t-1})}{\pi_1}, \dots, \frac{u_m(L_t, X_t; \zeta^{t-1})}{\pi_m}]^T$
- $\zeta^{(t-\frac{1}{2})} = \zeta^{(t-1)} + t^{-1}[\frac{w_1(X_t, \zeta^{(t-1)})}{\pi_1}, \dots, \frac{w_m(X_t, \zeta^{(t-1)})}{\pi_m}]^T$

算法名称中的 self – adjusted 的机制是这样的，当我们考虑迭代过程的每一步的时候，新的迭代结果ζ_j^{t-1}相对于标准值 ζ_j^* 有一个偏置量，那么我们对于下一次的更新就使用这个偏置结果就可以产生一个自适应的过程，不断地收敛到平稳分布。

4.Method

下面简述所有算法的具体实现过程。

1）AIS

为了能够尽可能地消除误差的影响，每次运行迭代的过程都同时处理了 10 组数据，并且把迭代过程中这 10 组数据的均值和方差记录下来，来判断迭代过程的收敛情况。

该算法的主要部分是使用 gibbs 进行 global-jump 的全局采样，由于每一次的采样结果只依赖于上一次的结果，所以计算的过程较为简单。

整个计算过程都是在对数的形式下进行的，所以最后得到了Z_A与Z_B的比值也是对数形式的，需要进行求和处理：

$$\frac{Z_B}{Z_A} \approx \frac{1}{M}\sum_{i=1}^{M} w_{\text{AIS}}^{(i)} = \hat{r}_{\text{AIS}}$$

迭代的标准是温度的倒数 β 的增大，经过前期的测试发现，在一开始温度较高的时候均值和方差的变化十分明显，但是后期需要较长时间趋于稳定，所以最终采用分段式的迭代方法：

β = [0： 0.001： 0.5 0.5： 0.0001： 0.95 0.95： 0.000001： 1]

各个模型的最佳迭代次数如下表所示

model parameter	iteration number
h10.mat	iter = 55000
h20.mat	iter = 55000
h100.mat	iter = 55000
h500.mat	iter = 55000

表 1　AIS 算法迭代次数

2）RTS

RTS 算法的实现思路与 AIS 有着本质上的不同，AIS 采用的 global-jump 式的采样方式的效率是非常低的，在这种全局随机的采样下想要平稳分布需要大量的时间；而 RTS 采用的是属于 local-jump 的采样，即在采样过程中有一个强迫收敛的标准，实现过程中这个标准使用的是一个均匀分布：

$$r_k \sim Uniform(1,K) \quad k = 1,2,3 \ldots \mathrm{K}$$

在此条件下的采样将会快速地得到一个平稳的收敛结果。

迭代的过程中互相更新的两个变量是 v 向量和 β，更新公式如下：

$$[\mathrm{v},\mathrm{h}] \sim gibbs\ sampler(v_{pre}, a, b, W)$$

$$\beta \sim \frac{p(v)r_k/Z_k}{\sum_{k=1}^{K} p(v)r_k/Z_k}$$

由于不同的参数模型的收敛速度和效率有所差别，为了达到较好的结果并提高效率，在迭代过程中使用下表中的迭代次数：

model parameter	iteration number
h10.mat	iter = 40
h20.mat	iter = 60
h100.mat	iter = 40
h500.mat	iter = 60

表 2　RTS 算法迭代次数

在实现过程中由于对 β 进行采样时需要计算一个总的归一化常数，当模型数量太大（h500.mat）时会大大降低计算的效率，这也是该算法的一个缺点，在最后会有所分析。

实现过程中的 gibbs 采样次数取为 N = 30，温度倒数细分间隔为 K = 100。

3）TAP

TAP 算法的实现过程相对于上面两种来说较为简单，因为虽然本身蕴含了退火的思想，但是并没有真正地去实现一个迭代的过程，而是直接通过求解 TAP 方程来逼近最终的结果。

算法的主要实现即为如下的约等式，可以利用迭代生成的m^v、m^h 求得勒让德变换形式下的二阶近似自由能：

$$\Gamma(\mathbf{m^v},\mathbf{m^h}) \approx -S(\mathbf{m}^v,\mathbf{m}^h) - \sum_i a_i m_i^v - \sum_j b_j m_j^h - \sum_{i,j} W_{ij} m_i^v m_j^h + \frac{W_{ij}^2}{2}(m_i^v - (m_i^v)^2)(m_j^h - (m_j^h)^2)$$

在使用二阶近似实现后发现在 h10 和 h20 模型中会出现较大幅度的振荡，显然是因为展开过程的截断量不满足足够小的条件，因此又增加了三阶项来修正这个振荡：

$$-\beta\Gamma(\mathbf{m}) = -\sum_i [m_i \ln m_i + (1-m_i)\ln(1-m_i)] + \beta \sum_i a_i m_i + \beta \sum_{(i,j)} W_{ij} m_i m_j + \frac{\beta^2}{2} \sum_{(i,j)} W_{ij}^2 (m_i - m_i^2)(m_j - m_j^2) + \frac{2\beta^3}{3} \sum_{(i,j)} W_{ij}^3 (m_i - m_i^2)\left(\frac{1}{2} - m_i\right)(m_j - m_j^2)\left(\frac{1}{2} - m_j\right)$$

此时的求导结果也会有所变化，展开到三阶形式下的交叉迭代方程的形式如下：

$$v_i[t+1] = sigmod(a_i + \sum_i w_{i,j} h_j[t] - (v_i[t] - \frac{1}{2}) w_{i,j}^2 \, (h_j[t] - (h_j[t])^2) + 2\sum_{i,j} (W_{i,j}^3 * \left(h[t+1]^2 - h[t+1] + \frac{1}{6}\right) * (v[t] - v[t]^2) * (\frac{1}{2} - v[t])))$$

$$h_i[t+1] = sigmod(b_i + \sum_j w_{i,j} v_j[t+1] - (h_i[t] - \frac{1}{2}) w_{i,j}^2 \, (v_j[t+1] - (v_j[t+1])^2) + 2\sum_{i,j} (W_{i,j}^3 * \left(v[t+1]^2 - v[t+1] + \frac{1}{6}\right) * (h[t] - h[t]^2) * (\frac{1}{2} - h[t])))$$

在实现过程中由于在对数形式下进行的运算，所以有可能出现 log(0)的不可计算情况，对于这种情况不予以计算直接舍去。

model parameter	iteration number
h10.mat	iter = 40
h20.mat	iter = 15
h100.mat	iter = 150
h500.mat	iter = 35

表 3　TAP 算法迭代次数

4）SAMS

由于 SAMS 算法包含了多种实现的方式，故只选取了其中的一部分组合方式实现。

该算法的第一部分采样过程采用 global-jump，为了简单起见直接使用 RTS 中的采样函数。主要的不同之处在于第二部分的更新迭代方式，文中提供了三种可以选择的方式，一开始尝试实现了第二种方式，但是发现尽管结果精确度较高，但是迭代的收敛速度非常慢，故改为第二种更新策略，下表为两种方法得到近似准确值时的时间比较：

	Theorem 1.	Theorem 2.
h10	15.9357	159.6754
h20	20.9373	396.3467
h100	137.2318	802.274518
h500	413.8776	4197.401509

表 4　SAMS 算法两种实现方式比较（单位：s）

不管采用哪一种更新策略，在实现的过程中都需要使用增益系数（gain factor）来限制最后收敛的过程，而这个增益系数t^{-1}的选取可以采用一种 burn-in 的方式进行分段处理，这样做的好处是可以加快一开始非稳定态的迭代速度，从而快速地进入相对稳定的解空间，增益系数的获得方式如下：

$$\min(\pi_j, t^{-\beta}) \quad if\ t \le t_0$$

$$\min[\pi_j, (t - t_0 + t_0^{\beta})^{-1}] \quad if\ t \le t_0$$

其中的 β 取值范围应该在 0.5 ~ 1 之间，也就是说在退火的后半段进行精确的迭代过程，采取的值为 0.5。t_0指的是 burn-in 过程的大小，这里采取总迭代次数的 80%。

下面得到的数值结果均采用 Theorem 1 获得，具体模型的迭代次数如下：

model parameter	iteration number
h10.mat	iter = 5000
h20.mat	iter = 5000
h100.mat	iter = 8000
h500.mat	iter = 4000

表 5　SAMS 算法迭代次数

5.Analysis of Method

对于上述的四种算法进行简要的分析和比较，比较的内容包括方法的性能、收敛性、稳定性和适用性。

1）AIS 与 RTS 的性能比较

AIS 和 RTS 算法都是基于模拟退火的思想，在高温时寻找全局最优解，随着温度的降低使最优解稳定，但不同的是 AIS 是人为规定了温度的变化方向为降低方向，并采用 MCMC 的方式利用各个温度构造了一个马尔科夫转移链，该链的最终状态就是我们需要的稳定状态。而 RTS 则是通过对温度采样的

方式整体更新整个转移链，在所有更新结束之后就可以得到稳定的状态。

上述两种方法的采样过程也有本质的不同，在 AIS 方法中我们只是利用了$T_k(x_{k+1} \leftarrow x_k)$这样一个转移的过程，在全局范围内进行采样，显然这种效率是很低的，要达到稳定的解需要大量的采样点，并且可以预见的是最后达到稳定的过程也将是非常缓慢的。但是正是因为迭代过程是缓和的，不会产生剧烈的振荡来干扰迭代过程，所以理论上来讲，在运行了足够长的时间之后所得到的解应当是十分接近真实值的。

RTS 方法的采样过程是一个“有目标”的过程，也可以说是一种局部采样，这样做的好处是能够快速地进入稳定解的区间，不必进行大量的采样。但是随之而来的问题是 RTS 在更新时是 N 次 gibbs 采样后同时更新整个Z_K这个马尔科夫链的，那么出现的问题就是整个迭代过程会出现较大的波动，在解的稳定性方面不够理想。

2）RTS 与 SAMS 比较

上面的分析中已经提到，RTS 采用的是一种 local-jump 的采样方式，有效地缩短了采样时间，但是其存在的缺陷是迭代过程不稳定，容易形成周期振荡。SAMS 中的更新策略有效地解决了这一问题，其思想是一次只更新一个Z_k，并且每次更新的都是对于收敛影响最大的点，这个点的选取则来自于 RTS 方法中的抽样。

之所以一次循环只更新一个Z_k可以得到更好的结果是因为这种 SA（self-adjust）方法能够稳定地向收敛值靠拢，呈现完美的对数收敛曲线，当然，这种较好的结果也得益于 burn-in 的分阶段约束方法，这个方法之后会详细讨论。

形象地来说，RTS 再每次更新时可能因为变动太多而“超过”了收敛值，来来回回形成振荡；但是 SAMS 可以缓慢地逼近收敛值，并结合一定的算法提高逼近速率。

3）TAP 的准确性分析

TAP 算法与其他三种算法不太相同，虽然也是基于一种成熟的物理模型，但是并不是采用采样的方式来求解，而是通过一个精确的 TAP 方程，利用导数为零的稳定态得到两个交叉迭代的方程，最后将迭代之后的结果带入一个三阶截断的方程中。

显然这种方式的效率是非常高的，相比于其他三种采样的算法可以在很短的时间内收敛，但是随之而来的问题是，由于迭代的过程中通过求导得到的解并不一定是全局的最小解，该方法很有可能陷入局部最小解，并且由于交叉迭代的步长较长，最后的解可能会导致周期振荡。

后面的数值结果部分就暴露出了该方法的这些问题，在牺牲正确率的前提下可以大大提高求解的速率。

6.Algorithm

对于四种算法的伪代码与流程图进行了整理，以便于进行比较。

1）AIS

Algorithm 2 Annealed Importance Sampling 算法

- 在 0 < β_0< β_1 … < β_K < 1 中选择一个β_k
- 从 $P_A = P_0$ 中抽样出x_1
- for k = 1： K-1 do

  通过$T_k(x_{k+1} \leftarrow x_k)$在给定 x_k 的条件下抽样得到 x_{k+1}

  end for

- 令 ω = $\prod_{k=1}^{K} \frac{P_k^*(x_k)}{P_{k-1}^*(x_k)}$

该算法的迭代过程如下，每次迭代由两个隐变量h^A、h^B共同决定下一个可见节点 v 的参数，从而形成一个迭代的马尔科夫链，示意图如下：

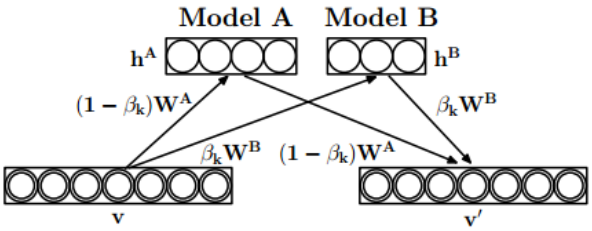

图 4 AIS 状态转移过程

2）RTS

Algorithm 3 Rao-Blackwellized Tempered Sampling

1）输入采样次数 N 和$\{\beta_k,\ r_k\}$，k = 1, 2⋯ K

2）初始化 $\log Z_k$ k = 2, 3⋯K

3）初始化 β ∈ { $\beta_1 \ldots \beta_K$ }

4）初始化$c_k = 0,\ k = 1,2, \ldots K$

5）for i = 1 : N

- 利用 q(x| β)对 x 进行采样，直到得到稳定的 x
- 在给定 x 的条件下对 β 进行采样
- 更新$c_k = c_k + \frac{1}{N}q(\beta|x)$

6）更新 $Z_k^{RTS} = Z_k * \frac{r_1 c_k}{r_k c_1}$ k = 2,… K

该算法在初始化Z_k时默认全部的元素初始为零，因为初始的数值对最终的结果没有影响。

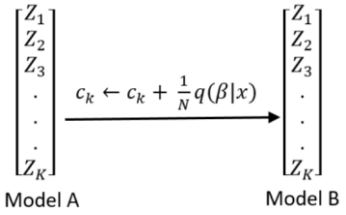

图 5 RTS 状态转移过程

状态的转移过程如上图所示。

3）TAP

Algorithm 4 Thouless-Anderson-Palmer

1）设置迭代次数 N

2）随机选取 v[0]和 h[0]

3）for t = 1 : N

- $v_i[t+1] = sigmod(a_i + \sum_i w_{i,j}h_j[t] - (v_i[t] - \frac{1}{2})w_{i,j}^2\ (h_j[t] - (h_j[t])^2))$
- $h_i[t+1] = sigmod(b_i + \sum_j w_{i,j}v_j[t+1] - (h_i[t] - \frac{1}{2})w_{i,j}^2\ (v_j[t+1] - (v_j[t+1])^2))$

end for

4）计算标准的平均场能量(MF)：

$$F = a^T v + b^T h + v^T W h$$

5）计算熵

$$S = v\ln v + (1-v)\ln(1-v) + h\ln h + (1-h)\ln(1-h)$$

6）计算 Onsager 修正项

$$F^{Onsager} = [(h-h^2)^T W^2 (v-v^2)]$$

7）计算三阶修正项

$$F^{three} = \frac{2\beta^3}{3}\sum_{(i,j)} W_{ij}^3\ (m_i - m_i^2)(\frac{1}{2} - m_i)(m_j - m_j^2)(\frac{1}{2} - m_j)$$

8）计算总自由能量

$$F^{free} = F - S + \frac{F^{Onsager}}{2} + F^{three}$$

4）SAMS

Algorithm 5 Self-adjusted mixture sampling

1）Labeled mixture sampling：

利用转移核函数采样（L_t, X_t）

- local-jump
  $(L_t, X_t) \sim K_\theta(y_{t-1}, y_t) = p_{L-J}(l_t|l_{t-1}, x_{t-1};\ \zeta^{t-1})p(x_t|l_t, x_{t-1})$
- global-jump
  $(L_t, X_t) \sim K_\theta(y_{t-1}, y_t) = p_{G-J}(l_t, x_{t-1}; \zeta)p(x_t|l_t, x_{t-1})$

2）Free energy update：

采用三种不同的方式进行更新

- $\zeta^{(t-\frac{1}{2})} = \zeta^{(t-1)} + t^{-1}[\frac{\delta_1(L_t)}{\pi_1}, \ldots, \frac{\delta_m(L_t)}{\pi_m}]^T$
- $\zeta^{(t-\frac{1}{2})} = \zeta^{(t-1)} + t^{-1}[\frac{u_1(L_t, X_t; \zeta^{t-1})}{\pi_1}, \ldots, \frac{u_m(L_t, X_t; \zeta^{t-1})}{\pi_m}]^T$
- $\zeta^{(t-\frac{1}{2})} = \zeta^{(t-1)} + t^{-1}[\frac{w_1(X_t, \zeta^{(t-1)})}{\pi_1}, \ldots, \frac{w_m(X_t, \zeta^{(t-1)})}{\pi_m}]^T$

3）重复上述过程，直至收敛

7.Numerical Result

所有算法的实现语言均为 MATLAB，软件平台为 MATLAB R2014a，测试使用的数据集为 MNIST 手写数据集，MNIST 数据集包含 60,000 个训练数据和 10,000 个测试数据，每一个图片的尺寸为 28×28 。

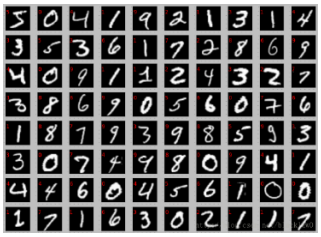

图 6 MNIST 数据集实例

数据的内容包括 0~9 是个手写数字，并且每一张图片的表示形式都是二值图（website[3]）。

在估计分割函数时使用的模型参数分别来自于四个文件：h10mat、h20.mat、h100.mat、h500.mat。以下所有结果均来自于上述四个模型参数。

1）AIS 运行结果

模型参数	h10	h20	h100	h500
Z 估计值	226.2469	220.9765	348.2846	459.2379
方差	0.0542	0.0754	0.3565	0.9114
所用时间	92.8389	114.4000	309.9933	653.4227

表 6　AIS 算法结果比较（单位：s）

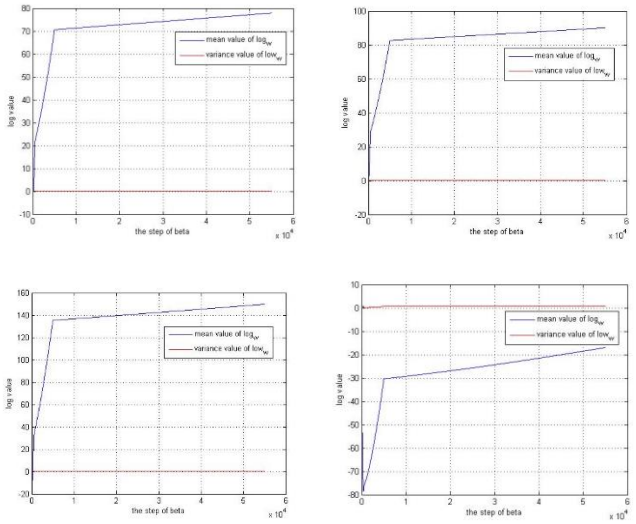

图 7　SAMS 算法迭代趋势

上图是一组（10 个）数据的迭代过程中的归一化常数 Z 的对数形式的方差（红线）和平均值（蓝线）的变化趋势。

2）RTS 运行结果

模型参数	h10	h20	h100	h500
Z 估计值	224.8172	219.0418	348.2256	449.5428
方差	2.0584	3.2571	15.5975	20.7845
所用时间	23.3928	81.1876	285.9873	466.6034

表 7　RTS 算法结果比较（单位：s）

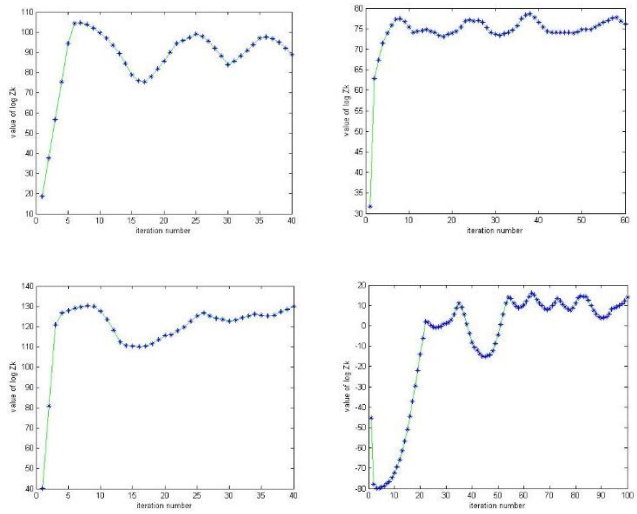

图 8　RTS 算法迭代趋势

上图是一组数据迭代过程中的最终状态的 $\log Z_B(K)$ 的对数形式的变化趋势。

3）SAMS 运行结果

模型参数	h10	h20	h100	h500
Z 估计值	225.8317	221.9415	348.3157	456.6511
方差	0.2514	0.0125	0.0245	2.1674
所用时间	15.9357	20.9373	137.2318	413.8776

表 8　SAMS 算法结果比较（单位：s）

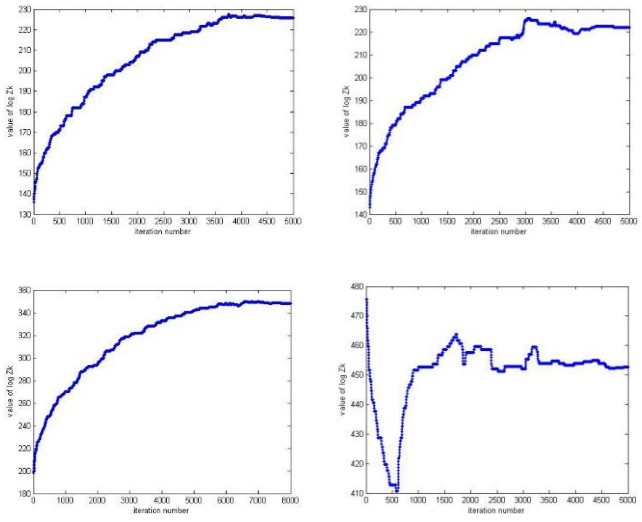

图 9　SAMS 算法迭代趋势

上图是四个模型的迭代过程中的归一化常数 Z 的对数形式的变化过程。

4）TAP 运行结果

模型参数	h10	h20	h100	h500
Z 估计值	225.9876	207.2171	335.8317	449.1477
方差	5.0654	0	0.5978	0
所用时间	1.6211	1.8851	27.5373	29.6302

表 9　TAP 算法结果比较（单位：s）

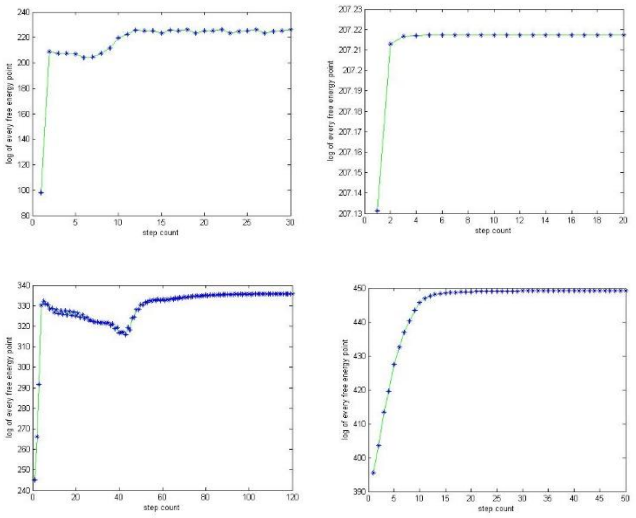

图 10　TAP 算法迭代趋势

上图是四个模型的迭代过程中的归一化常数 Z 的对数形式的变化过程。

5）综合参数比较

下面给出四种算法的最大似然值比较、迭代次数比较、运行时间比较，基于这些比较结果进行分析。

	h10.m	h20.m	h100.m	h500.m
AIS	1.1434^{-31}	6.6466^{-21}	8.4886^{-14}	1.1113^{-12}
RTS	5.8578^{-31}	7.1371^{-20}	9.7838^{-14}	2.3595^{-8}
SAMS	1.8173^{-31}	9.7482^{-15}	2.3611^{-8}	3.5028^{-8}
TAP	2.1239^{-31}	3.9283^{-21}	8.9408^{-14}	1.9307^{-11}

表 10 最大似然值比较

	AIS	RTS	SAMS	TAP
h10.m	55000	40	5000	40
h20.m	55000	60	5000	20
h100.m	55000	40	8000	120
h500.m	55000	60	4000	35

表 11　迭代次数比较

	AIS	RTS	SAMS	TAP
h10.m	92.8389	23.3928	15.9357	1.6211
h20.m	114.4000	81.1876	20.9373	1.8851
h100.m	309.9933	285.9873	137.2318	27.5373
h500.m	653.4227	3866.6034	413.8776	29.6302

表 12　运行时间的比较（单位：s）

8.Discussion

下面对上述数值结果进行理论上的分析，并以此作为基础再次比较各个算法的性能。

1）AIS

从 AIS 算法的均值曲线趋势可以看出，随着迭代次数的增长，均值还会有缓慢的变化，方差基本趋近于零。但是该增长过程的时间太长，并且后期的变化十分缓慢，所以在实际运行部运行的时候只取前面的主要变化的迭代次数来提高效率。

但是从算法流程和结果分析上看 AIS 算法理论上在足够长的时间后可以得到一个足够精确的值，所以之后的分析都以 AIS 算法的结果作为真实值。

2）RTS

与之前的分析相符，RTS 算法虽然相比于 AIS 能够更快地收敛，但是最终的收敛值会产生较大幅度的振荡，这是由于多个Z_K同时更新的结果，理论上来说在足够长的时间后也可以得到精确解。

由于采用了分段的温度下降速度，所以一开始的迭代过程有一个快速的上升，迅速到达收敛值附近的解，这对于算法效率的提升有很大帮助。

但是迭代解的波动确实会对解的精确性造成很大的影响，与 AIS 标准最大似然值比较会发现误差较大。

3）SAMS

SAMS 算法是一种相对完善的算法，所以其迭代曲线呈现了较好对数形式，可以看到，前三个模型中都是稳定地收敛到一个较好的结果，第四个模型虽然有所波动，但最终稳定收敛。

与 AIS 标准最大似然值的比较也可以看出与标准值基本一致，但是运行速度却比 AIS 算法大大提高了，所以该算法应该是四个算法中性能最佳的。

4）TAP

从趋势图中可以看出，该算法在后三个模型中都可以收敛到一个稳定的值，h10 模型有微小的振荡。

但是在 h20、h100 和 h500 模型中的结果与标准值（取 AIS 为标准）差距较大。但观察到后面这三个模型的偏差大概都差一个同常数，可能是实现过程存在问题。

总而言之，尽管这种算法是四种算法中速度最快的，但是其准确性不是特别可靠，不太适合作为精确计算的结果。

9.Conclusion

1）方法总结

经过上述的比较和计算过程，我们可以发现在关于计算归一化常数的这个问题上，各个算法中有一些关键性的提高，不论是在精确度还是在时间效率上，大致有如下三个方面：

- burn-in 方法

这种不等步长分段迭代的方式在模拟退化的算法中十分有效，甚至说是在其他迭代求解的过程中也适用，因为一开始的起始点的选取具有很强的随机性，也就是说一开始的几次迭代非常不准确，我们需要尽快找到一个小范围的解空间进行更加细致的迭代。

但是分段区域设置的不合理也会造成振荡甚至误差严重的问题，因为如果一开始的迭代步长太大也可能进入局部最小解，后面迭代次数不够也容易产生误差。

这种方法在不只 SAMS 算法中应用，在 AIS 的实现过程中也将 β 进行了区域的分段。

- 单次迭代取平均解

我认为这种迭代过程的内部收敛方法是一种保证收敛性和提高运行效率的非常好的方式，在迭代过程的每一步中都进行一定的收敛性处理或者同时迭代多组数据取均值，对于整体而言就可以避免漫长的全局收敛过程。

- local-jump 方法

这是在 MCMC 基础上的一个改进，传统的 MCMC 尽管是利用了马氏链的平稳性，但是全局的采样会造成太多无用状态的浪费，而局部限制下的采样方式可以有效地避免贡献极小的采样点，使结果快速地迭代到理想值。

2）最优算法

综合上述四种算法，我认为 SAMS 在时间效率和准确度上是最佳的算法，当然其余的算法也各有优点，其中的 TAP 算法在思想上突破了采样的方法，但是限于个人时间的原因，没有深入地探讨其中的深层物理模型和改进方式，目前还无法解释其对于后面三个模型的结果偏差较大的原因，不排除是模型参数本身的问题。

3）补充说明

在 MCMC 方法的基础上进行上述改进的这一类算法同样可以应用到其他大量数值计算问题上，尽管这些方法都或多或少有些问题，但是在现实中类似于本问题中的精确求解是几乎不可能完成的任务，所以 MCMC 方法的发展对于很多计算问题有着重要的贡献。

由于对于各个论文的算法理解可能有所偏差，所以造成实现过程引入了误差，在比较结果时已经尽可能多的对算法进行了改进，但限于时间的原因没有能够更多进行合理的优化，望指正。

10.Acknowledgement

[1] 感谢电子系无 48 班石莱茜同学，与她进行了 RTS 算法方面的探讨使我对于该算法了解更加深入。

[2] 感谢电子系博士生赵昊学长，帮助我了解深度学习的背景知识和 RBM 的训练框架。

11.Reference

Articles

[1] Persi Diaconis《The Markov Chain Monte Carlo Revolution》

[2] Hastings, W.K. 《Monte Carlo Sampling Methods Using Markov Chains and Their Applications》

[3] Christophe Andrieu, Nando De Freitas, Arnaud Doucet and Michael.Jordan. 《 An Introduction to MCMC for Machine Learning》

[4] Ruslan Salakhutdinov.《LEARNING DEEP GENERATIVE MODELS》

[5] David E. Carlson, Patrick Stinson, Ari Pakman, Liam Paninski.《Partition Functions from Rao-Blackwellized Tempered Sampling》

[6] Marylou Gabrie Eric W. Tramel Florent Krzakala 《Training Restricted Boltzmann Machines via the Thouless-Anderson-Palmer Free Energy》

[7] Zhiqiang Tan. 《Optimally adjusted mixture sampling and locally weighted histogram analysis》

[8] Geoffrey Hinton《A Practical Guide to Training Restricted Boltzmann Machines》

[9] Radford M. Neal《Annealed Importance Sampling》

websites

[1] http://www.tuicool.com/articles/jyqIfe6

[2]https://charlesmartin14.wordpress.com/2016/10/21/improving-rbms-with-physical-chemistry/

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