题目：马氏链蒙特卡洛方法

系别：电子工程系

专业：电子信息科学

姓名：牛源蕾

摘要

近年来，深度学习有了长足的发展，在生活中的应用方面（图像识别等）有着广阔的前景。作为深度学习的基础模型之一，受限玻尔兹曼机（Restricted Boltzmann Machine, RBM）的概率分布计算是非常重要的。但是，概率分布的计算涉及到归一化常数的估计，归一化常数估计的准确性会对模型的可靠性产生非常重大的影响。作为马氏链在工程领域的应用，马氏链蒙特卡洛方法（Markov Chain Monte Carlo, MCMC）很好的提供了随机抽样的方法，其在 RBM 归一化常数估计、二维高斯分布相关系数的估计等领域发挥了重要的作用。

本文首先探讨了 MCMC 算法在估值仿真中的应用，研究了二维高斯相关系数的估计，实现 Metropolis-Hastings(MH)算法，对给定的二维高斯分布进行随机采样，使用随机生成的样本估计二维高斯的相关系数。并对比了不同的抽样次数下的估计值的不同，分析原因。

随后，针对 RBM 模型进行归一化常数的估计。RBM 是深度学习最基础最重要的模型之一。通常有四种归一化常数的方法：Annealed Importance Sampling(AIS)、Thouless-Anderson-Palmer(TAP)、Rao-Blackwellized Tempered Sampling(RTS)和 Self-adjusted mixture sampling(SAMS)算法。其中，AIS、SAMS 和 RTS 方法均体现了 MCMC 的思想，但是每次更新的方式有所不同。本文融合 MCMC 思想，针对不同算法灵活运用 Metropolis-Hastings 算法及 Gibbs 采样，得到 RBM 模型归一化常数估计值。对比不同算法归一化常数结果，分析不同隐藏变量数目对于算法速度和精度的影响，以及算法实现过程中采样方式对于算法精度和速度的影响，给出最佳的估计。

关键词：马氏链蒙特卡洛方法；二维高斯相关系数；深度学习；受限玻尔兹曼机；归一化常数

目录

1.引言

1.1 课题背景和研究意义

MCMC 是进行仿真估值非常常用的方法。蒙特卡洛方法是产生高维随机数的有效方法[1]，结合马氏链的特性而产生的 MCMC 方法在对归一化常数估计中发挥了很重要的作用。

在不久以前，谷歌的机器人 AlphaGo 成功打败了韩国围棋选手李世石，轰动了世界，深度学习也由此进入越来越多人的视野。现在的深度学习在例如模式识别、计算机视觉和语音识别等方面都有着非常成功的发展，在生活中有着十分广阔的应用前景。RBM 作为深度学习最基础的模型之一，是更深入探索深度学习的基础。其中，对 RBM 的归一化常数估计非常重要。

人们基于 RBM 的物理学模型背景，结合 MCMC 方法，提出了 RBM 归一化常数的算法。追求算法的速度和准确性将对 RBM 模型深入研究带来很大帮助。

1.2 研究现状

在过去的时间里，有一些研究者已经认识到了 RBM 在深度学习领域的巨大潜力并开始研究这方面的内容。本文主要关注在 RBM 归一化常数估计算法的相关研究。

2009 年，Ruslan Salakhutdinov 在博士论文《Learn Deep Generative Models》[2]中提出了 Annealed Importance Sampling(AIS)算法，给出了给定样本下的归一化常数估计，得到均值和标准差，在误差允许的范围内，每次随机估计的结果可以认为是一致的，因此 AIS 方法估计得到的归一化常数估计值有一定的可靠性。文中还提到可以改变β分布的方式对算法进行优化，但是并没有给出具体的优化策略，也没有对性能的提升进行对比。

Marylou Gabri´e、Eric W. Tramel 以及 Florent Krzakala 撰写的《Training Restricted Boltzmann Machines via the Thouless-Anderson-Palmer Free Energy》[3]文中提到了 AIS 算法的种种不足：过于保守、收敛性质不佳、算法的效率低下。并声称自己提出的 Thouless-Anderson-Palmer(TAP)算法具有更加优越的收敛性和收敛速度。但是经本工作实验验证，TAP 算法在具有上述优势的同时，具有无法消除的截断误差，并且这种误差并不能通过保留更多的级数项而得到有效的降低。

2016 年，哥伦比亚大学的 David E. Carlson、Patrick Stinson、Ari Pakman、Liam Paninski 在《Partition Functions from Rao-Blackwellized Tempered Sampling》[4]一文中采用了引入辅助分布的方式对归一化常数进行估计，即引入了 AIS 方法中β的随机性，提出了 Rao-Blackwellized Tempered Sampling (RTS)算法。作者认为 RTS 的速度和精度都高于 AIS 算法。但在本工作对 RTS 的实现中，未能得出 RTS 性能远高于 AIS 的结论。

谭志强在 2014 年《Optimally adjusted mixture sampling and locally weighted histogram analysis》[5]文中提出了 Self-adjusted mixture sampling(SAMS)算法，该算法与 RTS 算法相似之处在于：同样引入了辅助分布。SAMS 算法具有自适应的特性，因此收敛的精度比较理想。但是 SAMS 算法随着迭代次数的增加，会出现收敛速度严重下降的问题，会导致运行时间极长。文中针对这一问题给出了优化方案，但是并没有进行前后对比。

总之，在现有的工作中，大多数的算法都具有每一方面比较突出的特性，但是又会在另外的方面存在明显的不足，具有一定的提升空间。并且在现有的工作中，没有将各种算法的性能进行综合的对比分析。因此，必须要针对各个算法的不足之处，通过调整参数、调整迭代次数、改变迭代公式等方法进行优化，并且将不同的算法进行对比。这样，才能得到各方面性能比较均衡的最佳的选择。

1.3 研究内容及主要贡献

本工作的研究目标是针对现有的 AIS、TAP、RTS、SAMS 四种归一化常数估计算法，通过改变辅助分布、改变迭代次数、优化迭代公式的方法，进行一定程度上的优化，得到更加准确可靠的归一化常数估计值。

本文工作主要分为三个部分：

1. 二维高斯的相关系数估计：利用 MCMC 对给定的高斯分布的相关系数进行估计。改变采样次数，分析采样次数对于相关系数估计值准确性的影响。采样时分别采用均匀分布和高斯分布生成采样点，比较其对估计值的影响；
2. RBM 模型归一化常数估计：分别对 AIS、TAP、RTS、SAMS 算法进行实现，并用每种算法计算四个不同 RBM 模型的归一化常数估计值。并针对算法特点进行分析，对各种算法进行不同程度的优化；
3. 不同模型对比分析：针对不同模型估计的归一化常数结果的准确性及效率进行对比分析，并分析不同算法在原理上的异同。

1.4 本文结构

在接下来的第二章中，将对 MCMC 进行介绍，并应用 MCMC 对给定的二维高斯分布进行相关系数的估计；第三章实现 AIS、TAP、RTS、SAMS 四种不同的算法，对给定的四个 RBM 模型进行归一化常数的估计，并进行算法分析；而四种算法的对比将在第四章进行分析，最后，将在第五章对整篇论文进行总结并对未来进行展望。

2.二维高斯的相关系数估计

2.1 马尔科夫链及 MCMC 介绍

马尔科夫链（Markov Chain）的一般理论框架可以参见《Statistical Digital Signal Processing and Modeling》（Hayes,1996）[6] 和《Probability, Markov Chains, Queues, and Simulation》(William J. Stewart ， 2013)[7] ，此处直接应用其相关性质。

蒙特卡罗算法往往涉及模拟观测和期望估计，以及多元分布的归一化常数估值，其精髓在于使用随机数（或更常见的伪随机数）来解决很多计算问题。而 MCMC 是马氏链理论与蒙特卡洛算法结合的一个重要应用。

蒙特卡洛算法的一个重要应用是估算积分，如下例，计算积分:

$$\int_a^b g(t)dt$$

蒙特卡洛思想是是把该积分转化为求某一概率密度$f(x)$下的期望，从而积分估值问题转化为从已知目标概率密度$f(x)$中产生随机样本的问题。其基本步骤就是产生(伪)随机数，使之服从该概率分布$f(x)$，再求出期望。难点就在于$f(x)$的产生。当变量 X 是一维的情况时，这很容易做到。但如果变量 X 取值于R^n，直接产生符合某一分布的独立样本通常是很难的。这就引入了 MCMC 方法：

MCMC 方法由 Metropolis(Metropolis.et.a1.1953)[8] 奠定基石，他提出在以π为平稳分布的马氏链上产生相互依赖的样本，进而马氏链的值可以看成是从分布π中抽去样本，从而可以根据遍历定理对积分进行估计。之后由 Hastings 对其加以推广形成了，Metropolis-Hastings(MH) 算法。换句话说，MCMC 方法是在蒙特卡洛产生随机样本时，样本产生根据一条马氏链的跳转概率来迭代更新。

2.2 Metropolis-Hastings 算法

对于下述二维高斯分布：

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

我们有μ= $\begin{pmatrix}5\\10\end{pmatrix}$，Σ=$\begin{pmatrix}1&1\\1&4\end{pmatrix}$，可立得$\begin{pmatrix}x_1\\x_2\end{pmatrix}$理论相关系数ρ=$\sqrt{\frac{\sigma_{12}\times\sigma_{21}}{\sigma_{11}\times\sigma_{22}}}$=0.5。(证明详见附录 A)

采用 MCMC 算法进行模拟，其核心是建立一个平稳分布为$p(x)$的马氏链来得到$p(x)$的样本分布。针对所给二元高斯分布，MH 的采样步骤可以总结为以下流程：

1. Set i = 1；
2. Generate initial value $X^{(1)}=\begin{pmatrix}x_1\\x_2\end{pmatrix}$;
3. for i=1: N
4. Generate a $y^{(i)}$ from a proposal distribution f(x);

   (Uniform and Gauss is tested effectively)
5. Evaluate the acceptance probability

$$\alpha=\min\left(1,\ \frac{f(y^{(i)})*p(X^{(i-1)}|y^{(i)})}{f(X^{(i-1)})*p(y^{(i)}|X^{(i-1)})}\right);$$

6. Generate a u from a Uniform(0,1) distribution;
7. If u < α, accept the proposal ,set $X^{(i)}=y^{(i)}$;else set $X^{(i)}=X^{(i-1)}$。

可以证明，通过 MH 方法构造出来的链满足马氏性且以$p(x)$为平稳分布(详见附录 B)。

在上述算法中，$p(x)$是已知的，建议分布$f(x)$的选择不同，得到的估计值会有一定的差异，分别选择$f(x)$均匀分布和高斯分布，在迭代 5000、10000、50000、100000、200000 次之后比较结果，重复 20 次计算估计值的均值和标准差，时间为运行 20 次的平均值：

采样次数	5000		10000		50000		100000	
	均匀分布	高斯分布	均匀分布	高斯分布	均匀分布	高斯分布	均匀分布	高斯分布
均值	0.5107	0.5082	0.4939	0.5033	0.4988	0.5020	0.5008	0.5003
标准差	0.0384	0.0260	0.0228	0.0191	0.0106	0.0101	0.0074	0.0068
时间	1.107	1.451	2.304	2.926	11.084	14.191	22.126	28.690

表格 1

对$f(x)$每种分布而言，迭代次数增加，估计值会更加接近真实值，同时标准差更小，说明估计的精度和稳定性会更好。但是迭代次数增多会导致运行时间变长。

在相同的迭代次数下比较两种$f(x)$分布，可以看到高斯分布比均匀分布的估计结果更加准确。由此可见，在选择$f(x)$的时候，性质越接近$p(x)$，仿真的精度越高。

在相同的精度要求下，$f(x)$选择高斯分布会比均匀分布收敛速度更快，下图为采样 50000 次，两种分布的相关系数收敛曲线：

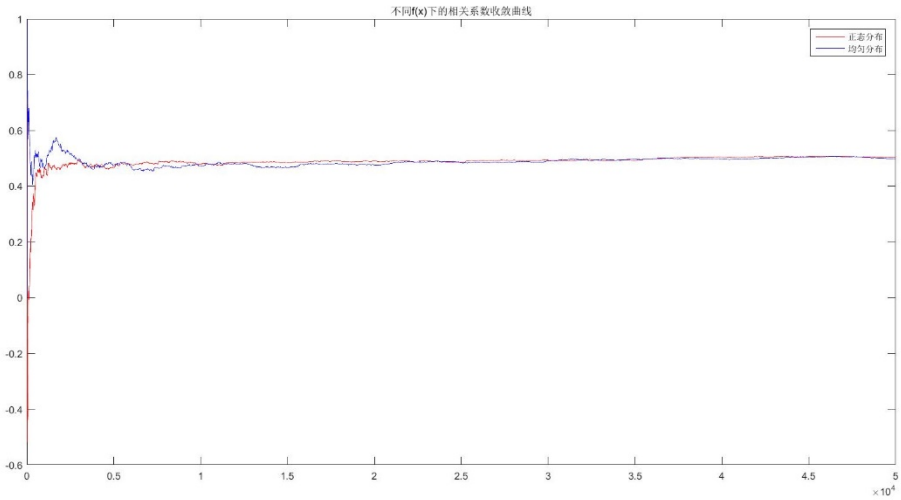

图 1 **不同**$f(x)$**下的相关系数收敛曲线**

可以从图中的曲线明显看出红线（正态分布）比蓝线（均匀分布）的收敛速

度要快，二者最终收敛到几乎同一个值。

但是采用高斯分布，算法的速度要低一些，这应当与 matlab 函数 normrnd() 和 unifrnd()本身的效率有关，在采样 50000 次的情况下，速度的差距可以接受，同时两种方法运行的时间都比较短（如果画图的话需要计算每一次迭代之后的相关系数，会大大降低运行速度），故$f(x)$为高斯分布，采样 50000 次，是比较合适的策略，估计值 0.5020，与真实值误差 0.4%。

3.RBM 模型归一化常数估计

3.1 RBM 模型介绍

受限玻尔兹曼机(Restricted Boltzmann Machin，RBM)是由 Hinton 和 Sejnowski 于 1986 年提出的一种生成式随机神经网络(generative stochastic neural network)，该网络由一些可见单元(visible unit，对应可见变量，亦即数据样本)和一些隐藏单元(hidden unit，对应隐藏变量)构成，可见变量和隐藏变量都是二元变量，亦即其状态取{0,1}。整个网络是一个二部图，只有可见单元和隐藏单元之间才会存在边，可见单元之间以及隐藏单元之间都不会有边连接，如下图所示：

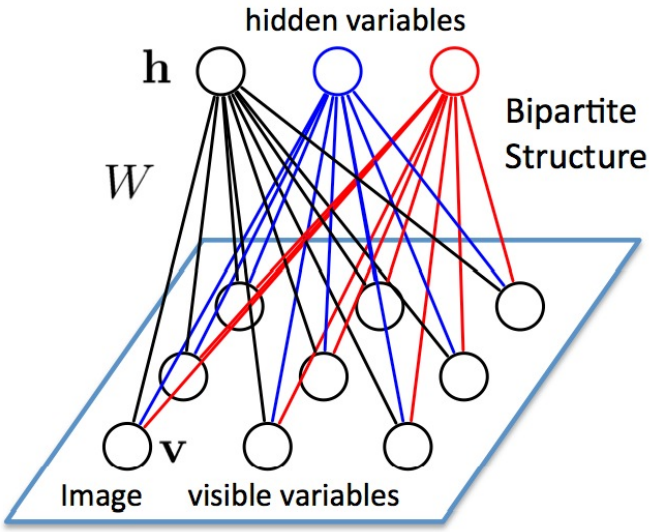

图 2

W 是一个矩阵，表示可见单元和隐藏单元之间的边的权重。

RBM 的学习目标-最大化似然(Maximizing likelihood)

RBM 是一种基于能量(Energy-based)的模型，其可见变量 *v* 和隐藏变量 *h* 的联合配置(joint configuration)的能量为:

$$E(v,h;\theta) = -\sum_{ij} W_{ij} v_i h_j - \sum_i b_i v_i - \sum_j a_j h_j \qquad (1)$$

其中 θ 是 RBM 的参数{W, a, b}, W 为可见单元和隐藏单元之间的边的权重，b 和 a 分别为可见单元和隐藏单元的偏置(bias)。

有了 v 和 h 的联合配置的能量之后，我们就可以得到 v 和 h 的联合概率：

$$P_\theta(v,h) = \frac{1}{Z(\theta)} exp(-E(v,h;\theta)) \quad (2)$$

其中 Z(θ)是归一化常数，也称为配分函数(partition function)，即本文关注的重点。

$$Z(\theta) = \sum_v \sum_h exp\,(-E(v,h;\theta)) \quad (3)$$

在对模型进行训练的时候，我们希望最大化观测数据的似然函数 P(v)，从而得到 RBM 的参数，P(v)可由 P(v,h)对 h 求和得到:

$$P_\theta(v) = \frac{1}{Z(\theta)} \sum_h exp[v^T W h + a^T h + b^T v] \quad (4)$$

本题中，对于已经给定的模型参数，我们需要做的工作仅仅是在给定模型下求得归一化常数$Z(\theta)$，并计算给定数据样本下的似然函数。需要注意的是，为了方便计算，归一化常数的估计值用自然对数表示。

在完成上述工作之后，可以尝试对模型进行重新训练，得到更好的模型参数。

3.2 Annealed Importance Sampling (AIS)

3.2.1 AIS 算法

AIS 算法的核心思想在于逼近，假设模型 A 和 B，模型参数分别为$\theta_A = \{W_A, a_A, b_A\}$和$\theta_B = \{W_B, a_B, b_B\}$。模型 B 即为给定的模型，A 模型为对 B 模型的估计，根据文章《Learning Deep Generative Models》（Ruslan Salakhutdinov，2009），A 模型的参数可以取为$W_A = 0, a_A = a_B, b_A = b_B$。在相同的 x 下，可以得到：$P_A(x) = P_A^*(x)/Z_A$，$P_B(x) = P_B^*(x)/Z_B$，其中$P^*(\ )$表示没有归一化的概率分布。根据蒙特卡洛估计，及《Learning Deep Generative Models》一文中的推导，可以

得到：

$$\frac{Z_A}{Z_B} \approx \frac{1}{M}\sum_{i=1}^{M}\frac{P_B^*(x^{(i)})}{P_A^*(x^{(i)})} \tag{5}$$

我们定义一系列中间概率分布：$P_0, \ldots P_K$，满足$P_0 = P_A$，$P_K = P_B$。我们希望利用一系列的中间概率分布逐渐从模型 A 逼近到模型 B，当相邻的两个中间概率足够接近的时候，可以得到：

$$\frac{Z_{k+1}}{Z_k} \approx \frac{1}{M}\sum_{i=1}^{M}\frac{P_{k+1}^*(x^{(i)})}{P_k^*(x^{(i)})} \tag{6}$$

其中$x^{(i)} \sim P_k$

上式累乘可以得到：

$$\frac{Z_K}{Z_0} \approx \prod_{k=0}^{k=K}\frac{Z_{k+1}}{Z_k} \tag{7}$$

根据 Neal [2001], Jarzynski [1997]估计，公式(6)中，当 M=1 时，利用马尔科夫链进行采样，可以得到$\frac{Z_K}{Z_0}$的无偏估计量，代入公式(7)，最终得到：

$$\frac{Z_K}{Z_0} \approx \prod_{k=1}^{k=K}\frac{P_k^*(x_k)}{P_{k-1}^*(x_k)} \tag{8}$$

具体的算法实现流程如下:

---

1. Select β_k with $0 = \beta_0 < \beta_1 < \cdots < \beta_K = 1$.
2. Sample x_1 from $P_A = P_0$.
3. for k = 1 : K – 1 do
4. Sample x_{k+1} given x_k using $T_k(x_{k+1} \leftarrow x_k)$.
5. end for
6. Set $w_{AIS} = \prod_{k=1}^{K} P_k^*(x_k) / P_{k-1}^*(x_k)$.

其中第 4 步，每次采样，用当前的β、v、h，利用 Gibbs 采样的方法，生成新的 v、h，Gibbs 采样中用到的条件概率按照如下的方式计算：

$$\begin{cases} p(h_j^A = 1|v) = g\left((1-\beta_k)\left(\sum_i W_{ij}^A v_i + a_j^A\right)\right) \\ p(h_j^B = 1|v) = g\left(\beta_k\left(\sum_i W_{ij}^B v_i + a_j^B\right)\right) \\ p(v_j' = 1|h) = g\left((1-\beta_k)\left(\sum_i W_{ij}^A h_j^A + b_i^A\right) + \beta_k\left(\sum_i W_{ij}^B h_j^B + b_i^B\right)\right) \end{cases} \quad (9)$$

其中$g(x)$为 sigmoid 函数：

$$g(x) = \frac{1}{1+e^{-x}} \quad (10)$$

同时，每一次迭代，要利用当前的 v计算$P_k^*(v)/P_{k-1}^*(v)$，为了简化运算，采用取对数的方式进行计算，即计算$\ln P_k^*(v) - \ln P_{k-1}^*(v)$，考虑到 A、B 模型的参数，根据《Learning Deep Generative Models》一文中的公式推导得到计算公式如下：

$$\ln P_k^*(v) = \sum_i b_i^A v_i + \sum_{j=1}^{F_A}(1 + e^{(1-\beta_k)a_j^A}) + \sum_{j=1}^{F_B}(1 + e^{\beta_k(\sum_i W_{ij}^B v_i + a_j^B)}) \quad (11)$$

对公式(8)两边取对数得到，可以得到：$\ln Z_B - \ln Z_A$，其中$\ln Z_A$的值可以利用如下公式计算：

$$\ln Z_A = \sum_j(1 + e^{a_j}) + \sum_i(1 + e^{b_i}) \quad (12)$$

最终求得Z_B

3.2.2 算法分析

在 AIS 算法中，β的分布会对估计的速度和精度产生很大的影响。

- **迭代次数的影响**

如果β符合 0~1 之间的均匀分布，则取值越密集，估计的精度越高，但是同时会带来计算速度的下降。对于给定的四种 RBM 模型，β的取值密度对于 h20 模型的估计值影响最为明显。

下述实验中均值与标准差来自 20 次估计，时间为 20 次的平均值。

当迭代次数为 10^4、10^5、10^6 次时，四个模型的归一化常数估计为：

	h10	h20	h100	h500
均值	224.9777	207.5697	347.2776	454.6772
标准差	0.6091	1.7438	1.1146	2.5073
时间/s	1.301	1.402	2.260	7.814

表格 2 迭代次数 10^4

	h10	h20	h100	h500
均值	226.1913	213.6344	348.3260	459.8581
标准差	0.6921	3.6248	0.5750	1.1364
时间/s	13.340	15.501	23.821	79.309

表格 3 迭代次数 10^5

	h10	h20	h100	h500
均值	226.0451	221.3501	348.6173	460.5572
标准差	0.2541	1.1831	1.9981	1.5623
时间/s	130.443	148.123	237.625	800.214

表格 4 迭代次数 10^6

可以看到随着迭代次数的增加，标准差的变化并不是很大，因此可以认为迭代次数对于估计值稳定性的影响不显著。h10、h100、h500 模型在迭代 10^5 和 10^6 次时的估计值差距均小于 0.16%，因此可以认为在迭代次数大于 10^5 时，迭代次数对于 h10、h100、h500 模型估计影响不显著，可以认为 10^5 迭代次数已经可以得到精度足够的 h10、h100、h500 模型归一化常数估计。

但是 h20 模型对于迭代次数敏感，迭代次数提升一个量级，估计值增加约 3%，因此，可以认为在迭代次数少于 10^6 时，归一化常数没有完全收敛，应当迭代 10^6 次，获得最佳精度的估计值。

下图为 h20 模型归一化常数估计随迭代次数增加的变化，可以看出迭代次数为 10^6 量级的时候，估计值的变化逐渐不显著，可以认为达到收敛。

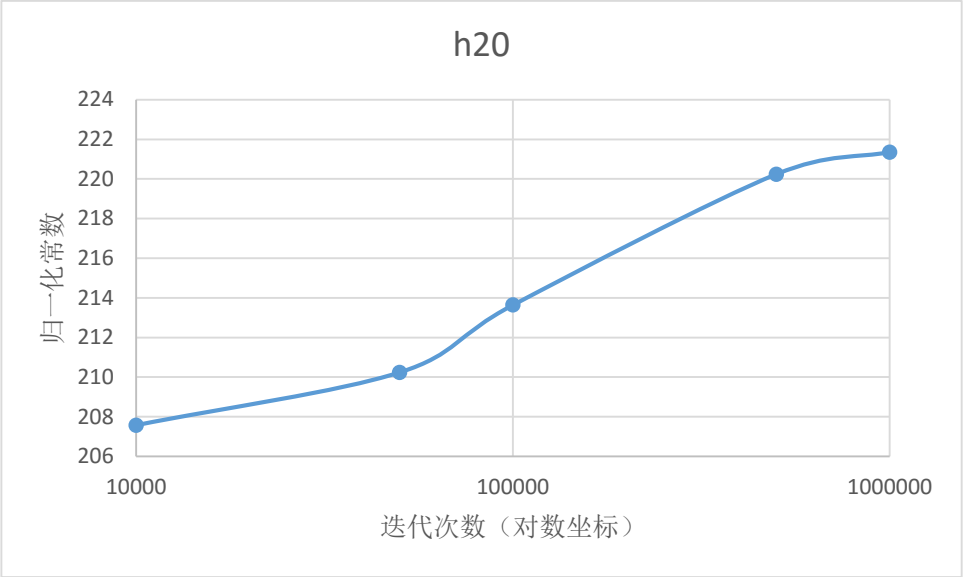

图 3

纵向对比每个 RBM 模型的归一化常数估计时间，可以看到迭代次数对时间的影响是线性关系，这一点由算法的实现原理可以解释，迭代次数的增加本身不影响每次迭代时的运算量，因此可以看到在迭代次数提升一个量级的时候，对应模型的估计时间也变为原来的十倍左右。

横向对比四个 RBM 模型归一化常数估计时间，可以看到时间与隐藏变量的数量是线性的关系，这是由于算法中的循环部分的复杂度关于输入向量的长度是线性的，即和隐藏变量的数量线性相关。

- **收敛性**

对四个 RBM 模型分别进行 10^6 次迭代，画出收敛曲线如下图：

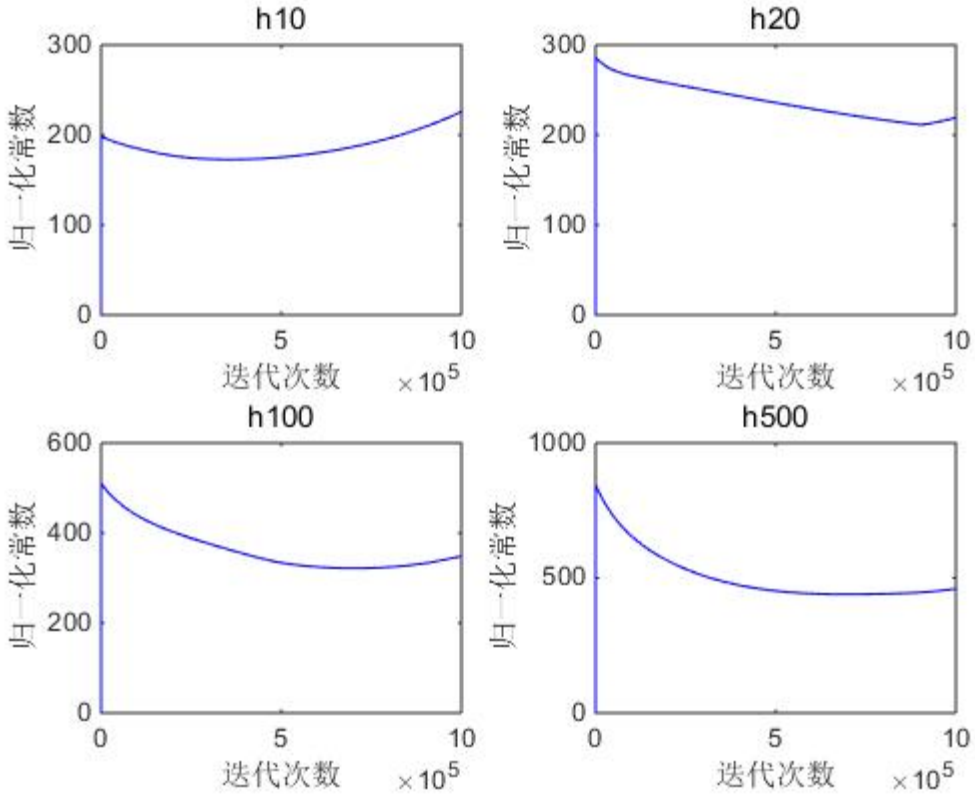

图 4 AIS 收敛曲线

可以看出 AIS 方法最终得到的收敛性并不是很好，h10、h20、h100 模型在迭代 10^6 次的时候依然没有很好的收敛趋势。h500 模型的收敛性较好。

- **算法优化**

上述实验的前提假设是 β 符合 0~1 之间的均匀分布。如果 β 采用前疏后密的采样方法，可以在一定程度上提升效率。上文已经证明迭代次数对每个模型估计时间的影响几乎线性的，因此仅对比 10^6 次迭代下，两种不同的 β 分布方式的差异。

下表为迭代 10^6 次，两种方法的运行时间对比：

	h10	h20	h100	h500
均匀分布/s	131.443	148.123	237.625	800.214

前疏后密/s	130.437	144.932	229.191	763.510
时间减少	0.7%	2.7%	3.4%	4.6%

表格 5

可以看到β采用前疏后密的采样方法在隐藏变量数目较少的时候和均匀分布的时间基本一致，时间略有减少，这种性能的提升会随着 RBM 模型隐藏变量个数的增加而增加。

3.3 Thouless-Anderson-Palmer (TAP)

3.3.1 TAP 算法

TAP 算法的思想来自于物理学中自由能的概念，将问题转化为求 RBM 模型自由能的最小值。引入共轭变量 m，经过 Lengendre 变换，得到如下方程：

$$-\beta\Gamma[m] = -\beta max_q[F[q] + \sum_i q_i m_i] = -\beta(F[q^*] + \sum_i q_i^*[m]m_i) \quad (13)$$

其中，F即为所求归一化常数。经共轭变换得到方程：

$$-\beta F = -\beta F[q = 0] = -\beta min_m\big[\Gamma[m]\big] = -\beta\Gamma[m^*] \quad (14)$$

上述方程的目标为求得极值下的 m 与 q，则须在极值处导函数为 0。对微分方程进行级数展开，整理后可以得到：

$$-\beta\,\Gamma\,(m) = -\sum_i [m_i \ln m_i + (1-m_i)\ln(1-m_i)] + \beta\sum_i a_i m_i + \beta\sum_{(i,j)} W_{ij} m_i m_j$$
$$+\frac{\beta^2}{2}\sum_{(i,j)} W_{ij}^2 (m_i - m_i^2)\left(\frac{1}{2} - m_i\right)\left(m_j - m_j^2\right)\left(\frac{1}{2} - m_j\right)$$
$$+\frac{2\beta^3}{3}\sum_{(i,j)} W_{ij}^3 (m_i - m_i^2)\left(\frac{1}{2} - m_i\right)\left(m_j - m_j^2\right)\left(\frac{1}{2} - m_j\right)$$
$$+\beta^3\sum_{(i,j,k)} W_{ij}W_{jk}W_{ki}(m_i - m_i^2)\left(m_j - m_j^2\right)(m_k - m_k^2) + \cdots \tag{15}$$

对上式保留至二阶项，可以得到$\Gamma\,(m)$的估计：

$$\Gamma\,(m^v, m^h) \approx -S(m^v, m^h) - \sum_i a_i m_i^v - \sum_j b_j m_j^h$$
$$-\sum_{(i,j)}\left[W_{ij} m_i^v m_j^h + \frac{W_{ij}^2}{2}(m_i^v - (m_i^v)^2)\left(m_j^h - \left(m_j^h\right)^2\right)\right] \tag{16}$$

$S(m^v, m^h)$为系统的熵。

对式(14)的求解，可以直接构造函数，对当前的 v 和 h 进行不动点迭代。根据《Training Restricted Boltzmann Machines via the Thouless-Anderson-Palmer Free Energy》（Marylou Gabri´e，Eric W. Tramel，Florent Krzakala）中提及的 sigmoid 函数，可以进行如下迭代：

$$m_j^h[t+1] \leftarrow sigm\left[b_j + \textstyle\sum_i W_{ij} m_i^v[t] - W_{ij}^2\left(m_j^h[t] - \frac{1}{2}\right)(m_i^v[t] - (m_i^v[t])^2)\right], \tag{17}$$

$$m_i^v[t+1] \leftarrow sigm\left[a_i + \textstyle\sum_i W_{ij} m_j^h[t+1] - W_{ij}^2\left(m_i^v[t] - \frac{1}{2}\right)\left(m_j^h[t+1] - \left(m_j^h[t+1]\right)^2\right)\right] \tag{18}$$

最终得到收敛的值。

3.3.2 算法分析

- **收敛性及误差**

TAP 算法利用不动点法进行函数迭代，不断逼近方程的解，速度主要取决于迭代函数的选择，使用 sigmoid 函数可以很快得到收敛的数值解，在十次左右就可以收敛。下图为给定四种不同 RAM 模型下的归一化常数收敛曲线，仅保留到二阶级数项：

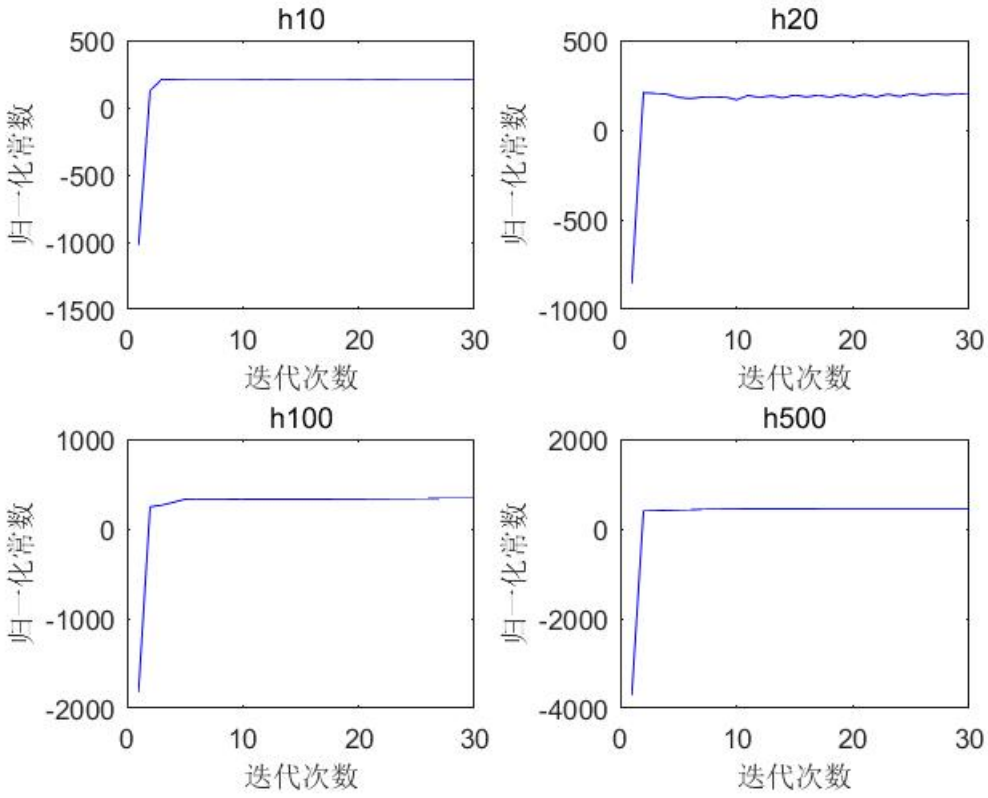

图 5 TAP **收敛曲线**

TAP 算法的误差在四种算法中属于比较大的，因为在计算归一化常数的时候，用到了 Lengendre 变换的方式，对无法得到解析解的微分方程进行展开，从而得到无穷级数项。这必然将引入截断误差，误差的大小很大程度上取决于保留级数的项数。截断误差是不可消除的，是 TAP 算法本身的固有特性。

➢ **算法优化**

TAP 算法的速度没有很大的优化空间，故希望通过增加级数项减小误差，进行优化。

图 5 绘制了仅保留到二阶级数的收敛曲线，下面采用保留到三阶级数的计算方法，即：

$$\Gamma(m^v, m^h) \approx -S(m^v, m^h) - \sum_i a_i m_i^v - \sum_j b_j m_j^h$$

$$- \sum_{(i,j)} \left[W_{ij} m_i^v m_j^h + \frac{W_{ij}^2}{2} (m_i^v - (m_i^v)^2) \left(m_j^h - \left(m_j^h \right)^2 \right) \right]$$

$$- \sum_{(i,j)} \frac{2W_{ij}^3}{3} (m_i^v - (m_i^v)^2) \left(\frac{1}{2} - m_i^v \right) \left(m_j^h - \left(m_j^h \right)^2 \right) \left(\frac{1}{2} - m_j^h \right)$$

对每种 RBM 模型计算十次，比较两种保留方式的估计结果如下表：

	h10	h20	h100	h500
迭代次数	30	30	30	30
均值（二阶）	212.4677	203.6168	340.6946	449.0903
均值（三阶）	212.5077	203.4163	340.7136	449.0738
标准差（二阶）	0.0733	$1.8948*10^{-14}$	0.0063	$1.9506*10^{-6}$
标准差（三阶）	$1.1557*10^{-4}$	$1.4221*10^{-14}$	$1.8595*10^{-4}$	$1.2034*10^{-6}$

表格 6

由于 TAP 算法仅在初始化 v，h 的时候具有随机性，此后每次迭代都代入确定的公式，因此归一化常数的结果标准差很小。

但是，保留三阶级数项的做法对归一化常数的估计的影响并不显著，对比四个 RBM 模型的数据，可以看到保留三阶级数与保留二阶计数得到的结果几乎是一样的。所以，增加级数阶数的方法在本问题中并不能减小截断误差。

3.4 Rao-Blackwellized Tempered Sampling (RTS)

3.4.1 RTS 算法

RTS 算法的实现与 AIS 的方法比较类似，都是通过不断的退火过程，从初始状态不断的逼近目标状态。区别在于β，RTS 引入了β的随机性，从而不必像 AIS 算法一样需要设置大量的β，这样可以减少空间的开销。

β的更新公式为：

$$q(\beta_k|x) = \frac{f_k(x)r_k/\widehat{Z_k}}{\sum_{k'=1}^{K} f_{k'}(x)r_{k'}Z_{k'}/\widehat{Z_{k'}}}. \tag{19}$$

其中$f_k(x)$利用公式(11)给出。

定义r_k为目标平稳分布，用$\widehat{c_k}$不断逼近：

$$\widehat{c_k} = \frac{1}{N}\sum_{i=1}^{N} q\left(\beta_k|x^{(i)}\right) \tag{20}$$

则同时可以保证$\widehat{Z_k}$逼近归一化常数：

$$\widehat{Z_k^{RTS}} = \widehat{Z_k}\frac{r_1}{r_k}\frac{\widehat{c_k}}{\widehat{c_1}}, \quad k = 2, \dots, K \tag{21}$$

$$Z_k = \widehat{Z_k}\frac{r_1}{r_k}\frac{q(\beta_k)}{q(\beta_1)}, \quad k = 2, \dots, K \tag{22}$$

具体算法流程为:

---

1. Input $\{\beta_k, r_k\}_{k=1,\dots,K}, N$
2. Initial $\log\widehat{Z_k}, k = 2, \dots, K$
3. Initial $\beta\epsilon\{\beta_1, \dots, \beta_K\}, 0 = \beta_1 < \cdots < \beta_K = 1.$
4. Initial $\widehat{c_k} = 0, \mathrm{k} = 1, \dots, \mathrm{K}$
5. for i=1 : N
6. Transition in x leaving $q(\mathrm{x}|\beta)$ invariant.

   Sample $\beta|\mathrm{x}\sim(\beta|\mathrm{x})$

Update $\widehat{c_k} \leftarrow \widehat{c_k} + \frac{1}{N} q(\beta_k|x)$

7. end for
8. $\widehat{Z_k^{RTS}} = \widehat{Z_k} \frac{r_1}{r_k} \frac{\widehat{c_k}}{\widehat{c_1}}, \quad k = 2, \dots, K$

3.4.2 算法分析

- **精度要求**

在 RTS 的算法流程中，循环的终止条件是判断$\widehat{c_k}$与给定的r_k是否足够接近，即比较二者的绝对值是否符合阈值要求。这与模型隐藏变量的数量有关，阈值的设置需要反复试验。随着隐藏变量数量的增加，阈值应该不断放宽，否则时间上的开销将会非常巨大，尽管最后得出的结果会更加精确，但是过长的运行时间导致更高的精度没有意义。

下表列出了综合衡量时间及精度后的运行结果及参数精度设置：

	h10	h20	h100	h500
精度要求	0.006	0.001	0.02	0.2
平均迭代次数	1800	101	106	101
时间	269.755	15.824	28.161	100.852
估计值	226.331	207.667	347.963	459.8109

表格 7

- **收敛性**

RTS 的收敛性较差，如下图：

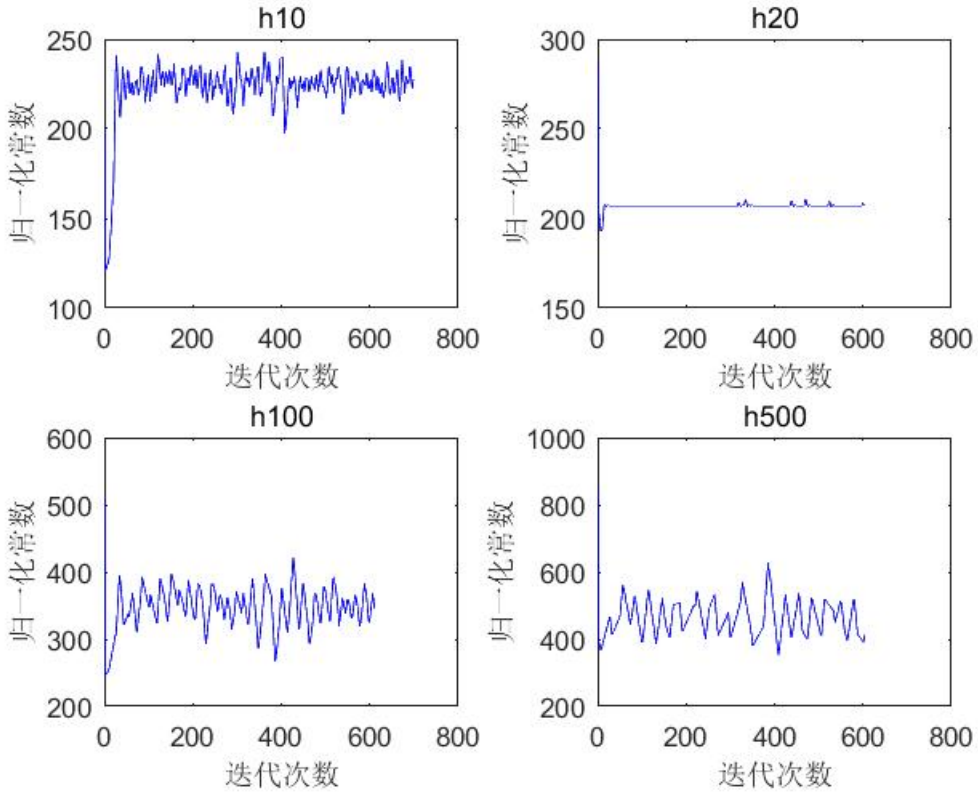

图 6 RTS 收敛曲线

可以看到在 h100、h500 模型中，RTS 的收敛曲线波动时非常巨大的，这一方面是阈值设置的问题，一方面是 RTS 算法本身的问题。因此 RTS 算法在进行 h100、h500 模型估计的时候，得出的结果方差较大，并不可靠。

3.5 Self-adjusted mixture Sampling (SAMS)

3.5.1 SAMS 算法

SAMS 算法本质上与 RTS 的思想相同，引入了辅助分布。首先定义了空间χ上的一个概率分布：

$$dP_j = \frac{q_j(x)}{Z_j} d\mu, \qquad j = 1, \dots, m$$

该分布的物理背景为基于体系中自由能的概率分布，如公式(1)所描述的情形，其中Z_j为本文关注的归一化常数。定义$\zeta_j^* = \log Z_j/Z_1$，$j = 0,1,\dots,m,$ 可知$\log Z_j$表征的即为 RBM 模型的自由能，其中Z_1为参考值，ζ_j^*表征的应当是体系自由能的差值。利用公式(12)可以得到$\log Z_1$

P_j为一系列随机分布，将P_j联合起来得到联合分布$(P_1,\dots,P_m)$，用 L 标记第l个分布，则引入了随机变量 L:

$$(L,X)\sim p(j,x;\zeta) \propto \frac{\pi_j}{e^{\zeta_j}} q_j(x) \tag{23}$$

记$\pi = (\pi_1,\dots,\pi_m)$为联合分布中每个分布$P_j$的权重，$\sum_{j=1}^m \pi_j = 1.$ 在下文的算法实现中，$\pi_1 = \cdots = \pi_m = m^{-1}$.得到 L 的边缘分布为：

$$p(L = j;\zeta) = \frac{\pi_j e^{-\zeta_j+\zeta_j^*}}{\sum_{l=1}^m \pi_l e^{-\zeta_l+\zeta_l^*}},\ j = 1,\dots,m \tag{24}$$

关于具体的 SAMS 算法实现，《Optimally adjusted mixture sampling and locally weighted histogram analysis》（Zhiqiang Tan，2014）一文提到了 global-jump 和 local-jump 两种方法，主要的区别在于转移核不同，即每次迭代过程中L_t的更新方式不同，自由能的计算公式略有不同，但都可以做到自由能估计的自适应。

本文中采用 global-jump 的方法实现 SAMS 算法，转移核为：

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

具体的流程如下：

1. Initial x, $0 = \beta_1 < \cdots < \beta_m = 1$
2. Sample x_1 from $L_t = m.$
3. for t = 1 : N do
4. Generate $L_t\sim p(L = \cdot\,|X_{t-1};\zeta)$
5. $X_t\sim \Psi_{L_t}(X_{t-1},\cdot)$
6. Update $\zeta^{(t)}$
7. end for

8. $\zeta^{(t)}$

$p_{GJ}(l_t|x_{t-1};\zeta)$描述$L$在现有粒子体系及自由能下的分布，即条件概率，对应算法中的第 4 步，计算L的条件概率利用如下公式：

$$p(L=j|x;\zeta)=\frac{\pi_j e^{-\zeta_j}q_j(x)}{\sum_{l=1}^{m}\pi_l e^{-\zeta_l}q_l(x)}\propto\frac{\pi_j}{e^{\zeta_j}}q_j(x) \tag{25}$$

其中$q_j(x)$由公式(11)中的$P_k^*(v)$给出。

利用L的条件分布以及概率积分原理（详见附录 C），可以生成新的L_t的值。

$p(x_t|l_t,x_{t-1})$描述在原有观测变量分布及L的基础上新的变量分布，计算公式由公式(9)给出，β由l_t给出下标，生成新的采样点。

gobal-jump 方法自由能$\zeta^{(t)}$的更新公式为：

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

$$\zeta^{(t-\frac{1}{2})}=\zeta^{(t-1)}+t^{-1}\left\{\frac{w_1(X_t;\zeta^{(t-1)})}{\pi_1},\dots,\frac{w_m(X_t;\zeta^{(t-1)})}{\pi_m}\right\} \tag{26}$$

其中，$w_j(X;\zeta)=\frac{\pi_j e^{-\zeta_j}q_j(x)}{\sum_{l=1}^{m}\pi_l e^{-\zeta_l}q_l(x)}$，$j=1,\dots,m$

3.5.2 算法分析

- **算法的自适应性**

SAMS 算法中自由能$\zeta^{(t)}$的更新应用了随机逼近的方式，去逼近ζ^*，ζ^*是满足方程$p(L=j|x;\zeta)=\pi_j\ \ (j=1,\dots,m)$ 的唯一解，这种逼近方法具有自适应的特性。这种自适应性是指: 如果$\zeta_j^{(t-1)}$小于（或大于）ζ^*，根据公式(25)可以得到，$L_t=j$的概率会高于（或低于）π_j，再由公式(26)指出的更新方式，使得$\zeta_j^{(t)}$比$\zeta_j^{(t-1)}$增加（或减少），即更加逼近ζ^*。

- **系数对收敛速度的影响**

$\zeta_j^{(t)}$的更新方式中，系数t^{-1}会随着更新次数的增大而逐渐变小，使得收敛的速度会越来越慢，从而需要较多的收敛次数和较长的收敛时间。如果用如下的优化公式:

$$\begin{cases} min(\pi_j, t^{-\alpha}) & if \ \ t \le t_0 \\ min\{\pi_j, (t - t_0 + t_0^{\alpha})^{-1}\} & if \ \ t \ge t_0 \end{cases}$$

代替t^{-1}作为系数，将会有效的解决收敛速度变慢的问题，从而大大提高收敛的效率。其中α的取值为(0.5,1)，t_0取为 0.2~0.5 倍的循环次数。

迭代 10^5 次，对比系数为t^{-1}和上述优化公式时的收敛曲线：

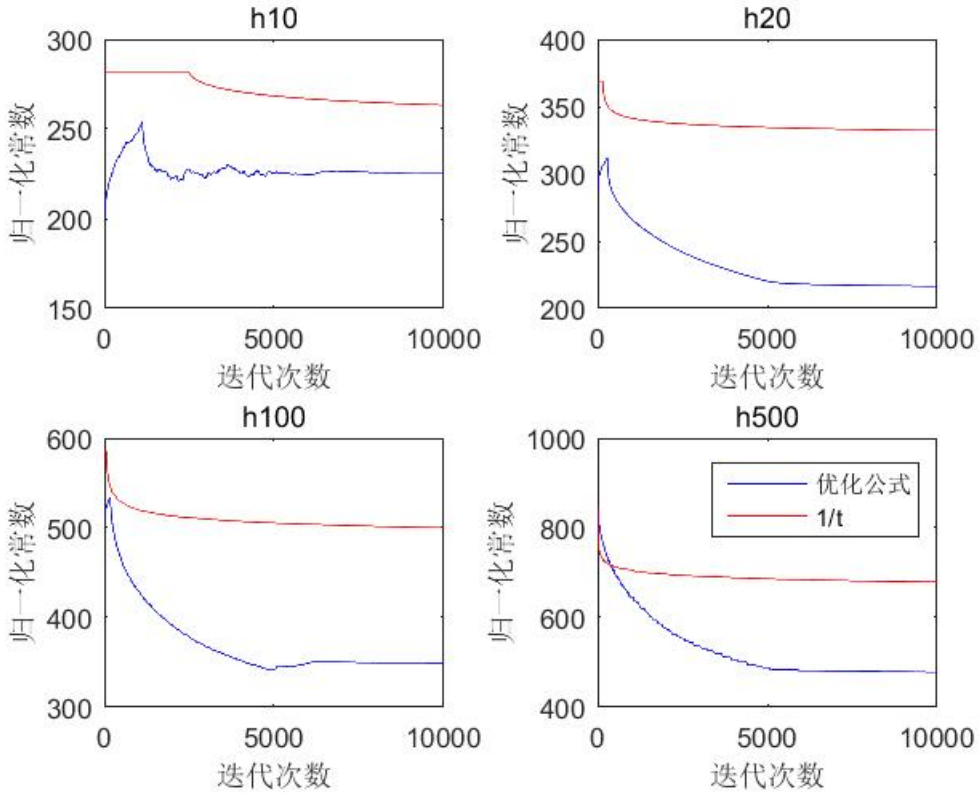

图 7 SAMS 收敛曲线

可以看到与优化公式相比，系数为t^{-1}时的收敛速度时非常缓慢的。10^4 迭代，系数为优化公式的情况下已经可以基本做到完全收敛，除 h500 模型仍有较为明显的下降外，其余三个模型曲线均以达到平稳状态。但是系数为t^{-1}的情形，可以看到所有的模型收敛曲线都有明显的下降趋势，没有完全收敛。

- **均值、标准差及时间**

采用优化公式代替t^{-1}作为系数，对给定的四种不同隐藏变量个数的模型进行归一化估计，每种模型都运行 10 遍，统计估计值的平均值和标准差，运行一次的时间为 10 次运行的平均值，结果如下表：

	h10	h20	h100	h500
迭代次数	10000	10000	10000	100000
均值	226.1507	221.2066	347.7681	460.1415
标准差	0.6274	2.8989	2.1407	2.5456
时间/s	26.119	34.640	55.032	以小时计算

表格 8

其中 h500 模型，迭代 10^4 次时的归一化常数估计值为 478.6562，但是可以从图中看到归一化常数值仍存在缓慢的下降，因此将迭代次数设置为 10^5 次，可以得到估计值为 460.1415，有非常明显的减少，因此可以断定在 10^4 次迭代的时候，归一化常数并没有完全收敛。

4.算法比较

本文第三章分别实现了四种算法并进行了分析，本章主要比较各个算法之间的性能差异。

4.1 准确性

本工作的主要目的是进行 RBM 模型归一化常数的估计，归一化常数估计值的准确程度将对后续对 RBM 模型的研究产生非常显著的影响，因此，估计的准确性是本文衡量算法性能的第一个指标。

对于给定的四个 RBM 模型，用四种算法分别可以获得的最好的估计值如下表：

	h10	h20	h100	h500
AIS	226.1913	221.3501	348.3260	459.8581
TAP	212.4677	203.6168	340.6946	449.0903
RTS	226.331	207.667	347.961	459.8109
SAMS	226.1507	221.2066	347.7681	460.1415

表格 9

从上表中可以非常明显地看出，AIS 算法与 SAMS 算法的估计结果基本吻合，在误差允许范围内可以认为是一致的，因此将 AIS 与 SAMS 的估计结果认为是较为准确的参考值。

TAP 模型的估计值与其他三种相比，相差最多（蓝色标注）。需要注意的是，本文所提供的归一化常数数据，均是取自然对数，则可知 TAP 算法结果与另外三种算法的结果实际上具有量级上的差异。因此可以认为 TAP 算法的估计值不准确。

RTS 算法在对 h10、h100、h500 模型进行估计的时候，结果较为准确。但是

在估计 h20 模型的时候出现了较大的偏差（表中橙色标注），对数下误差为 14，则真实误差在 5 个量级以上。同时，图 6 RTS 收敛曲线显示 RTS 算法的稳定性并不好。

综上，可以得出结论，AIS 和 SAMS 算法的准确性最好。

4.2 速度

在上一节进行算法准确性比较的时候，选取了各种算法对四个模型的最好的估计，没有考虑算法效率的问题，本节比较各个算法的速度。

下表为得到最好估计所需时间：

	h10	h20	h100	h500
AIS	13.340	148.123	23.821	79.309
TAP	0.027	0.029	0.046	0.409
RTS	269.755	15.824	28.161	100.852
SAMS	26.119	34.640	55.032	以小时计算

表格 10

可以看到 TAP 算法的时间远远短于另外三个算法，但是前文分析准确性的时候已经说过，TAP 算法的结果与其他三种相差甚远，因此认为 TAP 不准确，所以不予采用。

纵向对比每个模型在不同算法下的速度，可以看到，在 h10、h100、h500 模型的估计中速度最快，h20 模型速度最快的是 RTS，但是由于算法不稳定，故 SAMS 的结果是最好的。

- **算法选择、归一化常数及似然值**

综合考虑准确性、稳定性、时间因素，得到如下算法选择策略：

	算法	迭代	时间/s	均值	标准差	估值区间
h10	AIS	10^5	13.340	226.1913	0.6921	[225.4992,226.8834]
h20	SAMS	10^5	34.640	221.2066	2.8989	[218.3077,224.1055]
h100	AIS	10^5	23.821	348.3260	0.5750	[347.7510,348.9010]
h500	AIS	10^5	79.309	459.8581	1.1364	[458.7217,460.9945]

表格 11

估计值处于表中所示的估值区间内的概率较大。

对于给定的 test.mat 文件中的输入数据，根据公式(4)对四个模型计算似然值，得到如下结果，算法执行 10 次，得到均值：

	数量级	均值
h10	10^{-31}	1.6415
h20	10^{-19}	1.8278
h100	10^{-13}	1.7919
h500	10^{-12}	1.2819

表格 12

实际上，计算均值并没有太大的意义，在测试的过程中，可以看到似然值的标准差与均值在同一量级或低一个量级。在本工作进行的多次测试中，h10 的归一化常数和似然值估计比较准确，似然值量级稳定在 10^{-31}。h20 模型的归一化常数估计最不稳定，结果有可能在估计值区间之外，因此似然值的估计也并不稳定，量级可能为 10^{-20}~10^{-18}。h100、h500 的归一化常数值基本稳定，基本不会出现在估计值区间外的情况，但是似然值的计算结果也可能出现量级的变化，10^{-14}~10^{-12} 之间波动。

似然值衡量的是一个模型训练的好坏，现在的结果表明，隐藏变量的个数越多，在一定程度上模型越好。但是比较四个模型的似然值，可以看到隐藏变量个

数为 100 和 500 的时候，似然值的差距不大，说明隐藏变量个数并不是越多越好。

也有一种可能的解释是：给定的数据只有 10^5 个，远远小于可见变量的可能的数目，因此这个似然值的估计是没有意义的，不能够作为模型好坏的评判标准。

在 run.m 中，为了保证 h20 结果的稳定性，我将 SAMS 算法执行五遍取平均值计算 h20 的归一化常数。根据表格 11 中对代码运行时间的统计，run.m 的执行时间应当在 5min 左右，虽然运行时间较长，但是估计值应当较为准确。

5.结论

MCMC 在仿真估值方面有非常重要的应用。在深度学习领域有基础性地位的 RBM 模型利用 MCMC 方法，可以对归一化常数进行估计。针对二维高斯的相关系数估计及 RBM 模型归一化常数的估计，本文主要得出以下结论：

- MCMC 方法在对二维高斯相关系数进行估计的时候，对于给定的二维高斯分布，在误差允许范围内准确估计出相关系数。增加抽样次数可以使估计值更加接近真实值，减小若干次估计之间的标准差。建议分布$f(x)$为高斯分布时会比均匀分布时有更好的收敛性，估计更加准确，标准差更小，但是速度要略慢一些。采用 50000 次抽样，$f(x)$为高斯分布进行相关系数估计是较佳的选择；
- 用 AIS、TAP、RTS、SAMS 四种算法对 RBM 模型进行归一化常数估计的时候，依照相关文章的介绍进行 matlab 代码实现，对相关文章中提及的优势进行验证，发现了各个算法的优势及不足之处。用四种不同的算法对给定的四个 RBM 模型进行估计时，发现 AIS、RTS、SAMS 三种算法的结果在误差允许范围之内可以认为是一致的，TAP 算法的计算结果与其有量级上的差别，因此认为 TAP 算法的结果不准确。综合考量 AIS、RTS、SAMS 算法针对不同模型归一化常数估计的准确性、稳定性和时间因素，**最终采用 AIS 方法对 h10、h100、h500 模型进行估计，需要 10^5 次迭代；用 SAMS 算法对 h20 模型进行估计，需要 10^5 次迭代。最终估计 h10、h20、h100、h500 模型的归一化常数估计区间分别为：[225.4992, 226.8834]、[218.3077, 224.1055]、[347.7510, 348.9010]、[458.7217, 460.9945]。得到的似然值分别为：10^{-31} 量级、10^{-19} 量级、10^{-13} 量级、10^{-12} 量级。**
- 在一定范围内，隐藏变量越多，训练的模型效果越好，但是隐藏变量数量达到一定程度之后，增加隐藏变量的个数不能对提升模型性能有很大的影响。

然而，本工作仍有一些遗憾：

- 由于时间的关系，未能实现对给定 RBM 模型的训练。
- 对 AIS 算法的优化不足，没能尝试更多的 β 的分布方式对于算法效率的影响，仅仅比较了均匀分布是前疏后密的分布方式，前疏后密的方式对于效率的提升不足 5%。后续应当尝试更多的分布，比如：碗装的韦伯分布等。
- 没能进一步得出隐藏变量个数对于模型训练的影响，查阅相关文献之后得知这个问题一直没有很好的解决。
- 对于代码细节的优化不够充分，只进行了一处优化，将计算未归一化概率的时候将 for 循环改为了矩阵运算，速度提升了 40%。但是在其余代码中仍有一些 for 循环语句存在，对代码的执行速度有影响。

致谢

衷心感谢欧志坚老师在随机过程这门课上对我的指导，让我收获了非常多的知识。感谢助教学长的认真负责，不厌其烦地在微信上给我解答各种疑惑。感谢余东瀚同学，他对辅助函数的理解大大加深了我对算法的认识。感谢我的室友，连日来，不断有人在室友入睡之后找我讨论，难免影响室友休息，但都表示了理解和包容。

特别感谢我的妈妈，在我调代码心情沮丧的时候，通过电话给我鼓励和支持，并且没有在意我的坏脾气。

参考文献

[1] 林元烈. 应用随机过程[M]. 北京: 清华大学出版社, 2002：114-116.

[2] Ruslan Salakhutdinov. 2009. Learn Deep Generative Models.

[3] Marylou Gabri´e, Eric W. Tramel, Florent Krzakala. Training Restricted Boltzmann Machines via the Thouless-Anderson-Palmer Free Energy.

[4] David E. Carlson, Patrick Stinson, Ari Pakman, Liam Paninski. 2016. Partition Functions from Rao-Blackwellized Tempered Sampling.

[5] Zhiqiang Tan. 2015. Optimally adjusted mixture sampling and locally weighted histogram. In Technical Report, Department of Statistics, Rutgers University.

[6] Hayes. 1996. Statistical Digital Signal Processing and Modeling.

[7] William J. Stewart. 2013. Probability, Markov Chains, Queues, and Simulation.

[8] Metropolis, N; Rosenbluth, AW; Equation of State Calculations by Fast Computing Machines. Amer INST Physics, Circulation Fulfillment DIV.

附录 A

正态分布（*Normal distribution*）又名高斯分布（*Gaussian distribution*），是一个在数学、物理及工程等领域都非常重要的概率分布，在统计学的许多方面有着重大的影响力。

若随机变量X服从一个数学期望为μ、标准方差为σ^2的高斯分布，记为：$X\sim N(\mu\ ,\sigma^2)$,则其概率密度函数为

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

累积分布函数是指随机变量X小于或等于x的概率，一维高斯分布函数用密度函数表示为：

$$F(x;\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{x}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)dx.$$

若随机变量X是二维分布，则其概率密度表示为：

$$f(x,y)=\frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}}e^{\left\{\frac{-1}{2(1-\rho^2)}\left[\frac{(x-\mu_1)2}{\sigma_1^2}-2\rho\frac{(x-\mu_1)(y-\mu_2)}{\sigma_1\sigma_2}+\frac{(y-\mu_2)2}{\sigma_2^2}\right]\right\}}$$

其中$\mu_1\ ,\mu_2\ ,\sigma_1\ ,\sigma_2\ ,\rho$ 都是常数，且$\sigma_1>0\ ,\sigma_2>0\ ,-1<\ \rho<1$ 我们称$(x\ ,y)$服从参数为 $\mu_1\ ,\mu_2\ ,\sigma_1\ ,\sigma_2\ ,\rho$ 的二维正态分布，记为$(x\ ,y)\sim N(\mu_1\ ,\mu_2\ ,{\sigma_1}^2\ ,{\sigma_2}^2\ ,\rho).$

用期望值向量μ和协方差矩阵Σ表示，其中

$$\mu=\begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix}\ ,\Sigma=\begin{pmatrix}\sigma(x\ ,x) & \sigma(x\ ,y)\\ \sigma(y\ ,x) & \sigma(y\ ,y)\end{pmatrix}=\begin{pmatrix}{\sigma_1}^2 & \rho\sigma_1\sigma_2\\ \rho\sigma_2\sigma_1 & {\sigma_2}^2\end{pmatrix}$$

记为$X=\begin{pmatrix}x_1\\ x_2\end{pmatrix}\sim N\left\{\begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix}\ ,\ \begin{pmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{pmatrix}\right\}$,故可立得变量间相关系

$$\rho=\sqrt{\frac{\Sigma_{12}\times\Sigma_{21}}{\Sigma_{11}\times\Sigma_{22}}}$$

附录 B

对于一条马氏链，其可能达到一个平稳分布p^*，需满足$p^* = p^*F$；也就是说平稳分布的条件是马氏链是不可约非周期的。当马氏链是周期的它会在状态之间以一个确定的方式循环。是平稳分布的一个充分条件是下面的细致平衡条件成立：

$$F(j,k)p_j{}^* = \sum_{i,j} F(k,j)p_k{}^* \qquad (*)$$

(*)式被称为细致平衡条件。要证明通过 *MH* 方法构造出来的链满足马氏性且以$p(x)$为平稳分布，只需要证明 *Metroplis-Hastings* 方法的转移核满足细致平衡条件即可。首先，*MH* 方法构造出来的链满足马氏性是显然的，因为$X^{(t)}$的产生仅依赖于$X^{(t-1)}$。其次，对于满足细致平衡方程的证明，不失一般性，以二元函数为例说明，分三种情况：

1、$p(x)f(x\ ,y) = p(y)f(y\ ,x)$，则由(2)式可得$\alpha(x\ ,y) = \alpha(y\ ,x) = 1$；

又∵ $p(x\ ,y)$定义为$p(x \to y) = \alpha(x\ ,y)f(x\ ,y)$，加上假设条件$f(x\ ,y)\ p(x) = f(y\ ,x)\ p(y)$，立得$f(x\ ,y)p(x) = f(y\ ,x)p(y)$

(*)式细致平衡条件满足；

2、$p(x)f(x\ ,y) > p(y)f(y\ ,x)$，则由(2)式可得$\alpha(x\ ,y) = \frac{p(y)f(y,x)}{p(x)f(x,y)}$，$\alpha(y\ ,x) = 1$

∴ $f(x\ ,y)\ p(x) = f(x\ ,y)\alpha(x\ ,y)\ p(x) = f(x\ ,y)\frac{p(y)f(y,x)}{p(x)f(x,y)}\ p(x) = f(y\ ,x)\ p(y)$, (*)式细致平衡条件满足；

3、$p(x)f(x\ ,y) < p(y)f(y\ ,x)$, 则由(2)式可得$\alpha(x\ ,y) = 1$ ，$\alpha(y\ ,x) = \frac{p(x)f(x,y)}{p(y)f(y,x)}$

∴ $f(y\ ,x)p(y) = f(y\ ,x)\alpha(y\ ,x)\ p(y) = f(y\ ,x)\frac{p(x)f(x,y)}{p(y)f(y,x)}\ p(y) = f(x\ ,y)\ p(x)$,

(*)式细致平衡条件满足，证毕。

附录 C matlab 文件清单

提交文件	run.m
AIS 算法	AIS.m
TAP 算法	TAP.m
RTS 算法	RTS.m
SAMS 算法	SAMS.m
计算似然值的函数	PROB.m
Gibbs 采样	gibbs_sampling.m
未归一化的概率计算函数	log_Pk.m
sigmoid 函数	sigm.m
TAP 算法归一化常数迭代公式保留二阶级数	tao_TAP.m
TAP 算法归一化常数迭代公式保留三阶级数	tao_TAP_3.m
计算归一化常数的均值和标准差的函数	run_mean_std.m
计算似然值的均值和标准差	PROB_mean_std.m
画归一化常数的收敛曲线	run_plot.m
二维高斯相关系数计算	Gauss.m
二维高斯相关系数估计的均值和标准差的计算	Gauss_mean_std.m