马氏链蒙特卡洛方法的应用

无 47　　刘前*　　2014011216

2016 年 11 月 14 日

摘要

本文第一部分，以马氏链蒙特卡洛 (Markov Chain Monte Carlo, MCMC) 方法中的 Metropolis-Hastings(MH) 算法为基础，深入全面地讨论了 MH 算法中提议函数 (proposal function) 的参数选择、随机样本的选取方法等问题，逐步优化估计结果，使其尽量与真实参数相近，得到了一些令人满意的结果。另外，本文还介绍了一种针对于提议函数参数选择的自适应算法，讨论了算法的可行性。同时，使用 MH 算法的特例 Gibbs 采样法，得到质量更高的随机样本。该部分通过生成给定二维高斯分布的随机样本，对算法进行了测试，结果表明，优化后的 MH 算法生成的随机样本对相关系数估计的精确度和稳定性比优化之前明显更高。

本文第二部分实现了 AIS，TAP(包括二阶近似和三阶近似) 和 RTS 三种采样算法，对四种受限玻尔兹曼机 (RBM) 模型的归一化常数进行了估计，并尝试讨论了算法参数对估计结果的影响，将三种算法的性能加以对比。同时还尝试使用对比散度 (CD) 算法重新训练了 RBM 模型，在训练模型上进一步估计归一化常数。

*清华大学电子工程系 (E-mail: liuqian14@mails.tsinghua.edu.cn)

目录

1 背景介绍

马尔科夫链蒙特卡洛 (Markov Chain Monte Carlo, MCMC) 方法，从 1950 年萌芽，在实践中不断发展，被广泛应用于各学科领域 (如信息科学、物理、化学、生物学、金融、材料等) 的科学计算 [1]，并且随着机器学习和深度学习的流行，日益展示出其强大的影响力。

MCMC 方法常用于模拟复杂的非标准的多元分布，生成服从复杂分布的随机样本，其中最重要的一种方法是 Metropolis-Hastings(MH) 算法，最早由 Metropolis，Rosenbluth 等人提出并发展 [2]。MH 算法用处十分广泛，著名的 Gibbs 采样算法也只不过是 MH 算法的一个特例 [3]。尽管 MH 算法已经被提出四十余年，但是仍然有很多学者在研究这一算法。比如，Bishop 指出了 MH 算法在模式识别和机器学习领域的广泛应用 [4]，同时 Jun S. Liu 也指出了 MH 算法在科学计算的作用 [5]，从中可以看出 MH 算法在最前沿的课题中仍然意义重大。

本文从 MCMC 算法出发，主要分为两部分内容：PART A 将使用 MH 算法对给定的二维高斯分布进行随机采样，并根据生成的样本估计相关系数。为了提高相关系数估计的准确度，文中尝试对 MH 算法中各种参数的选择进行了系统而深入地讨论。PART B 将使用三种不同的采样算法对四种 RBM 模型 (隐变量分别为 10，20，100，500) 进行归一化常数的估计，并根据估计值计算出不同模型在测试数据上的似然值，也尝试对算法进行了进一步的分析讨论。

2 Part A: Metropolis-Hastings 算法应用

2.1 背景介绍

Metropolis-Hastings 算法是一种重要的随机样本生成方法，然而一般的 MH 算法没有对提议函数及其参数的选取给出明确的方法，同时对随机样本间的独立性也没有强制要求。[6] 对提议函数及其方差的选取对采样结果的影响作了基础的讨论，并由此推出一种改进的自适应算法用以寻找正态型提议函数的方差，一定程度了提高了算法的可控性，但是无法保证随机样本间的独立性，因而本文还尝试通过检验生成的随机样本的自相关系数，给出了

选择独立性较高的随机样本的方法，从而实现随机样本更接近真实分布。

2.2 Metropolis-Hastings 算法

设 $\pi = (\pi(i) > 0, i \in S)$ 为任意给定的概率分布,$\mathbf{T} = (T(i,j), i,j \in S)$ 为选择的易于实现的条件概率转移矩阵 $(T(i,j) > 0, i,j \in S)$(称 $\mathbf{T}$ 为参照矩阵)。$X = \{X_n, n \geq 0\}$ 由下列步骤生成 [7]:

(1) 给定 $X_n (X_n \in S, n \geq 0)$，根据 $\mathbf{T}(X_n, \cdot)$ 抽取 Y;

(2) 计算得到接受概率 (Accept Probability), 计算公式为:

$$p(X_n, Y) = min\{1, \frac{\pi(Y)\mathbf{T}(Y, X_n)}{\pi(X_n)\mathbf{T}(X_n, Y)}\}$$

(3) 抽取 $U \sim U[0,1]$, 如果 $U < p(X_n, Y)$, 则令 $X_{n+1} = Y$, 否则 $X_{n+1} = X_n$。返回步骤 (1)。

由上述步骤生成的 $X = \{X_n, n \geq 0\}$ 是不可约马氏链，其平稳分布即为给定概率分布 π。

Metropolis-Hastings 算法可以总结为 Algorithm1。

注意: MH 算法中的条件转移矩阵常常是一个辅助的概率密度函数 (提议函数)，记作 $q(x,y)$。同时，为了保证随机样本确实符合平稳分布 (目标分布)$\pi(x)$，常常选取 n 足够大之后的随机样本。

2.3 Metropolis-Hastings 算法的深入讨论

2.3.1 问题描述

给定二维高斯分布，

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

图 1: 目标分布 (二维高斯分布)

Algorithm 1 Metropolis-Hastings 算法

输入: 概率分布密度 $\pi(x)$; 条件概率转移矩阵 $\mathbf{T} = (T(i,j), i,j \in S)$; 生成的随机样本数 N;

初始化: X_1

输出: $X = \{X_n, n \geq 0\}, n = 1, 2, \cdots, N$

for $n = 1 : N - 1$

1. 由 $\mathbf{T}(X_n, \dot{)}$ 抽取 Y;

2. 计算接受概率 (Accept Probability):

$$p(X_n, Y) = min\{1, \frac{\pi(Y)\mathbf{T}(Y, X_n)}{\pi(X_n)\mathbf{T}(X_n, Y)}\}$$

3. $U \sim U[0,1]$, 如果 $U < p(X_n, Y)$, 则 $X_{n+1} = Y$;

   否则 $X_{n+1} = X_n$。

end

要求使用 Metropolis-Hastings 算法对该分布进行随机采样，并使用生成的随机样本估计该二维高斯分布的相关系数。事实上，很容易得出该二维高斯分布的相关系数为 0.5。

根据对 MH 算法的描述，可以看出算法中不确定的因素有很多，因而在实现算法之前，需要解决以下几个问题，包括:

(1) 如何选取迭代初值;

(2) 如何选取提议函数 $q(x,y)$(包括提议函数自身的参数);

(3) 需要生成多少随机样本;

(4) 随机样本在 n 至少为多大才能保证已经收敛到平稳分布;

(5) 提高随机样本独立性的具体方法; 等。

这些因素都会对 MH 算法的性能造成一定的影响。下文将逐一解决这些问题。

2.3.2 迭代初值的选取

MH 算法保证了只要迭代初值在分布函数定义的区间内，最后将会收敛到目标分布，因而，在选取迭代初值时可以不必太纠结。一般选取生成随机数的方法，可以帮助解决初值选取的困难。

对于**问题描述**中的二维高斯分布，X_1 和 X_2 可以使用均匀分布或者一维高斯分布生成随机样本。使用均匀分布时，可以先限定一个大致范围，范围的选取可以根据目标分布的均值来选择；使用一维高斯分布时，均值可以选择目标分布的均值，方差任选。

总之，MH 算法中迭代初值的选取比较随意，对算法的最终结果不会造成很大影响，但对算法的运行效率影响较大。

2.3.3 提议函数的选取

MH 算法中对提议函数 $q(x,y)$ 的选取没有给出特别的要求，因而从理论上说，提议函数 $q(x,y)$ 的选取可以是任意的。但是，实际中需要考虑到算法实现的难易程度以及算法的运行效率，因而选取提议函数 $q(x,y)$ 需要多加一些考虑。

一般认为，提议函数至少需要满足以下两个要求 [6]：

(1) 对于固定的 x, 能够便捷地从 $q(x,y)$ 中产生随机数。

(2) 提议函数形式与目标分布越接近，生成的随机样本效果越好。

对于**问题描述**中的二维高斯分布，满足上面要求的最佳函数 $q(x,y)$ 即为一维高斯分布，既满足方便生成随机数的要求, 而且与目标分布很接近。同时，一维高斯分布还有一个突出的优点是具有对称性，也就意味着在计算接受概率时，计算公式可以简化为：

$$p(X_n,Y) = min\{1, \frac{\pi(Y)\mathbf{T}(Y,X_n)}{\pi(X_n)\mathbf{T}(X_n,Y)}\} = min\{1, \frac{\pi(Y)}{\pi(X_n)}\}$$

综合以上特点，$q(x,y)$ 选取以 x 为均值的一维高斯分布。

但是问题随之而来，一维高斯分布的均值与方差如何确定?

[6] 中使用随机游动的方法，均值选择为当前的状态 X_n，对于方差则提出了一种寻找提议函数方差的自适应算法 Algorithm2，让接受概率落入一个可以接受的范围，以此为判断条件找到合适的方差值。

Algorithm 2 高斯形式提议函数的方差的自适应 Metropolis-Hastings 算法

输入: 高斯分布的提议函数 $N(X_i, \sigma_n)$; 接受概率的置信区间 $[a-\epsilon, a+\epsilon]$; 引入另一个方差 $\hat{\sigma}$; 操作次数 N; 每次生成的 Markov 链长度 M

初始化: $\sigma_0 = 0, n = 0$

输出: $\sigma_n, n = 0, 1, \cdots, N-1$

for $n = 0 : N-1$

1. 根据提议函数 $N(X_i, \sigma_n)$，采用随机游动采样法生成一条长度为 M 的 Markov 链;
2. 计算第 1 步中产生的 Markov 链接受新状态的比率 P_n;
3. 若 $P_n \in [a-\epsilon, a+\epsilon]$, 则令 $\sigma = \sigma_n$, 退出算法；否则继续执行下一步；
4. 若 $n > 2$ 且 $|P_n - a| < |P_{n-1} - a|$, 则置 $\sigma_n = \sigma_{n-1}$;
5. 从 $N(\sigma_n, \hat{\sigma})$ 产生一个随机数，记作 σ_{n+1};

end

从 Algorithm2描述中可以看出，为了获取一个较为合适的方差值进行了复杂的操作，引入了额外的计算负担，同时在算法中又引入了新的方差参数 $\hat{\sigma}$, 而 $\hat{\sigma}$ 的选取又带来了新的问题。因而这一算法的提出可能更多停留于理论层面，在实际应用中还要考虑到该算法带来的计算效率降低的问题。

至此，提议函数 $q(x, y)$ 的选取问题基本得到了解决。

2.3.4 生成随机样本的数目

问题描述中需要根据生成的随机样本估计目标分布中的相关系数，为了提高对相关系数估计的准确性，必须得到足够数量的随机样本。通过后面的仿真部分，可以基本得出随机样本数目越大，对目标分布参数的估计越精确的结论，效果越好。但在实际应用中，随机样本数不是越大越好，因为随机样本数目的增加会造成计算量和计算时间的增加，因而应该根据实际需要在估计精度和计算效率之间加以平衡。

2.3.5 判断随机样本收敛到目标分布

MH 算法生成的随机样本 $X_n(X_n \in S, n \geq 0)$ 中只有当 n 足够大时，才能保证此时生成的随机样本收敛到目标分布 (或者说与目标分布十分接近)。为了使得估计的相关系数与真实值更加接近，通常情况下，可以人为选择靠后生成的随机样本，虽然不能保证已经收敛到目标分布，但是已经能够满足实际应用的要求。事实上，如果判断随机样本与目标分布是否收敛，往往需要增加一些附加的运算和判断条件，从而降低算法的效率，这是得不偿失的。因而，本文采取普通的方法：选取靠后生成的随机样本，不对这一问题做特殊的处理。

2.3.6 提高随机样本独立性的具体方法

Bishop 在 [4] 中指出，MH 算法生成的连续的随机样本是高度相关的，这与我们生成独立同分布的随机样本的目的相悖。为了更好地保证得到独立的随机样本，常常需要间隔选取随机样本，当间隔足够大时，所得的随机样本之间的独立性就会非常好。

随机样本间的独立性可以用自相关系数来衡量 [6]，自相关系数绝对值越小，说明随机样本序列之间的相关性越弱，独立性越强。在后面的仿真中，针对生成二维高斯分布随机样本的问题，作出自相关系数随采样间隔变化的关系，可以看出间隔在一定范围内增加时，样本自相关系数绝对值会明显降低，表明样本间独立性明显提高；但是当间隔继续增加时，可能会呈现出上下摆动变化的情况，因而随机样本间的独立性并不严格随着间隔的增加而减弱。

总之，根据随机样本的自相关系数来选择采样间隔，能够有效提高随机样本的独立性。

2.4 Metropolis-Hastings 算法仿真 (Simulation) 与优化

下面将通过仿真 (Simulation) 得到数值结果，对上面关于 MH 算法的详细分析作直观地展示。本文中所有仿真结果均在 Windows 8.1 操作系统、使用 MATLAB R2014a 进行仿真得到。

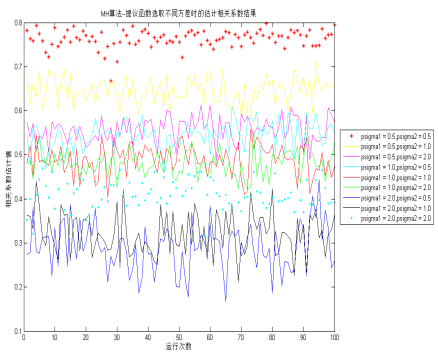

图 2: 提议函数选取不同方差时对相关系数的估计

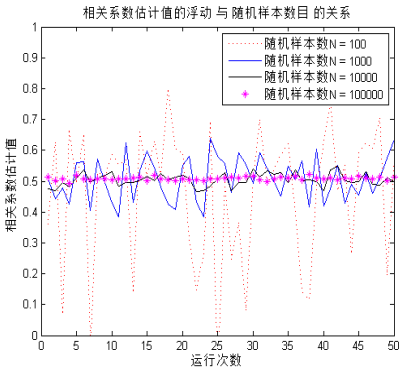

图 3: 随机样本数目不同时对相关系数的估计

2.4.1 提议函数方差的选择

仿真的基本参数设置

a) 初始值由均匀分布随机生成，X_1 的初值由 [0,10] 之间的均匀分布生成，X_2 的初值由 [0,20] 之间的均匀分布生成;

b) 随机样本数量均设定为 10000，选取后 5000 个 (不间隔选取) 用于估计目标分布的相关系数;

c) 独立运行次数为 100 次，用于观察估计值的精确度和稳定性。

在以上设定下，提议函数采用一维高斯的随机游动方法，X_1 和 X_2 的方差 (分别记作 psigma1[1]和 psigma2) 均为三组 0.5、1、2，相互组合共有九种情况。图2比较了不同方差下的运行结果。

根据图2，不同的 (psigma1,psigma2) 组合估计得到的相关系数结果差异很大，只有 (psigma1,psigma2) = (1.0,1.0) 的估计值相对集中在精确值 0.50 附近，精确度相对高一些，但是数值上仍然有很大的波动，因而估计的结果稳定性很差，仍需要进一步的优化。

2.4.2 随机样本数目的选择

仿真的基本参数设置

[1]表示 proposal function 的方差

a) 初始值由均匀分布随机生成，X_1 的初值由 [0,10] 之间的均匀分布生成，X_2 的初值由 [0,20] 之间的均匀分布生成;

b) 提议函数采用一维高斯形式，X_1 和 X_2 的方差均为 1;

c) 独立运行次数为 50 次，用于观察估计值的精确度和稳定性。

随机样本数量分为 100,1000,10000 和 100000 四组，不间隔选取随机样本，图3比较了随机样本数量不同的情况运行结果。

图3直观地展示出，随机样本数不同时，估计得到的相关系数结果差异很大。随着随机样本数目的增加，估计得到的相关系数明显越来越精确和稳定，当随机样本数达到 10000 时，可以看出相关系数估计值基本稳定于精确值 0.50 附近。

2.4.3 对随机样本间隔的选择

仿真的基本参数设置

a) 初始值由均匀分布随机生成，X_1 的初值由 [0,10] 之间的均匀分布生成，X_2 的初值由 [0,20] 之间的均匀分布生成;

b) 提议函数采用一维高斯形式，X_1 和 X_2 的方差均为 1;

c) 独立运行次数为 50 次，用于观察估计值的精确度和稳定性;

d) 用于估计相关系数的随机样本数均为 10000(生成的随机样本总数不同)。

图4直观地展示出，选取不同间隔的随机样本时，估计得到的相关系数结果差异也比较大。随着间隔的增加，估计得到的相关系数明显更精确，稳定性也更高。

事实上，从随机样本的自相关系数角度，也能得到类似的结论。使用 MATLAB R2014a 的 autocorr 函数对生成的随机样本进行分析。图5体现了随机样本的自相关系数与 lags(滞后系数) 的关系，本质上体现了间隔不同时选取的随机样本的独立性。可以看出，lags 越大，自相关系数越小，因而间隔越大，随机样本间的独立性越高。

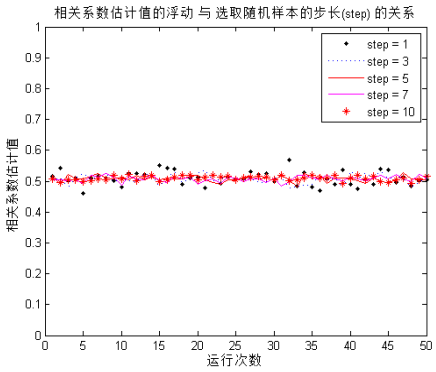

图 4: 随机样本间隔不同时对相关系数的估计

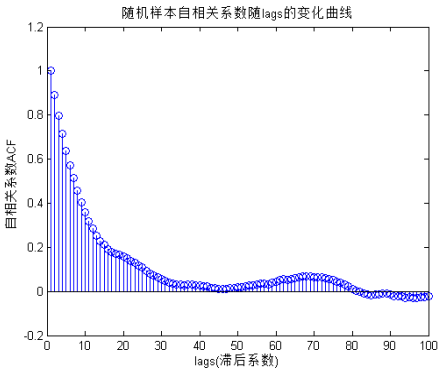

图 5: 随机样本自相关系数随间隔的变化

2.4.4 参数优化后的 MH 算法

经过上述对 MH 算法参数的深入讨论，可以得到一个针对二维高斯分布进行了参数优化的 MH 算法，优化后的参数设置如下：

a) 初始值由均匀分布随机生成，X_1 的初值由 [0,10] 之间的均匀分布生成，X_2 的初值由 [0,20] 之间的均匀分布生成;

b) 提议函数采用一维高斯形式，X_1 和 X_2 的方差均为 1;

c) 选取随机样本的间隔为 10;

d) 用于估计相关系数的随机样本数均为 10000，意味着共生成 100000 个随机样本。

之后**数值结果与算法比较**中所指的“参数优化后的 MH 算法”，采用上述参数进行仿真。

2.5 Gibbs 采样算法

2.5.1 背景介绍

在之前实现 Metropolis-Hastings 算法的过程中，不难发现，MH 算法的一个突出缺点是对提议函数的选择和调整比较复杂，有时还会掺杂着一些主观猜测的成分，给实际应用带来了一定的困难，而这些问题可以通过 Gibbs 采样得以解决。通过后面分析可以看出，在使用 Gibbs 采样方法时，不需要

Algorithm 3 Gibbs Sampling

输入: 条件分布 $f(\theta_1|\theta_2 = \theta_2^{(t)})$ 以及 $f(\theta_2|\theta_1 = \theta_1^{(t)})$；迭代次数 T.
初始化: $\theta_1^{(1)}$ 和 $\theta_2^{(1)}$
输出: $(\theta_1^{(t)}, \theta_2^{(t)}), t = 1, 2, \cdots, T$
for $t = 2 : T$

1. 根据条件分布 $f(\theta_1|\theta_2 = \theta_2^{(t-1)})$ 生成 $\theta_1^{(t)}$;
2. 根据条件分布 $f(\theta_2|\theta_1 = \theta_1^{(t)})$ 生成 $\theta_2^{(t)}$;

end

认为反复调整提议函数的形式；同时，Gibbs 采样接受了所有生成的随机样本，因而生成随机样本的效率得到明显提高 [5]。

2.5.2 算法描述

为了方便地解释 Gibbs 采样算法，以二维分布为例，现要生成服从联合分布 $f(\theta_1, \theta_2)$ 的二维随机样本。

使用 Gibbs 采样算法的前提是已知二维分布的联合分布。根据二维联合分布 $f(\theta_1, \theta_2)$，可以得到 θ_1 和 θ_2 的条件分布 $f(\theta_1|\theta_2 = \theta_2^{(t)})$ 以及 $f(\theta_2|\theta_1 = \theta_1^{(t)})$。得到以上条件分布之后，即可采用 Gibbs 采样算法对二维分布进行采样。

首先，对样本进行初始化，得到 $\theta_1^{(1)}$ 和 $\theta_2^{(1)}$，并且设定迭代次数 T。然后开始迭代生成随机样本，每次迭代的过程与 MH 算法类似。在第 t 次迭代时，根据 $f(\theta_1|\theta_2 = \theta_2^{(t-1)})$ 生成随机样本 $\theta_1^{(t)}$，只不过不同于 MH 算法需要计算接受概率，Gibbs 算法生成的随机样本都会被接受。生成 $\theta_1^{(t)}$ 后，再根据 $f(\theta_2|\theta_1 = \theta_1^{(t)}$ 生成 $\theta_2^{(t)}$。重复以上操作直至完成 T 次迭代。

Gibbs 采样算法可以总结为 Algorithm3.

2.5.3 算法实现

下面将使用 Gibbs 采样算法实现生成服从二维高斯分布的随机样本。

Algorithm 4 Gibbs Sampling for Gaussian Distribution

初始化: $\theta_1^{(1)}$ 和 $\theta_2^{(1)}$

输出: $(\theta_1^{(t)}, \theta_2^{(t)}), t = 1, 2, \cdots, T$

for $t = 2 : T$

1. 根据 $Norm(\mu_1 + \rho(\theta_2^{(t-1)} - \mu_2, \sqrt{1-\rho^2})$ 生成 $\theta_1^{(t)}$
2. 根据 $Norm(\mu_2 + \rho(\theta_1^{(t)} - \mu_1, \sqrt{1-\rho^2}$ 生成 $\theta_2^{(t)}$

end

已知二维高斯分布的联合分布函数 $\boldsymbol{\theta} \sim Norm(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ 其中，$\boldsymbol{\mu}$ 为均值向量，$\boldsymbol{\Sigma}$ 为协方差矩阵

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$$

从二维高斯分布的联合分布函数，可以得到可以得到 θ_1 和 θ_2 的条件分布。

$$\theta_1 \sim Norm(\mu_1 + \rho(\theta_2 - \mu_2), \sqrt{1-\rho^2})$$

$$\theta_2 \sim Norm(\mu_2 + \rho(\theta_1 - \mu_1), \sqrt{1-\rho^2})$$

所以 Gibbs 生成二维高斯分布的算法见 Algorithm4，算法具体的数值结果见**数值结果与算法比较**部分。

图6展示了 Gibbs 采样算法生成的随机样本的分布情况，左侧是给定二维高斯分布的真实概率分布，右侧是生成的随机样本的分布，比较左右两图，可以看出 Gibbs 采样算法生成的随机样本与真实分布十分相近。

2.6 数值结果与算法比较

下面对 Gibbs 采样算法、参数优化之后的 Metropolis-Hastings 算法、最初实现的 MH 算法共三种算法在图7和表1中加以对比。将各种算法独立运行 20 次，各算法对相关系数估计值的精确度、稳定性和计算效率分别用相关系数均值、相关系数方差和运行时间来表示。

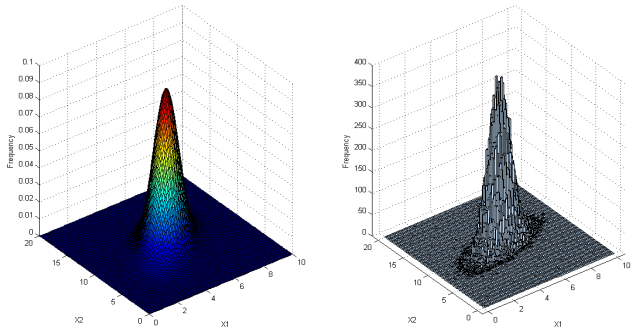

图 6: Gibbs 采样算法生成随机样本

算法	相关系数均值	相关系数方差	运行时间
MH 算法 (最初实现)	0.5085	4.4757e-04	8.152s
MH 算法 (参数优化后)	0.5038	8.3389e-06	238.573s
Gibbs 采样算法	0.4998	5.7696e-06	42.153s

表 1: 不同算法结果对比

从图7可以直观地看出参数优化后的 MH 算法和 Gibbs 采样算法的精确性和稳定性相当，且都优于未优化的 MH 算法。

表1则用具体的数据表明，Gibbs 算法的准确度、稳定性优于其他两种算法。参数未优化的 MH 算法估计值的精确度相对较低，稳定性也较差，但计算效率很高; 而参数优化后的 MH 算法估计的准确性和稳定性相对于优化之前有了明显的提高，但是计算时间非常长，效率很低。

2.7 小结

Part A 对 Metropolis-Hastings 算法进行了基本实现，并且针对 MH 算法中各种参数的选取进行了系统而深入地讨论，大大优化了 MH 算法的计算精确度，但是在计算效率上有所降低。通过生成给定二维高斯的分布随机样本估计相关系数，对优化后的算法和 Gibbs 算法进行了测试，对估计结果的准确性、稳定性和计算效率进行了对比。结果表明，Gibbs 算法的性能最

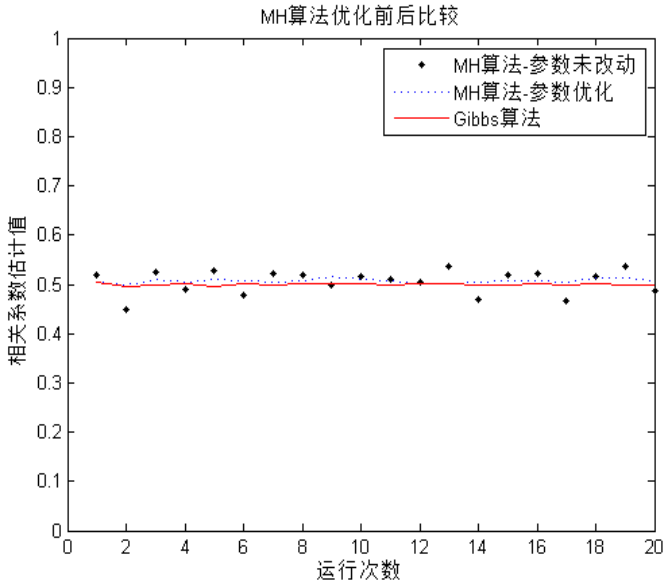

图 7: 不同算法对相关系数的估计值比较

优，参数优化后的 MH 算法对相关系数估计的精确度和稳定性比优化之前的 MH 算法明显提高，但计算效率有所降低。

3 Part B: RBM 模型归一化常数的估计

本部分将研究 RBM 模型归一化常数的估计问题，这一问题随着深度学习的发展受到越来越多的关注 [8]。如果能够估计得到更加精确的归一化常数，将有助于深度学习等诸多领域的发展。这一部分首先对受限玻尔兹曼机 (RBM) 进行基本介绍，然后给出三种估计归一化常数的采样算法，包括：AIS(Annealed Importance Sampling) 算法，TAP(Thouless-Anderson-Palmer) 算法和 RTS(Rao-Blackwellized Tempered Sampling) 算法。

本文实现并对比了以上算法，并尝试对算法中一些细节进行了补充，最终得到 RBM 模型的归一化常数，然后根据所得的结果，在测试数据上得到似然值。

3.1 基本介绍

3.1.1 受限玻尔兹曼机模型

受限玻尔兹曼机 (Restricted Boltzmann Machine,RBM) 由 Hinton 和 Sejnowski 于 1986 年提出 [9]，该模型由一些可见单元 (visible unit，对应可见变量) 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 $\{0, 1\}$。RBM 是一个二部图，只有可见单元和隐藏单元之间存在边，可见单元之间以及隐藏单元之间都没有边连接。

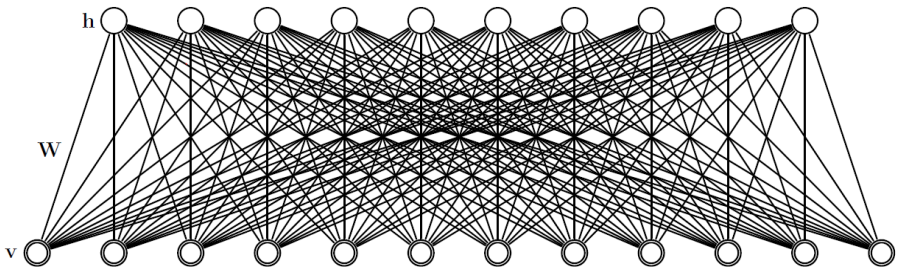

图 8: 受限玻尔兹曼机示意图

3.1.2 归一化常数及似然值

RBM 在 $\mathbf{v}, \mathbf{h}$ 状态下的能量为

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

其中，$\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{a}\}$ 是模型的参数，$\mathbf{W}$ 是连接可见单元与隐藏单元的权值矩阵，$\mathbf{a}$ 和 $\mathbf{b}$ 分别是隐藏单元和可见单元的偏置 (bias) 向量。[8]

可见单元与隐藏单元的联合分布为

$$P(\mathbf{v}, \mathbf{h}; \theta) = \frac{1}{Z(\theta)} exp(-E(\mathbf{v}, \mathbf{h}; \theta))$$

$$Z(\theta) = \sum_{\mathbf{v}} \sum_{\mathbf{h}} exp(-E(\mathbf{v}, \mathbf{h}; \theta))$$

$$P(v; \theta) = \frac{1}{Z(\theta)} \sum_{\mathbf{h}} exp(-E(\mathbf{v}, \mathbf{h}; \theta))$$

其中，$Z(\theta)$ 就是 RBM 模型的归一化常数，也称为 Partition Function。本部分正是通过不同的算法实现对这一常数的估计。而 $P(v; \theta)$ 表示 RBM 模型在测试数据 $\mathbf{v}$ 上的似然值。具体计算公式为:

$$P(v; \theta) = \frac{1}{Z(\theta)} exp(\mathbf{b}^{\mathrm{T}} \mathbf{v}) \prod_{j=1}^{F} (1 + exp(a_j + \sum_{i=1}^{D} W_{ij} v_i)).$$

后文提到的在测试数据上的总似然值是指：若测试数据共有 M 个 $\mathbf{v}$，则总似然值为

$$P = \sum_{n=1}^{M} P(\mathbf{v}_n; \theta)$$

事实上，更一般的似然函数常对 $P(v; \theta)$ 取自然对数 [8]，后面的对比给出取 log 的似然值，此时测试数据上的总似然值 (log-likelihood) 为

$$L(\theta) = \frac{1}{N} \sum_{n=1}^{N} log(P(\mathbf{v}_n; \theta)$$

Algorithm 5 AIS 算法

1. 输出 β_k,$0 = \beta_0 < \beta_1 < \cdots < \beta K = 1$
2. 从 $P_A = P_0$ 抽样得到 x_1;
3. **for**$k = 1 : K - 1$

   使用 Markov 转移算子 $T_k(\mathbf{x}_{k+1} \to \mathbf{x}_k)$ 由 $\mathbf{x}_k$ 抽样得到 $\mathbf{x}_{k+1}$

   end
4. 令 $\omega_{AIS} = \prod_{k=1}^{K} P_k^* \mathbf{x}_k / P_{k-1}^* \mathbf{x}_k$
5. $Z_B \approx Z_A \frac{1}{M} \sum_{i=1}^{M} \omega_{AIS}^{(i)}$

3.2 Annealed Importance Sampling 算法

3.2.1 算法描述

AIS(Annealed Importance Sampling) 算法由 Radford M. Neal 于 1998 年提出 [10]，其核心思路是：使用 “中间分布”，从一个已知的简单分布出发，经过一系列中间分布，最终估计得到 RBM 的归一化常数。算法描述详见 Algorithm5。

3.2.2 算法实现

AIS 算法的实现需要以下数据，包括：RBM 模型的参数 $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{a}\}$、用于生成中间分布的 β 以及 AIS 算法的执行次数 M，还需要选择参考的基准分布 (base-rate model)。其中，RBM 模型的参数已经给出，是根据训练数据使用相关算法所得。因而，参数 β 和 M 是影响 AIS 算法的两个重要因素。

初步实现 AIS 算法时，base-rate model 的选择越简单越好，因而使用的 base-rate model 极为简单：$\mathbf{W}$ 元素全为 0，$\mathbf{a}$ 的元素全为 1，$\mathbf{b}$ 的元素全为 0。其他参数使用建议参数:β_k 在 0 至 0.5 均匀取 500 个，0.5 至 0.9 均匀取 4000 个，0.9 至 1.0 均匀取 10000 个，共计 14500 个中间分布。AIS 算

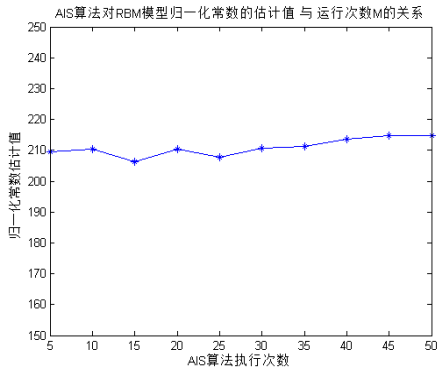

图 9: AIS 算法–归一化常数估计值与 AIS 执行次数的关系

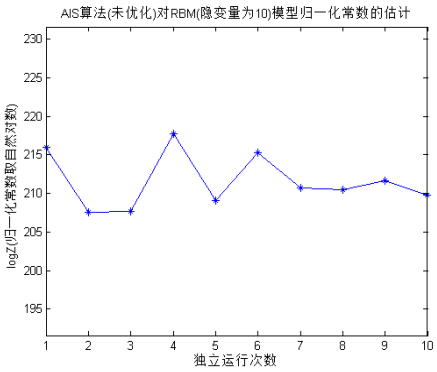

图 10: AIS 算法初步实现–对隐变量为 10 的 RBM 模型的归一化常数估计

法执行次数 M 选为 50 次。

以隐变量为 10 的 RBM 模型为例，图10展示了 AIS 算法初步实现的结果，从图中可以看出，当多次独立运行 AIS 算法时，对 R 归一化常数的估计值波动很大，表明当前 AIS 算法的参数选择不太合适，因而需要一定的优化。

3.2.3 算法分析和算法优化

本部分对 AIS 算法参数的选择进行了细致地对比分析，并根据结论对 AIS 算法的参数进行优化，同时对 base-rate model 的选择进行了思考。经过优化之后，归一化常数估计值稳定性明显提高。

a) AIS 算法执行次数 M

AIS 执行次数越多，最终估计得到的结果也越精确和稳定，但是当执行次数足够大时，多余的运行反而会浪费很长时间，降低计算效率。因而，为了兼顾估计值的准确性和计算的效率，现对 AIS 算法执行次数不同时的估计值结果进行比较，找到既能保证精确性又不会造成太大计算压力的执行次数 M。

图9展示了归一化常数估计值与 AIS 执行次数的关系，可见即使执行次数较低，估计值已经基本接近于稳定值，因而为了提高计算效率，可以适当减小 AIS 执行次数 M。

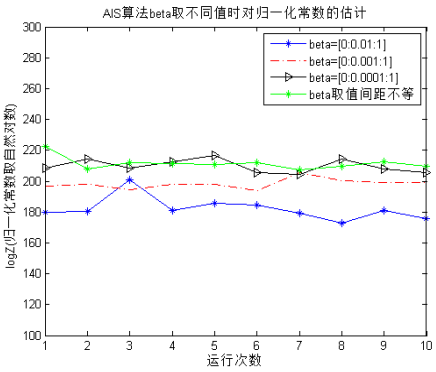

图 11: AIS 算法–归一化常数估计值与 β 取值的关系

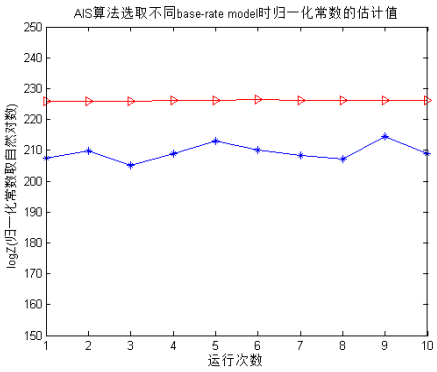

图 12: AIS 算法–base-rate model 不同时的归一化常数估计值

b) β_k 的选择

β 向量用于生成中间分布，β 元素的个数和取值之间影响了中间分布。与执行次数 M 一样，AIS 算法同样需要在中间分布数目和计算效率之间加以权衡。同时，β 的取值也需要研究。

图11展示了 β 取不同值时对归一化常数的估计 (AIS 执行次数 $M = 10$)。

c) 改变 base-rate model

在之前的实现中, base-rate model 都选择了一个非常简单的 RBM 模型, 这一模型的参数与真实的 RBM 模型相差太大, 为了减小 base-rate model 与实际 RBM 模型之间的差距, 可以使用测试数据的可见层数据生成 base-rate model 的可见层的偏置向量。

图12展示了 base-rate model 更改前后的对归一化常数的估计值比较。可以看出，参数优化后的 AIS 算法的估计值稳定性更好。

3.2.4 参数优化后的 AIS 算法

优化后的参数为:

(1) $M = 20$;

(2) $\beta = [0 : 1/1000 : 0.50.5 : 1/10000 : 0.90.9 : 1/100000 : 1.0]$;

(3) 使用测试数据的可见层数据生成 base-rate model 的可见层的偏置

向量。

表2展示了 AIS 算法的最初实现和优化后的结果对比，可以看出优化后的 AIS 算法其结果的稳定性明显增强，运行时间也大幅度缩短，说明在结果精确性和计算的效率两方面都得到了提高。

算法	归一化常数均值	归一化常数方差	运行时间
最初实现	213.5859	14.6869	240.246s
参数优化后	226.0064	0.0486	129.734s

表 2: AIS 算法优化前后结果对比

3.3 Thouless-Anderson-Palmer 算法

3.3.1 算法描述

TAP(Thouless-Anderson-Palmer) 算法，借用了统计物理的思想，通过高阶近似，提出一种计算 RBM 归一化常数的新思路 [11]。事实上，TAP 算法的本质是不动点迭代法求全局收敛值。TAP 算法计算归一化常数时使用了高阶近似，保留的阶数越高，对归一化常数的估计越精确。

3.3.2 TAP 算法 (二阶和三阶) 实现

TAP 算法的思路比较简单。首先引入了表征可见单元和隐藏单元的向量 $\mathbf{m^v}$ 和 $\mathbf{m^h}$, 然后 Γ 为 $(\mathbf{m^v}, \mathbf{m^h})$ 的函数，通过不动点迭代的方法，$\mathbf{m^v}$ 和 $\mathbf{m^h}$ 收敛到确定的值，此时 $(\mathbf{m^v}, \mathbf{m^h})$ 对应的 $\Gamma(\mathbf{m^v}, \mathbf{m^h})$ 即为所求的归一化常数 (取自然对数)。

由于 Γ 为 $(\mathbf{m^v}, \mathbf{m^h})$ 实际上是近似函数，一般选用二阶和三阶近似，更高阶的近似将会导致计算效率的严重下降。因而，本文实现了 TAP 算法的二阶近似和三阶近似，并对比两种近似的估计结果。

3.3.3 算法分析与比较

现分别对隐变量为 10,20,100,500 的 RBM 模型，二阶和三阶 TAP 算法独立运行多次，得到归一化常数的均值和方差。

隐变量数	算法	均值	方差	运行时间
10	TAP 算法 (二阶)	208.3563	89.5157	0.010s
10	TAP 算法 (三阶)	203.9581	255.3639	0.198s
20	TAP 算法 (二阶)	196.3531	92.8964	0.020s
20	TAP 算法 (三阶)	199.6899	84.9831	0.676s
100	TAP 算法 (二阶)	328.2481	47.9057	18.284s
100	TAP 算法 (三阶)	334.9081	7.6280	28.734s
500	TAP 算法 (二阶)	449.8376	0.0697	24.284s
500	TAP 算法 (三阶)	449.9967	0.0000	38.734s

表 3: TAP 算法二阶近似与三阶近似估计归一化常数对比

从表3中可以看出因变量数增加时,三阶 TAP 算法的稳定性比二阶 TAP 算法的稳定性相对较高。而且从运行时间来看，TAP 三阶算法的时间比二阶算法略长一些，计算效率有所降低。

另外，有以下两点值得关注的问题：

a) 即便 TAP 算法是使用不动点迭代的思想求得收敛值，但是估计得到的归一化常数仍然有一定波动，尤其是对隐藏变量较少的 RBM 模型，波动十分明显。因而 TAP 算法对隐藏变量较多的模型优势更加明显一些。

b) 如果使用 rand 函数生成随机数，那么得到的估计结果波动性很强；但是如果初值确定，得到的估计结果基本保持稳定。

3.4 Rao-Blackwellized Tempered Sampling 算法

RTS 算法的思想和 AIS 算法有异曲同工之处，同样都运用了中间分布的思想。使用 $\{0 = \beta_1 < \beta_2 < \cdots < \beta_K = 1\}$ 生成中间分布函数 $f_k(x)$。$p_1(x)$ 是易于采样的分布函数, 而 $f(x)$ 是尚未归一化的目标分布, $p(x|\beta_1) = p_1(x)$, Z_K 即为目标的归一化常数。[12]

RTS 算法计算归一化常数的式子如下式：

$$\hat{Z}_k^{\mathrm{RTS}} = \hat{Z}_k \frac{r_1}{r_k} \frac{\hat{c_k}}{\hat{c_1}}$$

其中，$\hat{Z}_k$ 是 Z_k 的近似值，$r_k = r(\beta_k)$ 是 β_k 的先验分布，[12] 中建议

Algorithm 6 Rao-Blackwellized Tempered Sampling 采样算法

输入: $\{\beta_k, r_k\}_{k=1,\cdots,K}$, N

初始化: $log\hat{Z}_k, k = 2, \cdots, K$;$\beta \in \{\beta_1, \cdots, \beta_K\}$;$\hat{c}_k = 0, k = 1, \cdots, K$

输出: $\hat{Z}_k^{\mathrm{RTS}}$

for $t = 1 : N$

1. 使用 Markov 转移算子使得 $q(x|\beta)$ 不变;
2. 从 $q(\beta|x)$ 中抽样得到 $\beta|x$
3. 更新 $\hat{c}_k = \hat{c}_k + \frac{1}{N} q(\beta_k|x)$

end

更新 $\hat{Z}_k^{\mathrm{RTS}} = \hat{Z}_k \frac{r_1}{r_k} \frac{\hat{c_k}}{\hat{c_1}}, k = 2, \cdots, K$

可以选择为均匀分布，即 $r_k = \frac{1}{K}$。其他参数的计算方法及解释见表4

公式	解释
$f_k(x) = f(x)^{\beta_k} p_1(x)^{1-\beta_k}$	中间分布
$p(x\|\beta_k) = \frac{f_k(x)}{Z_k}$	条件分布 $x\|\beta_k$
$Z_k = \int f_k(x)dx$	中间分布的归一化常数
$\hat{c_k} = \frac{1}{N} \sum_{i=1}^{N} q(\beta_k\|x^{(i)})$	$q(\beta_k)$ 的无偏估计

表 4: 对 RTS 算法参数的解释

RTS 算法可以总结为 Algorithm6.

3.5 总结

3.5.1 三种算法比较

本文一共实现了 AIS、TAP、RTS 共三种估计 RBM 模型归一化常数的算法，每种算法各有优缺点。图13和表5以隐变量为 10 的 RBM 模型 (对应于 h10.mat 数据) 为例，细致比较了三种算法 (TAP 算法使用二阶和三阶两种近似) 的各运行 10 次的结果 (均值和方差) 和运行时间。从图13和表5中

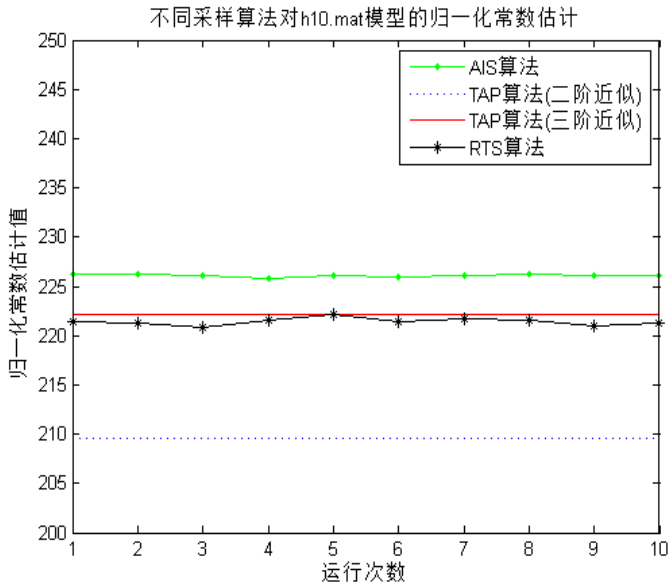

图 13: 不同采样算法对归一化常数的比较

可以得到以下结论:

算法	均值	方差	运行时间
AIS 算法	226.0957	0.0122	18.639s
TAP 算法 (二阶)	209.5460	8.9755e-28	0.420s
TAP 算法 (三阶)	222.1476	0	0.942s
RTS 算法	221.4095	0.0148	12.462s

表 5: 不同采样算法性能比较

(a) AIS 的优点是计算结果精确性和稳定性都很高，但是计算时间过长，即计算效率较低。

(b) TAP 算法整体的计算效率较高，但是计算结果的稳定性相对于 AIS 较差。并且，三阶近似时比二阶近似的结果稳定好一些，运行时间有所增加。

(c) RTS 算法的估计结果于 AIS 相比稳定性较差，但是略优于 TAP 算法；运行时间比 AIS 短。

总之，三种算法都存在着“精确性与计算效率”不可兼得的矛盾，因为如果要想得到更加准确稳定的结果，往往需要耗用更长的时间、动用更多的资源来实现这一点。在实际应用中，应当视需要选择合适的算法估计归一化常数。

事实上，由于超级计算机等技术的发展，计算时间已经不再是非常严重的影响因素，因而本文所给出的归一化常数和似然值的结果都是通过 AIS 算法得到的。

3.5.2 基本结论

表6给出了使用 AIS 算法对四种 RBM 模型的归一化常数的估计结果[2]，以及由归一化常数计算的在测试数据 test.mat 上的总似然值。

RBM 隐变量数	归一化常数估计 (取 log)	总似然值	log-likelihood
10	226.1046	1.5272e-31	-173.8708
20	221.0592	9.1561e-21	-145.1603
100	348.3669	8.6656e-14	-112.8921
500	460.1208	2.1974e-13 7	-100.6043

表 6: 归一化常数估计值和测试数据上的似然值结果汇总

3.6 训练受限玻尔兹曼机

根据已有的 train.mat 训练数据，本文尝试使用对比散度 (CD, Contrastive Divergence) 算法从训练数据中得到不同隐变量数目的 RBM 模型，再运用之前实现的 AIS 等算法估计模型归一化常数，并在估计结果上计算出测试数据上的似然值。

3.6.1 Contrastive Divergence 算法

Contrastive Divergence 算法是 Hinton 在 2002 年提出了 RBM 的一种快速学习算法 [13]，有关 CD 算法，详见 Algorithm7。

[2]表中的 e-31 表示 10 的-31 次方，依次类推。

Algorithm 7 Contrastive Divergence 算法

输入: 一个训练样本 x_0; 隐藏单元个数 m; 学习率 ϵ; 最大训练周期 T.

输出: 可见层和隐藏层的连接矩阵 $\mathbf{W}$，可见层的偏置 (bias) 向量 $\mathbf{a}$、隐层的偏置向量 $\mathbf{b}$.

训练模型

初始化: 令可见层单元的初始状态 $\mathbf{v}_1 = \mathbf{x}_0$;$\mathbf{W}$、$\mathbf{a}$、$\mathbf{b}$ 为随机的较小数值。

for $t = 1 : T$

for $j = 1 : m$(对所有隐藏单元) 计算 $P(\mathbf{h}_{1j} = 1|\mathbf{v}_1) = \sigma(b_j + \sum_i v_{1i}\mathbf{W}_{ij})$; 从条件分布 $P(\mathbf{h}_{1j}|\mathbf{v}_1)$ 中抽取 $h_{1j} \in \{0, 1\}$. **end**

for $i = 1 : n$(对所有可见单元) 计算 $P(\mathbf{v}_{2i} = 1|\mathbf{h}_1) = \sigma(a_i + \sum_j \mathbf{W}_{ij}h_{1j})$; 从条件分布 $P(\mathbf{v}_{2i}|\mathbf{h}_1)$ 中抽取 $h_{2i} \in \{0, 1\}$.

for $j = 1 : m$(对所有隐藏单元) 计算 $P(\mathbf{h}_{2j} = 1|\mathbf{v}_2) = \sigma(b_j + \sum_i v_{2i}\mathbf{W}_{ij})$; **end**

按照下式更新各参数:

1. $W = W + \epsilon(P(\mathbf{h}_1 = 1|\mathbf{v}_1\mathbf{v}_1^{\mathrm{T}}) - P(\mathbf{h}_2 = 1|\mathbf{v}_2\mathbf{v}_2^{\mathrm{T}}))$;
2. $a = a + \epsilon(\mathbf{v}_1 - \mathbf{v}_2)$;
3. $b = b + \epsilon(P(\mathbf{h}_1 = 1|\mathbf{v}_1) - P(\mathbf{h}_2 = 1|\mathbf{v}_2)$;

end

3.6.2 训练结果

本文主要使用 MATLAB 中的 DeepLearnToolBox[3]对 RBM 模型进行了训练，得到了隐变量分别为 10,20,100,500 的 RBM 模型。使用 AIS 算法估计训练的模型的归一化常数结果，并计算测试数据上的总似然值，得到表7所示的结果。

[3] `https://github.com/rasmusbergpalm/DeepLearnToolbox`

3.6.3 对比与分析

将表7与表6的结果对比，发现对于隐变量数目相同的 RBM 模型，本文训练的结果似然值较低，根据 RBM 模型最大似然的学习目标，可见已有的 RBM 模型比自行训练的 RBM 模型更好。

RBM 模型的隐变量数	归一化常数估计值 (取 log)	测试数据上的似然值
10	188.3565	8.2051e-37
20	219.1713	1.2873e-29
100	331.8111	1.4060e-19
500	442.8689	7.0230e-22

表 7: 重新训练 RBM 模型, 归一化常数估计值和测试数据上的似然值结果汇总

4 结论

本文主要讨论了马氏链蒙特卡洛方法的应用，没有提出新的算法，但是在实现算法的过程中，本文也做出了一些比较细致而有新意的工作，优化了算法对参数的估计效果。

第一部分通过马氏链蒙特卡洛方法中的 Metropolis-Hastings 算法生成了二维高斯分布的随机样本，并且从迭代初值的选取、提议函数的选取、生成随机样本数的选择、判断随机样本收敛以及提高随机样本独立性等五个方面深入讨论了算法中参数选择的原则，从而使得对相关系数的估计值精确性和稳定性明显提高。另外，还实现了 Gibbs 采样算法，将结果进行了对比分析，得到了一些有价值的结论。

第二部分实现了三种主要的采样算法：AIS、TAP(二阶和三阶)、RTS，对四种 RBM 模型的归一化常数进行估计，AIS 和 RTS 都用到了马氏链蒙特卡洛方法的基本思想。另外，本文着重对 AIS 的相关参数也进行了深入的讨论，获得了有价值的结果。最后还尝试使用对比散度 (CD) 算法训练了 RBM，与所给的 RBM 模型比较，发现自己训练的结果与所给的 RBM 模型仍有一定差距。

总之，本人尽力完成了力所能及的工作。由于我的学术水平有限，SAMS 算法无法按时实现，而且所写文章难免有不足之处，恳请老师和助教批评和指正！

5 致谢

历时将近一个月的时间，终于将本次关于马氏链蒙特卡洛的大作业基本完成。本次大作业在阅读文献时，遇到了不少疑问，首先感谢随机过程课程的欧智坚老师和戴音培等助教的耐心解释与帮助，同时也要感谢余东翰、王禹等大神的耐心解答和无私帮助，让我对算法有了比较高层次的认识。最后在自己训练受限玻尔兹曼机时，由于时间比较仓促，所以使用了 Rasmus Berg Palm 在 Github 上的 DeepLearnToolBox，这一工具箱对我的部分工作起到了很大的帮助。再次，特向以上对我有重要帮助的人表示衷心的感谢！

参考文献

[1] Chib, Siddhartha, and Edward Greenberg. “Understanding the metropolis-hastings algorithm.” *The American statistician* 49.4 (1995): 327-335.

[2] Metropolis, Nicholas, et al. ”Equation of state calculations by fast computing machines.” *The journal of chemical physics* 21.6 (1953): 1087-1092.

[3] Gelman, Andrew, and Donald B. Rubin. ”A single series from the Gibbs sampler provides a false sense of security.” *Bayesian statistics* 4 (1992): 625-631.

[4] Bishop, C. M. “Bishop Pattern Recognition and Machine Learning.” (2001).

[5] Liu, Jun S. Monte Carlo strategies in scientific computing. *Springer Science and Business Media*, 2008.

[6] 陈平, 徐若曦. Metropolis-Hastings 自适应算法及其应用 [J]. 系统工程理论与实践, 2008, 28(1): 100-108.

[7] 林元烈. 应用随机过程 [M]. 清华大学出版社有限公司, 2002.

[8] Salakhutdinov, Ruslan. “Learning deep generative models.” Diss. University of Toronto, 2009.

[9] Ackley, David H., Geoffrey E. Hinton, and Terrence J. Sejnowski. “A learning algorithm for Boltzmann machines.” *Cognitive science* 9.1 (1985): 147-169.

[10] Neal, Radford M. “Annealed importance sampling.” *Statistics and Computing* 11.2 (2001): 125-139.

[11] Gabrié, Marylou, Eric W. Tramel, and Florent Krzakala. "Training Restricted Boltzmann Machine via the Thouless-Anderson-Palmer free energy." *Advances in Neural Information Processing Systems.* 2015.

[12] Stinson, Patrick, Ari Pakman, and Liam Paninski. "Partition Functions from Rao-Blackwellized Tempered Sampling."

[13] Bengio, Yoshua. "Learning deep architectures for AI." *Foundations and trends® in Machine Learning* 2.1 (2009): 1-127.

随机过程第一次 Project 代码清单

无 47　　刘前*　　2014011216

2016 年 11 月 14 日

1 Part A: Metropolis-Hastings 算法应用

文件名	文件功能
bigauss.m	给定高斯分布的概率密度函数
MH_Origin.m	MH 算法的初步实现
MH_Origin_Test.m	MH 算法的初步实现 (测试性能)
MH_Sigma.m	对 MH 算法中提议函数方差的探究
MH_Sigma_Test.m	比较提议函数方差不同时 MH 算法的性能
MH_SampleNum.m	对 MH 算法中随机样本数目的探究
MH_SampleNum_Test.m	比较随机样本数不同时 MH 算法的性能
MH_Step.m	对 MH 算法中随机样本间隔选取的探究
MH_Step_Test.m	比较选取随机样本间隔不同时 MH 算法的性能
MH_Gibbs.m	实现 Gibbs 算法
MH_Gibbs_Test.m	测试 Gibbs 算法的性能
MH_Optimal.m	各参数优化后的 MH 算法
MH_Compare.m	对优化前后的算法进行对比

表 1: Part A 代码清单–MH 文件夹

*清华大学电子工程系 (E-mail: liuqian14@mails.tsinghua.edu.cn)

2 Part B: RBM 模型归一化常数的估计

文件名	文件功能
AIS2.m	AIS 算法实现
TAP2.m	TAP 算法 (二阶近似) 实现
TAP3.m	TAP 算法 (三阶近似) 实现
RTS.m	RTS 算法实现
LikeliHood.m	计算测试数据上的总似然值
Compare.m	比较各算法的性能
run.m	产生要求的 z.mat

表 2: Part B 代码清单–run 文件夹

数据文件名	数据
h10.mat	隐变量为 10 的 RBM 模型
h20.mat	隐变量为 20 的 RBM 模型
h100.mat	隐变量为 100 的 RBM 模型
h500.mat	隐变量为 500 的 RBM 模型
test.mat	测试数据
train.mat	训练数据

表 3: Part B 代码清单–Data 文件夹

文件名	文件功能
AIS1.m	AIS 算法实现 (无分批数据)
AIS2.m	AIS 算法实现 (有分批数据)
AIS_Test.m	AIS 算法性能测试
LikeliHood.m	计算测试数据上的总似然值
AIS_Beta.m	探究 β 参数对 AIS 算法的影响
AIS_Mrun.m	探究执行次数 M 对 AIS 算法的影响
AIS_Data.m	探究是否使用分批数据对 AIS 算法的影响

表 4: Part B 代码清单–01AIS 文件夹

文件名	文件功能
TAP2.m	TAP 算法 (二阶近似) 实现
TAP3.m	TAP 算法 (三阶近似) 实现
TAP_Test.m	TAP 算法测试
LikeliHood.m	计算测试数据上的总似然值

表 5: Part B 代码清单–02TAP 文件夹

文件名	文件功能
RTS.m	RTS 算法实现
RTS_Test.m	RTS 算法测试

表 6: Part B 代码清单–03RTS 文件夹

文件名	文件功能
sigm.m	Sigmoid 函数
sigmrnd.m	生成服从 Sigmoid 函数的随机数
rbmtrain.m	RBM 模型的训练函数
dbnsetup.m	DBN（深度信念网络）的建立
dbntrain.m	DBN 的训练
RBM_Training.m	执行 RBM 模型训练，得到 h*.mat
AIS2.m	AIS 算法估计归一化常数
LikeliHood.m	计算测试数据上的总似然值
trained$_r un.m$	估计归一化常数并计算似然值

表 7: Part B 代码清单–04Training 文件夹