The Markov Chain Monte Carlo Simulations

last modified October 18, 2007

在貝氏推論 (Bayesian Inference) 的應用範疇裡, 常需要計算後驗機率

$$p(\theta|\mathbf{y}) = \frac{p(\mathbf{y}|\theta)p(\theta)}{\int p(\mathbf{y}|\theta)p(\theta)d\theta} \tag{1}$$

或是其參數 (θ) 的貝氏估計

$$\hat{\theta} = E\left[\theta|\mathbf{y}\right] = \int \theta p(\theta|\mathbf{y})d\theta = \frac{\int \theta p(\mathbf{y}|\theta)p(\theta)d\theta}{\int p(\mathbf{y}|\theta)p(\theta)d\theta} \tag{2}$$

其中 $p(\theta)$ 爲先驗機率 (prior probability)。當計算式中的積分非常困難甚至不可能分析時, 數値的方法變成唯一的選擇, 其中 MCMC(Markov Chain Monte Carlo) 是近十年來最熱門的方法, 應用的領域十分廣泛。本單元將介紹幾種 MCMC 演算法, 配合簡單的範例, 將使讀者對 MCMC 型態的演算法有深刻的印象。

本章將學到關於程式設計

較複雜演算法的程式設計概念。

〈本章關於 MATLAB 的指令與語法〉

指令: mhsample, slicesample

語法:

1 背景介紹

MCMC由兩部分的觀念 (步驟) 組成, 其一為「Markov Chain process,」另一個是「Monte Carlo integration,」分述如下:

1.1 Monte Carlo Integration

Monte Carlo integration 以抽樣平均的方式計算式 (2) 的期望值

$$E\left[\theta|\mathbf{y}\right] \approx \frac{1}{n}\sum_{t=1}^{n}\theta_t|\mathbf{y} \tag{3}$$

其中樣本$\{\theta_t|t=1,2,\cdots,n\}$ 來自機率密度函數為 $p(\theta|\mathbf{y})$ 的母體。換句話說, 以樣本平均數來估計期望值。在大數法則的原理下, 當樣本數 n 夠大時, 樣本平均數將趨近母體的均數。

Monte Carlo integration 讓積分變得簡單, 但是實際的情況卻是, 從 $p(\theta|\mathbf{y})$ 抽樣並非易事, 特別當 $p(\theta|\mathbf{y})$ 不是一般標準的分配時, 時下常用的一些軟體並沒有支援。Markov Chain Process 提供了在這樣的情況下產生樣本的方式。

1.2 Markov Chain Process

假設 $X_0, X_1, X_2, \cdots$為一序列變數, 且 X_{t+1} 來自 $p(X_{t+1}|X_t)$ 的機率分配, 換句話說每個變數間是相關的, 但僅與前一個變數有關。這樣的序列變數稱為 Markov Chain(馬可夫鍊), $P(X_{t+1}|X_t)$被稱為這個馬可夫鍊的轉換核心 (transition kernel)。在一般假設的條件下, 這個馬可夫鍊的變數分配將收斂到目標機率函數 $\pi(\cdot)$ 並且與 X_0 的選擇無關 [1]。也因此在收斂情況下的樣本可以代表原分配, 而且當適當的選擇轉換核心 $p(X_{t+1}|X_t)$ 時, 解決前述 Monte Carlo 積分時樣本產生的問題。這兩個觀念的結合衍生出許多精彩的演算法。

1.3 Markov Chain Monte Carlo

本節介紹兩種結合 Markov Chain process 與 Monte Carlo integration 的演算

法(一般通稱 MCMC 演算法), 配合練習題, 將使讀者對 MCMC 有清楚與概括的瞭解。

Metropolis method:

Metropolis 演算法的概念很簡單, 可以下列步驟來說明:

---

1. 令 X_0 爲任何變數的樣本

2. 第 $k+1$ 個變數的樣本爲

$$y = X_k + s, \qquad k = 0, 1, 2, \cdots$$

   其中 s 服從均等分配, 即 $s \sim U(-a, a)$

3. 產生一個亂數 $u \sim U(0, 1)$

4. 如果

$$u \leq \frac{\pi(y)}{\pi(X_k)}$$

   則 $X_{k+1} = y$, 否則 $X_{k+1} = X_k$(即下一個變數仍維持 X_k), 其中 $\pi(\cdot)$ 爲目標機率函數。[1]

Metropolis 演算法有幾個地方值得注意

- 步驟 2 的均等分配範圍 $(-a, a)$ 的選擇決定了收斂的速度, 若 a 過小, 馬可夫鍊的移動太慢, 收斂比較慢。過大的 a 容易導致步驟 4 的拒絕率升高,X_{k+1} 傾向不變動。

- 透過上述的演算法將產生一系列的樣本, 最初的樣本與目標機率函數關係不大, 在不斷的透過步驟 4 的限制, 樣本將慢慢具備服從目標機率函數的特性。所以當

[1] 我們希望達穩定狀態後的馬可夫鍊樣本, $X_{m+1}, X_{m+2}, \cdots$, 服從目標機率函數 $\pi(\cdot)$ 分配。

以樣本平均估計期望值時, 通常會捨棄前 m 個樣本 (俗稱 burn-in samples), 即

$$\hat{\theta} = \frac{1}{n-m}\sum_{k=m+1}^{n} X_k$$

這當然是因爲前面的樣本還在調整修正中, 未達穩定狀態, 不適合當作目標機率函數的樣本。

Metropolis-Hastings method:

Hastings 修正了 Metropolis 對樣本的產生 (步驟 2) 與認定 (也就是步驟 4 的「接受或拒絕」的準則)。Metropolis-Hastings 演算法如下

---

1. 令 X_0 爲任何變數樣本
2. 第 $k+1$ 個變數的樣本爲
$$y \sim q(Y|X_k)$$
3. 產生一個亂數 $u \sim U(0,1)$
4. 如果
$$u \leq r_k$$
則 $X_{k+1} = y$, 否則 $X_{k+1} = X_k$(即下一個變數仍維持 X_k), 其中
$$r_k = min\left(1, \frac{\pi(Y)q(X_k|Y)}{\pi(X_k)q(Y|X_k)}\right)$$
$q(\cdot|\cdot)$ 爲提議機率函數 (proposal distribution) 或轉換函數 (transition function), Markov chain 的轉換機率寫成 $p(\cdot|\cdot) = q(\cdot|\cdot)r$

新樣本來自提議機率函數 $q(Y|X)$, 這個函數的選擇當然扮演重大的角色, 神奇的是, $q(Y|X)$ 可以是任何型態的函數, 其所產生的 Markov chain 達穩定狀態後 (stationary) 的分配將是 $\pi(\cdot)$。前面 Metropolis 演算法所採用的方式一般稱爲 random walk chain, 也是 $q(Y|X)$ 的選擇之一。不過適當的選擇卻關係到收斂 (stationary) 的速度, 一般而言, 選擇 $q(Y|X)$ 的原則不外乎

- $q(Y|X)$ 愈接近目標機率函數 $\pi(Y)$ 愈好 (如範例 1 所示)。
- 容易自 $q(Y|X)$ 產生樣本。

Metropolis 演算法是 Metropolis-Hastings 演算法的特例, 主要的差別在其具對稱性的提議機率函數, 即 $q(Y|X) = q(X|Y)$, 這個假設影響了樣本的接受/拒絕率。[2]以上述 Metropolis 演算法步驟 2 爲例, 新樣本的產生可以表示爲

$$y \sim q(Y|X_k) = q(|Y - X_k|) = s \qquad\qquad s \sim U(0, a)$$

另外, $q(Y|X) = N(X, \sigma^2)$ 的常態假設也是常見的特例, 因爲 $q(Y|X) = q(X|Y)$。以下利用簡單的實作問題來瞭解 MCMC 的運作方式。

範例1: 假設目標函數 $\pi(\cdot)$ 爲標準常態 $N(0, 1)$, 寫一支程式利用 Metropolis 演算法產生 Markov Chain 樣本並測試當提議分配 $q(Y|X)$ 分別爲

1. 如 Metropolis 演算法步驟 2 的 Random walk $q(Y|X) = q(|Y - X|)$
2. $q(Y|X) = N(X, 0.5), N(X, 0.1), N(X, 10)$ 三種

畫圖觀察 Markov Chain 達穩定狀態的情況。

圖 1 展示了三種常態分配作爲提議分配的表現, 不難看出當提議分配愈接近目標分配時, 其收斂的速度比較理想。請注意這些常態分配都是以 Markov Chain 變數 X 做爲中心點, 並非固定。即新的樣本從「以前一個樣本做爲均數」的常態分配中抽樣。
從圖 1 也可以觀察到經過足夠長 (多) 的時間 (樣本) 以後, 譬如 100 個以後, Markov Chain 產生的相依性樣本愈接近從目標函數所產生的。這時便可以利用樣本平均數來計算母體 (目標函數) 均數的期望值 $E(X)$, 或是進一步的統計量估計 $E(f(X))$。而前一段被捨棄的 100 個樣本稱爲 Burn-in samples。
MATLAB也提供指令「mhsample」產生樣本, 試著使用看看, 觀察與自己寫的程式所產生的結果有何不同。譬如

[2] $\frac{\pi(Y)q(X_k|Y)}{\pi(X_k)q(Y|X_k)} = \frac{\pi(Y)}{\pi(X_k)}$

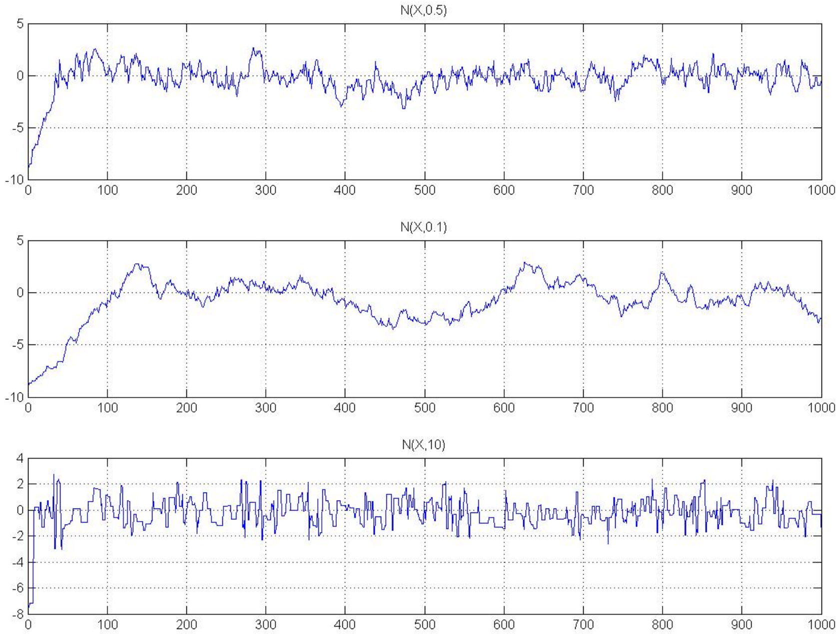

圖 1: Metropolis 演算法: 不同提議分配的收斂表現

```
sigma = sqrt(.5);
tarpdf = @(x) normpdf(x); % target pdf
proppdf = @(x,y) normpdf(y,x,sigma); %proposal pdf q(y|x)
proprnd = @(x) normrnd(x,sigma);%generate samples from proposal pdf
nsamples = 1500;
z = mhsample(1,nsamples,'pdf',tarpdf,'proprnd',proprnd,'symmetric',1);
histfit(z,50)
```

上述「mhsample」指令選擇了對稱性的提議分配, 所以第三行對提議分配 (proppdf) 的設定實際上是用不著的, 其結果與一般的用法相同。

```
z = mhsample(1,nsamples,'pdf',tarpdf,'proprnd',proprnd,'proppdf',proppdf);
```

「mhsample」指令除輸出樣本外, 也可以同時輸出「Acceptance Rate」, 以便觀察提議分配的表現, 讀者不妨參考線上手冊關於 mhsample 的其他選項。

1.4 Data Augmentation

當後驗機率密度函數 (1) 沒有分析解, 需要訴諸數值解時,Data Augmentation 的概念十分合適。如同 EM 演算法所引入的隱藏變數造成較簡單的概似函數型態,Data Augmentation 也是試圖在後驗機率的計算上引入隱藏變數, 讓這個計算變得可能與簡單, 其觀念如下式

$$p(\theta|\mathbf{y}) = \int_Z p(\theta|\mathbf{z},\mathbf{y})p(\mathbf{z}|\mathbf{y})dz \tag{4}$$

從 $p(\theta|\mathbf{y})$ 到 $p(\theta|\mathbf{z},\mathbf{y})$ 引進了隱藏變數 $\mathbf{z}$ 使得 $p(\theta|\mathbf{z},\mathbf{y})$ 變得更容易分析、計算或是抽樣, 當然最好是一個已知型態的分配。引進了 $\mathbf{z}$, 卻也同時多出個積分式, 當式(4) 的積分不容易做到時,Monte Carlo 積分是不錯的選擇, 於是式 (4) 可以看成

$$\begin{aligned} p(\theta|\mathbf{y}) &= \int_Z p(\theta|\mathbf{z},\mathbf{y})p(\mathbf{z}|\mathbf{y})dz \\ &= E_{\mathbf{z}|\mathbf{y}}\left(p(\theta|\mathbf{z},\mathbf{y})\right) \\ &\approx Ave(p(\theta|\mathbf{z}_i,\mathbf{y})) \end{aligned} \tag{5}$$

其中隱藏變數的資料 $\mathbf{z}_i$ 是從所謂的 preditive distribution $p(\mathbf{z}|\mathbf{y})$ 抽樣而來。$p(\mathbf{z}|\mathbf{y})$ 可以寫成

$$p(\mathbf{z}|\mathbf{y}) = \int_\Theta p(\mathbf{z}|\theta,\mathbf{y})p(\theta|\mathbf{y})d\theta \tag{6}$$

上式意謂著 $\mathbf{z}_i$ 的產生通常從 $p(\mathbf{z}|\theta,\mathbf{y})$ 更爲方便, 當利用 Monte Carlo 積分時, 積分變數 θ 的樣本必須從 $p(\theta|\mathbf{y})$ 抽樣而來。這於是形成了「雞生蛋, 蛋生雞」的循環遞迴問題: $p(\theta|\mathbf{y})$ 的計算依靠隱藏變數樣本 $\mathbf{z}_i$, 而 $\mathbf{z}_i$ 從 $p(\theta|\mathbf{y})$ 抽樣而來。這個 Data Augmentation 演算法可以寫成下列的兩個步驟:

I step(Imputation Step):執行下列兩個步驟 m 次, 產生 $\mathbf{z}_1,\mathbf{z}_2,\cdots,\mathbf{z}_m$

(1) 從 $p(\theta|\mathbf{y})$ 抽出一個樣本 θ_i。

(2) 依步驟 (1) 產生的 θ_i 值, 從 $p(\mathbf{z}|\theta_i, \mathbf{y})$ 抽出一個 $\mathbf{z}_i$

P step(Posterior Step): 更新後驗機率爲

$$p(\theta|\mathbf{y}) = \frac{1}{m}\sum_{i}^{m} p(\theta|\mathbf{z}_i, \mathbf{y}) \tag{7}$$

在 I step 中有一個 m 需要決定,Tanner & Wong 的論文 [2] 中說明 m 的選擇可以是變動的, 甚至低到 $m = 1$ 都可以保證迴圈移動的方向是正確的。另外在 P step 中更新的後驗機率可以被看成是一個混合機率密度函數(mixture distribution), 而在 I-step 中的 $p(\mathbf{z}|\theta_i, \mathbf{y})$ 必須是容易計算或容易抽樣的密度函數, 這個演算法才有實質的意義。這個演算法被證明 [2] 在大部分的假設情況下, 都能收斂到理論的後驗機率密度函數。

範例2: 舉著名的 Genetic Linkage Model 爲例,197 隻動物以多項分配的分佈屬於四種類別, 分別是

$$y = (y_1, y_2, y_3, y_4) = (125, 18, 20, 34)$$

其各項的機率分別是

$$\left(\frac{1}{2} + \frac{\theta}{4}, \frac{1-\theta}{4}, \frac{1-\theta}{4}, \frac{\theta}{4}\right)$$

欲估計未知參數 θ。

最直接的作法是訴諸最大概似函數的估計,

$$L(\mathbf{y}|\theta) = \max_{0\leq\theta\leq1} L(\mathbf{y}|\theta) \tag{8}$$

其中概似函數

$$L(\mathbf{y}|\theta) \propto (2+\theta)^{125}(1-\theta)^{38}\theta^{34}$$

參數 θ 的最大概似估計可以由解概似函數的一次導數爲 0, 或是輾轉的以 EM 演算法計算可得。另一種作法是更直接的以其條件式期望值作爲估計, 即

$$\hat{\theta} = E(\theta|\mathbf{y}) = \int \theta p(\theta|\mathbf{y}) d\theta \tag{9}$$

其中 $p(\theta|\underline{y})$ 爲後驗機率。上式可以進一步寫成

$$\hat{\theta} = E(\theta|\underline{y}) = \frac{\int \theta p(\underline{y}|\theta)\pi(\theta)d\theta}{\int p(\underline{y}|\theta)\pi(\theta)d\theta} \tag{10}$$

$\pi(\theta)$爲先驗機率。上式又稱爲貝氏估計。貝氏估計經常面臨到積分的問題, 特別當變數維度很高時, 積分的難度便高出許多, 往往得不到分析解(analytical solution)。以本題爲例, 假設對先驗機率一無所知, 即 $\pi(\theta) = 1, 0 \leq \theta \leq 1$, 式 (10) 寫成

$$\hat{\theta} = \frac{\int_0^1 \theta(2+\theta)^{125}(1-\theta)^{38}\theta^{34}d\theta}{\int_0^1 (2+\theta)^{125}(1-\theta)^{38}\theta^{34}d\theta} \tag{11}$$

僅是單一變數, 這樣的積分也夠嚇人的了!

這類的問題讓 MCMC 找到發揮的舞台,Monte Carlo 積分透過抽取大量樣本並計算其平均來近似期望值的作法, 可以避開繁瑣的積分。但是當抽取樣本所憑藉的後驗機率不是典型的機率密度函數時, 以本範例爲例

$$p(\theta|\mathbf{y}) \propto (2+\theta)^{125}(1-\theta)^{38}\theta^{34} \tag{12}$$

樣本的產生便是個問題。還好馬可夫鍊 (Markov Chain) 的概念提供一個解套的方式, 以下先利用1.3介紹的兩種 MCMC 演算法, 看看效果如何?

Metropolis Algorithm

圖2根據式 (12) 的機率密度函數以 Metropolis 演算法所產生 1000 個樣本畫出的直方圖。直方圖的分佈與式 (12) 非常的接近。從樣本計算式 (9) 的積分式, 得到的估計值如圖3中的紅線所示。當樣本數愈大時, 其估計值愈穩定, 表示從均等分配產生的

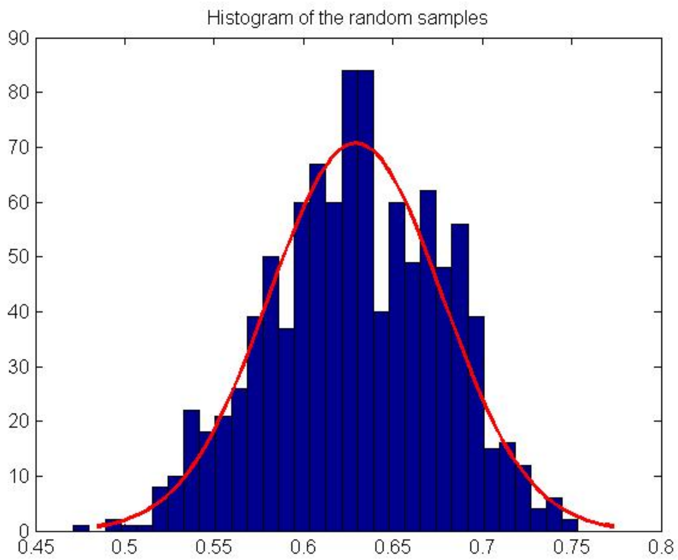

圖 2: 利用 Metropolis 演算法產生的樣本畫出的直方圖

樣本趨近目標機率函數 (12)。 請注意圖3的紅線是所謂的「累加平均值,」可以使用 MATLAB 的指令「cumsum」進行累加後再做平均, 譬如

```
y=cumsum(x)./(1:m)   % 向量 x 代表 1xm 的樣本
```

Metropolis-Hastings Algorithm

當應用 Metropolis-Hastings 演算法時, 轉換機率函數的選擇很關鍵。以本題的目標機率函數爲例 (如圖4實線), 可以選擇常態分配或選擇 Beta(56,38) (如圖4虛線) 當作轉換機率函數, 其實 Beta(56,34) 作爲 $q(Y|X)$ 最爲完美, 幾乎與目標機率函數一致, 以此轉換函數產生的樣本與平均數展示在圖5。程式如下

```
y=[125  18  20  34];
A=56;B=38;
tpdf=@(x) (2+x)^y(1)*(1-x)^(y(2)+y(3))*x^y(4);
ppdf = @(x,y) betapdf(x,A,B);
prnd = @(x) betarnd(A,B);
xsamples = mhsample(0.5,1500,'pdf',tpdf,'proprnd',prnd,'proppdf',ppdf);
histfit(xsamples,50)
```

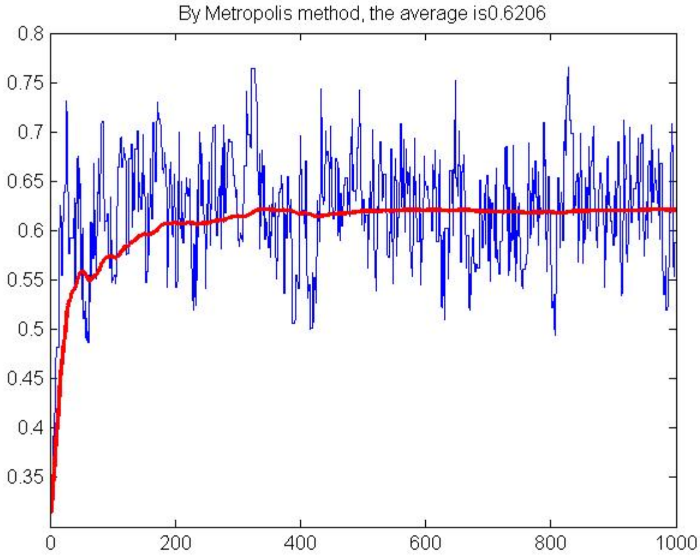

圖 3: 以樣本計算積分所得到的參數估計值。

Data Augmentation method

運用 Data Augmentation 的方式需要先解決三件事, 才開始進行其 I-P 步驟的演算法, 分別是

1. 選擇適當的隱藏變數 $\mathbf{z}$
2. 定義 preditive density $p(\mathbf{z}|\theta, \mathbf{y})$ 並能抽樣。
3. 定義加入隱藏變數後的後驗機率 $p(\theta|\mathbf{z}, \mathbf{y})$ 並能抽樣。

針對本題,Data Augmentation method 做如下的對應:

1. 完整的資料定義爲 $\mathbf{x} = (z_1, z_2, y_2, y_3, y_4)$, 其中 z_1, z_2 爲隱藏變數且 $z_1 + z_2 = y_1 = 125$。完整變數相對的機率分配爲

$$(\frac{1}{2}, \frac{\theta}{4}, \frac{1-\theta}{4}, \frac{1-\theta}{4}, \frac{\theta}{4})$$

這個定義使得完整資料的概似函數變得簡單(比較式(8) 的概似函數):

$$p(\mathbf{x}|\theta) \propto \theta^{z_2+y_4}(1-\theta)^{y_2+y_3}$$

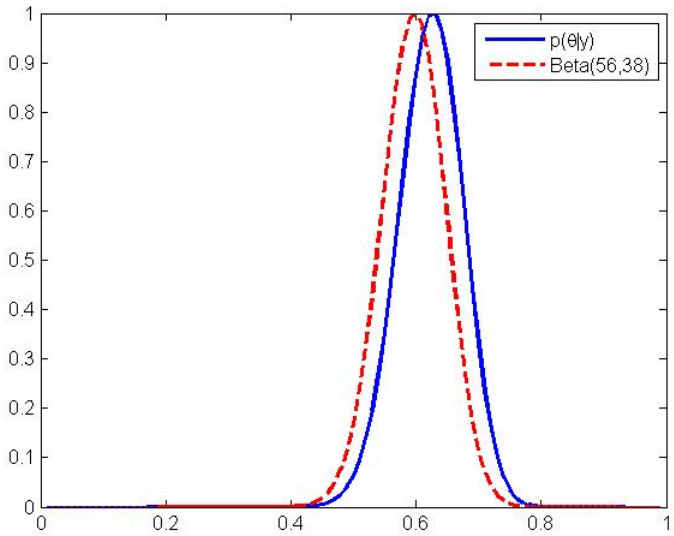

圖 4: 目標機率函數 $p(\theta|\mathbf{y})$ 與轉換機率函數 $q(Y|X)$

2. 隱藏變數的選擇使得 preditive density $p(\mathbf{z}|\theta, \mathbf{y})$ 定義爲二項分配

$$z_2 \sim Binomial(125, \frac{\theta}{\theta+2})$$

由於 (z_1, z_2) 爲二項分配,preditive density 可以直接以 $p(z_2|\theta, \mathbf{y})$ 表示。

3. 隱藏變數的後驗機率 $p(\theta|\mathbf{z}, \mathbf{y})$ 或直接寫成 $p(\theta|z_2, \mathbf{y})$ 表示爲

$$\begin{aligned} p(\theta|z_2, \mathbf{y}) &= \frac{p(z_2|\theta, \mathbf{y})p(\theta|\mathbf{y})p(\mathbf{y})}{\int_0^1 p(z_2|\theta, \mathbf{y})p(\theta|\mathbf{y})p(\mathbf{y})d\theta} \\ &= \frac{1}{B(\alpha, \beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1} \\ &= Beta_{\alpha,\beta}(\theta) \end{aligned} \quad (13)$$

其中假設 $p(\theta)$ 爲均等分配 $U(0, 1)$ 且 $Beta_{\alpha,\beta}$ 代表 Beta 分配的機率密度函數, 其參數分別爲

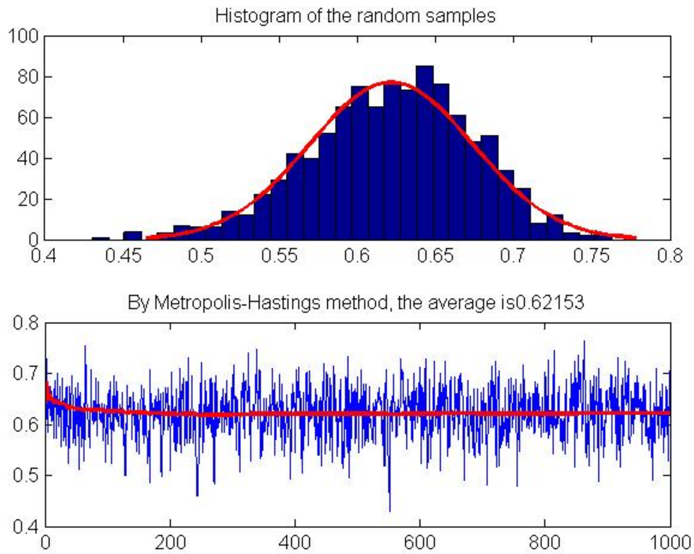

圖 5: 用 Metropolis-Hastings 演算法產生的樣本畫出的直方圖與樣本平均數

$$\begin{aligned}\alpha &= z_2 + y_4 + 1 \\ \beta &= y_2 + y_3 + 1\end{aligned}$$

$B(\alpha, \beta)$ 爲 Beta 函數。

於是 Data Augmentation I-P 演算法的第 i 次迴圈可以明確的寫成

I step(Imputation Step)：執行下列兩個步驟 m 次, 產生 $z_2^{(1)}, z_2^{(2)}, \cdots, z_2^{(m)}$

1. 依據式 (14) 從 $p(\theta^{(i-1)}|\mathbf{y})$ 抽出一個樣本 $\theta^{(i)}$。
2. 依前步驟產生的 $\theta^{(i)}$ 值, 從二項分配的 $p(z_2|\theta^{(i)}, \mathbf{y}) = Binomial(125, \frac{\theta^{(i)}}{\theta^{(i)}+2})$ 抽出一個 $z_2^{(k)}$。

P step(Posterior Step): 更新後驗機率爲

$$p(\theta|\mathbf{y}) = \frac{1}{m}\sum_{k=1}^{m} p(\theta|z_2^{(k)}, \mathbf{y}) = \frac{1}{m}\sum_{k=1}^{m} Beta_{\alpha_k,\beta_k}(\theta) \tag{14}$$

其中 $\alpha_k = z_2^{(k)} + y_4 + 1, \beta_k = y_2 + y_3 + 1$

圖6展示利用上述的演算法所估計的後驗機率密度函數, 與理論值非常貼近。主要的程式碼如下:

```
y=[125  18  20  34];  m=100;
theta=unifrnd(0,1,1,m);% initial guess for theta
%——- Do I-P iterations ——————————-
for j=1:100
      z2=binornd(y(1),theta./(theta+2));% I-step (2)
      a=z2+y(4)+1;b=y(2)+y(3)+1;
      theta=betarnd(a(unidrnd(m,1,m)),b,1,m);% I-step (1)
end
%——-To draw the final posterior density——-
t=0:0.01:1;  p_estimate=zeros(1,length(t))
for i=1:length(theta)
      p_estimate(i)=mean(betapdf(t(i),a,b*ones(1,m)));
end
plot(t,p_estimate/max(p_estimate),'r–')
%——- Compute where the maximum occurs——-
f=@(x) -mean(betapdf(x,a,b*ones(1,m)));
theta_max=fminbnd(f,0,1);
```

程式的第一個迴圈是 I-P 演算法, 其中的 P-step 並不明顯, 反而依附在 I-step(1) 裡面。也就是 I-step(1) 所需要 θ 是由 P-step 的後驗機率去抽樣的, 抽樣的過程已經使用了最新的後驗機率。第二個迴圈只是畫出最後更新完成的後驗機率。最後兩行計算出後驗機率的期望值。

2 觀察

1. 在使用 Metropolis 演算法時, 需要決定一個常數 a, 適當的選擇可以讓收斂的

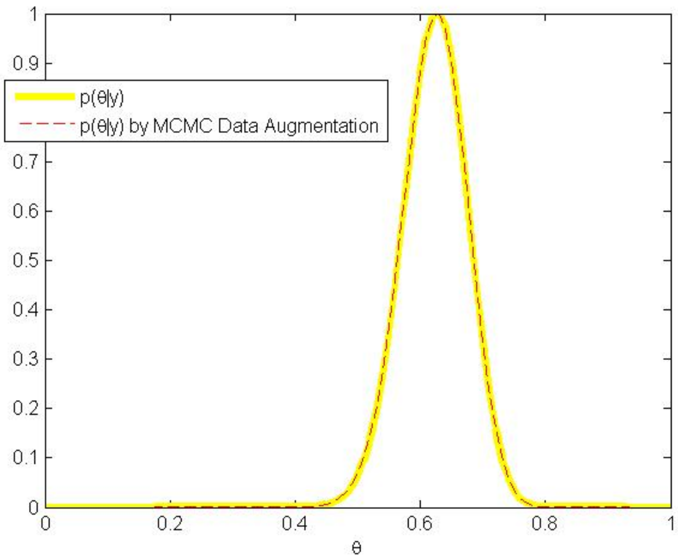

圖 6: 目標機率函數 $p(\theta|\mathbf{y})$ 與轉換機率函數 $q(Y|X)$

速度加快, 不妨試試看不同大小的 a 對收斂速度的影響, 畫出如圖3以方便觀察。

2. 範例 1 同時測試了 random walk 及常態分配兩種轉換函數, 這些選擇的差異值得觀察, 特別是 random walk 這個簡單又直覺的抽樣方式。

3. Metropolis-Hastings 演算法的收斂行爲受轉換機率影響甚鉅, 在範例中使用 Beta(56,34) 達到迅速的收斂效果, 但這並不表示任何目標機率函數都可以找到如此相近的轉換函數, 如果採用 Beta 分配的其他參數組合也可以達到收斂的效果嗎? 這值得動手做做看並比較收斂的情況。另外常態分配也是一種選擇, 都值得比較以瞭解轉換函數的選擇對樣本收斂的影響。

4. 在觀察轉換函數的適當與否時, 新樣本的「接受率」是觀察的指標, 這時可以在樣本產生的同時順便列印出 u 與 r 值, 更可以在最後計算接受率, 不失作爲判斷轉換機率函數優劣與否的依據。

5. 式 (14) 是一個 beta mixture 的混合機率密度函數, 抽樣時必須依據比例 $(1/m)$ 亂數選擇其中某個 Beta 分配。

6. Data Augmentation I-P 演算法需要 θ 起始值, 不同的選擇是否影響未來的結果?
7. MATLAB 除提供「mhsample」的 MCMC 指令外, 尚有「slicesample」利用 Slice Sample 的方式產生樣本, 不妨試看看。

3 作業

1. 求式 (8) 的最大概似函數估計。
2. 計算式 (11) 的積分式 (註: 利用 MATLAB 的積分指令)。
3. 利用 E-M 演算法計算式 (8) 的最大概似函數估計。
4. 區別 Data Augmentation 與 EM Algorithm 的異同。
5. 範例1測試不同提議分配的收斂速度, 請分別計算其接受/拒絕率。
6. 同範例 1, 將提議分配改爲與常態分配接近的 (1) 指數分配 (2) T 分配, 各分配的參數自行設定。畫出所有樣本與過程中的平均數 (不計 burn-in 的樣本), 如圖5下圖所示。
7. 詳細推展式 (13)。
8. 利用 MATLAB 的 MCMC 指令「mhsample」產生範例 2 M-H 演算法的樣本並繪製如圖5的直方圖與樣本的累加平均圖。

參考文獻

[1] W.R. Gilks, S.Richardson, D.J.Spiegelhalter, "Markov Chain Monte Carlo In Practice," Chapman&Hall.

[2] Martin A. Tanner and Wing Hung Wong,"The Calculation of Posterior Distribution by Data Augmentation", June 1987, vol. 82,Ni. 398, Theory and Methods, Journal of the American Statistical Association