多變量函數之最小值

last modified November 27, 2012

許多的應用問題, 如最大概似函數的參數估計, 最後都要計算多變量函數的最小值。[1]之前的單元也介紹過單一變數函數的最小值演算法, 本單元將之擴展到多個變數, 看看 MATLAB 如何解決這些問題, 並介紹著名的 Newton-Raphson method 及簡潔的 Steepest Descent method。

本章將學到關於程式設計

MATLAB 演算法指令的使用、一般演算法的程式寫作、立體圖形的繪製技巧及 MATLAB 圖形工具的使用。

⟨本章關於 MATLAB 的指令與語法⟩

指令:meshgrid, mesh, surf,contour, colorbar, fminsearch, optimset,fprintf

[1]最大值與最小值的計算被視爲同一個問題, 其差別只是目標函數的正負號而已。

1 背景介紹

多變量函數的最小值的問題表示如下:

$$\min_{\mathbf{x}} f(\mathbf{x}) \tag{1}$$

其中函數 $f(\mathbf{x}) : R^n \to R$, 假設爲可做二次微分的連續函數, 並且爲變數 $\mathbf{x}$ 的非線性函數。譬如圖1是兩個變量的函數圖, 有一個最小值在底部的凹陷處。

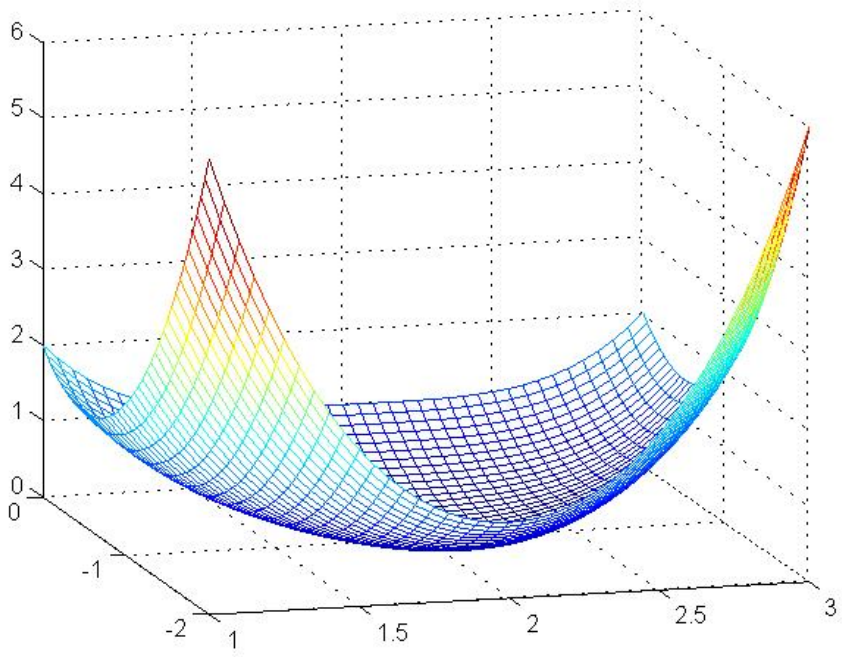


圖 1: 兩個變量的函數圖

計算 $f(\mathbf{x})$ 最小值的必要條件爲

$$\bigtriangledown f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix} = \mathbf{0} \tag{2}$$

即令梯度向量爲零向量。由於 $f(\mathbf{x})$ 的非線性關係, 上式爲一組非線性的聯立方程式, 通常推導不出所謂的 closed-form solution。因此必須藉助演算法的迭代方式逐步找到最小值。例如

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k \qquad\qquad k = 0, 1, 2, \cdots$$

在選擇一個適當的初始值 $\mathbf{x}_0$ 後, 透過適當的「方向」($\mathbf{d}_k$) 調整, 一步步的逼近函數的最小值。其中最負盛名的「方向」選擇爲 Newton-Raphson method, 其法則如下:

$$\mathbf{d}_k = -(\bigtriangledown^2 f(\mathbf{x}_k))^{-1} \bigtriangledown f(\mathbf{x}_k) \tag{3}$$

其中 $n \times 1$ 的向量 $\bigtriangledown f(\mathbf{x}_k)$ 稱爲梯度向量 (Gradient Vector), 而 $\bigtriangledown^2 f(\mathbf{x})$ 爲一 $n \times n$ 的對稱性矩陣, 一般稱爲 Hessian matrix, 在此並假設其反矩陣存在, 其定義如下

$$\bigtriangledown^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_n} \end{bmatrix} \tag{4}$$

由於 Hessian matrix 牽涉到非線性函數的二次微分, 在某些情況下並不容易推導或因反矩陣的計算上有困難 (或不存在), 因此出現了許多「改良」型的方式, 試圖避開這些問題。其中最簡單, 卻也非常有效的一個替代方案: 朝梯度向量的反方向前進, 即

$$\mathbf{d}_k = - \bigtriangledown f(\mathbf{x}) \tag{5}$$

一般稱爲 Steepest Descent direction。由於比較簡單, 通常也做爲一開始嘗試的選擇。不過當函數的本身或最小值附近呈現較平滑的趨勢時, 這個方式將出現收斂緩慢的情況。

2 練習

在進行以迭代法計算多變量函數最小值之前, 如果能將函數圖形畫出來, 將有助於起始值的選擇, 選取適當的迭代方向。以下將先練習三維函數圖形的繪製。

範例 **1.** 繪製下列函數的圖形

$$f(x,y) = xe^{-x^2-y^2}$$

繪製立體函數圖形的觀念如平面圖的作法:

1. 首先必須先訂出 $X-Y$ 平面的範圍與格子 (grids), 其 MATLAB 指令如下

   [X,Y]=meshgrid(變數 X 的範圍, 變數 Y 的範圍)

   矩陣 X 與 Y 代表 $X-Y$ 平面上格子點上的 X 與 Y 座標。關於 meshgrid 的詳細用法請自行 help meshgrid。圖 2是執行 [X,Y]=meshgrid(-2:2,-1:3) 的結果。如果 X 與 Y 的座標範圍一樣, 指令中只需輸入一組範圍即可。

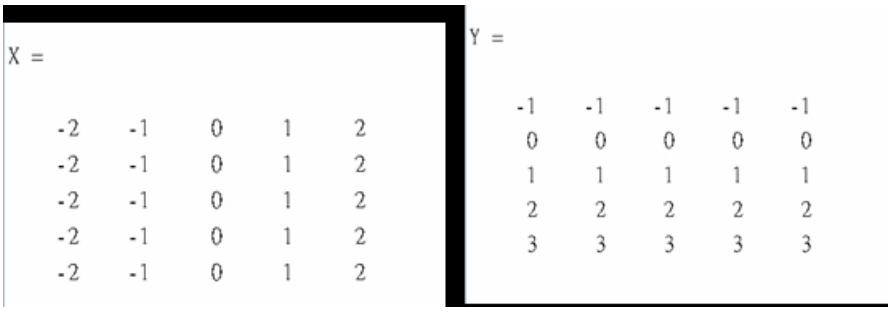
   

   圖 2: meshgrid 的概念

2. 利用 X 與 Y 計算函數在平面上每一個格子點上的函數值 Z。在 help meshgrid 裡面可以看到相關的用法。圖 3展示以 meshgrid 的概念繪製函數 $z=$

$f(x,y) = xe^{-x^2-y^2}$ 的立體圖。圖中因爲格子間隔較寬的原因, 所以可以清楚的看出 $X-Y$ 平面的格子點狀及其對應的 Z 值高度。指令如

```
Z = X. * exp(-X.^2 - Y.^2);
mesh(X,Y,Z)
```

3. 不妨試著從 meshgrid 指令調整格子的寬度, 看看效果如何。

4. 繪製函數圖。立體函數圖的指令很多, 如 mesh, surf, contour(等高線圖)等。

5. 利用 Matlab 的圖形旋轉功能 (在圖形工具列選擇「Rotate 3D」), 以滑鼠適度的旋轉圖形的角度, 找出一個容易觀察極值的位置, 並藉此調整繪圖的範圍。

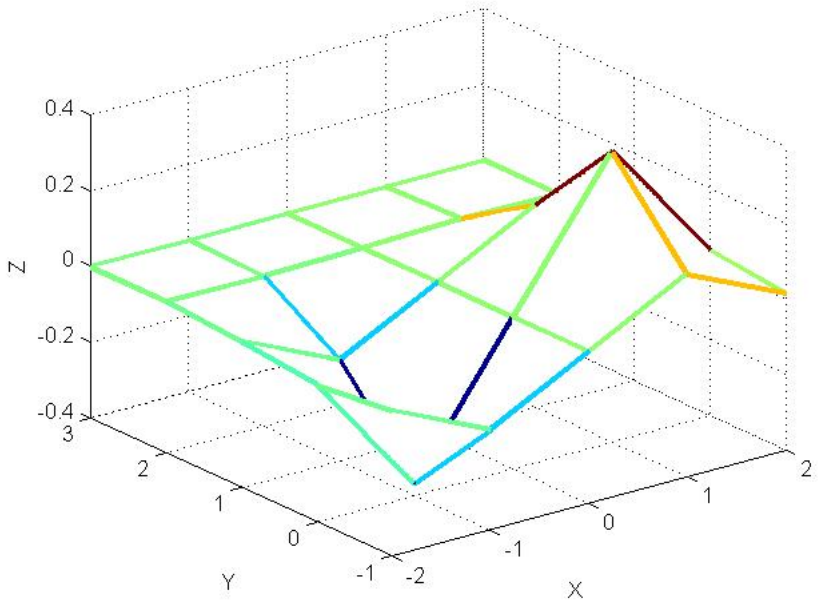
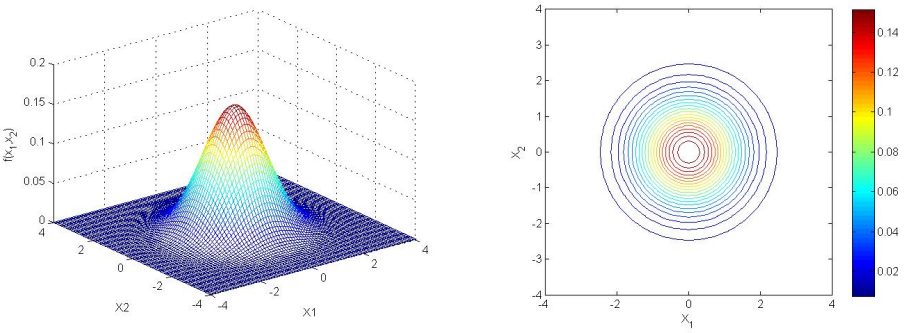


圖 3: meshgrid 的概念與立體圖

函數圖如圖 4所示。圖 4(b) 的等高線圖特別加了一條顏色帶 (colorbar), 表示高度或大小, 在圖形的工作列上可以找到這個功能, 或是在 contour 指令後面加上 colorbar 的指令。另外, 等高線圖是個平面圖, 有時會因爲 XY 軸的不均衡, 造成視覺上的落差, 譬如, 應該看起來是正圓形的圖, 卻變成橢圓。譬如圖 4, 這是個 XY 軸 (x_1 vs. x_2) 對稱的圖形, 爲彰顯此對稱性, 可以在圖形產生後加上以下的指令

```
axis square
```



(a) (b)

圖 4: (a) 函數的立體圖 (mesh)(b) 等高線圖 (contour)

製作立體圖也可以採迴圈的方式選取 $X - Y$ 平面上如棋盤式的 (x, y) 座標, 配合匿名函數, 逐點計算函數值, 如下列程式所示。其中第7行是函數值的計算, 並將結果放入預先設定記憶體的 Z 矩陣中。此處列向量代表 Y 的方向, 行向量代表 X 的方向, 配合指令 mesh 才能呈現正確的函數圖形。

```
f = @(x,y) x*exp(-x^2-y^2);
x=-3:0.1:3;
y=-4:0.1:4;
Z=zeros(length(y),length(x));
for i=1:length(x)
        for j=1:length(y)
                Z(j,i)=f(x(i),y(j));
        end
end
mesh(x,y,Z)
xlabel('X'),ylabel('Y')
```

範例 **2.** 繪製下列函數的圖形

$$f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2)^2 x_2^2 + (x_2 + 1)^2 \tag{6}$$

圖形如圖1所示。習者可以自行翻轉圖形找到最佳觀察的角度, 也可以多利用其他繪製立體圖的指令 (譬如,contour), 看看是否有更理想的視覺效果。使用 contour 指令時, 可以加入第四個參數, 決定多少條等高線, 如

```
contour(X, Y, Z, 100)
```

多變量函數的圖形僅限於兩個變量, 因此能在實務上發揮的機會不多。不過計算多變量的函數的極值, 倒是不受限制。只是當變數愈多時, 所花的時間愈長, 也愈容易陷入區域極值。以下先舉例說明 MATLAB 現成的處理多變量函數極值的指令, 之後再說明如何自己撰寫演算法。

範例 **3.** 利用 *MATLAB* 計算多變量函數的最小值的指令 *fminsearch* 計算

$$\min_{x_1,x_2} \ (x_1 - 2)^4 + (x_1 - 2)^2 x_2^2 + (x_2 + 1)^2$$

MATLAB 計算多變量函數最小值的方式, 依問題分爲兩類: 限制式 (constrained) 及非限制式 (unconstrained)。一般而言限制式比較麻煩, 也通常以非限制式做爲基礎, 容後討論。這裡先討輪 MATLAB 處理非限制式函數的指令 fminsearch, 方式如

```
f = @(x) (x(1) - 2)^4 + (x(1) - 2)^2 * x(2)^2 + (x(2) + 1)^2;
[x, fval] = fminsearch(f, [0, 0]);
```

MATLAB 處理函數的計算都半使用匿名函數的方式, 複雜些的函數甚至可以使用副程式 (function)。多變量函數的匿名函數, 其變數以向量方式表示。fminsearch 的第二個參數是初始值的設定。當函數的「長相」崎嶇不平時, 不同初始值的設定可能導致不同的答案與計算的時間長短, 特別當變數愈多時。所以使用者必須小心使用這個指令, 每次都試著給不同的初始值, 檢視結果, 就知道這個函數容不容易對付了。

除了初始值的選擇可能影響結果外, 設定 fminsearch 搜尋極值的停止條件有時候也很關鍵。下列程式展示 MATLAB 提供使用者介入指令功能的設定。

```
opts=optimset('fminsearch');
opts.Display='iter';
opts.MaxFunEvals=8000;
opts.MaxIter=8000;
opts.TolFun=1e-6;
opts.TolX=1e-6;
f = @(x) (x(1) - 2)^4 + (x(1) - 2)^2 * x(2)^2 + (x(2) + 1)^2;
[x, fval] = fminsearch(f, [0, 0], opts);
```

演算法涉及停止條件的設定,MATLAB 利用 optimset 指令給使用者一些自主的空間, 依據函數的特性或工作需求, 訂定演算法停止點。MATLAB 訂定的停止條件可以借第一行的結果得知。opts 是一個結構型變數, 從命令視窗輸入 opts 便可以知道有多少可以設定的參數。比較關鍵的幾個參數如第 2 到 6 行。

DIsplay 選項改爲 iter, 可以在執行時看到每個迴圈的部份結果, 這方面提供使用者了解這個函數的特性, 特別是接近極值時的平滑程度, 藉以設定下面幾個參

數。MaxFunEvals 決定做最大的函數計算次數, 有些函數非常複雜龐大, 計算時間長, 因時間或成本考量, 必須限制計算次數, 不管是否接近極值, 都要喊停。MATLAB 預設爲 200*變數個數。使用者有必要觀察演算法停止在哪個條件上, 才能適度的修改參數值。MaxIter 則是演算法的迴圈數, 預設值也是 200*變數個數。TolFun 是函數值的變化量, 當函數值在演算的過程中幾乎不再改變, 這是還停的時機,MATLAB 預設這個變化量爲 1e-4 ,[2]這裡下修爲 1e-6。另外從變數變化的程度也可以喊停, TolX 預設爲 1e-4, 這裡下修爲 1e-6。

不當的設定或完全不理會 MATLAB 的預設值, 有時會得到不正確的結果, 通常是還沒到達極值便停止計算。使用者必須謹愼小心。MATLAB 在 optimization toolbox 裡提供了另一個指令, fminuncon, 同樣用來計算非限制式多變量函數的極值。這個指令用了不同的演算法, 並允許使用者提供函數的偏微分函數, 採用梯度向量在演算法裡。

此外, 限制式極值的問題也很常見。在 MATLAB 官網的 file exchange 裡, 可以下載到從 fminsearch 改編的 fminsearchcon 指令, 用來處理限制式極小值問題, 非常出色。網址: http://www.mathworks.com/matlabcentral/fileexchange/8277-fminsearchbnd-fminsearchcon。以下的範例說明它的使用。另一個在 optimization toolbox 處理限制式多變量函數的極值是 fmincon。

範例 **4.** 利用 *John D'Errico* 爲 *fminsearch* 撰寫的延伸版, *fminsearchcon,* 計算下列限制式多變量函數的最小值。

$$\min_{x_1,x_2\in\Omega} (x_1-2)^4+(x_1-2)^2x_2^2+(x_2+1)^2$$

其中 Ω 爲

1. $\Omega=\{x_1,x_2\in R|\ x_1\geq 0, x_2\geq 0\}$
2. $\Omega=\{x_1,x_2\in R|\ x_1\leq 1, x_2\leq 2\}$
3. $\Omega=\{x_1,x_2\in R|\ 0\leq x_1\leq 1, 0\leq x_2\leq 2\}$

[2]在 MATLAB ,1e-4 代表 10^{-4}。

4. $\Omega = \{x_1, x_2 \in R | \ 0 \leq x_1 \leq \infty, -\infty \leq x_2 \leq 2\}$

5. $\Omega = \{x_1, x_2 \in R | \ x_1 + x_2 \leq 0.9\}$

6. $\Omega = \{x_1, x_2 \in R | \ 1.5 \leq x_1 + x_2 \leq 2\}$

7. $\Omega = \{x_1, x_2 \in R | \ \sqrt{x_1^2 + x_2^2} \leq 1\}$

8. $\Omega = \{x_1, x_2 \in R | \ \sqrt{x_1^2 + x_2^2} \leq 1, x_1 x_2 \geq 0\}$

9. $\Omega = \{x_1, x_2 \in R | \ x_1 + x_2 = 0.9\}$

限制式條件就是對未知變數的限制, 在實際應用上經常碰到。譬如變數值必須爲正, 或是代表機率值必須在 0 與 1 之間。這個範例蒐集了一些典型的限制式條件, 也是 fminsearchcon 指令要解決的。習者宜打開 fminsearchcon 程式, 好好研究如何置入這些限制式條件在指令中。典型的指令如

```
[x, fval]=fminsearchcon(fun, x0, LB, UB, A, b, nonlcon, options);
```

其中 fun 及 x_0 代表目標函數與初始值, LB, UB 是向量, 代表變數的下限與上限, 即 $LB \leq \mathbf{x} \leq UB$,$A, b$ 代表變數的線性不等式, 即 $A\mathbf{x} \leq b$, A 矩陣, b 爲向量。nonlcon 代表變數的非線性關係函數 $C(\mathbf{x}) \leq 0$, 譬如 $\sqrt{x_1^2 + x_2^2} \leq 1$, MATLAB 寫成 $@(x)norm(x) - 1$, 或是先寫成 $g = @(x)norm(x) - 1$, 再以 g 代表 nonlcon。當非線性限制條件超過一個時,$C(\mathbf{x})$ 代表一個向量函數, 如範例中的兩個非線性限制條件, 寫成 $g = @(x)[norm(x) - 1; -x(1) * x(2)]$
MATLAB 的 optimization toolbox 裡面的 fmincon, 在參數的安排與設定上有點不同, 習者應參照使用說明的指示, 計算本範例, 看看與 fminsearchcon 有何不同。下一個範例將 fminsearchcon 應用在實際的問題, 函數複雜許多, 變數也比較多, 很有挑戰性。

範例 **5.** 圖 *5* 是 *1000* 筆資料的直方圖。假設這些資料來自兩個 β 分配的組合, 即分配函數爲

$$f(x|\Omega) = \pi_1\beta(x|a_1, b_1) + \pi_2\beta(x|a_2, b_2)$$

其中 $\Omega = \{\pi_1, \pi_2, a_1, b_1, a_2, b_2\}, \pi_1 + \pi_2 = 1$。請試著採取最大概似參數估計法 *(MLE)* 來估計參數 Ω 。

寫出概似函數

$$L(\Omega) = \sum_{i=1}^{1000} \ln(\pi_1\beta(x_i|a_1, b_1) + \pi_2\beta(x_i|a_2, b_2))$$

問題變成

$$\max_{\Omega=\{\pi_1,\pi_2,a_1,b_1,a_2,b_2\},\pi_1+\pi_2=1,\pi_1>0,\pi_2>0} L(\Omega)$$

這是一個限制式的極值問題, 限制條件在兩個參數 $\pi_1 + \pi_2 = 1$ 及 $\pi_1, \pi_2 > 0$。這個限制式可以被修改爲 $0 < \pi_1 < 1$。其中 π_2 直接以 $1 - \pi_1$ 取代, 變數減爲 5 個。程式片段如下 (假設樣本資料已經存放在變數 x)

```
opts=optimset('fminsearch');
opts.Display='iter';
opts.MaxFunEvals=8000;
opts.MaxIter=8000;
opts.TolFun=1e-6;
opts.TolX=1e-6;
initial=[0.5 1 10 5 5];
LB=[0 0 0 0 0];
UB=[1 Inf Inf Inf Inf];
L = @(p) - sum(log(p(1) * betapdf(x,p(2),p(3)) + (1 - p(1)) *
betapdf(x,p(4),p(5))));
[mle,fval] = fminsearchcon(L,initial,LB,UB,[ ],[ ],[ ],opts);
```

結果如何呢? 習者不妨自己模擬資料, 看看估計的結果與眞實參數值相差多少。也可以試著將直方圖與估計出來的函數圖畫在一起, 看看估計的結果是否貼近資料的分佈。

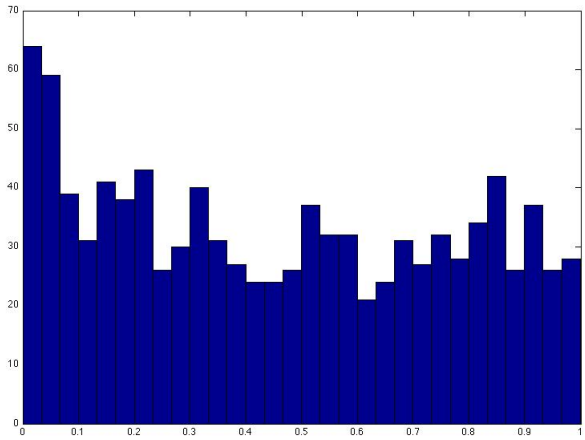


圖 5: 來自混合 β 分配樣本的直方圖

範例 **6.** 推導函數 *(6)* 的梯度向量 $\bigtriangledown f(\mathbf{x})$ 及 *Hessian Matrix* $\bigtriangledown^2 f(\mathbf{x})$

以演算法計算極值, 通常需要紙筆推導到一定的程度, 才進行程式的寫作, 不是忙亂的一陣砍殺或是坐在哪兒發呆不知從何下手。因此在眞正應用演算法之前, 計算梯度向量及 Hessian Matrix 是必要的。

範例 **7.** 分別利用 *Steepest Descent Method(5)* 及 *Newton-Raphson Method(3),* 寫程式計算函數 *(6)* 的最小值。

運用演算法前, 最好先將演算法的步驟寫下來 (作業1), 再利用紙筆推導函數的一次及二次導數, 最後才進行程式的寫作。程式部分可以參考之前關於計算函數極值的單元, 雖然是單一變數的程式, 但其邏輯卻是一致的, 可以直接套用, 再來修改。

圖 6 寫出 steepest descent 演算法的過程。當比較簡單的 steepest descent 演算法成功後, 牛頓法便可以直接套用, 其差別只在第 15 行後面加入 Hessian matrix 的計算, 並在第 18 行修正 xk1 的計算。

圖 6 的程式執行時, 會列印出每個迴圈的結果 (見圖右上角), 包含函數值與 x 的位置。這個動作方便觀察程式是否出錯, 也可以感受不同演算法的執行效率。做法請參考第 12 行的 fprintf 指令, 這個指令類似 C 語言的 printf, 使用方式非常豐富, 圖上展示的方式分爲兩個部分, 即單引號及後面的變數部分。單引號裡面放置欲表達的輔助文字與列印變數內容的格式, 如 %10.7f, 意思是以浮點 (floating point) 的方式顯示變數內容, 其中小數點以下 7 位, 整數 2 位, 加上小數點也算一位, 共 10 位。引號後面, 依序擺上欲列印內容的變數名稱。請注意單引號最後面的 $\backslash n$ 代表跳行。

```
1  clear
2  [X,Y]=meshgrid(1:0.05:3,-2:0.05:0);
3  Z=(X-2).^4+((X-2).^2).*(Y.^2)+(Y+1).^2;
4  contour(X,Y,Z,20)
5  N=20;
6  beta=0.5.^(0:N-1);
7  f=@(x,y)  (x-2)^4+((x-2)^2).*(y^2)+(y+1)^2;
8  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9  xk=[2.8 -0.1]';%初始值
10 fk=f(xk(1),xk(2));
11 while 1
12    fprintf('函數值 f(x)=%10.7f   x=%7.5f  y=%7.5f\n', fk, xk(1),xk(2))%列印過程
13    text(xk(1),xk(2),'X') %追蹤過程
14    pause(1)
15    fp= -[4*(xk(1)-2)^3+2*(xk(1)-2)*xk(2)^2 ; 2*(xk(1)-2)^2*xk(2)+2*(xk(2)+1)];
16 %--- 步伐調整 ----
17    for j=1:N
18       xk1=xk +beta(j)*fp;% steepest descent
19       fk1=f(xk1(1),xk1(2));
20       if fk1 < fk; break;end
21    end
22    if abs(fk1-fk) <10^(-6),break,end
23    xk=xk1; %接受 xk1，重設定爲 xk 繼續下一次遞迴
24    fk=fk1;  %同上
25 end
```

```
Command Window
To get started, select MATLAB Help or Demos from the Help menu.
函數值 f(x)= 1.2260000   x=2.80000  y=-0.10000
函數值 f(x)= 0.0541480   x=1.76800  y=-0.93600
函數值 f(x)= 0.0506347   x=2.22446  y=-0.96324
函數值 f(x)= 0.0023924   x=1.99358  y=-0.95147
函數值 f(x)= 0.0023772   x=2.00520  y=-1.04845
函數值 f(x)= 0.0000003   x=1.99948  y=-0.99997
>>
```

圖 6: Steepest descent 演算法程式

圖6的程式還運用了空照圖, 畫出 steepest descent 演算法的移動過程, 如圖**??**。第 13 行爲每個 xk 的位置化下一個 X, 第 14行暫停 1 秒鐘, 分辨觀察。這些都是寫程式時可以做爲程式偵錯與欣賞演算法的小技巧。除此之外, 第 23,24行是初學者最容易疏忽的地方, 將新的位置 (xk1) 設定回舊的位置 (xk), 才能在演算法中產生遞迴的效應, 初學者不可不詳辨焉。

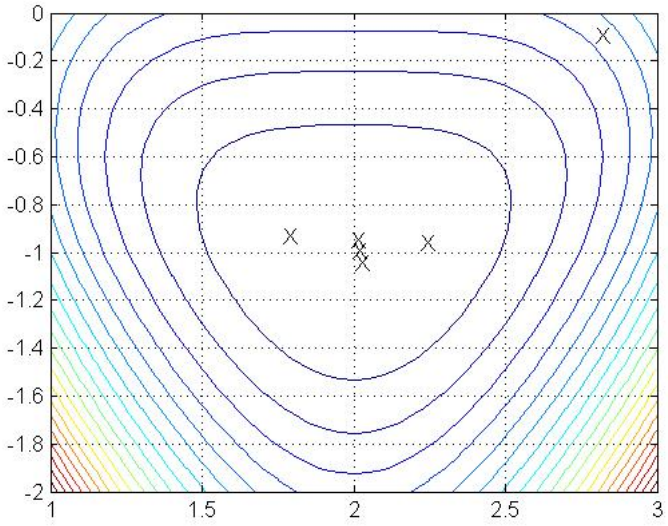


圖 7: Steepest descent 演算法程式:x 的移動過程

能力未敷之際, 切忌憑空築樓, 否則程式錯了不容易除錯。另外計算目標函數值時, 可以使用匿名函數的方式, 讓整個程式看起來更簡潔。由於 Newton-Raphson 牽涉到 Hessian Matrix 的二次導數, 有時候推導起來非常繁複。因此, 簡單的 steepest descent 法通常是演算法的首選。不過我們還是可以利用這個簡單的函數, 來看看兩者的收斂情況。

3 觀察

1. 不論是式 (3) 的 Steepest Descent 或式 (5) 的 Newton-Raphson 的移動

方向, 都不能保證可以帶領函數值往下探底, 因此所謂的「步伐調整」

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \beta \mathbf{d}_k$$

仍是必須的。請參考前章「演算法: 計算函數的極值」關於「步伐調整器」β 的選擇。

2. 關於 Newton-Raphson method 的合理性、限制, 請參考[1], 或從作業2中略探一二。

3. 仔細觀察不同的 $\mathbf{d}_k$(式 (3) 與式 (5)) 在程式進行過程的收斂速度。甚至可以記錄過程中的函數值 , 再繪製函數值隨著方向迭代的次數逐漸下降的趨勢。參考文獻[1]提供兩者收斂速度的理論分析。

4 作業

1. 請寫出適當的演算法, 用以計算函數的最小值。

2. Newton-Raphson method 的方向選擇, 如式(3), 可由函數的泰勒展開式得到驗證。多變量函數 $f(\mathbf{x}_{k+1})$ 的泰勒展開式寫爲

$$f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k + \mathbf{d}_k) = f(\mathbf{x}_k) + \bigtriangledown f(\mathbf{x})^T \mathbf{d}_k + \frac{1}{2} \mathbf{d}_k^T \bigtriangledown^2 f(\mathbf{x}) \mathbf{d}_k + O(||\mathbf{d}_k||^3)$$

假設 $||\mathbf{d}_k||$ 夠小, 可以省略 $O(||\mathbf{d}_k||^3)$, 證明式(3)的 Newton-Raphson direction 爲下列最小值問題的解

$$\min_{\mathbf{d}_k} f(\mathbf{x}_{k+1})$$

3. 著名的 Rosenbrock's Banana 測試函數

$$f(x, y) = 100(y - x^2)^2 + (1 - x)^2$$

常被用來測試極值演算法, 請試著繪製其立體圖 (mesh) 與空照圖 (contour), 再利用本單元介紹的兩個方法計算最小值, 並比較其收斂的情況。

4. 自行找一個兩個變數的函數, 繪製函數圖 (mesh 與 contour) 並計算最小值。

5. 比較不同的迭代方向式 (3) 與式 (5) 的收斂速度, 譬如到達同一個函數值所花的時間。此時可以觀察幾個不同函數值, 由大到小, 分別記錄兩種不同方式所花的時間, 最後畫出一張圖比較兩者收斂的情況; 橫軸代表「函數值」由大到小, 縱軸代表時間。注意: 除 $\mathbf{d}_k$ 不同外, 其他條件必須相同, 譬如起始點。

6. 估計下列 MLE 問題

$$\max_{\alpha,\beta} \log L(\alpha, \beta)$$

其中的概似函數爲

$$L(\alpha, \beta) = \prod_{i=1}^{n} f_t(v_i; \alpha, \beta) F_T(u_i; \alpha, \beta)^{-1}$$

where

$$f_t(v; \alpha, \beta) = \alpha\beta v^{\beta-1} exp(-\alpha v^{\beta})$$

$$F_T(u; \alpha, \beta) = 1 - exp(-\alpha u^{\beta})$$

變數 u, v 的 n 個樣本已知, 請至下列網址下載該樣本資料檔 UV.txt。
http://web.ntpu.edu.tw/∼ccw/statmath/book.htm ,「多變量函數的最小值」。

7. 匿名函數的使用可以非常簡單直接, 也可能拐彎抹角的有點複雜, 但如果能巧妙運用, 往往可以解決看起來極困難的問題。譬如以下的問題:
假設兩獨立變數 X, Y 分別服從 β 分配, 其 PDF 爲 $f_X(x) = \beta(x|a_1, b_1)$, $f_Y(x) = \beta(x|a_2, b_2)$。建立一新變數 Z

$$Z = XY$$

變數 Z 並不服從 β 分配, 但我們想以一 β分配, $\beta(a, b)$ 來近似 Z 的分配, 做法如下

$$\min_{a,b} \int_0^1 (f(z) - \beta(z|a,b))^2 \, dz$$

其中 $f(z)$ 爲變數 Z 的眞實 PDF, 經過推演後寫成

$$f(z) = \int_z^1 f_Y(y) f_X(z/y) \frac{1}{y} \, dy$$

其中 Beta 函數 $f_X(x), f_Y(x)$ 的參數分別指定爲 $a_1 = b_1 = a_2 = b_2 = 2$。

請估計上述極小值問題的參數 a, b。

Hint:

- 其中積分與最小值的部分可以採 quad 及 fminsearch 指令。
- Beta 函數則使用 betapdf(x,2,2)。
- 當匿名函數的方式且匿名函數比較複雜時, 可以搭配副程式 (function) 將複雜的部分拉到副程式去做。

8. 前面的範例應用了最大概似估計法, 估計了混合 β 分配的參數。另一個常見的問題是混合常態的參數估計。即,

$$\max_{\Omega=\{\pi_1,\pi_2,\mu_1,\sigma_1^2,\mu_2,\sigma_2^2\},\pi_1+\pi_2=1,\pi_1>0,\pi_2>0} L(\Omega)$$

其中概似函數爲

$$L(\Omega) = \sum_{i=1}^{N} \ln(\pi_1 f(x|\mu_1, \sigma_1^2) + \pi_2 f(x|\mu_2, \sigma_2^2))$$

$f(x|\mu, \sigma^2)$ 爲常態分配的機率密度函數。請自行產生適用的資料並運用 fminsearchcon 估計參數 $\Omega = \{\pi_1, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2\}$。

參考文獻

[1] J.E. Dennis, R.B. Schnabel, "Numerical Methods for Unconstrained Optimization," Prentice Hall.

```
[ab,hval]=fminsearch(@(a) quad(@(z)(fun_z(z)-betapdf(z,a(1),a(2))).^2,0,1),[5,5]);
```

```
function k=fun_z(z)
k=zeros(size(z));
for i=1:length(z)
    w=@(y)betapdf(y,2,2).*betapdf(z(i)./y,2,2)./y;
    k(i)=quad(w,z(i),1);
end
```