El método del descenso más pronunciado

Octavio Alberto Agustín Aquino Grupo 705

4 de junio de 2005

Índice

Índice de figuras

1. Optimización no lineal sin restricciones

Problema 1 *Dada una función diferenciable* $f : \mathbb{R}^2 \to \mathbb{R}$*, hallar* $x^* \in \mathbb{R}^2$ *de modo que existe una vecindad* N *de* x^* *tal que*

$$f(x^*) \leq f(x)$$

para todo $x \in N$*. Tal punto se llama mínimo local de* f*.*

Los algoritmos para resolver este problema, dada una aproximación x_0 al mínimo local, generan una sucesión de iteraciones $\{x_i\}_{i=0}^{k}$ que termina cuando no se puede mejorar la aproximación o se alcanza alguna precisión prestablecida. Para moverse de un punto x_k al siguiente x_{k+1} en la iteración, se utiliza la información de f en x_k (y ocasionalmente la de los valores previos $x_0, \ldots, x_k$).

Hay dos formas de lograr esta sucesión. La estrategia que analizaremos aquí brevemente es la de búsqueda de línea. En este caso, el algoritmo escoge una dirección p_k y busca a lo largo de esta dirección desde el punto actual por un nuevo punto que reduzca el valor de f. Se necesita, además, calcular la

distancia que recorrerse en esta dirección, lo cual se puede conseguir resolviendo el siguiente problema de minimización

$$\min_{\alpha>0} f(x_k + \alpha p_k).$$

Sin embargo, resolver este problema exactamente es muchas veces innecesario, y basta con aproximarse a una solución para calcular un nuevo punto de la sucesión.

1.1. Direcciones de búsqueda

La dirección de máximo descenso $-\nabla f_k$ es la elección más obvia para buscar el mínimo, pues es la dirección en la que f decrece más rápidamente. Este esquema nos conduce al método del descenso más pronunciado. En efecto, por el teorema de Taylor, tenemos que para cualquier dirección de búsqueda p y parámetro de tamaño de paso α, tenemos

$$f(x_k + \alpha p) = f(x_k) + \alpha p^T \nabla f_k + \frac{1}{2}\alpha^2 p^T \nabla^2 f(x_k + tp)\, p,$$

para algún $p \in (0, \alpha)$. La tasa de crecimiento de f a lo largo de p en x_k es el coeficiente de α, es decir, $p^T \nabla f_k$. Por lo tanto, la dirección unitaria p de máximo decrecimiento es la solución del problema

$$\min_{\|p\|=1} p^T \nabla f_k.$$

En virtud de que

$$p^T \nabla f_k = \|p\| \, \|\nabla f_k\| \cos\theta,$$

donde θ es ángulo entre p y ∇f_k, se infiere que el mínimo se alcanza cuando

$$p = -\frac{\nabla f_k}{\|\nabla f_k\|}.$$

Aunque con el método del descenso más pronunciado no necesita calcular segundas derivadas. De hecho, con cualquier dirección de descenso producirá un decremento en f, siempre que el tamaño de paso sea suficientemente pequeño. Para ver esto, acudimos al teorema de Taylor,

$$f(x_k + \epsilon p_k) = f(x_k) + \epsilon p_k^T \nabla f_k + O\left(\epsilon^2\right).$$

Si p_k es una dirección de descenso, el coseno del ángulo entre p_k y ∇f_k es negativo, luego $\epsilon p_k^T \nabla f_k < 0$, así que

$$f(x_k + \epsilon p_k) < f(x_k).$$

1.2. Cálculo del tamaño de paso

Tomar simplemente como dirección de descenso al vector unitario en dirección contraria al gradiente no es una buena elección pues puede no converger. En efecto, para una función tan simple como $f(x) = x^2 + y^2 + 1$, el gradiente $\nabla f = (2x, 2y)$ evaluado en $(0{,}5, 0)$ vale $(1, 0)$. Al movernos en la dirección opuesta a la este vector (es decir, buscando el mayor descenso), llegamos al punto $(-0{,}5, 0)$. Repitiendo el proceso de calcular el gradiente y moverse en dirección opuesta, nos mantendremos oscilando infinitamente entre $(0{,}5, 0)$ y $(-0{,}5, 0)$. Esto puede evitarse escogiendo un tamaño de paso para avanzar en la dirección opuesta a la del gradiente. Para calcular dicho tamaño de paso, utilizamos las condición de decrecimiento suficiente

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c\alpha_k \nabla f_k^T p_k, \tag{1}$$

donde $c_1 \in (0, 1)$ es una constante apropiada. Intuitivamente, que se cumpla esta desigualdad significa que la función

$$\phi(\alpha) = f(x + \alpha p)$$

está por debajo de la línea

$$l(\alpha) = f(x) + c\alpha \nabla f^T p,$$

la constante c se escoge generalmente como 1×10^{-4}.

Podemos utilizar el siguiente algoritmo para encontrar el tamaño de paso en el método del descenso más pronunciado.

Algoritmo 2 (Búsqueda de línea por rastreo hacia atrás) *Dados $\overline{\alpha} > 0$, ρ, $c \in (0, 1)$, devuelve α, un tamaño de paso para el método de descenso más pronunciado.*

1. *Hacer $\alpha \leftarrow \overline{\alpha}$*
2. *Repetir 3 mientras $f(x_k + \alpha p_k) > f(x_k) + c\alpha \nabla f_k^T p_k$.*
3. *Hacer $\alpha \leftarrow \rho\alpha$.*
4. *Devolver $\alpha_k = \alpha$.*

Éste algoritmo termina porque α se vuelve suficientemente pequeño como para satisfacer (1). El factor de contracción ρ puede variar en cada paso de la iteración del método de máximo descenso, pero siempre se halla en el intervalo $(0, 1)$.

Así, el algoritmo del método del descenso más pronunciado queda de la siguiente manera.

Algoritmo 3 (Descenso más pronunciado) *Dada f, x_0 aproximación inicial al mínimo local de f, n el máximo número de iteraciones y tol la tolerancia.*

1. *Hacer* $k = 1$.
2. *Hacer* $p_k \leftarrow -\frac{\nabla f_k}{\|\nabla f_k\|}$.
3. *Calcular* α_k *de acuerdo al algoritmo 2.*
4. *Hacer* $x_{k+1} = x_k + \alpha_k p_k$
5. *Repetir de 2 a 4 mientras* $|x_{k+1} - x_k| > tol$ *y* $k \leq n$.
6. *Devolver* x_k

En particular, vamos a aplicar el algoritmo 3 a la función de Rosenbrock

$$f(x_1, x_2) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2, \tag{2}$$

implementando particularmente para esta función los algoritmos en Octave 2.1.50 (http://www.octave.org).

1.3. Implementación

Como vamos a aplicar el método del descenso más pronunciado a la función de Rosenbrock, la graficaremos con ayuda de Octave.

```
function GraficaRosenbrock(a,b,n)
 x=[a:(b-a)/n:b];
 y=[a:(b-a)/n:b];
 for i=1:n+1
 for j=1:n+1
 ros(i,j)=100*(y(i)-x(i)^2)^2+(1-x(i))^2;
 endfor
 endfor
 mesh (x,y,ros);
endfunction
```

El resultado es el siguiente para `GraficaRosenbrock(-10,10,100)`.

Ahora reproducimos el código que calcula para (2) la iteración de descenso más pronunciado de acuerdo a los algoritmos 2 y 3, realizando este último un número de `maxit` de veces. No se toma en cuenta la tolerancia, porque el algoritmo converge con mucha lentitud para la función (2).

```
function alpha=busqueda(a,rho,c,u)
 gradiente=[-400*(u(2)-(u(1))^2)*u(1)-2*(1-u(1)) 200*(u(2)-(u(1))^2)];
 p=-gradiente/norm(gradiente);
 alpha=a;
 x=(u+alpha*p);
 f=100*(x(2)-x(1)^2)^2+(1-x(1))^2;
 x=u;
 g=100*(x(2)-x(1)^2)^2+(1-x(1))^2;
```

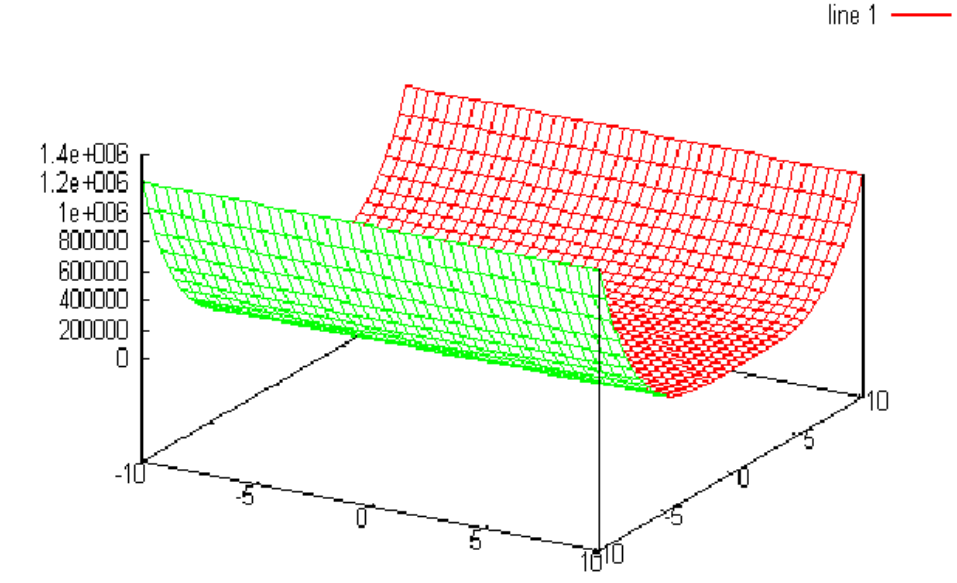

Figura 1: Función de Rosenbrock.

```
 while f>g+(c*alpha)*gradiente*p'
  alpha=alpha*rho;
  x=(u+alpha*p);
  f=100*(x(2)-x(1)^2)^2+(1-x(1))^2;
 endwhile
endfunction

function x=Descenso(x,maxit)
 clearplot;
 hold on;
 u=[];
 y=[];
 for i=0:maxit
  alpha=busqueda(1,0.5,0.0001,x);
  p=[-400*(x(2)-(x(1))^2)*x(1)-2*(1-x(1)) 200*(x(2)-(x(1))^2)];
  p=p/norm(p);
  x=x-(alpha*p);
  u=[u x(1)];
```

```
  y=[y x(2)];
 endfor
 plot(u,y);
 hold off
endfunction
```

La salida es la siguiente para `x=Descenso([0 0],1000)`.

```
x = 0.9054 0.8153
```

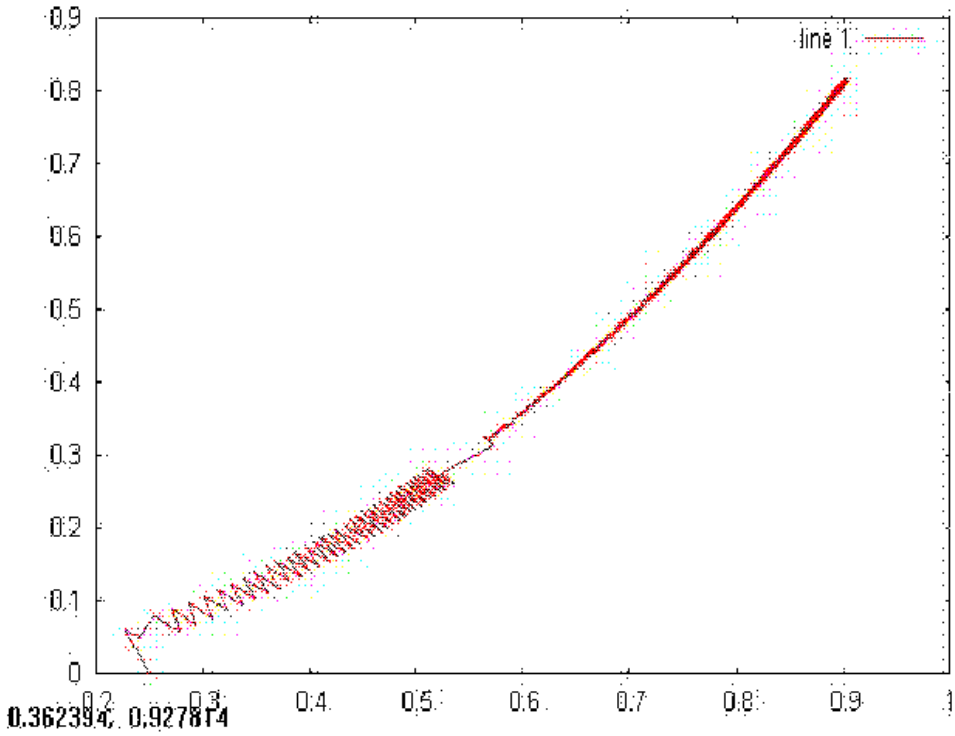

Figura 2: Convergencia del método del descenso más pronunciado para la función de Rosenbrock.

La gráfica representa el avance del punto $x_0 = (0, 0)$. Obsérvese que la convergencia se hace más lenta conforme se acerca al óptimo $(1, 1)$.

También se realizó una implementación genérica del método, aproximando el gradiente con la fórmula

$$\nabla f(x) = \frac{1}{\delta} \begin{pmatrix} f(x + \delta e_1) - f(x) \\ f(x + \delta e_2) - f(x) \end{pmatrix},$$

donde $\{e_1, e_2\}$ es la base canónica de $\mathbb{R}^2$.

```
function gra=gradiente(expr,u,delta)
 x=u(1)+delta;
 y=u(2);
 w1=eval(expr);
 x=u(1);
 y=u(2)+delta;
 w2=eval(expr);
 gra=[w1 w2];
 x=u(1);
 y=u(2);
 w1=eval(expr);
 gra=(gra-[w1 w1])/delta;
endfunction

function alpha=busqueda2(expr,delta,a,rho,c,u)
 x=u(1)+delta;
 y=u(2);
 w1=eval(expr);
 x=u(1);
 y=u(2)+delta;
 w2=eval(expr);
 gra=[w1 w2];
 x=u(1);
 y=u(2);
 w1=eval(expr);
 gra=(gra-[w1 w1])/delta;
 p=-gra/norm(gra);
 alpha=a;
 x=(u(1)+alpha*p(1));
 y=(u(2)+alpha*p(2));
 f=eval(expr);
 x=u(1);
 y=u(2);
 g=eval(expr);
 while f>g+(c*alpha)*gra*p'
  alpha=alpha*rho;
  x=(u(1)+alpha*p(1));
  y=(u(2)+alpha*p(2));
  f=eval(expr);
 endwhile
endfunction

function x=Descenso2(expr,x,delta,maxit)
 clearplot;
 hold on;
 u=[];
```

```
 y=[];
 for i=0:maxit
  alpha=busqueda2(expr,delta,1,0.5,0.0001,x);
  p=gradiente(expr,x,delta);
  p=p/norm(p);
  x=x-(alpha*p);
  u=[u x(1)];
  y=[y x(2)];
 endfor
 plot(u,y);
 hold off
endfunction
```

La salida para `x=Descenso2(''100*(y-x^2)^2+(1-x)^2)'',[0 0],0.0001,1000)` es:

```
x = 0.97095 0.94263
```

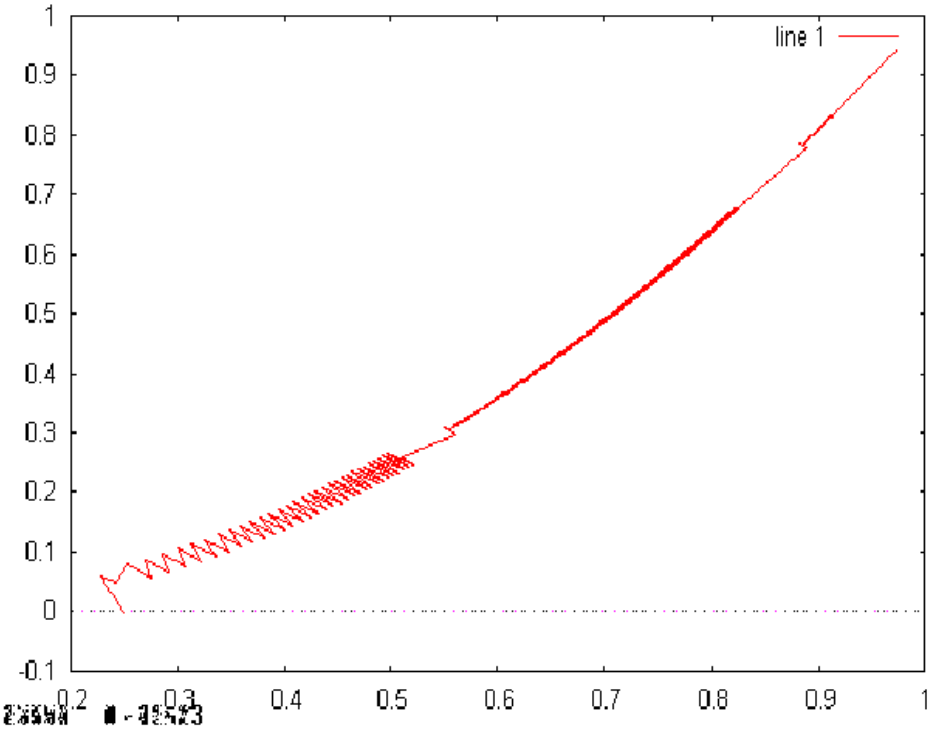

Figura 3: Máximo descenso con aproximación del gradiente.

1.4. Conclusiones

Si bien el método del descenso más pronunciado surge de manera muy natural y es muy intuitivo, aún escogiendo el tamaño de paso adecuado puede ser muy lento, como puede verse en la figura **??**.

Por otra parte, el mismo método ilustra una técnica muy importante de la optimización no lineal: la búsqueda por línea. Además, dentro de ésta estrategia, pudimos estimar tamaños de paso apropiados para asegurar la convergencia, la cual no puede garantizarse siguiendo una implementación inmediata del descenso más pronunciado.

Cabe observar que el desempeño de la segunda implementación del algoritmo y la primera son distintas. La segunda resultó mejor porque la función es casi plana alrededor de su mínimo global $(1, 1)$, y es por ello que la aproximación linea al gradiente es superior al valor exacto, en lo referente a escoger la ruta hacia el mínimo.

Referencias

[1] Nocedal, Jorge *et al. Numerical optimization*, Springer-Verlag, Nueva York, 1999.