Optimización sin restricciones

Prof. Cesar de Prada
Dpt. Ingeniería de Sistemas y Automática
UVA
prada@autom.uva.es

Optimización sin restricciones

$$\min_x J(x)$$
$$x \in R^n$$

Los métodos sin restricciones son importantes porque:

- ✓Hay problemas que se pueden formular sin restricciones
- ✓Permiten introducir muchos conceptos y explorar ideas que se usarán en problemas NLP
- ✓Muchos problemas de optimización utilizan en alguna fase algoritmos sin restricciones
- ✓Algunos problemas NLP pueden reformularse como problemas sin restricciones

Indice

- Condiciones de extremo
- Problemas monovariables
  - Métodos tipo Newton
  - Métodos de reducción de intervalos
  - Métodos de aproximación por polinomios
- Problemas multivariables
  - Algoritmos basados en gradientes
  - Algoritmos tipo Newton
  - Algoritmos basados solo en valores de la función
- Software

Existen muchos métodos. En el curso solo veremos algunos de los mas significativos

Condiciones de extremo

$$\min_x J(x)$$
$$x \in R^n$$

En problemas sin restricciones es posible encontrar condiciones analíticas de óptimo

Condición necesaria

$$\left.\frac{\partial J(x)}{\partial x}\right|_{x^*} = 0$$

El hessiano H, o matriz de derivadas segundas determina el carácter del posible óptimo

$$H = \left.\frac{\partial^2 J(x)}{\partial x^2}\right|_{x^*}$$

Condiciones de extremo

$$J(x) = J(x^*) + \left.\frac{\partial J}{\partial x}\right|_{x^*}' (x - x^*) + \frac{1}{2}(x - x^*)' \left.\frac{\partial^2 J(x)}{\partial x^2}\right|_{x^*} (x - x^*) + ...$$

$$\left.\frac{\partial J}{\partial x}\right|_{x^*} = 0$$ x^* cumpliendo esta ecuación se denomina punto estacionario

$$J(x) - J(x^*) \approx \frac{1}{2}(x - x^*)' \left.\frac{\partial^2 J(x)}{\partial x^2}\right|_{x^*} (x - x^*)$$ Aproximación de 2º orden

Si H(x*) es PD o PSD, J(x) presenta un mínimo en x^*

Si H(x*) es ND o NSD, J(x) presenta un máximo en x*

Si H(x*) es indefinida no hay extremo, J(x) presenta un punto de silla en x^*

Condiciones de extremo

$$\min_x J(x)$$
$$x \in R^n$$

$$\left.\frac{\partial J(x)}{\partial x}\right|_{x^*} = 0$$

La solución analítica de la condición de extremo, para funciones complejas, suele ser una ecuación no lineal difícil de resolver, por lo que suele ser preferible formular métodos numéricos directos para resolver el problema

Optimización monovariable

$$\min_x J(x) \qquad x \in R$$

Son problemas importantes porque:

✓Se usan como un paso intermedio en otros algoritmos

✓Muchos problemas son monovariables

Existen varios métodos basados en enfoques diferentes:

a) Imponer las condiciones analíticas

b) Minimizar un intervalo de incertidumbre

c) Aproximar la función por un polinomio

d) otros

Supondremos que las funciones J(x) son unimodales y solo presentan un mínimo local

Método de Newton (Newton-Raphson)

$$\min_x J(x) \qquad x \in R$$

La solución analítica es J'(x)=0

¿cómo resolver esta ecuación?

Aplicando el método de Newton. Para la solución de f(z) = 0:

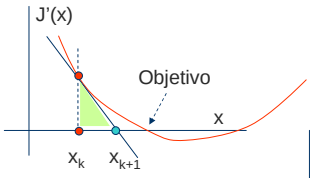
$$z_{k+1} = z_k - \frac{f(z_k)}{f'(z_k)}$$

Que conduce a:

$$x_{k+1} = x_k - \frac{J'(x_k)}{J''(x_k)}$$

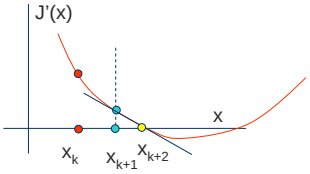
Se parte de un valor inicial x_0 y se generan aproximaciones x_1, x_2, ... hasta que se cumpla un criterio de finalización

Interpretación geométrica



$$J'(x_k) = (x_{k+1} - x_k) J''(x_k)$$

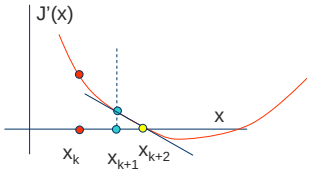
$$x_{k+1} = x_k - \frac{J'(x_k)}{J''(x_k)}$$



Cada iteración corresponde a una aproximación lineal

Criterios de finalización

$$x_{k+1} = x_k - \frac{J'(x_k)}{J''(x_k)}$$



Criterios de finalización:

$| J'(x_k) | < \varepsilon$

$| x_{k+1} - x_k | < \varepsilon$

$| J(x_{k+1}) - J(x_k) | < \varepsilon$

$k > N$

La principal dificultad y empleo de tiempo está en el cálculo de las derivadas

Derivadas

- Si las derivadas no son calculables analíticamente, pueden aproximarse por:

$$J'(x_k) = \frac{J(x_k + \sigma) - J(x_k - \sigma)}{2\sigma}$$

σ >0 pequeño

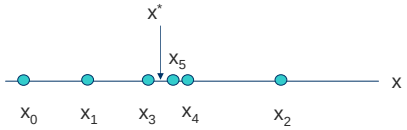
$$J''(x_k) = \frac{J(x_k + \sigma) - 2J(x_k) + J(x_k - \sigma)}{\sigma^2}$$

Por tanto:

$$x_{k+1} = x_k - \frac{J(x_k + \sigma) - J(x_k - \sigma)}{J(x_k + \sigma) - 2J(x_k) + J(x_k - \sigma)} \frac{\sigma^2}{2\sigma}$$

Convergencia

$$x_{k+1} = x_k - \frac{J'(x_k)}{J''(x_k)}$$



Diremos que la solución converge a un valor x^* cuando la secuencia de valores x_k generados por el algoritmo verifican

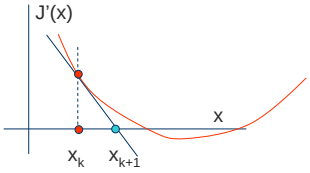
$$\left\| x_{k+1} - x^* \right\| \le c \left\| x_k - x^* \right\|$$

$$0 < c < 1$$

A partir de un k. De modo que los puntos x_k están cada vez mas próximos a x^*

Método de Newton

$$x_{k+1} = x_k - \frac{J'(x_k)}{J''(x_k)}$$



Ventajas:

Converge localmente en forma cuadrática

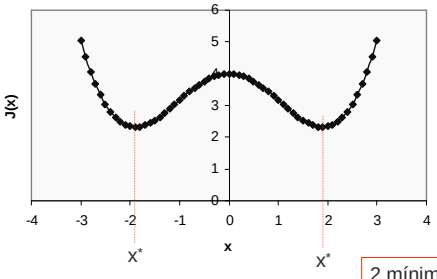
Inconvenientes:

Hay que calcular o estimar primeras y segundas derivadas

Si J''(x) $\rightarrow$0 converge lentamente

Si x_0 esta muy alejado de x^* puede no converger, o converger a otro valor

Ejemplo J(x) = x^2 + 4 cos(x)



Minimizar J(x)

J'(x^*) =

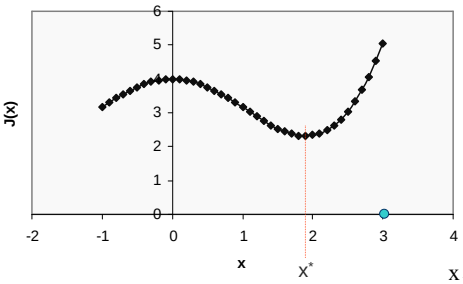
= 2x^* - 4sen(x^*) = 0

Necesita resolverse por métodos numéricos

x^* = ± 1.895.. , 0

2 mínimos (locales, globales) y un máximo

Minimizar J(x) = x^2 + 4 cos(x)



Minimizar J(x) usando Newton

J'(x) = 2x-4sen(x)

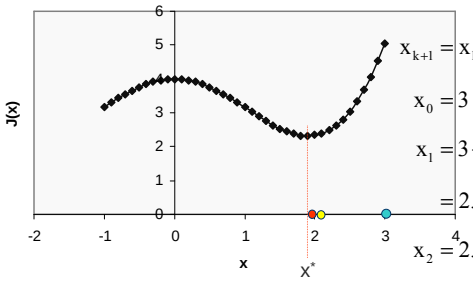
J''(x) = 2 – 4cos(x)

$$x_{k+1} = x_k - \frac{J'(x_k)}{J''(x_k)}$$

$$x_{k+1} = x_k - \frac{x_k - 2\mathrm{sen}(x_k)}{1 - 2\cos(x_k)}$$

$x_0 = 3$

Minimizar J(x) = x^2 + 4 cos(x)



$$x_{k+1} = x_k - \frac{J'(x_k)}{J''(x_k)}$$

$$x_{k+1} = x_k - \frac{x_k - 2\mathrm{sen}(x_k)}{1 - 2\cos(x_k)}$$

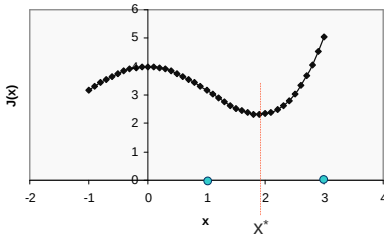
$x_0 = 3$

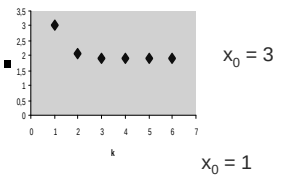
$$x_1 = 3 - \frac{3 - 2\mathrm{sen}(3)}{1 - 2\cos(3)} = 2.088$$

$$x_2 = 2.088 - \frac{2.088 - 2\mathrm{sen}(2.088)}{1 - 2\cos(2.088)} = 1.91$$

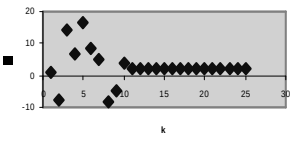
x_3 = ...

Ejemplo J(x) = x^2 + 4 cos(x)





$x_0 = 3$



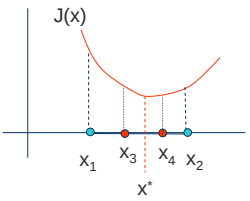
$x_0 = 1$

Converge a diferentes velocidades y puntos en función del punto inicial

Excel

Métodos de Reducción de Intervalos

Generar una sucesión de intervalos [x_1 , x_2], [x_3, x_4],.... de dentro de cada uno de los cuales esta el óptimo y longitud cada vez menor hasta llegar a la precisión requerida



2 pasos:

1. Encontrar un intervalo inicial que contenga a x^*
2. Reducir la longitud inicial hasta la precisión requerida

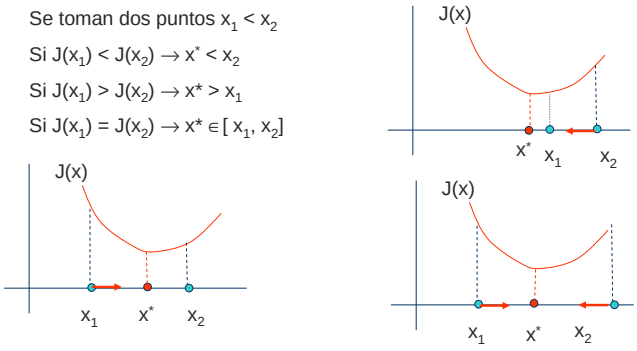
1 (Semi)Intervalo inicial

Se toman dos puntos $x_1 < x_2$

Si $J(x_1) < J(x_2) \rightarrow x^* < x_2$

Si $J(x_1) > J(x_2) \rightarrow x^* > x_1$

Si $J(x_1) = J(x_2) \rightarrow x^* \in [x_1, x_2]$



1 Intervalo inicial

Conocido un semi-intervalo inicial p.e. $[x_0, \infty)$ donde esta x*, para localizar un intervalo inicial se puede generar una secuencia de valores:

$x_1 = x_0 + \delta$

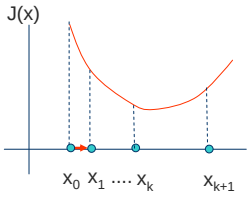
$x_2 = x_0 + 2\delta$

$x_3 = x_0 + 2^2\delta$

....

$x_k = x_0 + 2^{k-1}\delta$

Compromiso precisión / nº de iteraciones



Hasta que:

$J(x_{k-1}) > J(x_k) \leq J(x_{k+1})$

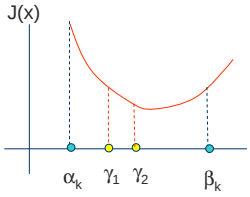
El intervalo inicial es $[x_{k-1}, x_{k+1}]$

δ Es positivo o negativo según el semiintervalo

2 Reducción del intervalo

Si en el paso k el intervalo en el que se encuentra el óptimo es

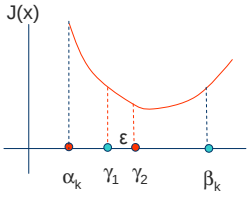
$[\alpha_k, \beta_k]$, se puede reducir su longitud $L_k = \beta_k - \alpha_k$ evaluando la función J(x) en dos puntos $\gamma_1 < \gamma_2$ internos al intervalo



$$J(\gamma_1) > J(\gamma_2) \Rightarrow [\alpha_{k+1}, \beta_{k+1}] = [\gamma_1, \beta_k]$$
$$J(\gamma_1) < J(\gamma_2) \Rightarrow [\alpha_{k+1}, \beta_{k+1}] = [\alpha_k, \gamma_2]$$
$$J(\gamma_1) = J(\gamma_2) \Rightarrow [\alpha_{k+1}, \beta_{k+1}] = [\gamma_1, \gamma_2]$$

¿Como elegir los dos puntos internos?

2 Reducción del intervalo, ε-minimax



$\gamma_1 + \varepsilon - \alpha_k = \beta_k - \gamma_1$

$$\gamma_1 = \frac{\beta_k + \alpha_k}{2} - \frac{\varepsilon}{2}$$
$$\gamma_2 = \frac{\beta_k + \alpha_k}{2} + \frac{\varepsilon}{2}$$

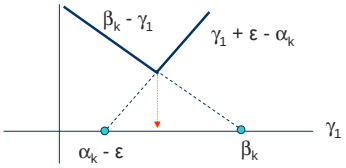
Simétricos respecto al centro del intervalo

Criterio: minimizar la longitud del mayor de los intervalos posibles

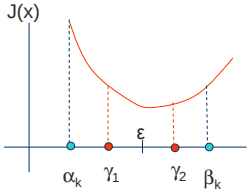
Min max { $\gamma_2 - \alpha_k$, $\beta_k - \gamma_1$, $\gamma_2 - \gamma_1$}

Si $\gamma_2 = \gamma_1 + \varepsilon$

Min max{$\gamma_1 + \varepsilon - \alpha_k$, $\beta_k - \gamma_1$}



2 Reducción del intervalo, ε-minimax



$\gamma_2 - \alpha_k = \beta_k - \gamma_1$

$$\gamma_1 = \frac{\beta_k + \alpha_k}{2} - \frac{\varepsilon}{2}$$
$$\gamma_2 = \frac{\beta_k + \alpha_k}{2} + \frac{\varepsilon}{2}$$

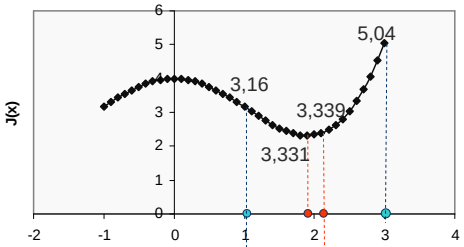
$$L_{k+1} = \gamma_2 - \alpha_k = \frac{\beta_k + \alpha_k}{2} + \frac{\varepsilon}{2} - \alpha_k =$$
$$= \frac{\beta_k - \alpha_k}{2} + \frac{\varepsilon}{2} = \frac{L_k + \varepsilon}{2}$$

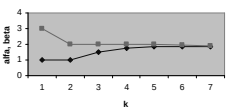
ε debe escogerse tan pequeño como se pueda para reducir la longitud del intervalo

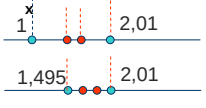
En N pasos:

$$L_N = \frac{L_0 + (2^N - 1)\varepsilon}{2^N} \approx \frac{L_0}{2^N}$$

Ejemplo $J(x) = x^2 + 4\cos(x)$







Minimizar J(x)

Intervalo inicial [1,3]

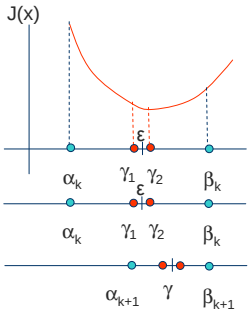
ε = 0.02

Puntos intermedios iniciales: 1 + (3-1)/2 ± ε/2

1,99, 2.01

Excel

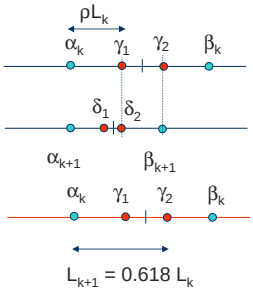
Sección dorada



El método del ε – minimax evalua la función J en dos puntos internos que no reutiliza en la siguiente iteración

Puede plantearse el mismo problema imponiendo la condición de que uno de los puntos sirva para la siguiente iteración. El algoritmo de la sección dorada está basado en esta idea

Método de la Sección dorada

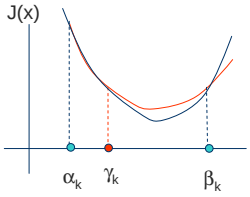


$L_{k+1} = 0.618\ L_k$

$$L_N \approx 0.618^N L_0$$

$$\gamma_1 - \alpha_k = \rho(\beta_k - \alpha_k) = \rho L_k = \beta_k - \gamma_2$$
$$\gamma_2 - \gamma_1 = (1-2\rho)L_k$$
$$\gamma_2 - \alpha_k = (1-\rho)L_k$$
$$\gamma_1 - \alpha_k = \delta_2 - \alpha_{k+1} = (1-\rho)L_{k+1} = (1-\rho)(\gamma_2 - \alpha_k)$$
$$\rho L_k = (1-\rho)^2 L_k$$
$$\rho^2 - 3\rho + 1 = 0$$
$$\rho = \frac{3 \pm \sqrt{5}}{2} = \begin{cases} \text{no es} < 0.5 \\ 0.382 \end{cases}$$
$$1 - \rho = 0.618$$

Metodos de aproximación por polinomios (2º orden)



$P(x) = a + bx + cx^2$

$P(\alpha_k) = a + b\,\alpha_k + c\,\alpha_k^{\,2} = J(\alpha_k)$

$P(\gamma_k) = a + b\,\gamma_k + c\,\gamma_k^{\,2} = J(\gamma_k)$

$P(\beta_k) = a + b\,\beta_k + c\,\beta_k^{\,2} = J(\beta_k)$

Si se conoce el valor de J(x) en tres puntos del intervalo $[\alpha_k, \beta_k]$ se puede calcular un polinomio P(x) de segundo orden que pase por dichos puntos y que se puede considerar una aproximación de J(x) en el intervalo.

El método usa el mínimo analítico del polinomio para estimar un nuevo punto (próximo al óptimo) y reducir la longitud del intervalo de incertidumbre, pero calculando un solo valor nuevo de J(x) en cada iteración

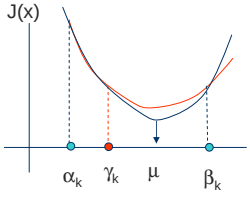
Métodos de aproximación por polinomios (2º orden)

$P(x) = a + bx + cx^2$

$P(\alpha_k) = a + b\,\alpha_k + c\,\alpha_k^{\,2} = J(\alpha_k)$

$P(\gamma_k) = a + b\,\gamma_k + c\,\gamma_k^{\,2} = J(\gamma_k)$

$P(\beta_k) = a + b\,\beta_k + c\,\beta_k^{\,2} = J(\beta_k)$



Sistema de ecuaciones lineales en a, b, c que pueden calcularse:

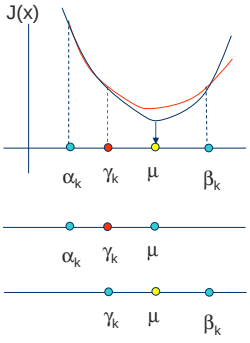
$$c = \frac{J(\beta_k) - J(\gamma_k)}{(\beta_k - \alpha_k)(\beta_k - \gamma_k)} + \frac{J(\alpha_k) - J(\gamma_k)}{(\beta_k - \alpha_k)(\gamma_k - \alpha_k)}$$

$$b = \frac{J(\gamma_k) - J(\alpha_k)}{\gamma_k - \alpha_k} - c(\gamma_k + \alpha_k)$$

Mínimo de $P(x) = a + bx + cx^2$

$\mu = -b / (2c)$, calcular $J(\mu)$ y reducir el intervalo

Métodos de aproximación por polinomios



Mínimo de $P(x) = a + bx + cx^2$

$\mu = -b / (2c)$, calcular $J(\mu)$ y reducir el intervalo

Cualquiera de los dos intervalos que pueden resultar de la reducción, contiene un punto interior y puede usarse para una nueva iteración

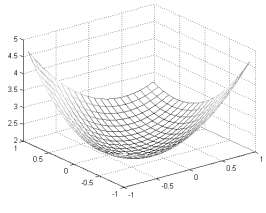
Son posibles otros métodos, p.e. de interpolación cúbica usando 4 puntos

Optimización multivariable

$$\min_x J(x)$$

$$x \in R^n$$

- Ejemplo

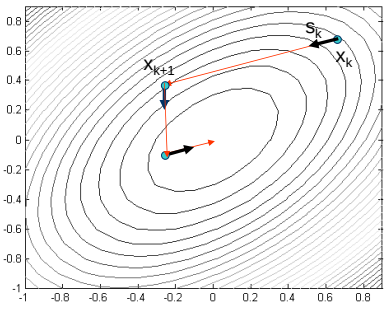


$$J(x_1, x_2) = x_1^2 + x_2^2 - x_1 x_2 + 2$$

$$J(x_1, x_2) = \frac{1}{2}(x_1, x_2)' \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + 2$$

x_k = valor del vector x en el paso k

Algoritmos de búsqueda del óptimo



Curvas de nivel de J(x)

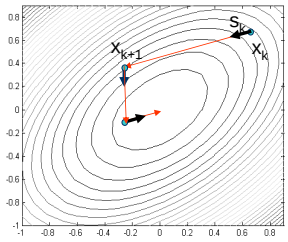
$$x_{k+1} = x_k + \Delta x_k = \\ = x_k + \sigma_k s_k$$

Métodos iterativos:

Partiendo de un valor inicial x_0 , se busca un nuevo punto en una cierta dirección de búsqueda s_k que de un mejor valor de J

Se continua iterando hasta estar suficientemente cerca del óptimo

Criterios de terminación



ε fija la tolerancia o precisión

$\varepsilon_0 > 0$ evita divisiones por cero

1 El gradiente es suficientemente pequeño

$$\left\| \frac{\partial J(x_k)}{\partial x} \right\| \le \varepsilon_1$$

2 La solución no avanza significativamente

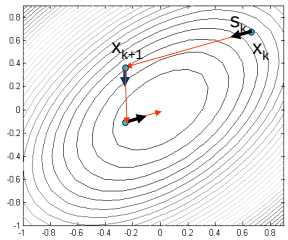
$$\frac{\|x_{k+1} - x_k\|}{\varepsilon_0 + \|x_k\|} \le \varepsilon_2$$

3 La función de costo no mejora significativamente

$$\frac{|J(x_{k+1}) - J(x_k)|}{\varepsilon_0 + |J(x_k)|} \le \varepsilon_3$$

4 El número de iteraciones excede un máximo N

Propiedades del algoritmo



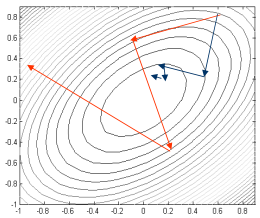
Convergencia local / global

El algoritmo es un sistema dinámico discreto

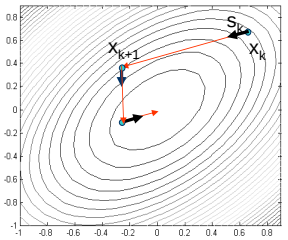
Estabilidad / Convergencia

La longitud del paso es cada vez menor

Convergencia al optimo



Propiedades del algoritmo



Velocidad de convergencia al óptimo de orden p

c velocidad de convergencia

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^p} \le c \qquad k \text{ grande}$$

$$0 < c < 1$$

Velocidad de convergencia superlineal

$$\lim_{k \to \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^p} = 0$$

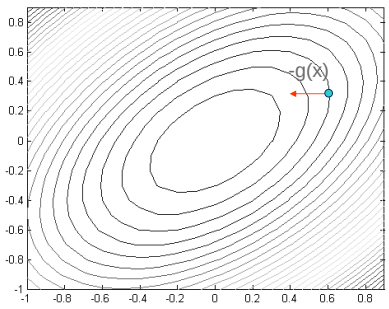
Optimización multivariable

$$\min_x J(x)$$

$$x \in R^n$$

- Variedad de enfoques:
  - Algoritmos basados en el gradiente
  - Algoritmos tipo Newton
  - Algoritmos basados en valores de la función

Métodos basados en el gradiente



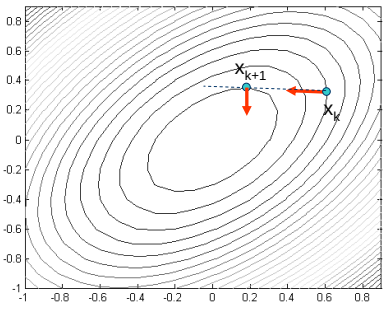
Curvas de nivel de J(x)

El vector gradiente de J(x) en un punto x indica la dirección en la que la función experimenta el máximo crecimiento a partir de ese punto.

El gradiente negativo indica la dirección de máxima disminución

$$g(x)' = \frac{\partial J}{\partial x}$$

Algoritmo del descenso mas pronunciado (Steepest descent)



Curvas de nivel de J(x)

$$x_{k+1} = x_k - \sigma_k \frac{\partial J(x_k)}{\partial x}' = $$
$$= x_k - \sigma_k g(x_k)$$
$$\min_{\sigma_k} J(x_k - \sigma_k g(x_k))$$
$$\text{parar si } \|g(x_k)\| \le \varepsilon$$

Avanzar todo lo que sea posible en la dirección de máximo descenso. En cada paso se hace una optimización escalar sobre σ_k

Algoritmo del descenso mas pronunciado con funciones cuadráticas

$$J(x) = a + b'x + \frac{1}{2}x'Cx$$
$$g(x) = b + Cx$$

C, simétrica positiva definida

$$\begin{cases} x_{k+1} = x_k - \sigma_k g(x_k) \\ \min_{\sigma_k} J(x_k - \sigma_k g(x_k)) \end{cases}$$

Las funciones cuadráticas son adecuadas para evaluar el algoritmo: muchas funciones pueden aproximarse adecuadamente por ellas cerca del óptimo y tienen solución conocida

¿Converge al óptimo cuando k→∞ ?

Velocidad de convergencia

Funciones cuadráticas

Tienen especial interés porque una función cualquiera continuamente diferenciable, puede aproximarse por una función cuadrática en las proximidades del óptimo:

$$J(x) = J(x^*) + \left.\frac{\partial J}{\partial x}\right|_{x^*}(x - x^*) + \frac{1}{2}(x - x^*)'\left.\frac{\partial^2 J(x)}{\partial x^2}\right|_{x^*}(x - x^*) + \ldots.$$

$$J(x) = a + b'x + \frac{1}{2}x'Cx \qquad C = \left.\frac{\partial^2 J(x)}{\partial x^2}\right|_{x^*}$$

La región x'Cx ≤ 1 es convexa si C es PSD

También son las funciones mas sencillas de modo que, si un método no va bien con una de ellas, probablemente no ira bien con otra función mas compleja

Algoritmo del descenso mas pronunciado con funciones cuadráticas

$$J(x) = a + b'x + \frac{1}{2}x'Cx \qquad g(x) = b + Cx$$

$$J(x_k - \sigma_k g(x_k)) = a + b'(x_k - \sigma_k g(x_k)) +$$
$$+ \frac{1}{2}(x_k - \sigma_k g(x_k))'C(x_k - \sigma_k g(x_k)) =$$
$$= J(x_k) - 2\frac{b'}{2}\sigma_k g(x_k) + \frac{1}{2}[-x_k'\sigma_k Cg(x_k) - \sigma_k g(x_k)'Cx_k +$$
$$+ \frac{1}{2}\sigma_k^2 g(x_k)'Cg(x_k)] = J(x_k) - \frac{1}{2}\left[\begin{matrix}(b + Cx_k)'\sigma_k g(x_k) - \\ -\sigma_k g(x_k)'(b + Cx_k) + \sigma_k^2 g(x_k)'Cg(x_k)\end{matrix}\right] =$$
$$= J(x_k) - \sigma_k \|g(x_k)\|^2 + \frac{1}{2}\sigma_k^2 g(x_k)'Cg(x_k)$$

Algoritmo del descenso mas pronunciado con funciones cuadráticas

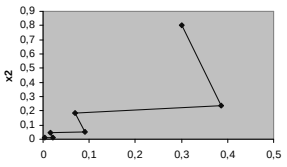
$$J(x_k - \sigma_k g(x_k)) = J(x_k) - \sigma_k \|g(x_k)\|^2 + \frac{1}{2}\sigma_k^2 g(x_k)'Cg(x_k)$$

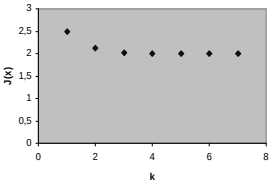
$$\min_{\sigma_k} J(x_k - \sigma_k g(x_k)) \Rightarrow \left.\frac{\partial J(x_k - \sigma_k g(x_k))}{\partial \sigma_k}\right|_{\sigma_k^*} = 0$$

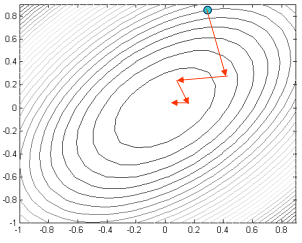
$$-\|g(x_k)\|^2 + \sigma_k^* g(x_k)'Cg(x_k) = 0$$

$$\sigma_k^* = \frac{\|g(x_k)\|^2}{g(x_k)'Cg(x_k)} \qquad x_{k+1} = x_k - \frac{\|g(x_k)\|^2}{g(x_k)'Cg(x_k)}g(x_k)$$

Ejemplo







$$J(x_1, x_2) = x_1^2 + x_2^2 - x_1x_2 + 2$$

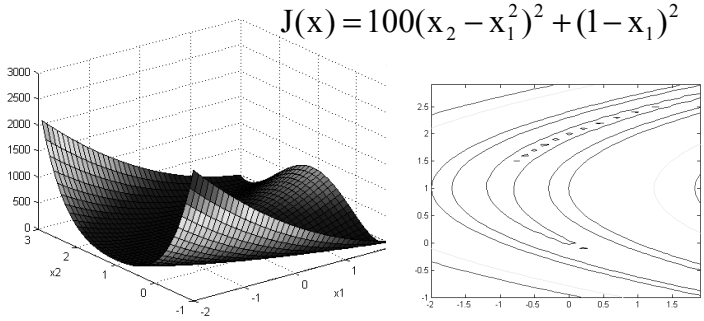
$$g(x) = \begin{bmatrix} 2x_1 - x_2 \\ 2x_2 - x_1 \end{bmatrix} \qquad C = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

$$x_0 = \begin{bmatrix} 0.8 \\ 0.3 \end{bmatrix} \qquad x_{k+1} = x_k - \sigma_k g(x_k)$$

Excel

Función de Rosenbrock (banana)

$$J(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$



Algoritmo del descenso mas pronunciado con funciones cuadráticas

¿Convergencia? Se alcanza el óptimo exactamente cuando $|g(x)|=0$

Con funciones cuadráticas, si no se alcanza el óptimo en la primera iteración, no se alcanza nunca

$$x_{k+1} = x_k - \sigma_k g(x_k) \qquad g(x) = b + Cx$$

$$g(x_{k+1}) = b + Cx_{k+1} = b + Cx_k - C\sigma_k g(x_k) =$$

$$= g(x_k) - \sigma_k C g(x_k)$$

Supongamos que $|g(x_0)| \neq 0$, Puede ocurrir que $g(x_0)$ sea o no sea vector propio de C.

Algoritmo del descenso mas pronunciado con funciones cuadráticas

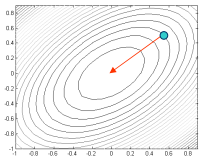
Si $g(x_0)$ es vector propio de C: $g(x_{k+1}) = g(x_k) - \sigma_k^* C g(x_k)$

$$Cg(x_0) = \lambda g(x_0)$$

$$g(x_1) = g(x_0) - \sigma_k^* C g(x_0) = g(x_0) - \sigma_k^* \lambda g(x_0) =$$

$$= g(x_0) - \frac{\|g(x_0)\|^2}{g(x_0)'\lambda g(x_0)} \lambda g(x_0) = 0$$

Y el optimo se alcanza en una iteración



Algoritmo del descenso mas pronunciado con funciones cuadráticas

En general si $g(x_0)$ no es vector propio de C:

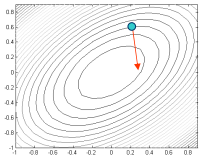
$$Cg(x_0) = \alpha g(x_0) + z \qquad z \neq 0 \quad \alpha \neq 0 \quad z \perp g(x_0)$$

$$g(x_1) = g(x_0) - \sigma_k^* C g(x_0) = g(x_0) - \sigma_1^* (\alpha g(x_0) + z) =$$

$$= g(x_0) - \frac{\|g(x_0)\|^2}{g(x_0)'(\alpha g(x_0) + z)} (\alpha g(x_0) + z) =$$

$$= g(x_0) - \frac{1}{\alpha}(\alpha g(x_0) + z) = -\frac{z}{\alpha} \neq 0$$

Y el optimo no se alcanza en la siguiente iteración



Algoritmo del descenso mas pronunciado con funciones cuadráticas

y además $g(x_1)$ no es vector propio de C:
En efecto, si se cumpliera:

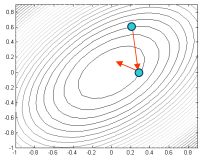
$$Cg(x_1) = \lambda g(x_1) \Rightarrow C(-\frac{z}{\alpha}) = \lambda(-\frac{z}{\alpha}) \Rightarrow Cz = \lambda z \Rightarrow z'C = \lambda z'$$

$$z'Cg(x_0) = \lambda z' g(x_0) = 0 \qquad (*)$$

pero

$$z'Cg(x_0) = z'(\alpha g(x_0) + z) = \|z\|^2 \neq 0$$

Lo que es contradictorio con la expresión (*), quedando probado por inducción.



En general el algoritmo del descenso mas pronunciado no alcanzará el óptimo exactamente cuando $k \to \infty$

Algoritmos tipo Newton

Se diseña para una función cuadrática y se aplica a una función cualquiera

Supongamos que J(x) es cuadrática. ¿Cómo escogeríamos Δx para alcanzar el óptimo en un paso?

$$J(x) = a + b'x + \frac{1}{2}x'Cx \qquad g(x) = b + Cx \qquad H = C$$

C es el hessiano o matriz de segundas derivadas de J

$$x_{k+1} = x_k + \Delta x_k$$

$$g(x_k + \Delta x_k) = b + C(x_k + \Delta x_k) = g(x_k) + C\Delta x_k$$

$$g(x_k + \Delta x_k) = 0 \Rightarrow g(x_k) + C\Delta x_k = 0$$

$$\Delta x_k = -C^{-1} g(x_k) \qquad x_{k+1} = x_k - C^{-1} g(x_k)$$

Algoritmo de Newton-Raphson

Por analogía, para una función cualquiera J(x) diferenciable dos veces, podemos usar el algoritmo:

$$s_k = -\left[\frac{\partial^2 J(x_k)}{\partial x^2}\right]^{-1} g(x_k) = -H(x_k)^{-1} g(x_k)$$

$$\min_{\sigma_k} J(x_k - \sigma_k H(x_k)^{-1} g(x_k))$$

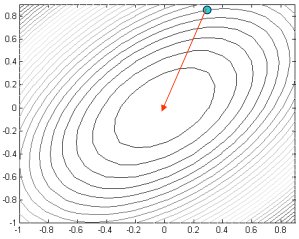
Método de segundo orden

$$x_{k+1} = x_k + \sigma_k s_k$$

s_k dirección de busqueda en el paso k

A medida que la función se acerque al óptimo se asemejará mas a una cuadrática y el algoritmo convergerá rápidamente al óptimo

Ejemplo



$$J(x_1, x_2) = x_1^2 + x_2^2 - x_1 x_2 + 2$$

$$g(\mathbf{x}) = \begin{bmatrix} 2x_1 - x_2 \\ 2x_2 - x_1 \end{bmatrix} \quad C = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

Al ser cuadrática:

$$x_{k+1} = x_k - Cg(x_k)$$

$$\mathbf{x}_0 = \begin{bmatrix} 0.8 \\ 0.3 \end{bmatrix}$$

$$\mathbf{x}_1 = \begin{bmatrix} 0.8 \\ 0.3 \end{bmatrix} - \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 1.3 \\ -0.2 \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.3 \end{bmatrix} - \begin{bmatrix} 0.8 \\ 0.3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$g(\mathbf{x}_1) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Excel

Ventajas / Inconvenientes del método de Newton-Raphson

Ventajas:

Normalmente requiere menos iteraciones

Inconvenientes:

- ✓Requiere conocer el gradiente y el hessiano
- ✓Requiere invertir el hessiano
- ✓No hay garantía de que el hessiano sea PD y el algoritmo converja

El gradiente y el hessiano pueden aproximarse por diferencias finitas

La inversión del hessiano puede cambiarse por la resolución de un sistema de ecuaciones lineales en s_k:

$$\left[\frac{\partial^2 J(x_k)}{\partial x^2}\right] s_k = -g(x_k)$$

Convergencia

$$s_k = -H(x_k)^{-1} g(x_k)$$

$$\min_{\sigma_k} J(x_k - \sigma_k H(x_k)^{-1} g(x_k))$$

$$x_{k+1} = x_k + \sigma_k s_k$$

En una aproximación de primer orden:

$$J(x_{k+1}) \approx J(x_k) + \left.\frac{\partial J(x_k)}{\partial x}\right| \Delta x_k =$$

$$= J(x_k) + g(x_k)' \sigma_k s_k =$$

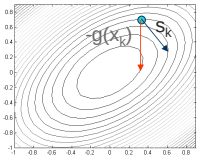
$$= J(x_k) - \sigma_k g(x_k)' \left[\frac{\partial^2 J(x_k)}{\partial x^2}\right]^{-1} g(x_k)$$

Comprobación: s_k es una dirección de descenso si:

$$-g(x_k)' s_k > 0$$

Si el hessiano no es PD no hay garantía de que J(x) vaya disminuyendo en cada iteración

Solo si J(x) es convexa hay garantía de que H es PD



Algoritmo de Marquardt-Levenberg

Modifica el Hessiano para asegurar que sea PD en cada paso

$$x_{k+1} = x_k + \sigma_k s_k$$

$$s_k = -\left[\frac{\partial^2 J(x_k)}{\partial x^2} + \beta_k I\right]^{-1} g(x_k) \qquad \beta_k \geq 0$$

$$\text{escoger } \beta_k \text{ para que } \left[\frac{\partial^2 J(x_k)}{\partial x^2} + \beta_k I\right] \text{ sea PD}$$

$$\min_{\sigma_k} J(x_k - \sigma_k \left[\frac{\partial^2 J(x_k)}{\partial x^2} + \beta_k I\right]^{-1} g(x_k))$$

Algoritmo de Marquardt-Levenberg

$$\text{escoger} \quad x_0, \quad \beta_0$$

$$\text{resolver en } s_k \quad \left[\frac{\partial^2 J(x_k)}{\partial x^2} + \beta_k I\right] s_k = -g(x_k)$$

$$\text{si} \quad -g(x_k)' s_k \leq 0 \Rightarrow \beta_k = 2\beta_k \quad \text{recalcular } s_k$$

$$\min_{\sigma_k} J(x_k - \sigma_k \left[\frac{\partial^2 J(x_k)}{\partial x^2} + \beta_k I\right]^{-1} g(x_k))$$

$$x_{k+1} = x_k + \sigma_k s_k$$

Minimización respecto a σ_k

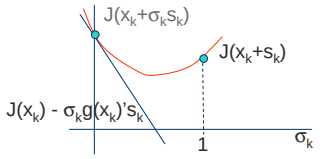
Pueden aplicarse los métodos de optimización monovariable

A veces se prefiere usar un procedimiento mas simple con la condición de que $J(x_{k})$ decrezca.

El menor valor de σ_k es 0 y el método de Newton puro es con σ_k =1

Puede evaluarse J para σ_k 0 y 1, si no decrece:

Puede hacerse una interpolación cuadrática conocidos $J(x_k+s_k)$ $J(x_k)$ y $g(x_k)'s_k$ y calcular su mínimo como estimación de σ_k



$$\hat{\sigma}_k = \frac{-g(x_k)'s_k}{2[J(x_k+s_k)-J(x_k)-g(x_k)s_k]}$$

Métodos quasi-Newton

Tratan de evitar el cálculo de la inversa del hessiano, sustituyéndolo por una matriz $\hat{H}_k$ positiva definida que se actualiza en cada paso y debe aproximarse a $H(x_k)^{-1}$

$s_k = -\tilde{H}_k g(x_k)$ Dirección de búsqueda

$\tilde{H}_{k+1} = \tilde{H}_k + T_k$ T_k matriz de corrección

Si J(x) fuese cuadrática, para cualquier Δx_k se verifica:

$$\Delta g(x_k) = g(x_{k+1}) - g(x_k) = C\Delta x_k$$

Si queremos que H_{k+1} se aproxime a C^{-1}:

$$\Delta x_k = C^{-1}\Delta g(x_k) \Rightarrow \Delta x_k = \tilde{H}_{k+1}\Delta g(x_k)$$

$$(\tilde{H}_k + T_k)\Delta g(x_k) = \Delta x_k$$

Métodos quasi-Newton

Si se encuentra una relación $H_{k+1} = H_k + T_k$ que verifique $\Delta x_k = H_{k+1}\Delta g(x_k)$ durante n pasos, y la función es cuadrática, entonces $H_n = C^{-1}$

$$\Delta x_k = C^{-1}\Delta g(x_k)$$
$$\Delta x_k = \tilde{H}_{k+1}\Delta g(x_k)$$
$$k = 0,1,2,...n-1$$

Si la función J(x) fuera cuadrática, despues de n pasos, se debe verificar que $H_n = C^{-1}$ pues verifican las mismas ecuaciones.

Por tanto, un algoritmo basado en $\hat{H}$ alcanzaría el mínimo en n pasos

Se La idea se aplica por extensión a una J(x) cualquiera

Métodos quasi-Newton

Hay muchas expresiones para T_k que pueden cumplir la relación

$$(\tilde{H}_k + T_k)\Delta g(x_k) = \Delta x_k \quad \text{p.e.}: \forall \alpha, \beta \in R^n, \neq 0$$

$$T_k = \frac{\Delta x_k \alpha'}{\alpha'\Delta g(x_k)} - \frac{\tilde{H}_k \Delta g(x_k)\beta'}{\beta'\Delta g(x_k)}$$

$$(\tilde{H}_k + T_k)\Delta g(x_k) = \tilde{H}_k\Delta g(x_k) + \frac{\Delta x_k \alpha'}{\alpha'\Delta g(x_k)}\Delta g(x_k) -$$

$$-\frac{\tilde{H}_k \Delta g(x_k)\beta'}{\beta'\Delta g(x_k)}\Delta g(x_k) = \tilde{H}_k\Delta g(x_k) + \Delta x_k - \tilde{H}_k\Delta g(x_k) = \Delta x_k$$

Algoritmo DFP (Davidon, Fletcher, Powell)

En particular para:

$$\alpha = \Delta x_k \quad \beta = \tilde{H}_k \Delta g(x_k)$$

$$T_k = \frac{\Delta x_k \alpha'}{\alpha'\Delta g(x_k)} - \frac{\tilde{H}_k \Delta g(x_k)\beta'}{\beta'\Delta g(x_k)}$$

$$T_k = \frac{\Delta x_k \Delta x_k'}{\Delta x_k'\Delta g(x_k)} - \frac{\tilde{H}_k \Delta g(x_k)(\tilde{H}_k\Delta g(x_k))'}{\Delta g(x_k)'\tilde{H}_k\Delta g(x_k)}$$

Algoritmo DFP (Davidon, Fletcher, Powell)

$$\text{escoger } x_0 \quad \tilde{H}_0 \text{ PD, simetrica}$$

$$\min_{\sigma_k} J(x_k - \sigma_k \tilde{H}_k g(x_k))$$

$$x_{k+1} = x_k - \sigma_k \tilde{H}_k g(x_k)$$

$$\Delta x_k = x_{k+1} - x_k \quad \Delta g(x_k) = g(x_{k+1}) - g(x_k)$$

$$\tilde{H}_{k+1} = \tilde{H}_k + \frac{\Delta x_k \Delta x_k'}{\Delta x_k'\Delta g(x_k)} - \frac{\tilde{H}_k \Delta g(x_k)(\tilde{H}_k\Delta g(x_k))'}{\Delta g(x_k)'\tilde{H}_k\Delta g(x_k)}$$

Algoritmo BFGS (Broyden, Fletcher, Goldfarb, Shanno) 1970

$$C\Delta x_k = \Delta g(x_k)$$
$$B_{k+1}\Delta x_k = \Delta g(x_k)$$
$$\Delta x_k = \tilde{H}_{k+1}\Delta g(x_k)$$

Estima recursivamente el Hessiano en lugar de su inversa

Δx y Δg juegan papeles simetricos al caso DFP

intercambiar Δx_k con $\Delta g(x_k)$ en la formula de T_k

$$B_{k+1} = B_k + \frac{\Delta g(x_k)\Delta g(x_k)'}{\Delta x_k'\Delta g(x_k)} - \frac{B_k\Delta x_k\Delta x_k'B_k}{\Delta x_k'B_k\Delta x_k}$$

B_k siempre es PD si J(x) es convexa

$\tilde{H}_{k+1} = B_{k+1}^{-1}$ puede estimarse mediante

$$(A+zv')^{-1} = A^{-1} - \frac{A^{-1}zv'A^{-1}}{1+v'A^{-1}z}$$

Si B_k es PD y $\Delta x_k'\Delta g(x_k) > 0$ entonces B_{k+1} es PD, si no se cumple no se actualiza B_k

Algoritmo BFGS (Broyden, Fletcher, Goldfarb, Shanno) 1970

escoger x_0, $\tilde{H}_0$ PD, simétrica

$s_k = -\tilde{H}_k g(x_k)$

En general es mas eficiente que DFP

$\min_{\sigma_k} J(x_k + \sigma_k s_k)$

$x_{k+1} = x_k + \sigma_k s_k$

$\Delta x_k = x_{k+1} - x_k \qquad \Delta g(x_k) = g(x_{k+1}) - g(x_k)$

$$\tilde{H}_{k+1} = \left[I - \frac{\Delta x_k\Delta g(x_k)'}{\Delta x_k'\Delta g(x_k)}\right]\tilde{H}_k\left[I - \frac{\Delta x_k\Delta g(x_k)'}{\Delta x_k'\Delta g(x_k)}\right] + \frac{\Delta x_k\Delta x_k'}{\Delta x_k'\Delta g(x_k)}$$

Métodos multivariables basados solo en valores de J(x) (Direct search)

- Los métodos basados en el gradiente son eficaces con funciones “suaves” y pueden funcionar adecuadamente con muchas variables
- No obstante, en muchos casos prácticos, la evaluación analítica del gradiente puede ser complicada o imposible en algunos puntos debido a discontinuidades, fuertes no linealidades, etc.
- En algunos casos, una solución es utilizar estimaciones del gradiente basadas en cocientes de incrementos
- Otra posibilidad es utilizar algoritmos que no estén basados en el cálculo del gradiente, p.e.:
  - Método del simplex
  - Método de las direcciones conjugadas de Powell

Método de la búsqueda Simplex

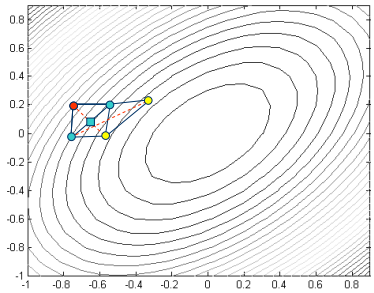
✓ Los métodos de búsqueda directa basados en patrones evalúan la función a minimizar J(x) en una serie de puntos mas o menos regularmente y se usan estos valores para evolucionar hacia un nuevo patrón de puntos mas cercano al óptimo

✓ La figura geométrica mas sencilla en un espacio de n dimensiones se denomina un Simplex y tiene n+1 vértices, así por ejemplo en R^2 es un triangulo, en R^3 un tetraedro, etc.

✓ El método de la búsqueda Simplex usa puntos situados en los n+1 vértices de esta figura geométrica para generar otro simplex más próximo al óptimo y continua iterando hasta alcanzarlo con la precisión deseada.

✓ No tiene nada que ver, excepto el nombre, con el método Simplex de LP

Método de la búsqueda Simplex



vértice
centroide

1 Se evalua la función en los n+1 vértices del simplex

2 Se escoge el peor vértice y se proyecta una cierta distancia a traves del centroide de los otros vértices

3 Se forma así un nuevo simplex con el vértice proyectado y el resto de los vértices

4 Si se mejora se sigue iterando hasta la tolerancia requerida

Método de la búsqueda Simplex

A medida que se progresa en las iteraciones es posible que o se alcanza el óptimo o se generen ciclos entre dos o mas simplex. Para evitar estas situaciones cíclicas se pueden aplicar tres reglas:

1 Si el vértice peor se generó en la anterior iteración entonces escoger el vértice con el siguiente valor mayor

2 Si un vértice no cambia durante mas de M iteraciones, reducir el tamaño del simplex por algun factor tomando como base el punto con menor valor. Sugerencia: $M = \text{int}(1.65n + 0.05n^2)$

3 La búsqueda se termina cuando el simplex es suficientemente pequeño o la desviación estándar de los valores de J(x) en los vértices es suficientemente pequeña

Generación de puntos

Partiendo del punto base $x^{(0)}$ y un factor de escala α las coordenadas de los otros vertices $x^{(i)}$, i= 1,...,n de un simplex regular inicial pueden calcularse mediante:

$$x_j^{(i)} = \begin{cases} x_j^{(0)} + \left[\dfrac{\sqrt{n+1}+n-1}{n\sqrt{2}}\right]\alpha & \text{si } j = i \\ x_j^{(0)} + \left[\dfrac{\sqrt{n+1}-1}{n\sqrt{2}}\right]\alpha & \text{si } j \neq i \end{cases}$$

Si $x^{(j)}$ es el vértice que debe reflejarse, el centroide de los otros puntos x_c y el nuevo punto reflejado se encuentra en:

$$x_c = \frac{1}{n}\sum_{\substack{i=0 \\ i\neq j}}^{n} x^{(i)} \qquad x_{new}^{(j)} = 2x_c - x_{old}^{(j)}$$

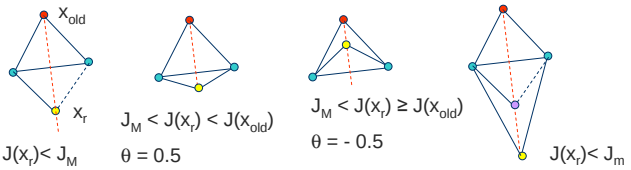
Método de Nelder - Mead

J_m valor inferior del simplex

J_M segundo valor superior del simplex

En lugar de mantener un simplex regular, expande o contrae el simplex en la dirección de avance de acuerdo a una serie de reglas

Reflexión normal



$J_m < J(x_r) < J_M$

$\theta = 1$

$J_M < J(x_r) < J(x_{old})$

$\theta = 0.5$

$J_M < J(x_r) \geq J(x_{old})$

$\theta = -0.5$

$J(x_r) < J_m$

$\theta = 2$

$$x_c = \frac{1}{n}\sum_{\substack{i=0 \\ i\neq j}}^{n} x^{(i)} \qquad x_{new}^{(j)} = x_{old}^{(j)} + (1+\theta)(x_c - x_{old}^{(j)}) \qquad -1 \leq \theta < 1$$

Método de Nelder- Mead

- Ventajas:
  - Cálculos sencillos con evaluaciones de la función únicamente y poco almacenamiento
  - Pocos parámetros ajustables
  - Robusto frente a errores y ruidos en la evaluación de la función al utilizar el peor valor
- Inconvenientes:
  - Se necesita un escalado de las variables
  - Es lento al no utilizar información de iteraciones pasadas ni estructural

Método de las direcciones conjugadas de Powell

Al igual que otros métodos, se diseña pensando en una función cuadrática y se aplica a una función cualquiera

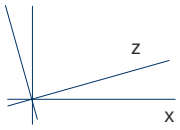
$$J(x) = a + b'x + \frac{1}{2}x'Cx$$

Trata de encontrar el mínimo de J(x) sin usar los valores del gradiente o el hessiano

La idea básica es buscar el mínimo a lo largo de unas direcciones tales que en cada una la función solo dependa de una componente del vector x facilitando así la búsqueda

Estas direcciones se denominan C conjugadas

Direcciones C conjugadas



Si una matriz S diagonaliza a C, de modo que S'CS = D diagonal, entonces en las coordenadas z = S^{-1}x :

$$J(x) = a + b'x + \frac{1}{2}x'Cx =$$

$$= a + b'Sz + \frac{1}{2}z'S'CSz = a + b'Sz + \frac{1}{2}z'Dz$$

Como no hay términos cruzados en z , la minimización de J(Sz) puede resolverse como n problemas de minimización respecto a cada componente z_j de z

$$J(x) = J(Sz) = 3 + \begin{bmatrix} 2 & 1 \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} z_1 & z_2 \end{bmatrix}\begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = 3 + 2z_1 + z_2 + 3z_1^2 + 2z_2^2 =$$

$$= \left(3 + 2z_1 + 3z_1^2\right) + \left(z_2 + 2z_2^2\right) = J_1(z_1) + J_2(z_2)$$

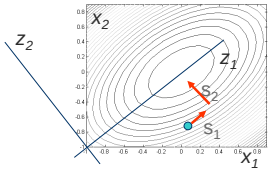
Direcciones C conjugadas

$$x = Sz = \begin{bmatrix} s_1 \vdots & s_2 \vdots & \dots & s_n \vdots \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} = z_1 s_1 + z_2 s_2 + \dots + z_n s_n$$

Minimizar respecto a cada componente z_j de z equivale a minimizar a lo largo de cada una de las n direcciones s_j a las que se denominan direcciones C conjugadas

Los nuevos ejes coinciden con las direcciones principales de la función

Así en n iteraciones llegaríamos al óptimo de una función cuadrática



Direcciones C conjugadas

La condición S'CS = D diagonal equivale a:

$$\begin{bmatrix} s_1' \\ s_2' \\ \vdots \\ s_n' \end{bmatrix} C \begin{bmatrix} s_1 & | & s_2 & | & \dots & | & s_n \end{bmatrix} = \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & a_{nn} \end{bmatrix}$$

$$s_i' C s_j = 0 \quad i \neq j$$

Definición: dada C (n x n) simétrica, las direcciones s_1, s_2, ...s_r $r \leq n$ son C-conjugadas si son linealmente independientes y verifican

$$\boxed{s_i' C s_j = 0 \quad i \neq j}$$

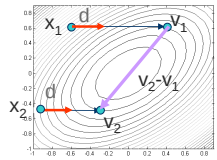
Propiedad de los subespacios paralelos

Dada una función J(x) cuadrática y una dirección d, $\forall\ x_1 \neq x_2 \in R^n$ se verifica que si v_1 es la solución de

$$\min_{\sigma} J(x_1 + \sigma d)$$

y si v_2 es la solución de $\min_{\sigma} J(x_2 + \sigma d)$

Entonces la dirección $v_2 - v_1$ es C-conjugada a d



Propiedad de los subespacios paralelos

Demostración

$$J(x) = a + b'x + \tfrac{1}{2} x'Cx \qquad g(x) = b + Cx$$

$$J(w) = J(x + \sigma d) \qquad \text{en el óptimo:}$$

$$\frac{\partial J}{\partial \sigma} = \frac{\partial J}{\partial w}\frac{\partial w}{\partial \sigma} = (b' + w'C)d = 0$$

$$\left.\begin{aligned}(b' + v_2'C)d = 0 \\ (b' + v_1'C)d = 0\end{aligned}\right\} (v_2' - v_1')Cd = 0$$

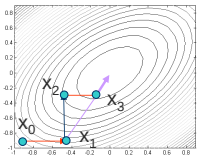
Luego la dirección $v_2 - v_1$ es C conjugada a d

Puede extenderse a n direcciones: Si partiendo de x_1 y x_2 se obtienen v_1 y v_2 a través de m < n búsquedas a lo largo de las m direcciones conjugadas s_1, s_2,...,s_m entonces $v_2 - v_1$ es C-conjugado con todas las direcciones s_1, s_2,...,s_m

Método de las direcciones conjugadas de Powell

Para generar dos direcciones conjugadas se han usado 2 puntos de partida y dos minimizaciones en una dirección d. Puede obtenerse el mismo resultado con un punto de partida y mas minimizaciones:

En la figura, minimizando sucesivamente a lo largo de las direcciones de los n ejes de x, la minimización en la iteración n+1 es paralela a la 1, con lo cual el vector $x_{n+1} - x_1$ es C-conjugado al del primer eje. Minimizando en esa dirección y aplicando el procedimiento sucesivamente puede hacerse la minimización a lo largo de n direcciones C-conjugadas y, por tanto, llegar al óptimo.



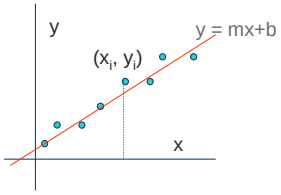
Método de las direcciones conjugadas de Powell

1. Escoger x_0 y n direcciones linealmente independientes, p.e. $s_i = e_i$
2. Formar el conjunto de n+1 direcciones de búsqueda s_n, s_1, s_2, s_3, ...,s_n
3. Minimizar J(x) a lo largo de las n+1 direcciones de búsqueda sucesivamente. Sea v_j el optimo en la iteración j
4. Calcular una nueva dirección de búsqueda como $s_{n+1} = v_{n+1} - v_1$ que será conjugada a s_n (y a las anteriores que se hayan generado)
5. Usar como nuevo conjunto de n+1 direcciones de búsqueda s_{n+1}, s_2, s_3, ...,s_n ,s_{n+1} donde s_1 se ha eliminado y se ha introducido $s_{n+1} = v_{n+1} - v_1$
6. Comprobar las condiciones de optimo y de indep. lineal de las n s_i diferentes
7. Volver a 3

Método de las direcciones conjugadas de Powell

- Si la función es cuadrática, después de n bucles, las n+1 búsquedas se hacen sobre direcciones conjugadas y se alcanzará el optimo
- Si no lo es, se puede probar que converge de forma superlineal a un óptimo
- Es un método eficiente y fiable

Ajuste de datos por mínimos cuadrados



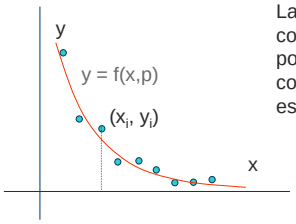
Encontrar la recta que mejor se ajusta a un conjunto de N parejas de datos (x_i, y_i). Puede plantearse como un problema de optimización: Buscar los parámetros de la recta (m , b) que minimiza la suma de los cuadrados de las desviaciones entre los datos y el valor de la recta

$$\min_{m,b} \sum_{i=1}^{N} (y_i - (mx_i + b))^2$$

Tiene solución analítica

$$\frac{\partial \sum_{i=1}^{N} (y_i - (mx_i + b))^2}{\partial m} = 0 \qquad \frac{\partial \sum_{i=1}^{N} (y_i - (mx_i + b))^2}{\partial b} = 0$$

Ajuste de datos



La idea se puede extender al ajuste del conjunto de las N parejas de datos (x_i , y_i) por una función cualquiera y= f(x,p) que contiene un conjunto de parámetros a estimar p

$$\min_{p} \sum_{i=1}^{N} (y_i - f(x_i, p))^2$$

El problema se plantea como la minimización de la suma de los cuadrados de los residuos $y_i - f(x_i,p)$ respecto a los parametros p de la función

Ecuación de Redlich-Kwong

Relación empírica entre la

Presión P

Temperatura T

Volumen molar v

de un gas real

$$P = \frac{RT}{v - b} - \frac{a}{v(v + b)\sqrt{T}}$$

a y b son coeficientes que deben ajustarse a datos experimentales

Ejemplo, datos de CO_2

Excel

volumen molar X1	Temperatura X2	Presión Y
500	273	33
500	323	43
600	373	45
700	273	26
600	323	37
700	373	39
400	272	38
400	373	63,6

Resolución de ecuaciones

En muchos problemas aparece la necesidad de resolver ecuaciones del tipo f(x) = 0

O sistemas de ecuaciones del tipo: $\begin{cases} f(x,y) = 0 \\ g(x,y) = 0 \end{cases}$

Existen métodos para resolverlas tales como:

- ✓Newton
- ✓Secante
- ✓Bisección

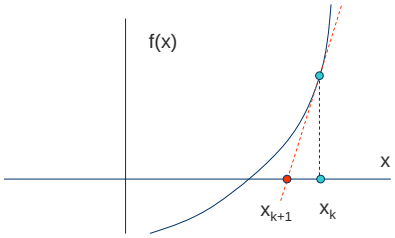
Pero también pueden resolverse como problemas de optimización

Método de Newton

$$f(x) = 0$$

$$f(x_{i+1}) = f(x_i) + \left.\frac{\partial f}{\partial x}\right|_{x_i} (x_{i+1} - x_i) + \ldots = 0$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$



Newton-Raphson

$$F(x) = 0$$

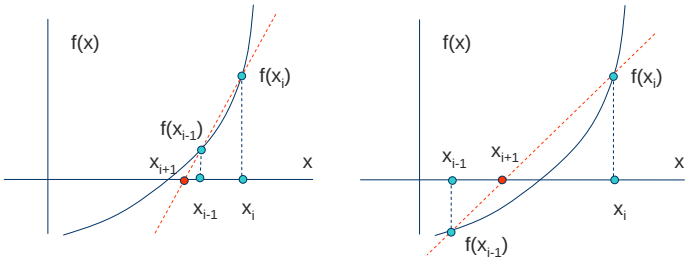
$$F(x_{i+1}) = F(x_i) + \left.\frac{\partial F}{\partial x}\right|_{x_i} (x_{i+1} - x_i) + \ldots = 0$$

$$x_{i+1} = x_i - \left[\left.\frac{\partial F}{\partial x}\right|_{x_i}\right]^{-1} F(x_i)$$

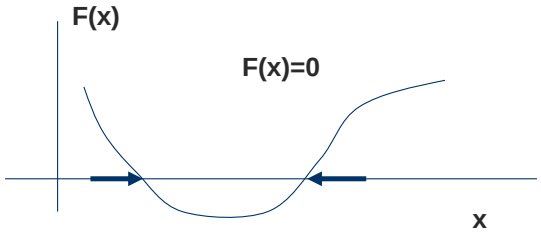
Exige estimar e invertir el Jacobiano en cada iteración

Método de la secante

$$x_{i+1} = \frac{x_{i-1} f(x_i) - x_i f(x_i)}{f(x_i) - f(x_{i-1})}$$



Problema de valores iniciales



Resolución por optimización

El problema puede formularse como:

$$\begin{cases} f(x,y) = 0 \\ g(x,y) = 0 \end{cases}$$

$$\min_{x,y} \quad \varepsilon_1^2 + \varepsilon_2^2$$
$$\begin{cases} f(x,y) = \varepsilon_1 \\ g(x,y) = \varepsilon_2 \end{cases}$$

El mínimo de $\varepsilon_1^2 + \varepsilon_2^2$, si existe, esta en (0,0), con lo que x e y verifican el sistema de ecuaciones

Resolución numérica de problemas de optimización

Una vez que se ha formulado un problema de optimización, suele ser conveniente ponerlo en forma tal que se facilite su resolución numérica y se aumente la eficiencia de la búsqueda de esa solución.

Entre las modificaciones posibles están:

- ✓El escalado de las variables independientes
- ✓Las transformaciones para evitar cálculos fuera de rango log(x), $x^{½}$, …
- ✓Las transformaciones para evitar no diferenciabilidades
- ✓Las transformaciones para mejorar la convexidad del problema

Además es importante un ajuste adecuado de las precisiones, tolerancias, nº de pasos, etc. del algoritmo de optimización

Escalado

El escalado se refiere a los valores relativos de las variables del problema, los cuales no deberían ser muy diferentes para evitar problemas numéricos ocasionados por sensibilidades distintas en distintas direcciones.

Ejemplo: x_1 toma valores en torno a 100 y x_2 en torno a 0.1

$$J(x_1, x_2) = 10x_1 + 5x_2 - x_1 x_2$$

Pueden ser reformulado en términos de las variables u_1, u_2 escaladas

$$J(u_1, u_2) = 1000\left(\frac{x_1}{100}\right) + 0.5\left(\frac{x_2}{0.1}\right) - 10\left(\frac{x_1}{100}\right)\left(\frac{x_2}{0.1}\right) = 1000u_1 + 0.5u_2 - 10u_1u_2$$

$$u_1 = \frac{x_1}{100} \quad u_2 = \frac{x_2}{0.1}$$

Ahora u_1 y u_2 toman valores en torno a 1

Convexificación

$J(x_1, x_2) = x_1 x_2$ Función no convexa en x

$x_1 = e^{v_1} \quad x_2 = e^{v_2}$ Cambio de variable

$x_1 x_2 = e^{v_1} e^{v_2} = e^{v_1 + v_2}$

$\min_{x_1, x_2} J(x_1, x_2) = \min_{v_1, v_2} J(v_1, v_2)$ Función convexa en v

Problema de rango!