Optimización

Primer Cuatrimestre 2025

Práctica de Laboratorio N° 3

Método de Gradiente Conjugado (Cuadráticas)

Sea $f : \mathbb{R}^n \to \mathbb{R}$ dada por $\frac{1}{2}x^T Ax + bx^T + c$ con $A \in \mathbb{R}^{n\times n}$ definida positiva, $b \in \mathbb{R}^n$, $c \in \mathbb{R}$. La idea consiste en que las direcciones conjugadas $\{d_0, \ldots, d_k\}$ sean definidas a partir del gradiente de f. Sea $x_0 \in \mathbb{R}^n$, se define $d_0 = -\nabla f(x_0)$ y, para $k = 0, 1, \ldots, n-2$ se define:

$$d^{k+1} = -\nabla f(x_{k+1}) + \beta_k d_k$$

donde β_k es calculado de manera tal que d_k y d_{k+1} sean A-conjugados:

$$\beta_k = \frac{(d_k)^T A \nabla f(x_{k+1})}{(d_k)^T A d_k}$$

Obs:

- d_k son direcciones de descenso
- la secuencia definida anteriormente alcanza el minimizador x^* en n pasos.

Ejercicio 1 Implementar el método de gradiente conjugado descripto anteriormente y usarlo para minimzar la función $f(x) = \frac{1}{2}x^t Qx - B^t \cdot x$, donde la matriz Q está dada por:

$$Q = \begin{pmatrix} Q1 & Q2 & Q3 & Q4 \\ Q2 & Q1 & Q2 & Q3 \\ Q3 & Q2 & Q1 & Q2 \\ Q4 & Q3 & Q2 & Q1 \end{pmatrix}$$

donde `Q1=[12 8 7 6;8 12 8 7;7 8 12 8;6 7 8 12]`

```
Q2=[3 2 1 0;2 3 2 1;1 2 3 2;0 1 2 3]
Q3=[2 1 0 0;1 2 1 0;0 1 2 1;0 0 1 2]
Q4=[1 0 0 0;0 1 0 0;0 0 1 0;0 0 0 1]
B=-[1;1;1;1;0;0;0;0;1;1;1;1;0;0;0;0]
```

Método de Gradiente Conjugado (Extensión para funciones no cuadráticas)

Se deben encontrar otras maneras de obtener la longitud del paso t_k y el coeficiente β_k. El paso se busca como hemos hecho antes usando Armijo, por ejemplo.

Para definir el coeficiente β_k se puede utilizar la fórmula de Fletcher y Reeves:

$$\beta_k^{FR} = \frac{\nabla f(x^{k+1})^T \nabla f(x^{k+1})}{\nabla f(x^k)^T \nabla f(x^k)}$$

Gradientes Conjugados para funciones no cuadráticas no necesariamente terminan en n pasos. Una práctica que da buenos resultados es reiniciar el β_k cada n iteraciones.

Ejercicio 2 Usar el algoritmo de Fletcher-Reeves para hallar el mínimo de $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$. Usarlo con tres puntos de partida distintos: $x_0 = (-2, 2)$, $x_0 = (2, -2)$, y $x_0 = (-2, -2)$.

Métodos de Cuasi-Newton

La idea, como en el caso del Método de Newton, es aproximar f mediante una expresión cuadrática. Sin embargo, en los métodos cuasi-Newton, se utiliza, en vez de $Hf(x)$, una matriz B_k que sea simétrica definida positiva.

Algoritmo general de un método Cuasi-Newton

Dados: f, $x_0 \in \mathbb{R}^n$, $H_0 \in \mathbb{R}^{n\times n}$ definida positiva, $\varepsilon > 0$, $k_{MAX} > 0$, $k = 0$.
while $\|\nabla f(x_k)|| > \varepsilon$, $\|x_{k+1} - x_k\| > \varepsilon$ y $k < k_{MAX}$
defino $d_k = -H_k \nabla f(x_k)$
tomo t_k por Armijo, Wolfe, etc
hacer $x_{k+1} = x_k + t_k d_k$
Determinar H_{k+1}
$k = k + 1$
end

Obs: Como H_0 podemos tomar la identidad, ya que es definida positiva. También podría tomarse $Hf(x_0)$ si es definida positiva.

Sean $q^k = \nabla f(x^{k+1}) - \nabla f(x^k)$ y $p^k = x^{k+1} - x^k$ se demuestra que una propiedad que debe satisfacer H_{k+1} definida por el algoritmo es que:

$$H_{k+1} q^j = p^j \quad \forall j = 0, 1, \ldots, k$$

Método DFP (Davindon, Fletcher y Powell)

$$H_{k+1} = H_k + \frac{p^k (p^k)^T}{(p^k)^T q^k} - \frac{H_k q^k (q^k)^T H_k}{(q^k)^T H_k q^k}$$

Método BFGS (Broyden, Fletcher, Goldfarb y Shanno)

$$H_{k+1} = H_k + \left(1 + \frac{(q^k)^T H_k q^k}{(p^k)^T q^k}\right) \frac{p^k (p^k)^T}{(p^k)^T q^k} - \frac{p^k (q^k)^T H_k + H_k q^k (p^k)^T}{(p^k)^T q^k}$$

Ejercicio 3 Implementar los algoritmos para DFP y BFGS y aplicarlos a la función de Rosenbrock y comparar ambos métodos.

Perfil de desempeño

Diseñado por Dolan y Moré en 2001 y es una herramienta que se emplea para comparar la aptitud de distintos algoritmos para resolver un conjunto de problemas.

Dados un conjunto de problemas P y un conjunto de algoritmos S, definimos como $c_{s,p}$ el costo de resolver el problema $p \in P$ con el algoritmo $s \in S$. Si el algoritmo s no puede resolver el problema p, se define $c_{s,p} = M$ con $M \in \mathbb{R}$ un valor adecuado de manera tal que:

$$c_{s,p} = M \Leftrightarrow \text{el algoritmo } s \text{ no resuelve el problema } p$$

Se asume que al menos un algoritmo resuelve el problema p. Definimos el índice de desempeño relativo como:

$$r_{s,p} = \frac{c_{s,p}}{\min\{c_{j,p} \colon j \in S\}}$$

Observar que r_s, $p \geq 1$ y, si $r_{s,p} = 1$, entonces es algoritmo s es uno de los mejores (o el mejor) para resolver el problema p. Mientras mayor sea $r_{s,p}$, peor es el desempeño de s al resolver p. Finalmente, definimos la función de desempeño $\rho_s : [1, +\infty) \to [0, 1]$ para el algoritmo s como:

$$\rho_s(\tau) = \frac{\#\{p \in P \colon r_{s,p} \leq \tau\}}{\#P}$$

Así, $\rho_s(\tau)$ es el porcentaje de problemas que el algoritmo s resuelve con hasta τ veces el costo del algoritmo más eficiente. Por ejemplo, $\rho_s(1)$ es la proporción de problemas que el algoritmo s resuelve con el menor costo.

Si se define:

$$r_{max} = \max_{s \in S, p \in P \,:\, c_{s,p} < M} r_{s,p}$$

se tiene que $\rho_s(r_{max})$ es el número de problemas resueltos por el algoritmo s. El valor $\rho_s(1)$ se denomina la eficiencia de s y $\rho_s(r_{max})$ es su robustez.

La definición del costo está relacionada a la medida de desempeño que se quiera estudiar. Se puede considerar como costo al tiempo que necesita el algoritmo para resolver el problema, a la cantidad de iteraciones, a la cantidad de veces que es evaluada la función objetivo, etc.

Para analizar el perfil de desempeño se suelen graficar en conjunto las funciones ρ_s para cada $s \in S$.

Ejercicio 4 Supongamos que tenemos cuatro algoritmos $S = \{A_0, A_1, A_2, A_3\}$ y queremos comparar su desempeño en base a seis problemas $P = \{P_0, P_1, P_2, P_3, P_4, P_5\}$. Consideraremos como costo $c_{s,p}$ a la cantidad de iteraciones que necesita el algoritmo s para resolver el problema p. Si no puede resolverlo, asignamos $c_{s,p} = 10^8$. Por "no puede resolverlo" entendemos que el algoritmo s supera el límite de iteraciones.

Obviamente, el primer paso consiste en ejecutar cada uno de los cuatro algoritmos sobre cada problema y registrar el costo (iteraciones) de cada uno. Así, podemos formar la matriz C de costos:

$$C = \begin{pmatrix} 23 & 45 & 12 & 54 \\ 56 & 34 & 67 & 11 \\ 10^8 & 10^8 & 15 & 10 \\ 10^8 & 10^8 & 120 & 10^8 \\ 19 & 56 & 10^8 & 37 \\ 10^8 & 111 & 56 & 10^8 \end{pmatrix}$$

Implementar una función que calcule la matriz R y grafique el desempeño de cada algoritmo.