Descenso Coordenado. Gradiente Proyectado Simple.

Alan Reyes-Figueroa

Métodos Numéricos II (Aula 25) 12.octubre.2023

Descenso Coordenado

Algoritmo: (*Descenso Coordenado*)
Inputs: $f : \mathbb{R}^n \to \mathbb{R}$ función de clase C^1, con gradiente $\nabla f^2 f$; $\mathbf{x}_0 \in \mathbb{R}^n$.
Outputs: $\mathbf{x}$ punto crítico de f.
Obtain n, the dimension of the domain.
For k = 0,1,2,... hasta que se cumpla un criterio de paro:
    Define $j = k \pmod{n}$ the coordinate to step j = k % n.
    Set $\mathbf{d}_k = \mathbf{e}_j$.
    Compute $-\nabla f(\mathbf{x}_k)[j] = \frac{\partial f}{\partial x_j}(\mathbf{x}_k)$.
    Set α_k by using any of Backtracking, Hessian or Cauchy strategies.
    Define

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k - \big(\alpha_k \nabla f(\mathbf{x}_k)[j]\big)\mathbf{e}_j \\ &= \mathbf{x}_k - \big(0, 0, \ldots, 0, \alpha_k \tfrac{\partial f}{\partial x_j}(\mathbf{x}_k), 0, \ldots, 0\big). \end{aligned}$$

Return $\mathbf{x}_{k+1}$.

Descenso Coordenado por Bloques

En ocasiones es conveniente dividir todas las n variables del dominio a optimizar en un conjunto de bloques

$$B_1, B_2, \ldots, B_r.$$

El **descenso coordenado por bloques** imita la idea del descenso coordenado, sólo que en lugar de moverse a lo largo de una dirección $\mathbf{e}_j$ en cada iteración, este nuevo método se mueve a lo largo de todas las direcciones del bloque B_j.

Por ejemplo, si el bloque B_j consiste de las variables $\mathbf{x}_1$, $\mathbf{x}_7$ y $\mathbf{x}_{13}$, entonces la actualización

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k) B_j$$

representa el ciclo

Define **temp** $= \mathbf{x}_k$.
For i in B_j : (for i in [1,7,13]:)
  temp $=$ **temp** $- \big(\alpha_k \nabla f(\mathbf{x}_k)[i]\big)\mathbf{e}_i$,
Set $\mathbf{x}_{k+1} =$ **temp**.

Descenso Coordenado por Bloques

Algoritmo: (*Descenso Coordenado por Bloques*)
Inputs: $f : \mathbb{R}^n \to \mathbb{R}$ función de clase C^1, con gradiente $\nabla f^2 f$; $\mathbf{x}_0 \in \mathbb{R}^n$; una lista de bloques $[B_1, B_2, \ldots, B_r]$ conformando una partición de las variables dominio.
Outputs: $\mathbf{x}$ punto crítico de f.

```
Obtain r, the number of blocks.
For k = 0,1,2,... hasta que se cumpla un criterio de paro:
    Define j = k (mod r) the coordinate Block to step j = k % r.
    Set d_k = B_j.
    Set α_k by using any of Backtracking, Hessian or Cauchy
strategies.
    Define temp = x_k.
    For i in B_j :
        temp = temp − (α_k ∇f(x_k)[i]) e_i ,
     Set x_{k+1} = temp.
Return x_{k+1}.
```

Gradiente Proyectado

Otro método sencillo, pero útil, se obtiene cuando queremos restringir los valores de nuestra solución de optimización a un subdominio rectangular de $\mathbb{R}^n$.

En este caso, supongamos que deseamos nuestra solución objetivo $\mathbf{x}$ en el rectángulo

$$[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n] \subseteq \mathbb{R}^n.$$

En otras palabras, queremos

$$a_i \leq x_i \leq b_i, \quad \text{para toda } i = 1, 2, \ldots, n. \tag{1}$$

El método de gradiente proyectado (simple) consiste en calcular la iteración del descenso gradiente usual

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{d}_k,$$

y luego obligar a que el nuevo punto $\mathbf{x}_{k+1}$ cumpla con todas las restricciones (1).

Gradiente Proyectado

En Python esto se logra fácilmente definiendo arrays

$$\begin{aligned} \mathbf{lo} &= \texttt{np.array}([a_1, a_2, \ldots, a_n]), \\ \mathbf{up} &= \texttt{np.array}([b_1, b_2, \ldots, b_n]), \end{aligned}$$

y haciendo

$$\begin{aligned} \mathbf{x}_{k+1}[\mathbf{x}_{k+1} < \mathbf{lo}] &= \mathbf{lo}[\mathbf{x}_{k+1} < \mathbf{lo}], \\ \mathbf{x}_{k+1}[\mathbf{x}_{k+1} > \mathbf{up}] &= \mathbf{up}[\mathbf{x}_{k+1} > \mathbf{up}]. \end{aligned}$$

Gradiente Proyectado

Algoritmo: (*Gradiente Proyectado*)
Inputs: $f : \mathbb{R}^n \to \mathbb{R}$ función de clase C^1, con gradiente $\nabla f^2 f$; $\mathbf{x}_0 \in \mathbb{R}^n$; $\mathbf{lo} = [a_1, \ldots, a_n]$ y $\mathbf{up} = [b_1, \ldots, b_n]$ los límites del dominio rectangular.
Outputs: $\mathbf{x}$ punto crítico de f.

```
For k = 0,1,2,... hasta que se cumpla un criterio de paro:
    Find d_k a descent direction.
    Compute α_k by using any of Backtracking or other method.
    Define
```

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{d}_k,$$

```
    Apply
```

$$\begin{aligned} \mathbf{x}_{k+1}[\mathbf{x}_{k+1} < \mathbf{lo}] &= \mathbf{lo}[\mathbf{x}_{k+1} < \mathbf{lo}], \\ \mathbf{x}_{k+1}[\mathbf{x}_{k+1} > \mathbf{up}] &= \mathbf{up}[\mathbf{x}_{k+1} > \mathbf{up}]. \end{aligned}$$

Return $\mathbf{x}_{k+1}$.