Método de descenso de gradiente

Juan F. Olivares Pacheco*

Índice

1. Fundamentos matemáticos

Para la comprensión y aplicación eficaces del **método de descenso de gradiente**, es indispensable una base sólida en **cálculo diferencial y análisis vectorial**. En esta unidad se revisan los conceptos teóricos fundamentales.

1.1. Repaso de cálculo diferencial

El cálculo diferencial es la rama de las matemáticas que se centra en el estudio de cómo las funciones varían en respuesta a pequeños cambios en sus variables. La

*jfolivar@uda.cl

derivada de una función $f(x)$ en un punto x_0, denotada como $f'(x_0)$, cuantifica la tasa de cambio instantánea de la función en dicho punto. Se define formalmente a través del límite:

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

Geométricamente, esta expresión representa la pendiente de la recta tangente a la gráfica de la función $f(x)$ en el punto x_0.

Cuando se trabaja con funciones de varias variables, el concepto análogo es el de las **derivadas parciales**. Estas miden la tasa de cambio de la función con respecto a una única variable, manteniendo las demás constantes. Para una función de dos variables $f(x, y)$, sus derivadas parciales se definen como:

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x + h, y) - f(x, y)}{h} \quad \text{y} \quad \frac{\partial f}{\partial y} = \lim_{h \to 0} \frac{f(x, y + h) - f(x, y)}{h}$$

A partir de las derivadas parciales se puede construir la **diferencia total** de una función $f(x_1, x_2, \ldots, x_n)$, la cual aproxima el cambio total en f como resultado de pequeños cambios simultáneos en todas sus variables. Se expresa como la suma ponderada de las diferencias de cada variable:

$$df = \frac{\partial f}{\partial x_1} \partial x_1 + \frac{\partial f}{\partial x_2} \partial x_2 + \cdots + \frac{\partial f}{\partial x_n} \partial x_n$$

Este concepto es crucial para entender cómo las variaciones infinitesimales en las variables de entrada afectan el valor de la función.

1.2. El gradiente y la matriz Hessiana

El **gradiente** generaliza el concepto de derivada a funciones escalares de múltiples variables y es una pieza central en la optimización numérica. Para una función $f : \mathbb{R}^n \to \mathbb{R}$, su gradiente, denotado como $\nabla f(\mathbf{x})$, es el vector compuesto por todas sus derivadas parciales:

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right]^\top$$

La interpretación geométrica del gradiente es fundamental: en cualquier punto $\mathbf{x}$, el vector gradiente apunta en la dirección de máximo incremento de la función

f. La magnitud o norma de este vector, $\|\nabla f(\mathbf{x})\|$, representa esta tasa máxima de cambio. Adicionalmente, el gradiente es siempre perpendicular a las curvas o superficies de nivel de la función en ese punto. En el contexto de la optimización, para minimizar una función se debe avanzar en la dirección opuesta al gradiente, es decir, la dirección de máximo descenso. Los puntos donde el gradiente se anula, $\nabla f(\mathbf{x}) = 0$, se conocen como puntos críticos y son candidatos a ser mínimos, máximos o puntos de silla.

Para caracterizar la naturaleza de estos puntos críticos, se utiliza la **matriz Hessiana** $\mathbf{H}_f(\mathbf{x})$, que es la matriz de las segundas derivadas parciales de la función f:

$$\mathbf{H}_f(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Algunas propiedades de la matriz Hessiana son:

- **Simetría:** Si las segundas derivadas son continuas, la matriz Hessiana es simétrica.
- **Positividad definida:**
  - Si $\mathbf{H}_f(\mathbf{x})$ es **positiva definida**, f tiene un mínimo local en $\mathbf{x}$.
  - Si $\mathbf{H}_f(\mathbf{x})$ es **negativa definida**, f tiene un máximo local en $\mathbf{x}$.
  - Si $\mathbf{H}_f(\mathbf{x})$ no es definida positiva ni negativa, $\mathbf{x}$ es un punto silla.

2. El método de descenso de gradiente

El **método de descenso de gradiente** es un algoritmo iterativo fundamental para encontrar el mínimo de una función diferenciable $f(\mathbf{x})$, con $\mathbf{x} \in \mathbb{R}^n$. Su principio operativo se basa en moverse desde un punto inicial en la dirección que produce la disminución más rápida del valor de la función.

2.1. Descripción del algoritmo

La idea central del método es que, en cualquier punto $\mathbf{x}$ del dominio de la función, el gradiente $\nabla f(\mathbf{x})$ señala la dirección de máximo crecimiento. Por consiguiente, la dirección opuesta, $-\nabla f(\mathbf{x})$, corresponde a la dirección de descenso más pronunciado. El algoritmo explota esta propiedad para construir una secuencia de puntos $\{\mathbf{x}_k\}$ que converge hacia un mínimo.

El proceso se inicia con una estimación inicial $\mathbf{x}_0$ y genera las siguientes aproximaciones mediante la regla de actualización:

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

En esta ecuación, el hiperparámetro $\alpha > 0$, conocido como **tasa de aprendizaje (learning rate)**, es un escalar que modula la longitud del paso en cada iteración. De esta forma, cada nuevo punto $\mathbf{x}_{k+1}$ se obtiene desplazando el punto actual $\mathbf{x}_k$ en la dirección opuesta al gradiente.

En consecuencia, el algoritmo puede definirse de la siguiente manera:

1. **Inicialización:** Elegir un punto inicial $\mathbf{x}_0$, es decir, se comienza con una suposición inicial para los valores de las variables que queremos optimizar.
2. **Iteración:** Para $k = 0, 1, 2, \ldots$:
   - Calcular el gradiente $\nabla f(\mathbf{x}_k)$.
   - Actualizar el punto:

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

   - Verificar algún criterio de convergencia.
3. Se finalizan las iteraciones si se cumple algún criterio de convergencia. Si es así, se retorna $\mathbf{x}_{k+1}$ como solución aproximada.

Se dice que el algoritmo converge cuando se aproxima suficientemente al mínimo de la función, es decir, cuando los cambios en los valores de las variables entre iteraciones se vuelven insignificantes. Dos criterios de convergencia generalmente utilizados, son:

1. $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \epsilon$
2. $\|\nabla f(\mathbf{x}_{k+1})\| < \epsilon$

Por otro lado, la elección de la **tasa de aprendizaje** es crucial en el método de descenso de gradiente. Si la tasa de aprendizaje es demasiado grande, el algoritmo puede oscilar alrededor del mínimo o incluso diverger. Si es demasiado pequeña, puede converger muy lentamente.

Es importante tener en cuenta que el método de descenso de gradiente no garantiza encontrar el mínimo global en funciones no convexas, ya que puede quedar atrapado en mínimos locales o puntos silla. Sin embargo, es ampliamente utilizado debido a su simplicidad y eficiencia en una variedad de problemas de optimización.

2.2. Ejemplos de implementación en R

2.2.1. Función cuadrática unidimensional

Se busca el mínimo de la función:

$$f(x) = x^2,\ x \in \mathbb{R}$$

El gradiente es $\nabla f(x) = f'(x) = 2x$.

```r
# Definición de la función y su gradiente
f <- function(x) {
  return(x^2)
}

grad_f <- function(x) {
  return(2 * x)
}

# Ajuste de parámetros del algoritmo
alpha <- 0.1
max_iter <- 1000
tol <- 1e-6

# Definición del punto inicial
x_actual <- 5

# Iteración del método
for (i in 1:max_iter) {
  x_nuevo <- x_actual - alpha * grad_f(x_actual)
  if (norm(grad_f(x_nuevo), "2") < tol) {
```

```
    break
  }
  x_actual <- x_nuevo
}

# Presentación de resultados
resultados <- list(minimo = f(x_nuevo),
                   objetivo = x_nuevo,
                   iteraciones = i)
print(resultados)
```

```
## $minimo
## [1] 1.774509e-13
##
## $objetivo
## [1] 4.212492e-07
##
## $iteraciones
## [1] 73
```

2.2.2. Función cuadrática desplazada

Se busca el mínimo de la función:

$$f(x) = (x-2)^2$$

El gradiente es $\nabla f(x) = f'(x) = 2(x-2)$.

```
# Definimos la función y su gradiente
f <- function(x) {
  return((x - 2)^2)
}

grad_f <- function(x) {
  return(2 * (x - 2))
}

# Ajuste de parámetros
alpha <- 0.1
max_iter <- 1000
```

```r
tol <- 1e-6

# Punto inicial
x_actual <- 10

# Algoritmo de descenso de gradiente
for (i in 1:max_iter) {
    x_nuevo <- x_actual - alpha * grad_f(x_actual)
    if (norm(grad_f(x_nuevo), "2") < tol) {
        break
    }
    x_actual <- x_nuevo
}

# Presentación de resultados
resultados <- list(minimo = f(x_nuevo),
                   objetivo = x_nuevo,
                   iteraciones = i)
print(resultados)
```

```
## $minimo
## [1] 1.860707e-13
##
## $objetivo
## [1] 2
##
## $iteraciones
## [1] 75
```

2.2.3. Función cuadrática bidimensional

Se busca el mínimo de la función:

$$f(x, y) = (x - 2)^2 + (y + 3)^2$$

El gradiente es:

$$\nabla f(x, y) = [2(x - 2),\ 2(y + 3)]^\top$$

```r
# Definimos la función y su gradiente
f <- function(x) {
  return((x[1] - 2)^2 + (x[2] + 3)^2)
}

grad_f <- function(x) {
  return(c(2 * (x[1] - 2),
           2 * (x[2] + 3)))
}

# Ajuste de parámetros
alpha <- 0.1
max_iter <- 1000
tol<- 1e-6

# Punto inicial
x_actual <- c(-5, 5)

# Algoritmo de descenso de gradiente
for (i in 1:max_iter) {
  x_nuevo <- x_actual - alpha * grad_f(x_actual)
  if (norm(grad_f(x_nuevo), "2") < tol) {
    break
  }
  x_actual <- x_nuevo
}

# Presentación de resultados
resultados <- list(minimo = f(x_nuevo),
                   objetivo = x_nuevo,
                   iteraciones = i)
print(resultados)
```

```
## $minimo
## [1] 2.102599e-13
##
## $objetivo
## [1]  2 -3
##
## $iteraciones
```

```
## [1] 76
```

3. Valor optimo de la tasa de aprendizaje

Determinar la tasa de aprendizaje adecuada es crucial para el éxito del método de descenso de gradiente. Determinar el valor óptimo de α es esencial para asegurar una convergencia eficiente y estable hacia el mínimo de la función.

La importancia en la elección óptima de la tasa de aprendizaje es:

- **Tasa de aprendizaje alta:** Puede acelerar la convergencia inicialmente, pero corre el riesgo de saltar el mínimo o incluso causar divergencia del algoritmo.
- **Tasa de aprendizaje baja:** Garantiza movimientos pequeños y estables, pero puede resultar en una convergencia muy lenta, aumentando el tiempo de cómputo.

Encontrar un equilibrio adecuado es crucial para el rendimiento del algoritmo. Existen diferentes métodos para determinar el valor óptimo de la tasa de aprendizaje, de los que podemos destacar:

1. **Prueba y error:** Se puede comenzar con una tasa de aprendizaje pequeña y ajustarla gradualmente. Se puede observar así como afecta la convergencia. Si el algoritmo oscila o no converge, se reduce la tasa de aprendizaje. Si converge muy lentamente, aumentamos la tasa de aprendizaje.
2. **Búsqueda en red:** Se prueba una serie de tasas de aprendizaje en una cuadrícula de valores posibles y elige la mejor. Por ejemplo, se puede probar tasas de aprendizaje como 10^{-3}, 10^{-2}, 10^{-1}, 1, 10.

Supongamos que queremos experimentar con diferentes tasas de aprendizajes para la función dada a continuación y usaremos el enfoque de prueba y error.

$$f(x) = (x-2)^2$$

```
# Definimos la función
f <- function(x) {
  return((x - 2)^2)
}

# Definimos el gradiente de la función
```

```r
grad_f <- function(x) {
  return(2 * (x - 2))
}

# Parámetros del algoritmo
tol <- 1e-6
max_iter <- 1000

# Vector con valores de alpha
alpha <- c(0.01, 0.10, 0.15, 0.25, 0.50)

# Almacenamos los resultados
resultados <- data.frame(alpha,
                         x_optimo = numeric(length(alpha)),
                         iteracion = numeric(length(alpha)))

# Algoritmo de descenso de gradiente
for (k in 1:length(alpha)) {
  x_actual <- 10
  for(i in 1:max_iter) {
    x_nuevo <- x_actual - alpha[k] * grad_f(x_actual)
    if (norm(grad_f(x_nuevo), "2") < tol) {
      resultados[k, 2] <- x_nuevo
      resultados[k, 3] <- i
      break
    }
    x_actual <- x_nuevo
  }
}

# Mostramos los resultados
print(resultados)
```

```
##   alpha x_optimo iteracion
## 1  0.01        2       822
## 2  0.10        2        75
## 3  0.15        2        47
## 4  0.25        2        24
## 5  0.50        2         1
```

Otros métodos ajustan automáticamente la tasa de aprendizaje durante el proceso de optimización. Por ejemplo, el **algoritmo con decaimiento de tasa de aprendizaje**, reduce la tasa de aprendizaje a medida que avanza el entrenamiento, por ejemplo:

$$\alpha_k = \frac{\alpha_0}{1 + k\tau}$$

donde α_0 es la tasa inicial, k es el número de iteración y τ es una constante de decaimiento.

Supongamos que queremos encontrar el **mínimo** de la siguiente función:

$$f(x, y) = (x - 2)^2 + (y + 3)^2$$

```
# Definimos la función
f <- function(x) {
  return((x[1] - 2)^2 + (x[2] + 3)^2)
}

# Definimos el gradiente de la función
grad_f <- function(x) {
  return(c(2 * (x[1] - 2),
           2 * (x[2] + 3)))
}

# Parámetros del algoritmo
max_iter <- 1000
tol<- 1e-6

# Tasa de aprendizaje inicial
alpha_0 <- 1

# Constante de decaimiento
tau <- 0.01

# Punto inicial
x_actual <- c(-5, 5)

# Algoritmo de descenso de gradiente
```

```r
for (i in 1:max_iter) {
  alpha_k <- alpha_0 / (1 + i * tau)
  x_nuevo <- x_actual - alpha_k * grad_f(x_actual)
  if (norm(grad_f(x_nuevo), "2") < tol) {
    break
  }
  x_actual <- x_nuevo
}

# Mostramos los resultados
resultados <- list(minimo = f(x_nuevo),
                   objetivo = x_nuevo,
                   iteraciones = i)
print(resultados)
```

```
## $minimo
## [1] 2.459381e-13
##
## $objetivo
## [1]  2 -3
##
## $iteraciones
## [1] 40
```

4. Ejercicios

1. Implemente el descenso de gradiente para minimizar la función:

$$f(x) = 3x + 4$$

   Y analice por qué el método no es necesario en este caso.

2. Minimice la función:

$$f(x) = (x - 5)^2$$

   Y encuentre el valor óptimo de x y verifique con la solución analítica.

3. Minimice la función:

$$f(x) = (x-2)^3$$

Observe y analice el comportamiento del gradiente en diferentes puntos.

4. Minimice la función:

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

Encuentre los valores óptimos de x y y.

5. Minimice la función de Rosenbrok:

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

Analice cómo cambia la trayectoria del descenso de gradiente con diferentes tasas de aprendizaje.

6. Minimice la función:

$$f(x,y) = \sin(x) + \cos(y)$$

Discute las dificultades que surgen debido a los múltiples mínimos locales.

7. Minimice la función:

$$f(x,y,z) = x^2 + y^2 + z^2$$

Encuentre el valor óptimo y verifique la convergencia del método.

8. Minimice la función de Beale.

$$f(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$$

Analice la complejidad de la superficie de error y discuta la convergencia.

9. Minimice la función Himmelblau:

$$f(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$$

Además, identifique los múltiples mínimos y discuta cómo elegir buenos puntos iniciales.

10. Minimice la función:

$$f(x, y, z) = (x - 1)^2 + (y - 2)^2 + (z - 3)^2 + \sin(x)\cos(y)\exp(z)$$

Experimente con diferentes tasas de aprendizaje y puntos iniciales, y analice los resultados.