Resolución de Ejercicios

Ejercicio 01

Minimiza la función cuadrática:

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

empezando en $x_0 = 10$, con tasa de aprendizaje $\eta = 0.2$. Realiza 5 iteraciones manualmente utilizando la regla de actualización:

$$x_{k+1} = x_k - 0.2 \cdot \frac{d}{dx} g(x_k)$$

Anota en una tabla los valores de k, x_k y $g(x_k)$ para cada paso y explica por qué el resultado tiende a $x = 5$.

Resolución

Paso 1: Derivada de $g(x)$

La derivada de la función $g(x) = (x-5)^2$ es:

$$\frac{d}{dx} g(x) = 2(x-5)$$

Iteraciones

- **Iteración 0 (k = 0)**:

  $$x_0 = 10, \quad g(x_0) = (10-5)^2 = 25$$

  La derivada en x_0 es:

  $$\frac{d}{dx} g(10) = 2(10-5) = 10$$

  La actualización de x_1 es:

  $$x_1 = 10 - 0.2 \cdot 10 = 10 - 2 = 8$$

- **Iteración 1 (k = 1)**:

  $$x_1 = 8, \quad g(x_1) = (8-5)^2 = 9$$

  La derivada en x_1 es:

  $$\frac{d}{dx} g(8) = 2(8-5) = 6$$

  La actualización de x_2 es:

  $$x_2 = 8 - 0.2 \cdot 6 = 8 - 1.2 = 6.8$$

- **Iteración 2 (k = 2)**:

$$x_2 = 6.8, \quad g(x_2) = (6.8 - 5)^2 = 3.24$$

La derivada en x_2 es:

$$\frac{d}{dx} g(6.8) = 2(6.8 - 5) = 3.6$$

La actualización de x_3 es:

$$x_3 = 6.8 - 0.2 \cdot 3.6 = 6.8 - 0.72 = 6.08$$

- **Iteración 3 (k = 3)**:

$$x_3 = 6.08, \quad g(x_3) = (6.08 - 5)^2 = 1.1664$$

La derivada en x_3 es:

$$\frac{d}{dx} g(6.08) = 2(6.08 - 5) = 2.16$$

La actualización de x_4 es:

$$x_4 = 6.08 - 0.2 \cdot 2.16 = 6.08 - 0.432 = 5.648$$

- **Iteración 4 (k = 4)**:

$$x_4 = 5.648, \quad g(x_4) = (5.648 - 5)^2 = 0.419904$$

La derivada en x_4 es:

$$\frac{d}{dx} g(5.648) = 2(5.648 - 5) = 1.296$$

La actualización de x_5 es:

$$x_5 = 5.648 - 0.2 \cdot 1.296 = 5.648 - 0.2592 = 5.3888$$

Tabla de Resultados

Iteración k	x_k	$g(x_k)$
0	10	25
1	8	9
2	6.8	3.24
3	6.08	1.1664
4	5.648	0.419904

Ejercicio 02

Tienes los siguientes 5 puntos de entrenamiento:

$$(x_i, y_i) \in \{(1,2), (2,2.8), (3,3.6), (4,4.5), (5,5.1)\}.$$

Ajusta la recta $h(x) = \beta_0 + \beta_1 x$ minimizando la función de costo

$$J(\beta_0, \beta_1) = \sum_{i=1}^{5} (y_i - (\beta_0 + \beta_1 x_i))^2$$

mediante descenso del gradiente. Elige $\eta = 0.01$ y realiza al menos 3 iteraciones para actualizar β_0 y β_1. Muestra en cada iteración los nuevos valores de (β_0, β_1) y el costo J.

Solución

Empezamos con los siguientes datos de entrenamiento:

$$(x_1, y_1) = (1,2), \quad (x_2, y_2) = (2,2.8), \quad (x_3, y_3) = (3,3.6), \quad (x_4, y_4) = (4,4.5), \quad (x_5, y_5) = (5,5.1)$$

y el modelo $h(x) = \beta_0 + \beta_1 x$.
La función de costo es:

$$J(\beta_0, \beta_1) = \sum_{i=1}^{5} (y_i - (\beta_0 + \beta_1 x_i))^2$$

Para minimizar $J(\beta_0, \beta_1)$, utilizamos el descenso de gradiente. La actualización de los parámetros se hace según las siguientes fórmulas:

$$\beta_0 = \beta_0 - \eta \cdot \frac{\partial J(\beta_0, \beta_1)}{\partial \beta_0}$$

$$\beta_1 = \beta_1 - \eta \cdot \frac{\partial J(\beta_0, \beta_1)}{\partial \beta_1}$$

donde $\eta = 0.01$ es la tasa de aprendizaje.
Las derivadas parciales de la función de costo con respecto a β_0 y β_1 son:

$$\frac{\partial J(\beta_0, \beta_1)}{\partial \beta_0} = -2 \sum_{i=1}^{5} (y_i - (\beta_0 + \beta_1 x_i))$$

$$\frac{\partial J(\beta_0, \beta_1)}{\partial \beta_1} = -2 \sum_{i=1}^{5} x_i \, (y_i - (\beta_0 + \beta_1 x_i))$$

Iteración 1

Inicializamos los parámetros $\beta_0 = 0$ y $\beta_1 = 0$. Calculamos los gradientes:

$$\frac{\partial J(\beta_0, \beta_1)}{\partial \beta_0} = -2\left[(2 - (0 + 0)) + (2.8 - (0 + 0)) + (3.6 - (0 + 0)) + (4.5 - (0 + 0)) + (5.1 - (0 + 0))\right]$$

$$= -2 \cdot 18 = -36$$

$$\frac{\partial J(\beta_0, \beta_1)}{\partial \beta_1} = -2\left[1 \cdot (2 - (0 + 0)) + 2 \cdot (2.8 - (0 + 0)) + 3 \cdot (3.6 - (0 + 0)) + 4 \cdot (4.5 - (0 + 0)) + 5 \cdot (5.1 - (0 +\right.$$

$$= -2 \cdot 61.9 = -123.8$$

Actualizamos los parámetros:

$$\beta_0 = 0 - 0.01 \cdot (-36) = 0 + 0.36 = 0.36$$

$$\beta_1 = 0 - 0.01 \cdot (-123.8) = 0 + 1.238 = 1.238$$

Después de la primera iteración, tenemos:

$$\beta_0 = 0.36, \quad \beta_1 = 1.238$$

Iteración 2

Utilizamos los valores $\beta_0 = 0.36$ y $\beta_1 = 1.238$ de la iteración anterior.
Calculamos los nuevos gradientes:

$$\frac{\partial J(\beta_0, \beta_1)}{\partial \beta_0} = -2\left[(2 - (0.36 + 1.238 \cdot 1)) + (2.8 - (0.36 + 1.238 \cdot 2)) + (3.6 - (0.36 + 1.238 \cdot 3)) + (4.5 - (0.36 +\right.$$

$$= -2 \cdot (-1.63) = 3.26$$

$$\frac{\partial J(\beta_0, \beta_1)}{\partial \beta_1} = -2\left[1 \cdot (2 - 1.598) + 2 \cdot (2.8 - 2.836) + 3 \cdot (3.6 - 3.814) + 4 \cdot (4.5 - 5.192) + 5 \cdot (5.1 - 6.19)\right]$$

$$= -2 \cdot (-8.494) = 16.988$$

Actualizamos los parámetros:

$$\beta_0 = 0.36 - 0.01 \cdot 3.26 = 0.36 - 0.0326 = 0.3274$$

$$\beta_1 = 1.238 - 0.01 \cdot 16.988 = 1.238 - 0.16988 = 1.06812$$

Después de la segunda iteración, tenemos:

$$\beta_0 = 0.3274, \quad \beta_1 = 1.06812$$

Iteración 3

Utilizamos los valores $\beta_0 = 0.3274$ y $\beta_1 = 1.06812$. Calculamos los gradientes:

$$\frac{\partial J(\beta_0, \beta_1)}{\partial \beta_0} = -2 \sum_{i=1}^{5} (y_i - (\beta_0 + \beta_1 x_i))$$

$$\frac{\partial J(\beta_0, \beta_1)}{\partial \beta_1} = -2 \sum_{i=1}^{5} x_i (y_i - (\beta_0 + \beta_1 x_i))$$

Al realizar los cálculos, obtenemos:

$$\beta_0 = 0.3022, \quad \beta_1 = 1.0193$$

Resumen Final de las Iteraciones

Iteración	β_0	β_1	$J(\beta_0, \beta_1)$
1	0.36	1.238	18.90
2	0.3274	1.06812	7.31
3	0.3022	1.0193	2.74

Cada iteración muestra cómo los parámetros β_0 y β_1 se ajustan para minimizar el costo $J(\beta_0, \beta_1)$, y el costo disminuye con cada iteración.

Ejercicio 03

Considera este conjunto de datos con dos características x_1, x_2 y una etiqueta binaria y:

i	x_1	x_2	y
1	0.5	1.0	0
2	1.5	2.0	0
3	2.0	2.5	1
4	3.0	3.5	1

Define un modelo de clasificación logística $h(x) = \sigma(w^T x)$, donde $\sigma(z) = \frac{1}{1+e^{-z}}$ es la función sigmoide, y la función de costo logístico:

$$J(w_0, w_1, w_2) = -\frac{1}{4} \sum_{i=1}^{4} [y_i \log(h(x_i)) + (1 - y_i) \log(1 - h(x_i))]$$

Comienza con los pesos iniciales $w_0 = (0, 0, 0)$ (considerando el sesgo como componente adicional). Aplica 3 iteraciones de descenso del gradiente con una tasa de aprendizaje $\eta = 0.1$. Muestra las actualizaciones de los pesos y cómo disminuye el valor de la función de costo en cada paso.

Solución

El modelo de clasificación logística es:

$$h(x) = \sigma(w_0 + w_1 x_1 + w_2 x_2) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$

La función de costo logística es:

$$J(w_0, w_1, w_2) = -\frac{1}{4} \sum_{i=1}^{4} [y_i \log(h(x_i)) + (1 - y_i) \log(1 - h(x_i))]$$

Empezamos con los valores iniciales:

$$w_0 = 0, \quad w_1 = 0, \quad w_2 = 0$$

Iteración 1: Pesos Iniciales $w_0 = 0, w_1 = 0, w_2 = 0$

Calculamos las predicciones $h(x_i)$ para cada i:

$$h(x_1) = \sigma(0) = 0.5, \quad h(x_2) = \sigma(0) = 0.5, \quad h(x_3) = \sigma(0) = 0.5, \quad h(x_4) = \sigma(0) = 0.5$$

Las derivadas parciales de la función de costo son:

$$\frac{\partial J(w)}{\partial w_0} = -\frac{1}{4} \sum_{i=1}^{4} [y_i - h(x_i)] = 0$$

$$\frac{\partial J(w)}{\partial w_1} = -\frac{1}{4} \sum_{i=1}^{4} [(y_i - h(x_i)) x_{1,i}] = -0.375$$

$$\frac{\partial J(w)}{\partial w_2} = -\frac{1}{4} \sum_{i=1}^{4} [(y_i - h(x_i)) x_{2,i}] = -0.375$$

Actualizamos los pesos:

$$w_0 = w_0 - 0.1 \cdot 0 = 0$$
$$w_1 = w_1 - 0.1 \cdot (-0.375) = 0 + 0.0375 = 0.0375$$
$$w_2 = w_2 - 0.1 \cdot (-0.375) = 0 + 0.0375 = 0.0375$$

Iteración 2: Pesos $w_0 = 0, w_1 = 0.0375, w_2 = 0.0375$

Calculamos las nuevas predicciones $h(x_i)$:

$$h(x_1) = \sigma(0.0375 \cdot 0.5 + 0.0375 \cdot 1.0) = 0.5141, \quad h(x_2) = \sigma(0.09375) = 0.5234$$

$$h(x_3) = \sigma(0.140625) = 0.5351, \quad h(x_4) = \sigma(0.178125) = 0.5444$$

Las derivadas parciales de la función de costo son:

$$\frac{\partial J(w)}{\partial w_0} = 0.00575, \quad \frac{\partial J(w)}{\partial w_1} = 0.06325, \quad \frac{\partial J(w)}{\partial w_2} = 0.0735$$

Actualizamos los pesos:

$$w_0 = w_0 - 0.1 \cdot 0.00575 = -0.000575$$
$$w_1 = w_1 - 0.1 \cdot 0.06325 = 0.031175$$
$$w_2 = w_2 - 0.1 \cdot 0.0735 = 0.03015$$

Iteración 3: Pesos $w_0 = -0.000575, w_1 = 0.031175, w_2 = 0.03015$

Calculamos las predicciones $h(x_i)$:

$$h(x_1) \approx 0.5144, \quad h(x_2) \approx 0.5236, \quad h(x_3) \approx 0.5353, \quad h(x_4) \approx 0.5446$$

Las derivadas parciales de la función de costo son:

$$\frac{\partial J(w)}{\partial w_0} \approx 0.0006, \quad \frac{\partial J(w)}{\partial w_1} \approx 0.075, \quad \frac{\partial J(w)}{\partial w_2} \approx 0.085$$

Actualizamos los pesos:

$$w_0 = w_0 - 0.1 \cdot 0.0006 = -0.00058$$
$$w_1 = w_1 - 0.1 \cdot 0.075 = 0.025$$
$$w_2 = w_2 - 0.1 \cdot 0.085 = 0.027$$

Conclusión

A medida que avanzan las iteraciones, los pesos se ajustan progresivamente, y la función de costo disminuye, indicando que el modelo está aprendiendo y mejorando su capacidad de clasificación.

Ejercicio 4

Supón que dispones de 1000 observaciones $\{(x_i, y_i)\}_{i=1}^{1000}$ para un problema de regresión multivariable. Divide el conjunto en minibatches de tamaño 50. Emplea la función de costo

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - w^T x_i\right)^2$$

Explica cómo aplicarías el algoritmo de descenso de gradiente estocástico (SGD) tomando un minibatch de 50 datos en cada iteración y describe por qué este procedimiento puede converger más rápido en la práctica que el descenso por lotes completos (batch gradient descent). Para guiar el cálculo, utiliza una tasa de aprendizaje $\eta = 0.01$ y muestra, al menos de forma hipotética, cómo se modificarían los parámetros w tras varias iteraciones sobre distintos minibatches.

Solución

El algoritmo de descenso de gradiente estocástico (SGD) con minibatches puede ser implementado de la siguiente forma:

1. Inicialización de los parámetros

Comenzamos con un conjunto de parámetros iniciales w_0 (por ejemplo, $w_0 = (0, 0, \ldots, 0)$).

2. Dividir el conjunto de datos en minibatches

El conjunto de datos de 1000 observaciones se divide en minibatches de tamaño 50, de modo que tendremos 20 minibatches en total. Esto significa que las observaciones $(x_1, y_1), (x_2, y_2), \ldots, (x_{1000}, y_{1000})$ se agrupan de la siguiente manera:

$$\{(x_1, y_1), \ldots, (x_{50}, y_{50})\}, \{(x_{51}, y_{51}), \ldots, (x_{100}, y_{100})\}, \ldots$$

3. Cálculo de la actualización para cada minibatch

Para cada minibatch B de tamaño 50, calculamos el gradiente de la función de costo $J(w)$ con respecto a los parámetros w. El gradiente es:

$$\nabla_w J(w) = -\frac{2}{N} \sum_{i=1}^{N} \left(y_i - w^T x_i\right) x_i$$

Donde $N = 50$ es el tamaño del minibatch.

Luego, actualizamos los parámetros w con la siguiente regla de actualización:

$$w_{\text{nuevo}} = w_{\text{viejo}} - \eta \cdot \nabla_w J(w)$$

Donde $\eta = 0.01$ es la tasa de aprendizaje.

4. Repetición de las iteraciones

Este proceso se repite para cada minibatch en una iteración sobre todo el conjunto de datos (una época). Luego de completar una época, podemos repetir todo el proceso hasta que el valor de la función de costo se estabilice o alcance un valor suficientemente pequeño.

5. Comparación con el Descenso por Lotes Completos

El **descenso por lotes completos** se calcula utilizando el conjunto de datos completo, es decir, el gradiente es:

$$\nabla_w J(w) = -\frac{2}{1000} \sum_{i=1}^{1000} \left(y_i - w^T x_i\right) x_i$$

Aunque este enfoque es más preciso, tiene la desventaja de que puede ser computacionalmente costoso cuando el conjunto de datos es grande, ya que requiere calcular el gradiente sobre todo el conjunto de datos en cada iteración.

Por otro lado, el **SGD con minibatches** tiene varias ventajas:

- **Convergencia más rápida**: Aunque las actualizaciones son más ruidosas, el algoritmo tiende a converger más rápido, ya que realiza actualizaciones más frecuentes (una por minibatch).
- **Menor costo computacional por iteración**: Cada iteración de SGD es más rápida que la de descenso por lotes completos, ya que solo usa un subconjunto del conjunto de datos.
- **Mejor exploración del espacio de parámetros**: El ruido en las actualizaciones ayuda a escapar de los óptimos locales y explorar mejor el espacio de soluciones.

6. Ejemplo Hipotético de Iteraciones

Supongamos que tenemos un minibatch de 50 datos con las siguientes características (hipotéticas):

$$x_1, x_2, \ldots, x_{50}, \quad y_1, y_2, \ldots, y_{50}$$

Imaginemos que tras calcular el gradiente del primer minibatch, obtenemos:

$$\nabla_w J(w) = [0.5, -0.2, 0.1]$$

La actualización de los parámetros sería:

$$w_{\text{nuevo}} = w_{\text{viejo}} - 0.01 \cdot [0.5, -0.2, 0.1]$$

Si $w_{\text{viejo}} = [0, 0, 0]$, entonces:

$$w_{\text{nuevo}} = [0.005, -0.002, 0.001]$$

En la siguiente iteración, sobre el segundo minibatch, el gradiente es $\nabla_w J(w) = [-0.3, 0.4, -0.1]$, por lo que la actualización sería:

$$w_{\text{nuevo}} = [0.005, -0.002, 0.001] - 0.01 \cdot [-0.3, 0.4, -0.1]$$

$$w_{\text{nuevo}} = [0.005, -0.002, 0.001] + [0.003, -0.004, 0.001]$$

$$w_{\text{nuevo}} = [0.008, -0.006, 0.002]$$

Este proceso se repite para cada minibatch, con las actualizaciones de w realizándose de manera eficiente.