

Journal Name

Crossmark

TAREA 5

RECEIVED
dd Month yyyyREVISED
dd Month yyyy

Descenso de Gradiente de n-Dimensiones

Daniel González¹ ¹Facultad de Ingeniería, Universidad Autónoma de Querétaro, Querétaro, México
08/09/2025**E-mail:** dgonzalez16@alumnos.uaq.mx**Keywords:** descenso gradiente, plano, mínimo, espacio, épocas, n-dimensiones**Abstract**

El descenso de gradiente es un método iterativo de optimización que utiliza la gradiente para aproximarse al mínimo de una función de costo. En el caso de n características, la gradiente se expresa como un vector de derivadas parciales, donde cada componente representa la dirección de mayor pendiente respecto a una variable. Al avanzar en dirección contraria de ese vector, el algoritmo actualiza los parámetros paso a paso según la regla

$$\theta_j := \theta_j - \alpha \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j}, \quad j = 1, 2, \dots, n,$$

donde α es la tasa de aprendizaje. De esta manera, se logra descender progresivamente sobre la superficie multidimensional de la función hasta aproximarse a un mínimo local o global.

1 Introducción

En el aprendizaje automático supervisado, el objetivo es encontrar una función que modele la relación entre un conjunto de características de entrada y una variable de salida. Para ilustrar el proceso, a menudo se comienza con el caso más simple, en el cual el modelo depende de una sola característica. Sin embargo, los problemas reales requieren generalizar a múltiples características, lo que hace necesaria una notación más compacta y eficiente.

Con este propósito, los parámetros del modelo se agrupan en un vector columna

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix},$$

mientras que las características de los m ejemplos de entrenamiento se organizan en una matriz

$$X = \begin{bmatrix} x_1^0 & x_1^1 & \cdots & x_1^n \\ x_2^0 & x_2^1 & \cdots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ x_m^0 & x_m^1 & \cdots & x_m^n \end{bmatrix},$$

donde $x_i^0 = 1$ para todo $i = 1, \dots, m$, lo que permite incorporar el término de sesgo en el modelo.

Bajo esta formulación, la hipótesis se expresa como

$$h_{\boldsymbol{\theta}} = X \cdot \boldsymbol{\theta},$$

que produce un vector de dimensión $m \times 1$ con las predicciones correspondientes a cada ejemplo de entrenamiento.

Para evaluar la calidad de dichas predicciones se emplea la función de costo basada en el error cuadrático medio (MSE), definida como

$$J(\boldsymbol{\theta}) = \frac{1}{2m} \sum_{i=1}^m (h_{\boldsymbol{\theta}}(x_i) - y_i)^2,$$

la cual, en notación vectorial, puede escribirse como

$$J(\theta) = \frac{1}{2m}(X \cdot \theta - Y)^T(X \cdot \theta - Y),$$

donde

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}.$$

El siguiente paso consiste en optimizar esta función de costo para encontrar los parámetros que mejor se ajustan a los datos. Para ello se utiliza el algoritmo de gradiente descendente, que actualiza los parámetros de manera iterativa siguiendo la regla

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta), \quad j = 0, 1, \dots, n,$$

donde α representa la tasa de aprendizaje. El proceso se repite hasta alcanzar la convergencia o completar un número predeterminado de iteraciones, permitiendo aproximarse al mínimo de la función de costo y, por tanto, al modelo más adecuado.

2 Desarrollo



```

1 def grad_multivariada(X, y, alpha=0.01, epochs=2000, return_history=False):
2     """
3     X: matriz de características (m muestras x n variables)
4     y: vector de salida (m)
5     """
6     X = np.asarray(X, dtype=float)
7     y = np.asarray(y, dtype=float).reshape(-1, 1)
8     m, n = X.shape
9
10    # Inicializar theta (n x 1)
11    theta = np.zeros((n, 1))
12    hist = {"theta": [], "J": []}
13
14    for _ in range(epochs):
15        y_pred = X @ theta
16        error = y_pred - y
17        grad = (1/m) * (X.T @ error)
18        theta -= alpha * grad
19
20        if return_history:
21            J = (1/(2*m)) * np.sum(error**2)
22            hist["theta"].append(theta.copy())
23            hist["J"].append(J)
24
25    if return_history:
26        return theta, hist
27    return theta

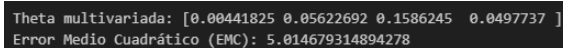
```

Figure 1. Funcion para N dimensiones

A partir de la función propuesta en la figura 1, tomaremos nuestra DB propuesta en problemas anteriores (Advertising.csv) y usaremos como características las 3 columnas de ventas, es decir, TV, Radio y Newspaper. Recordemos que se crea nuestra matriz de características $X + (Columnade1s)$ y nuestro vector de etiquetas Y el cual serán nuestras ventas (“Sales”).

Agregaremos también la opción de guardar el historico para poder graficar la evolución de los parámetros y la función de costo. Además de nuestro Error cuadrático medio (MSE) que nos ayudara a que es la mejor solución.

Con esto obtenemos lo siguiente:



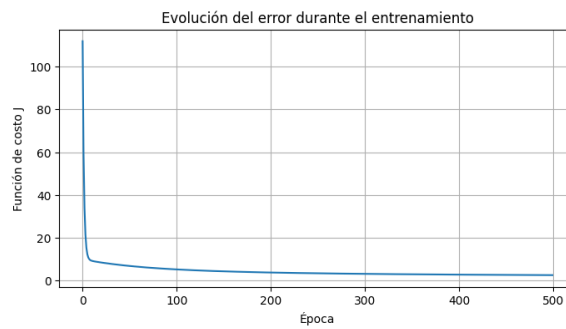
```

Theta multivariada: [0.00441825 0.05622692 0.1586245  0.0497737 ]
Error Medio Cuadrático (EMC): 5.014679314894278

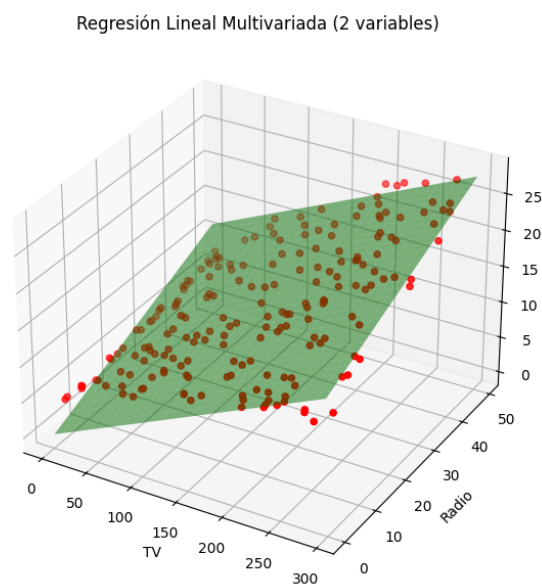
```

Figure 2. Resultados del modelo

Dado que al ser un hiperplano en 4 dimensiones no podemos graficar la función de costo, pero si podemos ver la evolución de los parámetros y el error cuadrático medio (MSE) en la figura 3.

**Figure 3.** Evolución del MSE

Como ejercicio complementario a nuestro ejercicio de n-Dimensiones, calculamos el descenso de gradiente para 2 variables, es decir, una función en 3 dimensiones la cual nos muestra un plano en el espacio tridimensional.

**Figure 4.** Regresion Lineal en 3D de 2 variables

3 Conclusión

La formulación vectorizada del modelo lineal permite generalizar de manera eficiente el uso de múltiples características en problemas de aprendizaje automático. Al expresar la hipótesis y la función de costo en notación matricial, se facilita la implementación computacional y se optimiza el cálculo en grandes volúmenes de datos. Con esto determinamos que el algoritmo de gradiente descendente es una herramienta fundamental para ajustar los parámetros de nuestro modelo, garantizando un entrenamiento más eficiente.