Asignatura: Optimización
Profesor: D. Sc. Gerardo García Gil
Alumno: 2023-A Miguel Angel Luis Espinoza 20110393
Ingeniería en Desarrollo de Software
Centro de Enseñanza Técnica Industrial (CETI)

Gradiente Descendente

Presentación

Realizar un programa del algoritmo de descenso de gradiente y su representación gráfica utilizando la plataforma de MATLAB.

Introducción

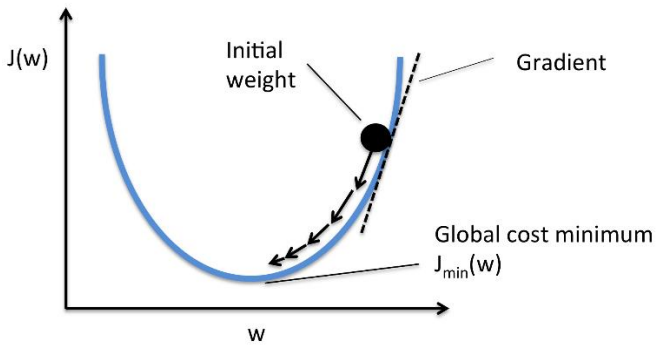
El descenso de gradiente es un algoritmo de optimización que se usa comúnmente para entrenar modelos de aprendizaje automático y redes neuronales. Los datos de entrenamiento ayudan a estos modelos a aprender con el tiempo, y la función de costo dentro del descenso de gradiente actúa específicamente como un barómetro, midiendo su precisión con cada iteración de actualizaciones de parámetros. Hasta que la función sea cercana o igual a cero, el modelo continuará ajustando sus parámetros para producir el menor error posible. Una vez que los modelos de aprendizaje automático se optimizan para la precisión, pueden ser herramientas poderosas para aplicaciones de inteligencia artificial (IA) y ciencias de la computación.

¿Cómo funciona?

la siguiente fórmula para la pendiente de una línea, que es y = mx + b, donde m representa la pendiente y b es la intersección en el eje y.

También puede recordar trazar un diagrama de dispersión en estadísticas y encontrar la línea de mejor ajuste, lo que requería calcular el error entre la salida real y la salida pronosticada utilizando la fórmula del error cuadrático medio. El algoritmo de descenso de gradiente se comporta de manera similar, pero se basa en una función convexa.

El punto de partida es solo un punto arbitrario para que podamos evaluar el rendimiento. Desde ese punto de partida, encontraremos la derivada (o pendiente), y desde allí, podemos usar una línea tangente para observar la inclinación de la pendiente. La pendiente informará las actualizaciones de los parámetros, es decir, los pesos y el sesgo. La pendiente en el punto inicial será más empinada, pero a medida que se generen nuevos parámetros, la pendiente debería reducirse gradualmente hasta alcanzar el punto más bajo de la curva, conocido como el punto de convergencia.



Similar a encontrar la línea de mejor ajuste en la regresión lineal, el objetivo del descenso de gradiente es minimizar la función de costo, o el error entre y predicho y real. Para hacer esto, requiere dos puntos de datos: una dirección y una tasa de aprendizaje. Estos factores determinan los cálculos de derivadas parciales de iteraciones futuras, lo que le permite llegar gradualmente al mínimo local o global (es decir, punto de convergencia).

La tasa de aprendizaje (también conocida como tamaño de paso o alfa) es el tamaño de

los pasos que se toman para alcanzar el mínimo. Suele ser un valor pequeño y se evalúa y actualiza en función del comportamiento de la función de coste.

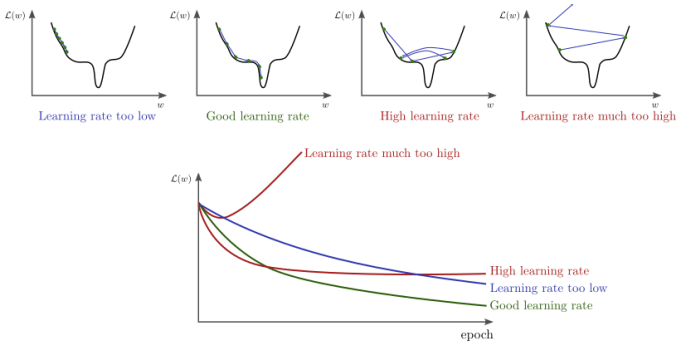
La función de costo (o pérdida) mide la diferencia, o error, entre y real y y pronosticado en su posición actual. Esto mejora la eficacia del modelo de aprendizaje automático al proporcionar retroalimentación al modelo para que pueda ajustar los parámetros para minimizar el error y encontrar el mínimo local o global.

Tipos

Descenso de gradiente por lotes:

Suma el error de cada punto en un conjunto de entrenamiento, actualizando el modelo solo después de que se hayan evaluado todos los ejemplos de entrenamiento. Este proceso se conoce como una época de entrenamiento.

Si bien este procesamiento por lotes proporciona eficiencia de cálculo, aún puede tener un tiempo de procesamiento prolongado para grandes conjuntos de datos de entrenamiento, ya que aún necesita almacenar todos los datos en la memoria.

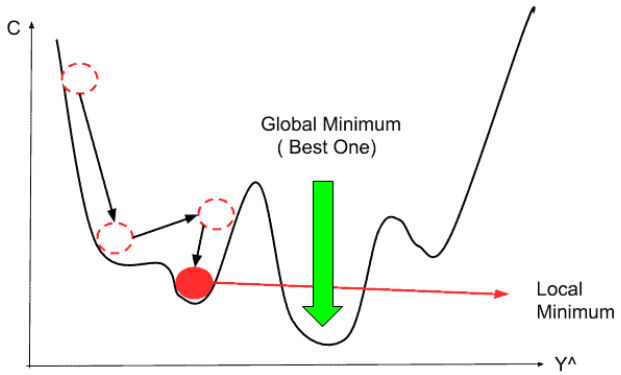


Descenso de gradiente estocástico:

El descenso de gradiente estocástico (SGD) ejecuta una época de entrenamiento para cada ejemplo dentro del conjunto de datos y actualiza los parámetros de cada ejemplo de entrenamiento uno a la vez. Dado que solo necesita mantener un ejemplo de entrenamiento, son más fáciles de almacenar en la memoria. Si bien estas actualizaciones frecuentes pueden ofrecer más detalles y velocidad, pueden generar pérdidas en la eficiencia computacional en comparación con el descenso de gradiente por lotes.

Descenso de gradiente de mini lotes:

El descenso de gradientes en minilotes combina conceptos tanto del descenso de gradientes por lotes como del descenso de gradientes estocásticos. Divide el conjunto de datos de entrenamiento en lotes pequeños y realiza actualizaciones en cada uno de esos lotes. Este enfoque logra un equilibrio entre la eficiencia computacional del descenso de gradiente por lotes y la velocidad del descenso de gradiente estocástico.



Fórmula matemática

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Now,

$$\frac{\partial}{\partial \theta} J_\theta = \frac{\partial}{\partial \theta} \frac{1}{2m} \sum_{i=1}^{m} \left[h_\theta(x_i) - y_i \right]^2$$

$$\frac{\partial}{\partial \theta} J_\theta = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x_i) - y_i \right) \cdot \frac{\partial}{\partial \theta_j} \left(\theta x_i - y_i \right)$$

$$\frac{\partial}{\partial \theta} J_\theta = \frac{1}{m} \sum_{i=1}^{m} \left[\left(h_\theta(x_i) - y \right) x_i \right]$$

Therefore,

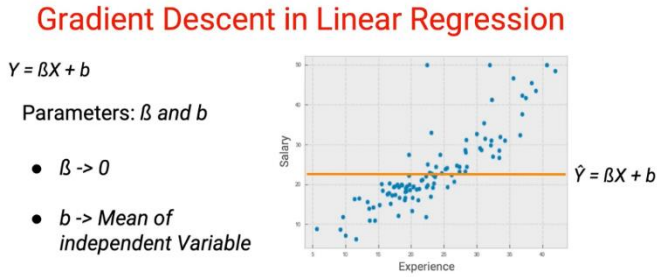
$$\theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} \left[\left(h_\theta(x_i) - y_i \right) x_i \right]$$

Inicialización aleatoria y generación de predicción

Aquí estamos usando un modelo de regresión lineal. Comenzar con un modelo de referencia siempre es una gran idea. En nuestro modelo

de referencia, usamos los valores 0 para B (pendiente de la línea) y b (intersección) es la media de todas las variables independientes.

Ahora podemos usar este modelo de referencia para tener nuestros valores predichos y sombrero.



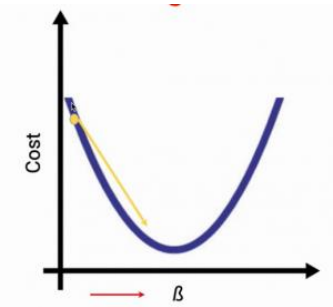
Calcular costo/error

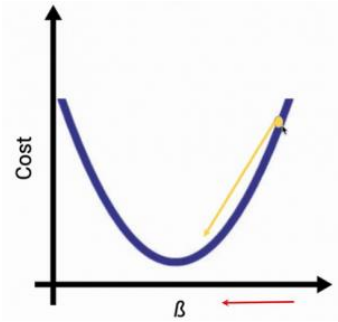
Una vez que tenemos nuestra predicción, podemos usarla en el cálculo de errores. Aquí, nuestra métrica de error es el error cuadrático medio (MSE).

El error cuadrático medio es la diferencia cuadrática promedio entre los valores estimados y el valor real.

$$J = \frac{\sum_{i=1}^{n} (\hat{Y}_i - Y_i)^2}{n}$$

Nuestro valor de error calculado del modelo de referencia puede llevarnos a cualquier parte de la curva de Costo-B, como se muestra en las siguientes imágenes. Ahora nuestra tarea es actualizar B de tal manera que lleve el error hacia la parte inferior de la curva.

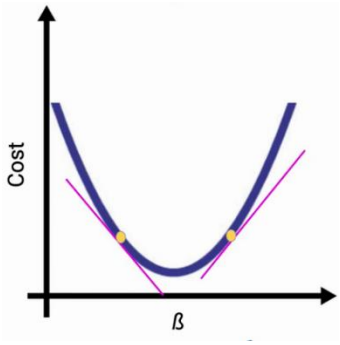




Actualizar parámetros

Ahora la pregunta es cómo se actualizarán los parámetros (B en este caso). Para actualizar nuestros parámetros vamos a utilizar las derivadas parciales. Las derivadas parciales también dan la pendiente de la línea, es el cambio en el costo con respecto al cambio en B.

Mira la imagen de abajo en cada caso la derivada parcial dará la pendiente de la tangente. En el primer caso la pendiente será negativa mientras que en el otro caso la pendiente será positiva.



Una vez que tenemos la derivada parcial, podemos actualizar los valores de B como se muestra en la imagen de abajo.

$$ß = ß - \alpha G_ß$$

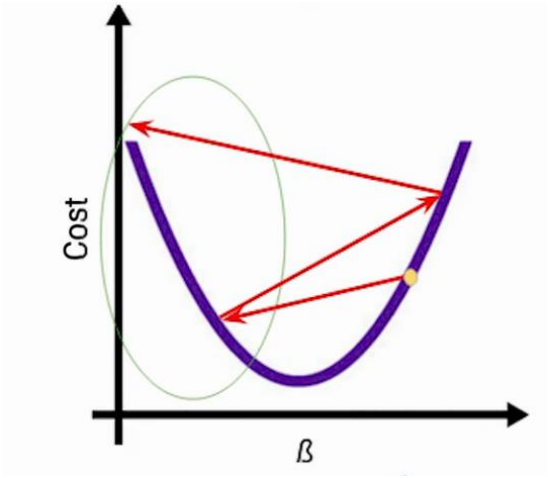
En general, todo el proceso de actualización de los parámetros se verá como en la siguiente imagen.

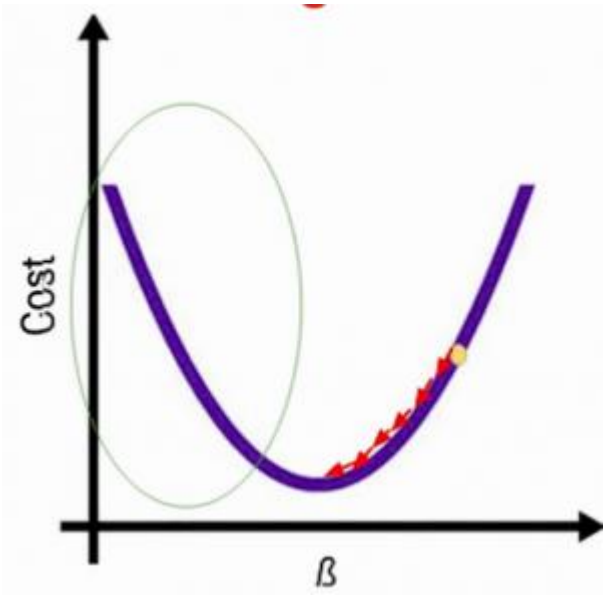
$$G_ß = \frac{\partial(J)}{\partial ß} = \frac{2\sum_{i=1}^{n}(ßX_i + b - Y_i)X_i}{n} \qquad ß = ß - \alpha G_ß$$

Otro aspecto importante de todo este proceso es la tasa de aprendizaje (a). La tasa de aprendizaje es un hiperparámetro que decide el curso y la velocidad del aprendizaje de nuestro modelo.

La tasa de aprendizaje debe ser un valor óptimo. Si la tasa de aprendizaje será alta, los pasos que se den serán grandes y podemos perder los mínimos. Como resultado, el modelo no convergerá.

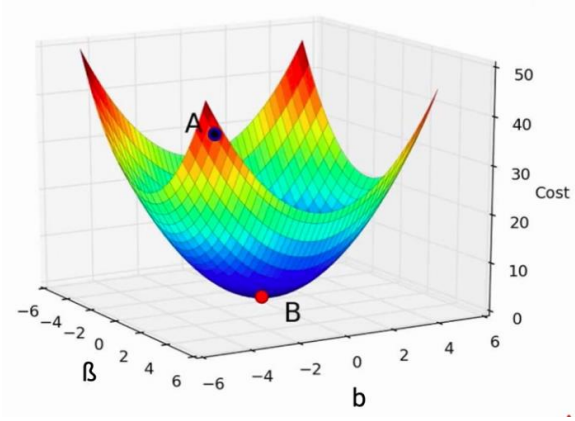
Por otro lado, si la tasa de aprendizaje es demasiado pequeña, el modelo tardará demasiado en alcanzar el costo mínimo.





Efecto de la tasa de aprendizaje (a) tasa de aprendizaje demasiado alta (b) tasa de aprendizaje demasiado baja

Ahora surge una pregunta, ¿qué pasa si hay múltiples parámetros en un modelo? En tales casos, nada cambiará, solo aumentarán las dimensiones de la función de costo. Como se convierte en un gráfico 3D en la siguiente imagen. Además, el objetivo seguirá siendo el mismo, es decir, alcanzar los mínimos.



Además, los parámetros se actualizarán exactamente como en el caso anterior, como se muestra a continuación.

$$G_{ß} = \frac{\partial(J)}{\partial ß} = \frac{2\sum_{i=1}^{n}(ßX_i + b - Y_i)X_i}{n} \qquad ß = ß - \alpha G_{ß}$$

$$G_{b} = \frac{\partial(J)}{\partial b} = \frac{2\sum_{i=1}^{n}(ßX_i + b - Y_i)}{n} \qquad b = b - \alpha G_{b}$$

Desarrollo

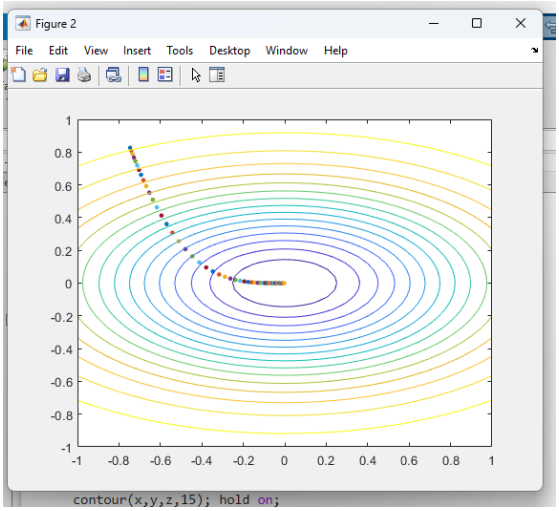
Con este algoritmo se desarrollará un programa con el IDE de MATLAB, donde la función será: 10-(exp(-1*(x^2+3*y^2))). Se grafica en 3D.

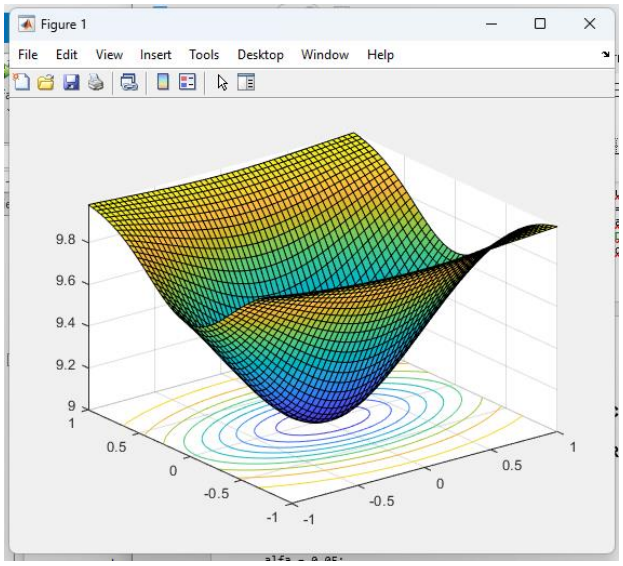
Código

```
%% PROGRAM 1.1 IMPLEMENTANTION OF GRADIENT
DESCENT METHOD IN MATLAB
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Gradient descent example
% Miguel Luis
% Clear memory
clear all

funstr='10-(exp(-1*(x^2+3*y^2)))';
f=vectorize(inline(funstr));
range=[-1 1 -1 1];
%Draw the function
Ndiv=50;
dx=(range(2)-range(1))/Ndiv; dy=(range(4)-
range(3))/Ndiv;
```

```
[x,y] = meshgrid(range(1):dx:range(2),
range(3):dy:range(4));
z=(f(x,y));
figure(1);surfc(x,y,z);
%Define the number of iterations
k=0;
niter=200;
%Gradient step size h definition
hstep = 0.001;
%Step size of the Gradient descent method
alfa = 0.05;
%Initial point selection
xrange=range(2)-range(1);
yrange=range(4)-range(3);
x1=rand*xrange+range(1);
x2=rand*yrange+range(3);
% Optimization process
while (k<niter)
    % Function evaluation
    zn=f(x1,x2);
    % Computation of gradients gx1 y gx2
    vx1=x1+hstep;
    vx2=x2+hstep;
    gx1=(f(vx1,x2)-zn)/hstep;
    gx2=(f(x1,vx2)-zn)/hstep;
    % Draw the current position
    figure(2)
    contour(x,y,z,15); hold on;

plot(x1,x2,'.','markersize',10,'markerfacecol
or','g');
    hold on;
    % Computation of the new solution
    x1=x1-alfa*gx1;
    x2=x2-alfa*gx2;
    k=k+1;

end
```

Resultados





Conclusión

Con la elaboración de esta práctica se familiarizo con la lógica del algoritmo descenso de gradiente y con el sistema de cómputo de MATLAB; dentro de su sintaxis y la elaboración de gráficas, uno de sus mayores potenciales.

Referencias

1. Saxena, S. (2021, 12 marzo). Understanding Gradient Descent Algorithm. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/03/understanding-gradient-descent-algorithm/
2. What is Gradient Descent? | IBM. (s. f.). https://www.ibm.com/topics/gradient-descent
3. Donges, N. (2021, 23 julio). Gradient Descent in Machine Learning: A Basic Introduction. Built In. https://builtin.com/data-science/gradient-descent