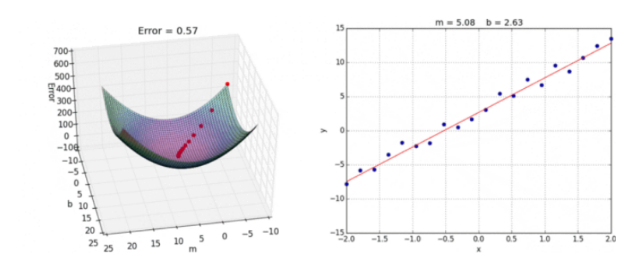
**Descenso por el Gradiente**

**Gradiente:** Es la derivada o la **tasa de cambio de una función**. Es un **vector** (dirección) que apunta en la **dirección de mayor aumento de una función** y es **cero en un máximo o mínimo local**, donde no hay una dirección única de aumento. Se usan los **gradientes en funciones con muchas variables y una única salida**. El gradiente de una recta es su pendiente. No tiene sentido hablar de gradientes en funciones de una sola variable.

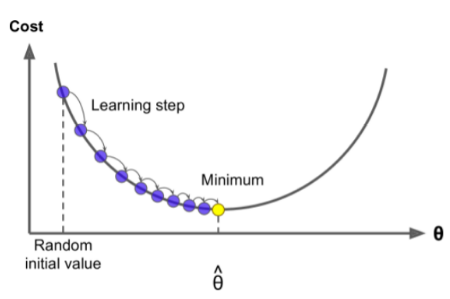
**Descenso por el Gradiente:** Es un **algoritmo de optimización genérico** que **busca encontrar el mínimo** de una función. Es **iterativo**: Se parte de un punto aleatorio en la función y nos vamos moviendo en la dirección negativa del gradiente de dicha función **para alcanzar mínimos locales o globales**. Al aplicarlo **en machine learning**, lo que estamos haciendo es **buscar los mínimos de la función de costo** (J: función que mide la **diferencia entre el valor real y la predicción** del modelo). Es una **función convexa**.

**Descenso por el Gradiente:** Si no podemos calcular los parámetros de un modelo analíticamente, entonces usamos el descenso por el gradiente para buscar dichos parámetros con un algoritmo de optimización:

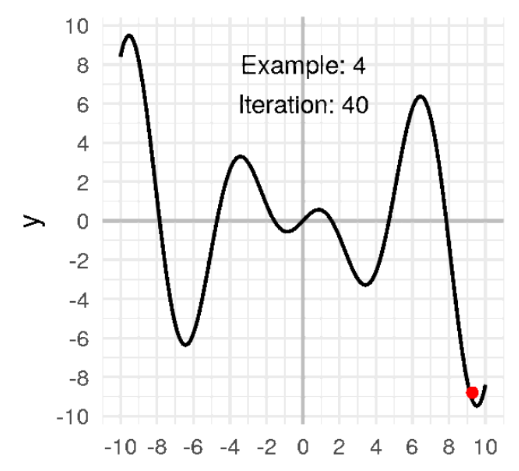


Si nuestro modelo tiene muchas variables, muchas observaciones o no podemos resolver la función de costo de forma analítica; tales como en una regresión logística o en redes neuronales; entrenamos el modelo usando el descenso por el gradiente.

**Función de Costo:** La función de costo de la regresión lineal es una función convexa. Entonces el descenso por el gradiente va a converger al mínimo global.



Si estamos trabajando con funciones de costo que tienen más de un mínimo, tales como la función logística, es importante ejecutar el descenso por el gradiente iniciándolos desde distintos lugares para evitar quedar atrapados en un mínimo local:

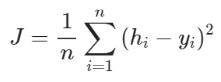


**Ejemplos de funciones de Costo**:

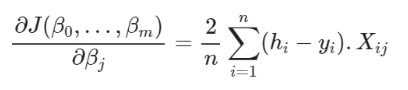
En la **Regresión Lineal**, la función de costo (J) es el **Error Cuadrático Medio** (RMSE) entre el valor predicho y el valor real. Si tenemos un dataset de n filas y m columnas, para la fila i del dataset el valor predicho hi es:



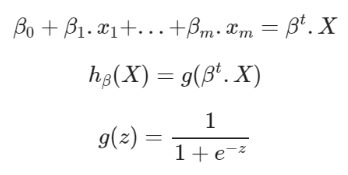
La función de costo es:



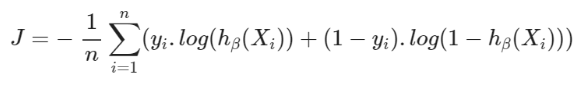
El gradiente es:



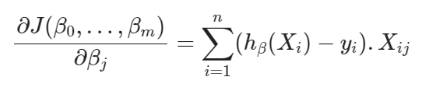
En la **Regresión Logística**:



Siendo la función de costo:



Y el gradiente:

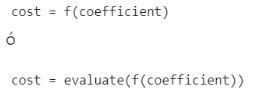


**Algoritmo de Descenso por el Gradiente:**

El algoritmo empieza estableciendo valores iniciales para el o los coeficientes de la función. Dichos valores pueden ser 0 o valores aleatorios pequeños.



El costo para estos valores en los coeficientes se calcula evaluando la función de costo en los valores iniciales.

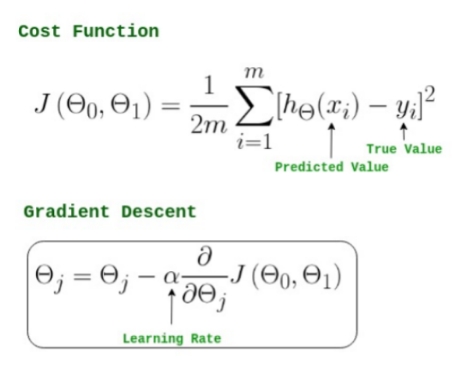


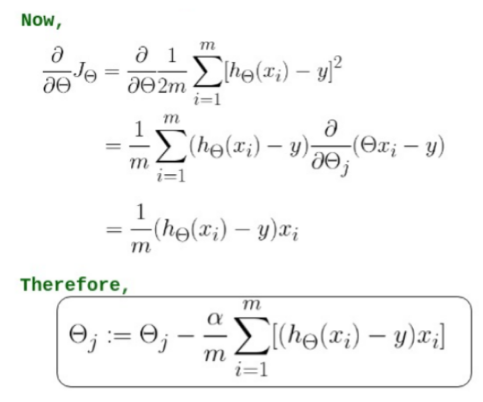
Luego, se calcula la derivada de esta función de costo. Esto nos da la pendiente de dicha función en un punto dado. Conociendo la pendiente podemos determinar la dirección (signo) en que debemos variar los valores de los coeficientes para que el costo en la siguiente iteración sea más chico.



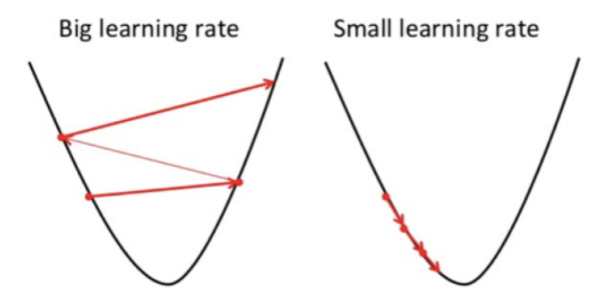
Conociendo a partir de la derivada cuál es la dirección “cuesta abajo”, podemos actualizar los valores de los coeficientes.

En una regresión lineal simple:

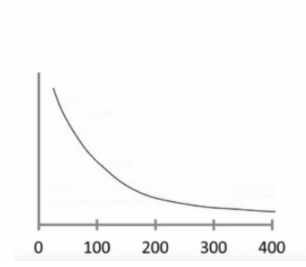




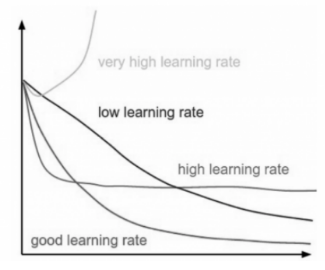
**Learning Rate:** La **tasa o velocidad de aprendizaje α** es un **hiperparámetro** muy importante del descenso por el gradiente. Para **reducir** el **tiempo de convergencia** del descenso por el gradiente, conviene trabajar con **variables normalizadas**.



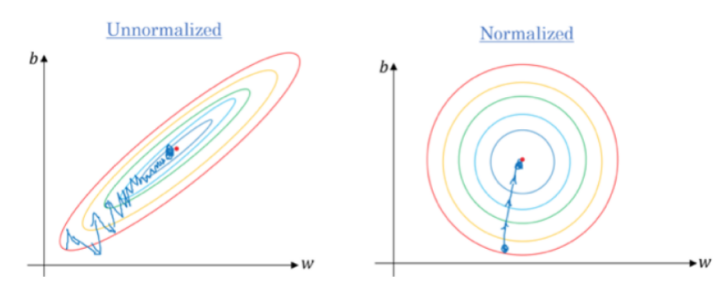
Si graficamos los valores de la función de costo mientras ejecutamos la optimización, podemos asegurarnos de estar ejecutando correctamente el descenso por el gradiente. Podemos graficar el número de iteraciones en el eje x y el valor de la función de costo en el eje y. Entonces podemos ver qué valor tiene la función de costo después de cada iteración y detectar si la tasa de aprendizaje es o no apropiada.



Acá tenemos la evolución de la función de costo aplicando varias tasas de aprendizaje:



Estos gráficos nos muestran que si normalizamos los datos antes de aplicar el método, llegamos al resultado de una manera mucho más rápida:



**Stopping Criteria:** El algoritmo de descenso por el gradiente puede terminarse a partir de una cantidad máxima de iteraciones; o bien una **tolerancia absoluta** (que el valor de la función de costo se acerque “lo suficiente” a cero) o bien, una **tolerancia relativa** (cuando la mejora en una iteración sea menor que un umbral determinado).

**Conclusión:** El **Descenso por el Gradiente** es un **Algoritmo de Optimización Simple** que **puede** **ser utilizado** en muchos algoritmos de **Machine Learning**.