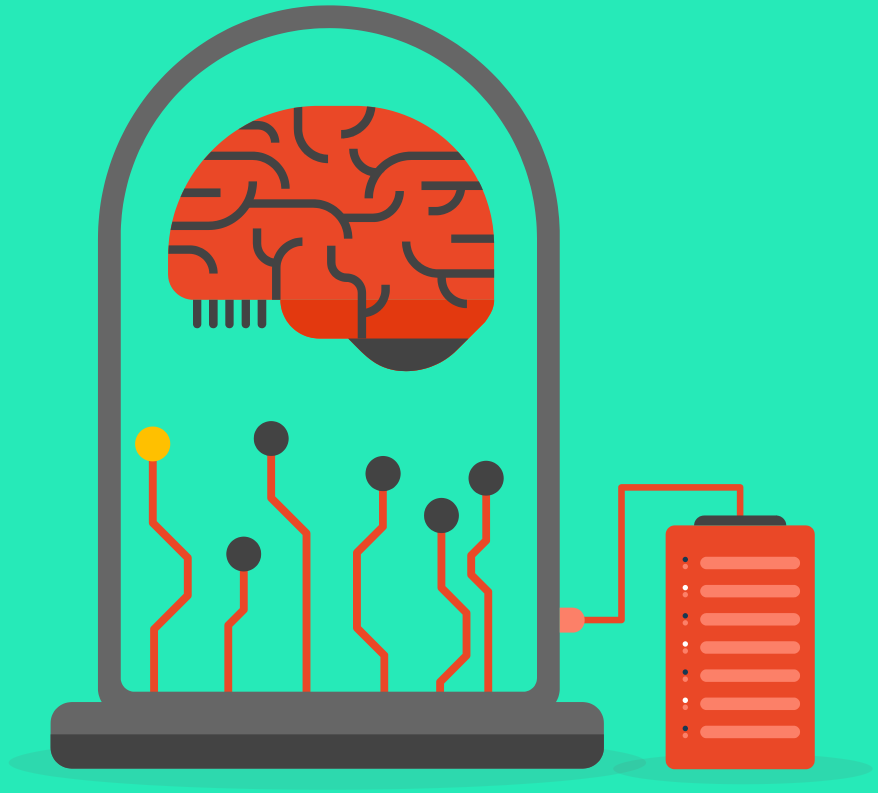
1. Regresión Lineal



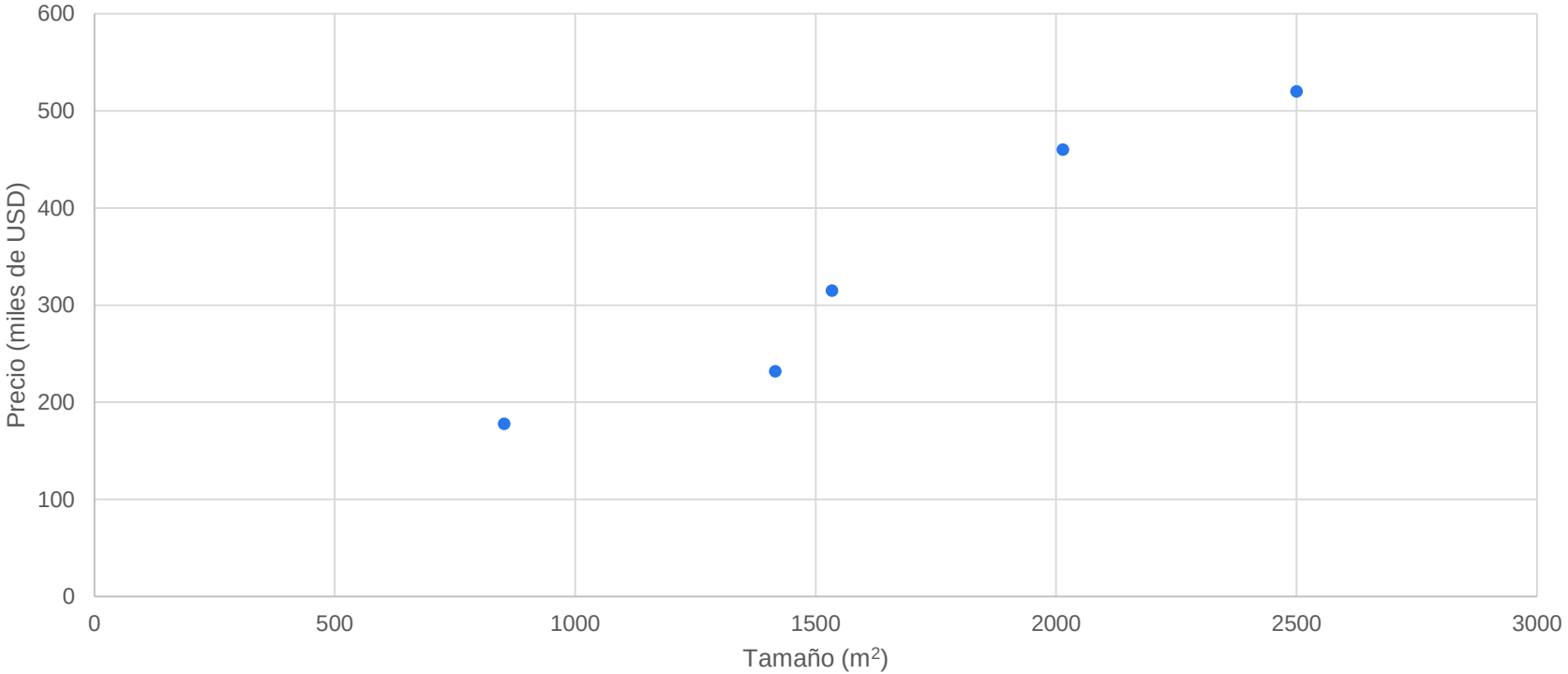
Planteamiento del Problema

Partamos de un conjunto de datos conocidos

Tamaño (metros cuadrados)	Precio (miles de USD)
2500	520
2014	460
1416	232
1534	315
852	178

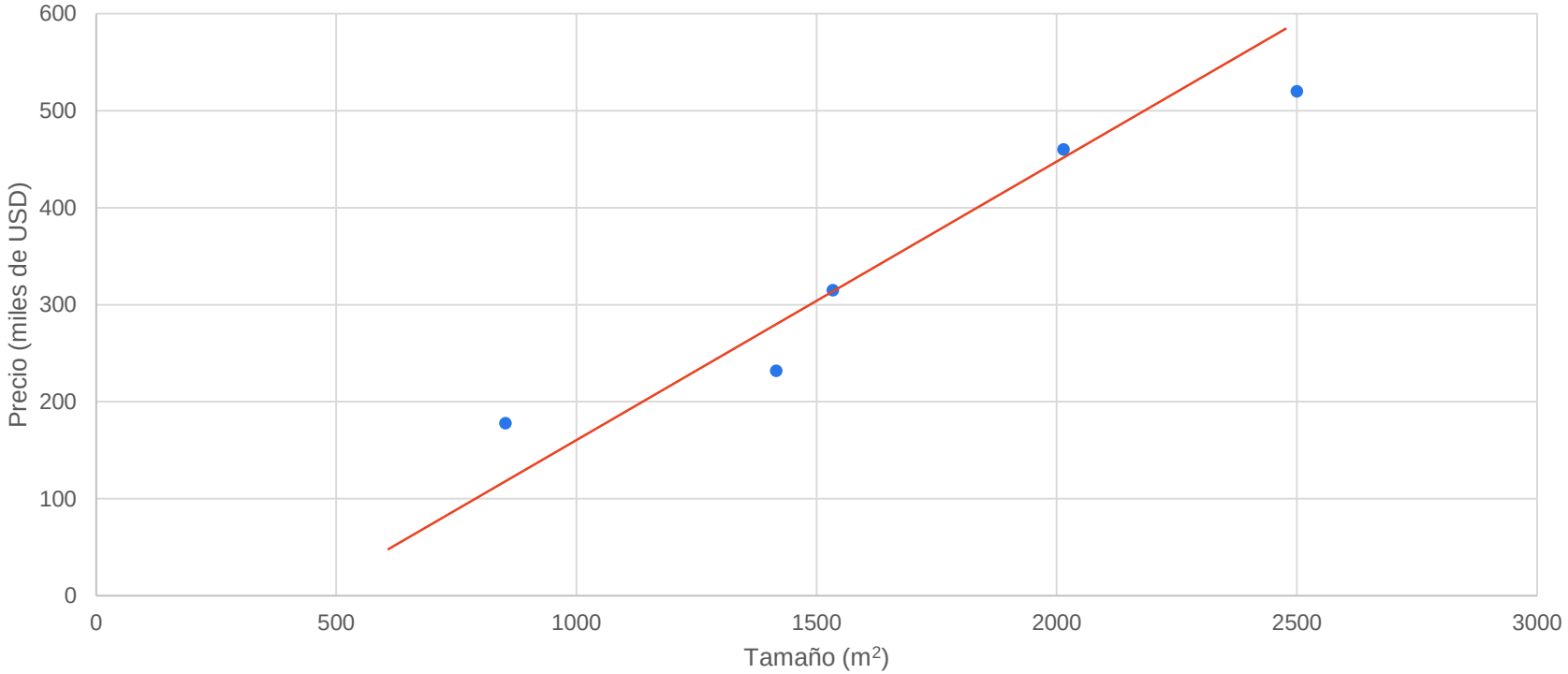
¿Es posible predecir el precio de cualquier casa a partir de estos datos?

Planteamiento del Problema

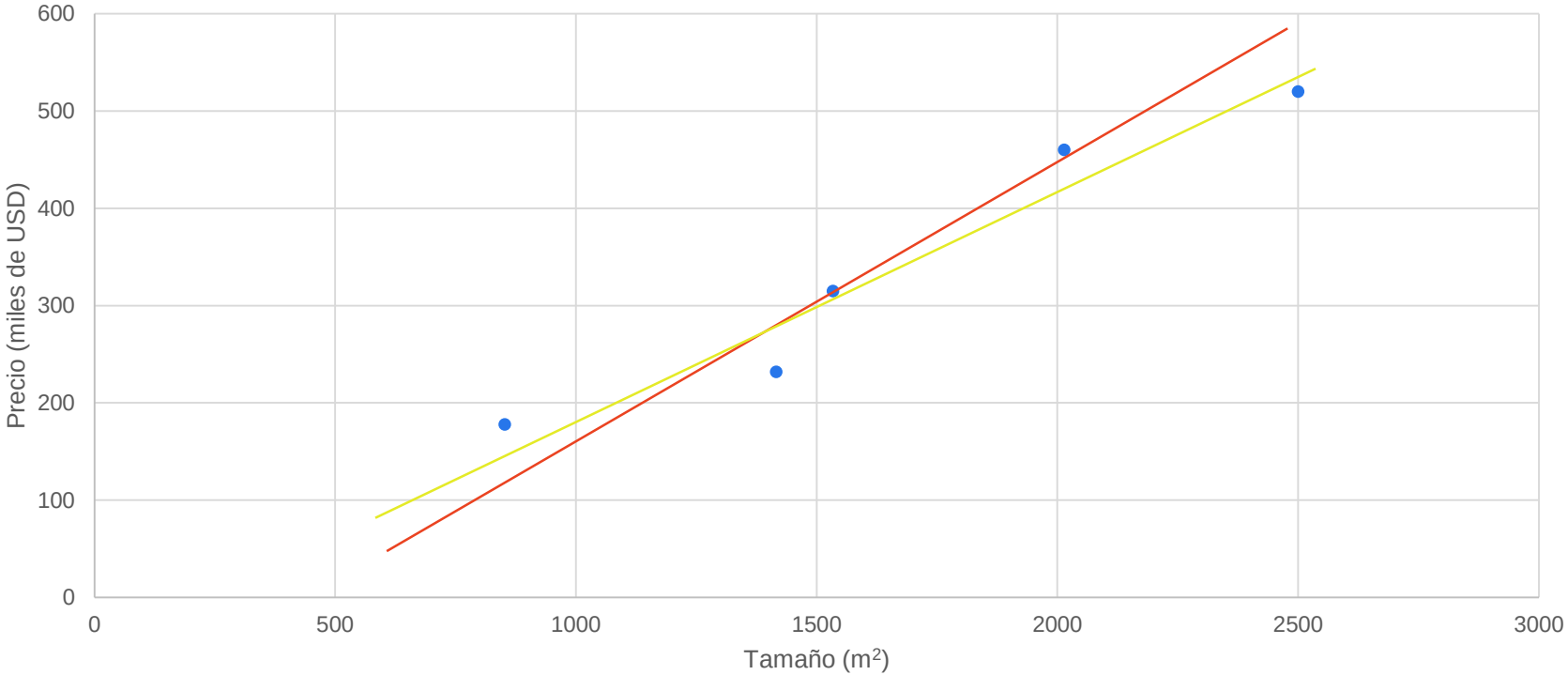


A fin de poder predecir los precios, tenemos que encontrar una ecuación que ***modele*** los datos.

Planteamiento del Problema



Planteamiento del Problema



¿Cuál es la recta que mejor modela los datos?

Planteamiento del Problema

x (entrenamiento) y (salida deseada)

Tamaño (metros cuadrados)	Precio (miles de USD)
2500	520
2014	460
1416	232
1534	315
852	178

m

$$y = h(x)$$

h es una función que llamaremos **hipótesis**.

Hipótesis

A partir de la gráfica, podemos suponer que la relación entre la variable dependiente y la variable independiente es lineal. Por lo tanto, establecemos que nuestra función hipótesis tiene la forma:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

En donde los θ_i son parámetros del modelo. A este tipo de función se le conoce como **regresión lineal univariada**.

Hipótesis

A partir de la gráfica, podemos suponer que la relación entre la variable dependiente y la variable independiente es lineal. Por lo tanto, establecemos que nuestra función hipótesis tiene la forma:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

En donde los θ_i son parámetros del modelo. A este tipo de función se le conoce como **regresión lineal univariada**.

Ahora bien, valores diferentes de los parámetros θ_i producirán ajustes diferentes a los datos de entrenamiento. **¿Cómo determinamos los mejores valores de los parámetros?**

Función de Costo

A fin de conocer cuáles son los parámetros que mejor modelan los datos de entrenamiento, es necesario primero definir alguna manera de medir el **desempeño** del modelo. Para ello, se establece una **Función de Costo** que permite conocer la distancia que existe entre las predicciones del modelo, y las salidas deseadas:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

En donde los y_i son las salidas deseadas para los datos de entrenamiento. A esta función de costo se le conoce como **Error Cuadrático Medio**.

Función de Costo

A fin de conocer cuáles son los parámetros que mejor modelan los datos de entrenamiento, es necesario primero definir alguna manera de medir el **desempeño** del modelo. Para ello, se establece una **Función de Costo** que permite conocer la distancia que existe entre las predicciones del modelo, y las salidas deseadas:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

En donde los y_i son las salidas deseadas para los datos de entrenamiento. A esta función de costo se le conoce como **Error Cuadrático Medio**.

El proceso de **entrenamiento** consiste en encontrar los valores de los parámetros θ que minimicen el valor de la función de costo $J(\theta)$.

Ecuación Normal

En el caso de la regresión lineal, es posible encontrar una *solución analítica cerrada* que permite obtener dichos valores que minimizan la función de costo:

$$\hat{\theta} = (X^T X)^{-1} X^T y$$

En donde $\hat{\theta}$ son los parámetros que minimizan la función de costo. $\mathbf{X}$ es un vector con los datos de entrenamiento $\{x_0, x_1, \ldots, x_n\}$ con $x_0 = 1$, y $\mathbf{y}$ es un vector con las salidas deseadas.

Medida del Error

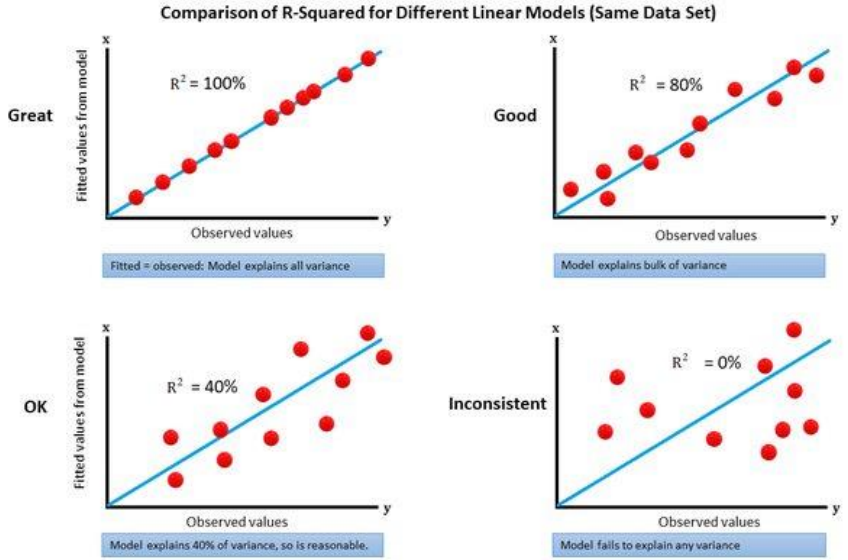
Una vez encontrados los parámetros que minimizan la función de costo, es necesario siempre tener alguna idea de la medida del error que comete el modelo obtenido. En regresión, se puede emplear para ello la raíz cuadrada del error cuadrático medio, conocido como el **RMSE** (root mean square error):

$$RMSE = \sqrt{\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2}$$

Esta cantidad viene a representar una estimación de las desviaciones estándar de los errores promedio, y nos dan una idea de qué tanto se aleja en promedio nuestro modelo de los valores reales.

Medida del Error

Otra cantidad que suele emplearse para medir la calidad del ajuste de un modelo de regresión es el conocido como **R^2** (R cuadrado), y que representa la proporción de variancia para la variable dependiente que es explicada por una variable independiente.



En términos generales, mientras mayor sea el valor de **R^2** para un modelo, mayor correlación y mejor ajuste habrá entre el modelo y los datos.

1er Notebook Práctico

Ya que conocemos los conceptos de regresión lineal, función de costo, ecuación normal y medidas del error para un modelo de regresión, vamos a realizar la implementación de la ecuación normal para los datos descritos, así como su solución empleando el método de los mínimos cuadrados, y veremos el resultado de los ajustes en función de las medidas del error discutidas.

¡Adelante con el Notebook!

Método del Descenso del Gradiente

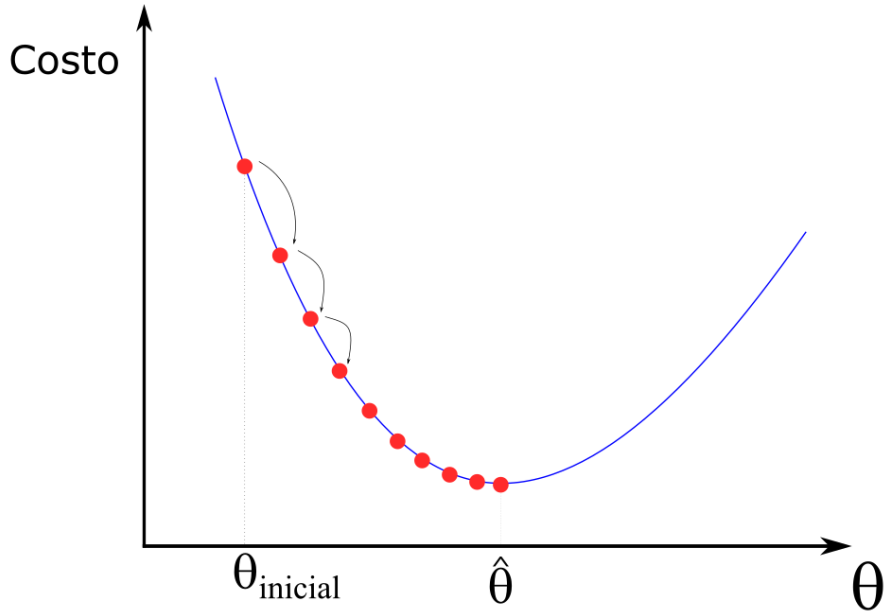
Como ya vimos, a partir de la Ecuación Normal es posible obtener los coeficientes que minimizan la función de costo de un problema de regresión. Sin embargo, ya que dicho calculo implica la inversión de la matriz $\boldsymbol{X^T X}$, dependiendo de la cantidad de variables que intervengan en un modelo, esta operación puede hacerse computacionalmente inmanejable (tanto por espacio en memoria como por capacidad de cómputo).

Una alternativa ampliamente usada para realizar el entrenamiento de modelos de regresión lineal (y muchos otros modelos de Machine Learning) como el visto, y que está diseñado para manejar grandes cantidades de variables y datos de entrenamiento, es el del algoritmo conocido como **Descenso del Gradiente**.

Veamos de qué se trata este algoritmo.

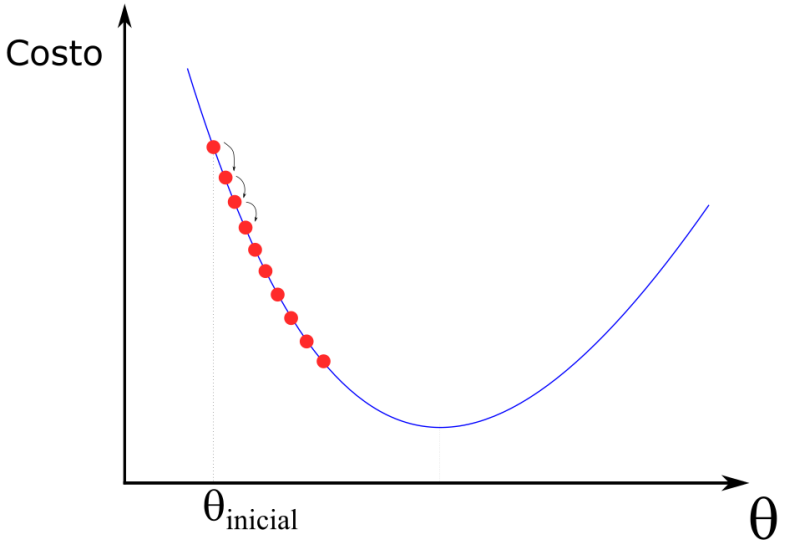
Método del Descenso del Gradiente

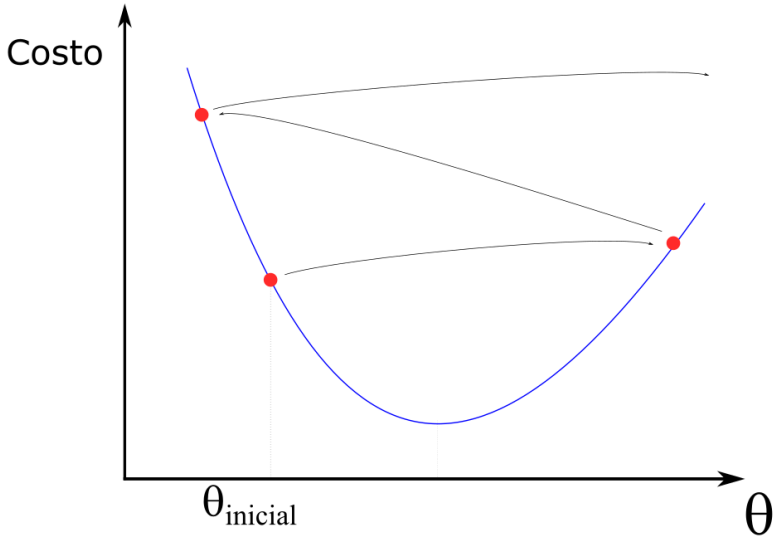
El Descenso del Gradiente es un algoritmo de optimización genérico capaz de encontrar soluciones para una gran variedad de problemas. La idea general del Descenso del Gradiente es la de variar ligeramente los parámetros de manera iterativa a fin de minimizar la función de costo.



Método del Descenso del Gradiente

El tamaño de los pasos en cada iteración está determinado por un *hiperparámetro* denominado constante de aprendizaje (learning rate). Dependiendo de su valor, el algoritmo podría tardar demasiado en encontrar el mínimo (si es muy pequeño), o bien saltar el mínimo y explotar (si es muy grande).





Descenso del Gradiente por Lotes (Batch Gradient Descent)

Para aplicar el Descenso del Gradiente, es necesario calcular el gradiente de la función de costo respecto a los parámetros de la misma, es decir, los θ_i. Esto significa calcular el cambio de la función respecto a los θ_i , es decir, la *derivada parcial* (el gradiente de una función con respecto a un parámetro en un punto nos da el valor de la pendiente de mayor crecimiento en ese punto), de modo que si nos movemos en la dirección contraria al gradiente en ese punto, nos dirigiremos hacia el mínimo de la función.
Recordemos la función de costo:

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2 = \frac{1}{2m}(\boldsymbol{X\theta} - \boldsymbol{y})^2$$

Descenso del Gradiente por Lotes (Batch Gradient Descent)

Calcular el gradiente de la función de costo significa calcular las derivadas parciales respecto a todos los coeficientes:

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2 = \frac{1}{2m}(\boldsymbol{X\theta} - \boldsymbol{y})^2$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \nabla_\theta J(\theta) = \frac{1}{m}\boldsymbol{X^T}(\boldsymbol{X\theta} - \boldsymbol{y})$$

Descenso del Gradiente por Lotes (Batch Gradient Descent)

La actualización de los parámetros de la función de costo, en cada iteración del algoritmo, estará dada finalmente por:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta)$$

En donde $\boldsymbol{\eta}$ es la constante de aprendizaje. De este modo, en cada iteración se corrigen los parámetros de la función de costo en la dirección contraria a la pendiente de la misma, y la actualización de dichos parámetros se hace tras usar **todos los datos de entrenamiento** en el cálculo en cada iteración. Por esa razón se llama Batch Gradient Descent.

Descenso del Gradiente Estocástico (Stochastic Gradient Descent)

El principal problema del Descenso del Gradiente por Lotes, es que utiliza todo el conjunto de entrenamiento completo en cada iteración, lo que puede resultar lento si dicho conjunto es muy grande. En el extremo opuesto, el Descenso del Gradiente Estocástico, **toma un solo dato de entrenamiento al azar** por iteración. La ventaja es que el algoritmo resulta muy rápido ya que se modifican los parámetros un dato cada vez, y se pueden entrenar conjuntos de datos muy grandes ya que no requieren mucha capacidad de almacenamiento en memoria.

Sin embargo, debido a su naturaleza estocástica (al azar), la función de costo no disminuirá de manera suave durante el entrenamiento, sino que oscilará progresivamente hacia el mínimo, y luego se mantendrá oscilando sin alcanzar el mínimo óptimo.

Descenso del Gradiente Estocástico (Stochastic Gradient Descent)

Para minimizar este efecto, se establece una constante de aprendizaje que vaya disminuyendo su valor de manera progresiva en cada iteración, de manera de atenuar las oscilaciones de la función de costo en el punto cercano al mínimo. Por ejemplo, usando una función de la forma:

$$\eta_{t+1} = \frac{t_0}{t + t_1}$$

En donde t_0 y t_1 son parámetros que regulan el ritmo de decaimiento de la constante.

2do Notebook Práctico

Ya que conocemos la teoría matemática detrás del Método del Descenso del Gradiente, veamos cómo podemos implementarlo para la resolución de un problema de Regresión Lineal

¡Adelante con el Notebook!