Created	@September 3, 2023 4:51 PM
Reviewed	☑

Regresión Lineal con Solo una variable

- En palabras simples, implica ajustar una línea recta, en un conjunto de datos
- Este tipo de modelos predice números, hay una cantidad “infinita” posibilidad de resultados, para cada x le corresponde un y. Por ejemplo, para calcular el precio de una acción, puede haber un resultado que de 1 dólar para un x1 y para un x2 5000 dólares.
- Otros modelos como clasificación, predice categorías, esto implica que los resultados son finitos, ya que se deben definir las categorías antes de la predicción. Por ejemplo para clasificar un animal, hay finitos posibles resultados que son la cantidad de animales.

Modelo:

$$\mathbf{x} \rightarrow \mathbf{f} \rightarrow \hat{y}$$

x: Entrada o input

f: Nuestra predicción o línea recta

$\hat{y}$: Output o predicción

¿Cómo se representa la f?

$$f_{w,b}(x) = wx + b$$

w, b: Son valores, pero estos afectarán el resultado de $\hat{y}$

w, b también son llamados como parámetros o coeficientes

Cómo se darán cuenta, es la fórmula de una función lineal donde:

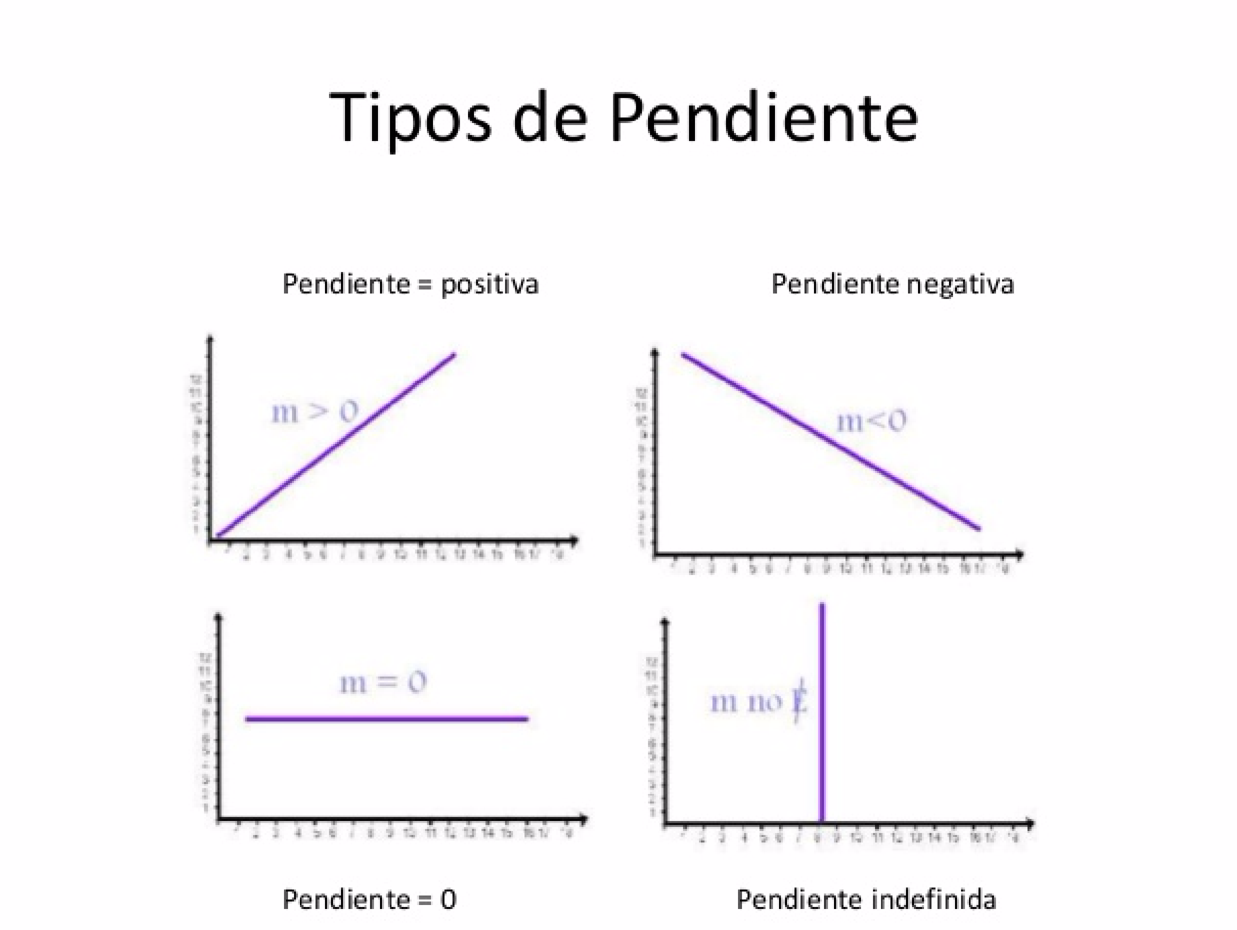
- w es la pendiente
- b es la intersección con el eje y

Función de costo:

Nuestra función de costo será $J(w,b)$

Es una función que nos dirá, que tan bien funciona el modelo, osea, si los valor w,b fueron bien escogidos

Por ejemplo, dependiendo de la pendiente (w), la función será totalmente diferente:



Y dependiendo de el insersecto (b), la función cambiará su valor inicial, o sea, cuando x = 0, y = b

1) $f(x) = wx + b$

2) $f(0) = w(0) + b$

3) $f(0) = b$

Entonces decimos lo siguiente:

- a) La predicción ŷ se obtiene, al pasar el valor x^i por la función f → $\hat{y}^i = f(x^i)$
- b) Nuestro modelo f, o sea la ecuación o función lineal, es → $f(x^i) = wx^i + b$

- Pero hay que recordar que $\hat{y}^i$ es la predicción para la función f en el punto x^i, pero el valor real para x^i es y^i
- Por lo que no siempre $\hat{y}^i = y^i$

El error entre nuestra predicción para el x^i esta dado por la diferencia: $\left(\hat{y}^i - y^i\right)$ Ya que esto nos indicará, que tan acertado está nuestra predicción del valor real, o sea que entre más cercano a 0 sea el error, nuestro modelo será más confiable.

Para el algoritmo que usaremos, será necesario elevar el error al cuadrado: $\left(\hat{y}^i - y^i\right)^2$

Pero este error es solo para un punto específico, para poder calcular el error de nuestro modelo completo usaremos:

$$J(w,b) = \sum_{i=1}^{m} (\hat{y}^i - y^i)^2$$

Donde m es igual, al tamaño de nuestro de set de datos, la cantidad de datos que hallamos usado para entrenar al modelo

Pero si recordamos, nos daremos cuenta que $\hat{y}^i = f_{w\,,b}\left(x^i\right)$, ya que la predicción $\hat{y}^i$ es el valor que predice nuestro modelo o función f en el punto x^i

$$J(w,b) = \sum_{i=1}^{m} (f_{w\,,b}\left(x^i\right) - y^i)^2$$

Por convención, necesitamos el promedio del error de la función → 1/2m

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} (f_{w\,,b}\left(x^i\right) - y^i)^2$$

Nuestro objetivo, es hacer que J sea lo más cercano a 0 posible, o sea:

$$min_x J(w, b)$$

Esto es encontrar los valores w y b, que minimizan al máximo la función J

Gradient Descent:

Para no tener que probar manualmente valores para w y b, que nos den la función J deseada, usaremos el algoritmo de descenso gradual

Este algoritmo ayudar a minimizar el error en cualquier tipo de regresión, no solo lineal

Algoritmo:

$$w = w - \alpha\nabla(_{w}, _{b})$$

- '=': Es el operador de asignación, no es una igualdad
- α: Tasa de aprendizaje ⇒ Es un número positivo muy pequeño, entre 0 y 1 (0.01); controls how big of a step take downhill
- $\nabla J(_{w}, _{b})$: Es el gradiente de la función de costo (un vector de derivadas parciales para cada parámetro). $\frac{d}{dw}J(_{w}, _{b})$

$$b = b - \alpha\nabla J(_{w}, _{b})$$

El algoritmo repite estos 2 pasos hasta que converge, esto quiere decir, cuando alcanza el mínimo local, donde los valores para w y b no cambien mucho en cada iteración adicional

Para esto, se requiere actualizar los valores para w y b al mismo tiempo

Implementación

repeat until convergence {

$tmp_w = w - \alpha \frac{d}{dw} J(_{w,b})$

$tmp_b = b - \alpha \frac{d}{db} J(_{w,b})$

$w = tmp_w$

$b = tmp_b$

}

Linear Regression Model

$$f_{w,b}(x) = wx + b$$

Cost Function

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} (f_{w,b}(x^i) - y^i)^2$$

Gradient Descent Algorithm

repeat until convergence {

$tmp_w = w - \alpha \frac{d}{dw} J(_{w,b})$

$tmp_b = b - \alpha \frac{d}{db} J(_{w,b})$

$w = tmp_w$

$b = tmp_b$

}

If we calculate the derivates of the algorithm, we will get:

$$\frac{d}{dw}J(_{w,b}) \Rightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^i) - y^i)x^i$$

$$\frac{d}{db}J(_{w,b}) \Rightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^i) - y^i)$$

They are derived using calculus

Demonstration:

$$\frac{d}{dw}J(_{w,b}) \Rightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^i) - y^i)x^i$$

def. of derivate

$$\frac{d}{dw}J(_{w,b}) = \frac{d}{dw}\frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^i) - y^i)^2$$

def. $f_{w,b}(x^i) = wx^i + b$

$$\frac{d}{dw}J(_{w,b}) = \frac{d}{dw}\frac{1}{2m}\sum_{i=1}^{m}(wx^i + b - y^i)^2$$

derivation

$$\frac{d}{dw}J(_{w,b}) = \frac{1}{2m}\sum_{i=1}^{m}(wx^i + b - y^i)2x^i$$

simplification

$$\frac{d}{dw}J(_{w,b}) = \frac{1}{m}\sum_{i=1}^{m}(wx^i + b - y^i)x^i$$

def. $wx^i + b = f_{w,b}(x^i)$

$$\frac{d}{dw}J(_{w,b}) = \frac{1}{m}\sum_{i=1}^{m}(f_{b,w}(x^i) - y^i)x^i$$

repeat until convergence {

$tmp_w = w - \alpha\frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^i) - y^i)x^i$

$tmp_b = b - \alpha\frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^i) - y^i)$

$w = tmp_w$

$b = tmp_b$

}