Regresión lineal con gradient descent

Conjunto de datos:

X	Y
1	3
3	4
4	3
7	15
9	17
10	15
11	18
13	27
14	18
16	20

Ecuación lineal por calcular:

$$y \approx f(x) = ax + b = \hat{y}$$

Donde:

- x es la variable independiente (dato)
- y es la variable dependiente (dato)
- $f(x)$ es la función lineal del modelo de ML que es igual a la predicción $\hat{y}$
- Los coeficientes por descubrir son a y b

Primer paso: Determinar la función de pérdida (Loss function). La diferencia entre lo que nos dice la data vs lo que el modelo predice, elevado al cuadrado para evitar que las diferencias negativas cancelen a las positivas.

$$\mathcal{L} = (y - \hat{y})^2$$

Segundo paso: Expandir la función de pérdida para luego calcular las derivadas parciales de la función de pérdida con respecto a a y b

$$\mathcal{L} = y^2 - 2y\hat{y} + \hat{y}^2$$

$$\mathcal{L} = y^2 - 2y(ax + b) + (ax + b)^2$$

$$\mathcal{L} = y^2 - 2y(ax + b) + (ax)^2 + 2axb + b^2$$

$$\mathcal{L} = y^2 - 2yax - 2yb + a^2x^2 + 2axb + b^2$$

Derivada parcial de la función de pérdida con **respecto a $\boldsymbol{a}$**	Derivada parcial de la función de pérdida con **respecto a $\boldsymbol{b}$**
Derivando $$\frac{\partial \mathcal{L}}{\partial a} = -2yx + 2ax^2 + 2xb$$	Derivando $$\frac{\partial \mathcal{L}}{\partial b} = -2y + 2ax + 2b$$

Simplificando $$\frac{\partial \mathcal{L}}{\partial a} = 2x(-y + ax + b)$$ Reordenando $$\frac{\partial \mathcal{L}}{\partial a} = 2x(ax + b - y)$$	Simplificando $$\frac{\partial \mathcal{L}}{\partial b} = 2(-y + ax + b)$$ Reordenando $$\frac{\partial \mathcal{L}}{\partial b} = 2(ax + b - y)$$

Recordando el update rule

$$a_{nuevo} = a - \boldsymbol{\alpha} * \frac{\partial \mathcal{L}}{\partial a}$$

$$b_{nuevo} = b - \boldsymbol{\alpha} * \frac{\partial \mathcal{L}}{\partial b}$$

Donde alfa $\boldsymbol{\alpha}$ es el learning rate, un número que nos indica cuánto modificar los coeficientes en cada iteración.

Podemos simplificar las derivadas eliminando el número 2, ya que puede ser incluido dentro del valor del learning rate $\boldsymbol{\alpha}$. Por otro lado, en la derivada tenemos la función lineal original con la que empezamos $ax + b$, que podemos reemplazar con $\hat{y}$, la predicción.

Por lo tanto, el update rule para a y b quedan de la siguiente forma:

$$a_{nuevo} = a - \boldsymbol{\alpha} * x(\hat{y} - y)$$

$$b_{nuevo} = b - \boldsymbol{\alpha} * (\hat{y} - y)$$

Ejecutando el Gradient Descent:

Inicializamos los valores

$$a = 0, b = 0$$

Tanto x como y son datos, que se extrae de la data. Lo único que tenemos que decidir es si tomamos un solo valor aleatorio, un sub grupo aleatorio o todos. Dependiendo de esto, estaremos ejecutando el algoritmo “stochastic gradient descent”, “mini-batch gradient descent”, o “gradient descent” respectivamente

Pasando todo a matrices.

Para generalizar este algoritmo a n dimensiones, es decir un número desconocido de coeficientes, tenemos que trabajar con vectores y matrices.

$$x \rightarrow \boldsymbol{X}$$

$$y \rightarrow \boldsymbol{Y}$$

$$a, b, c \ldots \rightarrow \boldsymbol{W}$$

$$\hat{y} \rightarrow \widehat{\boldsymbol{Y}}$$

La función lineal pasa a ser

$$XW = \hat{Y}$$

La función de costo:

$$\mathcal{L} = (Y - XW)^2$$

La derivada es similar, pero hay que tener en cuenta el orden de las matrices

$$(\mathrm{XW} - Y)X$$

$$X^T(\mathrm{XW} - Y)$$

Por lo tanto, el update rule pasa a ser:

$$W \leftarrow W - \boldsymbol{\alpha} X^T(\mathrm{XW} - Y)$$

El algoritmo se ejecuta igual que en la versión simple.