Gradient Descent

Prerequisite - OLS, differentiation

OLS - Gives value of m & c using formula. No need for iteration!

$$c = \bar{y} - m\bar{x}$$

$$m = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

} derivation.

where $\bar{y}$ = mean of y; $\bar{x}$ = mean of x.

We want $\frac{\partial J}{\partial c} = 0$ & $\frac{\partial J}{\partial m} = 0$. where $J = \frac{1}{2m} \sum_{i=0}^{n} (y_i - mx_i - c)^2$

So, $\frac{\partial J}{\partial c} \Rightarrow \frac{1}{m} \sum_{i=1}^{n} (y_i - mx_i - c) = 0$.

$$= \bar{y} - m\bar{x} - \frac{nc}{n} = 0 \Rightarrow c = \bar{y} - m\bar{x}$$

Putting this in $\frac{\partial J}{\partial m} = 0$.

$$\frac{\partial \left(\frac{1}{2n} \sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x})^2 \right)}{\partial m} = 0$$

as big matrix multiplication & inverse would have to be done (See OLS for multiple reg)

Disadv. of OLS:- It is computationally expensive with high no. of features. So, grad. descent is introduced.

$$\Rightarrow \sum_{i=1}^{n} \frac{2}{2m} (y_i - mx_i - \bar{y} + m\bar{x})(-x_i + \bar{x}) = 0$$

$$\Rightarrow \sum_{i=1}^{n} -(x_i - \bar{x})(y_i - \bar{y} - m(x_i - \bar{x})) = 0$$

$$\Rightarrow \sum_{i=1}^{n} \left[(x_i - \bar{x})(y_i - \bar{y}) - m(x_i - \bar{x})^2 \right] = 0$$

$$\Rightarrow \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) = m \sum_{i=1}^{n} (x_i - \bar{x})^2$$

$$\Rightarrow m = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

'n' rows of data)

Now, by solving $\frac{\partial e}{\partial \beta}$ $\rightarrow \Sigma(Y-\hat{Y})^2$, we get -

So, $\boxed{\beta = (X^T X)^{-1} X^T Y}$ ← OLS of multipl LR.
(can be used for Simple LR too).

Problem with OLS (Ordinary least squares) - OLS is computationally expensive when high number of features due to more matrix inverse calculation and matrix multiplication involved.

So, Gradient Descent introduced.

Formula -

(how it works ? +ve an -ve slope)
Learning rate - why ? what ?

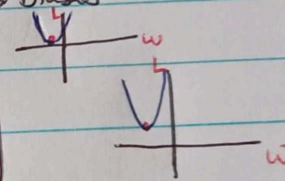
Gradient Descent

$$W = W - \text{learning-rate} \times \frac{dL}{dW}$$

For, mse & logloss L is a function of weights & biases

mse: $L_y = (3-(3w+4))^2$

Logloss: $L_y = -\left[\ln\left(\frac{1}{1+e^{-(4w+4)}}\right) + \ln\left(1-\frac{1}{1+e^{-(4w+4)}}\right)\right]$



It form a parabola & we want to find such value of w where L is minimum (red dot on graphs).

Cost function (MSE)

Solving this

$$\frac{\partial}{\partial\theta_j} J(\theta_0,\theta_1) = \frac{\partial}{\partial\theta_j}\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x)^{(i)} - y^{(i)}\right)^2$$

Differentiation -

When $j=0$,

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_0} \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2$$

$$\frac{\partial J(\theta_0, \theta_1)}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})$$

When $j=1$,

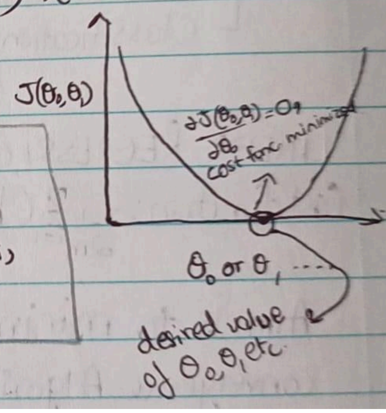
$$\frac{\partial J(\theta_0, \theta_1)}{\partial \theta_1} = \frac{\partial}{\partial \theta_1} \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2$$

$$\frac{\partial J(\theta_0, \theta_1)}{\partial \theta_1} = \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})\, x^{(i)}$$

look out for when cost becomes more or less stable/constant OR do early stopping

So, Repeat until convergence

$$\boxed{\begin{aligned} \theta_0 &= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \\ \theta_1 &= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})\, x^{(i)} \end{aligned}}$$



When to stop?

- When the weights becomes more or less constant(doesn't change with more iterations)
- When total number of iterations decided is reached

VIzualizations - link @58:00 - gradient descent

Link @1:37:00 - visualization (look at the code then watch this)

Link @1:44:00 - effect of learning rate

Types of gradient descents -

Case of Local minima - IN case of non convex functions, Batch gradient descent gets stuck in local minima. So stochastic and mini batch gradient descent used. These are used when the data is 'big' (as it converges faster with less no. of epochs)

Types of Grad. descent.

Grad descent → way to minimize loss function.

① Gradient descent (Batch) = Entire data for updation of wt in one epoch.
1 update in 1 epoch.

② Stochastic GD – wt. update for each data row in each epoch.
no. of eg. row/ ← 'n' update in 1 epoch.

③ miniBatch GD – make small batches, ~~eg~~ say 4 batches, so for each epoch, 4 updation would take place.

Batch.	SGD.
① Time – Faster	① Time – Slower.
② For equal epoch, this will converge slower	② For equal epoch, this will converge to proper solution faster – more updates
③ loss smooth loss decrease	③ Bewda behavior loss
④ Can move in local minima.	④ Helps in moving out of local minima.
⑤ Exact sol.	⑤ Exact solution not found, due to bewda behaviour.