Gradient Descent Linear Regression Derivation

R
ecall from Gradient Descent its an optimization algorithm, in the case of Linear Regression we use the Gradient Descent on the MSE which is the loss function of Linear Regression.

- Unlike the Ordinary Least Squares which compute the coefficients β directly
- Gradient Descent iteratively optimize and update the coefficients till we reach the optimal estimated coefficients β

Why using Gradient Descent?

- On smaller-moderate number of observations n and variables p the OLS method its always better and faster
- On large number of observations and variables calculating the inverse and loading big matrices into memory isn't practical and the computation cost will be higher
- The **Gradient Descent** Solves this problem by optimizing using **iterative** approximations

Derivation

As discuss in Gradient Descent notes the Gradient Descent Algorithm Follows these steps :

- Initialize β our coefficients randomly or to zero
- While β its not converged :
  - Compute the gradient $\nabla MSE(\beta)$
  - Update the coefficients $\beta_{new} = \beta_{old} - \alpha\nabla_{\beta} MSE(\beta)$
  - Check convergence (to break early)
- Return optimized β vector

While taking in mind these **Hyper parameters**

- Learning Rate $\rightarrow \alpha$ which is the step size
- **Batch Size** $\rightarrow$ Number of **training** examples to compute the **gradient** update
- **Epochs** $\rightarrow$ Even tho they are not **hyperparameters**, also called iterations they work closely with **Batch Size -** its one full pass through the dataset called **Epoch** they can be used to cheap and decrease the size of **Batches** while converging faster even if large number of iterations is needed

$$\text{Loss Function} = MSE(\beta) = \frac{1}{2n}\sum_{i}^{n}(y_i - \hat{y}_i)^2$$

- Using matrix form
  Note : $\hat{y} = X\hat{\beta}$

$$MSE(\beta) = \frac{1}{2n}(y - X\hat{\beta})^T(y - X\hat{\beta})$$

- Our goal is to calculate the gradient of the loss function :

$$\Delta MSE(\beta) = \langle\frac{\partial MSE}{\beta_0}, \frac{\partial MSE}{\beta_1}, \dots \frac{\partial MSE}{\beta_{p-1}}\rangle$$

$$\frac{\partial MSE(\beta)}{\partial\beta} = \frac{1}{2n}((y^T - \hat{\beta}^T x^T)(y - x\hat{\beta})) = \frac{1}{2n}(y^T y - y^T x\hat{\beta} - \hat{\beta}^T x^T y + \hat{\beta}^T x^T Y x\hat{\beta})$$

  Note :

$$y^T x\hat{\beta} = (1 \times n)(n \times p)(p \times 1) = 1$$

$$\hat{\beta}^T x^T y = (1 \times p)(p \times n)(n \times 1) = 1$$

$$\frac{\partial MSE(\beta)}{\partial\beta} = \frac{1}{2n}(y^t y - 2\hat{\beta}^T x^T y + \hat{\beta}^T x^T x\hat{\beta})$$

  Now after simplification we derive with respect to β

$$\frac{\partial MSE(\beta)}{\partial \beta} = \frac{1}{n}(-x^T y + x^T x \hat{\beta})$$

- Since β and β^T are symmetric they can be counted as β^2

$$\nabla MSE(\beta) = \frac{1}{n} X^T (X\beta - Y)$$

- By factorizing X^T we get the Gradient of the **Loss Function** for the **Linear Regression** model