Ramtin Behesht Aeen ramtinba145822@gmail.com

Summary of

1 Cost Function for one Dimension

Learning Model:

- Linear Regression Hypothesis:

$$\mathbf{h_w X} = w_0 + w_1 x_1 + ... + w_D x_D := \mathbf{W^T X} \tag{1}$$

- Input Vector X :

$$X = [x_0 = 1,\, x_1,\, x_2,\, \dots, x_D] \tag{2}$$

- Parameter Vector W (features) :

$$W = [w_0 = 1,\, w_1,\, w_2,\, \dots, w_D] \tag{3}$$

Squared Error(SE): Most Common error function in linear regression is:

$$SE : (y^{(i)} - h(x^{(i)}, w))^2 \tag{4}$$

Sum of Squared Errors (SSE): Cost function should measure all predictions. Thus a choice could be Sum of Squared Error(SE)

$$SSE : \sum_{i=1}^{N} (y^{(i)} - h(x^{(i)}, w)) \tag{5}$$

Solve it analytically for one dimension: Predicted:

$$\widehat{y} = w_0 + w_1 x \tag{6}$$

SSE or Cost Function:

$$J(w_0, w_1) := \sum_{i=1}^{N} (y^{(i)} - \widehat{y}^{(i)})^2 = \sum_{i=1}^{N} (y^{(i)} - w_0 + w_1 x^{(i)})^2 \tag{7}$$

Assumptions:

$$\frac{\partial J}{\partial w_0} = 0\,, \frac{\partial J}{\partial w_1} = 0 \tag{8}$$

$\frac{\partial J}{\partial w_0} = 0$: thus:

$$\frac{\partial}{\partial w_0} \Big(\sum_{i=1}^{N} (y^{(i)} - (w_0 + w_1 x^{(i)}))^2\Big) = 0 \text{ [1]} \tag{9}$$

$$-2 \sum_{i=1}^{N} (y^{(i)} - w_0 - w_1 x^{(i)}) = 0 \tag{10}$$

For this equation to equal zero, the following condition must be met:

- $\sum_{i=1}^{N} y^{(i)} = 0 := Y$
- $\sum_{i=1}^{N} -w_0 = 0 := n w_0$
- $\sum_{i=1}^{N} -w_1 x^{(i)} = 0 := X$

Thus:

$$0 = Y - n w_0 - w_1 x^{(i)} \longrightarrow w_0 = \frac{(Y - w_1 x^{(i)})}{n} \tag{11}$$

[1] $fog(x)' = f(g(x))' g(x)'$

Second Part $\frac{\partial J}{\partial w_1} = 0$:

$$\frac{\partial}{\partial w_1}(\sum_{i=1}^{N}(y^{(i)} - (w_0 + w_1 x^{(i)}))^2) = 0 \tag{12}$$

$$\sum_{i=1}^{N} 2(y^{(i)} - (w_0 + w_1 x^{(i)}))x^{(i)} = 0\,[2] \tag{13}$$

$$\sum_{i=1}^{N}(x^{(i)}y^{(i)} - w_0 x^{(i)} + w_1 x^{(i)^2} = 0 \tag{14}$$

$\sum_{i=1}^{N}(x^{(i)}) = X$, $w_0 = \frac{(Y - w_1 x^{(i)})}{n}\, so$:

$$\sum_{i=1}^{N}(x^{(i)}y^{(i)} - \frac{(Y - w_1 x^{(i)})}{n}X + w_1 x^{(i)^2} = 0 \tag{15}$$

$$n\sum_{i=1}^{N} x^{(i)}y^{(i)} - (YX - w_1 X^2) + n\sum_{i=1}^{N} w_1 x^{(i)^2} = 0 \tag{16}$$

$$n\sum_{i=1}^{N} x^{(i)}y^{(i)} - YX = -w_1 X^2 + n w_1 \sum_{i=1}^{N} x^{(i)^2} \tag{17}$$

$$n\sum_{i=1}^{N} x^{(i)}y^{(i)} - YX = w_1(n\sum_{i=1}^{N} x^{(i)^2} - X^2) \tag{18}$$

$$w_1 = \frac{n\sum_{i=1}^{N} x^{(i)}y^{(i)} - YX}{n\sum_{i=1}^{N} x^{(i)2} - X^2} \tag{19}$$

[2] $\frac{\partial}{\partial x}xy = y\frac{\partial}{\partial x}x = y$

2 Cost Function for two oe more Dimension

Analytical Solution for D Dimensions:: In this section, the solution for D dimensions is discussed. For example, in apartment price prediction, we cannot only consider the size, but also other features like the year it was built, the number of rooms, and other property characteristics. Therefore, our x vector will contain multiple properties.

$$x^{(1)} = \begin{pmatrix} 1 \\ 3 \\ 1390 \\ 80 \end{pmatrix} \tag{20}$$

For example in this three dimension, here 1 represents the intercept, 3 is the number of rooms, 1390 is the year it was built, and 80 indicates the size.
original Formula is: $\widehat{y} = w_0 + w_1 x$ but in D dimension will be like so:

$$\underbrace{\begin{bmatrix} 1 & 3 & 1390 & 80 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 2 & 1399 & 160 \end{bmatrix}}_{X(dataofnhouses),eachrowisfeaturesfrom0toD} \times \underbrace{\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{bmatrix}}_{textw} = \underbrace{\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_D \end{bmatrix}}_{y}$$

$$y_1 = w_0 \underbrace{X_{10}}_{\text{row1 column1}} + w_1 \underbrace{X_{11}}_{\text{row1 column2}} + ... + w_D X_{1D} \tag{21}$$

so:

$$\widehat{y} = Xw \tag{22}$$

our goal is to minimize the cost function(make $\widehat{y}$ as close as possible to y): first for multiply this 2 vectors we should transpose one of them:

$$J(w) = (y^{(i)} - \widehat{y}^{(i)})^2 = \underbrace{(y^{(i)} - \widehat{y}^{(i)})^T}_{1*n} \underbrace{(y^{(i)} - \widehat{y}^{(i)})}_{n*1} \tag{23}$$

based on this rule we have:

$$J(w) = (y - Xw)^T (y - Xw) \tag{24}$$

this equal to

$$J(w) = y^T y - y^T X w - w^T X^T y + w^T X^T X w \tag{25}$$

$$J(w) = y^T y - 2 y^T X w + w^T X^T X w \tag{26}$$

differentiate this equation (**??**) with respect to w and set it to zero:

$$\frac{\partial J}{\partial w} = -2X^T y + 2X^T X w = 0$$

[2] $(AB)^T = A^T B^T$

3 Gradient Descent

Gradient Descent:

$$w^{t+1} = w^t - \alpha \nabla J(w^t)$$

Gradient Ascent:

$$w^{t+1} = w^t + \alpha \nabla J(w^t)$$

$$\nabla J(w^t) = \begin{bmatrix} \frac{\partial J(w)}{\partial w_1} \\ \vdots \\ \frac{\partial J(w)}{\partial w_D} \end{bmatrix}$$

- w^t: Current weight vector
- w^{t+1}: Updated weight vector
- α: Learning rate (step size)
- ∇J: Gradient of the cost function

Variation of Gradient Descent:

- **Batch Gradient Descent**: Update weights using all training samples.
- **Stochastic Gradient Descent (SGD)**: Update weights using one training sample at a time.
- **Mini-Batch Gradient Descent**: Update weights using a subset of training samples.

Overfitting: Overfitting occurs when a model learns the training data too well, capturing noise and outliers that are not representative of the underlying pattern. This is like that the model memorizes the training data instead of learning the general pattern.

$$J_{train}(w) << J_{test}(w)$$

Underfitting: In this case model is too simple to capture the underlying pattern of the data.

$$J_{train}(w) \approx J_{test}(w) >> 0$$

4 Perceptron

[A perceptron is a type of artificial neuron that serves as a fundamental building block for more complex neural networks. It is primarily used for binary classification tasks in linear classification problems. Here's a breakdown of how a perceptron works in the context of linear classification:]
1. Structure of a Perceptron: A perceptron consists of: - **Inputs**: Features of the data (e.g., $x_1, x_2, \ldots, x_n$). - **Weights**: Each input is associated with a weight $(w_1, w_2, \ldots, w_n)$. - **Bias**: A bias term (b) that allows the model to fit the data better. - **Activation Function**: A function that determines the output of the perceptron based on the weighted sum of the inputs.

2. Mathematical Representation: The output of a perceptron can be represented mathematically as follows:
1. **Weighted Sum**:

$$z = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b \tag{27}$$

where z is the weighted sum of the inputs.
2. **Activation Function**: The perceptron uses a step function (or Heaviside function) as the activation function:

$$\text{output} = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases} \tag{28}$$

This means that if the weighted sum z is greater than or equal to zero, the perceptron outputs 1 (indicating one class), and if it is less than zero, it outputs 0 (indicating the other class).

3. Training the Perceptron:

$$w_i \leftarrow w_i + \eta(y - \hat{y})x_i \tag{29}$$

$$b \leftarrow b + \eta(y - \hat{y}) \tag{30}$$

4. Linear Classification:

$$w_1x_1 + w_2x_2 + \ldots + w_nx_n + b = 0 \tag{31}$$

5. Limitations:
- **Linearly Separable Data**: The perceptron can only classify data that is linearly separable. If the data cannot be separated by a straight line (or hyperplane), the perceptron will not converge to a solution. - **Single Layer**: A single perceptron cannot solve problems like XOR, which are not linearly separable. However, multiple perceptrons can be combined into multi-layer networks (neural networks) to handle more complex problems.