Contents

Chapter 1

Week 1

1.1 Supervised and Unsupervised Learning

The most basic thing to remember is that we already know what out correct output should look like in Supervised Learning. But, we have little or no idea about what out results should look like.

Supervised Learning:

- Classification: Spam/Not-spam.
- Regression: Predicting age.

Unsupervised Learning:

- Clustering: Grouping based on different variables.
- Non Clustering: Finding structue in chaotic environment.

1.2 Linear Regression with one variable

Regression being a part of Supervised Learning is used for estimatng data (Real-valued output).

1.2.1 Cost Function

This function measures the performance of a Machine Learning model for given data.

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

Cost Function:

$$J(\theta_0, \theta_1) = 1/2m \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{1.1}$$

Goal: Minimize cost function with θ_0, θ_1 as parameters.

1.2.2 Gradient Descent

Basic idea:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we end up at minima.

Algorithm: repeat until convergence:

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta_0, \theta_1)}{\partial \theta_j} \tag{1.2}$$

(for $j = 0, 1$,here).

Intution: If α is too small, descent can be slow and if too large, descent may fail to converge or even diverge. Gradient descent can converge to a local minimum, even with fixed learning rate α. As we approach local mimimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.

1.2.3 Gradient Descent for linear regression

Combining gradient descent algorithm with linear regression model, we get:

$$j = 0 : \frac{\partial J(\theta_0, \theta_1)}{\partial \theta_0} = 1/2 \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \tag{1.3}$$

$$j = 1 : \frac{\partial J(\theta_0, \theta_1)}{\partial \theta_1} = 1/2 \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}).x^{(i)} \tag{1.4}$$

Now, we can repeat 1.3 and 1.4 until convergence to obtain the minima.

"Batch" gradient descent: Each step of gradient descent uses all the training examples. For eq. "m" batches in equation 1.1.

Chapter 2

Week 2

2.1 Multivariate Linear Regression

Linear regression involving more than one variable. For eq., Predicting price of a house based on parameters "Plot Area", "No. of Floors", "Connectivity with markets", etc.

2.1.1 Multiple Features

The multivariable form of the hypothesis is as follows:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + ... + \theta_n x_n. \tag{2.1}$$

This hypothesis funtion can be concisely represented as:

$$h_\theta(x) = \theta^T x \tag{2.2}$$

where, θ^T is a 1xn matrix consisting of $\theta_0, \theta_1, \theta_2 ... \theta_n$.

2.1.2 Gradient Descent for Multiple Variables

Gradient descent formula for Multiple variable will be similar to that of single variable.

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}).x_j^{(i)} \tag{2.3}$$

Repeating this equation until convergence will give the minima. [1]

Feature Scaling

Feature Scaling is used to reduce the number of iterations in Gradient Descent. Basic idea of feature scaling is to bring all the features on the same scale. (in general we try to approximate every feature in the range $-1 < x_i < 1$)

[1] $x_0 = 1$ in equation **??**

Mean Normalisation

Mean Normalisation makes features to have approximately zero mean.

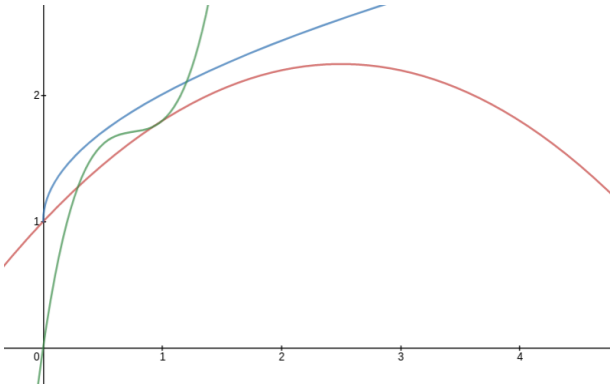
Learning Rate

If α is too small: slow convergence.

if α is too large: $J(\theta)$ may not decrease on every iteration, or may not converge.

Polynomial Regression

Selecting proper polynomial for fitting data is very important.



Red: Quadratic Blue: Square root funtion $\theta_0 + \theta_1 x + \theta_2 \sqrt{x}$

2.2 Normal equation

Normal Equation is a method to solve for θ_T analytically, by creating a $m \times (n+1)$ matrix X and another $m \times 1$ matrix Y.[2]

Mathematically θ is given as:

$$\theta = (X^T X)^{-1} X^t y \tag{2.4}$$

Gradient Descent	Normal Equation
Need to choose α	No need to choose α
Needs many iteration	Don't need to iterate
Works well with large n	Slow for large n

[2]Every element of first column of matrix X is 1 and other are the feature's coefficient

Reasons for non-invertiblity of $X^T X$

- Redundant features (linear dependence) [3]
- Too many features (m <= n)

[3]Eg. Using both m^2 & $(feet)^2$ features