Gradient Descent vs Normal Equation

Tan Ren Jie

January 5, 2018

Contents

1 Introduction

In this section, we would introduce two common optimization methods for a linear regression model, the Gradient Descent and the Normal Equation.

Some basic definitions:

Gradient Descent: Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function

Normal Equation: A method which minimizes the sum of the square differences between the left and right sides

A common cost function used in regression problems is the Mean Squared Error (MSE) function defined below:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i(x_i, \theta) - y_i)^2 \tag{1}$$

where,

$\hat{y}_i$ represents the prediction,

x_i represents the independent variable,

θ represents the model parameters,

y_i represents the dependent variable, a.k.a the ground truth.

In the case for linear regression of multiple variables (features), we can represent $\hat{y}$ in the following matrix representation:

$$\hat{y}_i = \sum_{j=0}^{n} x_{i,j}^T \theta_j \tag{2}$$

where we set $x_{i,0}$ to be 1 to represent the bias term.

By the definition of Total Derivative, we can derive the form of Gradient Descent represented by the following parameters updates:

$$\Delta\theta_j = -\eta \frac{\partial J}{\partial \theta_j} \tag{3}$$

where η is the learning rate.

By iterating, we can then converge to the optimal values of θ that minimizes the cost function, J

For Normal Equation, we define a matrix, $X_{i,j}$, commonly known as the design matrix where i indexes the training sample and j indexes the features. i.e. $X_{10,3}$ refers to the 3^{rd} feature of the 10^{th} training sample. With X, we have the following:

$$\hat{Y}_i = X_{i,j}\theta_j \tag{4}$$

By setting $J = 0$ we get the following form:

$$\theta = (X^T X)^{-1} X^T Y \tag{5}$$

By solving this, we can get the optimal values of θ that minimizes the cost function, J.

2 Comparison

Gradient Descent	**Normal Equation**
Need to choose α	No need to choose α
Needs many iterations	Solve in one step
$O(n^2)$ works better when m is large	Need to compute $(X^TX)^{-1}$ which is $O(n^3)$, slows when m is large
Need to do Feature scaling	No need to do Feature scaling
Don't need to invert X^TX	X^TX might not be able to invert all the time

According to Andrew Ng in the video in Coursera, there are some rare times when X^TX is non-invertible for linear regression models. In these rare times, it is mainly because of the following reasons:

- Redundant features (linearly dependent)
  - E.g. x_1 = size in feet^2
  - E.g. x_2 = size in m^2
  - $\rightarrow x_1 = (3.28)^2 x_2$
- Too many features (e.g. $m \leq n$)
  - Delete some features, or use regularization