Machine Learning - Andrew Ng

Matteo Esposito

November 14, 2018

0 Introduction

This document is a compilation of notes from the machine learning coursera MOOC taught by Prof. Andrew Ng of Stanford University offered by Coursera.

Supervised learning problems are categorized into "regression" and "classification" problems. In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function. In a classification problem, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories.

Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables. We can derive this structure by clustering the data based on relationships among the variables in the data. With unsupervised learning there is no feedback based on the prediction results.

From this point, we will use X to denote the space of input values, and Y to denote the space of output values. In this example, $X = Y = \mathbb{R}$.

To describe the supervised learning problem slightly more formally, our goal is, given a training set, to learn a function $h : X \longrightarrow Y$ so that $h(x)$ is a "good" predictor for the corresponding value of y. For historical reasons, this function h is called a hypothesis.

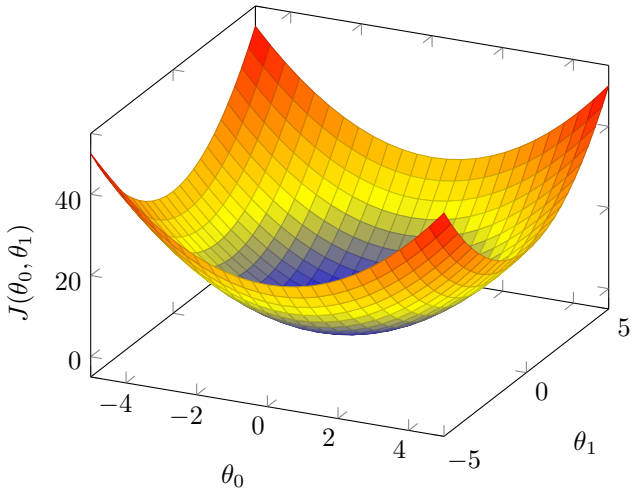
0.1 Cost Function

We can measure the accuracy of our hypothesis function by using a **cost function**. This takes an average difference (actually a fancier version of an average) of all the results of the hypothesis with inputs from x's and the actual output y's.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

This function is otherwise called the "Squared error function", or "Mean squared error". The mean is halved $\left(\frac{1}{2}\right)$ as a convenience for the computation of the gradient descent, as the derivative term of the square function will cancel out the $\left(\frac{1}{2}\right)$ term. J will be 0 if we can achieve a perfect fit (not always possible, and when possible, will probably mean overfitting). We can use a contour plot to visualize the minimizing of the cost function.

Convex Cost Function



0.2 Gradient Descent

Algorithm 1 General Gradient Descent

for $n >= 1$ **do**
  n =number of features, m =number of samples
  repeat until convergence:
    $\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$
end for

We will know that we have succeeded when our cost function is at the very bottom of the pits in our graph, i.e. when its value is the minimum.

The way we do this is by taking the derivative (the tangential line to a function) of our cost function. The slope of the tangent is the derivative at that point and it will give us a direction to move towards. We make steps down the cost function in the direction with the steepest descent. The size of each step is determined by the parameter α, which is called the learning rate.

1 Linear Regression

Algorithm 2 Gradient Descent for Linear Regression

n =number of features, m =number of samples
repeat until convergence:
  $\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y_i)$
  $\theta_1 \leftarrow \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x^{(i)}) - y_i) x_i)$

2 Logistic Regression

3 Neural Networks

4 Neural Networks/Backpropagation