[Summary]Linear Regression with One variable

Model Representation

Recall that in regression problems, we are taking input variables and trying to fit the output onto a continuous expected result function.

Linear regression with one variable is also known as "univariate linear regression."

Univariate linear regression is used when you want to predict a single output value y from a single input value x. We're doing supervised learning here, so that means we already have an idea about what the input/output cause and effect should be.

The Hypothesis Function

Our hypothesis function has the general form:

$$\hat{y} = \mathrm{h}_\theta(x) = \theta_0 + \theta_1 x$$

Note that this is like the equation of a straight line. We give to $\mathrm{h}_\theta(x)$ values for θ_0 and θ_1 to get our estimated output $\hat{y}$. In other words, we are trying to create a function called h_θ that is trying to map our input data (the x's) to our output data (the y's).

Example: Suppose we have the following set of training data:

input x	output y

0	4
1	7
2	7
3	8

Now we can make a random guess about our h_θ function: $\theta_0 = 2$ and $\theta_1 = 2$. The hypothesis function becomes $\mathrm{h}_\theta(x) = 2 + 2x$.

So for input of 1 to our hypothesis, y will be 4. This is off by 3. Note that we will be trying out various values of θ_0 and θ_1 to try to find values which provide the best possible "fit" or the most representative "straight line" through the data points mapped on the x–y plane.

Cost Function

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

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

To break it apart, it is $\frac{1}{2}\bar{x}$ where $\bar{x}$ is the mean of squares of $\mathrm{h}_\theta(x_i) - y_i$, or the difference between the predicted value and the actual value.

This function is otherwise called the "Squared error function", or "Mean squared error". The mean is halved $\left(\frac{1}{2\mathrm{m}}\right)$ as a convenience for the computation of the gradient descent, as the derivative term of the square function will cancel out the $\frac{1}{2}$ term.

Now we are able to concretely measure the accuracy of our predictor function against the correct results we have so that we can predict new results we don't have.

If we try to think of it in visual terms, our training data set is scattered on the x–y plane. We are trying to make straight line (defined by $\mathrm{h}_\theta(x)$) which passes through this scattered set of data. Our objective is to get the best possible line. The best possible line will be such so that the average squared vertical distances of the scattered points from the line will be the least. In the best case, the line should pass through all the points of our training data set. In such a case the value of $J(\theta_0, \theta_1)$ will be 0.

Gradient Descent

So we have our hypothesis function and we have a way of measuring how well it fits into the data. Now we need to estimate the parameters in hypothesis function. That's where gradient descent comes in.

We put θ_0 on the x axis and θ_1 on the asix y, with the cost function on the vertical z axis. The points on our graph will be the result of the cost function using our hypothesis with those specific theta parameters.

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, and the size of each step is determined by the parameter α, which is called the learning rate.

The gradient descent algorithm is:

repeat until convergence:

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

where

j=0,1 represents the feature index number.

Intuitively, this could be thought of as:

repeat until convergence:

$$\theta_j := \theta_j - \alpha[Slope\ of\ tangent\ aka\ derivative\ in\ j\ dimension]$$

Gradient Descent for Linear Regression

When specifically applied to the case of linear regression, a new form of the gradient descent equation can be derived. We can substitute our actual cost function and our actual hypothesis function and modify the equation to

repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (\mathrm{h}_\theta(x_i) - y_i)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((\mathrm{h}_\theta(x_i) - y_i) x_i)$$

}

where m is the size of the training set, θ_0 a constant that will be changing simultaneously with θ_1 and x_i , y_i are values of the given training set(data).

The point of all this is that if we start with a guess for our hypothesis and then repeatedly apply these gradient descent equations, our hypothesis will become more and more accurate.