[Summary]Linear Regression with Multiple Variables

Linear regression with multiple variables is also known as "multivariate linear regression".

We now introduce notation for equations where we can have any number of input variables.

$$x_j^i = value\ of\ feature\ j\ in\ \text{the}\ i^{\text{th}}\ training\ example$$
$$x^i = \text{the}\ column\ vector\ of\ all\ \text{the}\ feature\ inputs\ of\ \text{the}\ i^{\text{th}}\ training\ example$$
$$m = \text{the}\ number\ of\ training\ examples$$
$$n = \left|x^i\right|; (\text{the}\ number\ of\ features)$$

Now define the multivariable form of the hypothesis function as follows, accommodating these multiple features:

$$\mathrm{h}_\theta(x) = [\theta_0 \quad \theta_1 \quad \cdots \quad \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix} = \Theta^{\mathrm{T}}\mathrm{X}$$

This is a vectorization of our hypothesis function for one training example;

The training examples are stored in X row-wise, like such:

$$X = \begin{bmatrix} x_0^1 & x_1^1 \\ x_0^2 & x_1^2 \\ x_0^3 & x_1^3 \end{bmatrix}, \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

You can calculate the hypothesis as a column vector of size (m x 1) with:

$$h_\theta(X) = X\theta$$

Cost function

For the parameter vector θ (of type $\mathbb{R}^{n+1}$ $\mathbb{R}^{(n+1)\times 1}$), the cost function is:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^i\right) - y^i\right)^2$$

The vectorized version is:

$$J(\theta) = \frac{1}{2m}(X\theta - \vec{y})^T(X\theta - \vec{y})$$

Where $\vec{y}$ denotes the vector of all y values.

Gradient Descent for Multiple Variables

repeat until convergence:{

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta\left(x^i - y^i\right)\cdot x_j^i \quad for\ j := 0 \ldots n$$

}

Feature Normalization

We can speed up gradient descent by having each of our input values in roughly the same range. This is because θ will descend quickly on small ranges and slowly on large ranges, and so will oscillate inefficiently down to the optimum when the variables are very uneven.

The way to prevent this is to modify the ranges of our input variables so that they are all roughly the same. Ideally:

$$-1 \leq x_i \leq 1$$

or

$$-0.5 \leq x_i \leq 0.5$$

Two techniques to help with this are **feature scaling** and **mean normalization**. Feature scaling involves dividing the input values by the range (i.e. the maximum value minus the minimum value) of the input variable, resulting in a new range of just 1. Mean normalization involves subtracting the average value for an input variable from the values for that input variable, resulting in a new average value for the input variable of just zero. To implement both of these techniques, adjust your input values as shown in this formula:

$$x_i := \frac{x_i - \mu_i}{S_i}$$

Where μ_i is the **average** of all the values for feature (i) and S_i is the range of values (max – min), or S_i is the standard deviation.

Gradient Descent Tips

Debugging gradient descent. Make a plot with number of iterations on the x–axis. Now plot the cost function, J(θ) over the number of iterations of gradient descent. If J(θ) ever increases, then you probably need to decrease α.

Automatic convergence test. Declare convergence if J(θ) decreases by less than E in one iteration, where E is some small value such as 10^{-3}. However in practice it's difficult to choose this threshold value.

Features and Polynomial Regression

We can improve our features and the form of our hypothesis function in a couple different ways.

We can **combine** multiple features into one. For example, we can combine x_1 and x_2 into a new feature x_3 by taking $x_1 \cdot x_2$.

Polynomial Regression

Our hypothesis function need not be linear (a straight line) if that does not fit the data well.

We can **change the behavior or curve** of our hypothesis function by making it a quadratic, cubic or square root function (or any other form).

For example, if our hypothesis function is $h_\theta(x) = \theta_0 + \theta_1 x_1$ then we can create additional features based on x_1 , to get the quadratic function $h\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$ or the cubic function $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3$

In the cubic version, we have created new features x_2 and x_3 where $x_2 = x_1^2$ and $x_3 = x_1^3$.

To make it a squeare root function, we could do: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 \sqrt{x_1}$

One important thing to keep in mind is, if you choose your features this way then feature scaling becomes very important.

Normal Equation

The "Normal Equation" is a method of finding the optimum theta **without iteration.**

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

There is **no need** to do feature scaling with the normal equation.

Gradient Descent	**Normal Equation**
Need to choose alpha	No need
Needs many iterations	No need
$O(kn^2)$	$O(n^3)$, need to calculate inverse of $X^T X$
Works well wehn n is large	Slow if n is very large

Normal Equation Noninvertibility

$X^T X$ may be **noninvertible**. The common causes are:

- Redundant features, where two features are very closely related (i.e. they are linearly dependent)
- Too many features (e.g. m ⩽ n). In this case, delete some features or use "regularization" (to be explained in a later lesson).

Solutions to the above problems include deleting a feature that is linearly dependent with another or deleting one or more features when there are too many features.