**Logistic Regression:**

**Cost Function:**

J =  (y' \* -log(h) + (1-y)' \* -log(1-h))/m

**Gradient descent:**

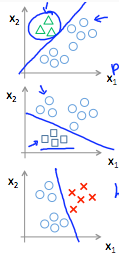
Applying gradient descent to calculate the parameters ‘theta’

:= – \*J

But often in Matlab we optimize the parameters using a built-in function called fminuc to optimize the parameters we pass the cost function, the gradient of the cost function and the initial values of theta.

**Multiclass Classification:**

Instead of working with 2 classes multivariable classification aims to classify the given data into a number of finite classes.

To do this we modify the classic logistic classification such that we classify only one subset at a time making the other subsets the same.

**Over-fitting:**

If there are a lot of features then the hypothesis function will try very hard to accommodate all the points in the dataset. This might cause waviness in the decision boundary.

One of the ways to solve over-fitting is by adopting regularization.

**Regularization:**

Regularization adds the term m; this term helps in reducing the effect of the parameters by increasing the penalty for the cost function. Now during gradient descent the regularization term tends to decrease the numerical value of the parameters thus reducing its effect.

Thus after regularization the cost function looks like:

J =  (y' \* -log(h) + (1-y)' \* (-log(1-h))/m + (sum(theta.^2)-theta(1).^2)\*lambda/(2\*m);