[Summary]Logistic Regression 2

Advanced Optimization

"Conjugate gradient", "BFGS", and "L–BFGS" are more sophisticated, faster ways to optimize θ that can be used instead of gradient descent.

Octave includes these functions, you can set the cost function and gradient descent function then it will automatically choose a function to optimize θ.

First need to provide a function that evaluates the following two functions for a given input value θ:

$$J(\theta)$$

$$\frac{\delta}{\delta\theta_j}J(\theta)$$

We can write a single function that returns both of these:

```
function [jVal, gradient] = costFunction(theta)
  jVal = [...code to compute J(theta)...];
  gradient = [...code to compute derivative of J(theta)...];
end
```

Then we can use octave's "fminunc()" optimization algorithm along with the "optimset()" function that creates an object containing the options we want to send to "fminunc()".

```
options = optimset('GradObj', 'on', 'MaxIter', 100);

initialTheta = zeros(2,1);

[optTheta, functionVal, exitFlag] = fminunc(@costFunction, initialTheta,
options);
```

We give to the function "fminunc()" our cost function, our initial vector of theta values, and the "options" object that we created beforehand.

Multiclass Classification: One-vs-all

Now we will approach the classification of data into more than two categories. Instead of y = {0,1} we will expand our definition so that y = {0,1...n}.

In this case we divide our problem into n+1 (+1 because the index starts at 0) binary classification problems; in each one, we predict the probability that 'y' is a member of one of our classes.

We are basically choosing one class and then lumping all the others into a single second class. We do this repeatedly, applying binary logistic regression to each case, and then use the hypothesis that returned the highest value as our prediction.