Advanced Optimization

Note: [7:35 - '100' should be 100 instead. The value provided should be an integer and not a character

"Conjugate gradient", "BFGS", and "L-BFGS" are more sophisticated, faster ways to optimize θ that can instead of gradient descent. We suggest that you should not write these more sophisticated algorithm: (unless you are an expert in numerical computing) but use the libraries instead, as they're already test optimized. Octave provides them.

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

$$J(\theta)$$

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

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 tha object containing the options we want to send to "fminunc()". (Note: the value for MaxIter should be ar a character string - errata in the video at 7:30)

```
options = optimset('GradObj', 'on', 'MaxIter', 100);
initialTheta = zeros(2,1);
   [optTheta, functionVal, exitFlag] = fminunc(@costFunction, initialThe

```

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

Go to next item ✓ Completed