Programming Ex.2

Note for MATLAB users: If you are using MATLAB version R2015a or later, the fminunc() function has been changed in this version. The function works better, but does not give the expected result for Figure 5 in ex2.pdf, and it throws some warning messages (about a local minimum) when you run ex2_reg.m. This is normal, and you should still be able to submit your work to the grader.

Typos in the lectures (updated):

There are typos in the week 3 lectures, specifically for regularized logistic regression. This could create some confusion while doing the the last part of exercise 2. The equations in ex2.pdf are correct.

Gradient and theta values for ex2.m

Here are the values of both cost J and the gradients for the "initial theta (zeros)" test (ex2.pdf Section 1.2.2):

```
Cost at initial theta (zeros): 0.693147
Gradient at initial theta (zeros):
 -0.100000
 -12.009217
 -11.262842

```

Here are the values for both cost J and theta for the "theta found by fminunc" test (ex2.pdf Section 1.2.3):

```
Cost at theta found by fminunc: 0.203498
theta:
 -25.164593
  0.206261
  0.201499

```

mapFeature() discussion:

For two features x1 and x2, mapFunction calculates following terms.
$1,\ x_1,\ x_2,\ x_1^2,\ x_1x_2,\ x_2^2,\ x_1^3,\ x_1^2x_2,\ x_1x_2^2,\ x_2^3,\ x_1^4,\ x_1^3x_2,\ x_1^2x_2^2,\ x_1x_2^3,\ x_2^4,\ x_1^5,\ x_1^4x_2,\ x_1^3x_2^2,\ x_1^2x_2^3,\ x_1x_2^4,\ x_2^5,\ x$

Not 100% sure about this, so please take this with a grain of salt.

It appears to me that the "mapFeature" vector displayed on page 9 of the ex2.pdf is the transpose of what is intended. Also, it would be more clear if each of the variables carried the (i) superscript denoting the trial

$$mapFeature(x^{(i)}) = \begin{bmatrix} 1 \\ x_1^{(i)} \\ x_2^{(i)} \\ \left(x_1^{(i)}\right)^2 \\ x_1^{(i)}x_2^{(i)} \\ \left(x_2^{(i)}\right)^2 \\ \left(x_1^{(i)}\right)^3 \\ \vdots \\ x_1^{(i)}\left(x_2^{(i)}\right)^5 \\ \left(x_2^{(i)}\right)^6 \end{bmatrix}^T$$

Of course this assumes exactly two features in the original dataset. I think of this more as "mapTrial" than as "mapFeature" because what we're really doing is mapping the original trials with two features onto a new set of trials with 28 features.

I would not have thought twice about this, had I not gulped hard at the imprecise use of the word "dimensions" in the phase, "a 28-dimensional vector" in the text which follows the expression.

This is how I interpreted it for the homework, and the results were accepted. But if I'm way off base, please delete this wiki entry.

==

I found this Octave expression quite useful for the regularization programming exercise:

```
ones(size(theta)) - eye(size(theta))

```

==

I found these other Octave expressions which also are quite useful for the regularization programming exercise:

```
theta(2:size(theta))
theta(2:end)

```

==

plotData.m - color attributes

The plot() attribute "MarkerFaceColor" may not be supported on your version of Octave or MATLAB. You may need to modify it. Use the command "plot help" to see what attributes are supported. (You might just try to replace "MarkerFaceColor" with "MarkerFace", then the plot should work, although you get a warning.)

Logistic Regression Gradient

[w.r.t.=with respect to]

Don't stumble over terminology - "the partial derivatives of the cost w.r.t. each parameter in theta" are:

$\frac{\alpha}{m} X^T(g(X\theta) - \vec{y})$

I was confused about this and kept trying to return the updated theta values . . .

UPDATE (the above was really helpful, thank you for putting it here) As an additional hint: the instructions say: "[...] the gradient of the cost with respect to the parameters" - you're only asked for a gradient, don't overdo it (see above). The fact that you're not given alpha should be a hint in itself. You don't need it. You won't be iterating neither.

Sigmoid function

I saw some confusion on how to implement the sigmoid function in the forum. There are two main areas that might cause trouble.

1) Trying to implement the hypothesis for logistic regression in the sigmoid function:

This exercise has nothing to do with H(X) = (theta' * X). Instead, this exercise requires us to implement a generic sigmoid function that can be used in later exercises. Don't get caught up on logistic regression when implementing your sigmoid function.

2) Handling scalars, vectors and matrices as input.

Since the sigmoid function may take a scalar, vector or matrix, you need to implement a solution that takes all three into consideration. My first thought was to use if statements using the size of the input, but with element-wise operators, you can complete the assignment in one line.

for instance, the following statement will return an error in Octave. 3 / [1 2 3; 4 5 6; 7 8 9]

However, 3 ./ [1 2 3; 4 5 6; 7 8 9] returns [1/3 2/3 3/3; 4/3 5/3 6/3; 7/3 8/3 9/3] (in decimal form, not fractions like I used)

Also, 2^[1 2 3; 4 5 6] fails, but 2.^[1 2 3; 4 5 6] returns [2 4 8; 16 32 64]

So, using element-wise operations works on scalars, vectors and matrices, returning the same type of element that was operated on. 2.^(scalar) returns a scalar, 2.^[vector] returns a vector and x.^[matrix] returns a matrix.

You could also use arrayfun in Octave to achieve the same result. I wonder which is a faster implementation.

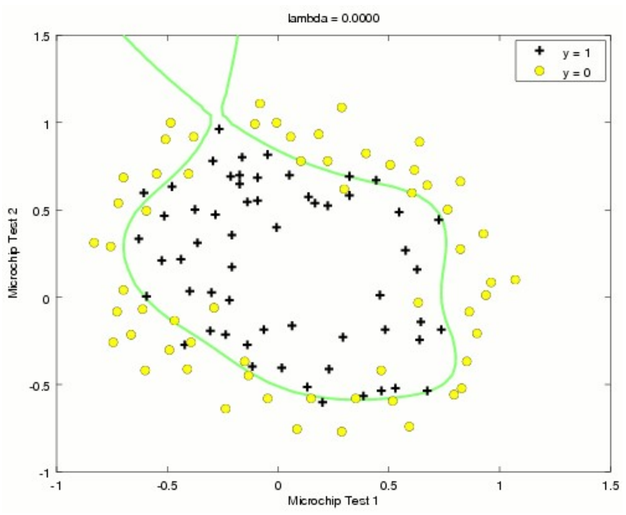
Decision Boundary

Thoughts regarding why the equation, $theta_1 + theta_2 x_2 + theta_3 x_3$, is set equal to 0 for determining a decision boundary:

In this exercise, we're solving a **classification** problem using logistic regression.

- The hypothesis equation is $h_\theta(x) = g(z)$, where g is the sigmoid function $\frac{1}{1+e^{-z}}$, and $z = \theta^T x$
- For classification, we usually interpret a hypothesis value $h_\theta(x) \geq 0.5$ as predicting class "1"
- Remember, $h_\theta(x) = g(z) = g(\theta^T x)$ for logistic regression
- This means that $g(\theta^T x) \geq 0.5$ predicts class "1"
- The sigmoid function g(z) outputs ≥0.5 when z≥0 (look at a graph of the sigmoid function)
- Remember, $z = \theta^T x$
- So, $\theta^T x \geq 0$ predicts class "1"
- Remember $\theta^T x = \theta_1 + \theta_2 x_2 + \theta_3 x_3$ in this example (using 1-indexing)
- So, $\theta_1 + \theta_2 x_2 + \theta_3 x_3 \geq 0$ predicts class "1"
- The decision boundary lets us see the line that has been learned in order to separate out the y=0 vs y=1 classes, in this example
- This boundary is at $h_\theta(x) = 0.5$ (remember, this is the lowest possible value for predicting that a class is "1")
- So, $\theta_1 + \theta_2 x_2 + \theta_3 x_3 = 0$ is the boundary
- The decision boundary will be a line composed of **any** (x2,x3) points that make this equation **equal zero**.
- In order to plot the line along the specific data we have, we arbitrarily decide to use values of x_2 from our data, by choosing the max and min, and then add/subtract a little bit in order to make the line fit nicely. Think about it, you could continue down the line in the above equation an infinite amount in either direction, and it will still be the line dividing the two classes. However, we only have data that lies around a certain area of this line, so we make sure to only plot the line and data in that region (otherwise it would just be a line and some blank space around it).
- Solve for x_3 since we're using x_2 values (the max & min values +/- 2 in order to make a nice line). --> $x_3 = \frac{-1}{theta_3} * (theta_2 x_2 + theta_1)$, as seen in the Octave function.
- Plugin the two x_2 values (stored in plot_x) into the above equation to get the two corresponding x_3 values (and store in the plot_y variable).
- Plot a line using these values -> this will be the decision boundary.
- Plot the rest of our data on the graph as well, and notice that the line should separate the classes.
- The above still applies even if you're using higher-order polynomial features, with the note that instead of a decision boundary "line", it will be a decision boundary "polynomial".

Lambda effect over Decision Boundary


lambda = 0.0000
y = 1
y = 0
Microchip Test 1
Microchip Test 2
1.5
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
1.5