**Machine Learning Programming Assignment 2**

**Comp540 Spring 2015 due 6 February 2015 at 8 pm**

**Miao Wang & Haoyue Zhang (mw56&hz32)**

**Problem 1: Logistic regression (15 points)**

**Visualizing the dataset**

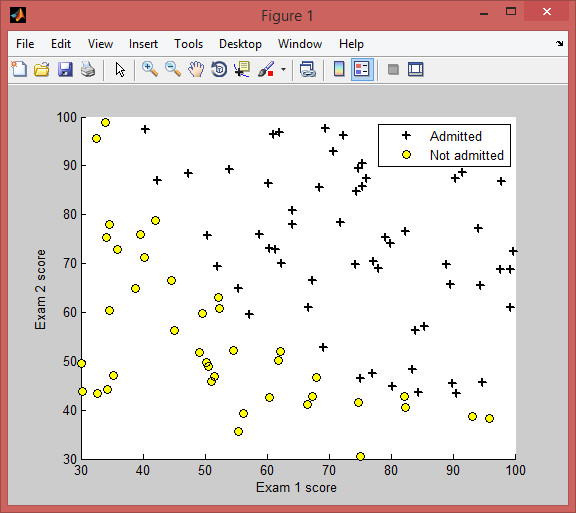


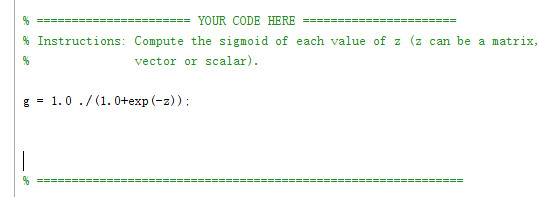
Figure 1: The training data

**Implementing logistic regression: the sigmoid function (5 points)**

The sigmoid function is defined as:

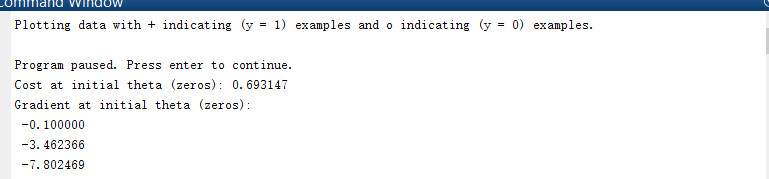
*g*(*z*) =

1 + *e−z*



**Cost function and gradient of logistic regression (5 points)**

Note that while this gradient looks identical to the linear regression gradient, the formula is actually different because linear and logistic regression have different definitions of *hθ* (*x*). Once you are done, ex2.m will call your costFunction using the initial parameters of *θ*. You should see that the cost is about 0.693.



**Learning parameters using fminunc**

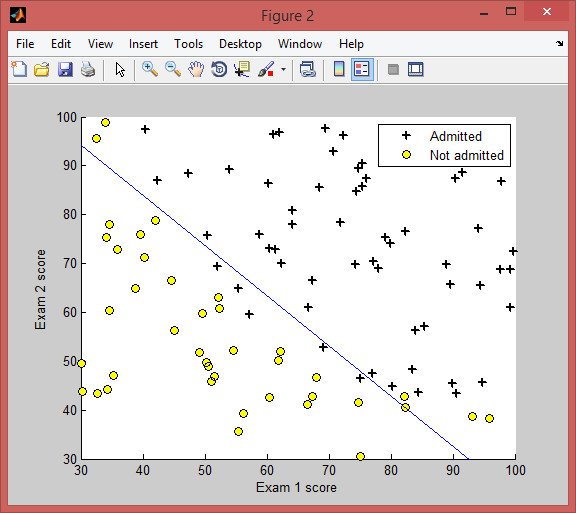
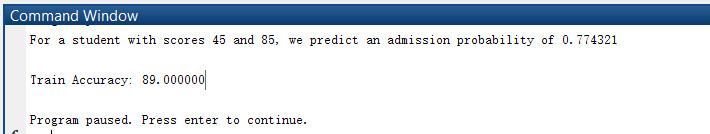


Figure 2: The decision boundary

**Evaluating logistic regression (5 points)** 5

After learning the parameters, you can use the model to predict whether a particular student will be admitted. For a student with an Exam 1 score of 45 and an Exam 2 score of 85, you should expect to see an admission probability of about 0.774. Another way to evaluate the quality of the parameters we have found is to see how well the learned model predicts on our training set. In this part, your task is to complete the code in predict.m. The predict function will produce 1 or 0 predictions given a dataset and a learned parameter vector *θ*. After you have completed the code in predict.m, the ex2.m script will proceed to report the training accuracy of your classifier by computing the percentage of examples it got correct. You should expect to see 89% accuracy on the training data.



**Problem 2: Regularized logistic regression (15 points)**

In this part of the exercise, you will implement regularized logistic regression to predict whether microchips from a fabrication plant pass quality assurance (QA). During QA, each microchip goes through various tests to ensure it is functioning correctly. Suppose you are the product manager of the factory and you have the test results for some microchips on two different tests. From these two tests, you would like to determine whether the microchips should be accepted or rejected. To help you make the decision, you have a dataset of test results on past microchips, from which you can build a logistic regression model. You will use another script, ex2 reg.m to complete this portion of the exercise.

**Visualizing the data**

Similar to the previous parts of this exercise, plotData is used to generate a figure like Figure 3, where the axes are the two test scores, and the positive (y = 1, accepted) and negative (y = 0, rejected) examples are shown with different markers. Figure 3 shows that our dataset cannot be separated into positive and negative examples by a straight-line through the plot. Therefore, a straightforward application of logistic regression will not perform well on this dataset since logistic regression will only be able to find a linear decision boundary.

**Feature mapping**

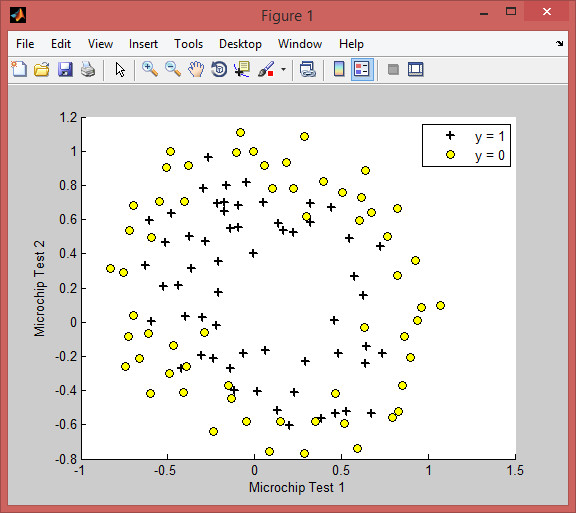


Figure 3: Plot of training data

**Cost function and gradient (10 points)**

**Plotting the decision boundary** 7

To help you visualize the model learned by this classifier, we have provided the function plotDecisionBoundary.m which plots the (non-linear) decision boundary that separates the posi- tive and negative examples. In plotDecisionBoundary.m, we plot the non-linear decision boundary by computing the classifiers predictions on an evenly spaced grid and then draw a a contour plot of where the predictions change from y = 0 to y = 1. After learning the parameters *θ*, the next step in ex2 reg.m will plot a decision boundary similar to Figure 4.

**Varying** *λ* **(5 points)**

In this part of the exercise, you will get to try out different regularization parameters for the dataset to understand how regularization prevents overfitting. Notice the changes in the decision boundary as you vary *λ*. With a small *λ* (say 0), you should find that the classifier gets almost every training example correct, but draws a very complicated boundary, thus overfitting the data. With a larger *λ*, you should get a simpler decision boundary which still separates the positives and negatives fairly well. However, if *λ* is set to too high a value (say 100), you will not get a good fit and the decision boundary will not follow the data so well, thus underfitting the data. Show plots of the decision boundary for two lambdas showing overfitting and underfitting on this data set.

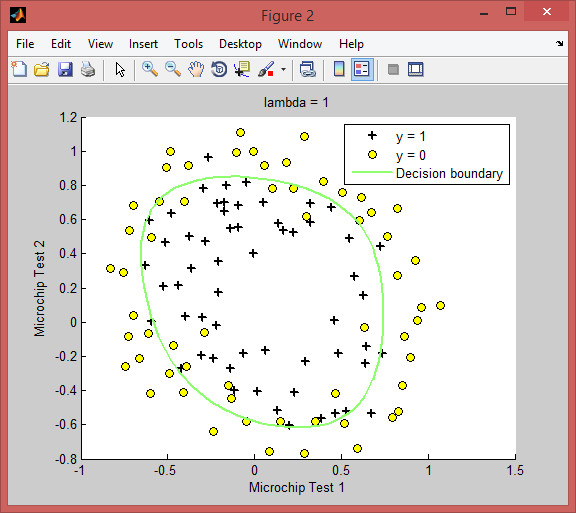


Figure 4: Training data with decision boundary for lambda = 1

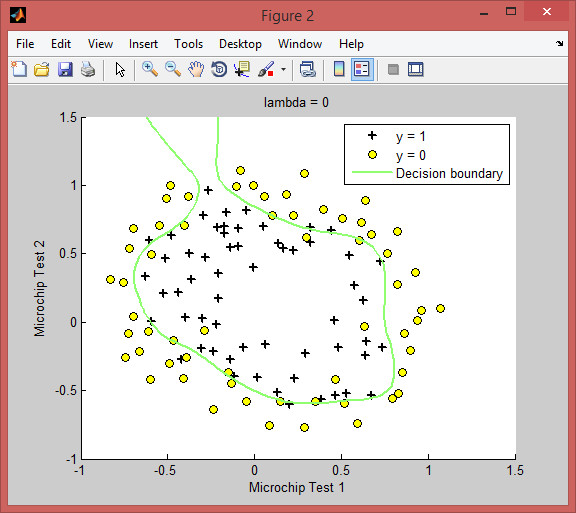


Figure 5: Training data with decision boundary for lambda = 0

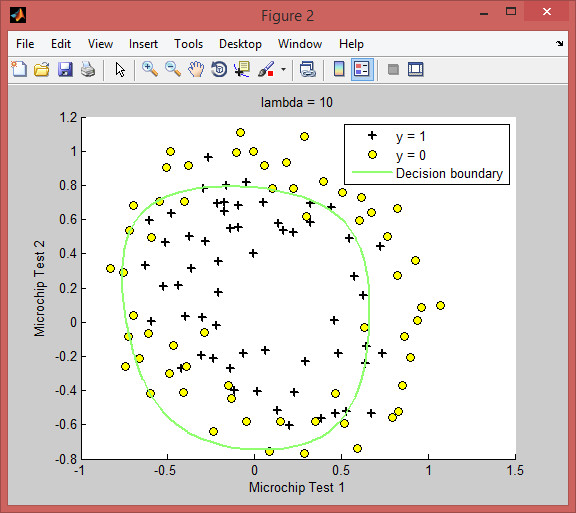
****

Figure 6: Training data with decision boundary for lambda = 10

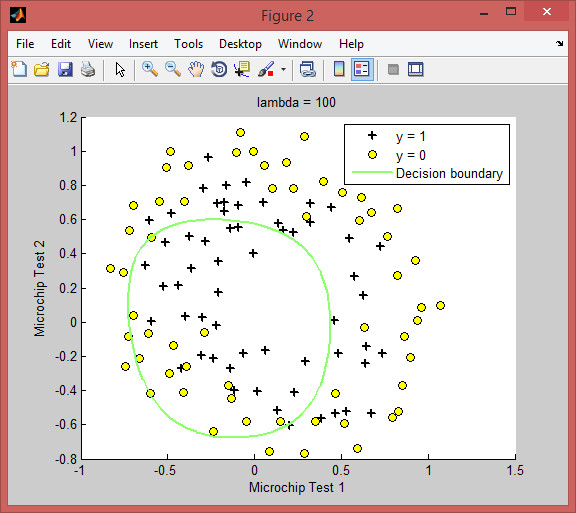


Figure 7: Training data with decision boundary for lambda = 100

The lambad = 0 is showing overfitting

The Lambad = 100 is showing underfitting

**Problem 3: Logistic regression for spam classification (20 points)** 8

(Source: Kevin Murphy) Consider the email spam data set developed by Hastie et. al. It has 4601 email messages, from which 57 features have been extracted. This data is in spamData.mat which has a training set of size 3065 and a test set of size 1536. The features are as follows:

*•* 48 features in [0,100], giving the percentage of words in a given email which match a given word on the list. The list contains words such as ”business”, ”free”, ”george”, etc. The data was collected by George Forman, so his name occurs quite a lot.

*•* 6 features in [0,100], giving the percentage of characters in the email that match a given character on the list. The characters are ;, (, [, !, $, #.

*•* Feature 55: the average length of an uninterrupted sequence of capital letters (max is 40.3, min is 4.9).

*•* Feature 56: the length of the longest uninterrupted sequence of capital letters (max is 45.0, mean is 52.6).

*•* Feature 57: the sum of the lengths of uninterrupted sequence of capital letters (max is 25.6, mean is 282.2).

**Feature transformation (6 points)**

Scaling features is important in logistic regression. Here you will implement three methods for trans- forming features in the spam data set: stdFeatures, logTransformFeatures and binarizeFeatures. There are *.m* files with these names in our directory and you will complete the code in them. Here are the descriptions of the transformations.

*•* Standardize features: transform each column of the data set by subtracting the mean of the column and dividing by the standard deviation. Thus each column has a mean of zero and unit variance.

(*i*)

(*i*)

*•* Log transform features: replace every *xj* by *log*(1 + *xj* ).

(*i*)

(*i*)

*•* Binarize features: replace every *xj* by 1 if *xj >* 0 or 0 otherwise.

**Fitting regularized logistic regression models (14 points)**

For each representation of the features, we will fit regularized logistic regression models. Your task is to complete the function select lambda crossval to select the best regularization parameter *λ* by 10-fold cross-validation on the training data. This function takes a training set X and y and sweeps a range of *λ*’s from lambda low to lambda high in steps of lambda step. Default values for these parameters are in ex2 spam. For each *λ*, divide the training data into ten equal portions using the Matlab function crossvalind. Train a logistic regression model on nine of those parts and test its accuracy on the left out portion. The accuracy of a model trained with that *λ* is the

average of the ten test errors you obtain. Do this for every *λ* in the swept range and return th9e lambda that yields the highest accuracy .

ex2 spam will then build the regularized model with the best lambda you calculate and then deter- mine the training and test set accuracies of the model. With features generated by stdFeatures you should see an accuracy of 91.8% on training data and 91.3% on test data. With logTransformFeatures you should get training data accuracy of 94.8% and test set accuracy of 93.9%. With binarizeFeatures you should see training set accuracy of 93.5%and a test set accuracy of about 92.6%.

**What to turn in**

Please zip up all the files in the archive (including files that you did not modify) and submit it as pa2 netid.zip on Owlspace before the deadline. Include a PDF file in the archive that presents your plots and your discussion of results from the problems above.

**Acknowledgment**

Problems 1 and 2 are adapted from Andrew Ng’s exercise on logistic regression.