COSC 190 Final Exam

Dr. Fred Park

Whittier College

Thursday May 4th, 2017

Name:_____

Student ID#:_____

Note: For full credit you must show all work! Only your class notes from COSC 190, Matlab, and instructor provided files can be used. No other resources allowed unless specified by the instructor.

1. **Machine Learning, 2 Label 2 Feature Classification:** Use your dataset that you have found that has at least 30 training examples. Here, you will break your data up into 2 parts: 1. training set and 2. a testing set. You will then apply Logistic Regression (LR) to the training data to obtain a decision boundary and then use that boundary to classify the testing set. Output all plots and classification accuracy as percentage of correctly classified points in both training and testing cases.

   - `lr_2label_2feature_cg_final.m` (Logistic Regression Script for 2 labels and 2 features)
   - `fmincg.m` (Conjugate Gradient Method called by the LR script.)

   If you have data entered as column separated entries in a text file with 1st column = feature 1, 2nd = feature 2, and last = label, then you can simply load in the data via the command
   `data = load(my_data.txt)';`
   Then your data will be loaded as a matrix of size [M,N] where M-1: # features and N: #data examples. Otherwise, you can type the following in the command line to eventually save the matrix data as `my_data.mat`. Then you can simply load it in the script by typing: `load my_data.mat` and the matrix `data` will be loaded. Follow the steps below:

   (i) Type: `feat1 = [x11 x12 x13 x14 ... x1N]` where `x1i` for i = 1,...,N are the first features for your data.

   (ii) Type: `feat2 = [x21 x22 x23 x24 ... x2N]` where `x2i` for i = 1,...,N are the second features for your data.

   (iii) Type: `label = [l1 l2 l3 l4 ... lN]` where `li` for i = 1,...,N are the labels for your data.

   (iv) Type: `data = [feat1; feat2; label];` which creates the data matrix.

   (v) Save the file as `my_data.mat` via typing: `save my_data.mat data` which saves the data file as a .mat file.

   (vi) Load the data file by having the command: `load my_data.mat` in the script.

   Now, use the `randperm` function to permute the indices of your data matrix. Then break up your data into 2/3 training and 1/3 testing as described in the script `lr_2label_2feature_cg_final.m`. Thus run training on 2/3 of your data to obtain a decision boundary and use it to classify the remaining 1/3 data. How well does the method work? Describe why or why not. Make sure to include all code and figures. Note: If you use the exam scores data from class for this problem, you will receive zero credit.

2. **Cross Validation via Repeated Random Sub-Sampling:** To make sure your classification decision boundary generalizes, you will need to perform cross validation. Follow the steps below:

   (a) Use the script `lr_2label_2feature_cg_final.m` and save it as `lr_2label_2feature_cg_Cross_Valid_final.m`

   (b) Write an outer loop in the script that upon each iteration, randomly selects 2/3 of the data for testing and 1/3 for training during each iteration. During each iteration, save the training and classification accuracy in 1×5 vectors `Train_A` and `Test_A`. Do this for 5 runs. You will see that the training and testing accuracies vary during each iteration. If there are large deviations in either one, then either you need to add more features to your data, or your data may not be linearly separable. What is the average training and testing accuracy after 5 repeated random sub-sampling runs?

   (c) Repeat (a) except now by splitting the data into half training and half testing. How much does your average accuracy change in both training and testing?

   (d) Repeat (a) except now by splitting the data into 90% training and 10% testing. How much does your average accuracy change in both training and testing?

3. **Optical Character Recognition, Classifying Handwritten Digits from the MNIST Database:** Here you will see if you can teach the computer to see the difference between a handwritten #3 and #5.

   (a) Load the MNIST data into your workspace by typing the command:
   `load mnist_all.mat` where if you type `whos` you will see
   test0, test1, ..., test9, train0, train1, ... train9 in the workspace. These correspond to training and testing images. For example: if you type `t3 = test3(1,:);` you will see that t3 is a vector of size 1×784. This is no coincidence as $28 \times 28 = 784$. Type: `t3r = reshape(t3,28,28)';` and `figure(1); imagesc(t3r); colormap(gray(256));` and you will now see a beautifully handwritten '3' image. Reshape simply reshapes the vector back into an image. If you repeat the same except use the second data item from test5, you will see a handwritten '5' image.

   (b) Using the save command above in problem (1), you will create a `data_train` matrix of size [785, 11552] that contains the 3's and the label 1 as the first 6131 columns and the 5's and respective 0 label as the remaining 5421 columns. Note, the very last row will be the labels for each column, i.e. label 1 for '3' and label 0 for '5'. Save it as `data_train_3vs5.mat` so that when this is called to be loaded in the script `LR_2label_Mfeature_CG_MNIST_final.m` script, the correct `data_train` matrix is loaded. Do the same for the testing data where you will create a matrix `data_test` of size [785, 1902] that contains the first 1010 columns as the test3 data along with the label 1. The remaining columns need to be test5 with label 0. and save it as `data_test_3vs5.mat` so that it can be loaded in the `LR_2label_Mfeature_CG_MNIST_final.m` script.

   (c) Repeat (b) except with the digits '5' and '8' from the MNIST dataset.

4. (extra credit) **Multi-class Logistic Regression:** Can you think of a way to do multi-class classification. e.g. three class classification. What if you want to train a classifier to tell the difference between a '3', a '5', and a '7'? How would you do this? Can you implement this? A one versus all approach may be in order. i.e. treat the '5' and '7' together as a single class and then train '3' vs that class to create a classifier and repeat with '3' and '5' vs 7 etc. Use the MNIST data for the '7' digit along with the data for '3' and '5' and run it.