HW1: KNN and Cross Validation

February 11, 2014

DUE DATE: 20 February 2014 Thursday, at 23:50

TAs' Note: This exercise was taken from Prof. Kevin Murphy's machine learning class, which in turn seems to have been inspired by the seminal book on the subject 'Elements of Statistical Learning'.

Matlab resources: KU students do have MATLAB access. Stephen Boyd's MOOC class on convex optimization has a list of links to videos that demonstrate basic MATLAB commands. (There are a lot of resources but these videos are, I think, particularly helpful. I suggest you review those to brush up or get up to speed on your MATLAB.)

TAs' Notes on file submission:

1. Please have a single zip file that contains all your files that form a complete solution to this assignment - with all the dependencies in a single folder.
2. Please have one script that generates plots and answers for every problem. Use the matlab command 'figure' to generate a new plot window.
3. Have a file titled 'README' for all the numeric answers to the problems plus all the explanations/clarifications you want to add for extra measure. Name your zip file.

   `<lastname>_<firstname>_hw1.zip`

   For instance,

   `smith_john_hw1.zip`

   We'd like you to put all the relevant code in only one file so that it's easier to check.
4. Upload your zip file to the assignment link at `https://courses.ku.edu.tr/comp541`. You should not need to hand in any work on paper.

1 KNN

In this homework, we will learn how to plot data in Matlab, how to apply a kNN classifier, and how to use cross-validation to select k. All the figures referenced are at the end of the exercise.

1. The file "knnClassify3CTrain.txt" contains 200 rows and 3 columns (separated by a space). The first 2 columns contain the input features, the last column contains the class label. Read this file using dlmread and create matrices Xtrain and ytrain. Similarly convert "knnClassify3CTest.txt" into Xtest and ytest. Turn in your code.

2. Plot the training data so that points in class 1 are red +'s, and points in class 2 are blue *'s and points in class 3 are green x's. The result should look like Figure 1. Turn in your code and plot. **(10 points)**

3. You are provided a function:

   ```
   [ypred] = knnClassify(Xtrain, ytrain, Xtest, K);
   ```

   that classifies each row of Xtest using the K-nearest neighbor algorithm. Apply this function to the test set using $K = 1$. Plot the test data with their predicted labels using the colors/ symbols above. Put a black circle around any points that are incorrectly classified. The result should look like Figure . How many errors did your classifier make? Turn in your code and plot. **(10 points)**

4. To visualize the prediction function $\hat{y} = f(x)$, we can apply it to a dense grid of test points x. The provided function makeGrid2d, which uses meshgrid, creates such a set of test points. Classify these test points and plot the result as follows:

   ```
   XtestGrid = makeGrid2d(Xtrain);
   ypredGrid = knnClassify(Xtrain, ytrain, XtestGrid, K);
   plotLabeledData(XtestGrid, ypredGrid) % you must implement this
   ```

   Do this for $K \in \{1, 5, 10\}$. The results should look like Figure 2. Turn in your plots. (We see that as K increases, the decision boundary tends towards a straight line, which is in fact optimal in this case (as we will see later), since the data was generated from a mixture of Gaussians.) **(20 points)**

5. Now compute the error rate on the training and test sets for $K \in \{1, 2, \ldots, 20\}$. Plot the error rate vs the degrees of freedom, N/K. Use a log scale for the x-axis. The result should look like Figure 3. Also plot the training and test error vs K. The result should look like Figure 3. What is the best K? Turn in your code and plots. **(30 points)**

6. In real applications, we don't have access to a test set to choose K. Instead we will use 5-fold cross-validation to pick K. You can use the provided function Kfold to compute the indices for each fold. Use the following code fragment:

```
nfolds = 5;
[trainfolds, testfolds] = Kfold(Ntrain, nfolds);
Ks = [1:20];
for k=1:length(Ks)
    K = Ks(k);
    for i=1:nfolds
        XtrainFold = Xtrain(trainfolds{i},:);
        ytrainFold = ytrain(trainfolds{i});
        XtestFold = Xtrain(testfolds{i},:);
        ytestFold = ytrain(testfolds{i});
        [ypred] = knnClassify(XtrainFold, ytrainFold, XtestFold, K);
        errorRateFold(k,i) = ??? % Fill in here
    end
end
```

Plot the mean error rate vs K. Also plot the standard error, $s_e = \frac{\sigma}{\sqrt{N}}$, using the errorbar command. The result should look like Figure 3. The key point is that the CV curve has the same shape as the test curve. Turn in your code and plots. **(30 points)**

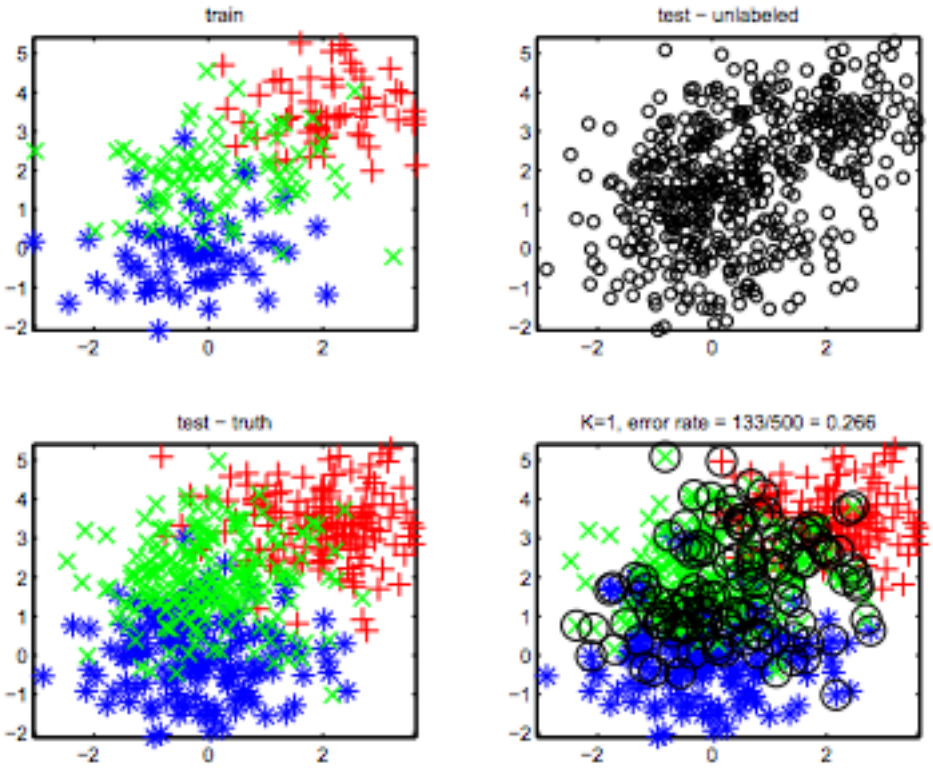

Figure 1: Data

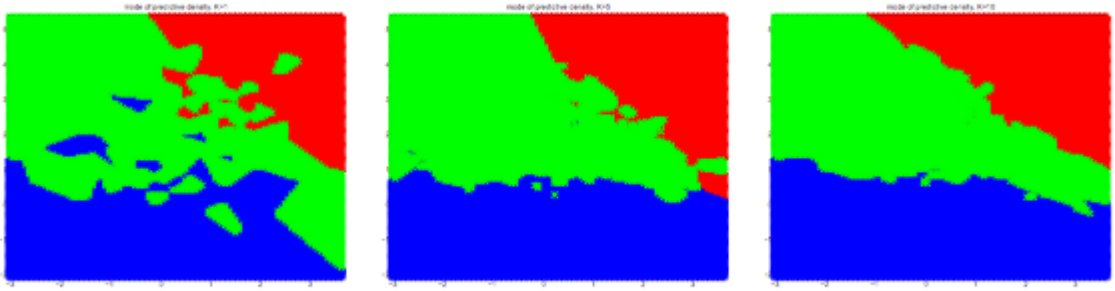

Figure 2: Predictive function for $K = 1$, $K = 5$ and $K = 10$