16:137:551 Advanced Analytics and Practicum

Homework 1
Due: June 27th at 11:55pm

140 Points plus 20 Points Extra Credit

1. Modeling Algorithms (100 points)

In this homework, you will explore some common data modeling tasks and algorithms in Matlab. You will use a sample dataset called "HW1Dataset.mat." This dataset has data from an accelerometer measuring 23,040 time periods with 8 features to predict an outcome. There are 8 features; if you know about analyzing time series data (not necessary for this assignment) the features are as follows:

- The first two features represent the "moving average" of the signal over the time period
- The next three features are wavelet analyses (specifically Daubechies level 5 wavelet transform) on the x, y and z axis of an accelerometer.
- The next three features are the results of a power spectrum density analysis on each of the three accelerometer axis.

The variable labels is 1 for the positive class (that we are trying to predict) and 0 for the negative class.

The variable cvind is a random assignment of each data point into one of 10 buckets for a 10-fold cross-validation analysis.

You will compare the performance of six modeling algorithms for classifying this data;

1. Logistic Regression (LR)
2. Gaussian Naïve Bayes (GNB)
3. Linear Discriminant (LD)
4. Support Vector Machine (SVM) with Linear Kernel
5. SVM with RBF (non linear) kernel
6. Artificial Neural Network (ANN)

For each algorithm, you will evaluate performance via 10-Fold cross validation. This means that you will train and test 10 different models for each algorithm. In each run, you will train on 9/10 of the data and test on 1/10 of the data, with a different 1/10 for the test set in each of the cross validation folds. At the end you will report the average total accuracy, sensitivity and specificity across both training and test sets for each algorithm.

1.1 Load the data "HW1Dataset.mat" into Matlab.

1.2 Standardize the features so they have zero mean and unit variance (ie subtract the feature mean and divide by the standard deviation).

1.3 In each of the 10 cross validation folds, build a logistic regression model using the function mnrfit. You may have to change the class labels for logistic regression. I suggest 1 for negative class and 2 for the positive (hence the old label 1 becomes 2 and the old label 0 becomes 1).

Essentially, the *model* you've built is a multinomial logistic regression polynomial function, where mnrfit() has calculated the coefficients that determine the decision boundary. Once you have a model (coefficents matrix returned by mnrfit), apply it back on the training data using the Matlab function mnrval.

mnrval will return a Nx2 matrix, with the first column the probability that the instance on a row is class 1 (Negative) and the second column with the probability that the instance on a row is class 2 (Positive). For each instance/row, assign the class with the higher probability (ie if the probability of class 1 is 0.9708 and class 2 is 0.0297, the instance belongs to class 1 or Negative). Now you have training predictions.

You can calculate accuracy (aka CorrectRate) as well as sensitivity and specificity using the Matlab function classperf().

1.4 Apply the trained logistic regression model to the test data using the function mnrval. You will have to again choose the class with the highest probability as outlined above in 1.3.

You can calculate the validation performance also using classperf.

1.5 Next explore the Gaussian Naïve Bayes Model using the Matlab function NaiveBayes.fit. Train and test 10 different models across the 10-fold cross validation data splits and report average training and test accuracy, sensitivity and specificity.

Apply the trained Naïve Bayes model to both the training and test data using the Naïve Bayes predict function.

As a note, you may have issues using classperf() with the output of the predict function. Converting that output to a double format from a logical should fix the issue.

1.6 Next explore the results of Linear Discriminant analysis. Use the Matlab function classify () to train a model and apply it to the test data 10 times in 10-fold cross validation.

1.7 Fit a Support Vector Machine Model with a linear kernel using the Matlab function svmtrain.

Given the size of our dataset, it would be good to use the SMO optimization method with a large number of iterations for faster modeling building. Depending on your version of Matlab, that may the default option or it may not be. Additionally, depending on your version of Matlab, the autoscale parameter may be set to true or not. We have already normalized the data, so the parameter should be set to false. To that end, if you have problems, it is suggested to use the following commands:

```
smoopt=svmsmoset('Maxiter',50000);
svm=svmtrain(data,labes,'autoscale','false','Method','SMO','SMO_Opts',smoopt)
;
```

You are welcome to read the documentation of these functions as well as related SVM theory papers/books to understand what is going on here.

Apply the trained SVM model to the training and test data with the function svmclassify. Train and test models via 10-fold cross validation and report average results.

1.8 Repeat 1.7, except this time use the non-linear RBF SVM kernel.

1.9 Apply an ANN model to the data. There are a variety of Matlab functions you can use. One of the simplest ones is newff(). There is also a GUI interface you could use. You can use a simple neural network with 3 layers and nodes running for 200 epochs. If using newff(), use the train() function to train the ANN, and then apply to both the training and test data with the sim() function. The outputs of the ANN may not be exactly 0 or 1, so round to the closest. Apply via 10-fold cross validation and report average training and test results.

1.10 Rerun the ANN in 1.9, but this time on the original non-normalized data (ie before you did the feature standardization in 1.2). Report the results. What happens?

1.11 Compare and contrast the methods. Not just in terms of overall accuracy, but also the sensitivity and the specificity. How about complexity of use and number of parameters to tune?

CAUTION: Don't make assumptions of how the algorithms will work on any dataset based on this homework. Each algorithm has strengths and weaknesses for different types of data. What this homework is designed to do is to give you experience in running the code for each in Matlab.

To summarize your results as you progress, the following accuracy table is provided.

HW #	Development Approach	TR Accuracy	TR Sensitivity	TR Specificity	TST Accuracy	TST Sensitivity	TST Specificity
1.3-1.4	Logistic Regression						
1.5	Naïve Bayes						
1.6	Linear Discriminant						
1.7	Linear SVM						
1.8	Non-linear (RBF) SVM						
1.9	ANN						
1.10	ANN – non normalized data						

Submit your Matlab code, and a written report of your results.

Extra Credit (20 points):
Redo the analysis 1.3-1.11 two more times, each time with a different cross-validation split of the data. What happens? What are the results and how do they compare with the original split given to you?

You can use the Matlab function crossvalind() to generate new 10-Fold cross validation splits of the data.

2. Feature Selection (40 points)

Now you will implement a forward feature selection approach. You will have to implement the algorithm yourself, you can't use built-in Matlab feature selection functions (or those belonging to any other toolbox) such as sequentialfs(), or stepwise() or stepwisefit().

Load in the dataset called HW1Part2Dataset. This also has data, labels and cvind.

In this part, you will use the email SPAM data set referenced in the well-known machine learning book The Elements of Statistical Learning Theory (http://statweb.stanford.edu/~tibs/ElemStatLearn/). It consists of 4601 email messages, form which 57 features have been extracted. The features are as follows:

- 48 features giving the percentage (0 to 100) of words in a given message that match a given word on a list that contains such words like "business", "free", etc.
- 6 features giving the percentage (0 to 100) of characters in the email that match a given character on a list ; ([! $ #
- Feature 55 is the average length of an uninterrupted sequence of capital letters (max is 40.3, mean is 4.9).
- Feature 56 is the length of the longest uninterrupted sequence of capital letters (max is 45.0, mean is 52.6).
- Feature 57 is the sum of the lengths of uninterrupted sequence of capital letters

The class label is binary: 1= SPAM and 2 = NOT SPAM.

Write a function called hwfs() which takes as input data, labels and cvind, and returns two variables:

- An array of selected features
- The performance in the dataset with the selected features.

For this approach, just use the Linear Discriminant algorithm using the function classify as outlined in 1.6. You don't have to use any other algorithms.

Write a sequential-forward feature selection algorithm. The algorithm starts out with all the N number of features, and builds models for one feature at a time. These models are built via 10-fold cross validation. It then selects the one feature which had the best average TEST (not train) performance across the 10 folds. This feature is selected.

It then repeats the process, this time building N-1 models of 2 features. The first selected feature is paired with every other feature. The two feature model which improves upon the previous performance by at least 0.01 is selected.

This process continues, adding one feature at a time, until there are no features which improve the average cross validation accuracy (CorrectRate) by at least 0.01. Keep in mind, that means you won't have to do this iteration 57 times, you'll probably end up with around 5-8 features after which performance won't improve by 0.01 and you can stop.

The function then builds one final LD model on the full dataset (no cross validation) with the selected set of features. It then returns the selected set of features and this training performance.

Submit your Matlab code, and a written report of your results. In the written report, list the index numbers (out of 57) of the features selected, and the LD training performance in the full dataset with all the selected features (not all 57, but the subset that was selected). Remember, while you may have other sub-functions, the main function should be called hwfs.

In your submission, include:

- Your analysis, training and test result table from part 1, as well as the selected features and performance from part 2, and answers to the requested questions in a report with your results in either.doc, or .pdf format. The report should be named as FirstName_LastName_HW1_Report.xxx. So for example, if Bob Smith submits a PDF report, the file should be called Bob_Smith_HW1_Report.pdf.
- Additionally, submit your Matlab code, along with any other functions you may write. DO NOT just run commands in the Matlab window, rather store all executed commands in a .m script file and submit that. If needed, we should be able to completely reproduce your work and get your results with the files you submit. Also, commenting your code is wise; in the event something is missing, partial credit is possible only if we can find the work.