COEN 281, Homework 3 – Non-Linear Classifiers
Due: Thursday, November 20

Please turn in a paper copy in class, or hand-deliver it to Apryl Roberts and have her time-stamp it. Email should only be used as a <u>*last*</u> *resource. Work turned in d days late is graded and the grade is multiplied by (1 – d/10) if d ≤ 5, and 0 otherwise.*

Work is to be done in groups of 2. Partner will be assigned randomly for each project. You must submit a confidential 1-to-5 (1=Poor; 5=Good) rating of your partner's contribution to the project. This rating will make 15% of the project's grade. Students with an average rating below 3 at the end of the quarter will have to submit himself/herself to a final exam. Please send an email to the instructor with the subject "HW2 – Group #," and the name of your partner and grade in the message body.

<u>*1. Maximal Margin Classifier.*</u> Textbook problem 9.7.3.

<u>*2. Neural Networks.*</u> We are going to use the "az-5000.txt" data set with the same 80/20 split between train/test from HW2. We want to use an $18\text{-}n_H\text{-}26$ network – i.e., 18 input features, n_H hidden units, and 26 output units.

a. Write an expression, as a function of n_H, for the total number of weights in the network. Based on the heuristic give in class, what approximate value of n_H should we expect to work well?

b. Use the *matrix* command to create a 4000x26 binary matrix of target row vectors for the training data. You may find the *as.numeric* command useful to convert the char column to an integer. Check the result by counting the numer of 1's in the resulting matrix – e.g., *sum*(targetMatrix == 1).

c. Use the *nnet* command to fit feed-forward neural networks with a single hidden layer to the training data. You may need to load the "nnet" package. In R, the syntax "*char* ~." indicates the formula for our functional model – i.e., that we are trying to predict char (column one in the data) as a function of all the other variables. Set the maximum number of iterations (*maxit* parameter) to 1000 and vary the *size* between 1 and 20.

d. For <u>each</u> nnet above, compute the Mean Square Error (MSE) i.e.,

$$\frac{1}{n}\sum_{i=1}^{n}\left\|\mathbf{t}^i - \mathbf{y}^i\right\|^2 = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{p}(t_j^i - y_j^i)^2$$

where $\boldsymbol{t}$ are the target vectors and $\boldsymbol{y}$ are the output vectors computed by the neural net. This can be easily computed in matrix form using the targetMatrix from b. and the fitted values returned by *nnet* – e.g.. *nnet*$fitted.values.

e. Use the *predict* command to compute the MSE for the test set too for each of the networks above.

f. Plot the results of d. and e. in a single figure. For the best net, report the total accuracy on the test and train sets.

3. Support-Vector Machines. The "spam.csv" data, collected at HP Labs, contains information on 4601 e-mails which were classified as *spam* or *non-spam*. Each e-mail is described by 57 variables indicating the frequency of certain words and characters in the message.

Put aside 15% of the data for testing. From the training data, select a smaller random sample of size 500 for "tuning" (see part a. below). We are going to build an SVM classifier using the Gaussian radial basis function (RBF) kernel – i.e., $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma\|\boldsymbol{x} - \boldsymbol{x}'\|^2)$. There are two "meta-parameters" in this type of SVM: γ, controlling the shape of the kernel, and *cost*, the penalty paid by the SVM for missclassifying a training point.

a. Use the "tuning" training data and the *tune.svm* command (library e1071) to conduct a grid search for the best values of γ and *cost*. Vary γ between 0.000001 and 0.001, and vary *cost* between 10 and 100. Use the *summary* command to identify the best values of γ and *cost*.

b. Using the best parameters, now train a final model on all the training data. Report the number of support vectors.

c. Report the confusion matrix and total accuracy on the test data.

4. Decision Trees. Use the "housetype_data.txt" data set from HW1. Refer to the documentation file housetype.info for attributes names and types. The goal in this problem is to construct a classification tree to predict the type of home from the other 13 demographics attributes and interpret the results. Put aside 10% of the data for estimating misclassification error. Report the number of NAs.

a. Use the *rpart* command (library rpart) to build a classification tree on the training data. help (rpart) will display the rpart help file. Note that some of the variables are categorical: be sure to mark them as such, using R function *as.factor()*, before running rpart. Set `method = "class"` and `cp = 0.0001` in the arguments to rpart.

b. Optimal tree size. Use the *plotcp()* command to view the graph of cross-validation error ("X-val") as a function of tree size. You can view the details of the data in this graph by examining the "cptable" attribute of the fitted rpart model – e.g., `print(mytree$cptable)` will print a table with number of splits (`nsplit`) and cross-validation error (`xerror`) among other things. Identify the row with the minimum cross-validation error in this table.

c. Use the `CP` value from the row identified in b. above as argument to the *prune()* command. Make sure you store the pruned tree in a new variable.

d. Use *plot()* and *text()* to plot the tree classifier in a nice way. Use ?plot.rpart to access the corresponding help file. Write a short report about the relation between the house type and the other demographic predictors as obtained from the RPART output.

e. Were any surrogate splits used in the construction of the optimal tree you obtained? What does a surrogate split mean? Give an example of a surrogate split from your optimal decision tree. Which variable is the split on? Which variable(s) is the surrogate split on?

f. Using your optimal decision tree, predict the house type on the test data. What was the total accuracy on the test set?