Assignment 4: Cross-Validation

Introduction

Cross validation (CV) is a technique to estimate out-of-sample prediction errors. It utilizes the standard technique of reserving part of your dataset and then testing model predictions using this test data set. However, it does this many (K) times, each time using a different test set. The idea is then that you can actually use all of your data to train the model, and use K-fold CV to estimate the prediction error.

In this homework, we will put your function writing skills to work, and become more familiar with CV. First, let's read in some data, which was provided in the file exercise3.data.csv.

```
dir="<name of directory where you put data here>"
#READS IN COMMA SEPARATED DATA AND TELLS R THAT THERE IS A HEADER LINE
data=read.csv(file=paste0(dir,"exercise2.data.csv"),header=T)
```

Now, let's write a function to compute the root mean squared error for a training and test data set. For simplicity, this function will assume that we wish to use a linear regression model, and that all variables in the data.frame will be used to predict y. The data.frame `train` will include the data used to train the model, and the data.frame `test` includes the data to test the model.

```
#COMPUTE ROOT MEAN SQUARE FOR TRAINING DATA AND TEST DATA
rmse = function(train,test) {
   fit = lm(y~., data=train)  #PERFORMS A LINEAR REGRESSION (see below)
   train.err = sqrt(mean((fit$resid)^2))  #RMS OF TRAINING RESIDUALS
   test.err = sqrt(mean((test$y - predict(fit,test))^2)) #RMS CV RESIDUALS
   c(train.err,test.err)    #RETURN BOTH
}
```

The lm command performs a linear regression of y on other variables in data.frame train (the dot tells R to use all variables, otherwise we could specific the name of specific variables using `train$var_name`.

Problem 1 (4 pts)

Now, write a function called cv.err that takes as input the name of a data.frame and a variable K. K corresponds to the number of splits used in cross validation. Hint, the first steps in this function should be to determine the total number of samples, and then identify K roughly equally sized groups of data. You can use the function cut() to do this. For example, the command

```
groups = cut(1:n,K,label=F)`
```

creates a vector of length n where each component specifies the group number of the corresponding component in the vector 1:n.

Once you have K different groups, you can run the rmse function K times, each time using a different subset as the test data. (Note: the cv.err function should be fewer than 15 lines, and should return a K by 2 array of training and test errors.)

Problem 2 (5 pts)

Try running your function cv.err for different values of K. Hand in a plot of test and training error from your CV function as a function of K. Also answer the following questions: Is the test error consistently above/below the training error? Does K affect the training error, and if so how? How does K affect the test error?

Note: for a K-fold CV, you will have K estimates of both training and test error, which means you can plot both the mean and standard error of CV error. An easy way to plot error bars is using the `errbar` function in the Hmisc library, or the `plotCI` function in the gplots library.

Problem 3 (6 pts)

Add three more columns to the dataset with random numbers from a normal distribution with mean 0 and standard deviation 1.

```
data[,5:7]=rnorm(3*nrow(data))
```

Now compute the training and cv error (using a value of k of your choice) for a model that uses just the first column of x, the first two, the first three, and so on until it uses all 6 x variables. Hand in a plot of the training and test cv error as a function of the number of predictor variables. Also answer the following questions: Does this show what you'd expect? You may wish to try this for a few different values of k and/or different levels of standard deviation in the noise.