STA 414/2104, Spring 2006 — Assignment #1

*Due at **start** of class on February 16. Note that this assignment is to be done by each student individually. You may discuss it in general terms with other students, but the work you hand in should be your own.*

In this assignment you will implement the k nearest neighbor method, and use cross validation to choose a suitable value for k, or to combine the predictions found using several values of k. A synthetic dataset is provided for testing.

These methods are aimed at predicting a real-valued response variable in a "test" case, based on a vector of real-valued input variables. For each test case, you should produce a single guess at the response variable. The accuracy of this guess should be measured by the squared difference between it and the actual value. The predictions are based on the examples of inputs and responses for N "training" cases.

The k-NN method for some fixed k predicts the response for a test case by finding the k training cases whose inputs are nearest to the test input (in Euclidean distance), and then averaging the responses for these k training cases to produce a guess for the test case. If some training cases are equally distant from the test case, you should resolve the tie in some arbitrary way when finding the k nearest training cases.

You should implement the k-NN method in a function called `knn`, which takes as arguments a vector of possible values for k, a matrix of inputs for training cases, a vector of responses for these training cases, and a matrix of inputs for test cases. It should return a matrix of guesses for the responses for each of these test cases. The first column of this matrix should contain the guesses using the first value of k in the first argument, the second column the guesses using the second value of k, etc.

One way to choose a suitable value for k is by *leave-one-out cross validation*. For each value of k in some set, we make predictions for every training case with k-NN using only the *other* training cases. We then pick the value of k for which the squared error averaged over these predictions is smallest, and use this value of k when making predictions for test cases.

You should write two functions to implement this method. The first, called `knncv`, should take as arguments a vector of possible values for k, a matrix of inputs for training cases, and a vector of responses for these training cases. It should return a matrix in which each row contains the predictions for a training case based only on the other training cases (not including itself), with the different columns containing predictions found using the various values of k. Of course, `knncv` should use the `knn` function you wrote earlier. The second function, called `knnsel`, should take as arguments a vector of possible values for k, a matrix of inputs for training cases, a vector of responses for these training cases, and a matrix of inputs for test cases. It should return a vector of predictions for the test cases using whatever value for k produced the smallest average squared error with leave-one-out cross validation (if there's a tie, break it arbitrarily). Your `knnsel` function should use the `knn` and `knncv` functions you wrote earlier.

Rather than select a single value of k, we can instead combine predictions using all the values of k that we considered. The best linear combination, based on the leave-one-out cross validation results, can be found using least-squares linear regression — just use the cross-validation predictions produced by `knncv` as inputs for a linear regression model of the corresponding responses. The resulting regression coefficients (including an intercept) can then be used to combine predictions for test cases produced by `knn`.

You should implement this idea in a function called `knncombo`, which should take as arguments a vector of possible values for k, a matrix of inputs for training cases, a vector of responses for these training cases, and a matrix of inputs for test cases. It should return a vector of predictions for the test cases found by combining the predictions for these values of k in the way described above.

Here is what these functions should return on a very small dataset, with two inputs, four training cases, and two test cases, using values of 1 and 2 for k:

```
> x.train
     [,1] [,2]
[1,]    3    7
[2,]    9    3
[3,]    7    2
[4,]    2    8
> y.train
[1] 1 5 6 3
> x.test
     [,1] [,2]
[1,]    2    9
[2,]    8    5
>
> knn (c(1,2), x.train, y.train, x.test)
     [,1] [,2]
[1,]    3  2.0
[2,]    5  5.5
> knncv (c(1,2), x.train, y.train)
     [,1] [,2]
[1,]    3  4.5
[2,]    6  3.5
[3,]    5  3.0
[4,]    1  3.5
> knnsel (c(1,2), x.train, y.train, x.test)
[1] 3 5
> knncombo (c(1,2), x.train, y.train, x.test)
[1]  8 -1
```

Note that because this dataset is so small, the above results may not make much sense. They're provided to help you debug your functions.

I have created a synthetic dataset for you to test your functions on, with two inputs, 200 training cases, and 1000 test cases. This dataset is available for downloading from the course web page, at `http://www.utstat.utoronto.ca/~radford/sta414`. You should apply the functions above to this dataset, with the set of possible values for k being $\{1, 3, 9, 27, 81\}$.

You should hand in a listing of your functions, written in R, Matlab, or (if you check with me first) some other suitable language. Your program should be reasonably easy to read, and in particular, should be *properly and consistently indented*. You should also hand in the output of `knn`, `knncv`, `knnsel`, and `knncombo` for the first ten test cases (*not* for all of them!). You should also hand in the average squared error over all test cases for predictions with each value of k as produced by `knn`, and for the predictions produced by `knnsel` and `knncombo`. You should discuss these results. In your discussion, you will need to refer to other information, such as what value of k was chosen by `knnsel`. You may (and should) extend the function descriptions to allow access to this information.

Finally, you should discuss how the `knncombo` function may or may not relate to the idea of using k nearest neighbors with unequal weightings — eg, with $k = 2$, one might form a weighted average in which the nearest neighbor has twice the weight of the second-nearest neighbor.