Project: 2 CS 170. Introduction to Artificial Intelligence

Due Date: December 5th

Feature Selection with Nearest Neighbor

As we have seen in class this quarter, the nearest neighbor algorithm is a very simple, yet very competitive classification algorithm. It does have one major drawback however; it is very sensitive to irrelevant features. With this in mind you will code up the nearest neighbor classifier, and then use it inside a "wrapper" which does various kinds of searches (listed below)

1) Forward Selection
2) Backward Elimination
3) Your original search algorithm.

Don't be scared by the phrase "search algorithm", the first two are simply nested loops, nothing else. The first two algorithms are as specified in class, the last one should be an original algorithm. If you need some hints on this, finish the first two algorithms early and come see me (don't wait for office hours, just stop by my office). Your original algorithm must be better than the two others. It could be better in either (or both) of two ways:

1) It could be faster.
2) It could give better results.

In your report, you must clearly explain why your algorithm is better. Without a clear explanation of this, your project will receive at most 50% of the possible points.

To make life simple, you can assume the following. I will only give you datasets that have two classes. I will only give you datasets that have continuous features (although you must normalize them).
Think carefully before you start coding this. Students in the past seem to have made this more complicated than it need be. In particular, in Matlab I was able to write the nearest neighbor algorithm in 8 lines of code, and the 3 search algorithms in another 17 lines of code. C++ and Java programs tend to be longer, but even so, I would be surprised if this took more than 100 lines of code (although I will not penalize you for this).

Very important: Make sure your nearest neighbor algorithm is working correctly before you attempt the search algorithms. We will provide some test datasets for this purpose.

You may use some predefined utility routines, for example sorting routines. However I expect all the major code to be original. You must document any book, webpage, person or other resources you consult in doing this project (see the first day's handout).

You may consult colleagues at a high level, discussing ways to implement the tree data structure for example. But you may **not** share code. At most, you might illustrate something to a colleague with pseudocode.

You will hand in

1) A coversheet.
2) A printout of your code.
3) A trace of your program on a particular dataset (these datasets will be online soon).
4) A report which summaries your findings. You need to compare the search algorithms on 3 different datasets (these datasets will be online soon).

You must keep the evolving versions of your code, so that, if necessary you can demonstrate to the course staff how you went about solving this problem (in other words, we may ask you to prove that you did the work, rather than copy it from somewhere).
In addition, you will need to be in lab (first hour only) in the week *before* December 5th to demonstrate your program, and have a TA sign off on it. If you cannot make those dates, you can stop by my office with a laptop. However you must book a date and time a week in advance.

You can use a simple text line interface or a more sophisticated GUI (but don't waste time making it pretty unless you are sure it works and you have lots free time). However your program should have a trace like the one below, so that it can be tested.

The data files will be in the following format. ASCII Text, IEEE standard for 8 place floating numbers. This is a common format; you should be able to find some code to load the data into your program, rather that writing it from scratch (as always, document borrowed code). The first column is the class, these values will always be either "1"s or "2"s. The other columns contain the features, which are **not** normalized. There may be an arbitrary number of features (for simplicity I will cap the maximum number at 64). There may an arbitrary number of instances (rows), for simplicity I will cap the maximum number at 2,048. Below is a trivial sample dataset. The first record is class "2", the second is class "1" etc. This example has just two features.

```
2.0000000e+000  1.2340000e+010  2.3440000e+000
1.0000000e+000  6.0668000e+000  5.0770000e+000
2.0000000e+000  2.3400000e+010  3.6460000e+000
1.0000000e+000  4.5645400e+010  3.0045000e+000
```

Welcome to Bertie Woosters Feature Selection Algorithm.
Type in the name of the file to test : **eamonns_test_2.txt**

Type the number of the algorithm you want to run.

1) Forward Selection
2) Backward Elimination
3) Bertie's Special Algorithm.

1

This dataset has 4 features (not including the class attribute), with 345 instances.

Please wait while I normalize the data… Done!

Running nearest neighbor with all 4 features, using "leaving-one-out" evaluation, I get an accuracy of 75.4%

Beginning search.

Using feature(s) {1} accuracy is 45.4%
Using feature(s) {2} accuracy is 63.7%
Using feature(s) {3} accuracy is 71.4%
Using feature(s) {4} accuracy is 48.1%

Feature set {3} was best, accuracy is 71.4%

Using feature(s) {1,3} accuracy is 48.9%
Using feature(s) {2,3} accuracy is 70.4%
Using feature(s) {4,3} accuracy is 78.1%

Feature set {4,3} was best, accuracy is 78.1%

Using feature(s) {1,4,3} accuracy is 56.9%
Using feature(s) {2,4,3} accuracy is 73.4%

(Warning, Accuracy has decreased! Continuing search in case of local maxima)
Feature set {2,4,3} was best, accuracy is 73.4%

Using feature(s) {1,2,4,3} accuracy is 75.4%

Finished search!! The best feature subset is {4,3}, which has an accuracy of 78.1%