Richard Ashley

SID# 12345678

Email: rick@cs.ucr.edu

March 19, 2021

Project 2 for CS 205 Fall 2022, with Dr Eamonn Keogh.

All code is original, except:

1) I used a LISP++ package called DUB to compute Euclidean distance.
2) I used a memory manager from (www.some.url.com) to adipiscing elit, sed do eiusmod tempor incididunt.

In this project we are tasked with consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation.

In Figure 1 we see the result of running forward selection on CS170_SMALLtestdata__123.txt, which was the small file assigned to me.

Figure 1: Accuracy of increasingly large subsets of features discovered by forward selection.

At the beginning of the search, we have no features (denoted by {}) so I reported the default rate, which was 51.3%. Adding feature '8' dramatically improved the accuracy to 71.6%, and then adding feature '6' gave us an accuracy of 92.3%. When we then added feature '9', the accuracy increases, but only by a tiny amount, to 92.6%. Because this is just a small gain, I suspect it does not reflect a true useful feature. Thereafter, each additional feature added reduces the accuracy, until we have the full set of features, which gives us an accuracy of 49.6%.

Next, as shown in Figure 2, I ran backward elimination on CS170_SMALLtestdata__123.txt.

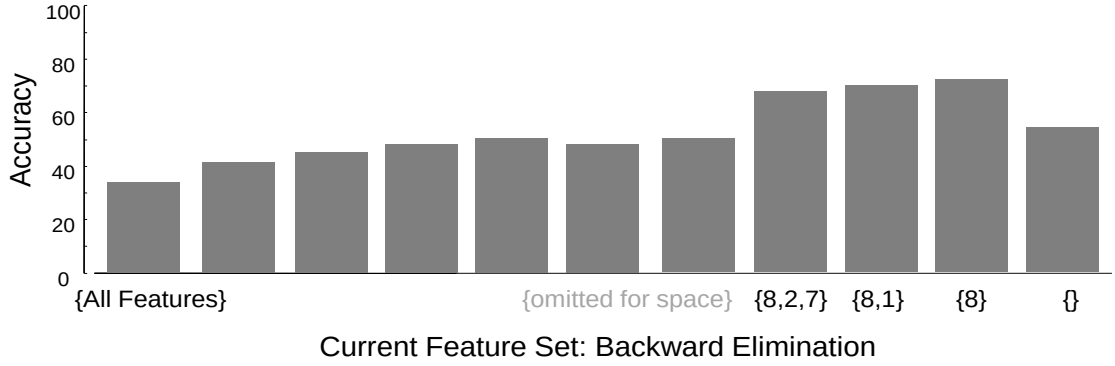

Figure 2: Accuracy of increasingly small subsets of features discovered by backward selection.

First, a sanity check. The first tested subset, that of *all* features, should have the same accuracy as the last tested subset in forward selection. With a score of 49.6%, that is the case. Likewise, the last tested subset ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation. For this problem, backward elimination is not as accurate as forward selection. However, because it does find feature '8' as its best feature subsets, it gives us confidence that feature ''8' is really a good feature.

Conclusion For Small Dataset: I believe that features '8' and '6' are the best features for this problem. There is very weak evidence that feature '9' might be useful, but that needs further investigation. If we deploy this {8,6} model, I believe the accuracy will be about 92.3%.

We now turn our attention to the more challenging dataset I was tasked with investigating. In Figure 3 we see the result of running forward selection on CS170_LARGEtestdata__123.txt, which was the small file assigned to me.

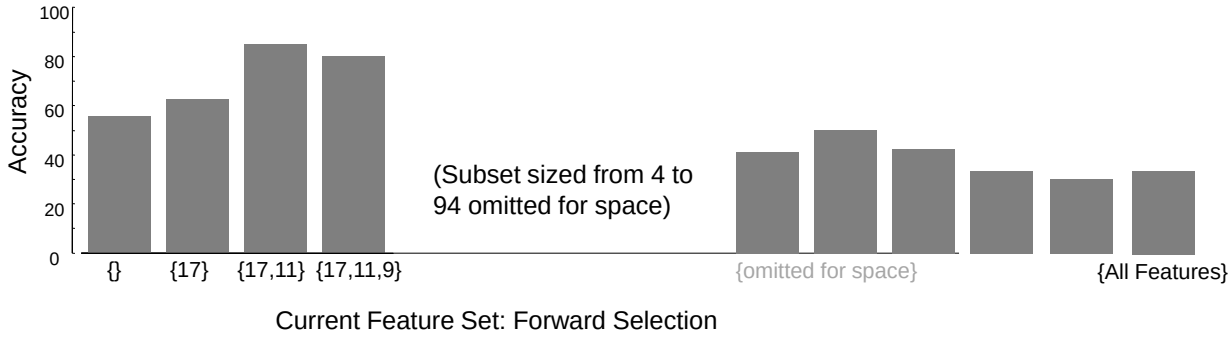

Figure 3: Accuracy of increasingly large subsets of features discovered by forward selection.

Nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum Next, as shown in Figure 4, I ran backward elimination on CS170_Largetestdata__123.txt.

Figure 4: Accuracy of increasingly small subsets of features discovered by backward selection.

Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid

Conclusion for Large Dataset: Taque earum rerum hic tenetur a sapiente delectus, ut aut reiciendis voluptatibus maiores alias consequatur aut perferendis doloribus asperiores repellat.

Computational effort for search: I implemented the search in LISP++, and ran all experiments on a laptop with a Intel Core i9-9900K and 4 gigs of main memory. In table 1 I report the running time for the four searches I conducted.

	Small dataset (10 features,300 instances)	Large dataset (** features,&& instances)
Forward Selection	2.2 minutes	0.89 hours
Backward Elimination	2.4 minutes	0.45 hours

Note that quis nostrum exercitationem ullam corporis suscipit laboriosam, Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore.

Below I show a single trace of my algorithm. I am *only* showing Forward Selection on the small dataset.

```
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

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

{Dear Professor: Here I deleted most of the trace to save space. However, I have retain at least the full first level of the search, and the last level of the search}

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

Code: Below is my code for this project (EK, an alternative is to put code on GitHub and point to it).

```
function accuracy = cs170demo()
%% Comments for code is a good idea!
%% Comments for code is a good idea!
%% Comments for code is a good idea!

data = load('C:\Users\eamon\Documents\MATLAB\CS170_SMALLtestdata__1.txt');
number_correctly_classfied = 0;

 for i = 1 : size(data,1)    %% Comments for code is a good idea!
    object_to_classify = data(i,2:end);
    label_object_to_classify = data(i,1);

    nearest_neighbor_distance = inf; %% Comments for code is a good idea!
    nearest_neighbor_location = inf; %% Comments for code is a good idea!

    for k = 1 : size(data,1)
        if k ~= i
             distance = sqrt(sum((object_to_classify - data(k,2:end)).^2));
             if  distance <   nearest_neighbor_distance
                    nearest_neighbor_distance = distance;
                    nearest_neighbor_location = k;
                    nearest_neighbor_label    = data(nearest_neighbor_location,1);
             end
        end
    end %% Comments for code is a good idea!


    disp(['Object ', num2str(i), ' is class ', num2str(label_object_to_classify)]);
    disp(['Its nearest_neighbor is ', num2str(nearest_neighbor_location), ' which is in class ',
num2str( nearest_neighbor_label )]);

    if label_object_to_classify == nearest_neighbor_label;
         number_correctly_classfied = number_correctly_classfied + 1;
    end%% Comments for code is a good idea!


 end %% Comments for code is a good idea!

 accuracy = number_correctly_classfied / size(data,1);
end
```

Hints!

Here I offer you some very useful hints.

1) Read the instructions
2) Read the instructions
3) Read the instructions
4) Brown M&Ms
5) Read the instructions
6) Read the instructions
7) Read the instructions