**Preston Reed**
SID: 862155221
Email: preed004@ucr.edu
June 10, 2022

**Project 2 Report for CS 205, with Dr. Eamonn Keogh**

All code is original, except:

1. Chrono code for timing the project

Code Repo: https://github.com/prestonmreed/CS205-Project2
*-Final version of the file is FeatureSelection_FINAL.cpp*
*-Compile using g++*

## Introduction

The goal of this project is to create a feature selection algorithm that searches for the best features that, when used in combination, provides the best accuracy for the test set. Leave-one-out cross validation with a nearest neighbor classifier is used to determine that accuracy fo a subset of features at any given step.

The search algorithm is implemented in two slightly different ways:

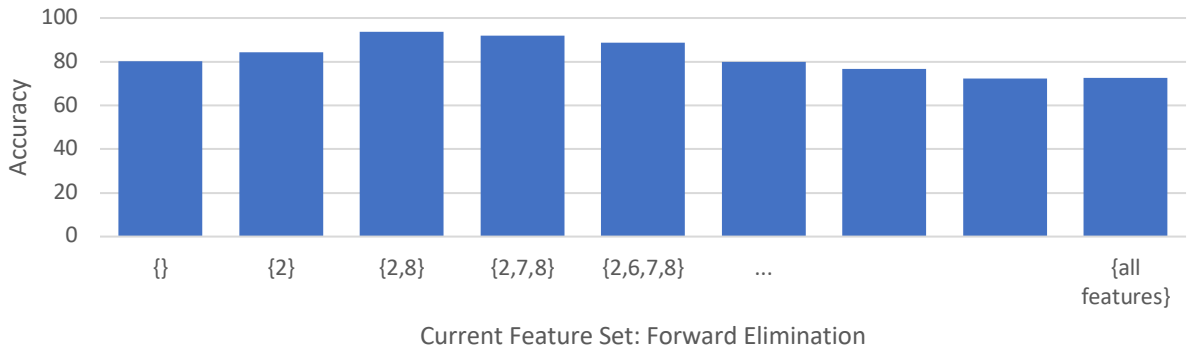
1. Forward Selection
2. Backward Elimination

Forward Selection begins with an empty set of features and adds features one at a time to the set based on which feature increases the accuracy of classification the most.

Backward Elimination, on the other hand, begins with the full set of features given by the dataset and eliminates features one at a time based on which feature being eliminated increases accuracy of classification the most.

## Forward Selection (Small Dataset)

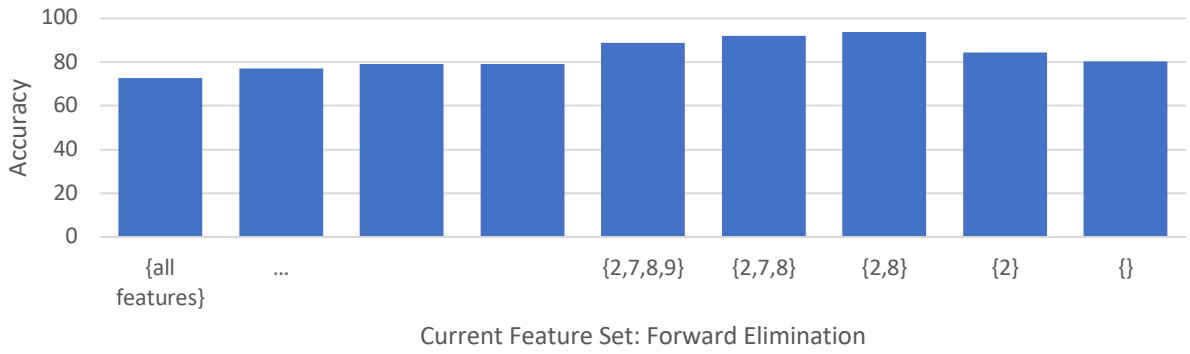
First, let's look at what Forward Selection found for the small dataset. Below is a chart showing how the accuracy changed over iterations of the forward selection algorithm.




The Forward Selection algorithm begins with the default rate (accuracy of the empty feature set) with an accuracy of 80.3%. This provides a base accuracy to be better than. The Forward Selection Algorithm found that feature subset {2,8} was the best with an accuracy of 93.7% by first selecting feature subset {2} with an accuracy of 84.3% and then adding feature 8 to the current feature set.

## Backward Elimination (Small Dataset)

Now, let's look at what Backward Elimination found for the small dataset.



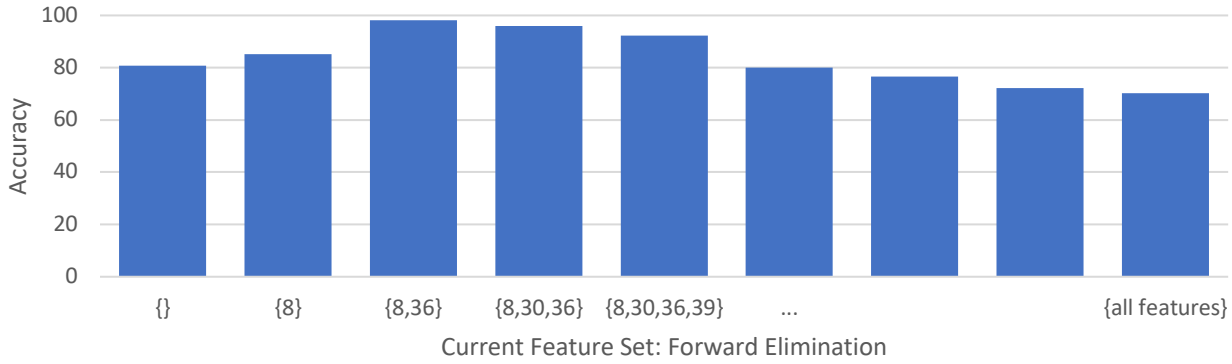

The Backward Elimination algorithm begins with the set of all features, which in this case would be all features from 1 to 10. It then removes one feature at a time, based on which removed feature provides the greatest increase in accuracy. We can see that the Backward Elimination ended up with the same feature set as Forward Elimination ({2,8} with an accuracy of 93.7%). However, it is interesting that it chose {2,7,8,9} with accuracy 88.7% as its size-4 feature set instead of {2,6,7,8} with an accuracy of also 88.7%.

## Small Dataset Conclusion

Both Feature Selection algorithms ended up with the same feature sets and accuracies, which points to feature set {2,8} as being the best possible feature set to choose out of all the features.

## Forward Selection (Large Dataset)

Now, we move on to the large dataset. Let's look at how Forward Selection found the most accurate subset of features for the small and large datasets that I was provided. Below is a chart showing the how the accuracy changed over iterations of feature subsets for Forward Selection on the large dataset.



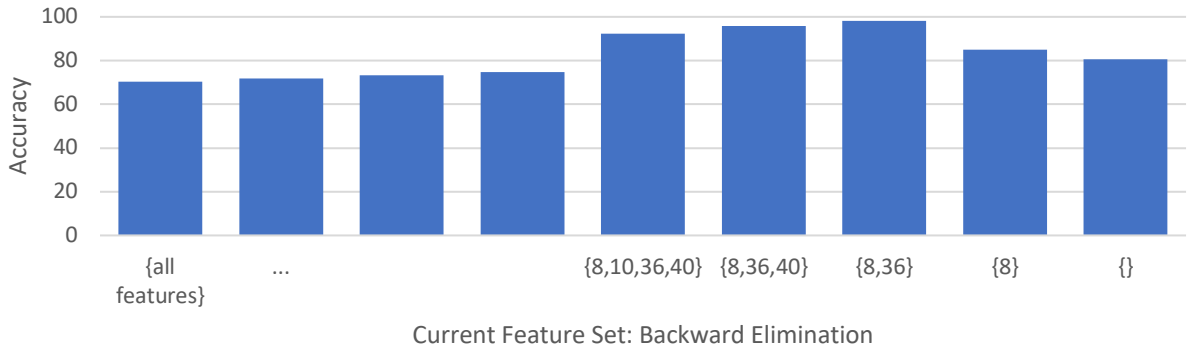

We can see that the accuracy began with the default rate, which is the empty set of features, and has an accuracy of 80.7%. The accuracy increases to 85.1% when feature 8 is chosen and increases once more to 98.2% when feature 36 is chosen. Because the increase is much greater

when feature 36 is added, my guess is that feature 36 is a very strong feature but it depends on the extra dimension that feature 8 provides.

When the algorithm adds feature 30, the accuracy begins to decrease (95.9%). The feature subset {8,36} turns out not to be a local maxima because the accuracy continues to decrease the more features we add. Therefore {8,36} is the most accurate feature subset with an accuracy of 95.9% based on forward selection.

## Backward Elimination (Large Dataset)

Now, let's look at how Backward Elimination found its feature subset for the same exact large dataset.




We can see that Backward Elimination found the same feature subset as Forward Elimination: {8,36} with the same accuracy of 98.2%. However, it arrived at this feature subset in a different way. As we know, it began with all features in the feature subset and eliminated features one by one. We can see that at the subset of size 4, it chose features {8,10,36,40}, which is different that Forward Elimination's size 4 subset, which was {8,30,36,39}.

## Large Dataset Conclusion

This difference in what features were kept/chosen demonstrates the difference in how these two search algorithms work in practice. It also shows that both features 8 and 36 were strong enough to have chosen them in forward selection and to have kept them in Backward Elimination. In other words, it was entirely possible that features 8 and/or 26 could have been eliminated in Backward Elimination and it's my speculation that this speaks to their individual strength as features.

In fact, looking at them individually, they are the most accurate features when used alone. Feature subset {8}, as we know, has an accuracy of 85.1% and feature subset {36} has an accuracy of 74%. Every other single-feature subset is less than both values.

## Performance Comparison

To compare the performance of these two algorithms, I recorded their execution times for both the large and small datasets. It is my hypothesis that Forward Selection will be faster in general. My reasoning is that the best feature subset is usually made up of three or less features, according to what we covered in lecture. Feature subsets of size 3 are much closer when starting from an empty subset and adding features than starting with a subset that contains all features (which could be very large) and eliminating features one by one.

For this comparison, all recorded execution times were done using the chrono library in C++. The chrono timer starts right before forward selection or backward elimination is called and ends when the chosen search function terminates. All tests were run on my Desktop PC which contains a Ryzen 3600x CPU and 16 GB of main memory.

| File Name | Forward Selection Time | Backward Elimination Time |
|---|---|---|
| Small Dataset | 1.34 s | 1.85 s |
| Large Dataset | 578.80 s | 1057.01 s |

As we can see, my hypothesis appears to be correct for the given datasets. Backward Elimination had a slower execution time compared to Forward Selection for both the small and large datasets. For the large dataset the execution time for Forward Selection was almost half of the execution time of Backward Elimination.

## Program Trace (Small Dataset, Forward Selection)

```
Welcome to Preston Reed's Feature Selection Algorithm
Type in the name of the file to test:
CS205_SP_2022_SMALLtestdata__41.txt

Type the name of the Algorithm you want to run.
1) Forward Selection
2) Backward ELimination
1
This dataset has 10 features (not including the class attribute), with
300 instances.
Beginning Search

Using feature(s) {} accuracy is 80.3333%
Feature set {} was best, accuracy is 80.3333%
Using feature(s) {1} accuracy is 67.3333%
Using feature(s) {2} accuracy is 84.3333%
Using feature(s) {3} accuracy is 68%
Using feature(s) {4} accuracy is 70.6667%
Using feature(s) {5} accuracy is 71%
Using feature(s) {6} accuracy is 70.3333%
Using feature(s) {7} accuracy is 68.6667%
Using feature(s) {8} accuracy is 74.6667%
Using feature(s) {9} accuracy is 72.3333%
Using feature(s) {10} accuracy is 69.6667%
Feature set {2} was best, accuracy is 84.3333%
```

**{ OMITTED INTERMEDIATE STEPS: full traces for all program runs can be found in GitHub Repository (linked on title page) with file names: small_forward_trace.txt, small_backward_trace.txt, large_forward_trace.txt, large_backward_trace.txt) }**

```
Using feature(s) {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} accuracy is 72.6667%
(Warning, Accuracy has decreased! Continuing search in case of local
maxima)
Feature set {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} was best, accuracy is
72.6667%
Finished Search!! The best feature subset is {2, 8}, which has an
accuracy of 93.6667%
```