Project 2

Prepared for: Professor Eamonn Keogh

Written By: Andre Tran

SID: 862051183

Email: atran112@ucr.edu

Date: Friday, March 19, 2021

CS 170: Introduction to Artificial Intelligence

Table of Contents

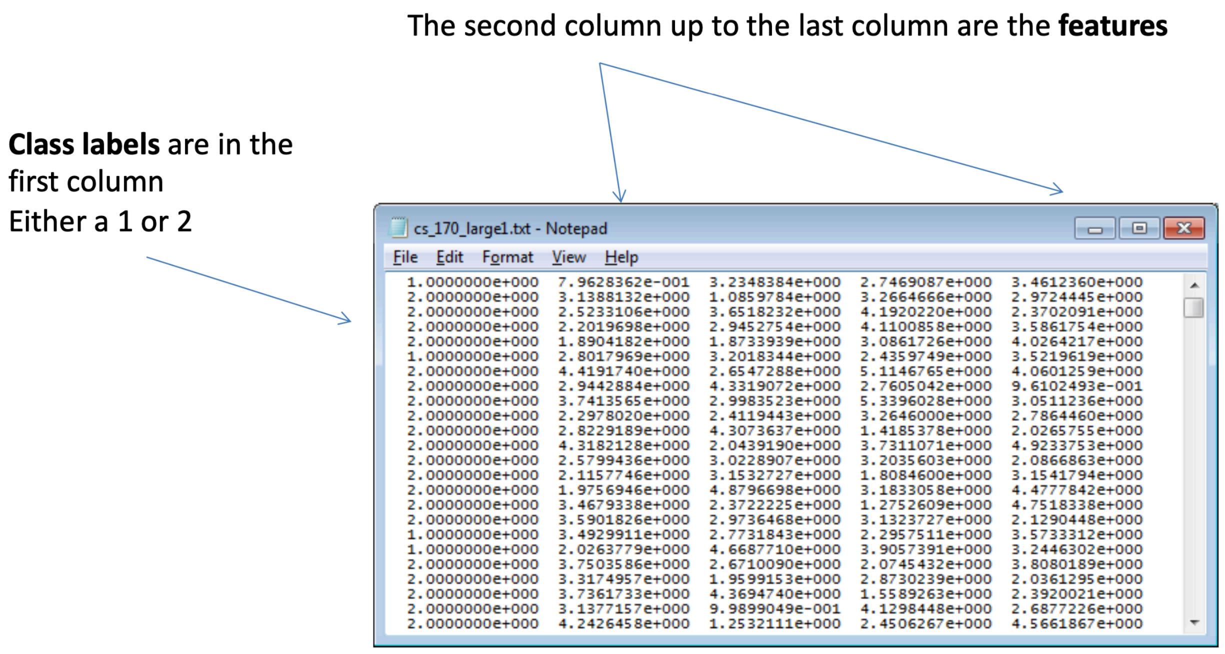
Resources

All important code is original. Unimportant subroutines that are not completely original are in the following:

- Reading tabular data with C++: https://www.daniweb.com/programming/software-development/threads/88262/how-do-i-read-a-text-file-into-a-2d-vector
- CS 170 Lecture Slides

Introduction

In this project we were tasked with analyzing a dataset using the nearest neighbor (or KNN) algorithm. In Figure 1, an example of a dataset is depicted.


The second column up to the last column are the **features**
Class labels are in the first column
Either a 1 or 2
cs_170_large1.txt - Notepad
File Edit Format View Help

```
  1.0000000e+000  7.9628362e-001  3.2348384e+000  2.7469087e+000  3.4612360e+000
  2.0000000e+000  3.1388132e+000  1.0859784e+000  3.2664666e+000  2.9724445e+000
  2.0000000e+000  2.5233106e+000  3.6518232e+000  4.1920220e+000  2.3702091e+000
  2.0000000e+000  2.2019698e+000  2.9452754e+000  4.1100858e+000  3.5861754e+000
  2.0000000e+000  1.8904182e+000  1.8733939e+000  3.0861726e+000  4.0264217e+000
  1.0000000e+000  2.8017969e+000  3.2018344e+000  2.4359749e+000  3.5219619e+000
  2.0000000e+000  4.4191740e+000  2.6547288e+000  5.1146765e+000  4.0601259e+000
  2.0000000e+000  2.9442884e+000  4.3319072e+000  2.7605042e+000  9.6102493e-001
  2.0000000e+000  3.7413565e+000  2.9983523e+000  5.3396028e+000  3.0511236e+000
  2.0000000e+000  2.2978020e+000  2.4119443e+000  3.2646000e+000  2.7864460e+000
  2.0000000e+000  2.8229189e+000  4.3073637e+000  1.4185378e+000  2.0265755e+000
  2.0000000e+000  4.3182128e+000  2.0439190e+000  3.7311071e+000  4.9233753e+000
  2.0000000e+000  2.5799436e+000  3.0228907e+000  3.2035603e+000  2.0866863e+000
  2.0000000e+000  2.1157746e+000  3.1532727e+000  1.8084600e+000  3.1541794e+000
  2.0000000e+000  1.9756946e+000  4.8796698e+000  3.1833058e+000  4.4777842e+000
  2.0000000e+000  3.4679338e+000  2.3722225e+000  1.2752609e+000  4.7518338e+000
  2.0000000e+000  3.5901826e+000  2.9736468e+000  3.1323727e+000  2.1290448e+000
  1.0000000e+000  3.4929911e+000  2.7731843e+000  2.2957511e+000  3.5733312e+000
  1.0000000e+000  2.0263779e+000  4.6687710e+000  3.9057391e+000  3.2446302e+000
  2.0000000e+000  3.7503586e+000  2.6710090e+000  2.0745432e+000  3.8080189e+000
  2.0000000e+000  3.3174957e+000  1.9599153e+000  2.8730239e+000  2.0361295e+000
  2.0000000e+000  3.7361733e+000  4.3694740e+000  1.5589263e+000  2.3920021e+000
  2.0000000e+000  3.1377157e+000  9.9899049e-001  4.1298448e+000  2.6877226e+000
  2.0000000e+000  4.2426458e+000  1.2532111e+000  2.4506267e+000  4.5661867e+000
```

FIGURE 1
Screenshot from Dr. Keogh's slides that explains classified tabular data

As can been seen in the figure, the left most column classifies the instance (or row). In this case, the data is classified as either (1) or (2). All columns to the right of the classifier column are considered features that makeup the instance. As an example, we can think of the classifier column as gender (male and female) and the features as height, weight, hair length, etc. So, through the KNN algorithm, we are trying to find the features that best describe the classification. We do this by adding or removing certain features and getting the Euclidean distance between two instances. We then assign an instance the classifier that it is closest to and count how many times were are correct in classifying the instance. This results in us finding the features with the highest accuracy, which then reveals the features that are most pertinent to classifying the data.

Analysis

In our analysis, two searches will be implemented to find the best features. The first is forward selection which incrementally adds features through each iteration. The other is backward elimination which starts with all features and removes a feature through each iteration. Each algorithm is designed to either append or eliminate the features that will increase the overall accuracy. At the end of these algorithms, the set of features with the highest accuracy is returned.

Small Forward Selection

In Figure 2, we can see the results of running forward selection on CS170_SMALLtestdata__77.txt.

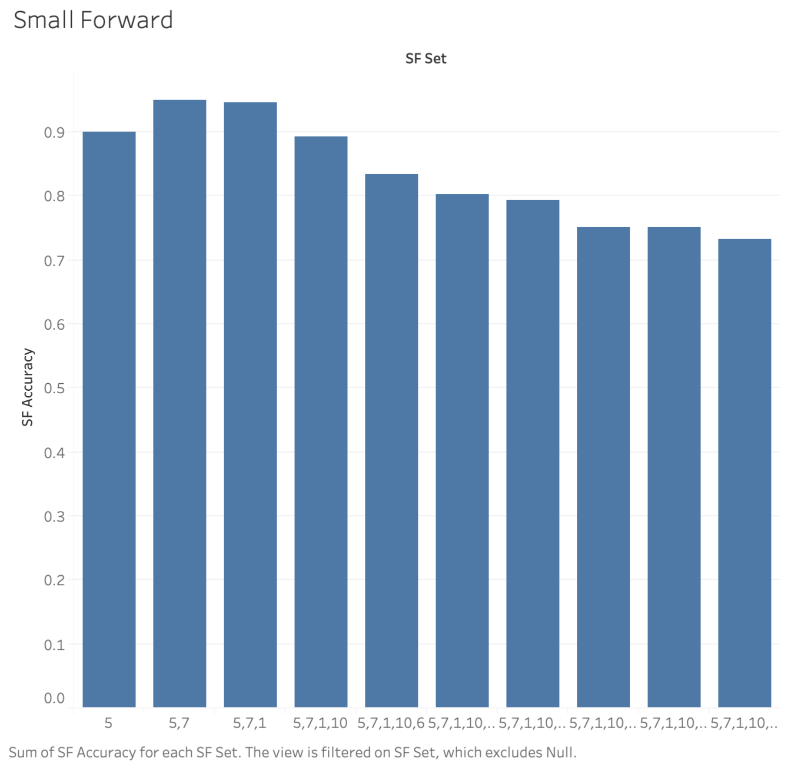


FIGURE 2
Image of the Small Forward Selection Results

In the beginning of the search, we start off with one feature (5) with an accuracy of 90%. Furthermore, another feature (7) is added to our set which increases our accuracy to 95%. From this point on, every feature that is added decreases in accuracy. For example, the next feature (1) is added and our accuracy lowers to 94.67%. Eventually, we reach the full set of features, which has an accuracy of 73.33%.

Small Backward Elimination

In Figure 3, we can see the results of running backward elimination on CS170_SMALLtestdata__77.txt.

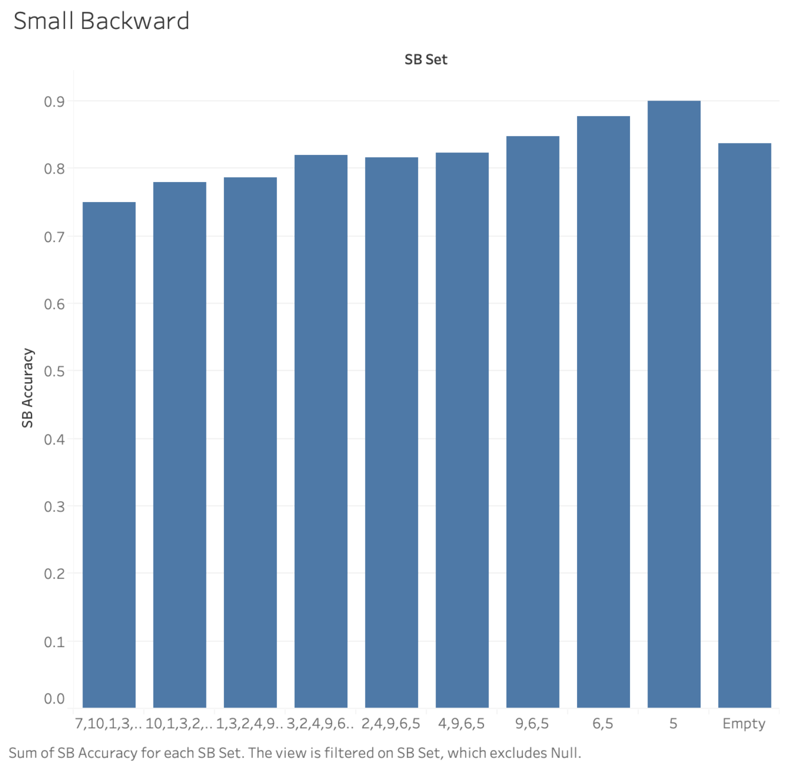


FIGURE 3
Image of the Small Backward Elimination Results

In the beginning of the search, we have all features except for feature (8) with an accuracy of 75%. Furthermore, another feature (7) is removed from our set which increases our accuracy to 78%. As we continue to remove features, our accuracy trends positive until we reach the set containing feature (5), which has an accuracy of 90%. Lastly, the empty set has an accuracy of 83.67%.

Conclusion I

From this terse analysis of the forward selection and backward elimination, we can see that the forward selection produced greater accuracy at 95% compared to 90%. Furthermore, it is evident that feature (5) is an important feature as it was featured in the sets with highest accuracy. Interestingly, however, feature (7) was the second feature to be eliminated from backward elimination, yet it was in the set with the highest accuracy for forward selection. More investigation should be done to see the usefulness of feature (7).

Larger Forward Selection

We now move onto the large data set. In Figure 4, we can see the results of running forward selection on CS170_largetestdata__23.txt.

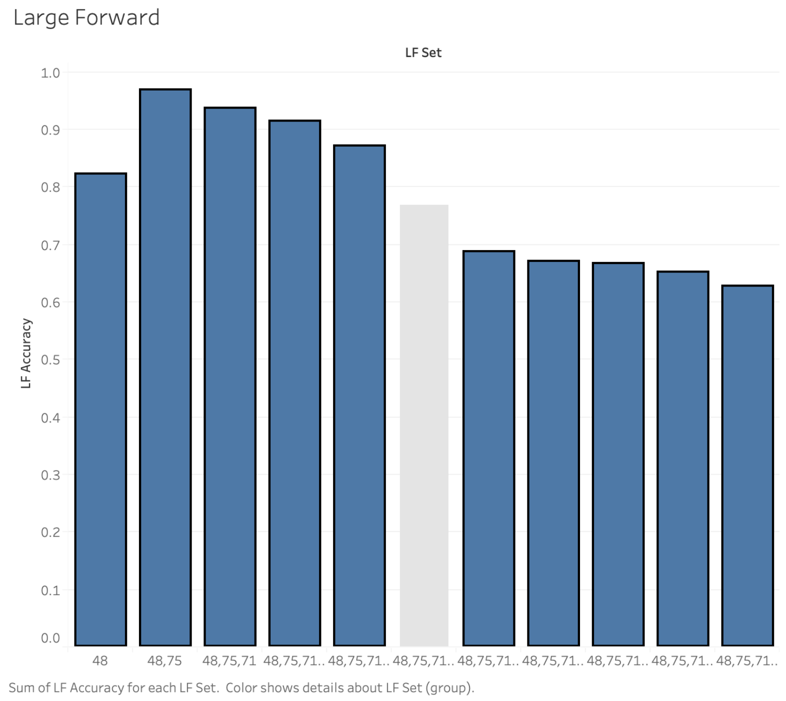


FIGURE 4

Image of the Large Forward Selection Results

In this search we start off by adding feature (48) which results in an accuracy of 82.2%. Furthermore, we add a second feature (75), and the accuracy of our set increases to 96.80%. Moving forward, every feature that is added decreases in accuracy. Please also note that the bar colored in light gray is a representation of the average accuracy from iterations 6 to 95. Eventually, we reach the full set of features, which has an accuracy of 62.6%.

Large Backward Selection

In Figure 5, we can see the results of running backward elimination on CS170_largetestdata__23.txt.

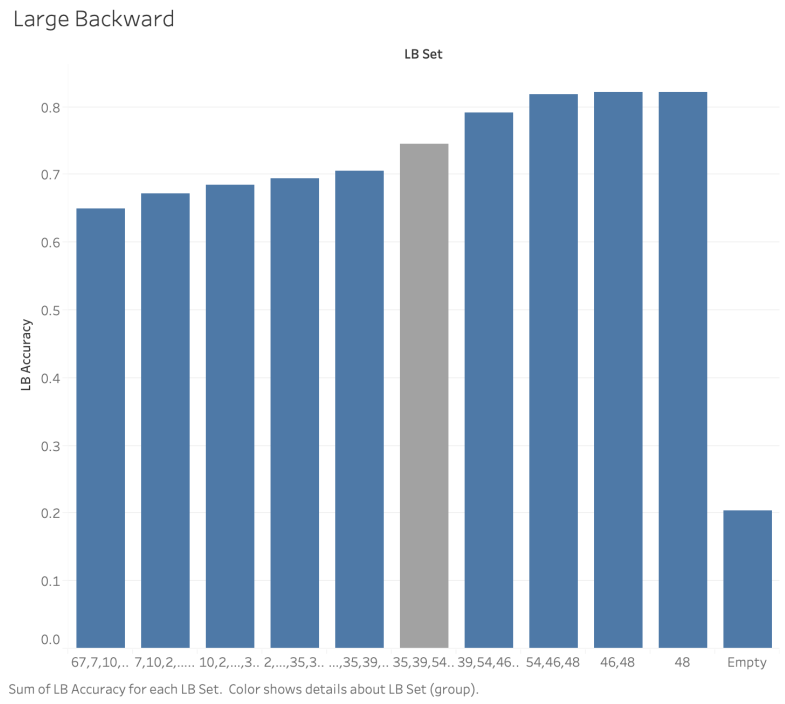


FIGURE 5

Image of the Large Backward Elimination Results

Starting off, we have all features except for feature (43) with an accuracy of 65%. Furthermore, another feature (67) is removed from our set which increases our accuracy to 67.2%. Please also note that the bar colored in gray is a representation of the average accuracy from iterations 6 to 95. As we continue to remove features, our accuracy trends positive until we reach the set containing features (46, 48), which has an accuracy of 82.2%. This accuracy, however, is the exact same accuracy as the next iteration, feature (48). The last iteration, the empty set, only has an accuracy of 20%.

Conclusion II

From the results, we can see that forward search produced better accuracy than backward elimination at 96.8% compared to 82.2%. Additionally, we can see that feature (48) is an important feature for classification as it ensures 82.2% accuracy alone. Feature (75) is also important as it increases the accuracy by 14.6% when paired with feature (48). Feature (46), however, is likely not important as, when removed from feature (48), does not change the accuracy at all.

Computational Effort for Search

I implemented the search in C++, and ran all experiments on a MacBook with 2 GHz Dual-Core Intel Core i5 and 8 GB of memory. In table 1, I report the running time for the four searches I conducted.

Table 1

	Small Dataset	**Large Dataset**
Forward Selection	10.28 seconds	3.09 hours
Backward Elimination	10.20 seconds	3.09 hours

Please note that, for the large dataset, I ran both forward selection and backward selection simultaneously in two different terminals. Furthermore, I occasionally had to open other applications while the programs were running. As a result, these choices may have resulted in slightly longer runtimes.

Trace

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

```
Please enter the filename
CS170_SMALLtestdata__77.txt
Type the number of the algorithm you want to run
   1) Forward Selection
   2) Backward Elimination
1

On the 1th level of the search tree
--Considering adding the 1 feature
--Considering adding the 2 feature
--Considering adding the 3 feature
--Considering adding the 4 feature
--Considering adding the 5 feature
--Considering adding the 6 feature
--Considering adding the 7 feature
--Considering adding the 8 feature
--Considering adding the 9 feature
--Considering adding the 10 feature

On level 1 I added feature 5 to current set
Accuracy: 0.9000
-----------------------------------------------

On the 2th level of the search tree
--Considering adding the 1 feature
--Considering adding the 2 feature
--Considering adding the 3 feature
--Considering adding the 4 feature
--Considering adding the 6 feature
--Considering adding the 7 feature
--Considering adding the 8 feature
--Considering adding the 9 feature
--Considering adding the 10 feature

On level 2 I added feature 7 to current set
Accuracy: 0.9500
-----------------------------------------------

On the 3th level of the search tree
--Considering adding the 1 feature
--Considering adding the 2 feature
--Considering adding the 3 feature
--Considering adding the 4 feature
--Considering adding the 6 feature
--Considering adding the 8 feature
```

```
--Considering adding the 9 feature
--Considering adding the 10 feature

On level 3 I added feature 1 to current set
Accuracy: 0.9467
------------------------------------------------

On the 4th level of the search tree
--Considering adding the 2 feature
--Considering adding the 3 feature
--Considering adding the 4 feature
--Considering adding the 6 feature
--Considering adding the 8 feature
--Considering adding the 9 feature
--Considering adding the 10 feature

On level 4 I added feature 10 to current set
Accuracy: 0.8933
------------------------------------------------

On the 5th level of the search tree
--Considering adding the 2 feature
--Considering adding the 3 feature
--Considering adding the 4 feature
--Considering adding the 6 feature
--Considering adding the 8 feature
--Considering adding the 9 feature

On level 5 I added feature 6 to current set
Accuracy: 0.8333
------------------------------------------------

On the 6th level of the search tree
--Considering adding the 2 feature
--Considering adding the 3 feature
--Considering adding the 4 feature
--Considering adding the 8 feature
--Considering adding the 9 feature

On level 6 I added feature 2 to current set
Accuracy: 0.8033
------------------------------------------------

On the 7th level of the search tree
--Considering adding the 3 feature
--Considering adding the 4 feature
--Considering adding the 8 feature
--Considering adding the 9 feature

On level 7 I added feature 3 to current set
Accuracy: 0.7933
------------------------------------------------

On the 8th level of the search tree
```

```
--Considering adding the 4 feature
--Considering adding the 8 feature
--Considering adding the 9 feature

On level 8 I added feature 4 to current set
Accuracy: 0.7500
-----------------------------------------------

On the 9th level of the search tree
--Considering adding the 8 feature
--Considering adding the 9 feature

On level 9 I added feature 8 to current set
Accuracy: 0.7500
-----------------------------------------------

On the 10th level of the search tree
--Considering adding the 9 feature

On level 10 I added feature 9 to current set
Accuracy: 0.7333
-----------------------------------------------

The best features are: 5 7
Accuracy: 0.9500
Elapsed time: 10.2979 seconds
```

Code

Please view the code for this project in the link below.

https://github.com/atran112/nearest-neighbor-classification