Prithviraj Yuvaraj

SID: 861258445

Pyuva001@ucr.edu

06/10/2021

Spring 2021

CS205 Project 2: Nearest Neighbor

Overview

The project was created in Python 3 and tested using Python 3.8. As reference the Python v3.1.4 documentation: https://docs.python.org/3.1/index.html was used to help with specific data structures. When initially testing search with a dummy validation function, a random number generator was used. Other Python libraries that were used were numpy, and math, these were used to make the nearest neighbor computation as simple as possible. As reference for the algorithm, the lecture slide and lecture video were consulted.

Introduction

The goal of this project was to create, and test two methods for feature selection, Forward Selection and Backward Elimination. The accuracy of the set of features was calculated using the Nearest Neighbor algorithm. My implementation was then tested with example datasets and specific datasets assigned to me personally.

The Nearest Neighbor algorithm is generally not overly sensitive to noisy data features, but overfitting is a problem that occurs on any machine learning algorithm. Overfitting occurs when algorithms learn data points from features that are not relevant to the task at hand, in this case our task is classification between two different labels. The combination of Nearest Neighbor and one of the two feature selection algorithms can mitigate the issue of overfitting with our specified dataset.

Now moving onto the feature selection algorithms, both are similar and use Nearest Neighbor to calculate the accuracy of adding or removing a specific feature. Forward Selection starts with an empty feature set and adds each feature one by one by computing the accuracy with the addition of the feature. Then the feature addition that has the highest accuracy will be added to the set. Inversely, Backward Elimination starts with the full feature set and eliminate each feature based on the accuracy calculated by removing it from the feature set. In addition to the algorithms, bookkeeping steps were added to find the best set of features by keeping track of the highest achievable accuracies.

Example Execution

The entire execution for CS205_small_testdata__24 is in execution.txt, the output of CS204_large_testdata__43 is far too large to put in the execution.txt file. Below is only a portion of the output. The red text are the inputs to run different datasets.

Nearest-Neighbor!

Would you like to use a small or large or real dataset: (1 for small, 2 for large, 3 real data)3

Dataset small 24, and large 43 avalible: (type the number)24

The Data set 24 has 10 features, and 300 samples

Starting feature search...

The current set is []

On the 1th level of the search tree

--Considering adding the 1 feature

Adding that feature gives an accuracy of 0.71

--Considering adding the 2 feature

Adding that feature gives an accuracy of 0.7433333333333333

--Considering adding the 3 feature

Adding that feature gives an accuracy of 0.84

--Considering adding the 4 feature

Adding that feature gives an accuracy of 0.74

--Considering adding the 5 feature

Adding that feature gives an accuracy of 0.71

--Considering adding the 6 feature

Adding that feature gives an accuracy of 0.7166666666666667

--Considering adding the 7 feature

Adding that feature gives an accuracy of 0.7366666666666667

--Considering adding the 8 feature

Adding that feature gives an accuracy of 0.7066666666666667

--Considering adding the 9 feature

Adding that feature gives an accuracy of 0.7433333333333333

--Considering adding the 10 feature

Adding that feature gives an accuracy of 0.7033333333333334

On level 1 i added feature 3 to current set

The current set is [3]

On the 2th level of the search tree

--Considering adding the 1 feature

Adding that feature gives an accuracy of 0.8166666666666667

--Considering adding the 2 feature

Adding that feature gives an accuracy of 0.96

--Considering adding the 4 feature

Adding that feature gives an accuracy of 0.83

--Considering adding the 5 feature

Adding that feature gives an accuracy of 0.8233333333333334

--Considering adding the 6 feature

Adding that feature gives an accuracy of 0.8033333333333333

--Considering adding the 7 feature

Adding that feature gives an accuracy of 0.8266666666666667

--Considering adding the 8 feature

Adding that feature gives an accuracy of 0.8133333333333334

--Considering adding the 9 feature

Adding that feature gives an accuracy of 0.8366666666666667

--Considering adding the 10 feature

Adding that feature gives an accuracy of 0.82

On level 2 i added feature 2 to current set

The current set is [3, 2]

…

On the 10th level of the search tree

--Considering adding the 8 feature

Adding that feature gives an accuracy of 0.7566666666666667

On level 10 i added feature 8 to current set

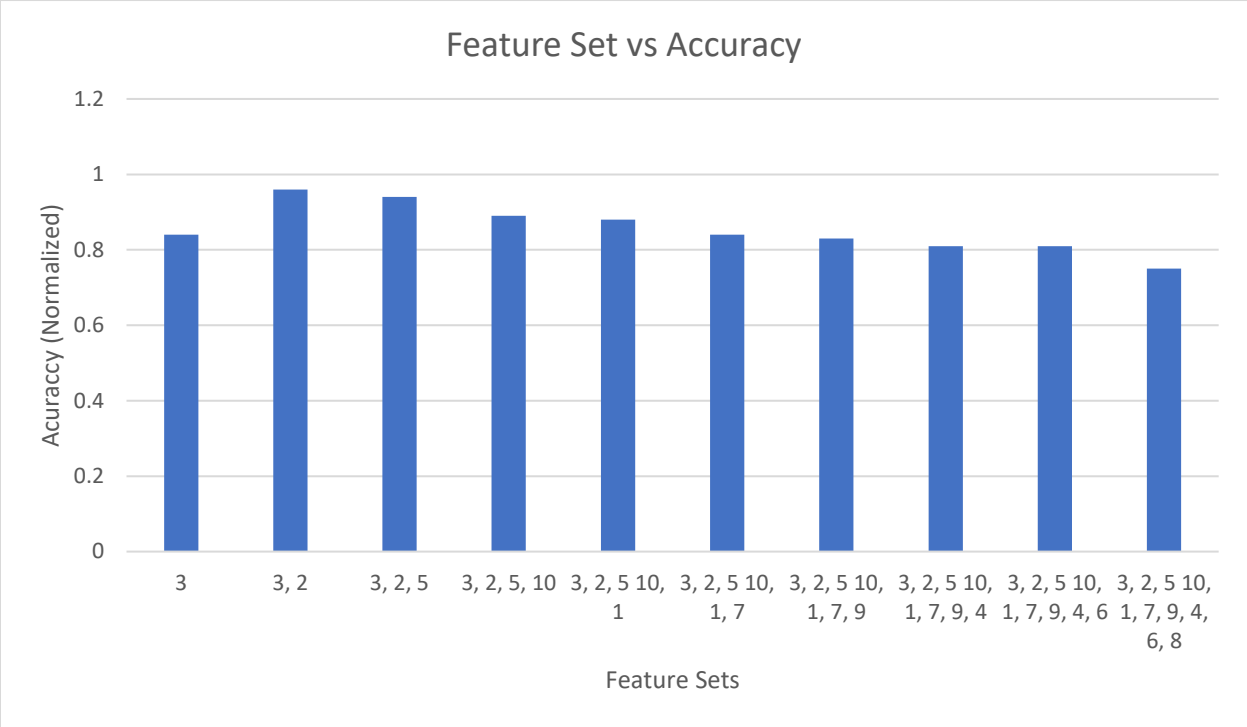
Best accuracy was acheived with features [3, 2] that has an accuracy of 0.96

Analysis

The runtime of the two real test datasets is in Table 1. The small dataset runs within 30 seconds, but the larger one takes a little less than 2 days. I believe the larger dataset takes a longtime due to being programmed in Python and using numpy arrays to make programming simpler. If given more time the runtime could be sped up with a GPU or using multiple threads with a CPU, this would make the larger datasets run much faster.

	Small 24	Large 43
Forward Selection	25 Seconds	1.3 Day
Backward Elimination	27.45 Seconds	1.5 Day

The graph below is of the feature selection for forward selection on small dataset 24:



As more and more features are added the noise causes the accuracy of the classifier to degrade. This trend was also present in the larger dataset but not shown here due to the size. The accuracy would steadily increase hit a peak then decrease, interestingly enough adding the last feature always has a much more significant drop than the other degradation.

Conclusion

After computing both the Forward Selection, and Backward Elimination the best accuracy for both the data sets was 96%. For small dataset 24 features [3, 2] creates the best accuracy, and large dataset 43 [194, 20, 34]. I believe Forward Selection is better than Backward Elimination for these datasets since they ideal feature sets are towards to beginning of the runtime. Not pictured here, the large dataset 43 also peaked in accuracy quickly then slowly decreased with each step size, this can be scene with the ideal feature set being only 3 features.