Project 2 — CS170: Introduction to Artificial Intelligence

Monica Canto — Dr. Eamonn Keogh
SID: 861230103
mcant004@ucr.edu
December 10, 2018

In completing this project, I consulted the following sources:

- The lecture slides + notes on Machine Learning
- Cloud9 online IDE
  - https://c9.io/
- Python 3 documentation
  - https://docs.python.org/3/
- Wikipedia, for definitions of the classification problem and feature selection
  - https://en.wikipedia.org/wiki/Statistical classification
  - https://en.wikipedia.org/wiki/Feature selection

All relevant code is original. Unimportant subroutines which are not entirely original include:

- The **math** module which provides access to mathematical functions such as **sqrt()** to calculate square root.
- The **sorted()** function to perform generic sorting of elements in a list.
- The **operator** module with **itemgetter** which returns an object (or a tuple of values) that obtains an item/s from its operand via a getItem() routine.

CS170 Project 2

Monica Canto, SID: 861230103

December 10, 2018

1 Introduction

This is the second and final project for the Introduction to Artificial Intelligence course taught by Dr. Eamonn Keogh at the University of California, Riverside campus during the Fall 2018 quarter. This write-up consists of my findings throughout completing the project. It explores the Feature Selection with the Nearest Neighbor Algorithm, in which the classifier works with Forward Selection, Backward Elimination, and an additional special search algorithm. My language of choice was Python 3. The entire Python Code is attached.

The classification problem is a problem that identifies to which a set of categories a new instance belongs, on the basis of a certain training data set consisting of instances with an already known categorical membership. This problem is often explored using the Nearest Neighbor Algorithm.

The Nearest Neighbor Algorithm is a simple but competitive algorithm that is used for the classification of data sets. A significant disadvantage for this algorithm is that it presents itself as extremely sensitive to any outlying features within a given data set which are irrelevant. To resolve this, the Nearest Neighbor Algorithm is generalized as the K-Nearest Neighbor (KNN) Algorithm. Measure the distance to the nearest K instances, then let them vote, where K is usually an odd number.

2 Search Comparison

The algorithms that were considered for this project are: K-Nearest Neighbor, Forward Selection, Backward Elimination, and an additional custom search algorithm.

2.1 K-Nearest Neighbor (KNN)

The K-Nearest Neighbor (KNN) Algorithm is a variation of the initial Nearest Neighbor Search Algorithm. This measures the distance to the nearest K instances, then lets them vote. K is usually an odd number.

2.2 Forward Selection

Also referred to as Greedy Forward Selection, this consists of an initial state of no features, and an operator to add features which will best improve the performance of a model until any more additions are no longer useful. Evaluation is done by a K-fold cross validation.

2.3 Backward Elimination

Also referred to as Greedy Backward Elimination, this is simply the Forward Selection Search Algorithm in the reverse perspective; an initial state with a given set of features, then removing irrelevant features in order to improve the performance of a model, until all features have been used.

2.4 Special Search Algorithm

This special search algorithm involves helping to avoid over-fitting in data classification, which prevents poor accuracy due to too many outliers. Known as pruning data, and can be categorized into two types:

- Pre-pruning: halts tree construction early on by not splitting a particular node if it results in the goodness measure falling below a threshold. However, there may be difficulty in selecting a proper threshold.

- Post-pruning: removes branches from a "fully-grown" tree by obtaining a sequence of progressively pruned trees. This is done by using a data set that is different from the training data in order to decide which is the "best pruned tree."

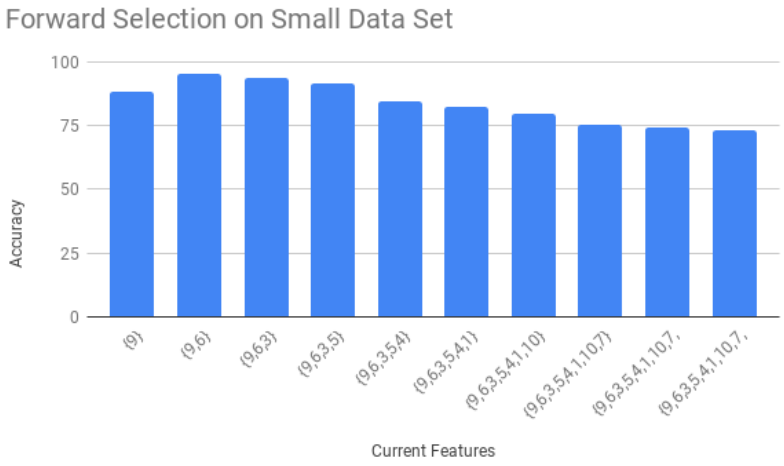
3 Comparison of Search Algorithms on Data Sets

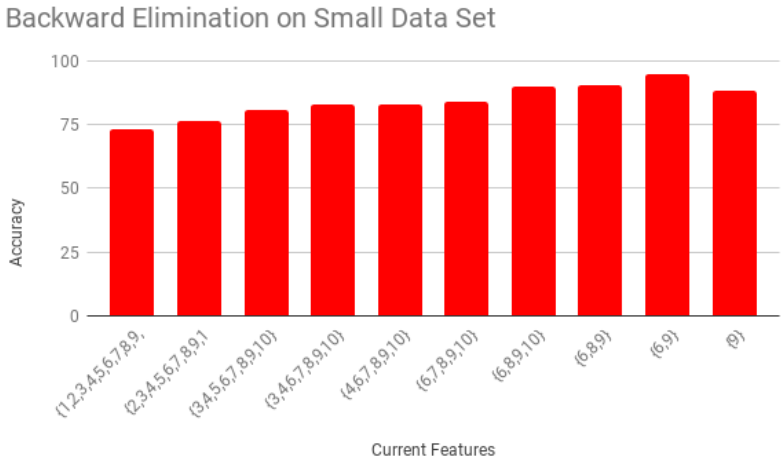
Two data sets were provided for the completion of this project. The first [small] data set consists of 10 features with 200 instances, and the second [large] data set consists of 100 features with 200 instances.

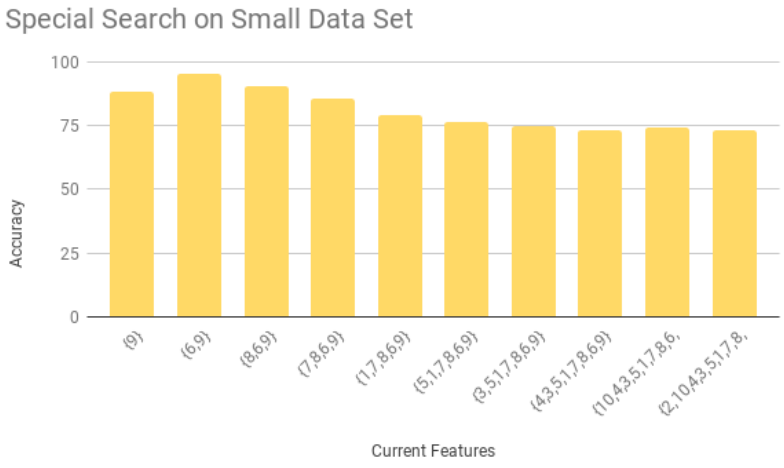
The following graphs show the best accuracy found with certain features, based on Forward Selection, Backward Elimination, and the Special Search. Individual graphs for each algorithm on the small data set are included. Due to the extended running time on the large data set, only the best features from each algorithm are included in one graph.

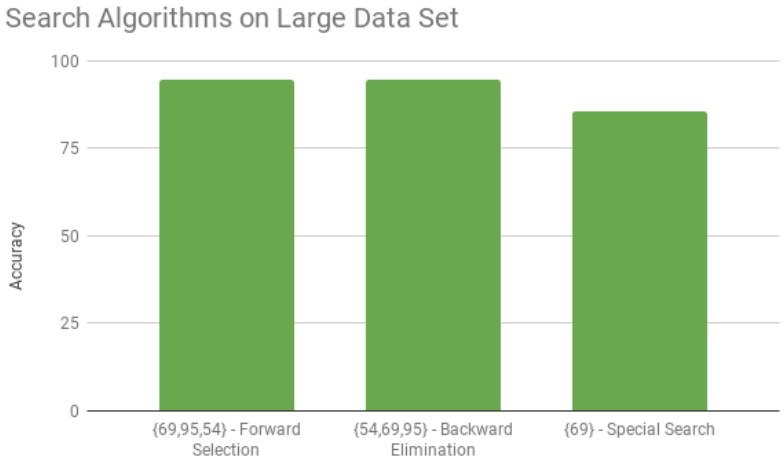
All 3 search algorithms have reported the same highest accuracy with the same features.

On small data set 72, the best features were {6,9} with an accuracy of 95.5%. On large data set 46, the best features were {54,69,95} with an accuracy of 94.5%.


Forward Selection on Small Data Set
100
75
50
25
0
Accuracy
{9}
{9,6}
{9,6,3}
{9,6,3,5}
{9,6,3,5,4}
{9,6,3,5,4,1}
{9,6,3,5,4,1,10}
{9,6,3,5,4,1,10,7}
{9,6,3,5,4,1,10,7,
{9,6,3,5,4,1,10,7,
Current Features


Backward Elimination on Small Data Set
100
75
50
25
0
Accuracy
{1,2,3,4,5,6,7,8,9,
{2,3,4,5,6,7,8,9,1
{3,4,5,6,7,8,9,10}
{3,4,6,7,8,9,10}
{4,6,7,8,9,10}
{6,7,8,9,10}
{6,8,9,10}
{6,8,9}
{6,9}
{9}
Current Features


Special Search on Small Data Set
100
75
50
25
0
Accuracy
{9}
{6,9}
{8,6,9}
{7,8,6,9}
{1,7,8,6,9}
{5,1,7,8,6,9}
{3,5,1,7,8,6,9}
{4,3,5,1,7,8,6,9}
{10,4,3,5,1,7,8,6,
{2,10,4,3,5,1,7,8,
Current Features


Search Algorithms on Large Data Set
100
75
50
25
0
Accuracy
{69,95,54} - Forward Selection
{54,69,95} - Backward Elimination
{69} - Special Search

4 Conclusion

Based on the results from running the provided data sets, it is found that Forward Selection and Backward Elimination tend to fluctuate in accuracy of what are the best features. This is primarily because of the contrast between how these algorithms deal with their given data sets.

Since Forward Selection begins with an empty data set of features then adds features until any further additions are unnecessary, one would attempt to conclude that this algorithm will run faster in order to find the best features with the highest accuracy. Backward Elimination on the other hand starts with all of the features and its corresponding accuracy, then removes irrelevant features one by one until a reasonable amount is left with the best accuracy.

The additional search algorithm essentially reduces the time of having to go through each individual feature and outputs the best accuracy with certain features directly.

5 Trace on Small Data Set

```
monicanto101:~/workspace/cs170/project2 $ python3 main.py
Welcome to the Feature Selection Algorithm!
Type in the name of the file to test: CS170_SMALLtestdata__72.txt

Type in the number of the algorithm you want to test:
1) Forward Selection
2) Backward Elimination
3) Special Algorithm
1

This dataset has 10 features (not including the class attribute), with 200 instances.

Running nearest neighbor with all 10 features, using "leaving-one-out" evaluation,
I get an accuracy of 73.0%

Beginning search.

Using feature(s) {1} accuracy is 64.5%
Using feature(s) {2} accuracy is 56.49999999999999%
Using feature(s) {3} accuracy is 61.5%
Using feature(s) {4} accuracy is 60.5%
Using feature(s) {5} accuracy is 63.0%
Using feature(s) {6} accuracy is 68.0%
Using feature(s) {7} accuracy is 64.5%
Using feature(s) {8} accuracy is 67.0%
Using feature(s) {9} accuracy is 88.5%
Using feature(s) {10} accuracy is 59.5%


Feature set {9} was best, accuracy is 88.5%

Using feature(s) {9,1} accuracy is 82.0%
Using feature(s) {9,2} accuracy is 83.0%
Using feature(s) {9,3} accuracy is 88.0%
Using feature(s) {9,4} accuracy is 84.0%
Using feature(s) {9,5} accuracy is 81.0%
Using feature(s) {9,6} accuracy is 95.5%
Using feature(s) {9,7} accuracy is 82.0%
Using feature(s) {9,8} accuracy is 80.0%
Using feature(s) {9,10} accuracy is 81.5%


Feature set {9,6} was best, accuracy is 95.5%
```

```
Using feature(s) {9,6,3,5,4,1,2} accuracy is 76.0%
Using feature(s) {9,6,3,5,4,1,7} accuracy is 73.0%
Using feature(s) {9,6,3,5,4,1,8} accuracy is 74.5%
Using feature(s) {9,6,3,5,4,1,10} accuracy is 79.5%


(Warning, Accuracy has decreased! Continuing search in case of local maxima)
Feature set {9,6,3,5,4,1,10} was best, accuracy is 79.5%

Using feature(s) {9,6,3,5,4,1,10,2} accuracy is 75.0%
Using feature(s) {9,6,3,5,4,1,10,7} accuracy is 75.5%
Using feature(s) {9,6,3,5,4,1,10,8} accuracy is 73.5%


(Warning, Accuracy has decreased! Continuing search in case of local maxima)
Feature set {9,6,3,5,4,1,10,7} was best, accuracy is 75.5%

Using feature(s) {9,6,3,5,4,1,10,7,2} accuracy is 69.0%
Using feature(s) {9,6,3,5,4,1,10,7,8} accuracy is 74.0%


(Warning, Accuracy has decreased! Continuing search in case of local maxima)
Feature set {9,6,3,5,4,1,10,7,8} was best, accuracy is 74.0%

Using feature(s) {9,6,3,5,4,1,10,7,8,2} accuracy is 73.0%


(Warning, Accuracy has decreased! Continuing search in case of local maxima)
Feature set {9,6,3,5,4,1,10,7,8,2} was best, accuracy is 73.0%

Finished search!! The best feature subset is {9,6}, which has an accuracy of 95.5%
```