Project 2: Feature Selection with Nearest Neighbor

Name: Yongfeng Liang SID: 862128237

Solution:

Dataset	Best Feature Set	Accuracy
Small Number: 45	Forward Selection = {10, 1}	0.95
	Backward Elimination = {1, 10}	0.95
	Custom Algorithm Not implemented	
Large Number: 45	Forward Selection = {27, 10}	0.955
	Backward Elimination = {1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 38}	0.73
	Custom Algorithm Not implemented	

----------<Begin Report>----------

In completing this project, I consulted following resources:

https://www.geeksforgeeks.org/sorting-2d-vector-in-c-set-1-by-row-and-column/

https://www.geeksforgeeks.org/rand-and-srand-in-ccpp/

https://en.wikipedia.org/wiki/Normalization_(statistics)

https://www.geeksforgeeks.org/vector-of-vectors-in-c-stl-with-examples/

https://www.geeksforgeeks.org/2d-vector-in-cpp-with-user-defined-size/

I. Introduction

Things we need to implement in this project:

- Greedy search
- The nearest neighbor classifier and Evaluation using leave-one-out validation
- Feature search using nearest neighbor classifier and the real evaluation function (leave-one-out)

The nearest neighbor algorithm is a very simple, yet very competitive classification algorithm. It does have one major drawback. However, it is sensitive to irrelevant features. Therefore, we must choose the features using feature search.

In feature search, given a set of possible features, we try different combination of features (i.e. all possible feature subsets and choose the combination (subset) that yields the highest accuracy. To do so, we use hill-climbing search.

In this hill-climbing search, each node represents a subset of features, and the evaluation function is the accuracy of the classifier when a particular subset of features is used.

This project has three parts, and we are building the project step by step:

Part I: Implementing the greedy search algorithms. Enter number of features as input.

Part II: Implementing the actual evaluation function (leave-one-out validation) and the Nearest Neighbor Classifier and testing it on sample dataset.

Part III: Run the greedy search algorithms on real data with the actual evaluation function, testing the completed system on the initial small and large dataset to make sure it works correctly. Then testing it on personal small and large datasets.

II. Challenges

- I had to review the material of 2d vectors to recall how to implement and use its functions.
- Did not know what class and functions to define in part 2 of the project until I see the TA's post on Piazza.
- I was confused on how to implement the train project. It turns out that we do not really need train function in this project.
- Did not know how to normalize the dataset until I review the lecture slide and research online.
- Did not know how to implement no feature search until I review the lecture slide and research online.

III. Code Design

There are three classes: Classifier, Validator, and Feature. The Classifier class focuses on testing the instances and storing them according to Euclidean distances. The

Validator stores the list of feature instances and the list of instances labels. It has the function to read the data, normalize the data, and validate the data using leave one out.

The Feature class stores the feature list and the accuracy. It has functions to add feature, drop feature, update accuracy, and print. In addition, we have two important functions for forward selection and backward elimination. There are less important functions for comparing the value in between two vectors and no feature search.

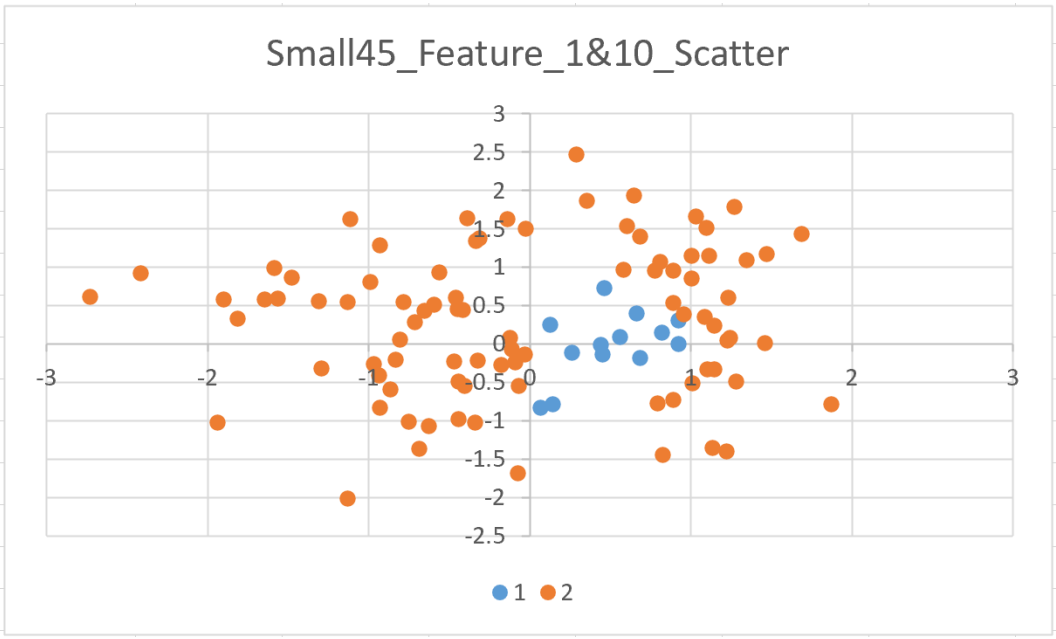
In the main function, we ask the user to enter the name of the file. We read the file and then ask for which one of the two search algorithms to use. Then, we call the corresponding algorithm. It computes the accuracy of feature subsets and if the accuracy begins to drop. Stop the search and report the highest accuracy found.

IV. Dataset details

Small Dataset: 100 instances and 10 features

Large Dataset: 1000 instances and 40 features

Here is the plot of normalized small45 dataset with feature 1 and 10. This feature subset has the highest accuracy of 95% among all other feature subsets. As you can see in the plot, the instances of class 1 are close together and most of the instances of class 2 are spread out and away from class 1 instances. As a result, the accuracy is high because it is good for the leave one out classifier to differentiate them.



V. Algorithms

1. Forward Selection

Forward Selection starts with an empty feature set, then it starts to add one random feature in the set each iteration to compute which set is the best to use. It finds the set that has the highest accuracy of all sets explored (including empty set) and continue the search with that specific set. If none of the sets satisfy the condition (higher than the global maxima), then it ends. Otherwise, it ends when there is no feature to add (full feature set).

2. Backward Elimination

Backward Elimination is the opposite of Forward Selection. It starts with a full feature set, then it starts to remove one random feature in the set each iteration to compute which set is the best to use. It finds the set that has the highest accuracy of all sets explored and continue the search with that specific set. If none of the sets satisfy the condition (higher than the global maxima), then it ends. Otherwise, the search ends when there is no feature to remove (empty set).

VI. Analysis

Experiment 1:

- Comparing Forward Selection vs Backward Elimination.

Forward Selection and Backward Elimination are opposite. They search in the opposite way. Forward Selection starts from an empty set and adds a feature to the set one at a time and compute the accuracy. It stops until all features are added. Backward Elimination start from a full set removes a feature from the set one at a time and compute the accuracy. It stops until all features removed.

- Compare accuracy with no feature selection vs with feature selection.

When using no feature selection, you look for the most frequent class label that appears in the entire dataset and assume all the data belong to that class. If there are two class labels like we have in this case, then you are guaranteed to get over 50% accuracy with no feature search. Even so, it is not too bad to use feature selection assuming that there are worst cases. The more class labels a dataset has, the worst the accuracy you get from no feature selection. So, it is recommended to use feature selection when there are more class labels. Feature selection also has drawbacks to consider in this project, which are irrelevant features, having too many irrelevant features in the subset could make the algorithm inefficient and produce low accuracy. When the dataset is large, feature search takes more time to find the best subset and accuracy.

- Compare feature set and accuracy for forward selection vs backward elimination.

Both Forward Selection and Backward Elimination are designed to deal with irrelevant features in the dataset. When the dataset and feature set are small, both

algorithms have similar performance. Forward selection is good for large number of features because it starts as an empty set and usually it does not have to reach the end (full set). Otherwise, use Backward Elimination would typically be a good approach. In terms of accuracy, Irrelevant features impact more on Backward Elimination than Forward Selection. Because Backward Elimination could start from low accuracy to high accuracy by removing irrelevant features, and this process takes time. Forward Selection tends to be the opposite.

Experiment 2: Effect of normalization

- Compare accuracy when using normalized vs unnormalized data.

The nearest neighbor algorithm is sensitive to the units of measurement (data is widely spread out, high std dev). For small dataset with high standard deviation, we must normalize the data, otherwise the accuracy will be low. However, for large dataset with high standard deviation, the change of accuracy would not be too obvious when using normalized vs unnormalized data.

VII. Conclusion

In conclusion, this project is not too complicated, the implementation is not hard as well. Once you understand the general idea and what functions you need to implement, you will be able to finish it quickly. I have found that the optimality of a search algorithm is important because an ordinary search algorithm can take a long time on large datasets. I believe a potential improvement to this approach of doing feature selection could be to combine forward selection and backward elimination together such that the new algorithm will add a feature and remove another feature at the same time (Bi-directional Search).

VIII. Trace of your small dataset

Small Dataset 45 with Forward Selection

```
Welcome to Yongfeng Liang's Feature Selection Algorithm.

Type in the name of the file to test: cs_170_small45.txt

This dataset has 10 features (not inculding the class attribute), with 100 instances

Please wait while I normalize the data...       Done!


Type the number of the algorithm you want to run.

       *Forward Selection
       *Backward Elimination

                        1

Running nearest neighbor with no features(default rate), using "leaving-one-out" evaluation, I get an accuracy of 87%

Beginning search.

Using feature(s) {1} accuracy is 79%
Using feature(s) {2} accuracy is 77%
Using feature(s) {3} accuracy is 83%
Using feature(s) {4} accuracy is 81%
Using feature(s) {5} accuracy is 84%
Using feature(s) {6} accuracy is 77%
Using feature(s) {7} accuracy is 83%
Using feature(s) {8} accuracy is 78%
Using feature(s) {9} accuracy is 70%
Using feature(s) {10} accuracy is 89%

Feature set {10} was the best, accuracy is 89%

Using feature(s) {10,1} accuracy is 95%
Using feature(s) {10,2} accuracy is 79%
Using feature(s) {10,3} accuracy is 84%
Using feature(s) {10,4} accuracy is 84%
Using feature(s) {10,5} accuracy is 83%
Using feature(s) {10,6} accuracy is 88%
Using feature(s) {10,7} accuracy is 88%
Using feature(s) {10,8} accuracy is 85%
Using feature(s) {10,9} accuracy is 78%

Feature set {10,1} was the best, accuracy is 95%
Using feature(s) {8} accuracy is 78%
Using feature(s) {9} accuracy is 70%
Using feature(s) {10} accuracy is 89%

Feature set {10} was the best, accuracy is 89%

Using feature(s) {10,1} accuracy is 95%
Using feature(s) {10,2} accuracy is 79%
Using feature(s) {10,3} accuracy is 84%
Using feature(s) {10,4} accuracy is 84%
Using feature(s) {10,5} accuracy is 83%
Using feature(s) {10,6} accuracy is 88%
Using feature(s) {10,7} accuracy is 88%
Using feature(s) {10,8} accuracy is 85%
Using feature(s) {10,9} accuracy is 78%

Feature set {10,1} was the best, accuracy is 95%

Using feature(s) {10,1,2} accuracy is 89%
Using feature(s) {10,1,3} accuracy is 90%
Using feature(s) {10,1,4} accuracy is 93%
Using feature(s) {10,1,5} accuracy is 86%
Using feature(s) {10,1,6} accuracy is 88%
Using feature(s) {10,1,7} accuracy is 87%
Using feature(s) {10,1,8} accuracy is 91%
Using feature(s) {10,1,9} accuracy is 89%

(Warning, Accuracy has decreased!)

Finished search!! The best feature subset is {10,1}, which has an accuracy of 95%
Press <RETURN> to close this window...
```

Small Dataset 45 with Backward Elimination

```
Welcome to Yongfeng Liang's Feature Selection Algorithm.

Type in the name of the file to test: cs_170_small45.txt

This dataset has 10 features (not inculding the class attribute), with 100 instances

Please wait while I normalize the data...       Done!


Type the number of the algorithm you want to run.

       *Forward Selection
       *Backward Elimination

                        2

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

Beginning search.

Using feature(s) {2,3,4,5,6,7,8,9,10} accuracy is 74.0%
Using feature(s) {1,3,4,5,6,7,8,9,10} accuracy is 75.0%
Using feature(s) {1,2,4,5,6,7,8,9,10} accuracy is 70.0%
Using feature(s) {1,2,3,5,6,7,8,9,10} accuracy is 74.0%
Using feature(s) {1,2,3,4,6,7,8,9,10} accuracy is 74.0%
Using feature(s) {1,2,3,4,5,7,8,9,10} accuracy is 74.0%
Using feature(s) {1,2,3,4,5,6,8,9,10} accuracy is 72.0%
Using feature(s) {1,2,3,4,5,6,7,9,10} accuracy is 72.0%
Using feature(s) {1,2,3,4,5,6,7,8,10} accuracy is 73.0%
Using feature(s) {1,2,3,4,5,6,7,8,9} accuracy is 75.0%

Feature set {1,3,4,5,6,7,8,9,10} was the best, accuracy is 75.0%

Using feature(s) {3,4,5,6,7,8,9,10} accuracy is 75.0%
Using feature(s) {1,4,5,6,7,8,9,10} accuracy is 71.0%
Using feature(s) {1,3,5,6,7,8,9,10} accuracy is 75.0%
Using feature(s) {1,3,4,6,7,8,9,10} accuracy is 77.0%
Using feature(s) {1,3,4,5,7,8,9,10} accuracy is 78.0%
Using feature(s) {1,3,4,5,6,8,9,10} accuracy is 76.0%
Using feature(s) {1,3,4,5,6,7,9,10} accuracy is 78.0%
Using feature(s) {1,3,4,5,6,7,8,10} accuracy is 76.0%
Using feature(s) {1,3,4,5,6,7,8,9} accuracy is 72.0%

Feature set {1,3,4,5,7,8,9,10} was the best, accuracy is 78.0%
```

```
Feature set {1,3,4,5,7,8,9,10} was the best, accuracy is 78.0%

Using feature(s) {3,4,5,7,8,9,10} accuracy is 80.0%
Using feature(s) {1,4,5,7,8,9,10} accuracy is 79.0%
Using feature(s) {1,3,5,7,8,9,10} accuracy is 77.0%
Using feature(s) {1,3,4,7,8,9,10} accuracy is 78.0%
Using feature(s) {1,3,4,5,8,9,10} accuracy is 82.0%
Using feature(s) {1,3,4,5,7,9,10} accuracy is 79.0%
Using feature(s) {1,3,4,5,7,8,10} accuracy is 77.0%
Using feature(s) {1,3,4,5,7,8,9} accuracy is 78.0%

Feature set {1,3,4,5,8,9,10} was the best, accuracy is 82.0%

Using feature(s) {3,4,5,8,9,10} accuracy is 76.0%
Using feature(s) {1,4,5,8,9,10} accuracy is 77.0%
Using feature(s) {1,3,5,8,9,10} accuracy is 82.0%
Using feature(s) {1,3,4,8,9,10} accuracy is 83.0%
Using feature(s) {1,3,4,5,9,10} accuracy is 82.0%
Using feature(s) {1,3,4,5,8,10} accuracy is 82.0%
Using feature(s) {1,3,4,5,8,9} accuracy is 78.0%

Feature set {1,3,4,8,9,10} was the best, accuracy is 83.0%

Using feature(s) {3,4,8,9,10} accuracy is 82.0%
Using feature(s) {1,4,8,9,10} accuracy is 84.0%
Using feature(s) {1,3,8,9,10} accuracy is 85.0%
Using feature(s) {1,3,4,9,10} accuracy is 81.0%
Using feature(s) {1,3,4,8,10} accuracy is 80.0%
Using feature(s) {1,3,4,8,9} accuracy is 80.0%

Feature set {1,3,8,9,10} was the best, accuracy is 85.0%

Using feature(s) {3,8,9,10} accuracy is 83.0%
Using feature(s) {1,8,9,10} accuracy is 84.0%
Using feature(s) {1,3,9,10} accuracy is 82.0%
Using feature(s) {1,3,8,10} accuracy is 89.0%
Using feature(s) {1,3,8,9} accuracy is 82.0%

Feature set {1,3,8,10} was the best, accuracy is 89.0%

Using feature(s) {3,8,10} accuracy is 85.0%
Using feature(s) {1,8,10} accuracy is 91.0%
Using feature(s) {1,3,10} accuracy is 90.0%
```

```
Using feature(s) {3,8,9,10} accuracy is 83.0%
Using feature(s) {1,8,9,10} accuracy is 84.0%
Using feature(s) {1,3,9,10} accuracy is 82.0%
Using feature(s) {1,3,8,10} accuracy is 89.0%
Using feature(s) {1,3,8,9} accuracy is 82.0%

Feature set {1,3,8,10} was the best, accuracy is 89.0%

Using feature(s) {3,8,10} accuracy is 85.0%
Using feature(s) {1,8,10} accuracy is 91.0%
Using feature(s) {1,3,10} accuracy is 90.0%
Using feature(s) {1,3,8} accuracy is 81.0%

Feature set {1,8,10} was the best, accuracy is 91.0%

Using feature(s) {8,10} accuracy is 85.0%
Using feature(s) {1,10} accuracy is 95.0%
Using feature(s) {1,8} accuracy is 82.0%

Feature set {1,10} was the best, accuracy is 95.0%

Using feature(s) {10} accuracy is 89.0%
Using feature(s) {1} accuracy is 79.0%

(Warning, Accuracy has decreased!)

Finished search!! The best feature subset is {1,10}, which has an accuracy of 95.0%
Press <RETURN> to close this window...
```