Project 2 & Extra Credit dataset CS205: Artificial Intelligence

Abhay Singh Instructor: DR. Eamonn Keogh
SID: 862041345
asing073@ucr.edu
20-March -2018

Project 2: Feature Selection Problem.

In completion of this project the following resources were consulted:

- MATLAB reference website
- Book, Artificial Intelligence: A modern approach and class slides for references.
- "Using kNN Model for Automatic Feature Selection", by Gongde Guo et. all.
- The Bike sharing rental database at https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset#
- Feature Engineering Blog at machinelearningmastery.com https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/
- "Introduction to variable and feature selection" by Isabelle Guyon.

All the code included is original, except for subroutines to do tasks unrelated to model building like combnk(..).

CS205 - Project II

Feature Selection with KNN

Objective - For any machine learning task to be efficient feature engineering is extremely crucial. The core of feature engineering is feature selection. Here we discuss methods to select the best possible features through Wrapper method of feature selection with KNN.

INTRODUCTION

Feature selection is important for building any given machine learning model not just in terms of accuracy of the model but also the dimensions of the feature. Learning models suffer from a "Curse of dimensionality". Thus the feature set must be as small as possible for our model to be as accurate as possible.

In feature engineering the task of feature selection is done after running a chi-square test which would indicate any dependence between features. We can remove the redundant features and proceed with feature selection. Features can also be ranked by a measure of information gain. NP-hard feature selection problems are typically solved by a meta-heuristic approach namely, wrapper, filter and embedded methods. If looked at as an optimization problem we can do feature search through any optimization algorithm.

Here for the second project the task is to build a Wrapper method using nearest neighbor. Tests were run for selecting best features on a given pair of dataset. I was assigned CS205_BIGtestdata_4 as the big test file and CS205SMALLtestdata_53 as the small test file. I will describe my implementations, learning and observations on the dataset.

Finally, for the extra credit I attempt to find strong features and try to justify my findings on the bike sharing dataset at UCI data repository.

The paper will present the implemented algorithms and their basic principle. A strong argument supporting why the each algorithm is better than the last will be made.

Wrapper Method

The Problem statement asks us to implement a forward selection and backwards elimination algorithm and another algorithm that improves on any of them. The following algorithms were implemented as a part of this project

I. Brute-Force Forward Selection
II. Brute-Force Backward Elimination
III. Greedy Forward Selection
IV. Greedy Backward Elimination
V. Hill Climb Forward Selection

Implementation

As discussed in the class presentation, the feature selection problem can be viewed as a search/optimization problem. The operators being adding or removing a feature. The accuracy produced by the features can be seen as a fitness function.

I. Brute-Force Forward Selection

This is the equivalent feature selection for blind breadth first search. We go about traversing the tree a level at a time. The feature search starts from an empty set and goes on to add features to this empty set one at a time till all the features are being considered. Since this is a search problem it is NP-hard.

II. Brute-Force Backward Elimination

This is the exact opposite of the last algorithm in terms of direction of tree traversal. It starts with a full set of features and starts removing a feature at a time while traversing the tree backwards.

III. Greedy Forward Selection

This algorithm traverses the tree in a way that uniform cost tree would do. Where cost can be imagined as errors that the model makes on the cross-validation data. So the algorithm traverses a level (basically the children of a node) and picks the node with the least error or highest accuracy to expand. Note that here we are not keeping a track of the nodes in the tree so we only choose the best node that gets expanded in a layer.

This algorithm is popularly known as the Sequential Forward selection algorithm. This algorithm is also somewhat similar to feature selection algorithms based on information gain. Since this is a greedy algorithm, the answer may not be the best one.

IV. Greedy Backward Elimination

Popularly known as the Sequential Backward Elimination Algorithm. It traverses the tree from the complete feature set and removes a feature to create a child. Similar to the last subsection, we expand a node, and keep the child that produces the maximum accuracy. Again since this is a greedy algorithm, it can give us an answer but it may not be the best one.

IV. Hill Climb Forward Selection

As instructed in the class, when in forward selection, the accuracy starts to decline on expanding a layer (adding a new feature in forward selection) it rarely increases on going forward. Let's say even if we do get a marginal improvement expanding our way to the complete feature set, the resultant feature set may suffer from the "curse of dimensionality", i.e. it is more likely to over-fit, the model will take more time to compute and its visualization will be harder. Therefore for this algorithm stops traversing the tree after the accuracy starts going down (or even stay equal).

For this part we build on top of the Greedy Forward selection algorithm. We track the local maxima or maximum accuracy at obtained by sequentially adding a feature. If this local maxima does not improve the accuracy at its parent's level we return the accuracy and feature set of the parent. It can be imagined as the hill climbing search.

Additional note on the algorithms, in case of an accuracy tie, which is more likely to happen with such small dataset, the node with lesser features is considered the best so far solution.

All of these have been implemented in MATLAB and the source code has been attached. The code may also be found at the link: goo.gl/sr1DSr

Defending Algorithms (I,II,V)

Please note as stated before, both sequential forward selection and sequential backward elimination are greedy solutions. They produce a good result which may not always be the best. Here are some theoretical points in favor of algorithms I, II and V:

- (I,II)Brute Force approach in forward and backward search will always produce the most optimal result because it will traverse the whole tree and not leave out any node. Thus, if there is a really good solution in the tree, the algorithm is bound to find it.
- (I,II)The Greedy algorithm assumes that if a feature is bad with a given combination, the feature is not considered in the solution. However, it may be the case that that feature in combination with some other feature may give drastic improvement accuracy.
- (V)When it comes to the hill climbing forward feature selection, It stops traversing down in the tree once no improvement of accuracy is found at a particular level. Thus, it is expected to deliver a feature set that gets us accuracy close to the one we would get from Greedy Search. The feature set would theoretically be computed faster, and should be at a lesser risk of overfitting.

Let us test the theoretical claims on the given datasets.

Results

The algorithms were used to train a nearest neighbor algorithm as a wrapper. The accuracy was measured by leave one out cross-validation. Also, it was necessary to determine the value of k. The results presented here take k as 1. The results (accuracy) deteriorate very quickly the moment we change k to anything besides 1.

The given test files mentioned in the introduction section were used to compute the time taken for each algorithm, the following observations were made with regards to the time take by different algorithms:

	(I)	*(II)*	*(III)*	*(IV)*	*(V)*
Small Data	36.27	36.60	1.69	1.77	0.838
Big Data	53563.48	55639.60	52.73	61.09	4.31

Table 1 Runtime comparison. Reported in seconds.

The following features were reported to have the best accuracy by each algorithm:

	(I)	*(II)*	*(III)*	*(IV)*	*(V)*
Small Data	{1,6} 95.0%	{1 6 } 95.0%	{1 6 } 95.0%	{1 6 } 95.0%	{1 6 } 95.0%
Big Data	{4 31 9 30 } 95.0%	{4 31 9 30} 95.0%	{4 31 30 9 } 95.0%	{1 2.. 49 }* 86.0%	{4 31} 93.0%

Table 2 Feature-set with accuracy

*A long array of features, basically a horrible solution

The above tests were run on an Intel i7 3rd generation, 6GB RAM, 2.6GHz computer. The Following diagrams are a scatter plot of the data in the datasets tested using features selected by our algorithm. Note that we can see distinct regions in both the cases validating the claim that the computed features produce a good result.

Although not documented in the results, by changing the value of k from anything but 1 reduces the maximum accuracy achieved by any feature set. For instance, setting k to 3 and running (V) gave the feature-set {8, 1} with 83% accuracy.

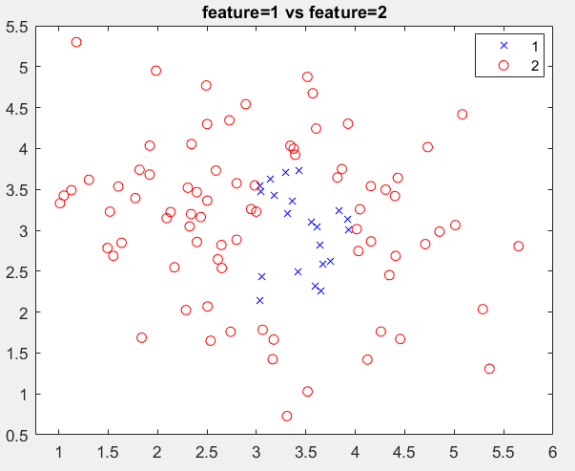

Figure 1 Scatter plot for smaller dataset, Feature 1(in figure) is feature 1 and Feature 2(in figure) is feature 6 accuracy: 95.0%

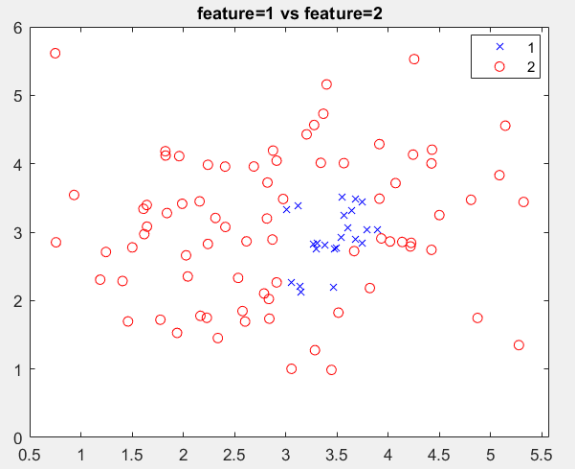

Figure 2 Scatter Plot for the big dataset Feature 1(in plot) is feature 4 and Feature 2(in plot) represents Feature 31. Accuracy: 93.0%

Observations

Having compiled the results, plotted and visualized them, the following observations can be made:

1. Although the Brute force algorithms (I) and (II) take exponentially longer than their standard sequential forward selection or backward selection peers, they did produce the optimal feature set.

2. On the small data set, all the algorithms returned the optimal set of features but the runtime was: Bruteforce>>Greedy>Hill Climbing
3. On performing greedy backward elimination on big dataset, it was seen that it went on to give an absurd feature set. Notably, the brute force version got it right.
4. Time taken by algorithm (V) is much smaller compared to algorithm (III) especially when big dataset is concerned.
5. Even while reducing the runtime, it does not compromise much on the accuracy of the model.

Conclusion

A total of five feature selection algorithms were explored. The forward selection, backward elimination, Hill climbing Forward selection and the brute force methods. We saw that the hill climbing was faster without straying far away from the optimum. The brute force algorithms provide us with the optimum answer but take exponentially greater time to produce results.

Extra Credit Dataset

A dataset from the UCI's repository on bike rental services was chosen to compute the best possible set of features. The data has the following features:

1. Season-converted to integers
2. Month
3. Hours the service was run
4. Holiday – if the day was a holiday
5. Weather- Converted to integer.
6. Temperature
7. Weekday/Weekend
8. Humidity
9. Wind Speed
10. Casual Customers
11. Registered Users
12. Day of the week

The Label used to identify two classes was if more than 4000 bikes were rented, if yes class is 2 if no the class is 1.

According to the selection algorithm,

Registered Users and Casual customers together as a feature could predict with an accuracy of 99.2% that if it would be a good at the bike rental service. Please refer to figure 3 for a scatter plot for the same. However if this were to look obvious, I tested the dataset again after removing the registered users feature. This time around it was found that the number of casual riders with two other features could yield an 83.8% accuracy. The other two features were if the day is a weekday and the temperature in decreasing order of importance.

Since bikers do not have the luxury of an air conditioner or a heater, the temperature factor could be expected. On the other hand if casual riders were to pick a bike up they would be more likely to do it on a weekend than a weekday.

The strongest four features in the dataset to predict if there would be more than 4000 bikes rented are registered users of the facility, number of casual riders they serve on average, Weekend or not and temperature weekday. Thus, the findings do make sense.

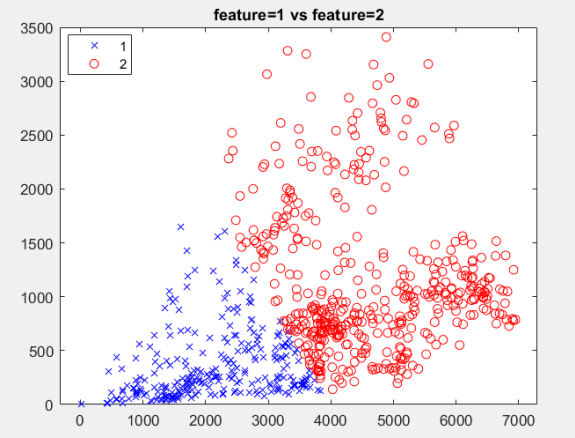

Figure 3 Bike-share feature scatter. Feature 1(in plot) is registered users Feature 2(in plot) is casual users, accuracy: 99.2%