CSE512 - Project - Floating search methods in feature selection

AJAY GOPAL KRISHNA
112688765
agopalkrishn@cs.stonybrook.edu

ABHIRAM M KAUSHIK
112686262
amkaushik@cs.stonybrook.edu

***Abstract*—The purpose of this project was to analyze and implement different feature selection algorithms presented in the research paper [1]. Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of our model. The paper [1] mainly describes two algorithms Sequential forward floating Selection (SFFS) and Sequential backward floating Selection (SBFS) to perform feature selection. For our project we have mainly implemented the SFFS algorithm in python.**

I. Introduction

Feature selection is the process of automatically selecting features in our data which are most useful. The features which has most significance/impact and most describes our dependent variable should be considered. The subset of features selected from the total **D** feature set should be such that it must not significantly degrade the performance of the recognition system. Feature selection helps to reduce the number of input dimensions which can greatly affect the computational cost while modeling and also in some cases might increase the accuracy/ performance of the model

Feature selection [2] can be broadly classified into

- Unsupervised : Feature selection doesn't depend on the dependent variable
- Supervised : Feature selection involves evaluating the relationship between each feature and the target variable

The algorithms presented in the paper talks above Supervised selection. Further in supervised feature selection, selecting/choosing a feature for the D feature set must depend on its significance w.r.t the dependent variable. This significance can be assumed as a suitable criterion function that has been chosen to evaluate the effectiveness of the selected feature.

II. Significance

The paper [1] describes significance as follow,
Let $Y = y_i : 1 \leq i \leq D$ be the set of **D** available features.
$X_k = x_i : 1 \leq i \leq k, x_i \in Y$ be the k features selected from Y

The individual significance of i^{th} feature is given by $S_0(y_i) = J(y_i)$ where $J(y_i)$ is significance value based on some feature selection criterion.
The significance of adding a new feature f_i which is not part of the set X_k is given by, $S_{k+1}(f_i) = J(X_k + f_i) - J(X_k)$. That is the change in significance by adding the f_i^{th} feature to X_k

We can say that feature f_j from the set of Y features that are not present in X_k is

- Most significant : If $J(X_k + f_j) = \max_{1 \leq i \leq D-k} J(X_k + f_i)$
  That is the maximum significance we get by adding a new feature f_i
- Least significant : If $J(X_k + f_j) = \min_{1 \leq i \leq D-k} J(X_k + f_i)$
  That is the worst significance we get by adding a new feature f_i

There are 2 techniques to calculate the feature significance [3].

- Filter Methods : Apply a statistical measure to assign a scoring to each feature
- Wrapper Methods : use a classifier/heuristics to evaluate subsets by their predictive accuracy (on test data) by statistical re-sampling or cross-validation

Filter methods usually are **faster** when compared to wrapper methods. This is usually due to the fact that wrapper methods involve heavy computation of training the model and prediction on a small subset to get the significance score.

Also, filters tend to be more generic as they evaluate the intrinsic properties of the data instead of fitting a model, while wrapper-methods can output more accurate results while selecting features as they can perform techniques like cross-fold validation and avoid over-fit.

III. Why SFFS

We already have many algorithms which does feature selection. Few of the existing algorithms include Sequential Forward Search (SFS), Sequential Backward Search (SBS). The paper describes that these methods are generally sub-optimal and suffer from the so-called "nesting effect". That in case of the "bottom-up" approach i.e SFS the features once selected later cannot be discarded and in case of the "top-down" approach i.e SBS the discarded features cannot be added back.

Further there is another algorithm called the branch-and-bound [4]. This algorithm is optimal in terms of the selected features but is computationally expensive and needs huge computer memory.

Therefore the existing algorithms for feature selection are either computationally feasible like SFS and SBS but dont select the optimal features, or they yield optimal or almost optimal feature subsets but are computationally expensive.

IV. SFFS

The basic idea of the algorithm presented in the paper [1] is the best feature set is constructed by adding to and/or removing features from the current feature set (initialized to null at start), until the required cardinality of selected features is obtained.

The algorithm is termed as **floating** because "the resulting dimensionality of the selected features in respective stages of the algorithm is not changing monotonously but is actually "floating" up and down" due to inclusion/deletion at each stage/iteration.

Algorithm 1: SequentialForwardSelection

Result: return X_k
$X_0 = \emptyset$
while $|X_k| \rightarrow k\_features$ **do**
  $f_j \leftarrow$ **most_significant_feature**$(Y - X_k)$
  $X_{k+1} = X_k + f_j$
  $k \leftarrow k + 1$
end

Algorithm 2: SequentialBackwardSelection

Result: return X_k
$X_0 = Y$
while $|X_k| \rightarrow k\_features$ **do**
  $f_j \leftarrow$ **least_significant_feature**$(Y - X_k)$
  $X_{k+1} = X_k - f_j$
  $k \leftarrow k + 1$
end

V. Results

We have implemented the below feature selection methods using **python**

- SequentialForwardSelection
- SequentialBackwardSelection
- SequentialForwardFloatingSelection

To calculate the significance of features we have mainly followed the **wrapper** based approach. We have used **k-NN** algorithm with $k = 1$ and a **5-fold cross-validation** to get the accuracy. This accuracy is used as our significance parameter to determine best/worst features.

Algorithm 3: SequentialForwardFloatingSelection

Result: return X_k
$X_0 = \emptyset$
while $|X_k| \rightarrow k\_features$ **do**
  Step 1 : **Inclusion** $f_j \leftarrow$ SequentialForwardSelection
  $X_{k+1} = X_k + f_j$
  Step 2 : **Conditional exclusion** $f_i \leftarrow$ SequentialBackwardSelection
  if $f_j == f_i$ **then**
    $k \leftarrow k + 1$
    GOTO STEP 1;
  else
    $X'_k = X_{k+1} - f_i$; $J(X_k) \leftarrow J(X'_k)$
    GOTO STEP 3;
  end
  STEP 3 : **Continue Exclusion** $f_s \leftarrow$ SequentialBackwardSelection
  if $J(X'_k - f_s) \leq J(X_{k-1})$ **then**
    $X_k \leftarrow X'_k$; $J(X_k) \leftarrow J(X'_k)$
    GOTO STEP 1;
  else
    $X'_{k-1} = X'_k - f_s$
    $k \leftarrow k - 1$; $J(X_k) \leftarrow J(X'_k)$
    GOTO STEP 3;
  end
end

We have also implemented the **mahalanobis distance** metric [5] to calculate the feature significance . The code presented gives a choice to select between a k-NN wrapper based approach or a filter based approach which uses mahalanobis distance to measure the significance.

Initially as SFFS uses both inclusion and condition exclusion steps we first implemented SFS and SBS. Using these as base classes we have implemented SFFS.

The paper [1] suggests that the selection of a feature set from data showing high statistical dependencies provides a more discriminative test. The mushroom dataset from UCI Machine Learning Repository [6] contains 22 features and inter-feature correlation is significant, due to these reasons this dataset was chosen.

The above dataset was cleaned by dropping the "NaN" values. Also, all categorical data was converted to numerical data by using **label-encoding**. The cleaned dataset can be found in [7]

The below table shows the run-time for SFFS-algorithm w.r.t k-features.

k	Time in sec
2	9.6
5	23.23
8	40
12	56.44
15	72.68

If we use **filter method** to calculate the feature significance, a huge difference in computation time was observed.

VI. Verification

Following were the best feature indexes which was obtained by running our code for $k = 15$.
1, 3, 4, 5, 6, 7, 9, 11, 12, 13, 14, 15, 16, 17, 18

To check the correctness of our algorithm, we cross-verified our results with an existing SFFS library from mlxtend [8].

We used a similar k-NN model as described above to calculate the significance of the features and passed it to the mlxtend model.

The best features for k=15 from the mlextend model are,
1, 3, 4, 5, 6, 7, 9, 11, 12, 13, 14, 15, 16, 17, 18

From the above output we can verify that the results are consistent.

The paper [1] further compares the run-time of different feature selection algorithms w.r.t computation time. Also, the paper verifies the features selected by different algorithms with the optimal features. Verification of this was beyond the scope of this project

References

[1] http://library.utia.cas.cz/separaty/historie/somol-floating%20search%20methods%20in%20feature%20selection.pdf

[2] https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/

[3] http://research.cs.tamu.edu/prism/lectures/pr/pr_l11.pdf

[4] Narendra, P.M. and K. Fukunaga (1977). A branch and bound algorithm for feature subset selection.

[5] https://aip.scitation.org/doi/pdf/10.1063/1.4915708

[6] https://archive.ics.uci.edu/ml/datasets/Mushroom

[7] https://drive.google.com/file/d/1FHpot9qiVdmqThi-ST4eTacwWqmZx5uS/

[8] http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/