Feature Selection using dependent feature removal and Probabilistic Neural Networks.

Kushal Karmani 2015A7PS0091G
Devesh Dhole 2015A7PS0102G
Faria Craig Baldwin Joseph 2015A7PS0120G
Surya Pratap Mehra 2015A7PS0063G
Shubhank Maheshwari 2015A7PS0099G

***Abstract*—Feature selection is one of the most important part of solving information retrieval and data-mining problems as it helps improve performance without significantly affecting accuracy. We propose a novel method of feature selection by using dependent feature reduction followed by a wrapper method based on Probabilistic Neural Networks. The purpose of the proposed method is to reduce the computational complexity and increase the classification accuracy of the selected feature subsets. Dependence between every pair of two attributes is determined based on the average of various correlation metrics of their joint values. We put all such pairs with average correlation above a threshold into a set and sort the set in descending order of their correlation. From each such pair, we evaluate both features using FSPP2(Feature based Sensitivity of Posterior Probabilities) measure, which finds difference in score between presence and absence of a feature using sensitivity analysis of SVM probabilistic outputs and select the better feature from the two and at the same time remove the poorer feature from all other pairs. After this we use a combination of Probabilistic Neural Networks and repeated bit-wise gradient descent with re-sampling for feature selection using Probabilistic Neural Networks.**

***Index Terms*—Dimensionality Reduction; Correlation; SVM; Probabilistic Neural Networks**

I. Introduction

A. Importance of Feature Selection

The central idea of using a feature selection technique is that the web data contains many attributes or features that are either redundant or irrelevant, and can thus be safely removed without incurring much loss of information. The concept of redundancy and irrelevance are two distinct notions, since one relevant feature may be redundant in the presence of another relevant feature with which it is strongly correlated, it is the very idea which inspires first part of our work.

Feature selection becomes more important than ever when the size of data set is huge. You need not use every feature at your disposal for creating an algorithm. You can assist your algorithm by feeding in only those features that are really important. In many cases it is observed that a model trained on a subset of the features gives a much greater accuracy than a model trained on all the features.

B. Feature Selection Algorithms

There are three general classes of feature selection algorithms: Filter methods, Wrapper methods and Embedded methods.

1) Filter Methods: Filter feature selection methods apply a statistical measure to assign a scoring to each feature. The features are ranked by the score and are either retained or removed from the data set. The methods are often uni variate and consider the feature independently, or with regard to the dependent variable.

Some examples of some filter methods include the Chi squared test, information gain and correlation coefficient scores.

2) Wrapper Methods: Wrapper methods consider the selection of a set of features as a search problem, where different combinations are prepared, evaluated and compared to other combinations. A predictive model like a classifier is used to evaluate a combination of features and assign a score based on model accuracy.

The search process may be methodical such as a best-first search, it may stochastic such as a random hill-climbing algorithm, or it may use heuristics, like forward and backward passes to add and remove features.

An example of a wrapper method is the recursive feature elimination algorithm.

3) Embedded Methods: Embedded methods learn which features best contribute to the accuracy of the model while the model is being created. The most common type of embedded feature selection methods are regularization methods.

Regularization methods are also called penalization methods that introduce additional constraints into the optimization of a predictive algorithm (such as a regression algorithm) that bias the model toward lower complexity (fewer coefficients).

Examples of regularization algorithms are the LASSO, Elastic Net and Ridge Regression.

C. Motivation

When ever you come across a new Machine Learning Problem you often face the following problems

- Huge number of parameters.
- Long training time.
- Over-fitting.
- High dimensionality.

The best way to overcome these difficulties is to do feature selection before actually training the data. So we decided to explore Feature Selection Algorithms and develop an efficient Algorithm both in terms of accuracy and time.

II. Feature Dependency using Similarity Measures

In this method we create two sets A and B. The Set A is initialized to have all the features of the corpus. We take all such possible pairs of features at a time from features in set A and find the correlation between them using a similarity measure. Then from these pairs we select all the pairs having their correlation over a experimentally-determined threshold. Since only one feature will be required out of two highly dependent features to predict outputs, we compare the two features in each pair using FSPP2 measure and put the less important feature of the two into set B. Now we remove the elements of set B from set A to get the final list of relevant features.

Following are some similarity measure we could use to find co-relation between two features

A. Co-relation

Correlation coefficient between two random features X and Y is defined as

$$\rho(X,Y) = \frac{\mathbf{Cov}(X,Y)}{\sqrt{\mathbf{Var}(X)\mathbf{Var}(Y)}}.$$

This is one of the most common similarity measures which tells us how much two features are co-related. If two features are very highly co-related then keeping both of them increases redundancy and hence we can remove one from our final feature set.

Index of Term 1	Index of Term 2	Correlation
34	35	0.722452231
35	1325	0.639071906
254	262	0.678511862
389	1663	0.843005772
448	975	0.616492818
469	470	0.601789962
485	1132	0.712593104
531	651	0.606925163

TABLE I: Pair-wise Correlation between terms

B. Cosine Similarity

Cosine Similarity between two feature vectors A and B is defined as

$$\cos\varphi = \frac{A \cdot B}{|A| \cdot |B|}. \tag{1}$$

This is another similarity measure which is commonly used in Information Retrieval and other Machine Learning fields. It tells us how much to features are co related based on their normalized dot product in a vector space.

C. Jaccard Coefficient

$$Jaccard(A,B) = \frac{n(|A \cap B|)}{n(|A \cup B|)}$$

where, $n(|A \cap B|)$ represents number of documents having both features A and B whereas $n(|A \cup B|)$ represents the number of documents having either feature A or B. This similarity measure tells us how much two features are related by checking the number of common features in all the instances of the data.

III. Choosing the better feature from the correlated pairs

After computing the similarities between two features we need some criteria to choose the better of the two co-related features.

A. Using FSPP2 method to choose the better of the two dependent features

To get a better accuracy, it is better to choose the features wisely rather than choosing them randomly. Many methods can be used to pick the better feature of the pair but this will take greater computation time than randomly choosing one of the two co-related features from the pairs. One such method which we used is discussed below:

1) Using Support Vector Machine: In this method we take the help of Support Vector Machines(SVM) probabilistic outputs to choose the better of the two features. We convert the output of SVM into a sigmoid We use the FSPP2(Feature based Sensitivity of Posterior Probability) measure which uses an approximation of SVM to test which of the two features is better. It is an approximation of the actual method since running complete SVM on every feature would incur high computational time. So we calculate the FSPP2 value for both the features in the pair and choose the one with the higher value i.e. the more important feature of the two. Here N is no of training test, c is the class, x_j is j^{th} data record and $x_{(i),j}$ is j^{th} data record without i^{th} feature.

$$FSPP2(i) = \frac{1}{N}\sum_{j=1}^{N} |\hat{p}\left(\frac{c}{x_j}\right) - \hat{p}\left(\frac{c}{x_{(i),j}}\right)|$$

Where,

$$\hat{p}\left(\frac{c}{x_j}\right) = \frac{1}{1 + exp(Af(x) + B)}$$

IV. Choosing the best set of non co-related Features

A. Using a Probabilistic Neural Network and Bit-wise Gradient Descent

1) Evaluating Feature Subsets using PNN: Consider the problem of multi-class classification. We are given a set of data points from each class. The objective is to classify any new data sample into one of the classes. Probabilistic neural network (PNN) is closely related to Parzen window pdf estimator. A PNN consists of several sub-networks, each of which is a Parzen window pdf estimator for each of the classes. The input nodes are the set of measurements. The second layer consists of the Gaussian functions formed using the given set of data points as centers. The third layer performs an average operation of the outputs from the second layer for each class. The fourth layer performs a vote, selecting the largest value. The associated class label is then determined.

In general, a PNN for M classes is defined as

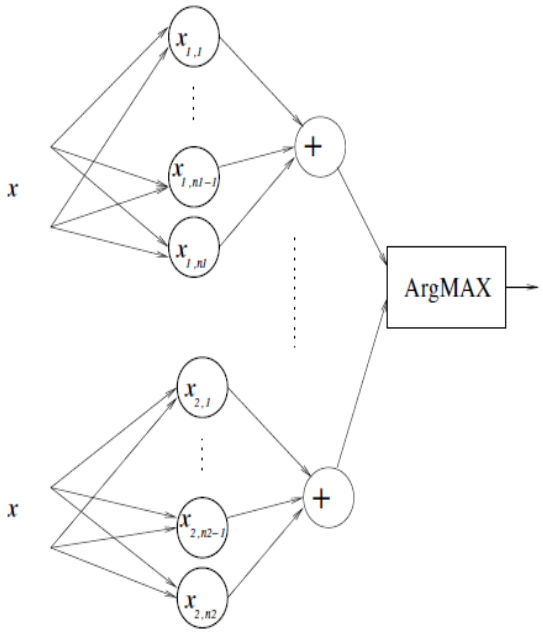
$$y_j(x) = \frac{1}{n_j} \sum_{i=1}^{n_j} exp\Big(- \frac{(||x_{j,i} - x||)^2}{2\sigma^2} \Big)$$

$$j = 1, ..., M$$

where n_j denotes the number of data points in class j. The PNN assigns x into class k if $y_k(x) > y_j(x)$, j ∈ [1, ..., M].

The feature subset is evaluated by calculating the accuracy of the PNN Classifier.

Fig. 1: A schematic illustration of a PNN.



2) Bit-wise Gradient Descent: The PNN can be used, as described in the previous section, to evaluate a variable subset. A variable subset is conveniently represented as a binary string, S, with the number of bits equal to the number of candidate input variables, V; $s_i = 0$ indicates that variable i should not be used; $s_i = 1$ indicates that it should be.

The Bit-wise Gradient Descent algorithm starts with a randomly initialized string. It then flips each bit in the string in turn, retaining the changed bit only if the change causes improvement in accuracy. The total number of evaluations is only V.

3) Selecting features based on frequency: We apply Bit-wise Gradient Descent using PNN for considerable number of times and then calculate the frequencies of all the variables from all the feature subsets. We then select the features having frequency above a substantial threshold.

V. Methodology

A. Removing Dependent Features

1) Finding Dependent Features using Correlation: Pair-wise we find out correlation between all such pairs of features. The process is computationally hard. Various other similarity measures like Jaccard and Normalized Correlation Measure can be used and mean of all such metrics can be taken over to find correlated pairs. Experimentally, a threshold of 0.6 is found to be the best. Finally we put all such pairs in a list. In Classic4 dataset we found 276 such pairs of correlated features, while in Jaccard we found 197 such pairs, hence it was not considered in the algorith.

2) Feature based Sensitivity of Posterior Probabilities(FSPP2): FSPP2 measure can be used compute a score to rank features. We find difference between Platt's probability of SVM output between presence and absence of a feature. From each pair created in previous step, we choose the better one using FSPP2 and remove the other one.

B. Bit-wise Gradient Descent using PNN

Bit-wise Gradient Descent using PNN is applied on remaining features. We choose a random bit sequence of size equal to number of features. At each step, we choose i^{th} bit representing i^{th} feature and find accuracy. We then flip it and again find accuracy. The configuration with better accuracy is retained and similarly the whole bit-wise sequence is parsed. This is repeated considerable number of times and then frequencies of all the features are calculated. We then use experimentally determined threshold of frequency to choose features having their frequency above that threshold.

VI. Experiments and Results

The dataset we used for experimentation was Classic4 dataset. Classic4 dataset is composed of 4 classes i.e. CACM, CISI, CRAN, MED. Each class has 3204, 1460, 1398 and 1033 documents respectively. We used the pre-processed dataset for experimentation. Files were available in term frequency, TF-IDF and normalized TF-IDF format. The terms are single words and Porter's Stemming algorithm was applied to stem

the dataset. After pre-processing, we obtained 7095 documents. and after using FSPP2 measure, we obtained 5896 term features. PNN method was applied on this dataset to obtain different bit sequences. In the end they were summed over and best features were taken after experimentally determining threshold value(K). We then estimated the performance of the reduced dataset after feature selection using different types of classifiers and different corresponding values of threshold. Note that a dataset with K=0 is a dataset without any feature selection and hence is the original dataset. The algorithm works best with K=4 after which all the performance metrics degrade steeply. SVM classifier turned out to be the best classifier for the proposed feature selection method. Multinomial Naive Bayes as well as other methods like Gaussian Naive Bayes and Decision Trees also responded positively with the feature set. We also tested the dataset against K-Nearest Neighbours, however it performed poorly, specially for lower number of features(or higher value of K) as the data became dense and became more prone to outliers. Refer to graphs in figures 2,3,4,5 attached for further details.

Threshold of Feature Frequency	Features Selected
k=0	5896
k=1	5819
k=2	5231
k=3	3880
k=4	1972
k=5	641
k=6	88

TABLE II: No. of features selected based on Frequency Threshold

Fig. 2: Evaluation Metric of SVM with Threshold(K)

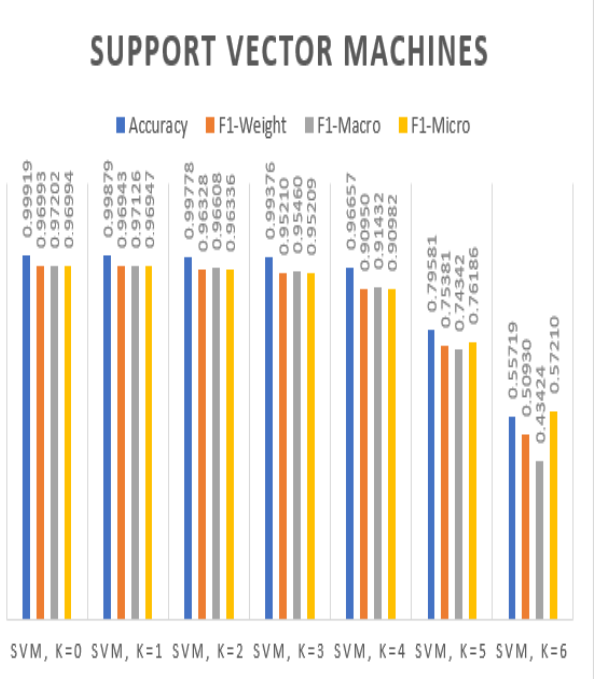


Fig. 3: Evaluation Metric of Decision Trees with Threshold(K)

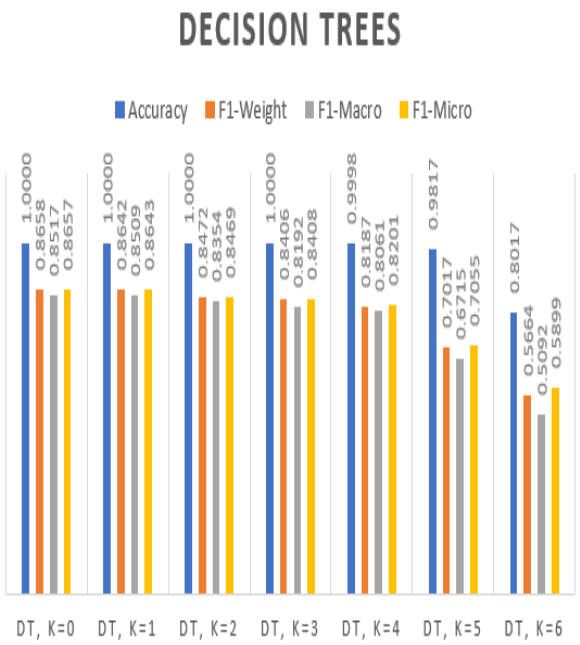


Fig. 4: Evaluation Metric of Gaussian Naive Bayes with Threshold(K)

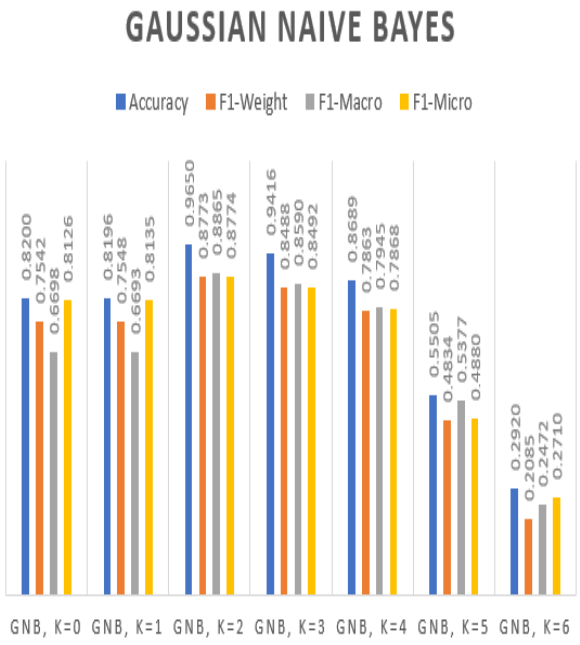
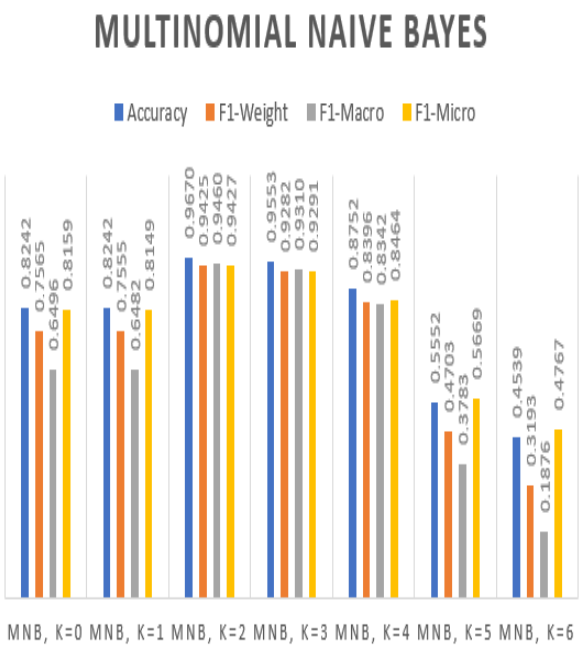


Fig. 5: Evaluation Metric of Multinomial Naive Bayes with Threshold(K)



VII. Conclusion and Future Scope

This paper introduces a new feature selection using correlation to find dependent features. One of the dependent features is chosen using Feature based Sensitivity of Posterior Probabilities of SVM output function, which is a really good option for ranking feature sets. We then use Bit-wise Gradient Descent which uses PNN for finding accuracy at its back-end. Although being computationally expensive, BGD with PNN gives excellent performance specially with SVM as a classifier.

However there is huge scope for improvement in the proposed feature selection method. First of all, more similarity metrics can be used to find dependent features. We have also not taken semantic similarity measures into consideration. Secondly, FSPP2 measure can only be a good ranking criterion but cannot itself be used for future selection. Thirdly, PNN is computationally very expensive and requires significant resources. The algorithm needs to be further optimized to improve its running time to bring into real use.

Acknowledgment

We would like to thank our instructor Dr. Rajendra Roul for giving us the opportunity and providing all the support and guidance for the completion of the project. We would also like to thank Mr. Kushagr Arora, who took interest in our project work and guided us by providing necessary information till the completion of the project.

References

[1] A. Hunter, Feature Selection Using Probabilistic Neural Networks, I. King et al. (Eds.): ICONIP 2006, Part I, LNCS 4232, pp. 782 791, 2006.

[2] Commonality-Rarity Score Computation [A novel Feature Selection Technique using Extended Feature Space of ELM for Text Classification], 2017 ACM. ISBN 123-4567-24-567/08/06.

[3] Kai Quan Shen, Chong Jin Ong, Xiao Ping Li, Hui Zheng, and Einar P.V. Wilder-Smith, Feature Selection Using SVM Probabilistic Outputs, I. King et al. (Eds.): ICONIP 2006, Part I, LNCS 4232, pp. 782 791, 2006.

[4] Kai-Quan Shen, Chong-Jin Ong, Xiao-Ping Li, Einar P.V. Wilder-Smith, Feature selection via sensitivity analysis of SVM probabilistic outputs, Mach Learn, 2008