***Report***

***by Sagiv Melamed 315092239***

***Introduction***

*One major flaw that all learning algorithms suffer from is the curse of dimensionality leading to poor accuracy results and poor execution time. Some features may be less relevant than others and therefore only contribute noise which will lead to performance degradation. A solution for this problem is called feature selection which is to choose a subset of the most contributing features and classify the dataset according to them. In this article we will explore different feature selection methods on various datasets. For the purpose of this article, we chose to inspect the trivial methods and 2 more nontrivial methods for feature selection, those nontrivial methods are*

* *GA-SVM*
* *COMSVM-FRFE*

*First we will explore in detail the 2 nontrivial methods explaining their implementation, advantages and disadvantages. We also added an enhancement the COMSVM-FRFE. Then we will test our algorithms against common feature selection algorithms on various datasets.*

***GA-SVM method***

*Reference*

*A robust hybrid between genetic algorithm and support vector machine for extracting an optimal feature gene subset*

*Short Description*

*The GA-SVM is a combination of two machine learning methods – genetic algorithms and SVM, and therefore considered as a wrapper method. The genetic algorithm goal is to find the best subset of features for decision making and SVM acts as the evaluation method used for the genetic algorithm at each generation evaluation. The full code can be found in this* [*link*](https://github.com/Sagivm/Cancer-Feature-Selection/blob/master/feature_selection/GA_SVM.py)

*Algorithm Explanation*

|  |  |
| --- | --- |
| *Algorithm Parameters* |  |
| *N* | *Number of samples for genetic algorithm* |
| *M* | *Number of features* |
| *GENERATION* | *Number of iterations of the GA* |
| *N\_CROSS* | *Number of crossovers to pull off* |

*The algorithm works as follows*

1. *Scale the data according to the dataset*
2. *Divide the data into train set and validation*
3. *Create a starting population of N vectors with the length of M. Each sample represents a subset of features and the binary value at index m represents the feature’s presence or absence.*
4. *For each feature selection sample, we compute the accuracy using the Linear SVM model which is built on the train set and tested on the and validation set with the mask of selected features.*
5. *For our next generation we create the population as follows*
   1. *We take the best N/2 feature selection of the previous generation out to the next*
   2. *Crossover - we match couples at random and create their children by doing a crossover on the parent samples. A single crossover will generate 2 children. The point index for the crossover is chosen at random.*
   3. *Mutation - A feature selection is chosen to be submitted into mutation at random at each locus*
6. *We return to stage 4 for a couple of iterations*
7. *We take the best feature selection mask out of the last population*

*We have tried multiple runs on the toy sample dataset and came up with the following graph which describes the absence or presence of a feature:*

*Strengths*

1. *The algorithm does not check all possibilities and use random features selection as it main feature selection base.*
2. *Easy to implement*
3. *Linear SVM is faster and simpler classification method than other more complex classification methods like neural networks, and therefore faster and tends to not be overfitted*

*DrawBacks*

1. *The combination of the two algorithms gives us a lot of hyper-parameters to choose.*
2. *The feature’s subset search domain is pretty predetermined by the base generation meaning if the optimal solution is not in the domain, we probably wont find it. Moreover, a feature that is present in the base population will be harder to get rid of.*
3. *This algorithm requires a lot of SVM models which are required for each feature selection sample. We can save the svm models for already tested feature selection but that would take a lot of space.*

***COMSVM-FRFE method***

*Reference*

Feature selection with kernelized multi-class support vector machine

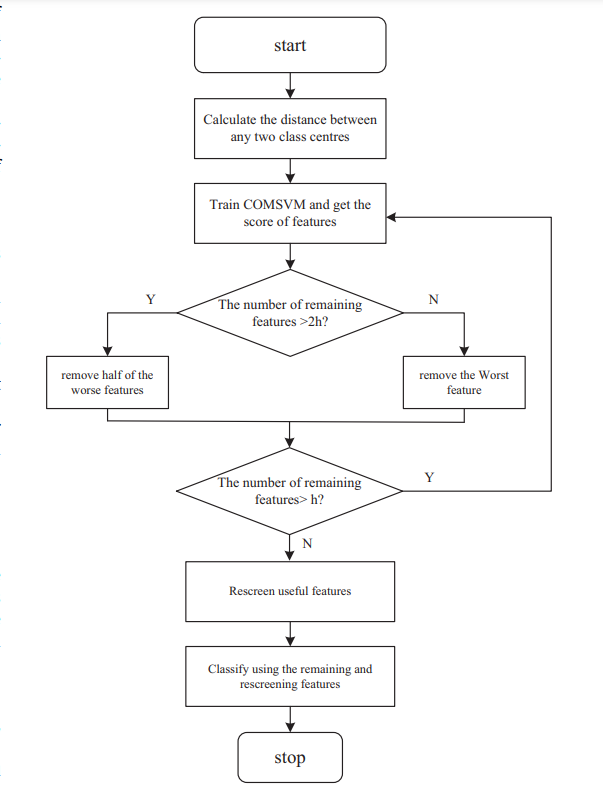
*Short Description*

*This method is an upgraded combination of two methods SVM and RFE. We use class wise optimization methods for classifying and evaluating multiclass dataset using SVM and fast RFE for the feature reduction and selection. The full code can be found in this* [*link*](https://github.com/Sagivm/Cancer-Feature-Selection/blob/master/feature_selection/COMSVM_FRFE.py)

*Algorithm*

*As was said before this algorithm is a combination of two algorithms. COMSVM is used as the class wised feature scorer method and FRFE is the feature elimination method that uses that scorer.*

|  |  |
| --- | --- |
| *Algorithm Parameters* |  |
| *H* | *Number of best features out of the MSVM* |
| *K* | *Features to re-add* |
| *N* | *Number of candidate groups to be readded* |

**

1. *At First we preform the class wise optimization on the data while amplifying the distance between the classes. This is done by calculating the distance between any two classes means.*
2. *Then we will preform the feature selection to get the best H features, this is done by performing the kernelized MSVM in one vs rest fashion thus obtaining K hyperplanes separating each k-th class out of the rest. Then we extract and combine the top features out of all K functions using the SVM coefficients which explains the relationship between the hyperplane and the relevant features of that class. In each iteration we take the top half of the features till we have the top H features.*
3. *Taking out the features in this manner although fast maybe removed important features therefore we create groups of K features which will be tested to be added to the best features thus we will get the H+K best features*

*We have tried multiple runs on the toy sample dataset and came up with the following graph which describes the absence or presence of a feature:*

*Strengths*

* *Extremely fast as it removed the overhead of RFE by reducing the number of features in logarithmic time*
* *Works for multiclass classifications*
* *The number of hyper-parameters is not that high making the algorithm easy to use*

*Drawbacks*

* *Calculating classes mean can give us inaccurate results for some clusters, for example for this cluster both means are close while the classes are not   
  A picture containing background pattern

  Description automatically generated*
* *The algorithm’s success depends too much on the SVM kernel. A bad choice for the SVM kernel will give us bad results and even eliminating important features. Choosing the SVM kernel is not simple in an overcrowded and features heavy environment.*

***Algorithm Enhancement***

*One advantage of the COMSVM-FRFE algorithm was it was considerably fast and very persistent for FS time. As we written before one major flaw of the COMSVM-FRFE algorithm is the dependency on the kernel of the SVM function. At each iteration of the algorithm, we compute one SVM model with a linear kernel to extract the feature importance. This would mean that a kernel function that was chosen to be best fitted for one feature selection will not necessarily be chosen for another feature selection of the same dataset.*

*When learning on the SVM model we found out that the separating hyperplane generated from SVM model is only determined by the support vectors of the SVM model. This would mean the if one support vector is not well placed due to noisy features or even false classification we could end up with a bad hyperplane. Therefore, we wanted to find a way to improve the robustness of the algorithm and the separating hyperplane while making the FS time in the same magnitude.*

*Our solution was as so, we chose to introduce bagging into our COMSVM-FRFE algorithm. At each iteration we would use a bagging SVM model instead of the regular linear SVM model. The bagging algorithm will choosing samples randomly to be part of a particular SVM model. The new features would be extracted out of all of the bagged SVMs for all classifications. The bagged SVMs could run coherently as each kernelized SVM can run on multiple cores and therefore independent.*

*Our name for this method would be class wise multiclass ensemble support vector machine with fast recursive feature elimination – COMESVM-FRFE*

***Evaluation***

*For the evaluation part we compared our 2 algorihtms against well knonwn feature selection algorithms. The algorithms we chose were – mRMR, f\_classif, SelectFdr, SVM-RFE, relief*

*We chose 20 different datasets and various number of features as test subjects for our algorithms, a list of them can be seen below and the full data about the run can be found in this in the attached xlx file*

|  |  |  |  |
| --- | --- | --- | --- |
| *Scikit fs* | *MicroArray* | *Biocon* | *Benchmark* |
| *Basehock* | *Golub* | *ALL* | *Fierer* |
| *Leukemia* | *Khan* | *CLL* | *Morgan* |
| *Lung* | *Sorile* | *ayestCLL* | *WU* |
| *Lung\_small* | *Su* | *DLBCL* | *Gevers\_ileum* |
| *Yale* | *Submarine* | *curatedOvarian* | *Gevers\_rectum* |

*After running all of the mentioned above algorithms for al database we came up with many results. In the table below we will list the best FS algorithm and learning algorithm for each database. For all datasets with the best run configuration, we added an augmented version of the datasets using PCA and* BorderlineSMOTE.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Database* | *Original number of features* | *K* | *FS Algorithm* | *Learning Algorithm* | *AUC* |
| *Basehock* | *4862* | *20* | *Select fdr* | *Naive bayes* | *0.94* |
| *Aug\_Basehock* | *4862* | *20* | *Select fdr* | *Naive bayes* | *0.835* |
| *Leukemia* | *7070* | *5* | *mRMR* | *Random Forest* | *1* |
| *Aug\_Leukemia* | *7070* | *5* | *mRMR* | *Random Forest* | *0.986* |
| *Lung* | *3312* | *5* | *SelectFdr* | *KNN* | *1* |
| *Aug\_Lung* | *3312* | *5* | *SelectFdr* | *KNN* | *0.75* |
| *Lung\_small* | *325* | *5* | *Select-Fdr* | *Random Forest* | *1* |
| *Aug\_Lung\_small* | *325* | *5* | *Select-Fdr* | *Random Forest* | *0.973* |
| *Yale* | *1024* | *10* | *GA-SVM* | *Logistic Regression* | *0.89* |
| *Aug\_Yale* | *1024* | *10* | *GA-SVM* | *Logistic Regression* | *0.81* |
| *Golub* | *7129* | *10* | *RFE* | *SVM* | *1* |
| *Aug\_Golub* | *7129* | *10* | *RFE* | *SVM* | *0.76* |
| *Khan* | *1308* | *5* | *COMSVM-FRFE* | *Random Forest* | *1* |
| *Aug\_Khan* | *1308* | *5* | *COMSVM-FRFE* | *Random Forest* | *0.982* |
| *Sorile* | *456* | *5* | *COMSVM-FRFE* | *Random Forest* | *1* |
| *Aug\_sorile* | *456* | *74* | *COMSVM-FRFE* | *Random Forest* | *0.72* |
| *Su* | *5655* | *5* | *mRMR* | *Random Forest* | *1* |
| *Aug\_Su* | *5655* | *5* | *mRMR* | *Random Forest* | *0.825* |
| *Submarine* | *10100* | *5* | *COMSVM-FRFE* | *SVM* | *0.976* |
| *Aug\_Submarine* | *10100* | *5* | *COMSVM-FRFE* | *SVM* | *0.857* |
| *ALL* | *12625* | *10* | *COMSVM-FRFE* | *KNN* | *0.75* |
| *Aug\_ALL* | *12625* | *10* | *COMSVM-FRFE* | *KNN* | *0.836* |
| *CLL* | *1265* | *5* | *mRMR* | *SVM* | *1* |
| *Aug\_CLL* | *1265* | *5* | *mRMR* | *SVM* | *0.81* |
| *ayestCLL* | *6228* | *5* | *Relief* | *SVM* | *1* |
| *Aug\_ayestCLL* | *6228* | *5* | *Relief* | *SVM* | *0.919* |
| *DLBCL* | *3583* | *20* | *Relief* | *KNN* | *0.76* |
| *Aug\_DLBCL* | *3583* | *20* | *Relief* | *KNN* | *0.304* |
| *curatedOvarian* | *3584* | *20* | *mRMR* | *Naive Bayes* | *0.77* |
| *Aug\_curatedOvarian* | *3584* | *20* | *mRMR* | *Naive Bayes* | *0.876* |
| *Fierer* | *1217* | *5* | *COMESVM-FRFE* | *SVM* | *1* |
| *Aug\_Fierer* | *1217* | *5* | *COMESVM-FRFE* | *SVM* | *1* |
| *Morgan* | *687* | *50* | *mRMR* | *Naive Bayes* | *0.75* |
| *Aug\_Morgan* | *687* | *50* | *mRMR* | *Naive Bayes* | *0.8818* |
| *WU* | *285* | *20* | *COMESVM-FRFE* | *Logistic Regression* | *1* |
| *Aug\_WU* | *285* | *20* | *COMESVM-FRFE* | *Logistic Regression* | *0.52* |
| *Gevers\_ileum* | *99* | *5* | *mRMR* | *SVM* | *0.78* |
| *Aug\_Gevers\_ileum* | *99* | *5* | *mRMR* | *SVM* | *0.714* |
| *Gevers\_rectum* | *108* | *50* | *Comsvm-frfe* | *Naive Bayes* | *0.76* |
| *Aug\_Gevers\_rectum* | *108* | *50* | *Comsvm-frfe* | *Naive Bayes* | *0.631* |

***Insights***

*After running all of the datasets with all of the configurations we can with some conclusions and insights*

*We think that proving which feature selection algorithm or learning algorithm is better cannot be done by only inspecting the AUC parameter. In our research we did not explore the relationship between the running time, number of features preserved the total memory storage and the AUC. All of these parameters should be considered when choosing the best fitted learning algorithm. Moreover when training and evaluating the results we should use a weighted evaluation method as it is more important to us to predict the non common class than the common one.*

*As we said before one thing the COMSVM-FRFE lacks in the dependency on the SVM kernel. In our implementation we used a linear kernel for its easy implementation on taking the weights of each feature. We further expended its usage using bagging in order to nullify some data structure by removing some samples. For most cases the bagging proven to be better or equal to the COMSVM-FRFE. So we conclude that further altering the kernel function and using a variety of different type of kernels can further improve the algorithm’s success.*

*The GA-SVM algorithm proved to be unreliable and mostly uses the randomness as its source of success. While experimenting with the algorithm we saw results that differ between iterations which mainly depended on the sub space presented from the initial feature selection samples. For further research we think that altering the hyperparameters of the algorithm would improve the results.*

*We used multiple methods for augmenting the data and expending it and for most cases those augmentations were not very successful. With further research we found out that the reason for the drop in the accuracy of the model was the PCA usage. PCA is an unsupervised method that does not account for the imbalanced state of the labels which rotates and scale the data to achieve orthogonal features. So assuming one positive and one negative sample and the key features for separating the data have low variance and other dummy features have high variance would mean that after applying the PCA those samples will be harder to distinguish between them and thus increasing the number of false predictions. We think that applying a supervised method instead of the PCA would fit better for imbalanced data.*

***LOO vs LPO***

*In this phase of the article, we will explore the difference between leave one out and leave pair out cross validation*

*In LOO each train instance consists its own fold and in LPO each positive and negative instances pair constitutes its own fold, and the CV estimate is calculated by averaging over all these pairs. When preforming cross validation, we expect that all of our folds will come from the same population, therefore the distribution of the classes should be the same for all folds. To preserve the distribution across all folds we randomly take samples out of the fold. In an unbalanced class distribution with small sample size this attitude can remove a lot of beneficial samples and therefore worsen the performance. For example, using SVM its more likely in a small dataset to remove the support vectors other than in a large dataset.*

*When using LPO the number of folds is as the number of the possible pairs of negative and positive classes leading to not only more folds to be tested but also balancing the importance in the test set of the unbalanced classes. Therefore for LPO CV we would not takeout samples as the distribution is the same across all folds leading to much better performance*