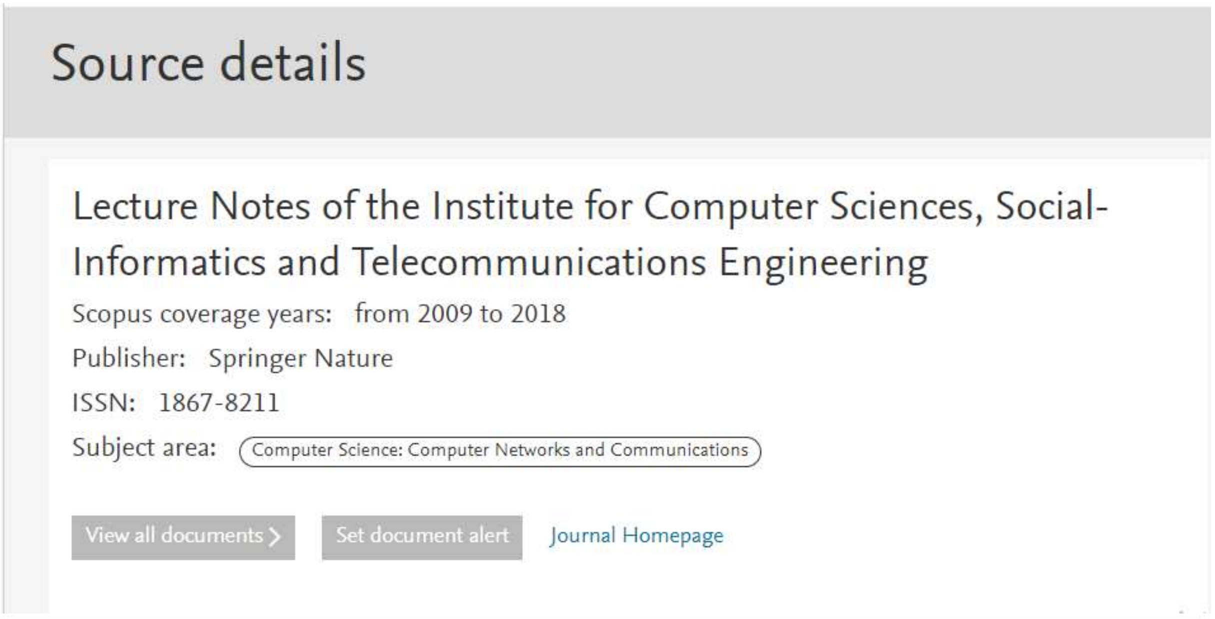
Publication pour la présente candidature LAFPT

Titre	A Parallelized mRMR Method for Feature Selection Based on Spark
Auteurs	Reine Marie Ndéla Marone ; Fodé Camara ; **Samba Ndiaye**
Référence	International Conference on Innovations and Interdisciplinary Solutions for Underserved Areas (InterSol 2018)
Editeur	Springer
Pages	187-198
Année	2018
URL	**https://link.springer.com/chapter/10.1007%2F978-3-319-98878-8_18**
DOI	https://doi.org/10.1007/978-3-319-98878-8_18
Index	https://www.scopus.com/sourceid/21100220348
ISBN	978-3-319-98877-1
Encadreur	Oui
Extrait d'une thèse	Oui



Source details

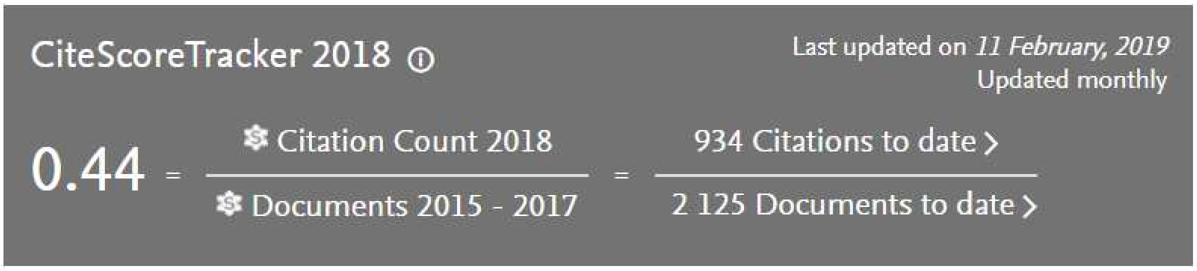
Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering

Scopus coverage years: from 2009 to 2018

Publisher: Springer Nature

ISSN: 1867-8211

Subject area: Computer Science: Computer Networks and Communications



CiteScoreTracker 2018

Last updated on *11 February, 2019*
Updated monthly

$$0.44 = \frac{\text{Citation Count 2018}}{\text{Documents 2015 - 2017}} = \frac{\text{934 Citations to date}}{\text{2 125 Documents to date}}$$

Springer Link

International Conference on Innovations and Interdisciplinary Solutions for Underserved Areas
InterSol 2018: Innovations and Interdisciplinary Solutions for Underserved Areas pp 187-198 | Cite as

A Parallelized Spark Based Version of mRMR

Authors Authors and affiliations

Reine Marie Ndéla Marone, Fodé Camara, Samba Ndiaye

Conference paper
First Online: 30 August 2018

118
Downloads

Part of the Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering book series (LNICST, volume 249)

Abstract

Nowadays, we are surrounded by enormous large-scale high dimensional data called big data and it is crucial to reduce the dimensionality of data for machine learning problems. That's why feature selection plays a vital role in the process of machine learning because it aims to reduce high-dimensionality by removing irrelevant and redundant features from original data. However some characteristics of big data like data velocity, volume and data variety have brought new challenges in the field of feature selection. In fact, most of existing feature selection algorithms were designed for running on a single machine (centralized computing architecture) and do not scale well when dealing with big data. Their efficiency may significantly deteriorate to the point of becoming inapplicable. For this reason, there is an increasing need for scalable yet efficient feature selection methods. That's why we present here a distributed and effective version of the mRMR (Max-Relevance and Min-Redundancy) algorithm to face real-world problems of data mining and evaluate the empirical performance of the proposed algorithms in selecting features in several public datasets. When we compared the efficiency and the scalability of our parallelized method in comparison with the centralized one we have found out that our parallelized method have given better results.

Keywords

Feature selection Filter method Parallel computing Apache Spark mRMR SVM

Copyright information

About this paper

Check for updates

Cite this paper as:
Marone R.M.N., Camara F., Ndiaye S. (2018) A Parallelized Spark Based Version of mRMR. In: Kebe C., Gueye A., Ndiaye A., Garba A. (eds) Innovations and Interdisciplinary Solutions for Underserved Areas. InterSol 2018. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 249. Springer, Cham

First Online 30 August 2018	**DOI** https://doi.org/10.1007/978-3-319-98878-8_18	**Publisher Name** Springer, Cham
Print ISBN 978-3-319-98877-1	**Online ISBN** 978-3-319-98878-8	**eBook Packages** Computer Science

Buy this book on publisher's site

A Parallelized mRMR Method for Feature Selection Based on Spark

Reine Marie Ndéla Marone[1] , Fodé Camara[2], Samba Ndiaye[1]

[1] Department of mathematics, Cheikh Anta Diop University, Dakar, Senegal
[2] Departement of mathematics, Alioune Diop University, Bambey, Senegal
email: fode.camara@uadb.edu.sn, reine.marie.marone@ucad.edu.sn

Abstract. Nowadays, we are surrounded by enormous large-scale high dimensional data called big data and it is crucial to reduce the dimensionality of data for machine learning problems. That's why feature selection plays a vital role in the process of machine learning because it aims to reduce high-dimensionality by removing irrelevant and redundant features from original data. However some characteristics of big data like data velocity, volume and data variety have brought new challenges in the field of feature selection. In fact, most of existing feature selection algorithms were designed for running on a single machine (centralized computing architecture) and do not scale well when dealing with big data. Their efficiency may significantly deteriorate to the point of becoming inapplicable. For this reason, there is an increasing need for scalable yet efficient feature selection methods. In this paper, we propose a parallel scalable feature selection algorithm based on mRMR (Max-Relevance and Min-Redundancy) in Spark: an in-memory parallel computing framework specialized in computation for large distributed datasets. Our experiments using real-world data of high dimensionality show that our distributed method outperforms the centralized feature selection mRMR method and scale well and efficiently with large datasets.

Keywords: feature selection, filter method, parallel computing, apache spark, mRMR, SVM.

1 Problematic and related works

1.1 Introduction

Feature selection is the process of selecting relevant features for uses in model construction and removing those irrelevant and redundant from the dataset [1]. It is widely used in many domains where there are many features and a few samples for example: genes selection, anomaly detection, pattern recognition and many others fields.

Unfortunately, most feature selection algorithms are designed for centralized computing and do not scale well with large-scale datasets, and their efficiency significantly deteriorates or even becomes inapplicable [3]. Distributed computing techniques such as MapReduce [3] along with its open-source implementation Apache Hadoop can help alleviate this problem; effectively allowing users to work with Big Data [3].

But the MapReduce parallel programming with Apache Hadoop is not suited for the feature selection because it causes very high I/O overhead for iterative computations [4]. More recently, Apache Spark [4] has been presented as an alternative to Hadoop and improves the IO read /write performance issue by processing intermediate data in-memory [4].

In regard to that, in this paper, we propose a parallel version of the centralized mRMR algorithm that we have named SFS-mRMR (for Spark Feature Selection method based on mRMR), on the open-source parallel processing framework Spark to improve its performance. The choice of mRMR is motivated by the fact that minimum-redundancy-maximum-relevance (mRMR) selector is considered one of the most relevant method for dimensionality reduction due to its high accuracy.

The results that we obtained show that our algorithm is scalable and outperforms the classical mRMR feature selection method.

The rest of the paper is structured as follows:

Section 2 reviews previous works. Section 3 deals with the formulation of the problem. Section 4 presents the centralized mRMR. Section 5 gives the metrics we used in our proposal. Section 6 consists of the presentation of our algorithm. Section 7 describes the working environment. Section 8 evaluates the performance of our algorithm. Section 9 is the conclusion of this paper.

1.2 Related works

Feature selection is a fundamental preprocessing step to reduce input dimensionality.

In general feature selection methods can be classified into 3 major categories: Filter, Wrapper and embedded [5].

In the wrapper methods the “usefulness” of a subset of features is evaluated on the basis of the classifier performance [5].

Embedded methods exploit intrinsic characteristics of a given model to guide the feature selection process, and choose features which best contribute to the accuracy performance of the model [5].

In Filter methods, features are selected on the basis of characteristics, which determine their relevance or discriminant powers with the outcome variable [5, 6]. Filter methods offer better computational complexity but do not take into account the interactions among the variables, which cannot be ignored. Many filter methods are based on information theory- specifically mutual Information. That’s the case with mRMR. But mRMR is a centralized method and do not scale well with ultrahigh dimensional datasets. So optimizing its implementation through efficient parallelization is crucial today [7]. That’s why, proposals have been made on the parallelization of mRMR algorithm the interest of which is to decrease the training time and improve modeling task (prediction, recognition, classification).

In [1] authors parallelize a large set of well-known information theory-based methods including mRMR in Apache Spark.

Experimental results for a large number of real-world datasets have led to competitive performance (in terms of generalization and efficiency) in case of ultra-high-dimensional datasets.

The work in [8] proposes to extend mRMR by using a number of approaches to better explore the feature space and build more robust predictors. To deal with the computational complexity of those approaches, authors implement and parallelize functions in C using the openMP Application Programming Interface. These methods show significant gains in terms of run-time.

Authors in [9] present a two-stage selection algorithm by combining ReliefF and mRMR. In the first stage, ReliefF is applied to find a candidate gene set; In the second stage, mRMR method is applied to directly and explicitly reduce redundancy for selecting a compact yet effective gene subset from the candidate set. The experimental results show that the mRMR-ReliefF gene selection algorithm is very effective.

In [10], authors present three implementations of an extension of mRMR named fast-mRMR in several platforms, namely, CPU for sequential execution, GPU (graphics processing units) for parallel computing, and Apache Spark for distributed computing using big data technologies.

In [11], authors combined dynamic sample space with mRMR and proposed a new feature selection method. In each iteration, the weighted mRMR values are calculated on dynamic sample space consisting of the current unlabelled samples. The feature with the largest weighted mRMR value among those that can improve the classification performance is selected in preference. Five public datasets were used to demonstrate the superiority of this method.

It is clear that the methods presented in these different works use the complex iterative computations because many of them include iteratively one or many features into a subset of features. In what we propose, a subset of relevant and non-redundant features can be selected in only one single pass. That allows a more significant reduction of the learning time while keeping good classification accuracy.

1.3 Problem definition

We focus on two-class classification issues, the target class label $l \in \{0, 1\}$. F is the given feature set $\{f_1,..,f_n\}$. An instance V is represented by a m-dimensional vector $(v_1,..,v_n)$, where v_j is the value of the feature f_j in V. Let $O(S, T)$ be the objective function which evaluates the subset S of F using the data T. The subset S_1 is better than S_2 if $O(S_1, T) > O(S_2, T)$.

In this paper, we assume n so large as in the big data context, and we proposed a large-scale filter method: SFS-mRMR for Spark Feature Selection method based on mRMR (Minimum Redundancy and Maximum Relevancy). We used the well-known parallel computing framework, Apache Spark^{TM} to implement the algorithm.

2 Improvement of mRMR

2.1 The classical mRMR

mRMR means Minimum Redundancy and Maximum Relevance. The concept of mRMR is to select the features so that they are mutually maximally dissimilar and maximally relevant with the class label l [12]. Let f_i and f_j be two variables in F. $MI(f_i, f_j)$ represents the measure of mutual information between the variables f_i and f_j. $MI(l, f_i)$ stands for the measure of mutual information between the class label l and f_i.

The redundancy among the features in *F,* which is determined by the mutual information among them, is given by

$$Q_I(F) = \frac{1}{|F|^2} \sum_{f_i, f_j \in F} MI(f_i, f_j) \qquad (1)$$

The relevance of the features in *F* with the class label *l* is computed as

$$R_I(F) = \frac{1}{|F|} \sum_{f_i \in F} MI(l, f_i) \qquad (2)$$

The maximally relevant and minimally redundant set of feature F^* among all sets F' in *F* is obtained by optimizing (1) and (2) as follows:

$$F^* = \text{argmax}_{F' \subseteq F} [R_I(F) - Q_I(F)] \qquad (3)$$

2.2 Our proposal

Many authors worked on ranking SVM using weights from linear SVM (support vector machines) and obtained good performances [13]. That's why, in our method SFS-mRMR, we use the metric given by the authors in [14] which combines linear support vector machines and the mRMR criterion to rank the features for better results. Let β ∈ [0, 1] determines the tradeoff between SVM ranking and mRMR ranking. The relevancy $R_{F,i}$ of feature f_i in the *F* set in classification is given by

$$R_{F,i} = \frac{1}{|F|} \sum_{l} MI(l, f_i) \qquad (4)$$

And $Q_{F,i}$ the redundancy of feature f_i in the set *F* in classification is given by

$$Q_{F,i} = \frac{1}{|F|^2} \sum_{f_i, f_j \in F} MI(f_i, f_j) \qquad (5)$$

Let ω_i denotes the SVM weight of the feature f_i.

For *i*-th feature, the ranking measure d_i is given by

$$d_i = \beta|\omega_i| + (1-\beta)\frac{R_{F,i}}{Q_{F,i}} \quad (6)$$

3 Our algorithm

Our proposed algorithm, called SFS_mRMR*,* is a feature selection method based on Spark, a parallel programming framework. Let *S* refer to the input dataset (with *n* features and *m* instances) and *K* the number of features to return. Let β be the tradeoff between SVM ranking and mRMR ranking**,** and *x* number of partitions for the dataset**.** *F* denotes the features space. The output *S'* will be the optimal subset of *K* features with max d_i score.

Our algorithm follows seven steps:

- **Step 1: create x partitions of features**

1. Construct *labels*={l_1,..,l_m} the set of the class label in each instance of the dataset.

2. Construct *values*={{v_i^1,..,v_i^m}, i=1 to n }

values represents the set of values v_i^j for each feature f_i in each instance I_j:

3. Construct *x* subspaces of features SF_t, *t = 1..x* from the entire feature space *F.*

4. Construct x subspaces sub_t of {{v_i^1,..,v_i^m}, $i \subset SF$}

5. Each sub_t will be sent to a unique worker (among the *x* workers).

- **Step 2: associate features by two with labels**

On each worker t:

6. Create several sets for each feature f_i in sub_t by mapping f_i with each other feature f_j in F as follows:

f_i=>{ f_i, {v_i^1,..,v_i^m}, {v_j^1,..,v_j^m} , {l_1,..,l_m}}

We call the set {f_i, {v_i^1,..,v_i^m}, {v_j^1,..,v_j^m}, {l_1,..,l_m}, i=1 to n, j=1 to m } obtained *rdd2.*

- **Step 3: calculate the mutual information between features and the relevance of each feature**

In this step, we use each element of rdd2 to calculate mutual information M_{ij} between each feature f_i and another feature f_j of F. We compute also the relevance R_i (mutual information with his class label) of f_i. We proceed as follows:

Foreach element *e* $\subset$ *rdd2*

```
7. rdd [($f_i$, $M_{ij}$, $R_i$)] = mapToPair (e=>{ $f_i$, $M_{ij}$, $R_i$ })
      $M_{ij}$ = MutualInformation ({ $v_i^1,\ldots,v_i^m$ }, { $v_j^1,\ldots,v_j^m$ })
      $R_i$= MutualInformation ({ $v_i^1,\ldots,v_i^m$ }, { $l_1,\ldots,l_m$ }) /n
where $l_{k\in 1..m}$ represents the label of class in instance $I_k$..
EndForeach
```

The constituted set of each feature f_i, its mutual information with another feature f_j and its mutual information R_i with the class label will be called rdd3.

- **Step 4 : aggregate the mutual information for each feature by summing them.**

In order to obtain the redundancy of each feature f_i, sum its mutual information with the other features of F and keep the mutual information with the class label (for the relevance). A new set is then obtained and we call it *rdd4*. Each element in rdd4 consists of {f_i, $sumM_{ij}$, R_i}, where f_i is the feature, $sumM_{ij}$ is the sum of mutual information between f_i and the other features of the space of features F and R_i the mutual information between f_i and the class label.

This corresponds to the following instructions:

```
Foreach element ($f_i$, $M_{ij}$, $R_i$ ) $\in$ rdd3
   8. rdd [($f_i$, $sumM_{ij}$, $R_i$ )]= reduceByKey (_+_)
         $sumM_{ij} = \sum_{i=1}^{n} M_{ij}$
EndForeach
```

- **Step 5: for each feature, compute the SVM weight**

In this step compute for each feature f_i its SVM weight ω_i as follows:

```
Foreach $f_i \in F$
9.  rdd [($f_i$, $\omega_i$ )]= map ($f_i$ =>{ $f_i$, $\omega_i$ })
         $\omega_i \in \omega$ where $\omega$ =SVMWeight(F)
EndForeach
```

- **Step 6: for each feature, compute the rapport d_i between relevance and redundancy**

In this step calculate for each feature f_i the ranking measure d_i that represents a tradeoff between the redundancy and the relevance of f_i. Then send all d_i values to the master.

This is done as follows:

```
Foreach element ($f_i$, $sumM_{ij}$, $R_i$) ∶ rdd4
 10.rdd [($f_i$,$d_i$ )]= mapToPair ({$f_i$, $sumM_{ij}$, $R_i$ } =>{$f_i$, $d_i$ })
    $Q_i$= $sumM_{ij}$/(n*n);
    $d_i$ =β + $\omega_i$+((1-β)* ($R_i$ /$Q_i$));
/* $\omega_i$ is the SVM weight of attribute $f_i$*/
EndForeach
 11.All workers send $d_i$ to the master
```

- **Step 7: Choose the best features in F**

Finally, master collects, orders and returns the *K* features with the best scores d_i. This corresponds to the following sentences:

```
On the master:
 12.Collect and take ordered
 13.Return S': optimal subset of K features in S with
    highest scores $d_i$.
```

4 Experimental setup and results

4.1 Data Description

We used support vector machine as classifier and LibSVM as the support vector machine tool.

We used two benchmark real-world datasets chosen from mldata.org [15]. Some informations of those datasets are presented in Table 1.

TABLE 1. CHARACTERISTICS OF BENCHMARK DATASETS

NAME	NUMBER OF FEATURES	NUMBER OF INSTANCES
COLON-CANCER	2000	62
COLON-TUMOR	2000	60

The experiments were performed on a cluster consisting of 4 nodes, where each node has 8 cores running at 2.60 GHz, with 56 GB memory and a 382 GB disk, then on a cluster of 6nodes with the sames parameters. The computing nodes are all running at the linux.

4.2 Experimental Results

In this part, we will first discuss the scalability of our solution then we will compare the execution time of our proposal with the one of centralized mRMR.

Figures 1, 2 and 3 show respectively how the execution time varies according to the number of nodes when we select 25%, 50% or 75% of the dataset.

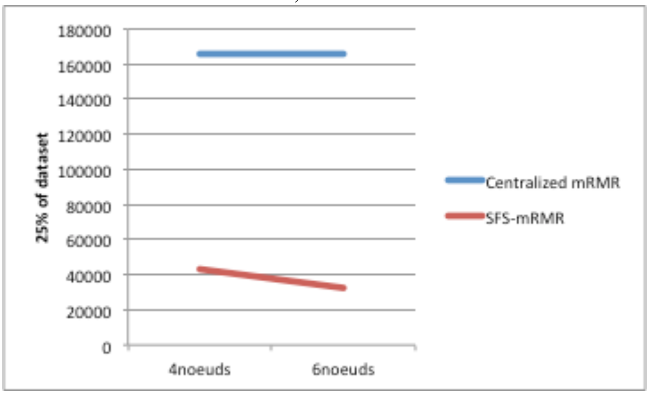


Figure 1. Scalability of SFS_mRMR and classical mRMR with 25%.

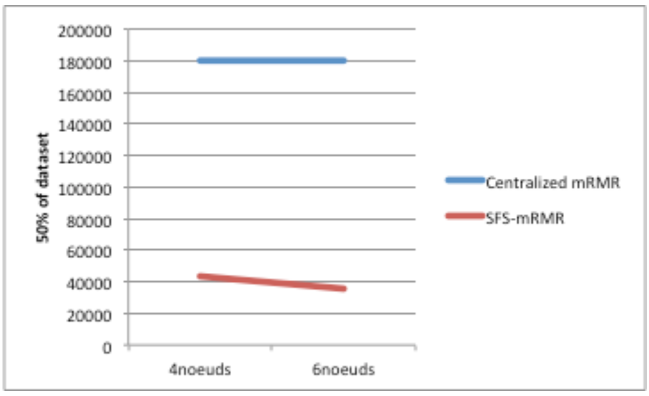


Figure 2. Scalability of SFS_mRMR and classical mRMR with 50%.

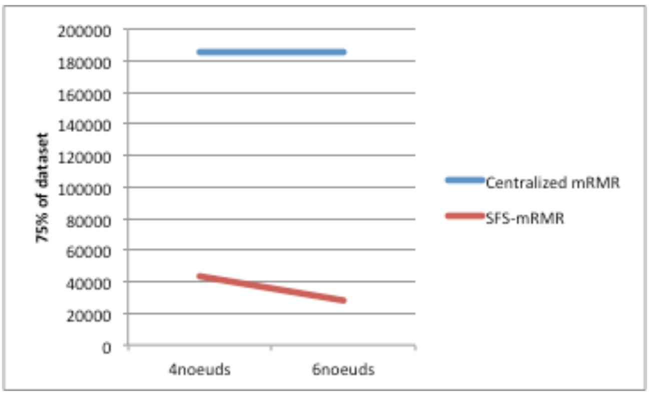


Figure 3. Scalability of SFS_mRMR and classical mRMR with 75%.

The concerned figures clearly show that the execution time of our proposal considerably decreases when the number of nodes increases whereas the time taken by classical mRMR remains constant.

We have used 4 then 6nodes for the scalability. And for every case we have run the tests using the same environment.

For every dataset we first select 25% then 50% and after 75% of features.

- ✓ **Colon-cancer**

Figure 4 and figure 5 shows the time taken by our method comparatively to the one of centralized mRMR for respectively 4 and 6nodes.

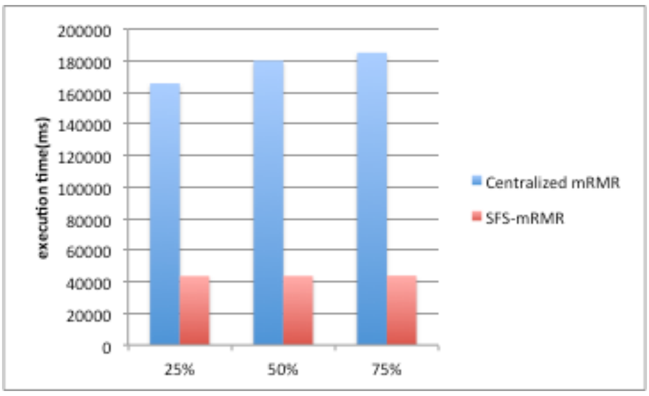


Figure 4. Time taken for colon-cancer with 4nodes

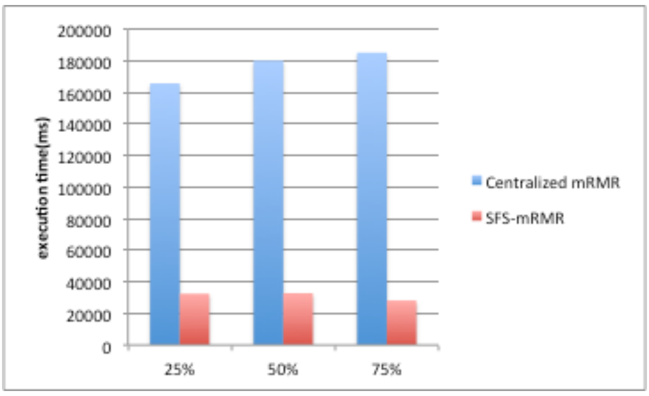


Figure 5. Time taken for colon-cancer with 6nodes

As we can notice, the execution time of our method is at least 4 times shorter compared to the one of centralized mRMR.

✓ **Colon-Tumor**

For colon-tumor the results obtained with 4nodes are the following ones given in figure 6:

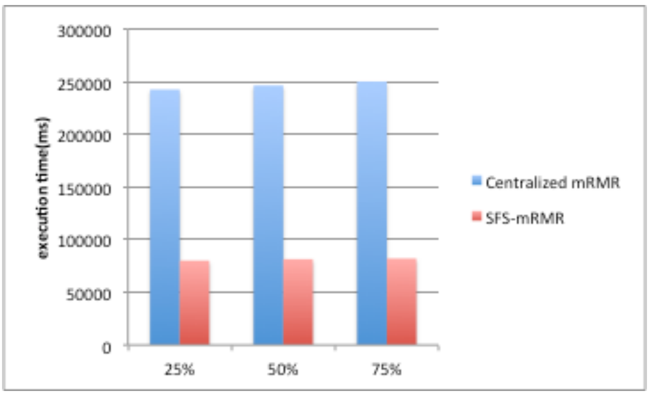


Figure 6. Time taken for colon-tumor with 4nodes

With a cluster of 6nodes the execution time is stated in figure 7.

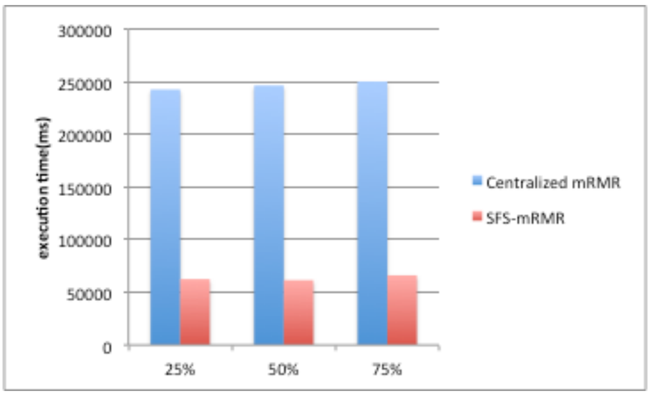


Figure 7. Time taken for colon-tumor with 6nodes

As for the colon-cancer we can notice that the execution time of our method SFS-mRMR is also 4times shorter at least.

Therefore, we can conclude from these experiments that our solution outperforms the centralized mRMR method in terms of execution time. Moreover, the more we increase the number of nodes, the shorter the execution time becomes in our method whereas the one of centralized mRMR remains constant.

In our experiments we have used datasets limited to 2000 features, because beyond that number, the centralized mRMR takes too much time to run. For example, for certain datasets above 2000 mRMR can take days to run completely.

5 Conclusion

In this paper, we have proposed a parallel and scalable version of a centralized feature selection method named mRMR. Our proposal was developed on Spark an in-memory parallel computing framework specialized in computation for large distributed datasets.

In our method, a score is given to each feature to evaluate its redundancy with the others features of the dataset and its relevance relatively to the class label. Then the features presenting the highest score are returned.

Experimental results demonstrates that our algorithm achieves a great performance improvement in scaling well and reduces the time taken by mRMR for selecting relevant and non redundant features.

In the future, we plan to parallelize many other *centralized feature selection methods* like relief or rfe-svm.

ACKNOWLEDGMENT

In this work, Microsoft Azure has sponsored us and we would like to take this occasion to express them our thanks.

Without their help we would not have been able to test our algorithms in a cluster and we would not have reached our goals.

References

1. Sergio Ramırez-Gallego, Hector Mourino-Talın, David Martınez-Rego,Veronica Bolon-Canedo, Jose Manuel Benıtez, Amparo Alonso-Betanzos and Francisco Herrera. An Information Theory-Based Feature Selection Framework for Big Data under Apache Spark.Journal of latex class files, vol. 13, no. 9, september 2014.
2. Vaishali Chahar, Rita Chhikara, Yogita Gigras and Latika Singh. Significance of Hybrid Feature Selection Technique for Intrusion Detection Systems.Indian Journal of Science and Technology, Vol 9(48), DOI: 10.17485/ijst/2016/v9i48/105827, December 2016.
3. Zhao Z., Cox J., Duling D., Sarle W. (2012) Massively Parallel Feature Selection: An Approach Based on Variance Preservation. In: Flach P.A., De Bie T., Cristianini N. (eds) Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2012. Lecture Notes in Computer Science, vol 7523. Springer, Berlin, Heidelberg.
4. Dilpreet Singh and Chandan K Reddy. A survey on platforms for big data analytics. J Big Data. 2015; 2(1): 8.Published online 2014 Oct 9.
5. Chuan Liu, Wenyong Wang, Qiang Zhao andMartin Konan. A new feature selection method based on a validity index of feature subset.Pattern Recognition Letters,Volume 92, 1 June 2017, Pages 1-8.
6. Wenyan Z, Xuewen L , Jingjing W. Feature Selection for Cancer Classi cation Using Microarray Gene Expression Data. Biostat Biometrics Open Acc J. 2017;1(2): 555557.
7. Jaseena K.U. and Julie M. David.Issues, challenges, and solutions: big data mining.Sixth International Conference on Networks & Communications,DOI: 10.5121/csit.2014.41311.
8. De Jay, Nicolas and Papillon, Simon and Olsen, Catharina and El-Hachem, Nehme and Bontempi, Gianluca and Haibe-Kains, Benjamin. MRMRe: An R package for parallelized mRMR ensemble feature selection. Bioinformatics (Oxford, England), July 2013, http://dx.doi.org/10.1093/bioinformatics/btt383.
9. Yi Zhang, Chris Ding and Tao Li. Gene selection algorithm by combining reliefF and mRMR. BMC Genomics. 2008; 9(Suppl 2): S27, doi: 10.1186/1471-2164-9-S2-S27.
10. Ramírez-Gallego, Sergio and Lastra, I and Martinez, David and Bolón-Canedo, Verónica and Benítez, José and Herrera, Francisco and Alonso-Betanzos, Amparo. Fast-mRMR: Fast Minimum Redundancy Maximum Relevance Algorithm for High-Dimensional Big Data: FAST-mRMR ALGORITHM FOR BIG DATA. International Journal of Intelligent Systems, July 2016, DOI: 10.1002/int.21833.
11. Yuansheng Yang, Haiyan Li, Xiaohui Lin and Di Ming. Recursive Feature Selection Based on Minimum Redundancy Maximum Relevancy. Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on, DOI:10.1109/PAAP.2010.52.
12. Monalisa Mandal, Anirban Mukhopadhyay. An Improved Minimum Redundancy Maximum Relevance Approach for Feature Selection in Gene Expression Data. Jul 2016 · IEEE/ACM Transactions on Computational Biology and Bioinformatics.
13. Yin-Wen Chang and Chih-Jen Lin. Feature ranking using linear SVM.Proceedings of the Workshop on the Causation and Prediction Challenge at WCCI 2008, PMLR 3:53-64, 2008.
14. Piyushkumar A. Mundra and Jagath C. Rajapakse.SVM-RFE With MRMR Filter for Gene Selection.IEEE transactions on nanobioscience, vol. 9, no. 1, march 2010.
15. http://mldata.org/repository/data/viewslug/ovarian-cancer-nci-pbsii-data/.