Parallel feature selection based on the trace ratio criterion

Thu Nguyen*1, Thanh Nhan Phan*3,4,5, Nhuong Nguyen2, Thanh Binh Nguyen3,4,5, Pål Halvorsen1, and Michael Riegler1

¹SimulaMet, Norway
²University of Connecticut, USA
³AISIA Research Lab, Vietnam
⁴University of Science, Vietnam
⁵Vietnam National University, Vietnam

Abstract—The growth of data today poses a challenge in management and inference. While feature extraction methods are capable of reducing the size of the data for inference, they do not help in minimizing the cost of data storage. On the other hand, feature selection helps to remove the redundant features and therefore is helpful not only in inference but also in reducing management costs. This work presents a novel parallel feature selection approach for classification, namely Parallel Feature Selection using Trace criterion (PFST), which scales up to very large datasets. Our method uses trace criterion, a measure of class separability used in Fisher's Discriminant Analysis, to evaluate feature usefulness. We analyzed the criterion's desirable properties theoretically. Based on the criterion, PFST rapidly finds important features out of a set of features for big datasets by first making a forward selection with early removal of seemingly redundant features parallelly. After the most important features are included in the model, we check back their contribution for possible interaction that may improve the fit. Lastly, we make a backward selection to check back possible redundant added by the forward steps. We evaluate our methods via various experiments using Linear Discriminant Analysis as the classifier on selected features. The experiments show that our method can produce a small set of features in a fraction of the amount of time by the other methods under comparison. In addition, the classifier trained on the features selected by PFST not only achieves better accuracy than the ones chosen by other approaches, but can also achieve better accuracy than the classification on all available

Keywords—feature selection, classification, trace ratio

I. INTRODUCTION

In this era of big data, the growth of data poses challenges for effective data management and inference. For example, a gene dataset can contain hundred thousands of features [1]. Hence, directly handling such datasets may face the curse of dimensionality. Moreover, the presence of redundant features can derail the learning performance of classification algorithms. For dealing with this issue, many dimension reduction techniques have been developed [2]–[6], and they are categorized into either feature extraction or feature selection methods [7], [8].

Feature extraction techniques (e.g., Principal Component Analysis [9], Linear Discriminant Analysis [9]) involve pro-

jecting the data into a new feature space with lower dimensionality via some linear or nonlinear transformation of the original features. However, this creates sets of new features that can not be directly interpreted. Moreover, since those approaches use all the features available during feature extraction, it does not help reduce the cost of data storage and the cost of collecting data in the future.

On the other hand, feature selection methods (forward selection [10], backward selection [10], etc.) select only a subset of useful features for model construction. Therefore, it maintains the original features' meanings while reducing the cost of storage and collecting data in the future by removing irrelevant features.

However, data from various fields, such as text mining, business analytics, and biology, are often measured in gigabytes or terabytes with millions of features [11], [12]. For example, the Amazon Review dataset [13] is a 34 gigabytes dataset. In such cases, the performance of the most current feature selection techniques may be jeopardized [12]. This is due to the search space for a set of relevant features increases significantly. One way to deal with this issue is to go for parallelization, which allows better use of computers' computational resources by data partitioning and running features selection on multiple cores at the same time.

In this work, we develop a novel parallel feature selection method for classification that can help remove redundant features and improve the model's performance without sacrificing the running time for feature selection. This is achieved by using the trace criterion in Linear Discriminant Analysis [9] as the evaluation criteria for the measure of separability between classes and therefore help evaluate if adding a feature helps to improve the model. Using this criterion, we propose a parallel feature selection approach, namely Parallel Feature Selection based on Trace criterion (PFST), which consists of three stages. First, in a forward-dropping stage, we do parallel forward selection on multiple workers and discard seemingly redundant features from subsequent steps of the stages. At the second stage, after the most important features are already included in the model, we check back their contribution after including many essential features for possible interaction that

^{*} denotes equal contribution

may improve the fit. Finally, we conduct backward selection in the last stage to check back possible irrelevant features added by the forward moves.

The rest of the article is organized as follows: First, Section II gives a review of related works in feature selection. Second, Section III reviews some basic approaches in feature selection, and Section IV details the trace criterion and some of its desirable properties as a measure of class separability for feature selection. Next, Section V describes our methodology, and Section VI illustrates the power of our method via experiments on various datasets. Lastly, we summarize our approach and discuss future works in Section VII.

II. RELATED WORKS

Due to the benefits of maintaining original features while reducing the storage cost of feature selection, there have been many efforts in the field of feature selection to develop new techniques for the growing data size challenges. Aside from hybrid approaches that combine different feature selection strategies [14], [15], most feature selection methods can be classified into three categories.

First, the wrapper approaches rely on the performance of a specified learning algorithm to evaluate the importance of selected features. A typical wrapper method will first search for a subset of features for a given learning algorithm and then evaluate them. These steps are repeated until some stopping criteria are satisfied. Methods in this category are usually computational expensive as subset evaluation requires multiple iterations. Even though many search strategies, such as best-first search [16] and the genetic algorithm [17], have been proposed, using these strategies for high dimensional data is still computational inefficient.

On the other hand, the filter approaches consist of techniques that evaluate feature subsets via ranking with some criteria such as information criteria [18], [19], reconstruction ability [20], [21]. These approaches choose features independently from learning algorithms and are typically more computationally efficient than wrapper methods [12]. However, since the are not optimized for any target learning algorithm, they may not be optimal for a specific learning algorithm.

Meanwhile, the embedded approaches use independent criteria to find optimal subsets for a known cardinality. After that, a learning algorithm is used to select the final optimal subset among the optimal ones across different cardinalities. Therefore, they are more computationally efficient than wrapper methods since they do not evaluate feature subsets iteratively. In addition, they are also guided by the learning algorithm. Therefore, they can be seen as a trade-off between the filter strategy and wrapper strategy [12].

Despite efforts in the field of feature selection so far, data from various fields may be too abundant even for filter methods to be computationally efficient. This motivates many types of research in parallel feature selection. For example, the methods proposed in [2]–[6] use parallel processing to evaluate multiple features or feature subsets simultaneously, and therefore speed up the feature selection process. Yet, these

algorithms require access to the whole data. On the other hand, in [22], the authors propose a parallel feature selection algorithm for logistic regression based on the MapReduce framework, and features are evaluated via the objective function of the logistic regression model. Meanwhile, the authors in [23] propose a Parallel, Forward-Backward with Pruning algorithm (PFBP) for feature selection by Early Dropping [23] some features from consideration in subsequent iterations, Early Stopping [23] of consideration of features within the same iteration, and Early Return of the winner in each iteration. However, this approach requires bootstrap computations of p-value, which is computationally expensive. Zhao et al. [24] introduce a parallel feature selection algorithm that selects features based on their abilities to explain the variance of the data. Yet, in their approach, determining the number of features in the model is based on transforming the categorical labels to numerical values and using the sum of squared errors. Getting the sum of squared errors requires fitting the model and, therefore, the algorithm is still computationally expensive.

III. PRELIMINARIES

Various feature selection algorithms have been developed up to now. However, forward, backward, and stepwise feature selection remains widely used. This section summarizes these three techniques, and the details are given in Algorithms 1, 2, and 3.

Forward feature selection (Algorithm 1) has been used widely due to its computational efficiency, along with the ability to deal with problems where the number of features highly exceeds the number of observations efficiently. However, some features included by forwarding steps may appear redundant after including some other features. About the sufficient conditions for forward feature selection to recover the original model and its stability, we refer to [25] and [26] for further readings.

Algorithm 1. Forward Feature Selection

Input: A dataset of p features $f_1, f_2, ..., f_p$, threshold α . **Output:** A set R of relevant features.

```
2: S \leftarrow \{f_1, f_2, \dots, f_p\}

3: while True do

4: f_j \leftarrow the most useful feature in S

5: if the model improves more than an amount of \alpha after including f_j then

6: R \leftarrow R \cup \{f_j\}

7: S \leftarrow S \setminus \{f_j\}

8: else

9: return R

10: end if

11: end while
```

1: $R \leftarrow \emptyset$

Backward feature selection [10], as shown in Algorithm 2, starts with the full model containing all the predictors and then iteratively removes the least useful one, one-at-a-time. It ensures that only redundant features are removed from the

model. However, backward feature selection is slow compared to forwarding selection.

Input: A dataset of p features $f_1, f_2, ..., f_p$, threshold β .

Algorithm 2. Backward Feature Selection

Output: A set R of relevant features.

1: $R \leftarrow \{f_1, f_2, \dots, f_p\}$ 2: while True, do

3: $f_j \leftarrow$ the least useful feature in R

4: if the amount of loss by excluding f_j is less than β then

5: $R \leftarrow R \setminus \{f_j\}$ 6: else

7: return R

8: end if

9: end while

As another alternative, a hybrid version of both forward and backward feature selection is stepwise selection, detailed in Algorithm 3. In this approach, the features are added to the model sequentially as in the forwarding feature selection. However, after adding each new feature, the method may remove any feature that no longer seems applicable.

Algorithm 3. Stepwise Feature Selection

Input: A dataset of p features $f_1, f_2, ..., f_p$, forward threshold α , backward threshold β .

Output: A set R of relevant features.

```
1: R \leftarrow \emptyset
2: S \leftarrow \{f_1, f_2, \dots, f_p\}
         f_i \leftarrow the most useful feature in S
         if the model improves more than an amount of \alpha after
    including f_i then
              R \leftarrow R \cup \{f_j\}
6:
              S \leftarrow S \setminus \{f_i\}
7:
              while True do
8:
                  f_k \leftarrow the least useful feature in R
9:
                  if the model performance decreases with an
10:
    amount less than \beta after excluding f_k then
                       R \leftarrow R \setminus \{f_k\}
11:
12:
                  else break
                  end if
13:
              end while
14:
         else
15:
              return R
16:
         end if
17:
```

IV. TRACE CRITERION AS A FEATURE SELECTION CRITERION

18: end while

Trace criterion is a useful class separability measure for feature selection in classification tasks (more details in [9], [27]). There are many equivalent versions. However, suppose that we have C classes, and there are n_i observation for the

 i^{th} class, and let \mathbf{x}_{ij} be the j^{th} sample of the i^{th} class. Then, one way to define the criterion is

$$trace(S_w^{-1}S_b), (1)$$

where

$$S_b = \sum_{i=1}^{C} n_i (\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}) (\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})'$$
 (2)

and

$$S_w = \sum_{i=1}^{C} \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i)(x_{ij} - \bar{x}_i)'$$
 (3)

are the between-class scatter matrix and within-class scatter matrix, respectively. Here, A' is the transpose of a matrix A, \bar{x}_i is the class mean of the i^{th} class, and \bar{x} is the overall mean, i.e.,

$$\bar{x}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} (\bar{x}_{ij} - \bar{x}_i)$$

$$\bar{x} = \frac{1}{\sum_{i=1}^{C} n_i} \sum_{i=1}^{C} \sum_{j=1}^{n_i} (\bar{x}_{ij} - \bar{x}_i).$$

This leads to the following result: *Theorem 1:*

$$trace(S_w^{-1}S_b) = \sum_{i=1}^{C} (\bar{x}_i - \bar{x})' S_w^{-1} (\bar{x}_i - \bar{x})$$
(4)

The proof of this statement is similar to the (population) mean and covariance matrix version in [9]. From this theorem, we can see that the trace criterion 4 can be considered as the sum of squared Mahalanobis distances from the class means to the overall means. In addition, when the number of classes is two, the criterion 4 can be considered as the empirical estimation of KL-divergence of two multivariate normal distributions that have the same covariance matrix [28].

Since this criterion measures the separability of classes, we aim to maximize it. For notational simplicity, we write $\{R,f\}$ instead of $R \cup \{f\}$, and $t_{R,f}$ instead of $t_{\{R,f\}}$. We then have the following theorem:

Theorem 2: Let R be the set of selected features, and f be an arbitrary feature that does not belong to R.

Let S_{Rf} be the value of S_w when $\{R, f\}$ is the set of selected features, and S_R be the value of S_w when R is the set of selected features. Partition

$$S_{Rf} = \begin{pmatrix} S_R & v \\ v' & u \end{pmatrix}, \tag{5}$$

where $u\in\mathbb{R}^+,v\in\mathbb{R}^{|R|\times 1},|R|$ is the cardinality of R. If $u-v'S_R^{-1}v>0$ then

$$t_{R,f} \ge t_R.$$
 (6)

Else, if $u - v' S_R^{-1} v < 0$ then

$$t_{R,f} < t_R. (7)$$

The proof of this theorem is given in Appendix A. This theorem implies that the improvement or degradation of the

model after adding a feature depends not only on the features themselves but also on the interaction between that feature and the features already in the model. In addition, from Equation 7, we can see that adding a feature may also reduce the class separability of a model. Hence, feature selection is needed to remove these features and the features that can only improve the model performance insignificantly. This property is desirable and not all feature selection criteria satisfy it. For example, for the mean square error (MSE), adding extra features, regardless of whether that feature is redundant or not, never increase MSE, as stated in the following theorem:

Theorem 3: Let MSE_X be the resulting mean squared error when we regress Y based on X. Suppose T contains arbitrary features to be added to the model and

$$U = (X \quad T). \tag{8}$$

Let MSE_U be the resulting mean squared error when we regress Y based on U. If $M = (T'T - T'X(X'X)^{-1}X'T)^{-1}$ exists then, $MSE_U \leq MSE_X$.

The proof of this theorem is given in Appendix B.

V. PARALLEL FEATURE SELECTION USING TRACE CRITERION (PFST ALGORITHM)

This section details the PFST algorithm for parallel feature selection. In addition to the desirable properties of the trace criterion as discussed in the previous section, the method is based on the following observations:

First, it is well known that forward feature selection is faster than backward feature selection because it sequentially adds the best relevant feature into the model. However, it is still computationally expensive. A potential way to speed up the process is to adapt the Early Dropping heuristic [29], in which features that are deemed unlikely to increase the performance of the model are discarded in subsequent iterations. In the original paper [29], the authors evaluate such possibilities by p-values. Yet, computing p-value using bootstrap is expensive. In our approach, we use the Early Dropping heuristic with the trace criterion as the evaluation tool instead. This does not require multiple iterations through the data for a forward step as using p-value for evaluation.

Second, stepwise feature selection appears to be a remedy to the forward error in the forward selection. It adds features to the model sequentially as in the forward feature selection. In addition, after adding a new feature, this approach removes from the model feature(s) that is no longer important according to the feature selection criterion being used. However, there is a computational cost associated with the backward steps that remove unnecessary features. [30] proposes an algorithm that takes a backward step only when the squared error is no more than half of the squared error decrease in the earlier forward steps. Yet, for a large-scale dataset, there is computational cost with checking a backward step, while there may not exist a redundant feature after a forward at all. In fact, [31] provides a simulation example to show that a backward step is rarely taken in stepwise feature selection. Hence, in our approach, we

Algorithm 4. Parallel forward-backward algorithm with early dropping

Input: Set A of all features, partitioned into feature blocks $F_1, ..., F_b$, response Y, forward threshold α , backward threshold β , early dropping threshold γ , maximum number of re-forward steps maxRef.

Output: List R of relevant features.

Procedure:

- 1: $f_b \leftarrow \arg\max_f \{t_{f_f}\}: f \in F_b\}, b = 1, ..., B$ parallelly on B workers.
- 2: $R \leftarrow \{f_1, ..., f_B\}$.
- 3: # Forward with forward-dropping stage:
- 4: **while** $\bigcup_{1 < b < B} F_b \neq \emptyset$ **do**
- $f_b, F_b \stackrel{\frown}{\leftarrow} OneForwardDropping(F_b, Y, \alpha, \gamma)$ (run parallel on B workers).
- $R \leftarrow R \cup \{f_1, ..., f_B\}$ 6:
- 7: end while
- 8: # Re-forward stage:
- 9: Reset the selection pool to $A \setminus R$ and partition it into $F_1, ..., F_B$.
- 10: runs $\leftarrow 0$
- 11: while runs < maxRef & $\bigcup_{1 \le b \le B} F_b \neq \emptyset$ do
- $f_b, F_b \leftarrow \text{OneReforward}(\overline{F_b}, \overline{Y}, \alpha)$ (run parallel on B workers)
- $R \leftarrow R \cup \{f_1, ..., f_B\}$ 13:
- $runs \leftarrow runs +1$ 14:
- 15: end while
- 16: # Backward stage:
- 17: Partition R into B blocks of features.
- 18: $f_b \leftarrow \arg\max_{f \in F_b} t_{R \setminus \{f\}}, b = 1, ...B$ parallelly on B
- $\begin{array}{ll} \text{19:} \ f_r \leftarrow \arg\max_{f \in \{f_1, \dots, f_B\}} t_{R \setminus \{f\}} \\ \text{20:} \ \ \textbf{if} \ t_R t_{R \setminus \{f_r\}} < \beta \ \ \textbf{then} \\ \text{21:} \ \ \ R \leftarrow R \setminus \{f_r\} \end{array}$

- 22: **else**
- 23: break
- 24: **end if**
- 25: return R

add features until reaching a satisfactory model and then use backward feature selection to remove those redundant features.

Next, note that for single feature, $trace(S_w^{-1}S_b)$ reduces to

$$t = \frac{\sum_{i=1}^{C} n_i (\bar{x}_i - \bar{x})^2}{\sum_{i=1}^{C} \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i)^2}$$
(9)

and for normalized features, S_b reduced to $S_b = \sum_{i=1}^C n_i \bar{x}_i \bar{x}_i'$.

$$t_R = trace(S_w^{-1} S_b) \tag{10}$$

when only the set of features R is included in the model.

For the sake of clarity, we added an overview of the descriptions for the used notations in Table I. Finally, with all these heuristics and observations, the proposed PFST algorithm is presented in Algorithm 4.

TABLE I: Table of notations

Symbol	Description
C	number of classes
A'	the transpose of matrix A
$\bar{\boldsymbol{x}}_i$	the class mean of the i^{th} class
$ar{oldsymbol{x}}$	the overall mean
R	the set of selected features
$ar{oldsymbol{x}}_{iRf}$	the mean of the i^{th} class when the selected features are $\{R, f\}$
$egin{array}{c} ar{oldsymbol{x}}_{iRf} \ ar{oldsymbol{x}}_{Rf} \end{array}$	the overall mean when the selected features are $\{R, f\}$
n_i	number of observations in the i^{th} class
x_{ij}	the j^{th} sample of the i^{th} class
$S_R^{x_{ij}}$	The value of S_w when R is the set of selected features
t_R	$trace(S_w^{-1}S_b)$ when R is the set of selected features

We firstly partition the features into B blocks to assign to B workers. At first, the set of selected features R is empty. Therefore, we compute $\arg\max_f\{t_f: f\in F_b\}, b=1,...,B$ parallel on B workers to produce an initial set R.

Next, we use the OneForwardDropping algorithm (Algorithm 5) to sequentially add relevant features to the model and discard seemingly unimportant ones from subsequent iterations. To be more specific, the algorithm removes all the features in the pool that can not improve the trace criterion more than any amount of β . This helps reduce the computational cost of re-scanning through the features that temporarily do not seem to improve the model a lot compared to other features. However, a major drawback of the forward with early dropping stage is that a seemingly useless feature at a step may be helpful later when combined with other features. Therefore, after these steps are finished, we run the re-forward stage, we reset the features pool to $P = A \setminus R$ and re-partition into B blocks, and conduct forward selection without early dropping. Since most of the important features are already in the model, this stage is not as computationally expensive as running forward selection parallelly when combined with the forward stage. We illustrate in the experiments that our method runs only in a fraction of time compared to the other techniques.

Algorithm 5. OneForwardDropping

Input: Feature block F_b , response Y, forward threshold α , early dropping threshold γ .

Output: a relevant feature f_b and the remaining features in the selection pool F_B for worker B.

```
1: f_b \leftarrow \arg \max_{f \in F_b} t_{R,f}

2: if t_{R,f_b} - t_R < \alpha then

3: # if there is no more relevant features, drop block by setting it to \emptyset

4: F_b \leftarrow \emptyset

5: else

6: # Early dropping:

7: D \leftarrow \{f: t_{\{R,f\}} - t_R < \gamma\}

8: F_b \leftarrow F_b \setminus \{D, f_b\}

9: end if

10: return f_b, F_b
```

Finally, note that the inclusion of new features may lead to

some features that are included before becoming redundant, possibly due to feature interaction. Therefore, the last stage of PFST is the Backward Stage, which tries to remove the redundant features that forward-dropping and re-forward stages may have made before. In this stage, we conduct backward feature selection steps to remove the redundant features that remained in the model. Again, this is to avoid the computational cost of checking for a backward move after every inclusion of a new feature. This is different from how backward steps are conducted in stepwise selection.

Algorithm 6. OneReforward

Input: Feature block F_b , response Y, forward threshold

Output: A relevant feature f_b and the remaining features in the selection pool F_B for worker B.

```
1: f_b \leftarrow \arg\max_{f \in F_b} t_{R,f}

2: if t_{R,f_b} - t_R < \alpha then

3: # if there is no more relevant features, drop block by setting it to \emptyset:

4: F_b \leftarrow \emptyset

5: else

6: F_b \leftarrow F_b \setminus \{f_b\}

7: end if

8: return f_b, F_b
```

Note that a higher γ will result in more feature dropping and less re-scanning during forward-dropping moves. However, depending on the data and the chosen criterion, we may prefer to use a lower γ for high dimensional data. The reason is that a higher γ may result in too many feature droppings, which implies that much fewer features have chances to get into the model during the forward-dropping moves. This causes the forward-dropping moves to terminate early, and we have to re-scan a large number of features during the re-forward stage.

It is also important to point out that after the forward-dropping steps are the re-forward steps. Therefore, after the forward-dropping steps, the algorithm ranks the importance of the features. It is possible to specify the maximum number of features to be included in the model if one wishes for a smaller set of features than what the thresholds may produce. Hence, we can specify the maximum number of features to be included in the model for the re-forward stage.

Last but not least, if the number of features selected after the re-forward stage is not significantly large, one should use the regular backward feature selection instead of parallel backward feature selection. This is because when the number of features in the model is small, parallelism makes the process run slower.

VI. EXPERIMENTS

To illustrate the performance of our approach, we compare our method with Parallel Sequential Forward Selection (PSFS) [32], Parallel Sequential Backward Selection (PSBS) [32], Parallel Support Vector Machine Feature Selection based on Recursive Feature Elimination with Cross-Validation (PSVMR)

TABLE II: The description of datasets that are used in our experiments

Dataset	# Classes	# Features	# Samples	
Breast cancer	2	30	569	
Parkinson	2	754	756	
Mutants	2	5408	31419	
Gene	5	20531	801	
Micromass	10	1087	360	

[33], and Parallel Mutual Information-based Feature Selection (PMI) [34].

A. Datasets & Settings

The experiments are done on the datasets from the Scikitlearn library [32] and UCI Machine Learning repository [35]. The details of these datasets are given in Table II.

For the p53 mutants dataset, we eliminated a row of all null values. In addition, we added a small amount of noise to the Gene expression cancer RNA-Seq dataset and the p53 mutants dataset to avoid inversion error resulting from matrix singularity when running PSFT.

For PSFT, the thresholds used are $\alpha=\gamma=0.05, \beta=0.01.$ For PSBS and PSFS, we used the K-nearest neighbors algorithm with K=3 as the estimator. For PSVMR, we used linear kernel for PSVMR and "JMI" for PMI. For a fair comparison, we force other feature selection techniques to select the same number of features as PSFT.

We run the experiments on an AMD Ryzen 7 3700X CPU with 8 Cores and 16 processing threads, 3.6GHz and 16GB RAM. After selecting the relevant features, we classify the samples using *linear discriminant analysis (LDA)* and report the 5-fold misclassification rate in Table III. In addition, we present the running time and the number of selected features compared to the total number of features in Table IV.

We terminate an experiment if no result is produced after five hours of running or when having the out-of-memory issue, and denote this as NA in the result Tables III and IV).

B. Results and discussion

From the Tables III and IV, we can see that our PFST feature selection method has excellent performance not only in terms of speed but also the accuracy after doing classification. For example, in the breast cancer dataset, PFST was able to produce a set of 3 relevant features out of 30 features in only 0.095 seconds (s), while PSBS and PSVMR need 8.268s and 12.470s, respectively. Even though being the fastest feature selection method, it can achieve the best classification result with a misclassification rate of only 0.042, which is as good as classification using all available features. Moreover, we can observe that the performance of PFST is better than PSBS, which is the parallel version of backward feature selection.

Interestingly, for the Parkinson dataset, PFST selects only 11 out of 754 features. Still, the misclassification rate is the lowest (0.112), which is only 33% of the misclassification rate of classification using all available features (0.362), and

66.87% of the misclassification rate of the next best performer PMI (0.181). This implies that removing redundant features may boost the performance of the classifier significantly. For the running time, note that PFST takes only 3.219s to get 11 features from 754 features, while PMI takes 77.269s, PSFS takes 163.32s. PSBS and PSVMR can not get the results when running for more than 5 hours.

In the dataset with the most features, the Gene Expression Cancer RNA-Seq dataset, PFST reduced 20531 features to 12 features within 3 minutes and got the best misclassification rate of only 0.006. Although PSFS and PMI have also gotten good performance with a misclassification rate of only 0.009 and 0.007, respectively, they need much longer than PFST, while PSBS and PSVMR cannot get the result within 5 hours.

Even though PFST is the fastest algorithm among the ones used in the experiments, it outperforms other approaches in four out of five datasets in terms of classification results. For the Micromass dataset, the best performer for the classification is PSVMR, and the next best is PFST. However, PSVMR is only slightly better than PFST in terms of classification result (1.9% lower misclassification rate), but its running time is almost three times longer than that of PFST (46.909s and 16.1s, respectively).

VII. CONCLUSION

This paper presents a novel parallel feature selection approach for classification, called PFST, for feature selection on large scale datasets. The method uses the trace criterion, i.e., a criterion with many desirable properties, to evaluate feature usefulness. We evaluate the approach via various experiments and datasets using Linear Discriminant Analysis as the classifier on selected features. The experiments show that our PFST method can select relevant features in a fraction amount of time compared to other compared state of the art approaches. In addition, the classifier trained on the features selected by PFST achieves better accuracy than those selected by other methods in four out of five cases and better than the model that is fitted on all available features. However, the trace criterion can only be used for continuous features. Therefore, in the future, it would be desirable to explore how to extend this work to the case where there are categorical features in the model as well.

REFERENCES

- I. Guyon, J. Li, T. Mader, P. A. Pletscher, G. Schneider, and M. Uhr, "Competitive baseline methods set new standards for the nips 2003 feature selection benchmark," *Pattern recognition letters*, vol. 28, no. 12, pp. 1438–1444, 2007.
- [2] N. Melab, S. Cahon, and E.-G. Talbi, "Grid computing for parallel bioinspired algorithms," *Journal of parallel and Distributed Computing*, vol. 66, no. 8, pp. 1052–1061, 2006.
- [3] J. T. de Souza, S. Matwin, and N. Japkowicz, "Parallelizing feature selection," *Algorithmica*, vol. 45, no. 3, pp. 433–456, 2006.
- [4] D. J. Garcia, L. O. Hall, D. B. Goldgof, and K. Kramer, "A parallel feature selection algorithm from random subsets," in *Proceedings of the* international workshop on parallel data mining, vol. 18. Citeseer, 2006, pp. 64–75.

TABLE III: 5-fold misclassification rate

Datasets	# Selected Features	PFST (our)	PSFS	PSBS	PSVMR	PMI	Full Features
Breast cancer	3	0.042	0.111	0.074	0.051	0.076	0.042
Parkinson	11	0.112	0.234	NA	NA	0.181	0.362
Mutants	6	0.008	NA	NA	NA	NA	0.010
Gene	12	0.006	0.009	NA	NA	0.007	0.042
Micromass	19	0.115	0.310	0.218	0.096	0.228	0.129

TABLE IV: Running time and number of selected features

Datasets	# selected	# Features	Running Time (s)				
	features		PFST (our)	PSFS	PSBS	PSVMR	PMI
Breast cancer	3	30	0.095	1.403	8.268	12.470	0.646
Parkinson	11	754	3.219	163.32	NA	NA	77.269
Mutants	6	5408	674.702	NA	NA	NA	NA
Gene	12	20531	172.386	5350.14	NA	NA	2706.45
Micromass	19	1087	16.1	252.9	10561.3	46.909	144.684

- [5] A. Guillén, A. Sorjamaa, Y. Miche, A. Lendasse, and I. Rojas, "Efficient parallel feature selection for steganography problems," in *International Work-Conference on Artificial Neural Networks*. Springer, 2009, pp. 1224–1231.
- [6] F. G. López, M. G. Torres, B. M. Batista, J. A. M. Pérez, and J. M. Moreno-Vega, "Solving feature subset selection problem by a parallel scatter search," *European Journal of Operational Research*, vol. 169, no. 2, pp. 477–489, 2006.
- [7] H. Liu and H. Motoda, Feature selection for knowledge discovery and data mining. Springer Science & Business Media, 2012, vol. 454.
- [8] V. Kumar and S. Minz, "Feature selection: a literature review," SmartCR, vol. 4, no. 3, pp. 211–229, 2014.
- [9] R. A. Johnson, D. W. Wichern et al., Applied multivariate statistical analysis. Prentice hall Upper Saddle River, NJ, 2002, vol. 5, no. 8.
- [10] G. James, D. Witten, T. Hastie, and R. Tibshirani, An introduction to statistical learning. Springer, 2013, vol. 112.
- [11] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, Feature selection for high-dimensional data. Springer, 2015.
- [12] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Computing Surveys* (CSUR), vol. 50, no. 6, pp. 1–45, 2017.
- [13] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proceedings of* the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 188–197.
- [14] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *bioinformatics*, vol. 23, no. 19, pp. 2507– 2517, 2007.
- [15] J. C. Ang, A. Mirzal, H. Haron, and H. N. A. Hamed, "Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 13, no. 5, pp. 971–989, 2015.
- [16] H. Arai, C. Maung, K. Xu, and H. Schweitzer, "Unsupervised feature selection by heuristic search with provable bounds on suboptimality," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.
- [17] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," 1988.
- [18] X. V. Nguyen, J. Chan, S. Romano, and J. Bailey, "Effective global approaches for mutual information based feature selection," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 512–521.
- [19] A. Shishkin, A. Bezzubtseva, A. Drutsa, I. Shishkov, E. Gladkikh, G. Gusev, and P. Serdyukov, "Efficient high-order interaction-aware feature selection based on conditional mutual information," in *Advances* in neural information processing systems, 2016, pp. 4637–4645.

- [20] A. K. Farahat, A. Ghodsi, and M. S. Kamel, "An efficient greedy method for unsupervised feature selection," in 2011 IEEE 11th International Conference on Data Mining. IEEE, 2011, pp. 161–170.
- [21] M. Masaeli, Y. Yan, Y. Cui, G. Fung, and J. G. Dy, "Convex principal feature selection," in *Proceedings of the 2010 SIAM International* Conference on Data Mining. SIAM, 2010, pp. 619–628.
- [22] S. Singh, J. Kubica, S. Larsen, and D. Sorokina, "Parallel large scale feature selection for logistic regression," in *Proceedings of the 2009* SIAM international conference on data mining. SIAM, 2009, pp. 1172– 1183.
- [23] I. Tsamardinos, G. Borboudakis, P. Katsogridakis, P. Pratikakis, and V. Christophides, "A greedy feature selection algorithm for big data of high dimensionality," *Machine learning*, vol. 108, no. 2, pp. 149–202, 2019.
- [24] Z. Zhao, R. Zhang, J. Cox, D. Duling, and W. Sarle, "Massively parallel feature selection: an approach based on variance preservation," *Machine learning*, vol. 92, no. 1, pp. 195–220, 2013.
- [25] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information theory*, vol. 50, no. 10, pp. 2231–2242, 2004.
- [26] D. L. Donoho, M. Elad, and V. N. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Transactions on information theory*, vol. 52, no. 1, pp. 6–18, 2005.
- [27] K. Fukunaga, Introduction to statistical pattern recognition. Elsevier, 2013.
- [28] D. Kong and C. Ding, "Pairwise-covariance linear discriminant analysis," in Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.
- [29] G. Borboudakis and I. Tsamardinos, "Forward-backward selection with early dropping," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 276–314, 2019.
- [30] T. Zhang, "Adaptive forward-backward greedy algorithm for learning sparse representations," *IEEE transactions on information theory*, vol. 57, no. 7, pp. 4689–4708, 2011.
- [31] T. Nguyen, "Faster feature selection with a dropping forward-backward algorithm," arXiv preprint arXiv:1910.08007, 2019.
- [32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [33] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1, pp. 389–422, 2002.
- [34] M. Bennasar, Y. Hicks, and R. Setchi, "Feature selection using joint mutual information maximisation," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8520–8532, 2015.
- [35] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[36] M. H. Kutner, C. J. Nachtsheim, J. Neter, W. Li et al., Applied linear statistical models. McGraw-Hill Irwin New York, 2005, vol. 5.

APPENDIX

A. Proof of Theorem 2

Let \bar{x}_{iR}, \bar{x}_R be the mean of the i^{th} class, and the overall mean when a set R of features are included in the model, respectively. Similarly, let $\bar{x}_{iRf}, \bar{x}_{Rf}$ be the mean of the i^{th} class, and the overall mean when a set $\{R, f\}$ of features are included in the model, respectively. Next, note that

$$\bar{\boldsymbol{x}}_{iRf} = \begin{pmatrix} \bar{\boldsymbol{x}}_{iR} \\ \bar{\boldsymbol{x}}_{if} \end{pmatrix}, \tag{11}$$

and

$$\bar{\boldsymbol{x}}_{Rf} = \begin{pmatrix} \bar{\boldsymbol{x}}_R \\ \bar{\boldsymbol{x}}_f \end{pmatrix},\tag{12}$$

where \bar{x}_{if} , \bar{x}_f is the i^{th} class mean, and the overall mean of the feature f, respectively.

If $u \neq v' S_R^{-1} v$ then $M = (u - v' S_R^{-1} v)^{-1}$ exists. Hence,

$$S_{Rf}^{-1} = \begin{pmatrix} S_R^{-1} + S_R^{-1} v M v' S_R^{-1} & -S_R^{-1} v M \\ -M v' S_R^{-1} & M \end{pmatrix}$$
(13)

Next, let

$$\eta_{iRf} = (\bar{x}_{iRf} - \bar{x}_{Rf})' S_{Rf}^{-1} (\bar{x}_{iRf} - \bar{x}_{Rf})
= \eta_{iR} + (\bar{x}_{iR} - \bar{x}_R)' S_R^{-1} v M v' S_R^{-1} (\bar{x}_{iR} - \bar{x}_R)
- 2(\bar{x}_{iR} - \bar{x}_R)' S_R^{-1} v M (\bar{x}_{if} - \bar{x}_f)
+ (\bar{x}_{if} - \bar{x}_f)' M (\bar{x}_{if} - \bar{x}_f),$$

where

$$\eta_{iR} = (\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R)' S_R^{-1} (\bar{\boldsymbol{x}}_{iR} - \bar{\boldsymbol{x}}_R).$$

Note that $M \in \mathbb{R}, M \neq 0$. Hence,

$$\frac{\eta_{iRf} - \eta_{iR}}{M} = (\bar{x}_{iR} - \bar{x}_R)' S_R^{-1} v v' S_R^{-1} (\bar{x}_{iR} - \bar{x}_R)
- 2(\bar{x}_{iR} - \bar{x}_R)' S_R^{-1} v (\bar{x}_{if} - \bar{x}_f)
+ (\bar{x}_{if} - \bar{x}_f)' (\bar{x}_{if} - \bar{x}_f).$$

Applying Cauchy-Schwarz inequality $||a||^2 + ||b||^2 \ge 2\langle a, b \rangle$, with $a = (\bar{x}_{iR} - \bar{x}_R)S_R^{-1}v$, and $b = (\bar{x}_{if} - \bar{x}_f)$, we see that

$$\frac{\eta_{iRf} - \eta_{iR}}{M} \ge 0.$$

Since,

$$t_{R,f} - t_R = \sum_{i=1}^{C} (\eta_{iRf} - \eta_{iR}),$$

it follows that if $u - v' S_B^{-1} v > 0$ then

$$t_{R,f} \geq t_R$$

and if $u - v' S_R^{-1} v < 0$ then

$$t_{R,f} < t_R$$
.

B. Proof of Theorem 3

Let $H = X(X'X)^{-1}X'$ then it is known from [36] that I - H is idempotent, i.e.,

$$(I-H)^2 = I - H \tag{14}$$

Hence, the sum of square when regressing Y over X is

$$SSE_X = (Y - \hat{Y}_X)'(Y - \hat{Y}_X)$$
$$= Y'(I - H)Y$$

where \hat{Y}_X is the predicted outcome when regressing Y over X.

Note that M is symmetric, and let \hat{Y}_U is the predicted outcome when regressing Y over U. Hence,

$$\begin{split} &(U'U)^{-1}U'Y\\ &=HY+HTMT'HY-HTMT'Y-TMT'HY+TMT'Y\\ &=HY-(I-H)TMT'+(I-H)TMT'Y. \end{split}$$

This implies

$$Y - \hat{Y}_U$$

$$= Y - \hat{Y}_X + (I - H)TMT'HY - (I - H)TMT'Y$$

$$= Y - \hat{Y}_X - (I - H)TMT'(I - H)Y$$

Hence, the sum of square error when regressing Y over U is

$$SSE_{U} = (Y - \hat{Y}_{U})'(Y - \hat{Y}_{U})$$

= $SSE_{X} - 2Y'(I - H)TMT'(I - H)(Y - \hat{Y}_{X})$
+ $Y'(I - H)TMT'(I - H)TMT'(I - H)Y$

In addition, note that (I-H)TMT' is idempotent, and that $Y'(I-H)TMT'(I-H)Y \ge 0$. Therefore,

$$Y'(I-H)TMT'(I-H)TMT'(I-H)Y$$

$$= Y'(I-H)TMT'(I-H)Y$$

$$\leq 2Y'(I-H)TMT'(I-H)Y.$$

Hence

$$SSE_{U} \leq SSE_{X}$$

and the proof follows.