Massively Parallel Feature Selection: An Approach Based on Variance Preservation

Zheng Zhao, James Cox, David Duling, and Warren Sarle

SAS Institute Inc. 600 Research Drive, Cary, NC 27513, USA

Abstract. Advances in computer technologies have enabled corporations to accumulate data at an unprecedented speed. Large-scale business data might contain billions of observations and thousands of features, which easily brings their scale to the level of terabytes. Most traditional feature selection algorithms are designed for a centralized computing architecture. Their usability significantly deteriorates when data size exceeds hundreds of gigabytes. High-performance distributed computing frameworks and protocols, such as the Message Passing Interface (MPI) and MapReduce, have been proposed to facilitate software development on grid infrastructures, enabling analysts to process large-scale problems efficiently. This paper presents a novel large-scale feature selection algorithm that is based on variance analysis. The algorithm selects features by evaluating their abilities to explain data variance. It supports both supervised and unsupervised feature selection and can be readily implemented in most distributed computing environments. The algorithm was developed as a SAS High-Performance Analytics procedure, which can read data in distributed form and perform parallel feature selection in both symmetric multiprocessing mode and massively parallel processing mode. Experimental results demonstrated the superior performance of the proposed method for large scale feature selection.

Keywords: Feature selection, parallel processing, big-data.

1 Introduction

Feature selection is an effective technique for dimensionality reduction and relevance detection [1]. It improves the performance of learning models in terms of their accuracy, efficiency, and model interpretability [2]. As an indispensable component for successful data mining applications, feature selection has been used in a variety of fields, including text mining, image processing, and genetic analysis, to name a few. Continual advances in computer-based technologies have enabled corporations and organizations to collect data at an increasingly fast pace. Business and scientific data from many fields, such as finance, genomics, and physics, are often measured in terabytes (10^{12} bytes). The enormous proliferation of large-scale data sets brings new challenges to data mining techniques and requires novel approaches to address the big-data problem [3] in feature selection. Scalability is critical for large-scale data mining. Unfortunately, most existing feature selection algorithms do not scale well, and their efficiency significantly deteriorates or even becomes inapplicable, when the data size reaches hundreds of gigabytes (10^9 bytes). Efficient distributed programming protocols and frameworks, such as the Message Passing Interface (MPI) [4] and MapReduce [5], are proposed to facilitate programming on high-performance distributed computing infrastructures to handle very large-scale problems.

This paper presents a novel distributed parallel algorithm for handling large-scale problems in feature selection. The algorithm can select a subset of features that best explain (preserve) the variance contained in the data. According to how data variance is defined, the algorithm can perform either unsupervised or supervised feature selection. And for the supervised case, the algorithm also supports both regression and classification. Redundant features increase data dimensionality unnecessarily and worsen the learning performance [6, 7]. The proposed algorithm selects features by evaluating feature subsets and can therefore handle redundant features effectively. For parallel feature selection, the computation of the proposed algorithm is fully optimized and parallelized based on data partitioning. The algorithm is implemented as a SAS High-Performance Analytics procedure[1], which can read data in a distributed form and perform parallel feature selection in both symmetric multiprocessing (SMP) mode via multithreading and massively parallel processing (MPP) mode via MPI.

A few approaches have been proposed for parallel feature selection. In [8, 9, 10, 11], parallel processing is used to speed up feature selection by evaluating multiple features or feature subsets simultaneously. Since all these algorithms require each parallel processing unit to access the whole data, they do not scale well when the sample size is huge. To handle large scale problems, an algorithm needs to rely on data partitioning to ensure its scalability [12]. In [13], a parallel feature selection algorithm is proposed for logistic regression. The algorithm is implemented under the MapReduce framework and can evaluate features using a criterion obtained by approximating the objective function of the logistic regression model. After selecting each new feature, the algorithm needs to retrain its model, which is an iterative process. In contrast, the proposed algorithm solves a problem with a closed form solution in each step and therefore might be more efficient. To the best knowledge of the authors, all existing parallel feature selection algorithms are for supervised learning, while the proposed algorithm supports both supervised and unsupervised feature selection.

The contributions of this paper are: (1) The proposed algorithm provides a unified approach for both unsupervised and supervised feature selection. For supervised feature selection, it also supports both regression and classification. (2) The proposed algorithm can effectively handle redundant features in feature selection. (3) The algorithm is fully optimized and parallelized based on data partitioning, which ensures its scalability for handling large-scale problems. To the best knowledge of the authors, this is the first distributed parallel algorithm for unsupervised feature selection.

[1] A SAS procedure is a c-based routine for statistical analysis in the SAS system.

2 Maximum Variance Preservation for Feature Selection

This section presents a multivariate formulation for feature selection based on maximum variance preservation. It first shows how to use the formulation to perform unsupervised feature selection, then extends it to support supervised feature selection for both regression and classification.

2.1 Unsupervised Feature Selection

When label information is unavailable, feature selection becomes challenging. To address this issue, researchers propose various criteria for unsupervised feature selection. For example, in [14], the performance of a clustering algorithm is used to evaluate the utility of a feature subset; in [15, 16], each feature's ability to preserve locality is evaluated and used to select features; and in [17] an entropy-based criterion is proposed and used for feature selection. This paper proposes a multivariate formulation for feature evaluation in a distributed computing environment. The criterion is based on maximum variance preservation, which promotes the selection of the features that can best preserve data variance.

Assume that k features need to be selected. Let $\mathbf{X} \in \mathbb{R}^{n \times m}$ be a data set that contains n samples, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, and m features, $\mathbf{f}_1, \ldots, \mathbf{f}_m$. In this work, it is assumed that all features have been centralized to have zero mean, $\mathbf{1}^\top \mathbf{f} = \mathbf{0}$, where $\mathbf{1}$ is a column vector with all its elements being 1. Let $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2)$, where $\mathbf{X}_1 \in \mathbb{R}^{n \times k}$ contains the k selected features and $\mathbf{X}_2 \in \mathbb{R}^{n \times (m-k)}$ contains the remaining ones. The proposed maximum variance preservation criterion selects features by minimizing the following expression:

$$arg \min_{\mathbf{X}_1} Trace\left(\mathbf{X}_2^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top\right) \mathbf{X}_2\right) \qquad (1)$$

Let $\mathbf{X}_1 = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be the singular value decomposition of $\mathbf{X}_1$, and let $\mathbf{U} = (\mathbf{U}_R, \mathbf{U}_N)$, where $\mathbf{U}_R$ contains the left singular vectors that correspond to the nonzero singular values and $\mathbf{U}_N$ contains the left singular vectors that correspond to the zero singular values. It can be verified that $\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top = \mathbf{U}_N {\mathbf{U}_N}^\top$, therefore the following equation holds:

$$Trace\left(\mathbf{X}_2^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top\right) \mathbf{X}_2\right) = Trace\left(\left(\mathbf{U}_N^\top \mathbf{X}_2\right)^\top \left(\mathbf{U}_N^\top \mathbf{X}_2\right)\right) \qquad (2)$$

The columns of $\mathbf{U}_N$ span the null space of $\mathbf{X}_1^T$. Since each row of $\mathbf{X}_1^\top$ corresponds to a feature in $\mathbf{X}_1$, $\mathbf{U}_N^\top \mathbf{X}_2$ effectively projects the features in $\mathbf{X}_2$ to the null space of the features in $\mathbf{X}_1$. Therefore, Expression (1) measures the variance that resides in the null space of $\mathbf{X}_1^\top$, which is the variance that cannot be explained by the features in $\mathbf{X}_1$. And minimizing it leads to the selection of the features that can jointly explain the maximum amount of the data variance.

2.2 Supervised Feature Selection

When label information is available, Expression (1) can be extended to support feature selection in both regression and classification (categorization) settings.

The Regression Case. In a regression setting, all responses are numerical. Let $\mathbf{Y} \in \mathbb{R}^{n \times t}$ be the response matrix that contains t response vectors, and $\mathbf{X}_1$ and $\mathbf{X}_2$ are defined as before. Assume that k features need to be selected. In a regression setting, feature selection can be achieved by minimizing:

$$arg \min_{\mathbf{X}_1} Trace\left(\mathbf{Y}^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top\right) \mathbf{Y}\right) \tag{3}$$

where $\left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top\right)^{\frac{1}{2}} = \mathbf{U}_N$ projects $\mathbf{Y}$ to the null space of $\mathbf{X}_1^\top$. Expression (3) measures the response variance that resides in the null space of $\mathbf{X}_1^\top$, which is the variance of $\mathbf{Y}$ that cannot be explained by the features in $\mathbf{X}_1$. Clearly, minimizing the expression leads to selecting the features that can jointly explain the maximum amount of response variance.

The Classification Case. In a classification setting, one categorical response is specified. Let the response vector be $\mathbf{y}$ with C different values, $\{1, \ldots, C\}$. A response matrix $\mathbf{Y} \in \mathbb{R}^{n \times C}$ can be created using the following equation:

$$\mathbf{Y}_{i,j} = \begin{cases} \left(\sqrt{\frac{1}{n_j}} - \frac{\sqrt{n_j}}{n}\right) & , y_i = j \\ -\frac{\sqrt{n_j}}{n} & , y_i \neq j \end{cases} \tag{4}$$

where n_j is the number of instances in class j, and $y_i = j$ denotes that the ith instance belongs to the jth class. This $\mathbf{Y}$ is first used in [18] for least square linear discriminant analysis. Applying it in Expression (3) enables feature selection in a classification setting, which leads to selecting the features that maximize the discriminant criterion of linear discriminant analysis (LDA).

Theorem 1. *Assume that features have been centralized to have zero mean and that the response matrix* $\mathbf{Y}$ *is defined by Equation (4). Minimizing Expression (3) is equivalent to maximizing the discriminant criterion of LDA,*

$$\max Trace\left(\mathbf{S}_t^{-1} \mathbf{S}_b\right) \tag{5}$$

where $\mathbf{S}_t$ *and* $\mathbf{S}_b$ *are the total and the between-class scatter matrices on* $\mathbf{X}_1$*.*

Proof. Let $\mathbf{Y}$ be defined in Equation (4), and all features have zero mean. The theorem can be proved by verifying the following equations:

$$\frac{1}{n}\mathbf{X}^\top \mathbf{X} = \mathbf{S}_t = \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{x}_i - \mathbf{c}\right) \left(\mathbf{x}_i - \mathbf{c}\right)^\top \tag{6}$$

$$\mathbf{X}^\top \mathbf{Y} \mathbf{Y}^\top \mathbf{X} = \mathbf{S}_b = \frac{1}{n} \sum_{j=1}^{C} n_j \left(\mathbf{c}_j - \mathbf{c}\right) \left(\mathbf{c}_j - \mathbf{c}\right)^\top \tag{7}$$

In the preceding equations, $\mathbf{c}$ is the mean of the whole data. Since features have been centralized to have zero mean, in the preceding equations $\mathbf{c} = 0$. $\mathbf{x}_i$ is the ith instance, and $\mathbf{c}_j$ is the mean of the instances that belong to class j.

The discriminant criterion of LDA measures the separability of the instances from different classes. For example, Expression (5) achieves a large value when instances from the same class are close, while instances from different classes are far away from each other. When Equation (4) is applied in Expression (3) for feature selection, the features that maximize the separability of the instances from different classes are selected. This is a desirable property for classifiers.

3 The Computation

Given m features, finding the k features minimizing Expressions (1) and (3) is a combinatorial optimization problem, which is NP-hard (nondeterministic polynomial-time hard). The sequential forward selection (SFS) strategy[2] is an efficient way of generating a suboptimal solution for the problem [1]. This section derives closed form solutions for the problem based on sequential forward selection, which significantly improves its efficiency. It also presents algorithms for computing the solutions in a distributed parallel computing environment.

3.1 Closed Form Solutions Based on SFS

Solution for Unsupervised Feature Selection. Assume that q features have been selected. Let $\mathbf{X}_1$ contain the q selected features, and let $\mathbf{X}_2$ contain the remaining ones. In the $q + 1$ step, a feature $\mathbf{f}$ is selected by

$$arg \min_{\mathbf{f}} Trace \left(\hat{\mathbf{X}}_2^\top \left(\mathbf{I} - \hat{\mathbf{X}}_1 \left(\hat{\mathbf{X}}_1^\top \hat{\mathbf{X}}_1 \right)^{-1} \hat{\mathbf{X}}_1^\top \right) \hat{\mathbf{X}}_2 \right) \tag{8}$$

where $\hat{\mathbf{X}}_1$ contains $\mathbf{f}$ and the q selected features, and $\hat{\mathbf{X}}_2$ contains the remaining ones. Let $\mathbf{U}_N = \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1 \right)^{-1} \mathbf{X}_1^\top \right)^{\frac{1}{2}}$, the following theorem applies:

Theorem 2. *Solving the problem specified in Expression (8) is equivalent to maximizing the following expression:*

$$arg \max_{\mathbf{f}} \frac{\left\| \mathbf{X}_2^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1 \right)^{-1} \mathbf{X}_1^\top \right) \mathbf{f} \right\|_2^2}{\left\| \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1 \right)^{-1} \mathbf{X}_1^\top \right)^{\frac{1}{2}} \mathbf{f} \right\|_2^2} \tag{9}$$

Proof. The theorem can be proved by applying block matrix inversion on $\hat{\mathbf{X}}_1^\top \hat{\mathbf{X}}_1$.

$$\left(\hat{\mathbf{X}}_1^\top \hat{\mathbf{X}}_1 \right)^{-1} = \begin{pmatrix} \mathbf{A}^{-1} + \frac{1}{w} \mathbf{A}^{-1} \mathbf{b} \mathbf{b}^\top \mathbf{A}^{-1} & -\frac{1}{w} \mathbf{A}^{-1} \mathbf{b} \\ -\frac{1}{w} \mathbf{b}^\top \mathbf{A}^{-1} & \frac{1}{w} \end{pmatrix} \tag{10}$$

where $\mathbf{A} = \mathbf{X}_1^\top \mathbf{X}_1$, $\mathbf{b} = \mathbf{X}_1^\top \mathbf{f}$, $c = \mathbf{f}^\top \mathbf{f}$, and $w = c - \mathbf{b}^\top \mathbf{A}^{-1} \mathbf{b}$. The details of the proof is omitted due to space limit.

[2] To select k features, the sequential forward selection (SFS) strategy applies k steps of greedy search and selects one feature in each step.

Assuming that all features have zero mean, $\left\|\mathbf{X}_2^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top\right) \mathbf{f}\right\|_2^2$ in Equation (9) is the summation of the squares of the covariance between the feature $\mathbf{f}$ and all the unselected features (columns of $\mathbf{X}_2$) in the null space of $\mathbf{X}_1^\top$. And $\left\|\left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top\right)^{\frac{1}{2}} \mathbf{f}\right\|_2^2$ is the square of the variance of the feature $\mathbf{f}$ in the null space of $\mathbf{X}_1^\top$, which is used for normalization. Essentially, Expression (9) measures how well the feature $\mathbf{f}$ can explain the variance that cannot be explained by the q selected features. Compared to Expression (8), Expression (9) singles out the computations that are common for evaluating different features. This makes it possible to compute them only once in each step and therefore improves the efficiency for solving the problem.

Let m be the number of all features, n the number of samples, and k the number of features to be selected. Also assume that $m \gg k$. It is easy to verify that in a traditional centralized computing environment, the time complexity for selecting k features by solving Expression (9) is:

$$O\left(m^2\left(n + k^2\right)\right) \tag{11}$$

In the preceding expression, $m^2 n$ corresponds to the complexity for computing the covariance matrix. And $m^2 k^2$ corresponds to selecting k features out of m.

Solution for Supervised Feature Selection. The following theorem enables efficient feature selection with Expression (3):

Theorem 3. *When the problem specified in Expression (3) is solved by sequential forward selection, in each step the selected feature* $\mathbf{f}$ *must maximize:*

$$arg \max_{\mathbf{f}} \frac{\left\|\mathbf{Y}^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top\right) \mathbf{f}\right\|_2^2}{\left\|\left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top\right)^{\frac{1}{2}} \mathbf{f}\right\|_2^2} \tag{12}$$

Proof. It can be proved in the same way as Theorem 2.

Let C be the number of columns in $\mathbf{Y}$. The time complexity of selecting k features using Expression (12) is

$$O\left(mk\left(n + k^2\right)\right) \tag{13}$$

To obtain Expression (13), it is assumed that $m \gg k > C$.

3.2 Parallel Computation through MPP and SMP

The operations for computing Expression (9) and (12) need to be carefully ordered, optimized, and parallelized to ensure efficiency and scalability.

Massive Parallel Processing (MPP). The master-worker/slave architecture based on MPI [4] is used to support massive parallel processing. In this architecture, given $p + 1$ parallel processing units, one unit is used as the master for control, and the remaining p units is used as workers for computation. In the implementation, all expensive operations for computing feature relevance are properly decomposed, so that they can be computed in parallel based on data partitioning. Assume that a data set has n instances and m features. p homogenous computers (the workers) are available. A data partitioning technique evenly distributes instances to the workers, so that each worker obtain $\frac{n}{p}$ instances for computation. It is shown in [20] that any operation fitting the Statistical Query model[3] can be computed in parallel based on data partitioning. Studies also showed that when data size is large enough, parallelization based on data partitioning can result in linear speedup as computing resources increase [20, 12]. Algorithms 1 and 2 contain the implementation details for distributed parallel feature selection based on MPI. The validness of the computation can be verified by decomposing operations into various summation forms over instances. The details about the verification is not presented due to space limit.

Symmetric Multiprocessing (SMP). Solving the problems specified in Expression (9) and (12) involves a series of matrix-vector operations. These operations are packed together and rewritten in the matrix-matrix operation form. This effectively simplifies programming and allows developers to use a highly optimized threaded BLAS library to speed up computation on the workers through multi-threading. As an example, in unsupervised feature selection, let $t_{i_{r,j}} = \mathbf{f}_{i_{r,j}}^\top \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top \mathbf{f}_{i_{r,j}}$, where $\mathbf{f}_{i_{r,j}}$ is the j-th feature on the r-th worker. $(t_{i_{r,1}}, \ldots, t_{i_{r,\frac{m}{p}}})$ can be computed as, $(t_{i_{r,1}}, \ldots, t_{i_{r,\frac{m}{p}}}) = \mathbf{1}^\top (\mathbf{B}_r \bigotimes \mathbf{E}_r)$, where $\bigotimes$ denotes element-wise matrix multiplication. Let $\mathbf{X}_r = (\mathbf{f}_{i_{r,1}}, \ldots, \mathbf{f}_{i_{r,\frac{m}{p}}})$ and $\mathbf{A} = \mathbf{X}_1^\top \mathbf{X}_1$, it can be verified that $\mathbf{B}_r = \mathbf{X}_1^\top \mathbf{X}_r$, and $\mathbf{E}_r = \mathbf{A}^{-1}\mathbf{B}_r$.

3.3 The Implementations

Algorithm 1 and 2 contain the pseudocode for unsupervised and supervised feature selection respectively. Both algorithms assume that the data have been properly partitioned and distributed to p worker nodes. In the algorithms, $\bigotimes$ and $\oslash$ denote element-wise matrix multiplication and division, respectively.

For unsupervised feature selection, the covariance among features is used repeatedly in the evaluation process. Therefore, it is more efficient to compute the whole covariance matrix $\mathbf{C}$ before feature selection. In Algorithm 1, **Line 2** to **Line 5** compute feature scores to select the first feature. Since no feature has been selected, Expression (9) can be simplified to $\frac{\|\mathbf{X}^\top \mathbf{f}_i\|_2^2}{\mathbf{f}_i^\top \mathbf{f}_i} = \frac{\|\mathbf{c}^i\|_2^2}{c_{i,i}}$, where $\mathbf{c}^i$ is the ith column of $\mathbf{C}$, and $C_{i,i}$ is the ith diagonal element. In **Line 2**, $\mathbf{v}_r$ contains the

[3] An operation fits the Statistical Query model if it can be decomposed and written in summation forms over the instances.

```
    Input: $\mathbf{X}_1, \ldots, \mathbf{X}_p, \in \mathbb{R}^{\frac{n}{p} \times m}$; $k$
    Output: $\mathbb{L}$, a list of $k$ selected features
 1  Compute covariance matrix $\mathbf{C} \in \mathbb{R}^{m \times m}$, and distribute the $r$th section of
    the covariance matrix, $\mathbf{C}_r \in \mathbb{R}^{m \times \frac{m}{p}}$, on the $r$th worker, $r = 1, \ldots, p$ ;
 2  Compute local feature scores on each worker
        $\mathbf{s}_r = \mathbf{1}^\top \left(\mathbf{C}_r \bigotimes \mathbf{C}_r\right),\ \mathbf{s}_r = \mathbf{s}_r \oslash \mathbf{v}_r;\ \mathbf{v}_r = \left(C_{i_{r,1}, i_{r,1}}, \ldots, C_{i_{r,\frac{m}{p}}, i_{r,\frac{m}{p}}}\right)$   (14)
 3  Workers send $\mathbf{s}_r$ to the master via MPI_Gather;
 4  On the master, select $i = arg\max\left(s_i \mid s_i \in (\mathbf{s}_1, \ldots, \mathbf{s}_p)\right)$;
 5  Initialization, $\mathbb{L} = \{F_i\}$, $l = 1$;
 6  while $l < k$ do
 7      The master sends $\mathbb{L}$ to all workers via MPI_Bcast;
 8      The worker that contains $\mathbf{c}^i$, the $i$th column of $\mathbf{C}$, sends $\mathbf{c}^i$ to all other
        workers via MPI_Bcast;
        /* ------------------simultaneously------------------- */
 9      Workers construct $\mathbf{A}^{-1} \in \mathbb{R}^{l \times l}$, $\mathbf{B}_r \in \mathbb{R}^{l \times t_r}$, $\mathbf{D}_r \in \mathbb{R}^{(m-l) \times t_r}$,
        $\mathbf{v}_r \in \mathbb{R}^{t_r \times 1}$, $\mathbf{C}_{2,1} \in \mathbb{R}^{(m-l) \times l}$;
10      Workers compute local feature scores
            $\mathbf{E}_r = \mathbf{A}^{-1}\mathbf{B}_r,\quad \mathbf{H}_r = \mathbf{C}_{2,1}\mathbf{E}_r,\quad \mathbf{G}_r = \mathbf{D}_r - \mathbf{H}_r,$   (15)
            $\mathbf{g}_r = \mathbf{1}^\top \left(\mathbf{G}_r \bigotimes \mathbf{G}_r\right), \mathbf{w}_r = \mathbf{v}_r - \mathbf{1}^\top \left(\mathbf{B}_r \bigotimes \mathbf{E}_r\right), \mathbf{s}_r = \mathbf{g}_r \oslash \mathbf{w}_r$   (16)
        Workers send $\mathbf{s}_r$ to the master via MPI_Gather;
        /* --------------------------------------------------- */
11      Master selects $i = arg\max\left(s_i \mid s_i \in (\mathbf{s}_1, \ldots, \mathbf{s}_p)\right)$, $\mathbb{L} = \mathbb{L} \cup \{F_i\}$, $l++$;
12  end
```

Algorithm 1. Distributed parallel unsupervised feature selection.

diagonal elements of **C** that corresponds to the variance of the features on the rth worker. The vector $\mathbf{s}_r$ contains the scores of the features on the rth worker. After a feature F_i has been selected, each worker updates $\mathbf{A}^{-1}$, $\mathbf{B}_r$, $\mathbf{D}_r$, $\mathbf{v}_r$, and $\mathbf{C}_{2,1}$ in **Line 9** using $\mathbf{C}_r$ and $\mathbf{c}^i$. Let $\mathbb{L}$ contain the index of selected features, $\mathbb{L}_r$ contain the index of unselected features on the rth worker, and $\mathbb{L}_u$ contain the index of all unselected features. $\mathbf{A} = \mathbf{X}_1^\top \mathbf{X}_1 = \mathbf{C}_{\mathbb{L} \times \mathbb{L}}$, $\mathbf{B}_r = \mathbf{X}_1^\top \mathbf{X}_r = \mathbf{C}_{\mathbb{L} \times \mathbb{L}_r}$, $\mathbf{D}_r = \mathbf{X}_2^\top \mathbf{X}_r = \mathbf{C}_{\mathbb{L}_u \times \mathbb{L}_r}$, $\mathbf{C}_{2,1} = \mathbf{X}_2^\top \mathbf{X}_1$, and $\mathbf{v}_r$ contains the variance of the unselected features on the rth worker. The scores of the features on the rth worker is computed in **Line 10**. Assume that the $\mathbf{A}^{-1}$ in **Line 9** can be computed by applying rank-one update, and a tree-based mechanism is used to implement `MPI_Bcast` and `MPI_Reduce`. The total time complexity of Algorithm 1 is

$$CPU\left(\frac{m^2\left(n + k^2\right)}{p} + m^2 \log p\right) + NET\left(m^2 \log p\right) \tag{17}$$

In the preceding expressions, $CPU\left(\cdot\right)$ and $NET\left(\cdot\right)$ denote the time used for computation and for network communication, respectively.

```
   Input: $\mathbf{X}_1, \ldots, \mathbf{X}_p \in \mathbb{R}^{\frac{n}{p} \times m}$, $\mathbf{Y}_1, \ldots, \mathbf{Y}_p \in \mathbb{R}^{\frac{n}{p} \times C}$, $k$
   Output: $\mathbb{L}$, a list of $k$ selected features
 1 On each worker, compute $\mathbf{E}_r \in \mathbb{R}^{C \times m}$, $\mathbf{v}_r \in \mathbb{R}^{1 \times m}$:
       $$\mathbf{E}_r = \mathbf{Y}_r^\top \mathbf{X}_r, \quad \mathbf{v}_r = \mathbf{1}^\top \left(\mathbf{X}_r \bigotimes \mathbf{X}_r\right); \qquad (18)$$
 2 Send $\mathbf{E}_r$ and $\mathbf{v}_r$ to the master via MPI_Reduce with MPI_SUM option:
       $$\mathbf{E} = \sum_{r=1}^{p} \mathbf{E}_r, \quad \mathbf{v} = \sum_{r=1}^{p} \mathbf{v}_r; \qquad (19)$$
 3 On the master, compute feature scores
       $$\mathbf{s} = \mathbf{1}^\top \left(\mathbf{E} \bigotimes \mathbf{E}\right), \quad \mathbf{s} = \mathbf{s} \bigoslash \mathbf{v}; \qquad (20)$$
 4 On the master, select $i = arg\max\left(s_i \mid s_i \in \mathbf{s}\right)$;
 5 Initialization, $\mathbb{L} = \{F_i\}$, $l = 1$;
 6 while $l < k$ do
 7     The master sends $\mathbb{L}$ to all workers via MPI_Bcast;
       /* --------------simultaneously---------------- */
 8     Workers compute $\mathbf{c}_r^i = \mathbf{X}_r^\top \mathbf{f}_r^i$, $\mathbf{c}_r^i \in \mathbb{R}^{m \times 1}$;
 9     Workers send $\mathbf{c}_r^i$ to the master via MPI_Reduce with MPI_SUM option
           $$\mathbf{c}^i = \sum_{r=1}^{p} \mathbf{c}_r^i, \ \mathbf{c}^i \in \mathbb{R}^{m \times 1} \qquad (21)$$
       /* -------------------------------------------- */
10     On the master, construct $\mathbf{A}^{-1} \in \mathbb{R}^{l \times l}$, $\mathbf{C}_{\mathbf{Y},1} \in \mathbb{R}^{C \times l}$,
       $\mathbf{C}_{1,2} \in \mathbb{R}^{l \times (m-l)}$, $\mathbf{C}_{\mathbf{Y},2} \in \mathbb{R}^{C \times (m-l)}$, $\mathbf{v}_2 \in \mathbb{R}^{1 \times (m-l)}$;
11     On the master, compute
           $$\mathbf{B} = \mathbf{A}^{-1}\mathbf{C}_{1,2}, \quad \mathbf{H} = \mathbf{C}_{\mathbf{Y},1}\mathbf{B}, \quad \mathbf{G} = \mathbf{C}_{\mathbf{Y},2} - \mathbf{H}; \qquad (22)$$
           $$\mathbf{g} = \mathbf{1}^\top \left(\mathbf{G} \bigotimes \mathbf{G}\right), \quad \mathbf{w} = \mathbf{v}_2 - \mathbf{1}^\top \left(\mathbf{C}_{1,2} \bigotimes \mathbf{B}\right), \quad \mathbf{s} = \mathbf{g} \bigoslash \mathbf{w}; \qquad (23)$$
12     Master selects $i = arg\max\left(s_i \mid s_i \in \mathbf{s}\right)$, $\mathbb{L} = \mathbb{L} \cup \{F_i\}$, $l++$;
13 end
```

Algorithm 2. Distributed parallel supervised feature selection.

For supervised feature selection, only a small portion of the covariance matrix is needed for feature evaluation. Therefore, the covariance matrix is not computed before feature selection. In Algorithm 2, **Line 1** to **Line 3** compute feature scores to select the first feature. Since no feature has been selected, Expression (12) simplifies to $\frac{\|\mathbf{X}^\top \mathbf{f}\|_2^2}{\mathbf{f}^\top \mathbf{f}}$. In **Line 10**, $\mathbf{A} = \mathbf{X}_1^\top \mathbf{X}_1$, $\mathbf{C}_{\mathbf{Y},1} = \mathbf{Y}^\top \mathbf{X}_1$, $\mathbf{C}_{\mathbf{Y},2} = \mathbf{Y}^\top \mathbf{X}_2$, $\mathbf{C}_{1,2} = \mathbf{X}_1^\top \mathbf{X}_2$, and $\mathbf{v}_2$ contains the variance of the unselected features. As both $\mathbf{A}^{-1}$ and $\mathbf{B}$ can be obtained by incrementally updating their previous versions, the complexity for selecting k features using Algorithm 2 is

$$CPU\left(mk\left(\frac{n}{p}+k^2\right)\right)+NET\left(m\left(C+k\right)\log p\right) \tag{24}$$

In the preceding expression, C is the number of columns in $\mathbf{Y}$.

Expression (17) and (24) suggest that when the number of instances is large and the network is fast enough, Algorithms 1 and 2 can speed up feature selection linearly as the number of parallel processing units increases.

4 Connections to Existing Methods

In an unsupervised setting, principal component analysis (PCA) [19] also reduces dimensionality by preserving data variance. The key difference between PCA and the proposed method is that PCA generates a small set of new features (feature extraction) by linearly combining the original features, while the proposed method selects a small set of the original features (feature selection). The features returned by the proposed method are the original ones. This is very important in applications where retaining the original features is useful for model exploration or interpretation (for example, genetic analysis and text mining).

In a regression setting, let $\mathbf{f}$ be a feature vector, it can be shown that

$$\mathbf{f}^\top\left(\mathbf{I}-\mathbf{X}_1\left(\mathbf{X}_1^\top\mathbf{X}_1\right)^{-1}\mathbf{X}_1^\top\right)\mathbf{Y}=\mathbf{f}^\top\left(\mathbf{Y}-\mathbf{X}_1\mathbf{W}_1\right) \tag{25}$$

where $\mathbf{W}_1=\left(\mathbf{X}_1^\top\mathbf{X}_1\right)^{-1}\mathbf{X}_1^\top\mathbf{Y}$ is the solution of a least squares regression. Let $\mathbf{R}$ be the residual, $\mathbf{R}=\mathbf{Y}-\mathbf{X}_1\mathbf{W}_1$. Expression (12) can be simplified to:

$$arg\max_{\mathbf{f}}\frac{\left\|\mathbf{f}^\top\mathbf{R}\right\|_2^2}{\left\|\left(\mathbf{I}-\mathbf{X}_1\left(\mathbf{X}_1^\top\mathbf{X}_1\right)^{-1}\mathbf{X}_1^\top\right)^{\frac{1}{2}}\mathbf{f}\right\|_2^2} \tag{26}$$

Therefore, in each step the proposed method selects the feature that has the largest normalized correlation with the current residual. This shows that in a regression setting the proposed method forms a special type of stepwise regression with Expression (12) as the selection criterion.

When used in a classification setting, the proposed method selects features with the discriminant criterion of LDA. LDA also reduces dimensionality. As for PCA, the key difference is that LDA generates a small set of new features, while the proposed method selects a small set of the original features.

5 Experimental Study

The proposed method was implemented as the HPREDUCE procedure based on SAS High-Performance Analytics foundation. This section evaluates its performance for both supervised and unsupervised feature selection. In the experiment, 12 representative feature selection algorithms are used for comparison. For unsupervised feature selection, six algorithms are selected as baselines: Laplacian score [15], SPEC-1 and SPEC-3 [16], trace-ratio [21], HSIC [22], and SPFS [23]. For supervised feature selection, in the classification setting, seven algorithms are compared: ReliefF [24], Fisher Score [25], trace-ratio, HSIC, mRMR [7], AROM-SVM [26], and SPFS. In the regression setting, LARS [27], and LASSO [28] are compared. Among the 12 algorithms, AROM-SVM, mRMR, SPFS, LARS and LASSO can handle redundant features.

Table 1. Summary of the benchmark data sets

Data Set	Features	Instances	Classes	Data Set	Features	Instances	Classes
RELATH	4,322	1,427	2	ORL	10,000	100	10
PCMAC	3,289	1,943	2	CRIME	147	2,215	-
AR	2,400	130	10	SLICELOC	386	53,500	-
PIE	2,400	210	10	s25mf5k	5,000	25,000,000	-
PIX	10,000	100	10	u10mf5k	5,000	10,000,000	-

Ten benchmark data sets are used in the experiment. Four are face image data: AR[4], PIE[5], PIX[6], and ORL[7] (images from 10 persons are used). Two are text data extracted from the 20-newsgroups data[8]: RELATH (BASEBALL vs. HOCKEY) and PCMAC (PC vs. MAC). Two are UCI data: CRIME (Communities and Crime Unnormalized) and SLICELOC (relative location of CT slices on axial axis)[9]. And two are large-scale data sets for performance tests. The u10mf5k data set contains 5,000 features and 10 million instances, which is used for testing unsupervised feature selection. The s25mf5k data set contains 5,000 features, 1 response, and 25 million instances, which is used for testing supervised feature selection. Each data set has 100 continuous variables sampled from uniform distribution. And the remains are binary variables sampled from Bernoulli distribution. Details on the ten data sets can be found in Table 1. The first six data sets are used to test unsupervised feature selection and supervised feature selection for classification. The seventh and the eighth data sets are used to test feature selection for regression. And the last two are used to evaluate the HPREDUCE procedure in a distributed computing environment.

Assume that $\mathbb{L}$ is the set of selected features and that $\mathbf{X}_{\mathbb{L}}$ is the data that contain only features in $\mathbb{L}$. For the classification setting, algorithms are compared on (1) classification accuracy and (2) redundancy rate which is defined as:

$$RED\left(\mathbb{L}\right) = \frac{1}{m(m-1)} \sum_{F_i, F_j \in \mathbb{L}, i>j} \rho_{i,j} \tag{27}$$

[4] http://rvl1.ecn.purdue.edu/~leix/aleix_face_DB.html
[5] http://peipa.essex.ac.uk/ipa/pix/faces/manchester/
[6] http://www.ri.cmu.edu/projects/project_418.html
[7] http://www.uk.research.att.com/facedatabase.html
[8] http://people.csail.mit.edu/jrennie/20Newsgroups/
[9] http://archive.ics.uci.edu/ml/index.html

where $\rho_{i,j}$ returns the correlation between feature F_i and feature F_j. Equation (27) assesses the average correlation among all feature pairs. A large value indicates that features in $\mathbb{L}$ are strongly correlated and thus redundant features might exist. In the regression setting, algorithms are compared on (1) rooted mean square error (RMSE) and (2) redundancy rate. For unsupervised feature selection, algorithms are compared on: (1) redundancy rate and (2) percentage of the total variance explained by features in $\mathbb{L}$,

$$PCT_{VAR}(\mathbb{L}) = \frac{Trace\left(\mathbf{X}^\top \mathbf{X}_{\mathbb{L}} \left(\mathbf{X}_{\mathbb{L}}^\top \mathbf{X}_{\mathbb{L}}\right)^{-1} \mathbf{X}_{\mathbb{L}}^\top \mathbf{X}\right)}{Trace\left(\mathbf{X}^\top \mathbf{X}\right)} \tag{28}$$

For each data set, half of the instances are randomly sampled for training and the remaining are used for test. The process is repeated 20 times, which results in 20 different partitions of the data set. Each feature selection algorithm is used to select $5, 10, \ldots, 100$ features on each partition. The obtained 20 feature subsets are then evaluated using a criterion $\mathcal{C}$. By doing this, a score matrix $\mathbf{S} \in \mathbb{R}^{20 \times 20}$ is generated for each algorithm, where each row of $\mathbf{S}$ corresponds to a data partition and each column corresponds to a size of the feature subset. The average score of $\mathcal{C}$ is obtained by $s = \frac{\mathbf{1}^\top \mathbf{S} \mathbf{1}}{20 \times 20}$. To calculate classification accuracy, linear support vector machine (SVM) is used. The parameters of SVM and all feature selection algorithms are tuned via 5 fold cross-validation on the training data. Let $\mathbf{s} = \frac{\mathbf{1}^\top \mathbf{S}}{20}$. The elements of $\mathbf{s}$ corresponds to the average score achieved when different numbers of features are selected. The paired Student's t test is applied to compare the $\mathbf{s}$ achieved by different algorithms to $\mathbf{s}^*$, the best $\mathbf{s}$ measured by $\mathbf{1}^\top \mathbf{s}$. And the threshold for rejecting the null hypothesis is set to 0.05. Rejecting the null hypothesis means that $\mathbf{s}$ and $\mathbf{s}^*$ are significantly different, and suggests that the performance of the algorithm is consistently different to the best algorithm when different numbers of selected features.

5.1 Study of Unsupervised Cases

Percentage of Explained Variance: Table 2 presents the average percentage of the data variance explained by the features selected by different algorithms. The result shows that compared with the baselines, the HPREDUCE procedure achieved the best performance on all six data sets. This is to be expected, since the HPREDUCE procedure is designed to preserve data variance. The result demonstrates the strong capability of the proposed algorithm for preserving variance in feature selection. It also suggests that using Expression (9) with sequential forward search is effective for minimizing Expression (1).

Redundancy Rate: Table 3 presents the average redundancy rate results. It shows that SPFS and the HPREDUCE procedure achieved much better results than the others. This is to be expected, since they are designed to handle redundant features, while the others are not.

Table 2. Unsupervised feature selection: explained variance with p-val

Algorithm	PCMAC	RELATH	PIX	PIE	AR	ORL	AVE	Best
Laplacian	0.13 (.00)	0.10 (.00)	0.57 (.00)	0.76 (.00)	0.55 (.00)	0.45 (.00)	0.427	0
SPEC-1	0.13 (.00)	0.10 (.00)	0.57 (.00)	0.75 (.00)	0.56 (.00)	0.45 (.00)	0.427	0
SPEC-3	0.21 (.00)	0.18 (.00)	0.61 (.00)	0.78 (.00)	0.58 (.00)	0.52 (.00)	0.481	0
Trace-ratio	0.44 (.00)	0.45 (.00)	0.57 (.00)	0.75 (.00)	0.56 (.00)	0.45 (.00)	0.537	0
HSIC	0.42 (.00)	0.44 (.00)	0.62 (.00)	0.75 (.00)	0.55 (.00)	0.45 (.00)	0.538	0
SPFS	0.45 (.00)	0.47 (.01)	0.74 (.01)	0.86 (.00)	0.72 (.01)	0.60 (.01)	0.639	0
HPREDUCE	**0.60** (1.0)	**0.54** (1.0)	**0.97** (1.0)	**0.97** (1.0)	**0.96** (1.0)	**0.97** (1.0)	**0.835**	6

Table 3. Unsupervised feature selection: redundancy rate with p-val

Algorithm	PCMAC	RELATH	PIX	PIE	AR	ORL	AVE	Best
Laplacian	0.70 (.00)	0.78 (.00)	0.90 (.00)	0.85 (.00)	0.82 (.00)	0.85 (.00)	0.817	0
SPEC-1	0.71 (.00)	0.78 (.00)	0.90 (.00)	0.87 (.00)	0.80 (.00)	0.85 (.00)	0.818	0
SPEC-3	0.84 (.00)	0.93 (.00)	0.89 (.00)	0.81 (.00)	0.78 (.00)	0.73 (.00)	0.829	0
Trace-ratio	0.20 (.00)	0.27 (.00)	0.90 (.00)	0.87 (.00)	0.80 (.00)	0.85 (.00)	0.649	0
HSIC	0.17 (.00)	0.25 (.00)	0.90 (.00)	0.84 (.00)	0.80 (.00)	0.85 (.00)	0.633	0
SPFS	0.08 (.00)	0.11 (.00)	0.36 (.00)	**0.31** (1.0)	**0.24** (1.0)	**0.26** (.05)	0.227	3
HPREDUCE	**0.02** (1.0)	**0.02** (1.0)	**0.22** (1.0)	0.34 (.01)	0.27 (.01)	**0.22** (1.0)	**0.181**	4

5.2 Study of Supervised Cases

Classification, Accuracy: Table 4 presents the average accuracy achieved by SVM using the features selected by algorithms. The HPREDUCE procedure achieved the best results on five data sets, which is followed by SPFS (three data sets) and Arom-SVM (two data sets). According to the average accuracy, the HPREDUCE procedure also performed the best (0.880), followed by SPFS (0.869) and HSIC (0.813). This result demonstrates the good performance of the HPREDUCE procedure in the classification setting.

Table 4. Supervised feature selection for classification: accuracy with p-val

Algorithm	PCMAC	RELATH	PIX	PIE	AR	ORL	AVE	Best
ReliefF	0.70 (.00)	0.66 (.00)	0.92 (.00)	0.92 (.00)	0.76 (.00)	0.78 (.00)	0.789	0
Fisher Score	**0.86** (1.0)	0.73 (.00)	0.92 (.00)	0.90 (.01)	0.72 (.00)	0.73 (.00)	0.810	1
Trace-ratio	**0.86** (1.0)	0.73 (.00)	0.92 (.00)	0.90 (.01)	0.72 (.00)	0.73 (.00)	0.810	1
HSIC	**0.85** (.14)	0.75 (.00)	0.92 (.00)	0.90 (.01)	0.72 (.00)	0.74 (.00)	0.813	1
mRMR	0.84 (.00)	**0.79** (.81)	0.85 (.00)	0.92 (.02)	0.64 (.00)	0.68 (.00)	0.787	1
Arom-SVM	**0.85** (.14)	0.75 (.00)	0.80 (.00)	**0.90** (.09)	0.55 (.00)	0.71 (.00)	0.761	2
SPFS	**0.85** (.32)	0.78 (.02)	0.95 (.02)	**0.94** (.14)	**0.80** (.13)	0.89 (.00)	0.869	3
HPREDUCE	0.84 (.00)	**0.80** (1.0)	**0.96** (1.0)	**0.95** (1.0)	**0.81** (1.0)	**0.92** (1.0)	**0.880**	5

Classification, Redundancy Rate: The average redundancy rate achieved by algorithms are presented in Table 5. Among the eight algorithms in the table, mRMR, Arom-SVM, SPFS, and the HPREDUCE procedure are designed to handle redundant features. In the experiment, on average these algorithms achieved redundancy rates at the level of 0.2. In contrast, the other four algorithms had much higher redundancy rates. The result shows that the HPREDUCE procedure is effective in handling redundant features for classification.

Table 5. Supervised feature selection for classification: redundancy rate with p-val

Algorithm	PCMAC	RELATH	PIX	PIE	AR	ORL	AVE	Best
ReliefF	0.10 (.00)	0.09 (.00)	0.78 (.00)	0.38 (.00)	0.76 (.00)	0.89 (.00)	0.501	0
Fisher Score	0.07 (.00)	0.15 (.00)	0.83 (.00)	0.40 (.00)	0.67 (.00)	0.77 (.00)	0.481	0
Trace-ratio	0.07 (.00)	0.15 (.00)	0.83 (.00)	0.40 (.00)	0.67 (.00)	0.77 (.00)	0.481	0
HSIC	0.13 (.00)	0.10 (.00)	0.83 (.00)	0.40 (.00)	0.67 (.00)	0.77 (.00)	0.483	0
mRMR	**0.04** (1.0)	0.04 (.00)	0.33 (.00)	**0.26** (.46)	**0.25** (1.0)	**0.25** (1.0)	**0.194**	4
Arom-SVM	0.05 (.00)	0.07 (.00)	**0.26** (1.0)	0.29 (.02)	**0.25** (.22)	**0.25** (.35)	0.196	3
SPFS	0.11 (.00)	0.07 (.00)	0.45 (.00)	**0.25** (1.0)	0.31 (.03)	0.36 (.00)	0.260	1
HPREDUCE	0.05 (.00)	**0.03** (1.0)	0.32 (.00)	0.31 (.00)	0.31 (.00)	0.27 (.00)	0.214	1

Regression: In the regression setting, the HPREDUCE procedure is compared to LARS and LASSO. The RMSE and redundancy rate results are presented in Tables 6, respectively. The results suggest that in terms of RMSE and redundancy rate, the performance of the three algorithms are largely comparable on the benchmark data sets. Compared to LARS and LASSO, the HPREDUCE procedure is a general method for both supervised and unsupervised feature selection, while LARS and LASSO are for supervised regression only.

Table 6. Supervised feature selection for regression, RMSE (col 2- col 4), the lower the better; redundancy rate (col 5 - col 7) with p-val

DATA	LARS	LASSO	HPREDUCE	LARS	LASSO	HPREDUCE
CRIME	3.6e-7 (.00)	3.6e-7 (.00)	**3.3e-7** (1.0)	**0.31** (1.0)	**0.31** (1.0)	0.32 (.00)
SLICELOC	2.8e-3 (.04)	2.8e-3 (.04)	**2.6e-3** (1.0)	0.17 (.00)	0.17 (.00)	**0.14** (1.0)
Average	1.38e-3	1.38e-3	**1.32e-3**	0.241	0.241	**0.233**
Best	0	0	2	1	1	1

5.3 Study of Scalability

To evaluate the scalability of the HPREDUCE procedure, it was tested in a distributed computing environment. The cluster has 32 nodes, and each node has two Intel Xeon CPUs, 16 GB memory, and two 186GB disk drives. In the experiment, different numbers of workers are used for selecting 200 features from the input data. Compared with the unsupervised case, supervised feature selection with the HPREDUCE procedure has a lower time complexity. Therefore, for supervised feature selection the maximum number of nodes is set to 20, while for unsupervised feature selection, this number is increased to 30. Multiple threads are used on each node for matrix computation.

The running time and the speedup information for both supervised and unsupervised feature selection is presented in Figure 1. It shows that the HPREDUCE procedure generally performs faster when more computing resource is available. For example, when only one worker node is used for computation in the unsupervised case, the HPREDUCE procedure finishes in 1,670.98 seconds. When 30 worker nodes are used, it finishes in just 83.69 seconds. In general, for both supervised and unsupervised feature selection, the speedup of the HPREDUCE procedure is linear. For the supervised case, the speedup ratio (slope of the line) of the HPREDUCE procedure is close to 1, which is quite good. And for the unsupervised case, the speedup ratio is about 0.66. The unsupervised case has

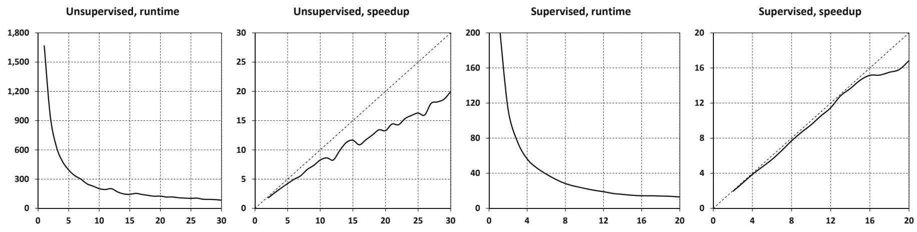

Fig. 1. Runtime and speedup in the unsupervised and the supervised settings with different number of workers for feature selection

a lower speedup ratio because it involves more network communication between the master and the workers in the feature selection process. It can also be observed from the s25mf5k data set that when more than 15 nodes are used for supervised feature selection, the speedup ratio of the HPREDUCE procedure decreases. For a fixed size problem, when too many nodes are used, the warm-up and the communication costs start to offset the increase of computing resources. The results clearly demonstrate the scalability of the proposed algorithm.

6 Conclusions

This paper presents a distributed parallel feature selection algorithm based on maximum variance preservation. The proposed algorithm forms a unified approach for feature selection. By defining the preserving target in different ways, the algorithm can achieve both supervised and unsupervised feature selection. And for supervised feature selection, it also supports both regression and classification. The algorithm performs feature selection by evaluating feature sets and can therefore handle redundant features. The computation of the algorithm is also optimized and parallelized to support both MPP an SMP. As illustrated by an extensive experimental study, the proposed algorithm can effectively remove redundant features and achieve superior performance for both supervised and unsupervised feature selection. The study also shows that given a large-scale data set, the proposed algorithm can significantly improve the efficiency of feature selection through distributed parallel computing. Our ongoing work will extend the HPREDUCE procedure to also support semi-supervised feature selection and sparse feature extraction, such as sparse PCA and sparse LDA.

Acknowledgments. The authors would like to thank An Shu, Anne Baxter, Russell Albright, and the anonymous reviewers for their valuable suggestions to improve this paper.

References

[1] Liu, H., Motoda, H.: Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers, Boston (1998)

[2] Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research 3, 1157–1182 (2003)
[3] Zaki, M.J., Ho, C.T. (eds.): Large-scale parallel data mining. Springer (2000)
[4] Snir, M., et al.: MPI: The Complete Reference. MIT Press, Cambridge (1995)
[5] Dean, J., Ghemawat, S.: System and method for efficient large-scale data processing, United States Patent 7650331 (2010)
[6] Hall, M.: Correlation-Based Feature Selection for Machine Learning. PhD thesis, University of Waikato, Dept. of Computer Science (1999)
[7] Ding, C., Peng, H.: Minimum redundancy feature selection from microarray gene expression data. In: Proceedings of the CSB, pp. 523–529 (2003)
[8] Felix, G.L., et al.: Solving feature subset selection problem by a parallel scatter search. European Journal of Operational Research 169(2), 477–489 (2006)
[9] Melab, N., et al.: Grid computing for parallel bioinspired algorithms. Journal of Parallel and Distributed Computing 66(8), 1052–1061 (2006)
[10] Garcia, D.J., et al.: A parallel feature selection algorithm from random subsets. In: Proceedings of the International Workshop on Parallel Data Mining (2006)
[11] Guillén, A., Sorjamaa, A., Miche, Y., Lendasse, A., Rojas, I.: Efficient Parallel Feature Selection for Steganography Problems. In: Cabestany, J., Sandoval, F., Prieto, A., Corchado, J.M. (eds.) IWANN 2009, Part I. LNCS, vol. 5517, pp. 1224–1231. Springer, Heidelberg (2009)
[12] Kent, P., Schabenberger, O.: SAS high performance computing: The future is not what it used to be (2011), `http://www.monash.com/uploads/SAS_HPA_2011-Longer.pdf`
[13] Singh, S., et al.: Parallel large scale feature selection for logistic regression. In: Proc. of SDM (2009)
[14] Dy, J.G., Brodley, C.E.: Feature selection for unsupervised learn. Journal of Machine Learning Research 5, 845–889 (2004)
[15] He, X., et al.: Laplacian score for feature selection. In: Proc. of NIPS (2005)
[16] Zhao, Z., Liu, H.: Spectral feature selection for supervised and unsupervised learning. In: Proceedings of ICML (2007)
[17] Dash, M., et al.: Feature selection for clustering, a filter solution. In: Proceedings of ICDM (2002)
[18] Ye, J.: Least squares linear discriminant analysis. In: Proceedings of ICML (2007)
[19] Jolliffe, I.T.: Principal Component Analysis, 2nd edn. Springer (2002)
[20] Chu, C.T., et al.: Map-reduce for machine learning on multicore. In: Proceedings of NIPS (2007)
[21] Nie, F., et al.: Trace ratio criterion for feature selection. In: Proc. of AAAI (2008)
[22] Song, L., et al.: Supervised feature selection via dependence estimation. In: Proceedings of ICML (2007)
[23] Zhao, Z., Wang, L., Liu, H., Ye, J.: On similarity preserving feature selection. IEEE Transactions on Knowledge and Data Engineering 99, 198–206 (2011)
[24] Sikonja, M.R., Kononenko, I.: Theoretical and empirical analysis of Relief and ReliefF. Machine Learning 53, 23–69 (2003)
[25] Duda, R., et al.: Pattern Classification, 2nd edn. John Wiley & Sons (2001)
[26] Weston, J., et al.: Use of the zero norm with linear models and kernel methods. Journal of Machine Learning Research 3, 1439–1461 (2003)
[27] Efron, B., et al.: Least angle regression. Annals of Statistics 32, 407–449 (2004)
[28] Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society, Series B 58(1), 267–288 (1994)