Mach Learn (2013) 92:195–220
DOI 10.1007/s10994-013-5373-4

Massively parallel feature selection: an approach based on variance preservation

Zheng Zhao · Ruiwen Zhang · James Cox · David Duling · Warren Sarle

Received: 17 November 2012 / Accepted: 27 April 2013 / Published online: 22 May 2013

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 and implemented for a centralized computing architecture. Their usability significantly deteriorates when data size exceeds tens 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 implemented as a SAS High-Performance Analytics procedure, which can read data in distributed form and perform parallel feature selection in both symmetric multiprocessing mode (SMP) and massively parallel processing mode (MPP). Experimental results demonstrated the superior performance of the proposed method for large scale feature selection.

Keywords Feature selection · Model selection · Parallel processing · Big-data

Editors: Tijl De Bie and Peter Flach

Z. Zhao (✉) · R. Zhang · J. Cox · D. Duling · W. Sarle
SAS Institute Inc., 600 Research Drive, Cary, NC 27513, USA
e-mail: zheng.zhao@sas.com

R. Zhang
e-mail: ruiwen.zhang@sas.com

J. Cox
e-mail: james.cox@sas.com

D. Duling
e-mail: david.duling@sas.com

W. Sarle
e-mail: warren.sarle@sas.com

1 Introduction

Feature selection is an effective technique for dimensionality reduction and relevance detection (Liu and Motoda 1998b; Guyon and Elisseeff 2003). It improves the performance of learning models in terms of their accuracy, efficiency, and model interpretability (Zhao and Liu 2011). As an indispensable component for successful data mining applications, feature selection has been used in a variety of fields, including text mining (Forman 2003), image processing (Manikandan and Rajamani 2008), and genetic analysis (Saeys et al. 2007), 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 (Zaki and Ho 2000) in feature selection. Scalability is critical for large-scale data mining. Unfortunately, most existing feature selection algorithms are implemented for serial computing, and their efficiency significantly deteriorates or even becomes inapplicable, when the data size reaches tens of gigabytes (10^9 bytes). Scalable distributed programming protocols and frameworks, such as the Message Passing Interface (MPI) (Snir et al. 1995) and MapReduce (Dean and Ghemawat 2010), 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 (Hall 1999; Ding and Peng 2003). The proposed algorithm selects features by evaluating feature subsets and can therefore handle redundant features effectively. Determining how many features to select is an important problem in feature selection. When target information is available, the proposed algorithm can automatically determine the number of features to select by using effective model selection techniques, such as the Akaike information criterion (AIC) (Akaike 1974), the Bayesian information criterion (BIC) (Schwarz 1978), and the corrected Hannan–Quinn information criterion (HQC) (Hannan and Quinn 1979). 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 Lopez et al. (2006), Melab et al. (2006), Souza et al. (2006), Garcia et al. (2006), Guillen et al. (2009), 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 (Kent and Schabenberger 2011). In Singh et al. (2009), a parallel feature selection algorithm is proposed for logistic regression. The algorithm is implemented under the MapReduce

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

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. Parallel algorithms have also been designed to generate sparse solution by applying L1-regularization (Bradley et al. 2011) in an SMP environment. Compared to the proposed algorithm, these algorithms only support supervised learning. While the proposed approach supports both supervised and unsupervised feature selection. To the best knowledge of the authors, all existing parallel feature selection algorithms are supervised, while the proposed algorithm supports both supervised and unsupervised learning.

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) It can effectively handle redundant features in feature selection. (3) It can automatically determine how many features to select when target information is available for model selection. (4) It 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. This paper is a significantly expanded version of a paper (Zhao et al. 2012) that appeared in the Proceedings of the 2012 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2012). Compared to the conference version, the following major improvements have been made: (1) The proposed feature selection algorithm is improved by using effective model selection techniques to allow it to automatically determine the number of features to select. (2) Extra experiments are conducted to evaluate the scalability of the proposed algorithm using real large scale data sets on a bigger cluster system. (3) Full revision has been made to adjust the paper structure, add clarifications, proofs, figures, details, and discussions to help readers to understand the proposed algorithm in a better way.

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 in both regression and classification (categorization) settings.

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 Dy and Brodley (2004), the performance of a clustering algorithm is used to evaluate the utility of a feature subset; in He et al. (2005), Zhao and Liu (2007), each feature's ability to preserve locality is evaluated and used to select features; and in Dash et al. (2002) 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 instances, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, 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:

$$\arg\min_{\mathbf{X}_1} \mathrm{Trace}\big(\mathbf{X}_2^\top \big(\mathbf{I} - \mathbf{X}_1\big(\mathbf{X}_1^\top \mathbf{X}_1\big)^{-1}\mathbf{X}_1^\top\big)\mathbf{X}_2\big) \tag{1}$$

Let $\mathbf{X}_1 = \mathbf{U}\Sigma\mathbf{V}^\top$ be the singular value decomposition (SVD) (Golub and Van Loan 1996) 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 $\mathbf{I} - \mathbf{X}_1(\mathbf{X}_1^\top\mathbf{X}_1)^{-1}\mathbf{X}_1^\top = \mathbf{U}_N\mathbf{U}_N{}^\top$. Therefore,

$$\mathrm{Trace}\big(\mathbf{X}_2^\top \big(\mathbf{I} - \mathbf{X}_1\big(\mathbf{X}_1^\top \mathbf{X}_1\big)^{-1}\mathbf{X}_1^\top\big)\mathbf{X}_2\big) = \mathrm{Trace}\big(\big(\mathbf{U}_N^\top\mathbf{X}_2\big)^\top\big(\mathbf{U}_N^\top\mathbf{X}_2\big)\big) \tag{2}$$

The columns of $\mathbf{U}_N$ span the null space of $\mathbf{X}_1^\top$, that is $\mathbf{X}_1^\top\mathbf{U}_N = \mathbf{0}$. Since each row of $\mathbf{X}_1^\top$ corresponds to a feature in $\mathbf{X}_1$, it holds that $\forall \mathbf{f}_i \in \mathbf{X}_1 \Rightarrow \mathbf{f}_i^\top\mathbf{U}_N = \mathbf{0}$. Therefore, $\mathbf{U}_N$ also spans the null space of all the features in $\mathbf{X}_1$. Taking $\mathbf{U}_N$ as a projection matrix, $\mathbf{U}_N^\top\mathbf{X}_2$ effectively project $\mathbf{X}_2$ to the null space of the features in $\mathbf{X}_1$. And $\mathrm{Trace}((\mathbf{U}_N^\top\mathbf{X}_2)^\top(\mathbf{U}_N^\top\mathbf{X}_2))$ measures the variance of $\mathbf{X}_2$ in the null space of $\mathbf{X}_1^\top$, which is the variance of $\mathbf{X}_2$ that cannot be explained by the features in $\mathbf{X}_1$. Therefore, minimizing Expression (1) leads to the selection of the features that can jointly explain the maximum amount of the data variance.

2.2 Supervised feature selection

When target information is available, Expression (1) can be extended to support supervised feature selection for both regression and classification.

2.2.1 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:

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

where $(\mathbf{I} - \mathbf{X}_1(\mathbf{X}_1^\top\mathbf{X}_1)^{-1}\mathbf{X}_1^\top) = \mathbf{U}_N\mathbf{U}_N^\top$, and $\mathbf{U}_N^\top\mathbf{Y}$ projects $\mathbf{Y}$ to the null space of $\mathbf{X}_1^\top$. Expression (3) measures the response variance 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 features that can jointly explain the maximum amount of the response variance.

2.2.2 The classification case

In a classification setting, one categorical response is specified. Let the response vector be $\mathbf{y} \in \mathbb{R}^{n\times 1}$ with C different values, $y_i \in \{1, \ldots, C\}$. A response matrix $\mathbf{Y} \in \mathbb{R}^{n\times C}$ can be created from $\mathbf{y}$ 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 Ye (2007) for least square linear discriminant

analysis (LSLDA). Let $\mathbf{S}_b$ be the between-class scatter matrix in linear discriminant analysis (LDA) (Fisher 1936), which is defined as below:

$$\mathbf{S}_b = \frac{1}{n}\sum_{j=1}^{C} n_j(\mathbf{c}_j - \mathbf{c})(\mathbf{c}_j - \mathbf{c})^\top \tag{5}$$

where $\mathbf{c}$ is the mean of all the instances and $\mathbf{c}_j$ is the mean of the instances in class j. $\mathbf{S}_b$ can be computed based on $\mathbf{Y}$ and $\mathbf{X}$ using the following equation:

$$\mathbf{S}_b = \mathbf{X}^\top \mathbf{Y}\mathbf{Y}^\top \mathbf{X} \tag{6}$$

The following theorem shows that applying this $\mathbf{Y}$ in Expression (3) enables feature selection in a classification setting, which leads to the selection of a set of features that maximize the discriminant criterion of LDA.

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

$$\max \operatorname{Trace}\left(\mathbf{S}_t^{-1}\mathbf{S}_b\right) \tag{7}$$

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

Proof Let $\mathbf{Y}$ be defined in (4), and all features have zero mean. It can be verified that the following two equations hold.

$$\frac{1}{n}\mathbf{X}^\top \mathbf{X} = \mathbf{S}_t = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \mathbf{c})(\mathbf{x}_i - \mathbf{c})^\top \tag{8}$$

$$\mathbf{X}^\top \mathbf{Y}\mathbf{Y}^\top \mathbf{X} = \mathbf{S}_b = \frac{1}{n}\sum_{j=1}^{C} n_j(\mathbf{c}_j - \mathbf{c})(\mathbf{c}_j - \mathbf{c})^\top \tag{9}$$

In the preceding equations, $\mathbf{x}_i$ is the ith instance. $\mathbf{c}$ is the mean of the whole data. Since features have been centralized to have zero mean, $\mathbf{c} = 0$. The theorem can be proved by plugging (8) and (9) into Expression (7). □

The discriminant criterion of LDA measures the separability of the instances from different classes. For example, Expression (7) achieves a large value when instances from the same class are close, while instances from different classes are far away from each other. When (4) is applied in Expression (3) for feature selection, it leads to the selection of the features that maximize the separability of the instances from different classes. This is a desirable property for classifiers to achieve good classification performance.

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; Garey and Johnson 1979). The sequential forward selection (SFS) strategy is an efficient way of generating a suboptimal solution for the problem (Liu and Motoda 1998b). To select k features, the SFS strategy applies k steps of greedy search and selects one feature in each step. This section derives closed form solutions for selecting the best feature in each

SFS step. The closed form solutions significantly improve the efficiency of feature selection by eliminating the redundant computations in computing feature scores. This section also presents efficient algorithms to compute solutions for feature selection with different learning settings in a distributed parallel computing environment.

3.1 Closed form solutions for each SFS step

3.1.1 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

$$\arg\min_{\mathbf{f}} \operatorname{Trace}\big(\hat{\mathbf{X}}_2^\top \big(\mathbf{I} - \hat{\mathbf{X}}_1 \big(\hat{\mathbf{X}}_1^\top \hat{\mathbf{X}}_1\big)^{-1} \hat{\mathbf{X}}_1^\top\big)\hat{\mathbf{X}}_2\big) \tag{10}$$

where $\hat{\mathbf{X}}_1$ contains $\mathbf{f}$ and the q selected features, and $\hat{\mathbf{X}}_2$ contains the remaining ones. Computing Expression (10) for all m features can be prohibitively expensive, when m is large. Let $\mathbf{U}_N^\top = (\mathbf{I} - \mathbf{X}_1(\mathbf{X}_1^\top \mathbf{X}_1)^{-1}\mathbf{X}_1^\top)^{\frac{1}{2}}$, Theorem 2 shows that it can be significantly simplified.

Theorem 2 *Solving the problem specified in Expression* (10) *is equivalent to maximizing*:

$$\arg\max_{\mathbf{f}} \frac{\|\mathbf{X}_2^\top (\mathbf{I} - \mathbf{X}_1(\mathbf{X}_1^\top \mathbf{X}_1)^{-1}\mathbf{X}_1^\top)\mathbf{f}\|_2^2}{\|(\mathbf{I} - \mathbf{X}_1(\mathbf{X}_1^\top \mathbf{X}_1)^{-1}\mathbf{X}_1^\top)^{\frac{1}{2}}\mathbf{f}\|_2^2} \tag{11}$$

Proof It is easy to verify that

$$\operatorname{Trace}\big(\hat{\mathbf{X}}_2^\top \hat{\mathbf{X}}_2\big) = \operatorname{Trace}\big(\mathbf{X}_2^\top \mathbf{X}_2\big) - \mathbf{f}^\top \mathbf{f} \tag{12}$$

Let $\hat{\mathbf{X}}_1 = (\mathbf{X}, \mathbf{f})$. Since $\mathbf{f}$ is in the range (column space) of $\hat{\mathbf{X}}_1$, the following equation holds:

$$\operatorname{Trace}\big(\mathbf{f}^\top \hat{\mathbf{X}}_1 \big(\hat{\mathbf{X}}_1^\top \hat{\mathbf{X}}_1\big)^{-1} \hat{\mathbf{X}}_1^\top \mathbf{f}\big) = \mathbf{f}^\top \mathbf{f} \tag{13}$$

Substituting (12) and (13) into Expression (10) yields

$$\begin{aligned} &\operatorname{Trace}\big(\hat{\mathbf{X}}_2^\top \big(\mathbf{I} - \hat{\mathbf{X}}_1 \big(\hat{\mathbf{X}}_1^\top \hat{\mathbf{X}}_1\big)^{-1} \hat{\mathbf{X}}_1^\top\big)\hat{\mathbf{X}}_2\big) \\ &\quad = \operatorname{Trace}\big(\mathbf{X}_2^\top \mathbf{X}_2\big) - \operatorname{Trace}\big(\mathbf{X}_2^\top \hat{\mathbf{X}}_1 \big(\hat{\mathbf{X}}_1^\top \hat{\mathbf{X}}_1\big)^{-1} \hat{\mathbf{X}}_1^\top \mathbf{X}_2\big) \end{aligned} \tag{14}$$

Let $\mathbf{A} = \mathbf{X}_1^\top \mathbf{X}$, $\mathbf{b} = \mathbf{X}_1^\top \mathbf{f}$, and $c = \mathbf{f}^\top \mathbf{f}$. Then,

$$\hat{\mathbf{X}}_1^\top \hat{\mathbf{X}}_1 = (\mathbf{X}_1, \mathbf{f})^\top (\mathbf{X}_1, \mathbf{f}) = \begin{pmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b}^\top & c \end{pmatrix} \tag{15}$$

Inverting this block matrix (Petersen and Pedersen 2008) yields:

$$\big(\hat{\mathbf{X}}_1^\top \hat{\mathbf{X}}_1\big)^{-1} = \begin{pmatrix} \mathbf{A}^{-1} + \frac{1}{w}\mathbf{A}^{-1}\mathbf{b}\mathbf{b}^\top \mathbf{A}^{-1} & -\frac{1}{w}\mathbf{A}^{-1}\mathbf{b} \\ -\frac{1}{w}\mathbf{b}^\top \mathbf{A}^{-1} & \frac{1}{w} \end{pmatrix} \tag{16}$$

where $w = c - \mathbf{b}^\top \mathbf{A}^{-1}\mathbf{b}$. Let $\mathbf{d} = \mathbf{X}_2^\top \mathbf{f}$, and let $\mathbf{h} = \mathbf{X}_2^\top \mathbf{X}_1 \mathbf{A}^{-1}\mathbf{b}$. By substituting (16) into (14) and simplifying, it can be shown that

$$\begin{aligned} &\operatorname{Trace}\big(\mathbf{X}_2^\top \big(\mathbf{I} - \mathbf{X}_1 \big(\mathbf{X}_1^\top \mathbf{X}_1\big)^{-1} \mathbf{X}_1^\top\big)\mathbf{X}_2\big) \\ &\quad - \operatorname{Trace}\big(\hat{\mathbf{X}}_2^\top \big(\mathbf{I} - \hat{\mathbf{X}}_1 \big(\hat{\mathbf{X}}_1^\top \hat{\mathbf{X}}_1\big)^{-1} \hat{\mathbf{X}}_1^\top\big)\hat{\mathbf{X}}_2\big) \\ &\quad = w^{-1} \|\mathbf{d} - \mathbf{h}\|_2^2 \end{aligned} \tag{17}$$

The theorem can then be proved by verifying that

$$\mathbf{d}-\mathbf{h}=\mathbf{X}_2^\top\left(\mathbf{I}-\mathbf{X}_1\left(\mathbf{X}_1^\top\mathbf{X}_1\right)^{-1}\mathbf{X}_1^\top\right)\mathbf{f} \tag{18}$$

and

$$w=\mathbf{f}^\top\left(\mathbf{I}-\mathbf{X}_1\left(\mathbf{X}_1^\top\mathbf{X}_1\right)^{-1}\mathbf{X}_1^\top\right)\mathbf{f} \tag{19}$$

□

Assuming that all features have zero mean, $\|\mathbf{X}_2^\top(\mathbf{I}-\mathbf{X}_1(\mathbf{X}_1^\top\mathbf{X}_1)^{-1}\mathbf{X}_1^\top)\mathbf{f}\|_2^2$ in (11) 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 $\mathbf{X}_1^\top$. $\|(\mathbf{I}-\mathbf{X}_1(\mathbf{X}_1^\top\mathbf{X}_1)^{-1}\mathbf{X}_1^\top)^{\frac{1}{2}}\mathbf{f}\|_2^2$ 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 (11) measures how well the feature $\mathbf{f}$ can explain the variance that cannot be explained by the q selected features. Compared to Expression (10), Expression (11) 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 significantly improves the efficiency for solving the problem.

Let m be the number of all features, n the number of instances, and k the number of features to select. Also assume that $m \gg k$. In a centralized computing environment, the time complexity for selecting k features by solving Expression (11) is:

$$O\left(m^2\left(n+k^2\right)\right) \tag{20}$$

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

3.1.2 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*:

$$\arg\max_{\mathbf{f}} \frac{\|\mathbf{Y}^\top(\mathbf{I}-\mathbf{X}_1(\mathbf{X}_1^\top\mathbf{X}_1)^{-1}\mathbf{X}_1^\top)\mathbf{f}\|_2^2}{\|(\mathbf{I}-\mathbf{X}_1(\mathbf{X}_1^\top\mathbf{X}_1)^{-1}\mathbf{X}_1^\top)^{\frac{1}{2}}\mathbf{f}\|_2^2} \tag{21}$$

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

Let C be the number of columns in $\mathbf{Y}$. In a centralized computing environment, the time complexity of selecting k features using Expression (21) is

$$O\left(mk\left(n+k^2\right)\right) \tag{22}$$

To obtain Expression (22), it is assumed that $m \gg k > C$.

3.2 Parallel computation through MPP and SMP

The operations for computing Expression (11) and (21) need to be carefully ordered, optimized, and parallelized in a distributed computing environment to ensure the efficiency and scalability of the proposed algorithm for different learning contexts.

3.2.1 Massive parallel processing (MPP)

The master-worker/slave architecture based on MPI 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, and p homogeneous 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 Chu et al. (2007) that any operation fitting the Statistical Query model[2] 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 (Chu et al. 2007; Kent and Schabenberger 2011). Good examples on how to parallelize computation based on data partitioning and the Statistical Query model can be found in Chu et al. (2007).

3.2.2 Symmetric multiprocessing (SMP)

Solving the problems specified in Expression (11) and (21) 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 $t_{i_{r,j}} = \mathbf{f}_{i_{r,j}}^\top \mathbf{X}_1 (\mathbf{X}_1^\top \mathbf{X}_1)^{-1} \mathbf{X}_1^\top \mathbf{f}_{i_{r,j}}$, 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

$$(t_{i_{r,1}}, \ldots, t_{i_{r,\frac{m}{p}}}) = \mathbf{1}^\top (\mathbf{B}_r \otimes \mathbf{E}_r),$$

where $\otimes$ 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 $\mathbf{E}_r = \mathbf{A}^{-1} \mathbf{B}_r$.

Figure 1 illustrates how feature scores are computed in parallel on three workers for unsupervised feature selection. Assume that the data set contains n instances and m features. In Fig. 1(a), the n instances are evenly partitioned to three segments $(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3)$ and each worker obtains one segment of the data. Given this data distribution, Chu et al. (2007) show that the covariance matrix $\mathbf{C}$ can be computed in parallel on the workers by first computing a local covariance matrix on each worker, and then aggregating the local covariance matrix on the master to obtain the global covariance matrix. After $\mathbf{C}$ is computed on the master, it is again evenly partitioned to three segments $(\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3)$, and each worker obtains one segment of the covariance matrix. $\mathbf{C}_i \in \mathbb{R}^{\frac{m}{3}, m}$, $i \in \{1, 2, 3\}$, and each row of $\mathbf{C}_i$ corresponds to one of the m features. After $\mathbf{C}$ is distributed, feature scores can be computed in parallel on the three workers in each SFS step. The computation involves partitioning the $\mathbf{C}_i$ on each worker into $\mathbf{B}_r$ and $\mathbf{D}_r$, constructing $\mathbf{A}^{-1}$ and $\mathbf{C}_{2,1}$, and applying matrix computation to calculate feature scores on each worker (see Fig. 1(b)). If the workers support SMP, matrix computation can be done in parallel on each worker through multithreading. Each worker computes the scores for $\frac{m}{3}$ features and sends these scores to the master, which selects the best feature in the current SFS step. Section 3.3.1 provides the details of this process.

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

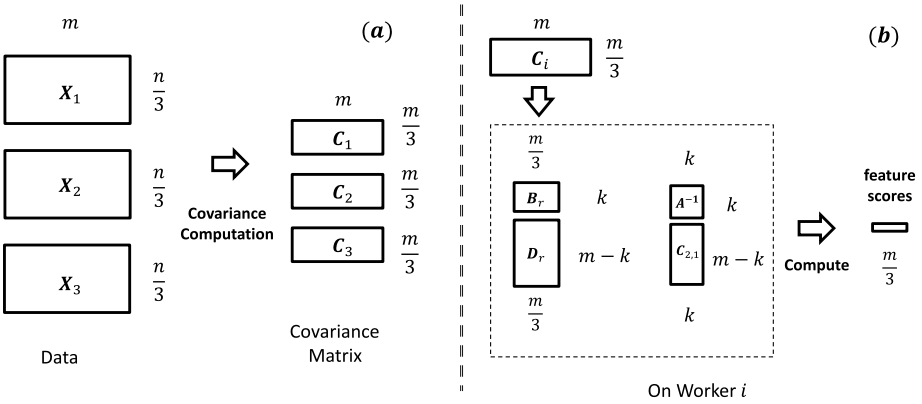

Fig. 1 Feature scores are computed in parallel on three workers for unsupervised feature selection

3.3 The implementations

Algorithms 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 workers. In the algorithms, $\otimes$ and $\oslash$ denote element-wise matrix multiplication and division, respectively.

3.3.1 Unsupervised feature selection

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 1 computes the covariance matrix, $\mathbf{C} \in \mathbb{R}^{m \times m}$. Given $\mathbf{X}_1, \ldots, \mathbf{X}_p$ located on p workers, the covariance matrix can be computed efficiently using mature distributed matrix-matrix multiplication techniques (Alonso et al. 2009). For brevity, the detail for the distributed covariance matrix computation is omitted. Assuming that grid nodes are homogeneous, given p nodes and on each node there is one worker, $\mathbf{C}$ is partitioned to p parts, $\mathbf{C} = (\mathbf{C}_1, \ldots, \mathbf{C}_p)$, and $\mathbf{C}_r \in \mathbb{R}^{m \times \frac{m}{p}}$ is stored on the rth node. Line 2 to Line 5 compute feature scores to select the first feature. Since no feature has been selected, Expression (11) 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. Let $\mathbf{C}_r$ contain the $i_{r,1}, \ldots, i_{r,\frac{m}{p}}$ columns of $\mathbf{C}$. In Line 2, $\mathbf{v}_r = (c_{i_{r,1},i_{r,1}}, \ldots, c_{i_{r,\frac{m}{p}},ir,\frac{m}{p}})$ contains the diagonal elements of $\mathbf{C}$ that corresponds to the variance of features from $F_{i_{r,1}}$ to $F_{i_{r,\frac{m}{p}}}$. The vector $\mathbf{s}_r$ contains the scores of features from $F_{i_{r,1}}$ to $F_{i_{r,\frac{m}{p}}}$. After a feature F_i has been selected, Line 8 broadcasts $\mathbf{c}^i$, since it is needed for updating $\mathbf{A}^{-1}$ and $\mathbf{C}_{2,1}$ on each 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. 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. Then $\mathbf{A}^{-1} = (\mathbf{X}_1^\top \mathbf{X}_1)^{-1} = \mathbf{C}_{\mathbb{L}\times\mathbb{L}}$ is a symmetric matrix that contains the covariance of the selected features, $\mathbf{B}_r = \mathbf{X}_1 \mathbf{X}_r = \mathbf{C}_{\mathbb{L}\times\mathbb{L}_r}$ contains the covariance between the selected features and the unselected features on the rth worker, $\mathbf{D}_r = \mathbf{X}_2^\top \mathbf{X}_r = \mathbf{C}_{\mathbb{L}_u \times \mathbb{L}_r}$ contains the covariance between all unselected features and the unselected features on the rth worker, $\mathbf{v}_r$ contains the variance of the unselected features on the rth worker; and $\mathbf{C}_{2,1} =$

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 the rth section of the covariance matrix, $\mathbf{C}_r \in \mathbb{R}^{m \times \frac{m}{p}}$, on the rth worker, $r = 1, \ldots, p$;
2 Compute local feature scores on each **worker**

$$\mathbf{s}_r = \mathbf{1}^\top (\mathbf{C}_r \otimes \mathbf{C}_r), \qquad \mathbf{s}_r = \mathbf{s}_r \oslash \mathbf{v}_r; \tag{23}$$

3 **Workers** send $\mathbf{s}_r$ to the master via MPI_Gather;
4 On the **master**, select $i = \arg\max(s_i \mid s_i \in (\mathbf{s}_1, \ldots, \mathbf{s}_p))$;
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 ith column of $\mathbf{C}$, sends $\mathbf{c}^i$ to all other workers via MPI_Bcast;
/* ---------- */
/* simultaneously, all workers do */
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

$$\mathbf{E}_r = \mathbf{A}^{-1}\mathbf{B}_r, \qquad \mathbf{H}_r = \mathbf{C}_{2,1}\mathbf{E}_r, \qquad \mathbf{G}_r = \mathbf{D}_r - \mathbf{H}_r, \tag{24}$$

$$\mathbf{g}_r = \mathbf{1}^\top (\mathbf{G}_r \otimes \mathbf{G}_r), \qquad \mathbf{w}_r = \mathbf{v}_r - \mathbf{1}^\top (\mathbf{B}_r \otimes \mathbf{E}_r), \tag{25}$$

$$\mathbf{s}_r = \mathbf{g}_r \oslash \mathbf{w}_r \tag{26}$$

11 **Workers** send $\mathbf{s}_r$ to the master via MPI_Gather;
/* ---------- */
12 On the **master**, select $i = \arg\max(s_i \mid s_i \in (\mathbf{s}_1, \ldots, \mathbf{s}_p))$;
13 On the **master**, $\mathbb{L} = \mathbb{L} \cup \{F_i\}$, $l++$;
14 **end**

Algorithm 1: Distributed parallel unsupervised feature selection

$\mathbf{X}_2^\top \mathbf{X}_1$ contains the covariance between all selected and unselected features. The scores of the features on the rth worker can be computed using the equations specified in Line 10. The master selects the feature with the maximum score in Line 12 and updates the list $\mathbb{L}$ accordingly in Line 13. The matrix $\mathbf{A}^{-1}$ in Line 9 can be computed by applying rank-one update using Equation (16).

Let $CPU(\cdot)$ and $NET(\cdot)$ denote the time used for computation and for network communication, respectively. Assume that a tree-based mechanism is used to develop the collective operations, such as `MPI_Bcast` and `MPI_Reduce`, in the MPI implementation. The time complexity for computing and distributing the covariance matrix $\mathbf{C}$ is

$$CPU\left(\frac{m^2 n}{p} + m^2 \log p\right) + NET\left(m^2 \log p\right) \tag{27}$$

After obtained $\mathbf{C}$, time complexity of selecting k features using Algorithm 1 is

$$CPU\left(\frac{m^2 k^2}{p}\right) + NET(mk) \tag{28}$$

Therefore, the total time complexity of Algorithm 1 is

$$CPU\left(\frac{m^2 (n + k^2)}{p} + m^2 \log p\right) + NET\left(m^2 \log p\right) \tag{29}$$

3.3.2 Supervised feature selection

Algorithm 2 performs 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 (21) simplifies to $\frac{\|\mathbf{Y}^\top \mathbf{f}\|_2^2}{\mathbf{f}^\top \mathbf{f}}$. Line 1 computes the local feature-response covariance $\mathbf{E}_r$ and the local feature variance $\mathbf{v}_r$ on p workers, which are then sent to the master to compute the global $\mathbf{E}$ and $\mathbf{v}$ using `MPI_REDUCE(MPI_SUM)`. $\mathbf{E}$ and $\mathbf{v}$ can be computed in this way, since

$$\mathbf{Y}^\top \mathbf{X} = \left(\mathbf{Y}_1^\top, \ldots, \mathbf{Y}_p^\top\right) \begin{pmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_p \end{pmatrix} = \sum_{r=1}^{p} \mathbf{Y}_r^\top \mathbf{X}_r \tag{30}$$

After $\mathbf{E}$ and $\mathbf{v}$ are obtained, feature scores are computed in Line 3 and a feature is selected in Line 4. Let F_i be the selected feature, which has been partitioned into p segments and stored on p nodes. Line 7 to Line 9 compute the covariance between F_i and all other features,

$$\mathbf{c}^i = \mathbf{X}^\top \mathbf{f}_i = \left(\mathbf{X}_1^\top, \ldots, \mathbf{X}_p^\top\right) \begin{pmatrix} \mathbf{f}_1^i \\ \vdots \\ \mathbf{f}_p^i \end{pmatrix} = \sum_{r=1}^{p} \mathbf{c}_r^i \tag{31}$$

Line 10 constructs $\mathbf{A}^{-1} = (\mathbf{X}_1^\top \mathbf{X}_1)^{-1}$, $\mathbf{C}_{\mathbf{Y},1} = \mathbf{Y}^\top \mathbf{X}_1$, $\mathbf{C}_{\mathbf{Y},2} = \mathbf{Y}^\top \mathbf{X}_2$, $\mathbf{C}_{1,2} = \mathbf{X}_1^\top \mathbf{X}_2$, and $\mathbf{v}_2$. Here $\mathbf{v}_2$ contains the variance of the unselected features, and the $\mathbf{v}$ obtained in Line 2 can be used to construct it. The $\mathbf{c}^i$ obtained in Line 9 can be used to construct $\mathbf{A}^{-1}$ and $\mathbf{C}_{1,2}$ incrementally from their former versions, and the $\mathbf{E}$ obtained in Line 2 can be used to construct $\mathbf{C}_{\mathbf{Y},1}$ and $\mathbf{C}_{\mathbf{Y},2}$ incrementally from their previous versions, too. After these components are obtained, Line 11 to Line 14 compute feature scores and select a feature with the highest score. The process (Line 7 to Line 15) is repeated until k features have been selected.

Because both $\mathbf{A}^{-1}$ and $\mathbf{B}$ can be obtained by incrementally updating their previous versions, the time complexity for selecting k features using Algorithm 2 is

$$CPU\left(mk\left(\frac{n}{p} + k^2\right)\right) + NET\left(m(C+k)\log p\right) \tag{32}$$

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

Expression (29) and Expression (32) suggest that when the number of instances is large and the network is fast enough, Algorithm 1 and Algorithm 2 can speed up feature selection linearly as the number of available workers increases.

4 Connections to existing methods

4.1 Unsupervised feature selection

In an unsupervised setting, principal component analysis (PCA) (Jolliffe 2002) also reduces dimensionality by preserving data variance. The key difference between PCA and the proposed method is that PCA is for feature extraction (Liu and Motoda 1998a; Lee and Seung 1999; Saul et al. 2006), which reduce dimensionality via generating a small set of new features by linearly combining the original features, while the proposed method

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

On each **worker**, compute $\mathbf{E}_r \in \mathbb{R}^{C \times m}$, $\mathbf{v}_r \in \mathbb{R}^{1 \times m}$:

$$\mathbf{E}_r = \mathbf{Y}_r^\top \mathbf{X}_r, \qquad \mathbf{v}_r = \mathbf{1}^\top (\mathbf{X}_r \otimes \mathbf{X}_r); \tag{33}$$

Send $\mathbf{E}_r$ and $\mathbf{v}_r$ to the master via MPI_Reduce with MPI_SUM option. On the **master**, obtain $\mathbf{E}$ and $\mathbf{v}$:

$$\mathbf{E} = \sum_{r=1}^{p} \mathbf{E}_r, \qquad \mathbf{v} = \sum_{r=1}^{p} \mathbf{v}_r; \tag{34}$$

On the **master**, compute feature scores

$$\mathbf{s} = \mathbf{1}^\top (\mathbf{E} \otimes \mathbf{E}), \qquad \mathbf{s} = \mathbf{s} \oslash \mathbf{v}; \tag{35}$$

On the **master**, select $i = \arg\max(s_i \mid s_i \in \mathbf{s})$;
Initialization, $\mathbb{L} = \{F_i\}$, $l = 1$;
while $l < k$ **do**
    The **master** sends $\mathbb{L}$ to all workers via MPI_Bcast;
    Workers compute $\mathbf{c}_r^i = \mathbf{X}_r^\top \mathbf{f}_r^i$, $\mathbf{c}_r^i \in \mathbb{R}^{m \times 1}$;
    Workers send $\mathbf{c}_r^i$ to the master via MPI_Reduce with MPI_SUM option. The **master** obtains $\mathbf{c}^i$,

$$\mathbf{c}^i = \sum_{r=1}^{p} \mathbf{c}_r^i, \quad \mathbf{c}^i \in \mathbb{R}^{m \times 1}; \tag{36}$$

    On the **master**, construct

$$\mathbf{A}^{-1} \in \mathbb{R}^{l \times l}, \qquad \mathbf{C}_{\mathbf{Y},1} \in \mathbb{R}^{C \times l}, \qquad \mathbf{C}_{1,2} \in \mathbb{R}^{l \times (m-l)},$$
$$\mathbf{C}_{\mathbf{Y},2} \in \mathbb{R}^{C \times (m-l)}, \qquad \mathbf{v}_2 \in \mathbb{R}^{1 \times (m=1)};$$

    On the **master**, compute

$$\mathbf{B} = \mathbf{A}^{-1} \mathbf{C}_{1,2}, \qquad \mathbf{B} \in \mathbb{R}^{l \times (m-l)}; \tag{37}$$

    On the **master**, compute

$$\mathbf{H} = \mathbf{C}_{\mathbf{Y},1} \mathbf{B}, \qquad \mathbf{G} = \mathbf{C}_{\mathbf{Y},2} - \mathbf{H}, \qquad \mathbf{g} = \mathbf{1}^\top (\mathbf{G} \otimes \mathbf{G}); \tag{38}$$

    On the **master**, compute

$$\mathbf{w} = \mathbf{v}_2 - \mathbf{1}^\top (\mathbf{C}_{1,2} \otimes \mathbf{B}), \qquad \mathbf{s} = \mathbf{g} \oslash \mathbf{w}; \tag{39}$$

    On the **master**, select $i = \arg\max(s_i \mid s_i \in \mathbf{s})$;
    On the **master**, $\mathbb{L} = \mathbb{L} \cup \{F_i\}$, $l++$;
end

Algorithm 2: Distributed parallel supervised feature selection

is for feature selection, which reduce dimensionality by selecting a small set of the original features. The features returned by the proposed method are the original ones. And this is very important in applications where retaining the original features is useful for model exploration or interpretation (for example, in genetic analysis and text mining).

Sparse principal component analysis (SPCA) (Zou et al. 2004; d'Aspremont et al. 2007; Zhang and d' Aspremont 2011) has been studied in recent years to improve the interpretability of PCA. The principal components generated by SPCA are sparse, i.e., only a few fea-

tures are assigned nonzero weights in each of the principal components computed by SPCA. However, different sparse principal components may have different sparsity patterns. When multiple sparse principal components are considered together, there may still be many features assigned nonzero weights. And it is not straightforward to precisely control the number of selected features in SPCA. Compared to SPCA, the proposed method can precisely control it. Also, since the optimization technique utilized in the proposed method is simple, it is easy to distribute and parallelize it for achieving better efficiency and scalability.

4.2 Supervised feature selection, regression

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

$$\mathbf{f}^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top\right) \mathbf{Y} = \mathbf{f}^\top (\mathbf{Y} - \mathbf{X}_1 \mathbf{W}_1) \tag{40}$$

where $\mathbf{W}_1 = (\mathbf{X}_1^\top \mathbf{X}_1)^{-1} \mathbf{X}_1^\top \mathbf{Y}$ is the solution of a least squares regression based on $\mathbf{X}_1$. Let $\mathbf{R}$ be the residual, $\mathbf{R} = \mathbf{Y} - \mathbf{X}_1 \mathbf{W}_1$. Expression (21) can be simplified to

$$\arg\max_{\mathbf{f}} \frac{\|\mathbf{f}^\top \mathbf{R}\|_2^2}{\|(\mathbf{I} - \mathbf{X}_1 (\mathbf{X}_1^\top \mathbf{X}_1)^{-1} \mathbf{X}_1^\top)^{\frac{1}{2}} \mathbf{f}\|_2^2} \tag{41}$$

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 method forms a special type of stepwise regression with Expression (21) as the selection criterion.

4.3 Supervised feature selection, classification

When used in a classification setting, the proposed method selects features by maximizing 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 Automatically determine *k*

In Algorithms 1 and 2, k is the number of features to select. However, in real applications this number might not always be known. Determine how many features to select is an important research problem in feature selection. In a supervised learning setting, some very effective model selection techniques can be conveniently used in the proposed algorithm to automatically determine the number of features to select. These techniques include Akaike's information criterion (AIC), small-sample-size corrected version of AIC (AICC) (Sugiura 1976), Bayesian information criterion (BIC), and corrected Hannan-Quinn information criterion (HQC). Assume that the model errors are normally and independently distributed. Also assume that when k features are selected, the sum of squared errors of the model is sse_k. Let C be the number of the columns in the response matrix $\mathbf{Y}$. Al-Subaihi (2002) shows that for multivariate linear regression the, AIC, AICC, BIC, and HQC can be computed as

$$AIC_k = \log(sse_k) + \frac{2kC + (C+1)C}{n} \tag{42}$$

$$AICC_k = \log(sse_k) + \frac{(n+k)C}{n-k-C-1} \tag{43}$$

$$BIC_k = \log(sse_k) + \frac{k\log(n)}{n} \tag{44}$$

$$HQC_k = \log(sse_k) + \frac{2\log(\log(n))kC}{n-k-C-1} \tag{45}$$

The preceding equations suggest that computing sse_k plays a central role in estimating AIC, AICC, BIC and HQC. The following theorem shows that sse_k can be computed conveniently by using the intermediate result that is generated by the proposed algorithm.

Theorem 4 *Let $\mathbf{X}_k$ be the data set that contain the k selected features. Also, let sse_k be the sum of squared errors that are achieved by applying regression on $\mathbf{X}_k$. Assume that in step $k+1$, the proposed algorithm selects $\mathbf{f}^*$ and its feature score is s^*_{k+1}. The sum of squared errors achieved by applying regression on the data set $(\mathbf{X}_k, \mathbf{f}^*)$ can be computed as*

$$sse_{k+1} = sse_k - s^*_{k+1} \tag{46}$$

$$\text{s.t.} \quad s^*_{k+1} = \arg\max_{\mathbf{f}^*} \frac{\|\mathbf{Y}^\top(\mathbf{I} - \mathbf{X}_1(\mathbf{X}_1^\top\mathbf{X}_1)^{-1}\mathbf{X}_1^\top)\mathbf{f}^*\|_2^2}{\|(\mathbf{I} - \mathbf{X}_1(\mathbf{X}_1^\top\mathbf{X}_1)^{-1}\mathbf{X}_1^\top)^{\frac{1}{2}}\mathbf{f}^*\|_2^2}$$

Proof Let $\mathbf{Y}$ be the target matrix. The closed form solution of a linear least square regression is $\mathbf{W}_k = (\mathbf{X}_k\mathbf{X}_k^\top)^{-1}\mathbf{X}_k^\top\mathbf{Y}$, and the residual matrix is $\mathbf{R} = \mathbf{Y} - \mathbf{X}_k\mathbf{W}_k$. The sum of squared errors of applying regression on $\mathbf{X}_k$ can be computed as

$$\begin{aligned} sse_k &= \text{Trace}\big(\mathbf{R}^\top\mathbf{R}\big) \\ &= \text{Trace}\big(\big(\mathbf{Y} - \mathbf{X}_k\big(\mathbf{X}_k^\top\mathbf{X}_k\big)^{-1}\mathbf{X}_k^\top\mathbf{Y}\big)^\top\big(\mathbf{Y} - \mathbf{X}_k\big(\mathbf{X}_k^\top\mathbf{X}_k\big)^{-1}\mathbf{X}_k^\top\mathbf{Y}\big)\big) \\ &= \text{Trace}\big(\mathbf{Y}^\top\big(\mathbf{I} - \mathbf{X}_k\big(\mathbf{X}_k^\top\mathbf{X}_k\big)^{-1}\mathbf{X}_k^\top\big)\mathbf{Y}\big) \end{aligned} \tag{47}$$

Similar to (17), it can be verified that

$$sse_k - sse_{k+1} = s^*_{k+1} = \frac{\|\mathbf{Y}^\top(\mathbf{I} - \mathbf{X}_k(\mathbf{X}_k^\top\mathbf{X}_k)^{-1}\mathbf{X}_k^\top)\mathbf{f}^*\|_2^2}{\|(\mathbf{I} - \mathbf{X}_k(\mathbf{X}_k^\top\mathbf{X}_k)^{-1}\mathbf{X}_k^\top)^{\frac{1}{2}}\mathbf{f}^*\|_2^2} \tag{48}$$

When no feature is selected, it is easy to verify that $sse_0 = \text{Trace}(\mathbf{Y}^\top\mathbf{Y})$. □

The preceding theorem shows that in each SFS step the sum of squared errors of the current step can be computed incrementally by deducting the score of the selected feature from the sum of squared errors of the previous step. The score of features is an intermediate result for feature selection and has already been computed by the proposed algorithm. Therefore, computing the sum of squared errors in each SFS step does not incur additional computational complexity.

6 Experimental study

The proposed method was implemented as the HPREDUCE procedure in the SAS High-Performance Analytics server. This section evaluates its performance for both supervised and unsupervised feature selection.

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	EPSILON	2,000	900,000	2
PIX	10,000	100	10	OCR	1,156	5,670,000	2

6.1 Experiment setup

In the experiment, 12 representative feature selection algorithms are used for comparison. For unsupervised feature selection, six algorithms are selected as baselines: Laplacian score (He et al. 2005), SPEC-1 and SPEC-3 (Zhao and Liu 2007), trace-ratio (Nie et al. 2008), HSIC (Song et al. 2007), and SPFS (Zhao et al. 2011). For supervised feature selection, in the classification setting, seven algorithms are compared: ReliefF (Sikonja and Kononenko 2003), Fisher Score (Duda et al. 2001), trace-ratio, HSIC, mRMR (Ding and Peng 2003), AROM-SVM (Weston et al. 2003), and SPFS. In the regression setting, LARS (Efron et al. 2004), and LASSO (Tibshirani 1994) are compared. Among the 12 baseline feature selection algorithms, AROM-SVM, mRMR, SPFS, LARS, and LASSO can handle redundant features.

Ten benchmark data sets are used in the experiment. Four are face image data: AR,[3] PIE,[4] PIX,[5] and ORL[6] (images from 10 persons are used). Two are text data extracted from the 20-newsgroups data:[7] 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).[8] And two are large-scale data sets from the Pascal large scale learning challenge[9] for performance tests. Compared to the u10mf5k and s25mf5k data sets used in Zhao et al. (2012), the EPSILON and OCR data sets are dense, therefore their size (#features×#instances) provides a more precise view on the amount of computation involved in the feature selection process.

Among the ten data sets used in the experiment, the first eight data sets are small-scale. They are used to compare the performance of the HPREDUCE procedure to existing feature selection algorithms, since the implementations of the existing algorithms cannot handle large scale problems. The last two data sets are large-scale and are used to evaluate the scalability of the HPREDUCE procedure in a distributed computing environment. Among the eight small-scale data sets used for comparison, the first six data sets are used to test unsupervised feature selection and supervised feature selection for classification. And the seventh and the eighth data sets are used to test feature selection for regression. Details on the ten data sets can be found in Table 1.

[3] http://www2.ece.ohio-state.edu/~aleix/ARdatabase.html.

[4] http://peipa.essex.ac.uk/ipa/pix/faces/manchester/.

[5] http://www.ri.cmu.edu/projects/project_418.html.

[6] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

[7] http://qwone.com/~jason/20Newsgroups/.

[8] http://archive.ics.uci.edu/ml/index.html.

[9] http://largescale.ml.tu-berlin.de.

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

$$RED(\mathbb{L}) = \frac{1}{m(m-1)} \sum_{F_i, F_j \in \mathbb{L}, i>j} |\rho_{i,j}| \tag{49}$$

where $|\rho_{i,j}|$ returns the absolute value of the correlation between features F_i and F_j. Equation (49) 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 $\mathbb{L}$,

$$PCT_{VAR}(\mathbb{L}) = \frac{\text{Trace}(\mathbf{X}^\top \mathbf{X}_{\mathbb{L}} (\mathbf{X}_{\mathbb{L}}^\top \mathbf{X}_{\mathbb{L}})^{-1} \mathbf{X}_{\mathbb{L}}^\top \mathbf{X})}{\text{Trace}(\mathbf{X}^\top \mathbf{X})} \tag{50}$$

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, a 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 features are selected.

6.2 Study of unsupervised cases

Percentage of explained variance Figure 2 shows the percentage of explained variance of algorithms when different numbers of features are selected. Table 2 presents the average results which are computed by averaging the results obtained when different numbers of features are selected. The results show 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 (11) with the sequential forward selection strategy is effective for minimizing Expression (1).

Redundancy rate Table 3 presents the average redundancy rates achieved by algorithms, which is computed by averaging the results obtained when different numbers of features are selected. It shows that SPFS and the HPREDUCE procedure achieved much better results than other algorithms. This is also to be expected, since they are designed to handle redundant features, while the others are not.

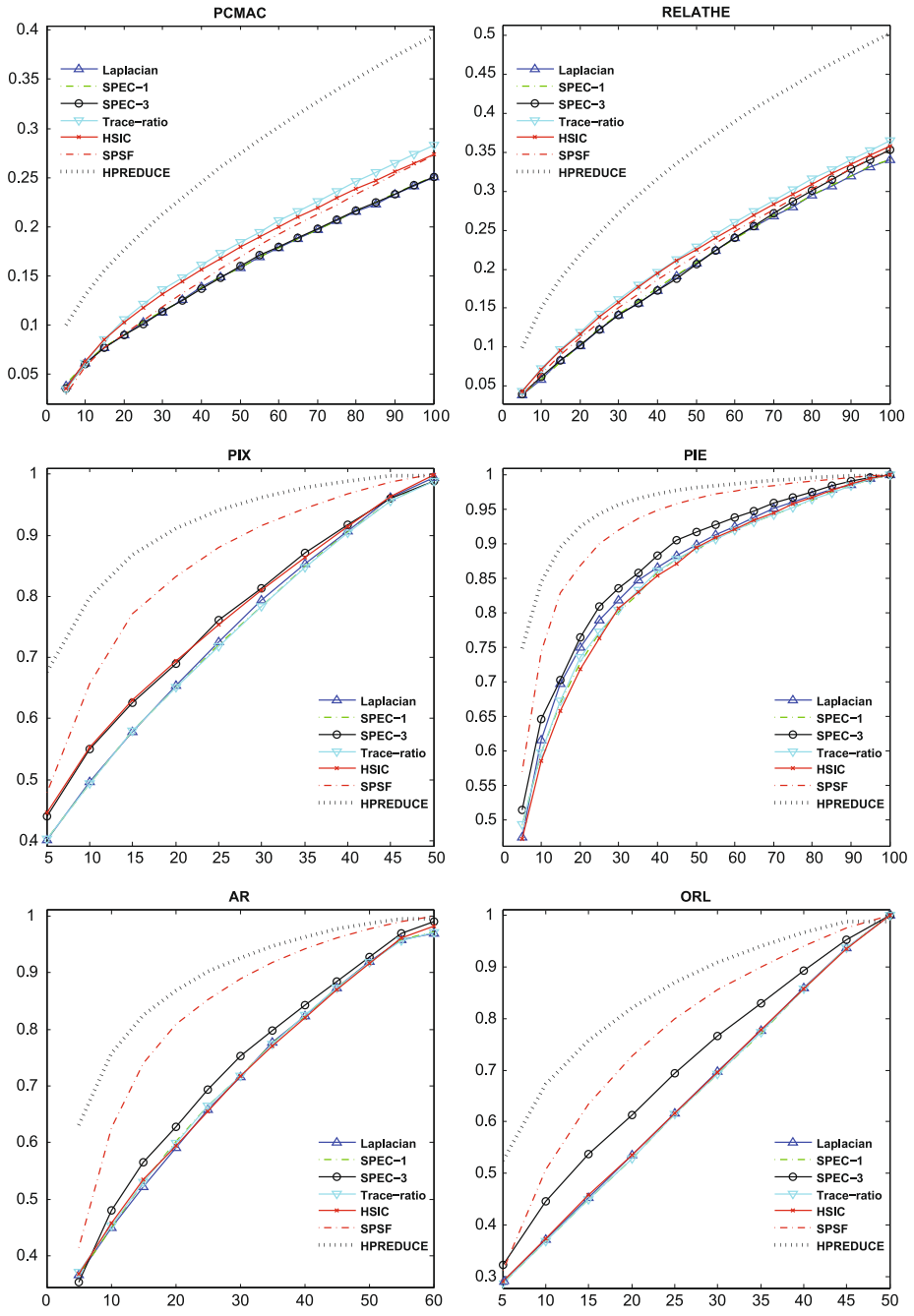

Fig. 2 Unsupervised feature selection: explained variance achieved by algorithms when different numbers of features are selected. In the plots, the x-axis corresponds to the number of selected features, and the y-axis corresponds to the explained variance (higher is better)

Table 2 Unsupervised feature selection: average explained variance achieved by algorithms (higher is better). The number in parentheses is the p-value that is computed using the Student's t-test by comparing each algorithm to the one with the highest explained variance. Bold font indicates the explained variance that is the highest in each column or is not significantly different to the highest one according to p-value > 0.05

Algorithm	PCMAC	RELATH	PIX	PIE	AR	ORL	AVE	Best
Laplacian	0.16 (0.0)	0.21 (0.0)	0.74 (0.0)	0.86 (0.0)	0.72 (0.0)	0.65 (0.0)	0.557	0
SPEC-1	0.16 (0.0)	0.21 (0.0)	0.73 (0.0)	0.85 (0.0)	0.72 (0.0)	0.65 (0.0)	0.553	0
SPEC-3	0.16 (0.0)	0.21 (0.0)	0.76 (0.0)	0.88 (0.0)	0.74 (0.0)	0.71 (0.0)	0.577	0
Trace-ratio	0.18 (0.0)	0.23 (0.0)	0.73 (0.0)	0.86 (0.0)	0.72 (0.0)	0.65 (0.0)	0.562	0
HSIC	0.18 (0.0)	0.22 (0.0)	0.76 (0.0)	0.85 (0.0)	0.72 (0.0)	0.65 (0.0)	0.563	0
SPFS	0.17 (0.0)	0.22 (0.0)	0.84 (.01)	0.93 (0.0)	0.84 (.01)	0.77 (.01)	0.628	0
HPREDUCE	**0.27**	**0.34**	**0.91**	**0.96**	**0.90**	**0.84**	**0.703**	**6**

Table 3 Unsupervised feature selection: redundancy rates achieved by algorithms (lower is better). The number in parentheses is the p-value that is computed using the Student's t-test by comparing each algorithm to the one with the lowest redundancy rate. Bold font indicates the redundancy rates that are the lowest in each column or are not significant different from the lowest one according to p-value > 0.05

Algorithm	PCMAC	RELATH	PIX	PIE	AR	ORL	AVE	Best
Laplacian	0.23 (0.0)	0.28 (0.0)	0.93 (0.0)	0.85 (0.0)	0.82 (0.0)	0.86 (0.0)	0.662	0
SPEC-1	0.23 (0.0)	0.28 (0.0)	0.93 (0.0)	0.88 (0.0)	0.81 (0.0)	0.87 (0.0)	0.667	0
SPEC-3	0.28 (0.0)	0.37 (0.0)	0.93 (0.0)	0.80 (0.0)	0.77 (0.0)	0.73 (0.0)	0.647	0
Trace-ratio	0.13 (0.0)	0.21 (0.0)	0.93 (0.0)	0.88 (0.0)	0.81 (0.0)	0.87 (0.0)	0.638	0
HSIC	0.11 (0.0)	0.21 (0.0)	0.93 (0.0)	0.83 (0.0)	0.80 (0.0)	0.87 (0.0)	0.625	0
SPFS	0.08 (0.0)	0.11 (0.0)	0.36 (0.0)	**0.33**	**0.25**	0.27 (0.0)	0.233	2
HPREDUCE	**0.02**	**0.03**	**0.25**	0.35 (0.0)	0.28 (0.0)	**0.23**	**0.193**	**4**

6.3 Study of supervised cases

Classification, accuracy Figure 3 shows the accuracy achieved by SVM using different numbers of features selected by algorithms. Table 4 presents the average results which are computed by averaging the accuracy obtained when different numbers of features are selected. The last column of Table 4 shows that 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.

Classification, redundancy rate The average redundancy rates 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.

Regression In the regression setting, the HPREDUCE procedure is compared to LARS and LASSO. The average RMSE and average redundancy rate results are presented in Ta-

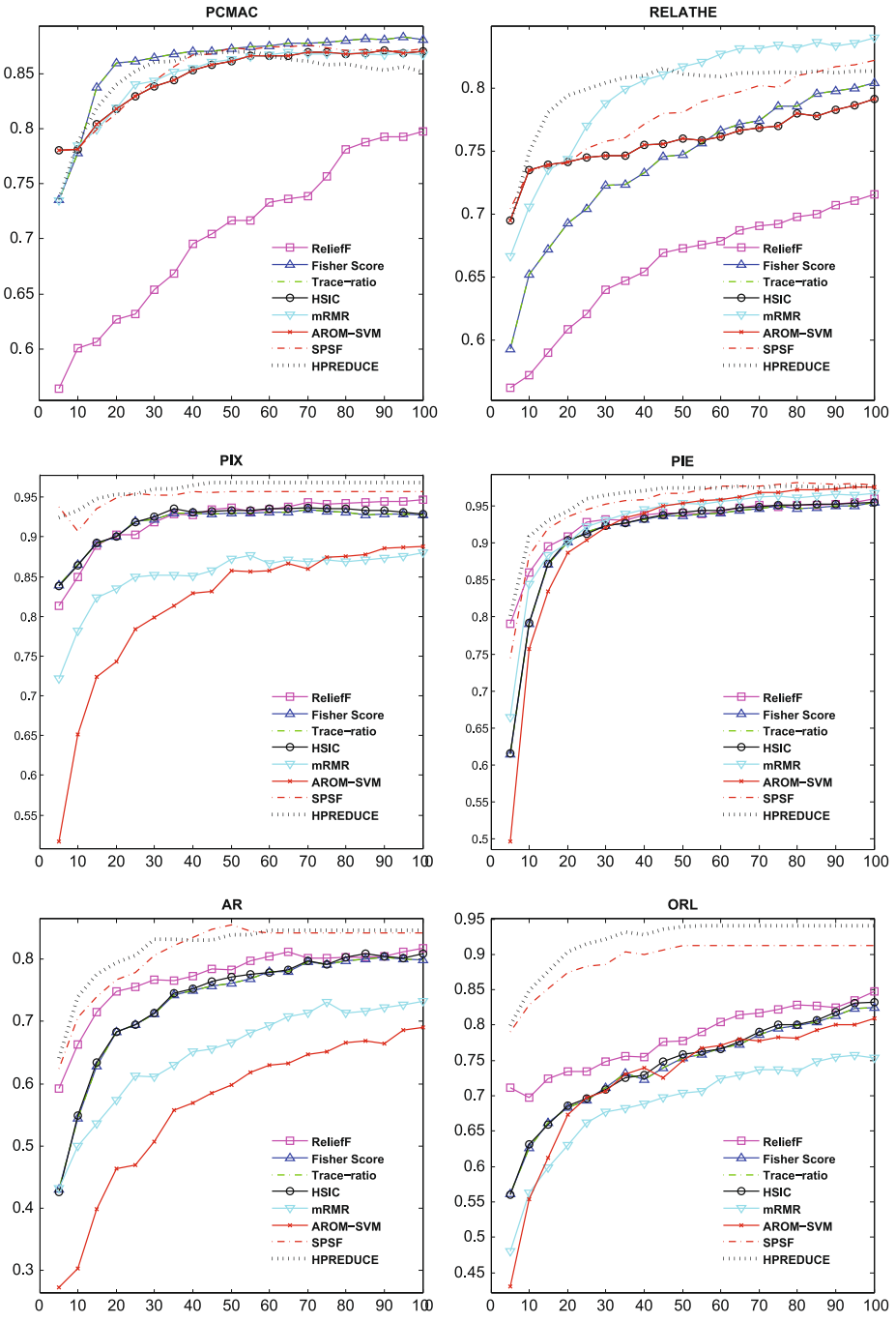

Fig. 3 Supervised feature selection for classification: accuracy achieved by algorithms when different numbers of features are selected. In the plots, the x-axis corresponds to the number of selected features, and the y-axis corresponds to the accuracy (higher is better)

Table 4 Supervised feature selection for classification: average accuracy achieved by algorithms (higher is better). The number in parentheses is the p-value that is computed using the Student's t-test by comparing each algorithm to the one with the highest average accuracy. Bold font indicates the accuracy that is the highest in each column or is not significantly different from the highest one according to p-value > 0.05

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**	0.73 (.00)	0.92 (.00)	0.90 (.01)	0.72 (.00)	0.73 (.00)	0.810	1
Trace-ratio	**0.86**	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**	**0.96**	**0.95**	**0.81**	**0.92**	**0.880**	5

Table 5 Supervised feature selection for classification: redundancy rates achieved by algorithms (lower is better). The number in parentheses is the p-value that is computed using the Student's t-test by comparing each algorithm to the one with the lowest redundancy rate. Bold font indicates redundancy rates that are the lowest in each column or are not significantly different from the lowest one according to p-value > 0.05

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**	0.04 (.00)	0.33 (.00)	**0.26** (.46)	**0.25**	**0.25**	**0.194**	4
Arom-SVM	0.05 (.00)	0.07 (.00)	**0.26**	0.29 (.02)	**0.25** (.22)	**0.25** (.35)	0.196	3
SPFS	0.11 (.00)	0.07 (.00)	0.45 (.00)	**0.25**	0.31 (.03)	0.36 (.00)	0.260	1
HPREDUCE	0.05 (.00)	**0.03**	0.32 (.00)	0.31 (.00)	0.31 (.00)	0.27 (.00)	0.214	1

ble 6. 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.

6.4 Study of model selection criteria

Table 7 shows the results of using different model selection criteria to determine how many features to select. In the experiment, PCMAC and RELATHE are used for classification, and CRIME and SLICELOC are used for regression. The four image data sets are not used, because compared to the number of features their sample sizes are too small for the four model selection criteria to provide reliable estimation[10] (Yang and Barron 1998; Casella et al. 2009). For each data set, the four model selection criteria are used to determine the number of features to select on each of its 20 partitions. The selected features are then used

[10] As a distributed parallel feature selection algorithm, the HPREDUCE procedure is usually used to handle large scale problems with huge sample size.

Table 6 Supervised feature selection for regression, RMSE (col 2–col 4) and redundancy rate (column 5–column 7) achieved by algorithms (both lower is better). The number in parentheses is the p-value that computed using the Student's t-test by comparing each algorithm to the one with the lowest RMSE or redundancy rate. Bold font indicates the RMSE or redundancy rates that are the lowest in each row or are not significantly different from the one according to p-value > 0.05

DATA	RMSE			Redundancy rate		
	LARS	LASSO	HPREDUCE	LARS	LASSO	HPREDUCE
CRIME	**1.91e–2**	**1.91e–2**	1.94e–2 (.00)	**0.23** (0.46)	**0.23** (0.46)	**0.22**
SLICELOC	2.9e–3 (.00)	2.9e–3 (.00)	**2.6e–3**	0.19 (.00)	0.19 (.00)	**0.14**
Average	**1.10e–2**	**1.10e–2**	**1.10e–2**	0.210	0.210	**0.180**
Best	1	1	1	1	1	2

Table 7 Model selection, automatically determine the number of features to select. In the classification case (PCMAC and RELATHE) the performance measurement is classification accuracy (higher is better). In the regression case (CRIME and SLICELOC) the performance measurement is RMSE (lower is better). The number in parentheses is the averaged number of selected features

DATA	AIC	AICC	BIC	HQC
PCMAC	0.81 (400)	0.82 (203)	0.83 (197)	**0.87** (**52**)
RELATHE	**0.82** (400)	0.81 (230)	**0.82** (399)	0.81 (**71**)
CRIME	1.94e–2 (50)	1.94e–2 (48)	**1.91e–2** (**12**)	**1.91e–2** (24)
SLICELOC	**2.30e–3** (240)	**2.30e–3** (240)	2.31e–3 (**167**)	**2.30e–3** (206)

in SVM and linear regression for computing classification accuracy and RMSE, respectively. The obtained results are averaged and reported in Table 7 .

AIC, AICC, BIC, and HQC all aim to minimize the combination of a goodness-of-fit measurement and a model complexity measurement. Compared to BIC, AIC tends to favor more complicated models (Wagenmakers and Farrel 2004). In the experiment, AIC selected more features on all benchmark data sets. In contrast, BIC selected fewer features but achieved higher accuracy and lower RMSE. AICC is the corrected AIC, which improves AIC when the sample size is small compared to the number of features. For PCMAC and RELATHE which contain more features than instances, AICC selected fewer features than AIC while achieved comparable accuracy and RMSE. For CRIME and SLICELOC which contain more instances than features, AICC and AIC act the same. HQC is similar to BIC, but its model complexity measurement also considers C, the number of columns in $\mathbf{Y}$.

Table 7 shows that in terms of classification accuracy and RMSE, HQC performed best on three of the four benchmark data sets. In terms of the number of selected features, both HQC and BIC selected the smallest sets on two of the four benchmark data sets. The results suggest that when used in the HPREDUCE procedure, HQC and BIC might be good model selection criteria for determining how many features to select. In practice, it can also be helpful to use multiple model selection criteria to select multiple feature sets and use domain knowledge to determine which set serves the analysis better.

6.5 Study of scalability

To evaluate the scalability of the HPREDUCE procedure, it was tested in a distributed computing environment. The cluster has 208 blades (nodes), and each blade has 16 GB memory

Table 8 The large-scale data sets used in the experiment

Data set	Features	Instances	Classes	SAS Data File Size
EPSILON-whole	2000	900,000	–	13.7 GB
EPSILON-labeled	2000	500,000	2	7.6 GB
OCR-whole	1156	5,670,000	–	49.4 GB
OCR-labeled	1156	3,500,000	2	30.5 GB

and two Intel L5420 Xeon CPUs (2.5 GHz). Since each L5420 CPU has 4 cores, there are a total of 8 cores on each node for processing concurrent jobs. In the experiment, there is one worker on each node, and each worker runs with 8 threads.

The EPSILON and the OCR data are downloaded from the website of the Pascal large scale learning challenge. The obtained data are converted and stored in SAS data format. Each dataset contains three parts: the training part, the validation part, and the test part. Only the training part of each data contains label information. The EPSILON-labeled and the OCR-labeled data sets are created from the training part of the EPSILON and the OCR data, respectively. And they are used for testing supervised feature selection. The EPSILON-whole and the OCR-whole data sets are created by combining the training, the validation, and the testing parts of the EPSILON and the OCR data, respectively. And they are used for testing unsupervised feature selection. Details on the four data sets can be found in Table 8. In the experiment, different numbers of nodes are used for selecting 200 features from the input data. Compared to the OCR data, the EPSILON data are smaller. Therefore, for the EPSILON data the maximum number of nodes is set to 50 ($50 \times 8 = 400$ cores), while for the OCR data, this number is increased to 200 ($200 \times 8 = 1600$ cores).

The running time[11] and the speedup results for both supervised and unsupervised feature selection on the EPSILON data as well as the OCR data are presented in Figs. 4 and 5, respectively. It shows that the HPREDUCE procedure generally performs faster when more computing resource is available. For example, on the OCR-whole data set, when only 10 worker nodes are used for computation in the unsupervised case, the HPREDUCE procedure finishes in 629.0 seconds. When 200 worker nodes are used, it finishes in just 38.1 seconds. On the EPSILON-labeled data set, when only 5 worker nodes are used for computation in the supervised case, the HPREDUCE procedure finishes in 370.2 seconds. When 50 worker nodes are used, it finishes in just 42.4 seconds. In general, for both supervised and unsupervised feature selection, the speedup of the HPREDUCE procedure is high. On the EPSILON data sets, when the number of worker nodes is less than 15, the speedup ratio (slope of the line) is close to 1. As the number of worker nodes increases, the speedup ratio decreases gradually. And when the number of worker nodes reaches 50, the speedup ratio is still about 0.9. Similarly, on the OCR data sets, when the number of worker nodes is less than 60, the speedup ratio of the HPREDUCE procedure is close to 1. As the number of worker nodes increases, the speedup ratio decreases gradually. When the number of worker nodes reaches 150, the speedup ratio is still about 0.9. And when the number of worker nodes is 200, the speedup ratio is about 0.83. For a fixed size problem, when more nodes are used, the warm-up and the communication costs

[11]The running time does not include the time for loading a data set and sending it to the cluster system. In the experiment, with an Ethernet of a link speed of 1 Gpbs, it takes about 130 seconds to load and send the EPSILON-whole data set. And for the OCR-whole data set, it is about 470 seconds.

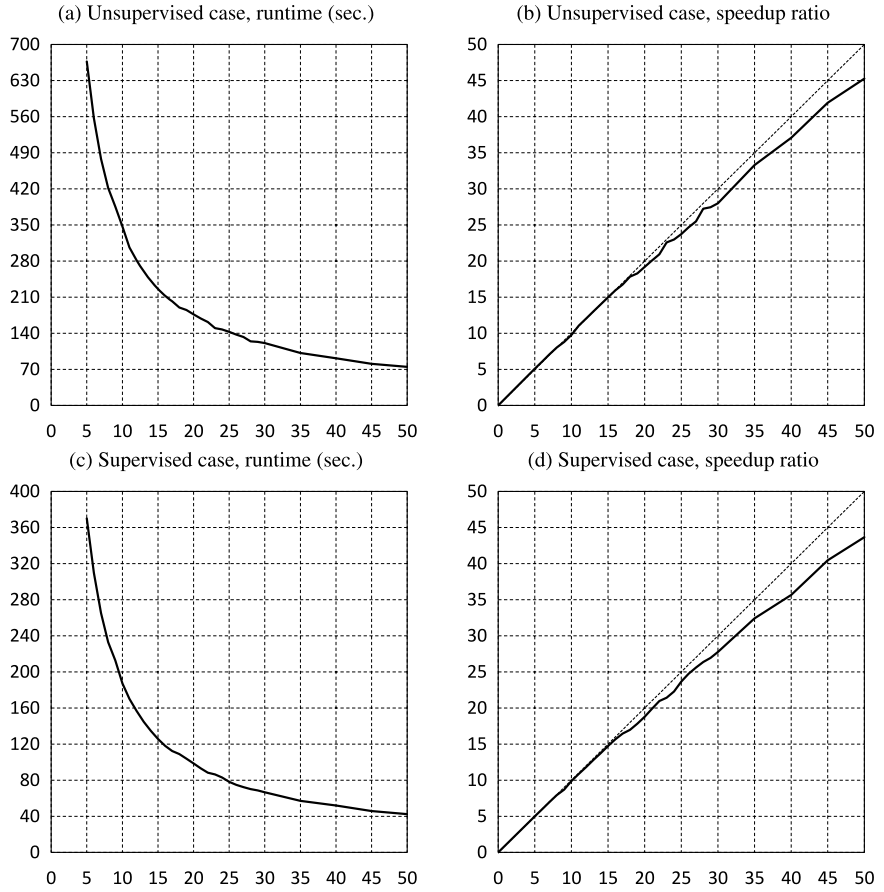

Fig. 4 EPSILON data sets: runtime and speedup of the HPREDUCE procedure in the unsupervised and the supervised settings when different numbers of workers are used for feature selection. The result for the unsupervised setting is obtained using the EPSILON-whole data set, and the result for the supervised setting is obtained using the EPSILON-labeled data set

start to offset the increase of computing resources, which is inevitable in distributed computing. This explains why the speedup ratio decreases when more worker nodes are used for computation. The results clearly demonstrate the scalability of the HPREDUCE procedure.

7 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. It can also automatically

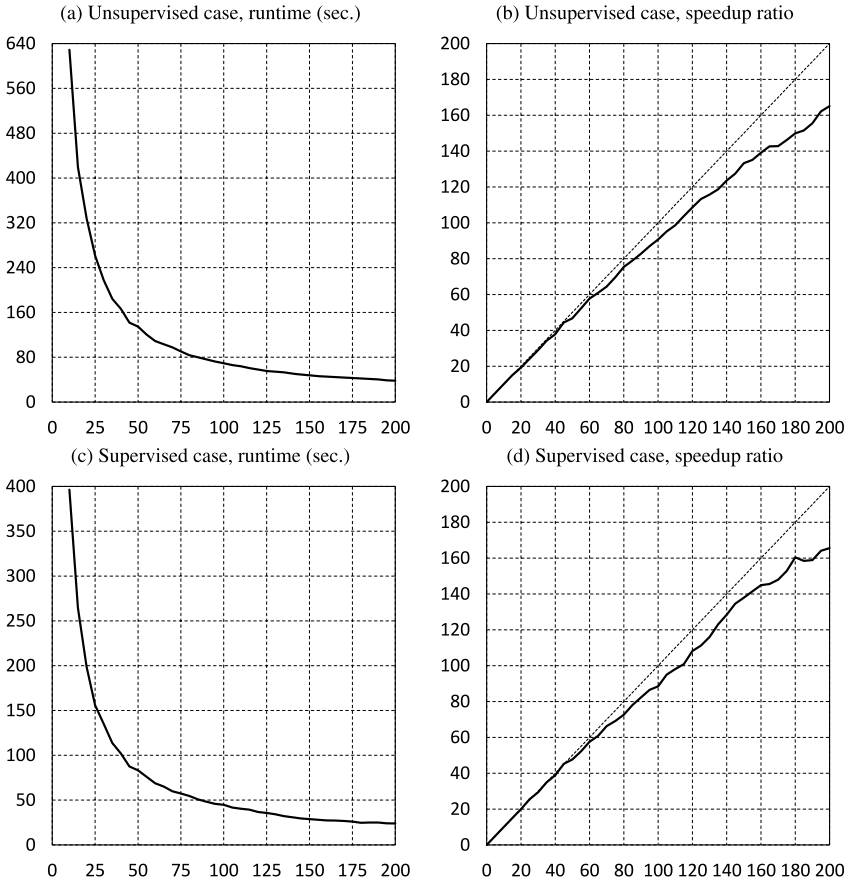

Fig. 5 OCR data sets: runtime and speedup of the HPREDUCE procedure in the unsupervised and the supervised settings when different numbers of workers are used for feature selection. The result for the unsupervised setting is obtained using the OCR-whole data set, and the result for the supervised setting is obtained using the OCR-labeled data set

determine the number of features to selected using effective model selection techniques for supervised learning. The computation of the algorithm is 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. We will also study how to automatically determine the number of features to select for unsupervised learning.

Acknowledgements 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

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, *19*(6), 716–723.

Al-Subaihi, A. A. (2002). Variable selection in multivariable regression using SAS/IML. *Journal of Statistical Software*. 7(12).

Alonso, P., Reddy, R., & Lastovetsky, A. (2009). Experimental study of six different parallel matrix multiplication applications for heterogeneous computational clusters of multicore processors. Tech. rep., UCD School of Computer Science and Informatics.

d'Aspremont, A., Ghaoui, L. E., Jordan, M., & Lanckriet, G. (2007). A direct formulation of sparse PCA using semidefinite programming. *SIAM Review*, *49*(3), 434–448.

Bradley, J., Kyrola, A., Bickson, D., & Guestrin, C. (2011). Parallel coordinate descent for L1-regularized loss minimization. In *Proceedings of international conference on machine learning (ICML'11)*.

Casella, G., Giron, F. J., Martinez, M. L., & Moreno, E. (2009). Consistency of Bayesian procedure for variable selection. *The Annals of Statistics*, *37*, 1207–1228.

Chu, C. T., Kim, S. K., Lin, Y. A., Yu, Y., Bradski, G., Ng, A., & Olukotun, K. (2007). Map-reduce for machine learning on multicore. In *Proceedings of neural information processing systems*.

Dash, M., Choi, K., Scheuermann, P., & Liu, H. (2002). Feature selection for clustering—a filter solution. In *Proceedings of international conference on data mining*.

Dean, J., & Ghemawat, S. (2010). System and method for efficient large-scale data processing.

Ding, C., & Peng, H. (2003). Minimum redundancy feature selection from microarray gene expression data. In *Proceedings of the computational systems bioinformatics* (pp. 523–529).

Duda, R., Hart, P., & Stork, D. (2001). *Pattern classification* (2nd ed.). New York: Wiley.

Dy, J. G., & Brodley, C. E. (2004). Feature selection for unsupervised learn. *Journal of Machine Learning Research*, *5*, 845–889.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, *32*, 407–449.

Fisher, R. (1936). The use of multiple measurements in taxonomic problems. *Annual of Eugenics*, *7*(2), 179–188.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, *3*, 1289–1305.

Garcia, D. J., Hall, L. O., Goldgof, D. B., & Kramer, K. (2006). A parallel feature selection algorithm from random subsets. In *Proceedings of the intl. workshop on parallel data mining*.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. New York: Freeman.

Golub, G. H., & Van Loan, C. F. (1996). *Matrix computations*. Baltimore: Johns Hopkins.

Guillen, A., Sorjamaa, A., Miche, Y., Lendasse, A., & Rojas, I. (2009). Efficient parallel feature selection for steganography problems. In *Bio-inspired systems: computational and ambient intelligence* (Vol. 5517, pp. 1224–1231).

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Hall, M. (1999). *Correlation-based feature selection for machine learning*. PhD thesis, University of Waikato, Dept. of Computer Science.

Hannan, E. J., & Quinn, B. G. (1979). The determination of the order of an autoregression. *Journal of the Royal Statistical Society. Series B. Methodological*, *41*, 190–195.

He, X., Cai, D., & Niyogi, P. (2005). Laplacian score for feature selection. In Y. Weiss, B. Schölkopf, & J. Platt (Eds.), *Advances in neural information processing systems* (Vol. 18).

Jolliffe, I. T. (2002). *Principal component analysis* (2nd ed.). Berlin: Springer.

Kent, P., & Schabenberger, O. (2011). SAS high performance computing: the future is not what it used to be. http://www.monash.com/uploads/sas_HPA_2011-Longer.pdf.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, *401*(6755), 788–791.

Liu, H. & Motoda, H. (Eds.) (1998a). *Feature extraction, construction and selection: a data mining perspective* (2nd edn.). Boston: Kluwer Academic.

Liu, H., & Motoda, H. (1998b). *Feature selection for knowledge discovery and data mining*. Boston: Kluwer Academic.

Lopez, F. G., Torres, M. G., Batista, B. M., Perez, J. A. M., & Moreno-Vega, J. M. (2006). Solving feature subset selection problem by a parallel scatter search. *European Journal of Operational Research*, *169*(2), 477–489.

Manikandan, S., & Rajamani, V. (2008). A mathematical approach for feature selection and image retrieval of ultra sound kidney image databases. *European Journal of Scientific Research*, *24*, 163–171.

Melab, N., Cahon, S., & Talbi, E. G. (2006). Grid computing for parallel bioinspired algorithms. *Journal of Parallel and Distributed Computing*, *66*(8), 1052–1061. Special Issue on Parallel.

Nie, F., Nie, F., Xiang, S., Jia, Y., Zhang, C., & Yan, S. (2008). Trace ratio criterion for feature selection. In *Proceedings of the 23rd national conference on artificial intelligence (AAAI)*.

Petersen, K. B., & Pedersen, M. S. (2008). The matrix cookbook.

Saeys, Y., Inza, I., & Larraaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, *23*(19), 2507–2517.

Saul, L., Weinberger, K. Q., Sha, F., Ham, J., & Lee, D. D. (2006). *Spectral methods for dimensionality reduction* (pp. 279–293). Cambridge: MIT Press. Chap. 16.

Schwarz, G. E. (1978). Estimating the dimension of a model. *The Annals of Statistics*, *6*(2), 461–464.

Sikonja, M. R., & Kononenko, I. (2003). Theoretical and empirical analysis of Relief and ReliefF. *Machine Learning*, *53*, 23–69.

Singh, S., Kubica, J., Larsen, S., & Sorokina, D. (2009). Parallel large scale feature selection for logistic regression. In *SIAM data mining. conference (SDM)*.

Snir, M., Otto, S., Huss-Lederman, S., Walker, D., & Dongarra, J. (1995). *MPI: the complete reference*. Cambridge: MIT Press.

Song, L., Smola, A., Gretton, A., Borgwardt, K., & Bedo, J. (2007). Supervised feature selection via dependence estimation. In *Proceedings of international conference on machine learning (ICML'07)*.

Souza, J. T., Matwin, S., & Japkowicz, N. (2006). Parallelizing feature selection. *Algorithmica*, *45*(3), 433–456.

Sugiura, N. (1976). Further analysts of the data by Akaike' s information criterion and the finite corrections. *Communications in Statistics. Theory and Methods*, *7*, 13–26.

Tibshirani, R. (1994). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B. Methodological*, *58*(1), 267–288.

Wagenmakers, E. J., & Farrel, S. (2004). AIC model selection using Akaike weights. *Psychonomic Bulletin & Review*, *11*, 192–196.

Weston, J., Elisseff, A., Schoelkopf, B., & Tipping, M. (2003). Use of the zero norm with linear models and kernel methods. *Journal of Machine Learning Research*, *3*, 1439–1461.

Yang, Y. H., & Barron, A. R. (1998). An asymptotic property of model selection criteria. *IEEE Transactions on Information Theory*, *44*(1), 95–116.

Ye, J. (2007). Least squares linear discriminant analysis. In *Proceedings of the 24th international conference on machine learning (ICML'07)*.

Zaki, M. J. & Ho, C. T. (Eds.) (2000). *Large-scale parallel data mining*. Berlin: Springer.

Zhang, Y., & d' Aspremont, A. (2011). *Handbook on semidefinite, cone and polynomial optimization: theory, algorithms, software and applications, Chap.: Sparse PCA: convex relaxations, algorithms and applications* (pp. 915–941). Berlin: Springer.

Zhao, Z., & Liu, H. (2007). Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the international conference on machine Learning (ICML)*.

Zhao, Z., & Liu, H. (2011). *Spectral feature selection for data mining*. London: Chapman & Hall/CRC.

Zhao, Z., Wang, L., Liu, H., & Ye, J. (2011). On similarity preserving feature selection. *IEEE Transactions on Knowledge and Data Engineering*, *99*, 198–206.

Zhao, Z., Cox, J., Duling, D., & Sarel, W. (2012). Massively parallel feature selection: an approach based on variance preservation. In *Proceedings of the European conference on machine learning and principles and practice of knowledge discovery in databases (ECML'12)*.

Zou, H., Hastiey, T., & Tibshirani, R. (2004). *Sparse principal component analysis*. Tech. rep., Department of Statistics at Stanford University.