Knowledge and Information Systems manuscript No.
(will be inserted by the editor)

Efficient Greedy Feature Selection for Unsupervised Learning

Ahmed K. Farahat · Ali Ghodsi ·
Mohamed S. Kamel

Received: date / Accepted: date

Abstract Reducing the dimensionality of the data has been a challenging task in data mining and machine learning applications. In these applications, the existence of irrelevant and redundant features negatively affects the efficiency and effectiveness of different learning algorithms. Feature selection is one of the dimension reduction techniques which has been used to allow a better understanding of data and improve the performance of other learning tasks. Although the selection of relevant features has been extensively studied in supervised learning, feature selection with the absence of class labels is still a challenging task. This paper proposes a novel method for unsupervised feature selection, which efficiently selects features in a greedy manner. The paper first defines an effective criterion for unsupervised feature selection which measures the reconstruction error of the data matrix based on the selected subset of features. The paper then presents a novel algorithm for greedily minimizing the reconstruction error based on the features selected so far. The greedy

Ahmed K. Farahat
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
Tel.: +1 519-888-4567 x31463
E-mail: afarahat@uwaterloo.ca

Ali Ghodsi
Department of Statistics and Actuarial Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
Tel.: +1 519-888-4567 x37316
E-mail: aghodsib@uwaterloo.ca

Mohamed S. Kamel
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
Tel.: +1 519-888-4567 x35761
E-mail: mkamel@uwaterloo.ca

algorithm is based on an efficient recursive formula for calculating the reconstruction error. Experiments on real data sets demonstrate the effectiveness of the proposed algorithm in comparison to the state-of-the-art methods for unsupervised feature selection.[1]

Keywords Feature selection · Greedy algorithms · Unsupervised learning

1 Introduction

Data instances are typically described by a huge number of features. Most of these features are either redundant, or irrelevant to the data mining task at hand. Having a large number of redundant and irrelevant features negatively affects the performance of the underlying learning algorithms, and makes them more computationally demanding. Therefore, reducing the dimensionality of the data is a fundamental task for machine learning and data mining applications.

Throughout past years, two approaches have been proposed for dimension reduction; feature selection, and feature extraction. Feature selection (also known as variable selection or subset selection) searches for a relevant subset of existing features [1][13], while feature extraction (also known as feature transformation) learns a new set of features which combines existing features [9][12]. These methods have been employed with both supervised and unsupervised learning, where in the case of supervised learning class labels are used to guide the selection or extraction of features.

Feature extraction methods produce a set of continuous vectors which represent data instances in the space of the extracted features. Accordingly, most of these methods obtain unique solutions in polynomial time, which make these methods more attractive in terms of computational complexity. On the other hand, feature selection is a combinatorial optimization problem which is NP-hard, and most feature selection methods depend on heuristics to obtain a subset of relevant features in a manageable time. Nevertheless, feature extraction methods usually produce features which are difficult to interpret, and accordingly feature selection is more appealing in applications where understanding the meaning of features is crucial for data analysis.

Feature selection methods can be categorized into wrapper and filter methods. Wrapper methods wrap feature selection around the learning process and search for features which enhance the performance of the learning task. Filter methods, on the other hand, analyze the intrinsic properties of the data, and select highly-ranked features according to some criterion before doing the learning task. Wrapper methods are computationally more complex than filter methods as they depend on deploying the learning models many times until a subset of relevant features are found.

This paper presents an effective filter method for unsupervised feature selection. The method is based on a novel criterion for feature selection which

[1] A preliminary version of this paper appeared as [10]

measures the reconstruction error of the data matrix based on the subset of selected features. The paper presents a novel recursive formula for calculating the criterion function as well as an efficient greedy algorithm to select features. The greedy algorithm selects at each iteration the most representative feature among the remaining features, and then eliminates the effect of the selected features from the data matrix. This step makes it less likely for the algorithm to select features that are similar to previously selected features, which accordingly reduces the redundancy between the selected features. In addition, the use of the recursive criterion makes the algorithm computationally feasible and memory efficient compared to the state of the art methods for unsupervised feature selection.

The rest of this paper is organized as follows. Section 2 defines the notations used throughout the paper. Section 3 discusses previous work on filter methods for unsupervised feature selection. Section 4 presents the proposed feature selection criterion. Section 5 presents a novel recursive formula for the feature selection criterion. Section 6 proposes an effective greedy algorithm for feature selection as well as memory and time efficient variants of the algorithm. Section 7 presents an empirical evaluation of the proposed method. Finally, Section 8 concludes the paper.

2 Notations

Throughout the paper, scalars, vectors, sets, and matrices are shown in small, small bold italic, script, and capital letters, respectively. In addition, the following notations are used.

For a vector $\boldsymbol{x} \in \mathbb{R}^p$:

$\boldsymbol{x}_i$	i-th element of $\boldsymbol{x}$.
$\|\boldsymbol{x}\|$	the Euclidean norm (ℓ_2-norm) of $\boldsymbol{x}$.

For a matrix $A \in \mathbb{R}^{p \times q}$:

A_{ij}	(i, j)-th entry of A.
$A_{i:}$	i-th row of A.
$A_{:j}$	j-th column of A.
$A_{\mathcal{S}:}$	the sub-matrix of A which consists of the set $\mathcal{S}$ of rows.
$A_{:\mathcal{S}}$	the sub-matrix of A which consists of the set $\mathcal{S}$ of columns.
$\tilde{A}$	a low rank approximation of A.
$\tilde{A}_{\mathcal{S}}$	a rank-k approximation of A based on the set $\mathcal{S}$ of columns, where $\|\mathcal{S}\| = k$.
$\|A\|_F$	the Frobenius norm of A: $\|A\|_F^2 = \Sigma_{i,j} A_{ij}^2$

3 Previous Work

Many filter methods for unsupervised feature selection depend on the Principal Component Analysis (PCA) method [17] to search for the most representative features. PCA is the best-known method for unsupervised feature extraction which finds directions with maximum variance in the feature space (namely principal components). The principal components are also those directions that achieve the minimum reconstruction error for the data matrix. Jolliffe [17] suggests different algorithms to use PCA for unsupervised feature selection. In these algorithms, features are first associated with principal components based on the absolute value of their coefficients, and then features corresponding to the first (or last) principal components are selected (or deleted). This can be done once or recursively (i.e., by first selecting or deleting some features and then recomputing the principal components based on the remaining features). Similarly, sparse PCA [29], a variant of PCA which produces sparse principal components, can also be used for feature selection. This can be done by selecting for each principal component the subset of features with non-zero coefficients. However, Masaeli et al. [20] showed that these sparse coefficients may be distributed across different features and accordingly are not always useful for feature selection. Another iterative approach is suggested by Cui and Dy [7], in which the feature that is most correlated with the first principal component is selected, and then other features are projected onto the direction orthogonal to that feature. These steps are repeated until the required number of features are selected. Lu et al. [18] suggests a different PCA-based approach which applies k-means clustering to the principal components, and then selects the features that are close to clusters' centroids. Boutsidis et al. [2][3] propose a feature selection method that randomly samples features based on probabilities calculated using the k-leading singular values of the data matrix. In [3], random sampling is used to reduce the number of candidate features, and then the required number of features is selected by applying a complex subset selection algorithm on the reduced matrix. In [2], the authors derive a theoretical guarantee for the error of the k-means clustering when features are selected using random sampling. However, theoretical guarantees for other clustering algorithms were not explored in this work. Recently, Masaeli et al. [20] propose an algorithm called Convex Principal Feature Selection (CPFS). CPFS formulates feature selection as a convex continuous optimization problem which minimizes the mean-squared-reconstruction error of the data matrix (a PCA-like criterion) with sparsity constraints. This is a quadratic programming problem with linear constraints, which was solved using a projected quasi-Newton method.

Another category of unsupervised feature selection methods are based on selecting features that preserve similarities between data instances. Most of these methods first construct a k nearest neighbor graph between data instances, and then select features that preserve the structure of that graph. Examples for these methods include the Laplacian score (LS) [14] and the spectral feature selection method (a.k.a., SPEC) [28]. The Laplacian score (LS) [14] calculates a score for each feature based on the graph Laplacian and degree matrices. This score quantifies how each feature preserves similarity between data instances and their neighbors in the graph. Spectral feature selection [28] extends this idea and presents a general framework for ranking features on a k nearest neighbor graph.

Some methods directly select features which preserve the cluster structure of the data. The $Q - \alpha$ algorithm [26] measures the goodness of a subset of features based on the clustering quality (namely cluster coherence) when data is represented using only those features. The authors define a feature weight vector, and propose an iterative algorithm that alternates between calculating the cluster coherence based on current weight vector and estimating a new weight vector that maximizes that coherence. This algorithm converges to a local minimum of the cluster coherence and produces a sparse weight vector that indicates which features should be selected. Recently, Cai et al. [4] propose an algorithm called Multi-Cluster Feature Selection (MCFS) which selects a subset of features such that the multi-cluster structure of the data is preserved. To achieve that, the authors employ a method similar to spectral clustering [23], which first constructs a k nearest neighbor graph over the data instances, and then solves a generalized eigenproblem over the graph Laplacian and degree matrices. After that, for each eigenvector, an $L1$-regularized regression problem is solved to represent each eigenvector using a sparse combination of features. Features are then assigned scores based on these coefficients and highly scored features are selected. The authors show experimentally that the MCFS algorithm outperforms Laplacian score (SC) and the $Q - \alpha$ algorithm.

Another well-known approach for unsupervised feature selection is the Feature Selection using Feature Similarity (FSFS) method suggested by Mitra et al. [21]. The FSFS method groups features into clusters and then selects a representative feature for each cluster. To group features, the algorithm starts by calculating pairwise similarities between features, and then it constructs a k nearest neighbor graph over the features. The algorithm then selects the feature with the most compact neighborhood and removes all its neighbors. This process is repeated on the remaining features until all features are either selected or removed. The authors also suggested a new feature similarity measure, namely maximal information compression, which quantifies the minimum amount of information loss when one feature is represented by the other.

3.1 Comparison to Previous Work

The greedy feature selection method proposed in this paper uses a PCA-like criterion which minimizes the reconstruction error of the data matrix based on the selected subset of features. In contrast to traditional PCA-based methods, the proposed algorithm does not calculate the principal components, which is computationally demanding. Unlike Laplacian score (LS) [14] and its extension [28], the greedy feature selection method does not depend on calculating pairwise similarity between instances. It also does not calculate eigenvalue decomposition over the similarity matrix as the $Q - \alpha$ algorithm [26] and Multi-Cluster Feature Selection (MCFS) [4] do. The feature selection criterion presented in this paper is similar to that of Convex Principal Feature Selection (CPFS) [20] as both minimize the reconstruction error of the data matrix. While the method presented here uses a greedy algorithm to minimize a discrete optimization problem, CPFS solves a quadratic programming problem with sparsity constraints. In addition, the number of features selected by the CPFS depends on a regularization parameter λ which is difficult to tune. Similar to the method proposed by Cui and Dy [7], the method presented in this paper removes the effect of each selected feature by projecting other features to the direction orthogonal to that selected feature. However, the method proposed by Cui and Dy is computationally very complex as it requires the calculation of the first principal component for the whole matrix after each iteration. The Feature Selection using Feature Similarity (FSFS) [21] method employs a similar greedy approach which selects the most representative feature, and then eliminates its neighbors in the feature similarity graph. The FSFS method, however, depends on a computationally complex measure for calculating similarity between features. As shown in Section 7, experiments on real data sets show that the proposed algorithm outperforms the Feature Selection using Feature Similarity (FSFS) method [21], Laplacian score (SC) [14], and Multi-Cluster Feature Selection (MCFS) [4] when applied with different clustering algorithms.

4 Feature Selection Criterion

This section defines a novel criterion for unsupervised feature selection. The criterion measures the reconstruction error of data matrix based on the selected subset of features. The goal of the proposed feature selection algorithm is to select a subset of features that minimizes this reconstruction error.

Definition 1 (Unsupervised Feature Selection Criterion) Let A be an $m \times n$ data matrix whose rows represent the set of data instances and whose columns represent the set of features. The feature selection criterion is defined as:

$$F(\mathcal{S}) = \|A - P^{(\mathcal{S})}A\|_F^2$$

where $\mathcal{S}$ is the set of the indices of selected features, and $P^{(\mathcal{S})}$ is an $m \times m$ projection matrix which projects the columns of A onto the span of the set $\mathcal{S}$ of columns.

The criterion $F(\mathcal{S})$ represents the sum of squared errors between original data matrix A and its rank-k approximation based on the selected set of features (where $k = |\mathcal{S}|$):

$$\tilde{A}_{\mathcal{S}} = P^{(\mathcal{S})}A. \tag{1}$$

The projection matrix $P^{(\mathcal{S})}$ can be calculated as:

$$P^{(\mathcal{S})} = A_{:\mathcal{S}}\left(A_{:\mathcal{S}}^T A_{:\mathcal{S}}\right)^{-1} A_{:\mathcal{S}}^T \tag{2}$$

where $A_{:\mathcal{S}}$ is the sub-matrix of A which consists of the columns corresponding to $\mathcal{S}$. It should be noted that if the subset of features $\mathcal{S}$ is known, the projection matrix $P^{(\mathcal{S})}$ is the closed-form solution of the least-squares problem which minimizes $F(\mathcal{S})$.

The goal of the feature selection algorithm presented in this paper is to select a subset $\mathcal{S}$ of features such that $F(\mathcal{S})$ is minimized.

Problem 1 (Unsupervised Feature Selection) Find a subset of features $\mathcal{L}$ such that,

$$\mathcal{L} = arg \min_{\mathcal{S}} F(\mathcal{S}) .$$

This is an NP-hard combinatorial optimization problem. In Section 5, a recursive formula for the selection criterion is presented. This formula allows the development of an efficient algorithm to greedily minimize $F(\mathcal{S})$. The greedy algorithm is presented in Section 6.

5 Recursive Selection Criterion

In this section, a recursive formula is derived for the feature selection criterion presented in Section 4. This formula is based on a recursive formula for the projection matrix $P^{(\mathcal{S})}$ which can be derived as follows.

Lemma 1 *Given a set of features $\mathcal{S}$. For any $\mathcal{P} \subset \mathcal{S}$,*

$$P^{(\mathcal{S})} = P^{(\mathcal{P})} + R^{(\mathcal{R})}$$

where $R^{(\mathcal{R})}$ is a projection matrix which projects the columns of $E = A - P^{(\mathcal{P})} A$ onto the span of the subset $\mathcal{R} = \mathcal{S} \setminus \mathcal{P}$ of columns:

$$R^{(\mathcal{R})} = E_{:\mathcal{R}} \left(E_{:\mathcal{R}}^T E_{:\mathcal{R}} \right)^{-1} E_{:\mathcal{R}}^T .$$

Proof Define a matrix $B = A_{:\mathcal{S}}^T A_{:\mathcal{S}}$ which represents the inner-product over the columns of the sub-matrix $A_{:\mathcal{S}}$. The projection matrix $P^{(\mathcal{S})}$ can be written as:

$$P^{(\mathcal{S})} = A_{:\mathcal{S}} B^{-1} A_{:\mathcal{S}}^T \tag{3}$$

Without loss of generality, the columns and rows of $A_{:\mathcal{S}}$ and B in Eq. (3) can be rearranged such that the first sets of rows and columns correspond to $\mathcal{P}$:

$$A_{:\mathcal{S}} = \begin{bmatrix} A_{:\mathcal{P}} & A_{:\mathcal{R}} \end{bmatrix} , \quad B = \begin{bmatrix} B_{\mathcal{P}\mathcal{P}} & B_{\mathcal{P}\mathcal{R}} \\ B_{\mathcal{P}\mathcal{R}}^T & B_{\mathcal{R}\mathcal{R}} \end{bmatrix}$$

where $B_{\mathcal{P}\mathcal{P}} = A_{:\mathcal{P}}^T A_{:\mathcal{P}}$, $B_{\mathcal{P}\mathcal{R}} = A_{:\mathcal{P}}^T A_{:\mathcal{R}}$ and $B_{\mathcal{R}\mathcal{R}} = A_{:\mathcal{R}}^T A_{:\mathcal{R}}$.

Let $B_{\mathcal{R}\mathcal{R}} - B_{\mathcal{P}\mathcal{R}}^T B_{\mathcal{P}\mathcal{P}}^{-1} B_{\mathcal{P}\mathcal{R}}$ be the Schur complement [19] of $B_{\mathcal{P}\mathcal{P}}$ in B. Use the block-wise inversion formula [19] of B^{-1} and substitute with $A_{:\mathcal{S}}$ and B^{-1} in Eq. (3):

$$P^{(\mathcal{S})} = \begin{bmatrix} A_{:\mathcal{P}} & A_{:\mathcal{R}} \end{bmatrix} \begin{bmatrix} B_{\mathcal{P}\mathcal{P}}^{-1} + B_{\mathcal{P}\mathcal{P}}^{-1} B_{\mathcal{P}\mathcal{R}} S^{-1} B_{\mathcal{P}\mathcal{R}}^T B_{\mathcal{P}\mathcal{P}}^{-1} & -B_{\mathcal{P}\mathcal{P}}^{-1} B_{\mathcal{P}\mathcal{R}} S^{-1} \\ -S^{-1} B_{\mathcal{P}\mathcal{R}}^T B_{\mathcal{P}\mathcal{P}}^{-1} & S^{-1} \end{bmatrix} \begin{bmatrix} A_{:\mathcal{P}}^T \\ A_{:\mathcal{R}}^T \end{bmatrix}$$

The right-hand side can be simplified to:

$$P^{(\mathcal{S})} = A_{:\mathcal{P}} B_{\mathcal{P}\mathcal{P}}^{-1} A_{:\mathcal{P}}^T + \left(A_{:\mathcal{R}} - A_{:\mathcal{P}} B_{\mathcal{P}\mathcal{P}}^{-1} B_{\mathcal{P}\mathcal{R}} \right) S^{-1} \left(A_{:\mathcal{R}}^T - B_{\mathcal{P}\mathcal{R}}^T B_{\mathcal{P}\mathcal{P}}^{-1} A_{:\mathcal{P}}^T \right) \tag{4}$$

The first term of Eq. (4) is the projection matrix which projects the columns of A onto the span of the subset $\mathcal{P}$ of columns: $P^{(\mathcal{P})} = A_{:\mathcal{P}} B_{\mathcal{P}\mathcal{P}}^{-1} A_{:\mathcal{P}}^T$. The second term can be simplified as follows. Let E be an $m \times n$ residual matrix which is calculated as: $E = A - P^{(\mathcal{P})} A$. It can be shown that $E_{:\mathcal{R}} = A_{:\mathcal{R}} - A_{:\mathcal{P}} B_{\mathcal{P}\mathcal{P}}^{-1} B_{\mathcal{P}\mathcal{R}}$, and $S = E_{:\mathcal{R}}^T E_{:\mathcal{R}}$. Hence, the second term of Eq. (4) is the projection matrix which projects the columns of E onto the span of the subset $\mathcal{R}$ of columns:

$$R^{(\mathcal{R})} = E_{:\mathcal{R}} \left(E_{:\mathcal{R}}^T E_{:\mathcal{R}} \right)^{-1} E_{:\mathcal{R}}^T. \tag{5}$$

This proves that $P^{(\mathcal{S})}$ can be written in terms of $P^{(\mathcal{P})}$ and R as: $P^{(\mathcal{S})} = P^{(\mathcal{P})} + R^{(\mathcal{R})}$ ■

This means that projection matrix $P^{(\mathcal{S})}$ can be constructed in a recursive manner by first calculating the projection matrix which projects the columns of A onto the span of the subset $\mathcal{P}$ of columns, and then calculating the projection matrix which projects the columns of the residual matrix onto the span of the remaining columns. Based on this lemma, a recursive formula can be developed for $\tilde{A}_{\mathcal{S}}$.

Corollary 1 *Given a matrix A and a subset of columns $\mathcal{S}$. For any $\mathcal{P} \subset \mathcal{S}$,*

$$\tilde{A}_{\mathcal{S}} = \tilde{A}_{\mathcal{P}} + \tilde{E}_{\mathcal{R}}$$

where $E = A - P^{(\mathcal{P})} A$, and $\tilde{E}_{\mathcal{R}}$ is the low-rank approximation of E based on the subset $\mathcal{R} = \mathcal{S} \setminus \mathcal{P}$ of columns.

Proof Using Lemma (1), and substituting with $P^{(\mathcal{S})}$ in Eq. (1) gives:

$$\tilde{A}_{\mathcal{S}} = P^{(\mathcal{P})} A + E_{:\mathcal{R}} \left(E_{:\mathcal{R}}^T E_{:\mathcal{R}} \right)^{-1} E_{:\mathcal{R}}^T A \tag{6}$$

The first term is the low-rank approximation of A based on $\mathcal{P}$: $\tilde{A}_{\mathcal{P}} = P^{(\mathcal{P})} A$. The second term is equal to $\tilde{E}_{\mathcal{R}}$ as $E_{:\mathcal{R}}^T A = E_{:\mathcal{R}}^T E$. To prove that, multiplying $E_{:\mathcal{R}}^T$ by $E = A - P^{(\mathcal{P})} A$ gives:

$$E_{:\mathcal{R}}^T E = E_{:\mathcal{R}}^T A - E_{:\mathcal{R}}^T P^{(\mathcal{P})} A.$$

Using $E_{:\mathcal{R}} = A_{:\mathcal{R}} - P^{(\mathcal{P})} A_{:\mathcal{R}}$, the expression $E_{:\mathcal{R}}^T P^{(\mathcal{P})}$ can be written as:

$$E_{:\mathcal{R}}^T P^{(\mathcal{P})} = A_{:\mathcal{R}}^T P^{(\mathcal{P})} - A_{:\mathcal{R}}^T P^{(\mathcal{P})} P^{(\mathcal{P})}.$$

This is equal to 0 as $P^{(\mathcal{P})} P^{(\mathcal{P})} = P^{(\mathcal{P})}$ (A property of projection matrices). This means that $E_{:\mathcal{R}}^T A = E_{:\mathcal{R}}^T E$. Substituting $E_{:\mathcal{R}}^T A$ with $E_{:\mathcal{R}}^T E$ in Eq. (6) proves the corollary. ■

Based on Corollary (1), a recursive formula for the feature selection criterion can be developed as follows.

Theorem 1 *Given a set of features $\mathcal{S}$. For any $\mathcal{P} \subset \mathcal{S}$,*

$$F(\mathcal{S}) = F(\mathcal{P}) - \|\tilde{E}_{\mathcal{R}}\|_F^2$$

where $E = A - P^{(\mathcal{P})} A$, and $\tilde{E}_{\mathcal{R}}$ is the low-rank approximation of E based on the subset $\mathcal{R} = \mathcal{S} \setminus \mathcal{P}$ of columns.

Proof Substituting with $P^{(\mathcal{S})}$ in Eq. (1) gives:

$$F(\mathcal{S}) = \|A - \tilde{A}_{\mathcal{S}}\|_F^2 = \|A - \tilde{A}_{\mathcal{P}} - \tilde{E}_{\mathcal{R}}\|_F^2 = \|E - \tilde{E}_{\mathcal{R}}\|_F^2$$

Using the relation between the Frobenius norm and the trace function[2], the right-hand side can be expressed as:

$$\begin{aligned}\|E - \tilde{E}_{\mathcal{R}}\|_F^2 &= trace\left(\left(E - \tilde{E}_{(\mathcal{R})}\right)^T \left(E - \tilde{E}_{\mathcal{R}}\right)\right) \\ &= trace(E^T E - 2E^T \tilde{E}_{\mathcal{R}} + \tilde{E}_{\mathcal{R}}^T \tilde{E}_{\mathcal{R}})\end{aligned}$$

As $R^{(\mathcal{R})} R^{(\mathcal{R})} = R^{(\mathcal{R})}$, the expression $\tilde{E}_{\mathcal{R}}^T \tilde{E}_{\mathcal{R}}$ can be written as:

$$\tilde{E}_{\mathcal{R}}^T \tilde{E}_{\mathcal{R}} = E^T R^{(\mathcal{R})} R^{(\mathcal{R})} E = E^T R^{(\mathcal{R})} E = E^T \tilde{E}_{\mathcal{R}}$$

This means that: $F(\mathcal{S}) = \|E - \tilde{E}_{\mathcal{R}}\|_F^2 = trace(E^T E - \tilde{E}_{\mathcal{R}} \tilde{E}_{\mathcal{R}}) = \|E\|_F^2 - \|\tilde{E}_{\mathcal{R}}\|_F^2$. Replacing $\|E\|_F^2$ with $F(\mathcal{P})$ proves the theorem. ■

The term $\|\tilde{E}_{\mathcal{R}}\|_F^2$ represents the decrease in reconstruction error achieved by adding the subset $\mathcal{R}$ of features to $\mathcal{P}$. In the following section, a novel greedy heuristic is presented to optimize the feature selection criterion based on this recursive formula.

6 Greedy Selection Algorithm

This section presents an efficient greedy algorithm to optimize the feature selection criterion presented in Section 4. The algorithm selects at each iteration one feature such that the reconstruction error for the new set of features is minimum. This problem can be formulated as follows.

Problem 2 At iteration t, find feature l such that,

$$l = \underset{i}{arg\,min} \quad F(\mathcal{S} \cup \{i\}) \tag{7}$$

where $\mathcal{S}$ is the set of features selected during the first $t-1$ iterations.

A naïve implementation of the greedy algorithm is to calculate the reconstruction error for each candidate feature, and then select the feature with the smallest error. This implementation is however computationally very complex as it requires $\mathcal{O}(m^2n^2)$ floating-point operations per iteration. A more efficient approach is to use the recursive formula for calculating the reconstruction error. Using Theorem 1,

$$F(\mathcal{S} \cup \{i\}) = F(\mathcal{S}) - \|\tilde{E}_{\{i\}}\|_F^2,$$

where $E = A - \tilde{A}_{\mathcal{S}}$. Since $F(\mathcal{S})$ is a constant for all candidate features, an equivalent criterion is:

$$l = \underset{i}{arg\,max} \quad \|\tilde{E}_{\{i\}}\|_F^2 \tag{8}$$

[2] $\|A\|_F^2 = trace(A^T A)$

This formulation selects the feature l which achieves the maximum decrease in reconstruction error. The new objective function $\left\|\tilde{E}_{\{i\}}\right\|_F^2$ can be simplified as follows:

$$\begin{aligned}\left\|\tilde{E}_{\{i\}}\right\|_F^2 &= trace\left(\tilde{E}_{\{i\}}^T \tilde{E}_{\{i\}}\right) = trace\left(E^T R^{(\{i\})} E\right)\\ &= trace\left(E^T E_{:i}\left(E_{:i}^T E_{:i}\right)^{-1} E_{:i}^T E\right)\\ &= \frac{1}{E_{:i}^T E_{:i}} trace\left(E^T E_{:i} E_{:i}^T E\right) = \frac{\left\|E^T E_{:i}\right\|^2}{E_{:i}^T E_{:i}}.\end{aligned}$$

This defines the following equivalent problem.

Problem 3 (Greedy Feature Selection) At iteration t, find feature l such that,

$$l = \underset{i}{arg\,max} \quad \frac{\left\|E^T E_{:i}\right\|^2}{E_{:i}^T E_{:i}} \tag{9}$$

where $E = A - \tilde{A}_{\mathcal{S}}$, and $\mathcal{S}$ is the set of features selected during the first $t-1$ iterations.

The computational complexity of this selection criterion is $\mathcal{O}\left(n^2 m\right)$ per iteration, and it requires $\mathcal{O}\left(nm\right)$ memory to store the residual of the whole matrix, E, after each iteration. In the rest of this section, two novel techniques are proposed to reduce the memory and time requirements of this selection criterion.

6.1 Memory-Efficient Criterion

This section proposes a memory-efficient algorithm to calculate the feature selection criterion without explicitly calculating and storing the residual matrix E at each iteration. The algorithm is based on a recursive formula for calculating the residual matrix E.

Let $\mathcal{S}^{(t)}$ denote the set of features selected during the first $t-1$ iterations, $E^{(t)}$ denote the residual matrix at the start of the t-th iteration (i.e., $E^{(t)} = A - \tilde{A}_{\mathcal{S}^{(t)}}$), and $l^{(t)}$ be the feature selected at iteration t. The following lemma gives a recursive formula for residual matrix at the end of iteration t, $E^{(t+1)}$.

Lemma 2 $E^{(t+1)}$ *can be calculated recursively as:*

$$E^{(t+1)} = (E - \frac{E_{:l} E_{:l}^T}{E_{:l}^T E_{:l}} E)^{(t)}.$$

Proof Using Corollary 1, $\tilde{A}_{\mathcal{S}\cup\{l\}} = \tilde{A}_{\mathcal{S}} + \tilde{E}_{\{l\}}$. Subtracting both sides from A, and substituting $A - \tilde{A}_{\mathcal{S}\cup\{l\}}$ and $A - \tilde{A}_{\mathcal{S}}$ with $E^{(t+1)}$ and $E^{(t)}$ respectively gives:

$$E^{(t+1)} = (E - \tilde{E}_{\{l\}})^{(t)}$$

Using Eqs (1) and (2), $\tilde{E}_{\{l\}}$ can be expressed as $\left(E_{:l}(E_{:l}^T E_{:l})^{-1} E_{:l}^T\right) E$. Substituting $\tilde{E}_{\{l\}}$ with this formula in the above equation proves the lemma. ■

Let G be an $n \times n$ which represents the inner-products over the columns of the residual matrix E: $G = E^T E$. The following corollary is a direct result of Lemma 2.

Corollary 2 *$G^{(t+1)}$ can be calculated recursively as:*

$$G^{(t+1)} = (G - \frac{G_{:l} G_{:l}^T}{G_{ll}})^{(t)}.$$

Proof This corollary can be proved by substituting with $E^{(t+1)^T}$ (Lemma 2) in $G^{(t+1)} = E^{(t+1)^T} E^{(t+1)}$, and using the fact that:

$$\left(E_{:l}(E_{:l}^T E_{:l})^{-1} E_{:l}^T\right) \left(E_{:l}(E_{:l}^T E_{:l})^{-1} E_{:l}^T\right) = E_{:l}(E_{:l}^T E_{:l})^{-1} E_{:l}^T.$$

■

To simplify the derivation of the memory-efficient algorithm, at iteration t, define $\boldsymbol{\delta} = G_{:l}$ and $\boldsymbol{\omega} = G_{:l}/\sqrt{G_{ll}} = \boldsymbol{\delta}/\sqrt{\boldsymbol{\delta}_l}$. This means that $G^{(t+1)}$ can be calculated in terms of $G^{(t)}$ and $\boldsymbol{\omega}^{(t)}$ as follows:

$$G^{(t+1)} = (G - \boldsymbol{\omega}\boldsymbol{\omega}^T)^{(t)}, \tag{10}$$

or in terms of A and previous $\boldsymbol{\omega}$'s as:

$$G^{(t+1)} = A^T A - \sum_{r=1}^{t} (\boldsymbol{\omega}\boldsymbol{\omega}^T)^{(r)}. \tag{11}$$

$\boldsymbol{\delta}^{(t)}$ and $\boldsymbol{\omega}^{(t)}$ can be calculated in terms of A and previous $\boldsymbol{\omega}$'s as follows:

$$\boldsymbol{\delta}^{(t)} = A^T A_{:l} - \sum_{r=1}^{t-1} \boldsymbol{\omega}_l^{(r)} \boldsymbol{\omega}^{(r)},$$
$$\boldsymbol{\omega}^{(t)} = \boldsymbol{\delta}^{(t)} / \sqrt{\boldsymbol{\delta}_l^{(t)}}.$$

The feature selection criterion can be expressed in terms of G as:

$$l = \underset{i}{arg\,max} \quad \frac{\|G_{:i}\|^2}{G_{ii}}$$

The following theorem gives recursive formulas for calculating the feature selection criterion without explicitly calculating E nor G.

Theorem 2 *Let $\boldsymbol{f}_i = \|G_{:i}\|^2$ and $\boldsymbol{g}_i = G_{ii}$ be the numerator and denominator of the criterion function for a feature i respectively, $\boldsymbol{f} = [\boldsymbol{f}_i]_{i=1..n}$, and $\boldsymbol{g} = [\boldsymbol{g}_i]_{i=1..n}$. Then,*

$$\boldsymbol{f}^{(t)} = \Big(\boldsymbol{f} - 2\left(\boldsymbol{\omega} \circ \left(A^T A\boldsymbol{\omega} - \Sigma_{r=1}^{t-2}\left(\boldsymbol{\omega}^{(r)^T}\boldsymbol{\omega}\right)\boldsymbol{\omega}^{(r)}\right)\right)$$
$$+ \|\boldsymbol{\omega}\|^2 (\boldsymbol{\omega} \circ \boldsymbol{\omega})\Big)^{(t-1)},$$
$$\boldsymbol{g}^{(t)} = \Big(\boldsymbol{g} - (\boldsymbol{\omega} \circ \boldsymbol{\omega})\Big)^{(t-1)}.$$

where $\circ$ represents the Hadamard product operator.

Proof Based on Eq. (10), $\boldsymbol{f}_i^{(t)}$ can be calculated as:

$$\begin{aligned}\boldsymbol{f}_i^{(t)} &= \left(\|G_{:i}\|^2\right)^{(t)} = \left(\|G_{:i} - \boldsymbol{\omega}_i\boldsymbol{\omega}\|^2\right)^{(t-1)} \\ &= \left((G_{:i} - \boldsymbol{\omega}_i\boldsymbol{\omega})^T(G_{:i} - \boldsymbol{\omega}_i\boldsymbol{\omega})\right)^{(t-1)} \\ &= \left(G_{:i}^T G_{:i} - 2\boldsymbol{\omega}_i G_{:i}^T\boldsymbol{\omega} + \boldsymbol{\omega}_i^2\|\boldsymbol{\omega}\|^2\right)^{(t-1)} \\ &= \left(\boldsymbol{f}_i - 2\boldsymbol{\omega}_i G_{:i}^T\boldsymbol{\omega} + \boldsymbol{\omega}_i^2\|\boldsymbol{\omega}\|^2\right)^{(t-1)} .\end{aligned} \tag{12}$$

Similarly, $\boldsymbol{g}_i^{(t)}$ can be calculated as:

$$\begin{aligned}\boldsymbol{g}_i^{(t)} &= G_{ii}^{(t)} = \left(G_{ii} - \boldsymbol{\omega}_i^2\right)^{(t-1)} \\ &= \left(\boldsymbol{g}_i - \boldsymbol{\omega}_i^2\right)^{(t-1)} .\end{aligned} \tag{13}$$

Let $\boldsymbol{f} = [\boldsymbol{f}_i]_{i=1..n}$and $\boldsymbol{g} = [\boldsymbol{g}_i]_{i=1..n}$, $\boldsymbol{f}^{(t)}$ and $\boldsymbol{g}^{(t)}$ can be expressed as:

$$\begin{aligned}\boldsymbol{f}^{(t)} &= \left(\boldsymbol{f} - 2\left(\boldsymbol{\omega} \circ G\boldsymbol{\omega}\right) + \|\boldsymbol{\omega}\|^2\left(\boldsymbol{\omega} \circ \boldsymbol{\omega}\right)\right)^{(t-1)} , \\ \boldsymbol{g}^{(t)} &= \left(\boldsymbol{g} - \left(\boldsymbol{\omega} \circ \boldsymbol{\omega}\right)\right)^{(t-1)} ,\end{aligned} \tag{14}$$

where $\circ$ represents the Hadamard product operator, and $\|.\|$ is the ℓ_2 norm.

Based on the recursive formula of G (Eq. 11), the term $G\boldsymbol{\omega}$ at iteration $(t-1)$ can be expressed as:

$$\begin{aligned}G\boldsymbol{\omega} &= \left(A^TA - \Sigma_{r=1}^{t-2}\left(\boldsymbol{\omega}\boldsymbol{\omega}^T\right)^{(r)}\right)\boldsymbol{\omega} \\ &= A^TA\boldsymbol{\omega} - \Sigma_{r=1}^{t-2}\left(\boldsymbol{\omega}^{(r)^T}\boldsymbol{\omega}\right)\boldsymbol{\omega}^{(r)}\end{aligned} \tag{15}$$

Substitute with $G\boldsymbol{\omega}$ in Equation (14) gives the update formulas for $\boldsymbol{f}$ and $\boldsymbol{g}$ ■

This means that the greedy criterion can be memory-efficient by only maintaining two score variables for each feature, $\boldsymbol{f}_i$ and $\boldsymbol{g}_i$, and updating them at each iteration based on their previous values and the selected features so far. Algorithm 1 shows the complete memory-efficient greedy algorithm.

6.2 Partition-Based Criterion

The feature selection criterion calculates, at each iteration, the inner-products between each candidate feature $E_{:i}$ and other features E. The computational complexity of these inner-products is $\mathcal{O}(nm)$ per candidate feature (or $\mathcal{O}(n^2m)$ per iteration). When the memory-efficient update formulas are used, the computational complexity is reduced to $\mathcal{O}(nm)$ per iteration (that of calculating $A^TA\boldsymbol{\omega}$). However, the complexity of calculating the initial value of $\boldsymbol{f}$ is still $\mathcal{O}(n^2m)$.

Algorithm 1 Greedy Feature Selection

```
Input: Data matrix $A$, Number of features $k$
Output: Selected features $\mathcal{S}$,
Steps:
 1. Initialize $\mathcal{S} = \{ \}$
 2. Initialize $\boldsymbol{f}_i^{(0)} = \|A^T A_{:i}\|^2$, and $\boldsymbol{g}_i^{(0)} = A_{:i}^T A_{:i}$
 3. Repeat $t = 1 \rightarrow k$:
    (a) $l = arg \max_i \boldsymbol{f}_i^{(t)} / \boldsymbol{g}_i^{(t)}, \quad \mathcal{S} = \mathcal{S} \cup \{l\}$
    (b) $\boldsymbol{\delta}^{(t)} = A^T A_{:l} - \sum_{r=1}^{t-1} \boldsymbol{\omega}_l^{(r)} \boldsymbol{\omega}^{(r)}$
    (c) $\boldsymbol{\omega}^{(t)} = \boldsymbol{\delta}^{(t)} / \sqrt{\boldsymbol{\delta}_l^{(t)}}$
    (d) Update $\boldsymbol{f}_i$'s, $\boldsymbol{g}_i$'s (Theorem 2)
```

In order to reduce this computational complexity, a novel partition-based criterion is proposed, which reduces the number of inner products to be calculated at each iteration. The criterion partitions features into $c \ll n$ random groups, and selects the feature which best represents the centroids of these groups. Let $\mathcal{P}_j$ be the set of feature that belong to the j-th partition, $P = \{\mathcal{P}_1, \mathcal{P}_2, ...\mathcal{P}_c\}$ be a random partitioning of features into c groups, and B be an $m \times c$ matrix whose element j-th column is the sum of feature vectors that belong to the j-th group: $B_{:j} = \sum_{r \in \mathcal{P}_j} A_{:r}$. The use of the sum function (instead of mean) weights each column of B with the size of the corresponding group. This avoids any bias towards larger groups when calculating the sum of inner-products.

The selection criterion can be written as:

Problem 4 (Partition-Based Greedy Feature Selection) At iteration t, find feature l such that,

$$l = arg \max_i \quad \frac{\left\|F^T E_{:i}\right\|^2}{E_{:i}^T E_{:i}} \tag{16}$$

where $E = A - \tilde{A}_{\mathcal{S}}$, $\mathcal{S}$ is the set of features selected during the first $t-1$ iterations, $F_{:j} = \sum_{r \in \mathcal{P}_j} E_{:r}$, and $P = \{\mathcal{P}_1, \mathcal{P}_2, ...\mathcal{P}_c\}$ is a random partitioning of features into c groups.

Similar to E (Lemma 2), F can be calculated in a recursive manner as follows:

$$F^{(t+1)} = (F - \frac{E_{:l} E_{:l}^T}{E_{:l}^T E_{:l}} F)^{(t)}.$$

This means that random partitioning can be done once at the start of the algorithm. After that, F is initialized to B and then updated recursively using the above formula. The computational complexity of calculating B is $\mathcal{O}(nm)$ if the data matrix is full. However, this complexity could be considerably reduced if the data matrix is very sparse.

Further, a memory-efficient variant of the partition-based algorithm can be developed as follows. Let H be an $c \times n$ matrix whose element H_{ji} is the inner-product of the centroid of the j-th group and the i-th feature, weighted with the size of the j-th group: $H = F^T E$. Similarly, H can be calculated recursively as follows:

$$H^{(t+1)} = (H - \frac{H_{:l} G_{:l}^T}{G_{ll}})^{(t)}.$$

Define $\boldsymbol{\gamma} = H_{:l}$ and $\boldsymbol{\upsilon} = H_{:l}/\sqrt{G_{ll}} = \boldsymbol{\gamma}/\sqrt{\boldsymbol{\delta}_l}$. $H^{(t+1)}$ can be calculated in terms of $H^{(t)}$, $\boldsymbol{\upsilon}^{(t)}$ and $\boldsymbol{\omega}^{(t)}$ as follows:

$$H^{(t+1)} = (H - \boldsymbol{\upsilon}\boldsymbol{\omega}^T)^{(t)}, \tag{17}$$

or in terms of A and previous $\boldsymbol{\omega}$'s and $\boldsymbol{\upsilon}$'s as:

$$H^{(t+1)} = B^T A - \sum_{r=1}^{t} (\boldsymbol{\upsilon}\boldsymbol{\omega}^T)^{(r)}. \tag{18}$$

$\boldsymbol{\gamma}^{(t)}$ and $\boldsymbol{\upsilon}^{(t)}$ can be calculated in terms of A, B and previous $\boldsymbol{\omega}$'s and $\boldsymbol{\upsilon}$'s as follows:

$$\begin{aligned}\boldsymbol{\gamma}^{(t)} &= B^T A_{:l} - \sum_{r=1}^{t-1} \boldsymbol{\omega}_l^{(r)} \boldsymbol{\upsilon}^{(r)}, \\ \boldsymbol{\upsilon}^{(t)} &= \boldsymbol{\gamma}^{(t)}/\sqrt{\boldsymbol{\delta}_l^{(t)}}.\end{aligned}$$

The partition-based selection criterion can be expressed in terms of H and G as:

$$l = \underset{i}{arg\,max} \quad \frac{\|H_{:i}\|^2}{G_{ii}}$$

Similar to Theorem 2, the following theorem derives recursive formulas for the partition-based criterion function.

Theorem 3 *Let $\boldsymbol{f}_i = \|H_{:i}\|^2$ and $\boldsymbol{g}_i = G_{ii}$ be the numerator and denominator of the partition-based criterion function for a feature i respectively, $\boldsymbol{f} = [\boldsymbol{f}_i]_{i=1..n}$, and $\boldsymbol{g} = [\boldsymbol{g}_i]_{i=1..n}$. Then,*

$$\begin{aligned}\boldsymbol{f}^{(t)} &= \Big(\boldsymbol{f} - 2\left(\boldsymbol{\omega} \circ \left(A^T B \boldsymbol{\upsilon} - \Sigma_{r=1}^{t-2}\left(\boldsymbol{\upsilon}^{(r)T}\boldsymbol{\upsilon}\right)\boldsymbol{\omega}^{(r)}\right)\right) \\ &\quad + \|\boldsymbol{\upsilon}\|^2 \left(\boldsymbol{\omega} \circ \boldsymbol{\omega}\right)\Big)^{(t-1)}, \\ \boldsymbol{g}^{(t)} &= \Big(\boldsymbol{g} - \left(\boldsymbol{\omega} \circ \boldsymbol{\omega}\right)\Big)^{(t-1)}.\end{aligned}$$

where $\circ$ represents the Hadamard product operator.

Algorithm 2 Partition-based Greedy Feature Selection

```
Input: Data matrix $A$, Number of features $k$
Output: Selected features $\mathcal{S}$,
Steps:
1. Initialize $\mathcal{S} = \{\ \}$, Generate a random partitioning $P$, Calculate $B$: $B_{:j} = \sum_{r \in \mathcal{P}_j} A_{:r}$
2. Initialize $\boldsymbol{f}_i^{(0)} = \|B^T A_{:i}\|^2$, and $\boldsymbol{g}_i^{(0)} = A_{:i}^T A_{:i}$
3. Repeat $t = 1 \to k$:
   (a) $l = arg\ \underset{i}{max}\ \boldsymbol{f}_i^{(t)} / \boldsymbol{g}_i^{(t)}$, $\mathcal{S} = \mathcal{S} \cup \{l\}$
   (b) $\boldsymbol{\delta}^{(t)} = A^T A_{:l} - \sum_{r=1}^{t-1} \boldsymbol{\omega}_l^{(r)} \boldsymbol{\omega}^{(r)}$
   (c) $\boldsymbol{\gamma}^{(t)} = B^T A_{:l} - \sum_{r=1}^{t-1} \boldsymbol{\omega}_l^{(r)} \boldsymbol{\upsilon}^{(r)}$
   (d) $\boldsymbol{\omega}^{(t)} = \boldsymbol{\delta}^{(t)} / \sqrt{\boldsymbol{\delta}_l^{(t)}}$, $\boldsymbol{\upsilon}^{(t)} = \boldsymbol{\gamma}^{(t)} / \sqrt{\boldsymbol{\delta}_l^{(t)}}$
   (e) Update $\boldsymbol{f}_i$'s, $\boldsymbol{g}_i$'s (Theorem 3)
```

Proof The proof is similar to that of Theorem 2. It can be easily derived by using the recursive formula for $H_{:i}$ instead of that for $G_{:i}$. ■

In these update formulas, $A^T B$ can be calculated once and then used in different iterations. This makes the computational complexity of the new update formulas is $\mathcal{O}(nc)$ per iteration. Algorithm 2 shows the complete partition-based greedy algorithm. The computational complexity of the algorithm is dominated by that of calculating $A^T A_{:l}$ in Step (b) which is of $\mathcal{O}(mn)$ per iteration. The other complex step is that of calculating the initial $\boldsymbol{f}$, which is $\mathcal{O}(mnc)$. However, these steps can be implemented in an efficient way if the data matrix is sparse.

The total complexity of the algorithm is $\mathcal{O}(\max(mnk, mnc))$, where k is the number of features and c is the number of random partitions.

7 Experiments and Results

Experiments have been conducted on six benchmark data sets, whose properties are summarized in Table 1[3]. The first four data sets were recently used by Cai et al. [4] to evaluate different feature selection techniques in comparison to the Multi-Cluster Feature Selection (MCFS) method, while the last two data sets consist of documents and are characterized by very high dimensional feature vectors. The *ORL* data set consists of 400 face images, and it has been used to evaluate algorithms for the face identification task [24]. The *COIL'20* data set is the Columbia University Image Library [22] which consists of 1440 images for 20 different objects. The *ISOLET* is a subset of the ISOLET data set [6] whose data instances represent spoken letters. The *USPS* is the US postal handwritten digit data set [15] which consists of 9298 of handwritten

[3] Data sets are available in MATLAB format at:
`http://www.zjucadcg.cn/dengcai/Data/FaceData.html`
`http://www.zjucadcg.cn/dengcai/Data/MLData.html`
`http://www.zjucadcg.cn/dengcai/Data/TextData.html`

Table 1 The properties of data sets used to evaluate different feature selection methods.

Data set	# Instances	# Features	# Classes	Data Types	Feature Types
ORL	400	1024	40	Face images	Pixels
COIL20	1440	1024	20	Object images	Pixels
ISOLET	1560	617	26	Speech signals	Different properties [6]
USPS	9298	256	10	Digit images	Pixels
TDT2-30	9394	19677	30	Documents	Terms
20NG	18774	29360	20	Documents	Terms

digits. This data set has been used to evaluate different algorithms for handwritten digit recognition. The *TDT2-35* data set is a subset of the NIST Topic Detection and Tracking corpus [5] which consists of the top 30 categories. The *20NG* is the 20 newsgroups data set[4]. The *TDT2-35* and *20NG* data sets have been used to evaluate different algorithms for document clustering and classification. The data sets were preprocessed as follows. For image data sets (*ORL*, *COIL20* and *USPS*), the intensity values of each image were scaled to lie in the range [0 1]. For document data sets (*TDT2-35* and *20NG*), the terms that appear in less than 5 documents were removed and the normalized term frequency - inverse document frequency (*tf-idf*) weighting scheme was used to encode the importance of terms inside documents.

In the conducted experiments, seven methods for unsupervised feature selection are compared[5]:

1. **PCA-LRG**: is a PCA-based method that selects features associated with the first k principal components [17]. It has been shown that by Masaeli et al. [20] that this method achieves a low reconstruction error of the data matrix compared to other PCA-based methods[6].
2. **FSFS**: is the Feature Selection using Feature Similarity [21] method with the maximal information compression as the feature similarity measure.
3. **LS**: is the Laplacian Score (LS) [14] method.
4. **SPEC**: is the spectral feature selection method [28] using all the eigenvectors of the graph Laplacian.
5. **MCFS**: is the Multi-Cluster Feature Selection [4] method which has been shown to outperform other methods that preserve the cluster structure of the data.
6. **GreedyFS**: is the basic greedy algorithm presented in this paper (Algorithm 1).

[4] `http://people.csail.mit.edu/jrennie/20Newsgroups/`

[5] The following implementations were used:
FSFS: `http://www.facweb.iitkgp.ernet.in/~pabitra/paper/fsfs.tar.gz`
LS: `http://www.zjucadcg.cn/dengcai/Data/code/LaplacianScore.m`
SPEC: `http://featureselection.asu.edu/algorithms/fs_uns_spec.zip`
MCFS: `http://www.zjucadcg.cn/dengcai/Data/code/MCFS_p.m`

[6] The CPFA method was not included in the comparison as its implementation details were not completely specified in [20].

7. **PartGreedyFS**: is the partition-based greedy algorithm (Algorithm 2). In the conducted experiments, the number of partitions is set to 1% of the number of features. For each data set, the algorithm is repeated 10 times with different random partitions, and the average and standard deviation of the performance measures are calculated.

For methods that depend on constructing a k-nearest neighbor graph over the data instances (i.e., **LS**, **SPEC**, and **MCFS**), a five-nearest neighbor graph is constructed for each data set, and the weighs of the graph edges are calculated as follows:

$$W_{ij} = \exp\left(-\frac{D_{ij}^2}{2\left(\sum_k D_{ik}\right)\left(\sum_k D_{jk}\right)}\right),$$

where D is an $n \times n$ matrix of Euclidean distances between data instances, and W_{ij} is the weight between nodes i and j of the graph. This weighting function is a variant of the Gaussian kernel used with self-tuned spectral clustering [27], which has been shown to achieve better clustering performance compared to Gaussian kernels with manually tuned parameters.

Similar to previous work [4][14], the feature selection methods were compared based on their performance in clustering tasks. Two clustering algorithms were used to compare different methods:

- The well-known k-means algorithm [16]: For each feature selection method, the k-means algorithm is applied to the rows of the data matrix whose columns are the subset of the selected features. For document data sets, the spherical k-mean algorithm [8] is applied, where the cosine similarity between data points are used instead of the Euclidean distance. Each run of the k-means algorithm is repeated 10 times with different initial centroids and the clustering with the minimum objective function is selected. In addition, for each experiment, the k-means algorithm is repeated 20 times and the mean and standard deviation of the performance measures are calculated.
- The state-of-the-art affinity propagation (AP) algorithm [11]: The distance matrix between data instances is first calculated based on the selected subset of features, and then the AP algorithm is applied to the negative of this distance matrix. The preference vector, which controls the number of clusters, is set to the median of each column of the similarity matrix, as suggested by Frey and Dueck [11].

After the clustering is performed using the subset of selected features, the cluster labels are compared to ground-truth labels provided by human annotators and the Normalized Mutual Information (NMI) [25] between clustering labels and the class labels is calculated. The clustering performance with all features is also calculated and used as a baseline. In addition to clustering performance, the run times of different feature selection methods are compared. This run time includes the time for selecting features only, and not the run time of the clustering algorithm[7]. For all data sets, the number of selected features were changed from 1% to 10% of the total number of features.

Figures 1-3 show the clustering performance for the k-means and affinity propagation (AP) algorithms for different data sets[8], and Table 2 shows the k-means clustering performance for the best performing feature selection methods (**LS**, **MCFS**, **GreedyFS**, and **GreedyFSPart**). In Table 2, each sub-table represents a data set and each column represents a percentage of selected features. The NMI measures in each sub-column are divided into groups according to their statistical significance. The best group of methods is highlighted in bold, and the second best group is underlined. The tests of statistical significance were performed as follows. The methods in each sub-column are first sorted in a descending order according to their average NMI measures and a one-tailed t-test is then used to assess the significance of each method with respect to its successor. The t-test uses the null-hypothesis that two methods are equivalent and the alternative hypothesis that the method is superior to its successor. For each pair of methods, the t-statistic is calculated as:

$$t = \frac{\overline{q_1} - \overline{q_2}}{\sqrt{\frac{s_1^2}{r_1} + \frac{s_2^2}{r_2}}},$$

where $\overline{q_1}$ and $\overline{q_2}$ are the average NMI measures for the two methods, s_1 and s_2 are the standard deviations of the NMI measures, and r_1 and r_2 are the number of k-means runs used to estimate $\overline{q_1}$ and $\overline{q_2}$ respectively. The value of t-statistic is then compared to the critical value $t_{critical}$ obtained from the t-distribution table for a 95% confidence interval. If $t > t_{critical}$, the null-hypothesis is rejected and the method is considered superior to its successor.

It can be observed from Figures 1-3 and Table 2 that the greedy feature selection methods (**GreedyFS** and **PartGreedyFS**) outperforms the **PCA-LRG**, **FSFS**, **LS**, and **SPEC** methods for almost all data sets. The **GreedyFS** method outperforms **MCFS** for the *COIL20* data set as well as the three large data sets (*USPS*, *TDT2-30* and *20NG*), while its partition-based variant, **PartGreedyFS**, outperforms **MCFS** for the two document data sets (*TDT2-30* and *20NG*) and gives comparable performance for the *COIL20* and *USPS* data sets. The **MCFS** method mostly outperforms the two greedy algorithms for the *ORL* and *ISOLET* data sets.

Figures 4 and 5 show the run times of different feature selection methods. It can be observed that **FSFS** is computationally more expensive than other methods as it depends on calculating complex similarities between features. The **FSFS** method does not even scale to run on the document data sets. The **MCFS** method, however efficient, is more computationally complex than

[7] The experiments on the first four data sets were conducted on an Intel P4 3.6GHz machine with 2GB RAM, while the experiments on the last two last sets were conducted on an Intel Core i5 650 3.2GHz machine with 8GB RAM.

[8] The implementations of AP and SPEC algorithms do not scale to run on the **USPS** data set, and those of AP, PCA-LRG, FSFS, and SPEC do not scale to run on the **TDT2-30** and **20NG** data sets on the used simulation machines.

Laplacian score (**LS**) and the proposed greedy methods. It can be also observed that for data sets with large number of instances (like *USPS*, *TDT2-30* and *20NG*), the **MCFS** method becomes very computationally demanding as it depends on computing the eigenvectors of the data similarity matrix, and then solving an $L1$-regularized regression problem for each eigenvector.

Figure 6 and 7 show the run times of the **PCA-LRG** and Laplacian score (**LS**) methods in comparison to the proposed greedy methods. It can be observed that the **PCA-LRG** method is computationally more demanding than the proposed greedy methods for the first four data sets, and it does not scale to run on data sets with large number of features as it depends on computing the principal components of the data matrix. On the other hand, the **LS** method is computationally efficient relative to greedy methods when the number of data instances is comparable to the number of features. However, the **LS** method becomes very computationally demanding for data sets with very large number of data instances (like the *USPS* data set). It can also be observed that the partition-based greedy feature selection (**PartGreedyFS**) is more efficient than the basic greedy feature selection (**GreedyFS**).

8 Conclusions

This paper presents a novel greedy algorithm for unsupervised feature selection. The algorithm optimizes a feature selection criterion which measures the reconstruction error of the data matrix based on the subset of selected features. The paper proposes a novel recursive formula for calculating the feature selection criterion, which is then employed to develop an efficient greedy algorithm for feature selection. In addition, two memory and time efficient variants of the feature selection algorithm are proposed. It has been empirically shown that the proposed algorithm achieves better clustering performance compared to state-of-the-art methods for feature selection especially for high-dimensional data sets, and is less computationally demanding than methods that give comparable clustering performance.

References

1. Boln-Canedo, V., Snchez-Maroo, N., Alonso-Betanzos, A.: A review of feature selection methods on synthetic data. Knowl. Inf. Syst. (2012). DOI 10.1007/s10115-012-0487-8
2. Boutsidis, C., Mahoney, M., Drineas, P.: Unsupervised feature selection for the k-means clustering problem. In: Proceedings of Advances in Neural Information Processing Systems (NIPS) 22, pp. 153–161. Curran Associates, Inc., Red Hook, NY, USA (2009)
3. Boutsidis, C., Mahoney, M.W., Drineas, P.: Unsupervised feature selection for principal components analysis. In: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 61–69. ACM, New York, NY, USA (2008)
4. Cai, D., Zhang, C., He, X.: Unsupervised feature selection for multi-cluster data. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 333–342. ACM, New York, NY, USA (2010)
5. Cieri, C., Graff, D., Liberman, M., Martey, N., Strassel, S.: The TDT-2 text and speech corpus. In: Proceedings of the DARPA Broadcast News Workshop, pp. 57–60 (1999)

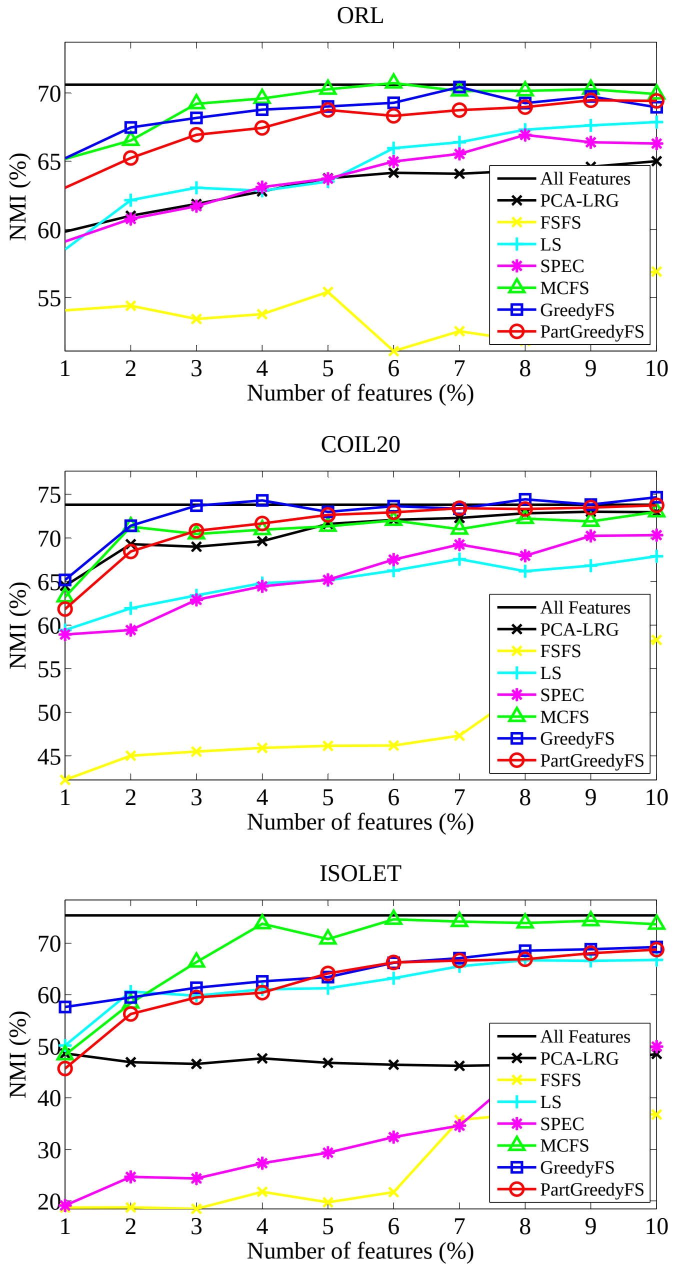

Fig. 1 The k-means clustering performance of different feature selection methods for the *ORL*, *COIL20* and *ISOLET* data sets.

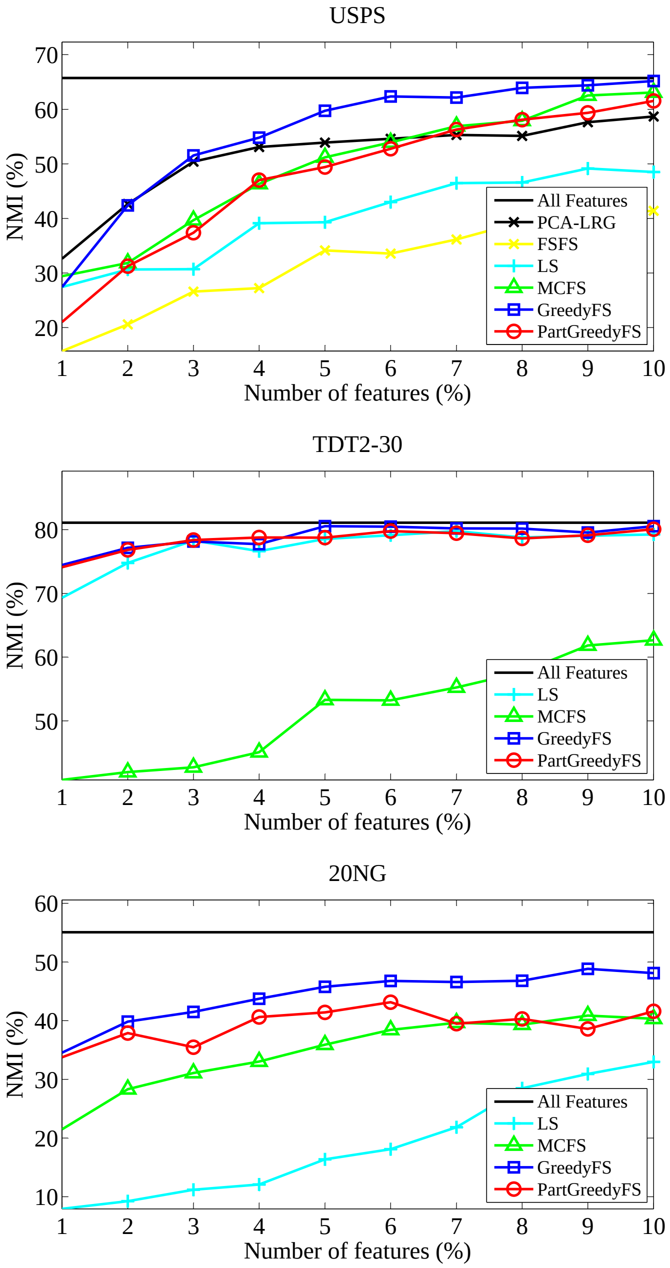

Fig. 2 The k-means clustering performance of different feature selection methods for the *USPS*, *TDT2-30* and *20NG* data sets.

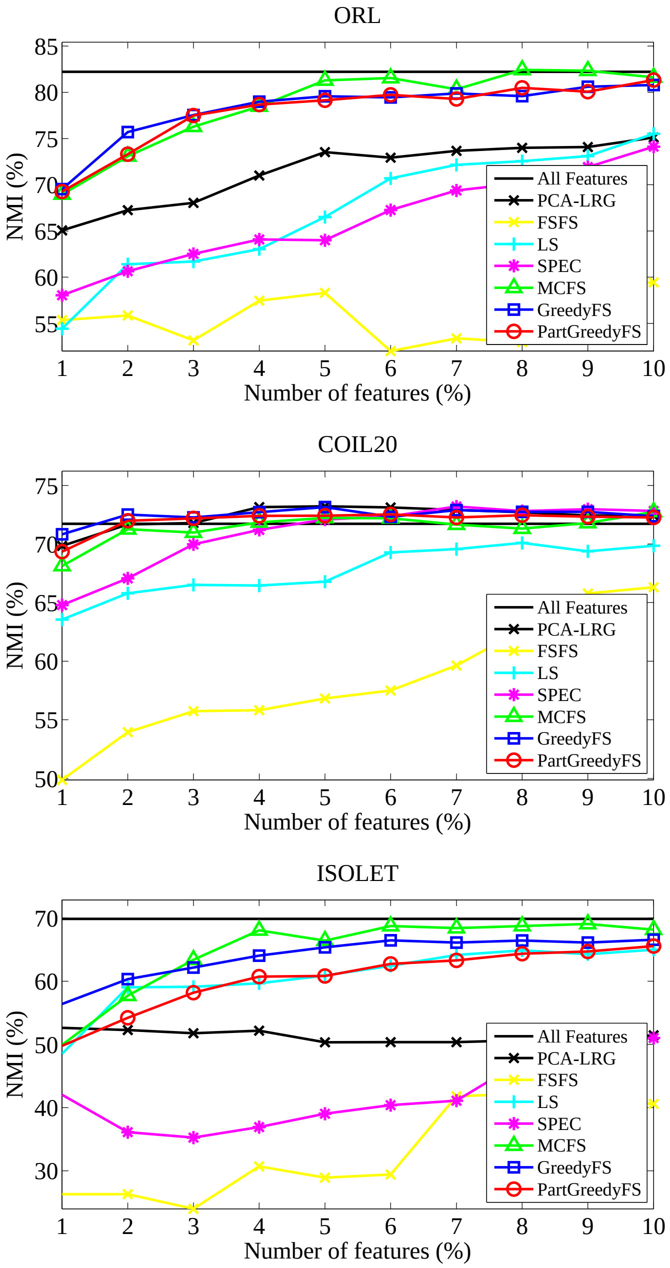

Fig. 3 The affinity propagation (AP) clustering performance of different feature selection methods for the *ORL*, *COIL20* and *ISOLET* data sets.

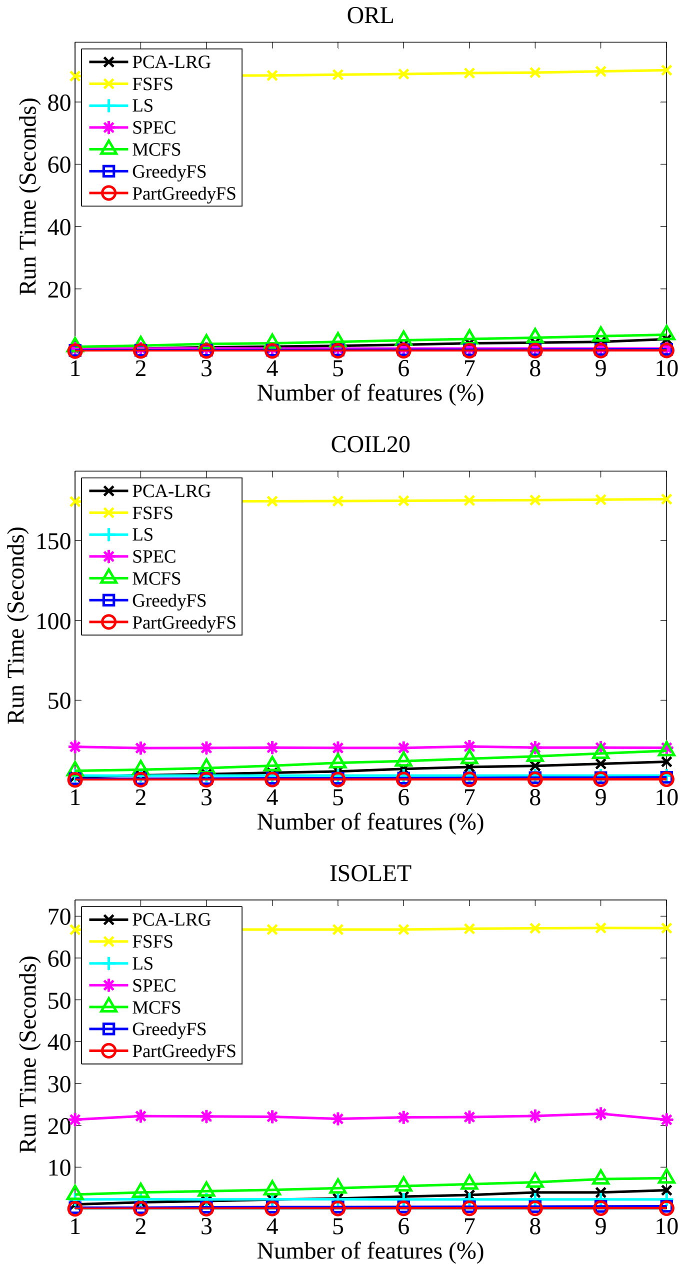

Fig. 4 The run times of different feature selection methods for the *ORL*, *COIL20* and *ISOLET* data sets.

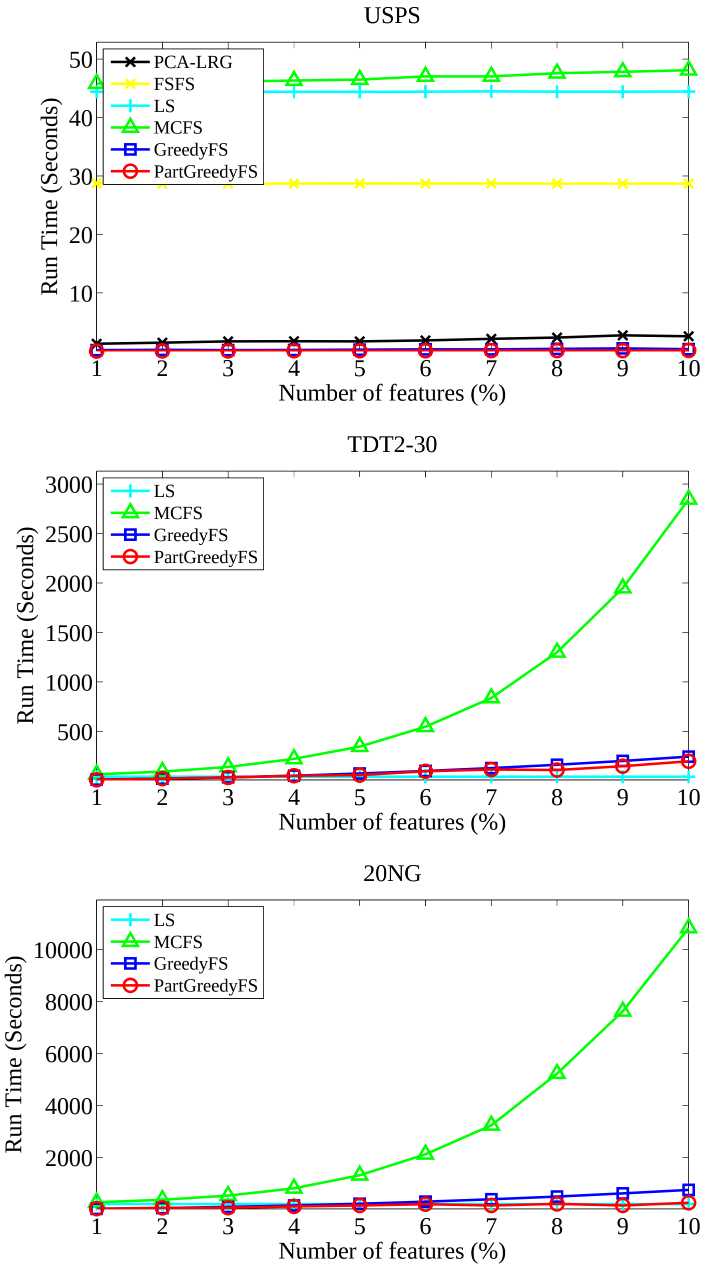

Fig. 5 The run times of different feature selection methods for the *USPS*, *TDT2-30* and *20NG* data sets.

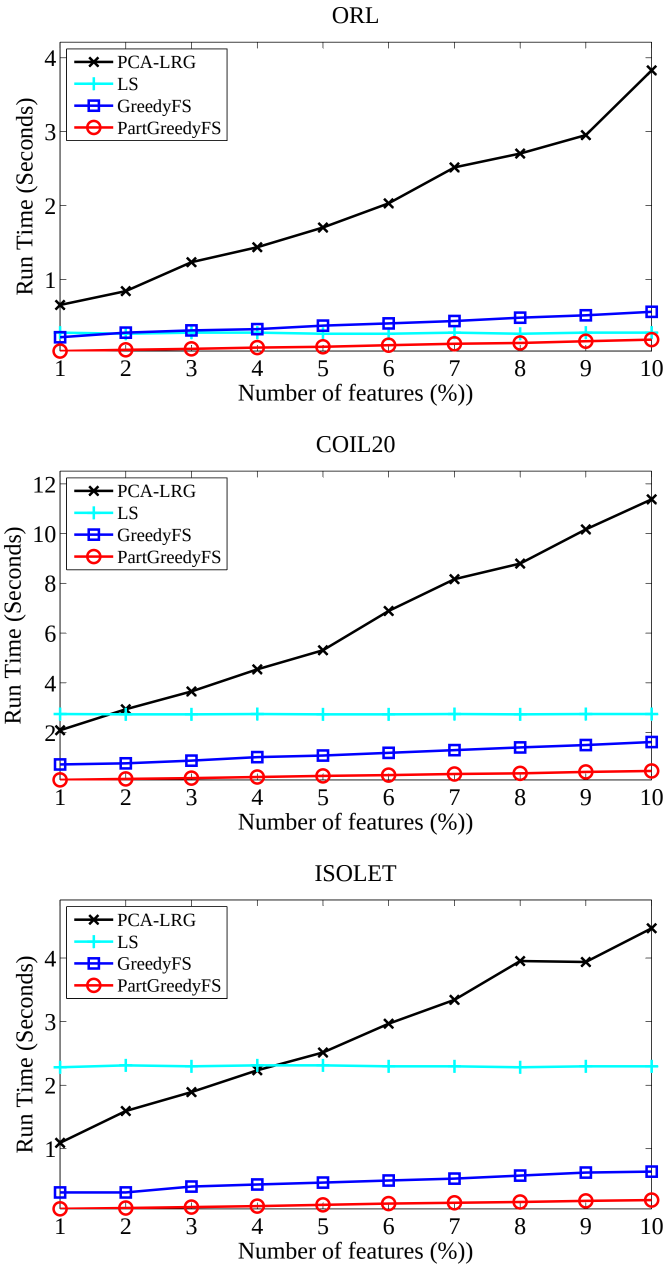

Fig. 6 The run times of the **PCA-LRG** and **LS** methods in comparison to the proposed greedy algorithms for the *ORL*, *COIL20* and *ISOLET* data sets.

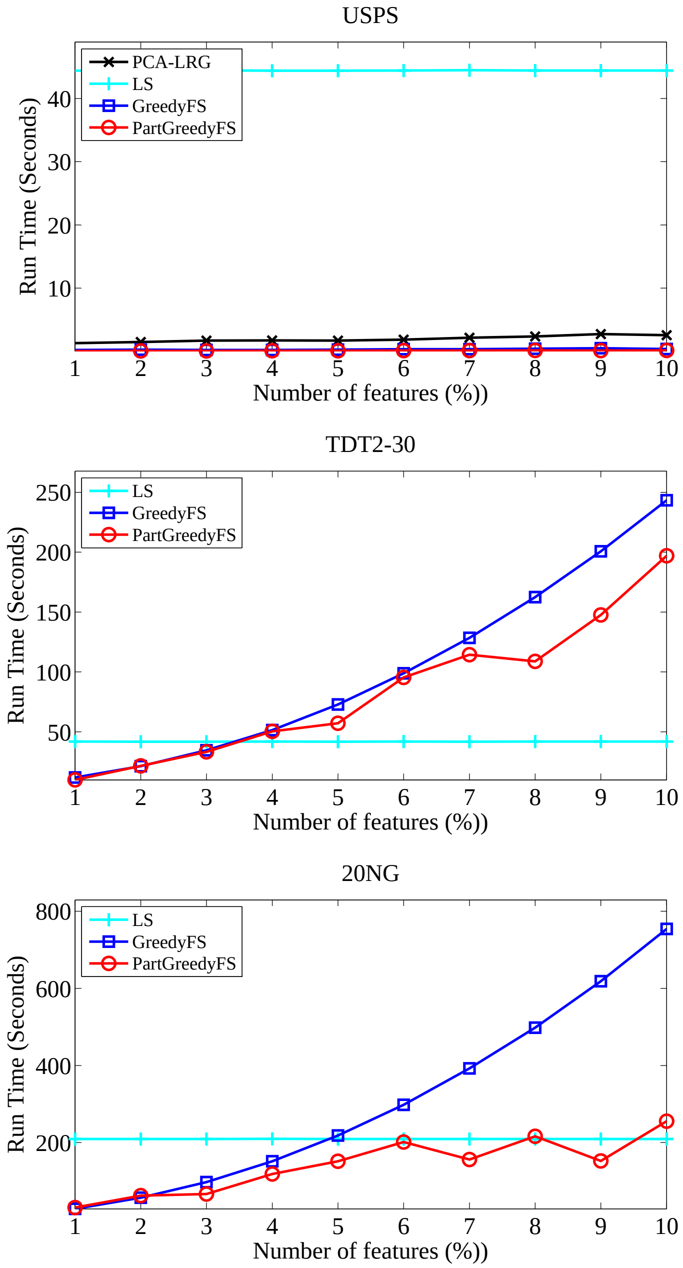

Fig. 7 The run times of the **PCA-LRG** and **LS** methods in comparison to the proposed greedy algorithms for the *USPS*, *TDT2-30* and *20NG* data sets.

Table 2 The clustering performance of the k-means algorithm for the top performing methods. In each sub-column, the best group of methods (according to t-test) is highlighted in bold, and the second best group is underlined.

Method	$k/n = 1\%$	$k/n = 4\%$	$k/n = 7\%$	$k/n = 10\%$
		ORL		
All Features	70.61±01.51	70.61±01.51	70.61±01.51	70.61±01.51
LS	58.52±00.85	62.83±00.69	66.39±01.21	67.87±01.28
MCFS	**65.17±01.09**	**69.59±01.10**	**70.15±01.45**	**69.93±01.63**
GreedyFS	**65.22±00.74**	68.78±01.42	**70.43±01.64**	**68.96±01.60**
PartGreedyFS	63.05±00.98	67.43±00.84	68.74±00.61	**69.42±00.64**
		COIL20		
All Features	73.80±02.20	73.80±02.20	73.80±02.20	73.80±02.20
LS	59.44±01.36	64.81±01.57	67.57±01.48	67.90±01.28
MCFS	63.22±01.42	70.94±01.32	71.00±01.48	72.98±01.29
GreedyFS	**65.18±01.91**	**74.30±01.49**	**73.34±02.20**	**74.66±01.43**
PartGreedyFS	61.84±01.98	71.65±00.74	**73.41±00.75**	73.73±00.66
		ISOLET		
All Features	75.40±01.82	75.40±01.82	75.40±01.82	75.40±01.82
LS	50.13±00.63	61.03±00.68	65.51±00.91	66.75±01.13
MCFS	48.32±00.92	**73.77±01.19**	**74.20±00.88**	**73.68±00.89**
GreedyFS	**57.62±00.81**	62.59±01.43	67.09±01.94	69.24±01.49
PartGreedyFS	45.66±01.75	60.39±03.55	66.64±02.73	68.77±01.84
		USPS		
All Features	65.73±00.58	65.73±00.58	65.73±00.58	65.73±00.58
LS	27.43±00.14	39.12±00.73	46.47±00.87	48.51±00.74
MCFS	**29.41±00.67**	46.31±01.80	56.91±01.02	63.08±01.27
GreedyFS	27.44±00.59	**54.81±01.04**	**62.15±01.28**	**65.17±00.88**
PartGreedyFS	21.01±01.12	47.02±01.75	56.30±02.35	61.54±01.61
		TDT2-30		
All Features	81.10±01.65	81.10±01.65	81.10±01.65	81.10±01.65
LS	69.33±01.53	76.63±02.51	**79.79±01.08**	79.26±00.95
MCFS	40.74±00.95	45.07±00.97	55.22±00.88	62.65±01.09
GreedyFS	**74.48±01.34**	77.73±02.07	**80.21±01.88**	**80.55±01.48**
PartGreedyFS	**74.10±00.62**	**78.79±01.15**	**79.47±00.54**	**80.09±00.90**
		20NG		
All Features	55.08±01.75	55.08±01.75	55.08±01.75	55.08±01.75
LS	07.95±00.45	12.08±00.48	21.85±01.00	32.97±00.81
MCFS	21.50±00.68	33.03±00.87	39.64±01.12	40.33±01.03
GreedyFS	**34.54±02.45**	**43.74±01.43**	**46.58±02.25**	**48.11±01.09**
PartGreedyFS	**33.78±00.49**	40.62±02.45	39.48±04.92	41.60±05.76

6. Cole, R., Fanty, M.: Spoken letter recognition. In: Proceedings of the Third DARPA Workshop on Speech and Natural Language, pp. 385–390 (1990)
7. Cui, Y., Dy, J.: Orthogonal principal feature selection. In: the Sparse Optimization and Variable Selection Workshop at the International Conference on Machine Learning (ICML) (2008)
8. Dhillon, I., Modha, D.: Concept decompositions for large sparse text data using clustering. Mach. Learn. **42**(1), 143–175 (2001)
9. Dhir, C., Lee, J., Lee, S.Y.: Extraction of independent discriminant features for data with asymmetric distribution. Knowl. Inf. Syst. **30**, 359–375 (2012). DOI 10.1007/s10115-011-0381-9
10. Farahat, A., Ghodsi, A., Kamel, M.: An efficient greedy method for unsupervised feature selection. In: Proceedings of the 2011 IEEE 11th International Conference on Data Mining (ICDM), pp. 161 –170 (2011)

11. Frey, B., Dueck, D.: Clustering by passing messages between data points. Science **315**(5814), 972 (2007)
12. Guyon, I.: Feature extraction: foundations and applications. Springer Verlag (2006)
13. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**, 1157–1182 (2003)
14. He, X., Cai, D., Niyogi, P.: Laplacian score for feature selection. In: Proceedings of Advances in Neural Information Processing Systems (NIPS) 18, pp. 507–514. MIT Press, Cambridge, MA, USA (2006)
15. Hull, J.: A database for handwritten text recognition research. IEEE Trans. Pattern Anal. Mach. Intell. **16**(5), 550–554 (1994)
16. Jain, A., Dubes, R.: Algorithms for Clustering Data. Prentice-Hall, Inc. (1988)
17. Jolliffe, I.: Principal Component Analysis, 2nd edn. Springer (2002)
18. Lu, Y., Cohen, I., Zhou, X., Tian, Q.: Feature selection using principal feature analysis. In: Proceedings of the 15th International Conference on Multimedia, pp. 301–304. ACM, New York, NY, USA (2007)
19. Lütkepohl, H.: Handbook of Matrices. John Wiley & Sons Inc (1996)
20. Masaeli, M., Yan, Y., Cui, Y., Fung, G., Dy, J.: Convex principal feature selection. In: Proceedings of SIAM International Conference on Data Mining (SDM), pp. 619–628. SIAM, Philadelphia, PA, USA (2010)
21. Mitra, P., Murthy, C., Pal, S.: Unsupervised feature selection using feature similarity. IEEE Trans. Pattern Anal. Mach. Intell. **24**(3), 301–312 (2002)
22. Nene, S., Nayar, S., Murase, H.: Columbia object image library (COIL-20). Tech. Rep. CUCS-005-96, Columbia University (1996)
23. Ng, A., Jordan, M., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: Proceedings of Advances in Neural Information Processing Systems (NIPS) 14, pp. 849–856. MIT Press, Cambridge, MA, USA (2001)
24. Samaria, F., Harter, A.: Parameterisation of a stochastic model for human face identification. In: Proceedings of the Second IEEE Workshop on Applications of Computer Vision, pp. 138–142 (1994)
25. Strehl, A., Ghosh, J.: Cluster ensembles – A knowledge reuse framework for combining multiple partitions. J. Mach. Learn. Res. **3**, 583–617 (2003)
26. Wolf, L., Shashua, A.: Feature selection for unsupervised and supervised inference: The emergence of sparsity in a weight-based approach. J. Mach. Learn. Res. **6**, 1855–1887 (2005)
27. Zelnik-Manor, L., Perona, P.: Self-tuning spectral clustering. In: Proceedings of Advances in Neural Information Processing Systems (NIPS) 16, pp. 1601–1608. MIT Press, Cambridge, MA, USA (2004)
28. Zhao, Z., Liu, H.: Spectral feature selection for supervised and unsupervised learning. In: Proceedings of the 24th International Conference on Machine Learning (ICML), pp. 1151–1157. ACM, New York, NY, USA (2007)
29. Zou, H., Hastie, T., Tibshirani, R.: Sparse principal component analysis. J. Comput. Graph. Stat. **15**(2), 265–286 (2006)