Unsupervised Feature Selection for Multi-Cluster Data

Deng Cai Chiyuan Zhang Xiaofei He

State Key Lab of CAD&CG, College of Computer Science
Zhejiang University, China
{dengcai,xiaofeihe}@cad.zju.edu.cn, pluskid@gmail.com

ABSTRACT

In many data analysis tasks, one is often confronted with very high dimensional data. Feature selection techniques are designed to find the relevant feature subset of the original features which can facilitate clustering, classification and retrieval. In this paper, we consider the feature selection problem in unsupervised learning scenario, which is particularly difficult due to the absence of class labels that would guide the search for relevant information. The feature selection problem is essentially a combinatorial optimization problem which is computationally expensive. Traditional unsupervised feature selection methods address this issue by selecting the top ranked features based on certain scores computed independently for each feature. These approaches neglect the possible correlation between different features and thus can not produce an optimal feature subset. Inspired from the recent developments on manifold learning and L1-regularized models for subset selection, we propose in this paper a new approach, called *Multi-Cluster Feature Selection* (MCFS), for unsupervised feature selection. Specifically, we select those features such that the multi-cluster structure of the data can be best preserved. The corresponding optimization problem can be efficiently solved since it only involves a sparse eigen-problem and a L1-regularized least squares problem. Extensive experimental results over various real-life data sets have demonstrated the superiority of the proposed algorithm.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology—*Feature evaluation and selection*

General Terms

Algorithms, Theory

Keywords

Feature selection, Unsupervised, Clustering

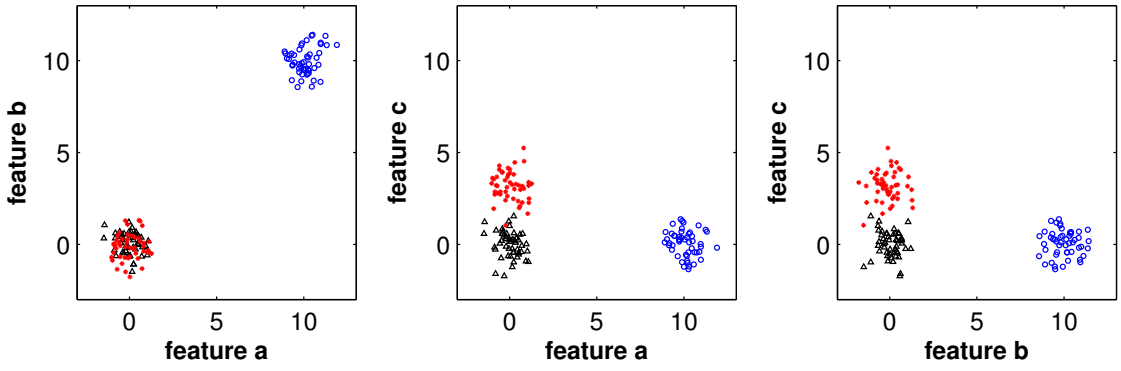
KDD'10, July 25–28, 2010, Washington, DC, USA.

1. INTRODUCTION

In many applications in computer vision, pattern recognition and data mining, one is often confronted with very high dimensional data. High dimensionality significantly increases the time and space requirements for processing the data. Moreover, various data mining and machine learning tasks, such as classification and clustering, that are analytically or computationally manageable in low dimensional spaces may become completely intractable in spaces of several hundred or thousand dimensions [12]. To overcome this problem, feature selection techniques [3, 4, 17, 21, 29, 30] are designed to reduce the dimensionality by finding a relevant feature subset. Once a small number of relevant features are selected, conventional data analysis techniques can then be applied.

Based on whether the label information is available, feature selection methods can be classified into supervised and unsupervised methods. Supervised feature selection methods usually evaluate the importance of features by the correlation between features and class label. The typical supervised feature selection methods include Pearson correlation coefficients [23], Fisher score [12], and Information gain [11]. However, in practice, there is usually no shortage of unlabeled data but labels are expensive. Hence, it is of great significance to develop unsupervised feature selection algorithms which can make use of all the data points. In this paper, we consider the problem of selecting features in unsupervised learning scenarios, which is a much harder problem due to the absence of class labels that would guide the search for relevant information.

The feature selection aims at selecting the most relevant feature subset based on certain evaluation criteria. This problem is essentially a combinatorial optimization problem which is computationally expensive. Traditional feature selection methods address this issue by selecting the top ranked features based on some scores computed independently for each feature. The scores are usually defined to reflect the power of each feature in differentiating different classes/clusters. This approach may work well on binary classes/clusters problems. However, it is very likely to fail in multi classes/clusters cases. Fig. (1) shows an intuitive example. There are three Gaussians in a three dimensional space. Without the label information, some popular unsupervised feature selection methods (*e.g.*, Maximum variance and LaplacianScore [17]) rank the features as $a > b > c$. If one is asking to select two features, these methods will select features a and b, which is obviously sub-optimal. When dealing with multi classes/clusters data, different features have



(a) plane $a \otimes b$ (b) plane $a \otimes c$ (c) plane $b \otimes c$

Figure 1: A failed example for binary clusters/classes feature selection methods. (a)-(c) show the projections of the data on the plane of two joint features, respectively. Without the label information, both Maximum variance and LaplacianScore [17] methods rank the features as $a > b > c$. If one is asking to select two features, both Maximum variance and LaplacianScore methods will select features a and b, which is obviously sub-optimal.

different powers on differentiating different classes/clusters (*e.g.*, cluster 1 vs. cluster 2 and cluster 1 vs. cluster 3). There are some studies on supervised feature selection [2] trying to solve this issue. However, without label information, it is unclear how to apply the similar ideas to unsupervised feature selection methods.

Inspired from the recent developments on spectral analysis of the data (manifold learning) [1, 22] and L1-regularized models for subset selection [14, 16], we propose in this paper a new approach, called *Multi-Cluster Feature Selection* (MCFS), for unsupervised feature selection. Specifically, we select those features such that the multi-cluster structure of the data can be well preserved. By using spectral analysis techniques, MCFS suggests a principled way to measure the correlations between different features without label information. Thus, MCFS can well handle the data with multiple cluster structure. The corresponding optimization problem only involves a sparse eigen-problem and a L1-regularized least squares problem, thus can be efficiently solved. It is important to note that our method essentially follows our previous work on spectral regression [5] and sparse subspace learning [6, 7].

The rest of the paper is organized as follows: in Section 2, we provide a brief review of the related work. Our multi cluster feature selection algorithm is introduced in Section 3. The experimental results are presented in Section 4. Finally, we provide the concluding remarks in Section 5.

2. RELATED WORK

Feature selection methods can be classified into "wrapper" methods and "filter" methods [19, 21]. The wrapper model techniques evaluate the features using the mining algorithm that will ultimately be employed. Thus, they "wrap" the selection process around the mining algorithm. Algorithms based on the filter model examine intrinsic properties of the data to evaluate the features prior to the mining tasks.

For unsupervised "wrapper" methods, the clustering is a commonly used mining algorithm [10, 13, 20, 24]. These algorithms consider feature selection and clustering simultaneously and search for features better suited to clustering aiming to improve clustering performance. However, these "wrapper" methods are usually computationally expensive [19] and may not be able to be applied on large scale data mining problems. In this paper, we are particularly interested in the filter methods which are much more efficient.

Most of the existing filter methods are supervised. Maximum variance might be the most simple yet effective *unsupervised* evaluation criterion for selecting features. This criterion essentially projects the data points along the dimensions of maximum variances. Note that, the Principal Component Analysis (PCA) algorithm shares the same principle of maximizing variance, but it involves feature transformation and obtains a set of transformed features rather than a subset of the original features.

Although the maximum variance criteria finds features that are useful for representing data, there is no reason to assume that these features must be useful for discriminating between data in different classes. Recently, the LaplacianScore algorithm [17] and its extensions [30] have been proposed to select those features which can best reflect the underlying manifold structure. LaplacianScore uses a nearest neighbor graph to model the local geometric structure of the data and selects those features which are smoothest on the graph. It has been proven [17] that with label information LaplacianScore becomes Fisher criterion score. The latter is a supervised feature selection method (filter method) which seeks features that are efficient for discrimination [12]. Fisher criterion score assigns the highest score to the feature on which the data points of different classes are far from each other while requiring data points of the same class to be close to each other.

Wolf et al. proposed a feature selection algorithm called Q-α [29]. The algorithm optimizes over a least-squares criterion function which measures the clusterability of the input data points projected onto the selected coordinates. The optimal coordinates are those for which the cluster coherence, measured by the spectral gap of the corresponding affinity

matrix, is maximized [29]. A remarkable property of the algorithm is that it always yields sparse solutions.

3. MULTI-CLUSTER FEATURE SELECTION

The generic problem of unsupervised feature selection is the following. Given a set of points $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$, $\mathbf{x}_i \in \mathbb{R}^M$, find a feature subset with size d which contains the most informative features. In other words, the points $\{\mathbf{x}'_1, \mathbf{x}'_2, \cdots, \mathbf{x}'_N\}$ represented in the d-dimensional space $\mathbb{R}^d$ can well preserve the geometric structure as the data represented in the original M-dimensional space.

Since naturally occurring data usually have multiple clusters structure, a good feature selection algorithm should consider the following two aspects:

- The selected features can best preserve the cluster structure of the data. Previous studies on unsupervised feature selection [13, 20, 24] usually use Gaussian shape clusters. However, recent studies have shown that human generated data are probably sampled from a submanifold of the ambient Euclidean space [1, 25, 28]. The intrinsic manifold structure should be considered while measuring the goodness of the clusters [22].
- The selected features can "cover" all the possible clusters in the data. Since different features have different power on differentiating different clusters, it is certainly undesirable that all the select features can well differentiate cluster 1 and cluster 2 but failed on differentiating cluster 1 and cluster 3.

In the remaining part of this section, we will introduce our *Multi-Cluster Feature Selection* (MCFS) algorithm which considers the above two aspects. We begin with a discussion on spectral embedding for cluster analysis with arbitrary shapes.

3.1 Spectral Embedding for Cluster Analysis

To detect the cluster (arbitrary shapes) structure of data, spectral clustering techniques [8, 22, 26] received significant interests recently. The spectral clustering usually clusters the data points using the top eigenvectors of *graph Laplacian* [9], which is defined on the affinity matrix of data points. From the graph partitioning perspective, spectral clustering tries to find the best cut of the graph so that the predefined criterion function can be optimized. Many criterion functions, such as ratio cut [8], average association [26], and normalized cut [26] have been proposed along with the corresponding eigen-problems for finding their optimal solutions.

Spectral clustering has a close connection with the studies on *manifold learning* [1, 25, 28], which consider the case when the data are drawn from sampling a probability distribution that has support on or near to a *submanifold* of the ambient space. In order to detect the underlying manifold structure, many manifold learning algorithms have been proposed [1, 25, 28]. These algorithms construct a nearest neighbor graph to model the local geometric structure and perform spectral analysis on the graph weight matrix. This way, these manifold learning algorithms can "unfold" the data manifold and provide the "flat" embedding for the data points. The spectral clustering can be thought as a two-step approach [1]. The first step is "unfolding" the data manifold using the manifold learning algorithms and the second step is performing traditional clustering (typically k-means) on the "flat" embedding for the data points [22].

Consider a graph with N vertices where each vertex corresponds to a data point. For each data point $\mathbf{x}_i$, we find its p nearest neighbors and put an edge between $\mathbf{x}_i$ and its neighbors. There are many choices to define the weight matrix $\mathbf{W}$ on the graph. Three of the most commonly used are as follows:

1. **0-1 weighting**. $\mathbf{W}_{ij} = 1$ if and only if nodes i and j are connected by an edge. This is the simplest weighting method and is very easy to compute.
2. **Heat kernel weighting**. If nodes i and j are connected, put
$$\mathbf{W}_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma}}$$
Heat kernel has an intrinsic connection to the Laplace Beltrami operator on differentiable functions on a manifold [1].
3. **Dot-product weighting**. If nodes i and j are connected, put
$$\mathbf{W}_{ij} = \mathbf{x}_i^T \mathbf{x}_j$$
Note that, if $\mathbf{x}$ is normalized to have unit norm, the dot product of two vectors is equivalent to the cosine similarity of the two vectors.

If the heat kernel or dot-product weighting is used, some researchers [22] use a complete graph (*i.e.*, put an edge between any two points) instead of the p-nearest neighbors graph.

Define a diagonal matrix $\mathbf{D}$ whose entries are column (or row, since $\mathbf{W}$ is symmetric) sums of $\mathbf{W}$, $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ij}$, we can compute the graph Lapalcian $\mathbf{L} = \mathbf{D} - \mathbf{W}$ [9]. The "flat" embedding for the data points which "unfold" the data manifold can be found by solving the following generalized eigen-problem [1]:

$$\mathbf{L}\mathbf{y} = \lambda \mathbf{D}\mathbf{y} \tag{1}$$

Let $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_K]$, $\mathbf{y}_k$'s are the eigenvectors of the above generalized eigen-problem with respect to the smallest eigenvalue. Each row of $\mathbf{Y}$ is the "flat" embedding for each data point. The K is the intrinsic dimensionality of the data and each $\mathbf{y}_k$ reflects the data distribution along the corresponding dimension (topic, concept, *etc.*) [1]. When one tries to perform cluster analysis of the data, each $\mathbf{y}_k$ can reflect the data distribution on the corresponding cluster. Thus, if the cluster number of the data is known, the K is usually set to be equal to the number of clusters [22].

3.2 Learning Sparse Coefficient Vectors

After we obtain the "flat" embedding $\mathbf{Y}$ for the data points, we can measure the importance of each feature along each intrinsic dimension (each column of $\mathbf{Y}$), correspondingly, the contribution of each feature for differentiating each cluster.

Given $\mathbf{y}_k$, a column of $\mathbf{Y}$, we can find a relevant subset of features by minimizing the fitting error as follows:

$$\min_{\mathbf{a}_k} \|\mathbf{y}_k - \mathbf{X}^T \mathbf{a}_k\|^2 + \beta |\mathbf{a}_k| \tag{2}$$

where $\mathbf{a}_k$ is a M-dimensional vector and $|\mathbf{a}_k| = \sum_{j=1}^{M} |a_{k,j}|$ denotes the L1-norm of $\mathbf{a}_k$. $\mathbf{a}_k$ essentially contains the combination coefficients for different features in approximating

$\mathbf{y}_k$. Due to the nature of the L1-norm penalty, some coefficients will be shrunk to exact zero if β is large enough. In this case, we can select a subset containing the most relevant features (corresponding to the non-zero coefficients in $\mathbf{a}_k$) with respect to $\mathbf{y}_k$. Eq. (2) is essentially a regression problem. In statistics, this L1-regularized regression problem is called LASSO [16].

The regression problem in Eq. (2) has the following equivalent formulation:

$$\begin{aligned} &\min_{\mathbf{a}_k} \|\mathbf{y}_k - \mathbf{X}^T\mathbf{a}_k\|^2 \\ &s.t. \quad |\mathbf{a}_k| \leq \gamma \end{aligned} \tag{3}$$

The Least Angel Regression (LARs) algorithm [14] can be used to solve the optimization problem in Eq. (3). Instead of setting the parameter γ, LARs provides another choice to control the sparseness of $\mathbf{a}_k$ by specifying the cardinality (the number of non-zero entries) of $\mathbf{a}_k$, which is particularly convenient for feature selection.

It is very possible that some features are correlated. And the combination of several "weak" features[1] can better differentiate different clusters. Several supervised feature selection algorithms [2] have been designed to address this issue. Thus, the advantage of using a L1-regularized regression model to find the subset of features instead of evaluating the contribution of each feature independently is clear.

3.3 Feature Selection on Sparse Coefficient Vectors

We consider selecting d features from the M feature candidates. For a data set containing K clusters, we can use the method discussed in the previous subsections to compute K sparse coefficient vectors $\{\mathbf{a}_k\}_{k=1}^K \in \mathbb{R}^M$. The cardinality of each $\mathbf{a}_k$ is d and each entry in $\mathbf{a}_k$ corresponds to a feature. If we select all the features that have at least one non-zero coefficient in K vectors $\{\mathbf{a}_k\}_{k=1}^K$, it is very possible that we will obtain more than d features. In reality, we can use the following simple yet effective method for selecting exactly d features from the K sparse coefficient vectors.

For every feature j, we define the MCFS score for the feature as

$$\text{MCFS}(j) = \max_k |a_{k,j}| \tag{4}$$

where $a_{k,j}$ is the j-th element of vector $\mathbf{a}_k$. We then sort all the features according to their MCFS scores in descending order and select the top d features.

We summarize the complete MCFS algorithm for feature selection in Table (1).

3.4 Computational Complexity Analysis

Our MCFS algorithm consists of five steps as shown in Table (1). The computational cost for each step can be computed as follows:

- The p-nearest neighbor graph construction step needs $O(N^2M)$ to compute the pair wise distances and $O(N^2p)$ to find p neighbors for each data point.
- For a p-nearest neighbor graph, each row of the weight matrix $\mathbf{W}$ contains approximate p non-zero values. We can use Lanczos algorithm to compute the top K eigenvectors of eigen-problem in Eq. (1) within $O(KNp)$ time [27].
- The LARs algorithm can solve the L1-regularize regression problem in Eq. (3) with cardinality constraint ($\text{cardi}(\mathbf{a}_k) = d$) in $O(d^3 + Nd^2)$ [14]. Thus, we need $O(Kd^3 + NKd^2)$ to solve the K regression problems in total.
- The MCFS scores for all the features can be computed within $O(KM)$.
- The top d features can be found within $O(M \log M)$.[2]

Considering $K \ll N$ and p is usually fixed as a constant 5, the total cost for our MCFS algorithm is:

$$O(N^2M + Kd^3 + NKd^2 + M \log M). \tag{5}$$

Table 1: MCFS for Feature Selection

Input:	N data points with M features; The number of clusters K; The number of selected features d; The number of nearest neighbors p; the weighting scheme (and the parameter σ if choosing to use the heat kernel weighting),
Output:	d selected features.

1: Construct a p nearest neighbor graph as discussed in Section 3.1.

2: Solve the generalized eigen-problem in Eq. (1), Let $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_K]$ contain the top K eigenvectors with respect to the smallest eigenvalues.

3: Solve K L1-regularized regression problems in Eq. (3) using LARs algorithm with the cardinality constraint set to d. We get K sparse coefficient vectors $\{\mathbf{a}_k\}_{k=1}^K \in \mathbb{R}^M$.

4: Compute the MCFS score for each feature according to Eq. (4).

5: return the top d features according to their MCFS scores.

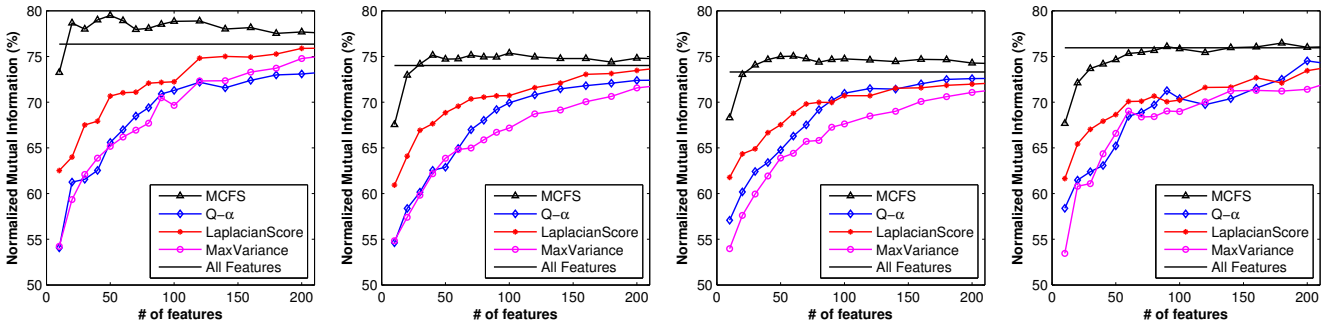
4. EXPERIMENTS

In this section, several experiments were performed to show the effectiveness of our proposed MCFS for unsupervised feature selection. These experiments include clustering and nearest neighbor classification. The following four unsupervised feature selection algorithms (filter methods) are compared:

- Our proposed **MCFS** algorithm. The number of nearest neighbors (p) is set to be 5 and we use the binary weighting for its simplicity.
- **Q-α** algorithm [29], which aims to maximize the cluster coherence.

[1] They are not very informative in differentiating different clusters if evaluated independently

[2] If d is very small, this cost can be reduced to $O(dM)$



(a) 10 Clusters (b) 20 Clusters (c) 30 Clusters (d) 40 Clusters

Figure 2: Clustering performance vs. the number of selected features on ORL data set.

Table 3: Clustering performance (%) by using 50 features on the ORL data set. The last row shows the performance by using all the 1024 features.

	10 Clusters	20 Clusters	30 Clusters	40 Clusters	Average
MCFS	**79.5±6.7**	**74.7±2.4**	**75.0±1.7**	**74.7**	**76.0**
Q-α	65.6±10.1	62.9±2.6	64.8±1.9	65.2	64.6
LaplacianScore	70.7±8.4	68.8±4.4	67.5±2.7	68.6	68.9
MaxVaiance	65.2±7.9	63.9±2.9	63.9±1.5	66.6	64.9
All Features	76.4±7.2	74.0±2.9	73.3±2.2	75.9	74.9

Table 2: Statistics of the four data sets

data set	size	# of features	# of classes
ORL	400	1024	40
USPS	9298	256	10
COIL20	1440	1024	20
Isolet	1560	617	26

- **LaplacianScore** [17], which selects those features that can best preserve the local manifold structure.
- Feature selection based on maximum variance (**MaxVariance**), which selects those features of maximum variances in order to obtain the best expressive power.

After selecting the features, the clustering and classification are then performed by only using the selected features.

4.1 Data Sets

Four real world data sets were used in our experiments. The important statistics of these data sets are summarized below (see also Table 2):

- The first one is ORL face database which consists of a total of 400 face images, of a total of 40 subjects (10 samples per subject). The images were captured at different times and have different variations including expressions (open or closed eyes, smiling or non-smiling) and facial details (glasses or no glasses). The images were taken with a tolerance for some tilting and rotation of the face up to 20 degrees. The original images were normalized (in scale and orientation) such that the two eyes were aligned at the same position. Then, the facial areas were cropped into the final images for matching. The size of each cropped image is 32 × 32 pixels, with 256 grey levels per pixel. Thus, each face image can be represented by a 1024-dimensional vector.
- The second one is the USPS handwritten digit database [18]. A popular subset[3] contains 9298 16 × 16 handwritten digit images in total is used in this experiment.
- The third one is COIL20 image library from Columbia which contains 20 objects. The images of each object were taken 5 degrees apart as the object is rotated on a turntable and each objects has 72 images. The size of each image is 32 × 32 pixels, with 256 grey levels per pixel.
- The fourth one is Isolet spoken letter recognition data[4]. This data set was first used in [15]. It contains 150 subjects who spoke the name of each letter of the alphabet twice. The speakers are grouped into sets of 30 speakers each, and are referred to as isolet1 through isolet5. In our experiment, we use isolet1 which consists 1560 examples with 617 features.

4.2 Clustering

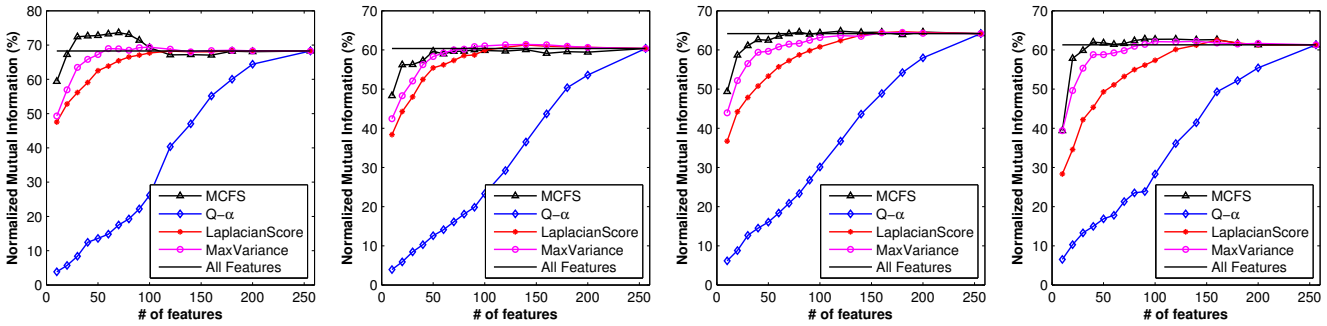
Clustering is a common technique for exploratory data analysis. In this experiment, we perform k-means clustering by using the selected features and compare the results of different algorithms.

4.2.1 Evaluation Metrics

The clustering result is evaluated by comparing the obtained label of each data point using clustering algorithms with that provided by the data set. We use the normalized mutual information metric (NMI) [17] to measure the performance. Let C denote the set of clusters obtained from the

[3]http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/multiclass.html#usps

[4]http://www.ics.uci.edu/∼mlearn/MLSummary.html



(a) 3 Clusters (b) 5 Clusters (c) 7 Clusters (d) 10 Clusters

Figure 3: Clustering performance vs. the number of selected features on USPS data set.

Table 4: Clustering performance (%) by using 50 features on the USPS data set. The last row shows the performance by using all the 256 features.

	3 Clusters	5 Clusters	7 Clusters	10 Clusters	Average
MCFS	**72.8±16.6**	**59.7±10.9**	**62.5±2.8**	**61.8**	**64.2**
Q-α	13.6±3.8	12.6±5.3	16.0±2.0	16.9	14.8
LaplacianScore	62.5±17.0	55.4±8.2	53.3±4.9	49.3	55.1
MaxVariance	67.3±16.2	58.3±7.9	59.6±5.4	58.8	61.0
All Features	68.3±15.0	60.4±10.2	64.2±3.9	61.3	63.5

ground truth and C' obtained from a clustering algorithm. Their mutual information metric $\mathrm{MI}(C, C')$ is defined as follows:

$$\mathrm{MI}(C, C') = \sum_{c_i \in C, c'_j \in C'} p(c_i, c'_j) \cdot log_2 \frac{p(c_i, c'_j)}{p(c_i) \cdot p(c'_j)}$$

where $p(c_i)$ and $p(c'_j)$ are the probabilities that a data point arbitrarily selected from the data set belongs to the clusters c_i and c'_j, respectively, and $p(c_i, c'_j)$ is the joint probability that the arbitrarily selected data point belongs to the clusters c_i as well as c'_j at the same time. In our experiments, we use the normalized mutual information NMI as follows:

$$\mathrm{NMI}(C, C') = \frac{\mathrm{MI}(C, C')}{\max(H(C), H(C'))}$$

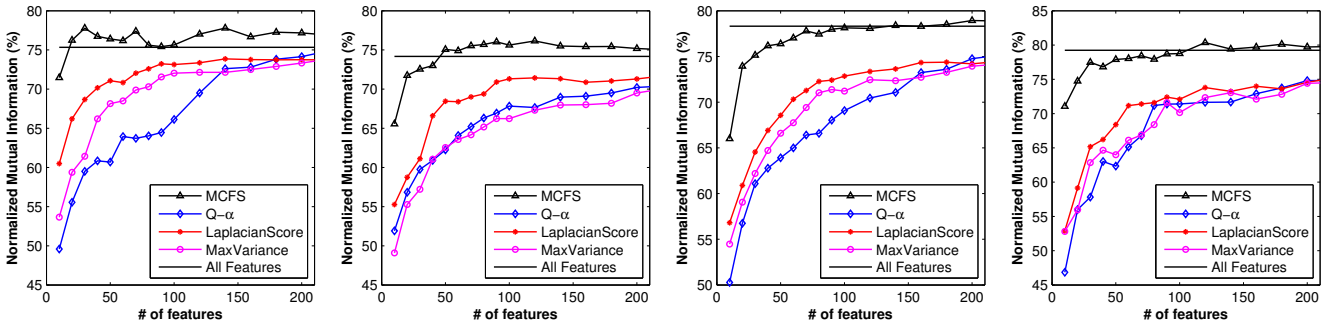
where $H(C)$ and $H(C')$ are the entropies of C and C', respectively. It is easy to check that $\mathrm{NMI}(C, C')$ ranges from 0 to 1. NMI= 1 if the two sets of clusters are identical, and NMI= 0 if the two sets are independent.

4.2.2 Clustering Results

In order to randomize the experiments, we evaluate the clustering performance with different number of clusters (K = 10, 20, 30, 40 on ORL; K = 3, 5, 7, 10 on USPS; K = 5, 10, 15, 20 on COIL20 and K = 10, 15, 20, 26 on Isolet). For each given cluster number K (except using the entire data set), 20 tests were conducted on different randomly chosen clusters, and the average performance as well as the standard deviation was computed over these 20 tests. In each test, we applied different algorithms to select d features and applied k-means for clustering. The k-means algorithm was applied 10 times with different random starting points and the best result in terms of the objective function of k-means was recorded.

Fig. (2, 3, 4 and 5) show the plots of clustering performance versus the number of selected features (d) on ORL, USPS, COIL20 and Isolet, respectively. As we can see, our proposed MCFS algorithm consistently outperforms all its competitors on all the four data sets. MCFS converges to the best results very fast, with typically around 50 features. For all the other methods, they usually require more than 200 features to achieve a reasonably good result, as can be seen from Fig. 2∼5. It would be interesting to note that, on the ORL data set, our proposed MCFS algorithm performs surprisingly well by using only 20 features. For example, in 10 clusters case and only 20 features are selected, the clustering normalized mutual information for MCFS is 78.7%, which is even better than the clustering result by using all the 1024 features (76.4%). We can see similar results in 20 and 30 clusters cases. The MaxVariance, LaplacianScore, and Q-α algorithms perform comparably to one another on ORL data set. On USPS data set, Variance is slightly better than LaplacianScore while LaplacianScore becomes slightly better than Variance on COIL 20 data set. These two methods also perfom comparably to each other on Isolet data set. The Q-α performs very bad on USPS and Isolet data sets.

Since the goal of feature selection is to reduce the dimensionality of the data, in Table 3∼6, we report the detailed clustering performance by using 50 features for each algorithm. The last column of each table records the average clustering performance over different numbers of clusters. As can be seen, MCFS significantly outperforms the other three methods on all the four data sets. LaplacianScore performs the second best on ORL, COIL20 and Isolet data sets. MaxVariance performs the second best on USPS data set. Comparing with second best method, MCFS achieves 10.3%, 5.2%, 10.6% and 10.6% relative improvements in average when measured by normalized mutual information on the ORL, USPS, COIL20 and Isolet data sets, respectively. The last row of each table records the clustering performances by using all the features.

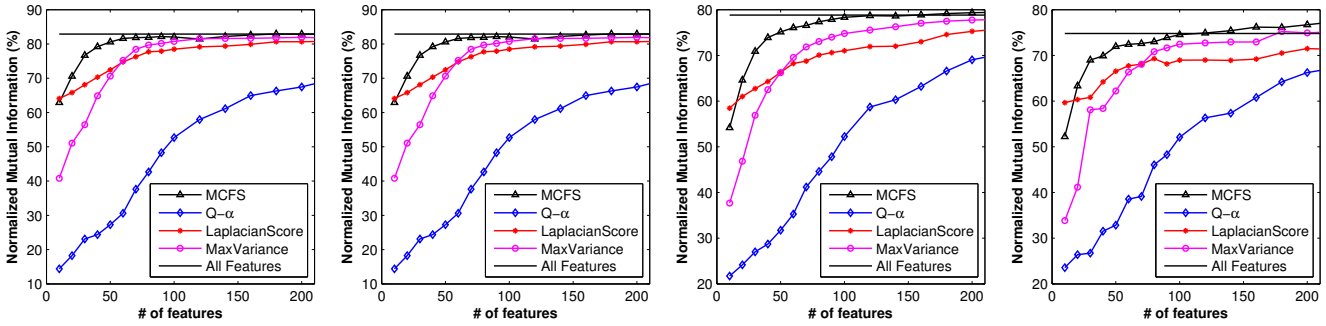


(a) 5 Clusters (b) 10 Clusters (c) 15 Clusters (d) 20 Clusters

Figure 4: Clustering performance vs. the number of selected features on COIL20 data set.

Table 5: Clustering performance (%) by using 50 features on the COIL20 data set. The last row shows the performance by using all the 1024 features.

	5 Clusters	10 Clusters	15 Clusters	20 Clusters	Average
MCFS	**76.4±13.6**	**75.1±7.7**	**76.4±5.7**	**77.9**	**76.4**
Q-α	60.7±14.9	62.2±10.5	63.9±5.5	62.4	62.3
LaplacianScore	71.1±11.7	68.5±7.0	68.6±4.6	68.4	69.1
MaxVariance	68.2±11.5	62.5±9.1	66.6±4.6	64.0	65.3
All Features	75.3±11.4	74.2±7.5	78.3±4.8	79.2	76.8



(a) 10 Clusters (b) 15 Clusters (c) 20 Clusters (d) 26 Clusters

Figure 5: Clustering performance vs. the number of selected features on Isolet data set.

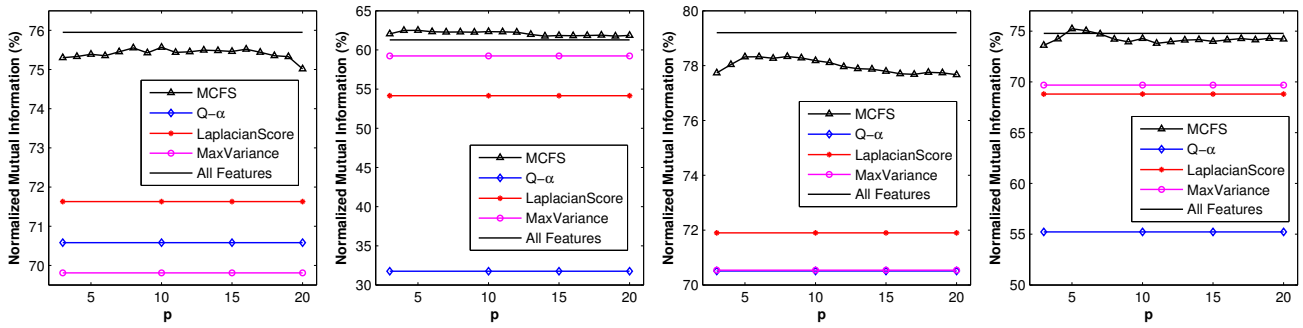
Table 6: Clustering performance (%) by using 50 features on the Isolet data set. The last row shows the performance by using all the 617 features.

	10 Clusters	15 Clusters	20 Clusters	26 Clusters	Average
MCFS	**80.6±6.0**	**76.9±3.4**	**75.1±4.1**	**72.0**	**76.1**
Q-α	27.3±9.2	30.3±4.1	31.7±3.5	32.8	30.5
LaplacianScore	72.4±6.2	70.1±5.7	66.3±3.0	66.5	68.8
MaxVariance	70.6±3.5	67.9±3.0	66.2±1.5	62.2	66.8
All Features	82.9±5.3	80.3±3.2	78.8±2.5	74.8	79.2

4.2.3 Parameter Selection

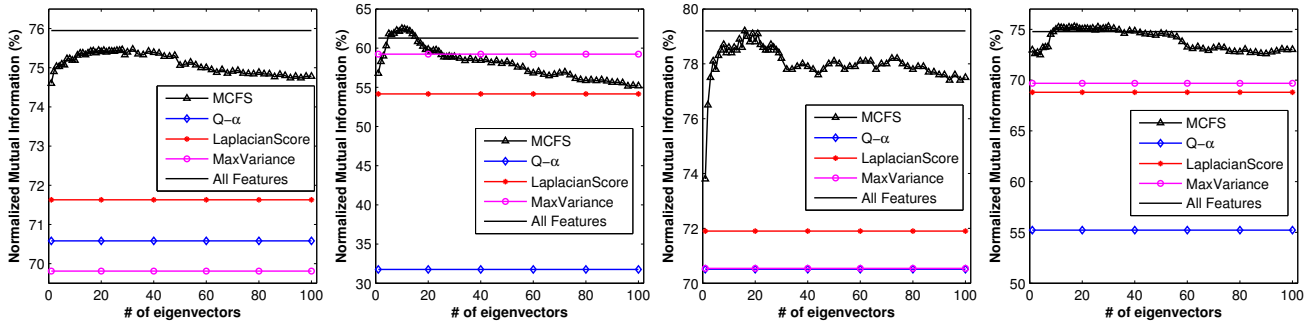
Our MCFS has only one parameter, which is the p in p-nearest neighbor graph construction. Figure (6) shows the clustering performance of MCFS versus the number of nearest neighbors (parameter p). The clustering is performed using the entire data set. The feature selection algorithms select $d = 10, 20, \cdots, 190, 200$ features (20 sets) and the average performance over these 20 feature subsets is shown in the figure. As we can see, the performance of MCFS is very stable with respect to the nearest neighbors parameter p. MCFS achieves consistent good performance with the p varying from 3 to 10 on all the four data sets. When p is larger than 10, the performance slightly decreases as the p increases.

In our MCFS algorithm, we use multiple eigenvectors of eigen-problem in Eq. (1) to capture the multi-cluster structure of the data. To demonstrate the effectiveness and the necessity to use multiple eigenvectors in multi-cluster problems, we conduct the clustering experiments by modifying



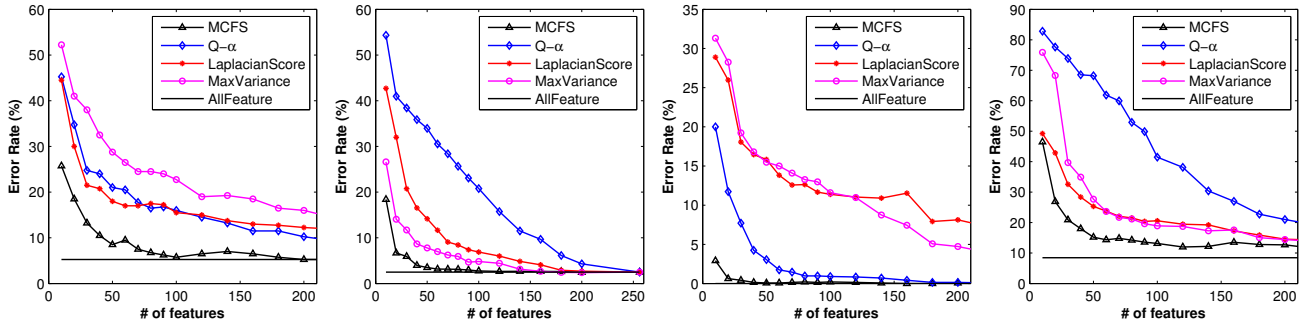
(a) ORL (b) USPS (c) COIL20 (d) Isolet

Figure 6: The performance of MCFS is very stable with respect to the nearest neighbors parameter p. MCFS achieves consistent good performance with the p varying from 3 to 10 on all the four data sets. When p is larger than 10, the performance slightly decreases as the p increases.



(a) ORL (b) USPS (c) COIL20 (d) Isolet

Figure 7: Clustering performance vs. the number of used eigenvectors. MCFS achieves its best performance as the number of used eigenvectors is equal to the number of clusters.



(a) ORL (b) USPS (c) COIL20 (d) Isolet

Figure 8: Classification error rate vs. the number of selected features.

MCFS algorithm to use various number of eigenvectors. Fig. (7) shows the average clustering performance versus the number of used eigenvectors. Same to the previous experiment, MCFS selects $d = 10, 20, \cdots, 190, 200$ features (20 sets) and averages the performance over these 20 feature subsets. As we can see, the clustering performance increases rapidly as the number of used eigenvectors increases. MCFS achieves its best performance as the number of used eigenvectors is equal to the number of clusters. In USPS and COIL20 data sets, the performances of MCFS decrease as the number of used eigenvectors keep increasing. While in ORL and Isolet data sets, the performance is relatively stable as the number of eigenvectors increases. This result clearly shows the effectiveness and the necessity to use multiple eigenvectors in multi-cluster problems. Since the performance drop of MCFS is comparatively small as the number of eigenvectors become larger than the number of clusters, it is not a crucial problem to estimate the number of clusters if it is unknown.

Table 7: Classification error rate (%) by using 50 features. The last row shows the error rate by using all the features.

	ORL	USPS	COIL20	ISOLET
MCFS	**8.5**	**3.5**	**0.1**	**15.2**
Q-α	21.0	33.9	3.1	68.2
LaplacianScore	18.0	14.2	15.8	25.3
MaxVariance	28.8	7.8	15.5	27.7
All Features	5.3	2.5	0	8.5

4.3 Nearest Neighbor Classification

In this subsection, we evaluate the discriminating power of different feature selection algorithms. We consider the nearest neighbor classifier and the good features should yield high classification accuracy.

We perform leave-one-out cross validation as follows: For each data point $\mathbf{x}_i$, we find its nearest neighbor $\mathbf{x}_i'$. Let $c(\mathbf{x}_i)$ be the class label of $\mathbf{x}_i$. The nearest neighbor classification error rate (ER) is thus defined as

$$\mathrm{ER} = 1 - \frac{1}{N}\sum_{i=1}^{N}\delta\big(c(\mathbf{x}_i), c(\mathbf{x}_i')\big), \tag{6}$$

where N is the number of data points and $\delta(a,b) = 1$ if $a = b$ and 0 otherwise.

Fig. (8) shows the plots of nearest neighbor classification error rate versus the number of selected features. As can be seen, on all the four data sets, MCFS consistently outperforms the other three methods. Similar to clustering, MCFS converges to the best result very fast, with no more than 100 features. Particularly, on the COIL20 data set, MCFS can achieve 0.1% classification error rate by using only 50 features. On this data set, the Q-α algorithm performs comparably to our algorithms and much better than MaxVariance and LaplacianScore. On the ORL data set, the LaplacianScore and Q-α algorithms perform comparably to each other, and MaxVariance performs the worst. On the USPS and Isolet data sets, LaplacianScore and MaxVariance algorithms perform comparably and Q-α performs very bad. Similar to clustering, in Table (7) we show the nearest neighbor classification error rate for each algorithm using only 50 features. As can be seen, with only 50 features, MCFS achieves comparable results with that using all the features.

4.4 Summary

The clustering and nearest neighbor classification experiments on four real world data sets have been systematically performed. These experiments reveal a number of interesting points:

- On all the four data sets, MCFS consistently outperforms the other three algorithms for both clustering and nearest neighbor classification. As the number of selected features increases, the clustering and nearest neighbor classification performance for all the methods increase and the performance difference among different methods gets smaller.

- Our proposed algorithm performs especially well when the number of selected features is small (*e.g.*, $d \leq 50$). By using only 50 features, the performance of our proposed MCFS algorithm is comparable with and sometimes even better than (see Table 3∼6) the performance by using all the features. Therefore, comparing to MaxVariance, LaplacianScore, and Q-α, our algorithm can achieve much more compact representation without sacrifice of discriminating power.

- The experiments on clustering performance versus the number of used eigenvectors show the effectiveness and the necessity to use multiple eigenvectors in multi-cluster problems.

5. CONCLUSIONS

This paper presents a novel unsupervised feature selection algorithm, called *Multi-Cluster Feature Selection* (MCFS). Inspired from the recent developments on spectral analysis of the data (manifold learning) [1, 22] and L1-regularized models for subset selection [14, 16], we propose to use multiple eigenvectors of graph Laplacian, which is defined on the affinity matrix of data points, to capture the multi-cluster structure of the data. Thus, MCFS can well handle multi-cluster data. In comparison with one simple method, that is, MaxVariance, and two state-of-the-art methods, namely, LaplacianScore and Q-α, the experimental results validate that the new method achieves significantly higher performance for clustering and classification. Our proposed MCFS algorithm performs especially well when the number of selected features is less than 50.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China under Grants 60905001 and 90920303, National Key Basic Research Foundation of China under Grant 2009CB320801. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of the funding agencies.

6. REFERENCES

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, pages 585–591. 2001.

[2] J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3:1229–1243, 2003.

[3] S. Boutemedjet, N. Bouguila, and D. Ziou. A hybrid feature extraction selection approach for high-dimensional non-gaussian data clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(8):1429–1443, 2009.

[4] C. Boutsidis, M. W. Mahoney, and P. Drineas. Unsupervised feature selection for principal components analysis. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08)*, pages 61–69, 2008.

[5] D. Cai. *Spectral Regression: A Regression Framework for Efficient Regularized Subspace Learning*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, May 2009.

[6] D. Cai, X. He, and J. Han. Spectral regression: A unified approach for sparse subspace learning. In *Proc. Int. Conf. on Data Mining (ICDM'07)*, 2007.

[7] D. Cai, X. He, and J. Han. Sparse projections over graph. In *Proc. 2008 AAAI Conf. on Artificial Intelligence (AAAI'08)*, 2008.

[8] P. K. Chan, D. F. Schlag, and J. Y. Zien. Spectral k-way ratio-cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design*, 13:1088–1096, 1994.

[9] F. R. K. Chung. *Spectral Graph Theory*, volume 92 of *Regional Conference Series in Mathematics*. AMS, 1997.

[10] C. Constantinopoulos, M. K. Titsias, and A. Likas. Bayesian feature and model selection for gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):1013–1018, 2006.

[11] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

[12] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, Hoboken, NJ, 2nd edition, 2000.

[13] J. G. Dy and C. E. Brodley. Feature selection for unsupervised learning. *Journal of Machine Learning Research*, 5:845–889, 2004.

[14] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

[15] M. A. Fanty and R. Cole. Spoken letter recognition. In *Advances in Neural Information Processing Systems 3*, 1990.

[16] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag, 2001.

[17] X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. In *Advances in Neural Information Processing Systems 18*, 2005.

[18] J. J. Hull. A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(5), 1994.

[19] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[20] M. H. C. Law, M. A. T. Figueiredo, and A. K. Jain. Simultaneous feature selection and clustering using mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1154–1166, 2004.

[21] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):491–502, 2005.

[22] A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, Cambridge, MA, 2001.

[23] J. L. Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988.

[24] V. Roth and T. Lange. Feature selection in clustering problems. In *Advances in Neural Information Processing Systems 16*. 2003.

[25] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[26] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[27] G. W. Stewart. *Matrix Algorithms Volume II: Eigensystems*. SIAM, 2001.

[28] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[29] L. Wolf and A. Shashua. Feature selection for unsupervised and supervised inference: The emergence of sparsity in a weight-based approach. *Journal of Machine Learning Research*, 6:1855–1887, 2005.

[30] Z. Zhao and H. Liu. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th Annual International Conference on Machine Learning (ICML'07)*, pages 1151–1157, 2007.