2010 International Conference on Technologies and Applications of Artificial Intelligence

Unsupervised Feature Selection: Minimize Information Redundancy of Features

Chun-Chao Yen Liang-Chieh Chen Shou-De Lin
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
r96944016@csie.ntu.edu.tw aquariusjay@gmail.com sdlin@csie.ntu.edu.tw

***Abstract*—This paper proposes an unsupervised feature selection method to remove the redundant features from datasets. The major contributions are twofold. First, we propose an eigen-decomposition method to rank the hyperplanes (which describes the relations between features) based on their linear dependency characteristic, and then design an efficient Gaussian-elimination method to sequentially remove the feature that is best represented by the rest of the features. Second, we provide a proof showing that our method is similar to removing the features that contribute the most to the Principal Components with the smallest eigen-value, but considering the effect of each removal of features with complexity about max(O(nm), $O(n^2)$) instead of $O(n^3)$, where n is the number of features and m is the number of observations. We perform experiments on an artificial and real-world datasets. The results show that our method can almost perfectly remove those dependent features without losing any independent dimension in the artificial dataset and outperforms two other competitive algorithms in the real-world datasets.**

Keywords-unsupervised feature selection; eigen-decomposition; Gaussian-elimination; PCA

I. INTRODUCTION

A knowledge discovery and data mining (KDD) system aims at extracting knowledge from data. Usually a KDD method takes data as a set of observations which are described by some features. Since different features can provide different information about the observations, for real-world tasks it is very common that human beings tend to collect as many features as possible in the first place so as to not omit any possible clues. Furthermore, in many real world cases the data are collected through different resources by different individuals, therefore very often certain level of dependency or redundancy is introduced once the evidence from different sources are merged. As a large number of features bring the "curse of dimensionality" as well as the increasing need of computational time and space, how to remove redundant features becomes an important research problem [1].

Feature selection methods can be generally divided into two categories: supervised [2-4], and unsupervised [5-8] methods depending on the involvement of the target with the problem at hand (e.g. the prediction or classification problems). In the case of unsupervised feature selection, the methods search for a subset of features which is a best subset under certain criteria. One of the interesting criteria that has been used in the literature is the measurement of relatedness or redundancy among features [6] [9]. One redundant feature can be represented by some other features, and thus the removal of a redundant feature causes no loss of information if such representation can be captured by the classifier. There are two types of methods to measure the redundancy among features, namely clustering based methods and principal component analysis (PCA) based methods that have been investigated to solve this problem. Clustering based methods first cluster the features based on certain similarity metrics, and then select one representative feature in each cluster [10]. However, usually only pair wise similarity between two features are measured to build the hierarchical tree. PCA-based methods [11] project the data onto the principal components, and each principal component is a linear combination of original features. There are many criteria to select the features defined in [11]. The simplest way is to pick the features that have largest loadings in the most important principal components, or to reject the features that have largest loadings in the least important principal components. One main drawback of many PCA-based methods lies in the high computation to perform PCA operation after each removal or selection of feature, and thus usually suboptimal strategy is taken (e.g. only perform PCA operation once). Recently, Cumming [12] evaluated the average relatedness of features to all the principal components and feed back the effect of forward selection of each feature by recalculating the partial covariance matrix of remaining unselected features. However, its summation operation to identify the important features may be dominated by extremely large eigenvalues, and thus overlook some important features.

In this paper, we propose an unsupervised feature selection method that considers the effect of backward feature removal with acceptable complexity. The method we proposed is based on the mapping of the concept of redundancy to dependency. Our goal is to find dependent features by choosing a set of coefficients with which the linear combination of features is close to zero. Such near linear dependency of features can be considered as the equations of a hyperplane, and we propose an eigen-decomposition method to rank the hyperplanes based on their linear dependency characteristics. A backward process is designed to remove the corresponding dependency by substituting the feature with largest absolute coefficient. Geometrically, if there are dependent features in the dataset, our proposed method can find a hyperplane to which most

978-0-7695-4253-9/10 $26.00 © 2010 IEEE
DOI 10.1109/TAAI.2010.49

data points lie close. Furthermore, we prove that our proposed method is similar to removing the features that have largest loadings in the least important principal components, when features in the data matrix are transformed to zero mean and unit variance. Different from PCA-based operations which generally requires feeding back the effect of each removal of features, our proposed method only needs the computation complexity proportional to $\max(O(nm), O(n^2))$, rather than $O(n^3)$, where n is the number of features and m is the number of observations.

The major contributions of this paper can be summarized as follows:

1. We propose a new backward linear feature selection that uses eigen-decomposition to rank the linear dependency. Furthermore, the geometrical meaning of our method is explored.
2. We prove that our method is similar to PCA-based feature selection methods when features in the data matrix are transformed to zero mean and unit variance. However, our method takes into consideration the effect of removal of each feature with complexity proportional to $\max(O(nm), O(n^2))$, rather than $O(n^3)$.

II. Related Works

Feature selection is an important topic which has been studied for a long time [1]. It is possible to divide the works of feature selection into two categories: supervised feature selection and unsupervised feature selection. Supervised feature selection is applied for classification or regression. Generally, the methods consist of two major parts: goodness measurement of features and search strategy.

The method to select features can be categorized as filters and wrappers. For filters methods [4] [9] [13], the selection is independent to the underlying learning model and thus, this method is considered as a pre-processing step. However, wrappers [14] directly appeal to the performance of the learning model used. The features that can result in better prediction accuracy will be selected. The search strategy defines the way of producing possible feature subsets. Some sequential methods [15] such as forward search or backward search were proposed. To prune the size of search space, this type of search generally takes the greedy strategy at the sacrifice of optimality. There are other search methods [16] trying to improve the chance of approaching the optimal solution via "randomization" instead of searching the whole feature space.

In the case of unsupervised learning, the process of feature selection searches a feature subset which is considered as the best subset under some criteria. The categorization of supervised feature selection can be applied to unsupervised ones. However, the goodness measurement of features is defined differently so as to meet the needs in unsupervised learning. In the filter methods, without any prediction target, the measurement can only be conducted between features themselves. Madsen, et al. [8] introduced a strategy to select features by analyzing the relatedness between them through clustering. For unsupervised wrapper methods, instead of prediction accuracy, other measurements of performance of a learning algorithm are defined. Dy, et al. [5] measured clustering results by how well each cluster is separated under a specific algorithm. One interesting point mentioned in the literature is that the redundancy of features could be one criterion for measuring the goodness of features, and the goal of selection is to reduce the redundancy [6] [9]. The redundant features are those that provide none or little information, which describes the discrimination of targets (classification problems), or represents the structure of a data set.

Discarding redundant variables is an often-seen technique in multivariate analysis. Its goal is to throw away those variables while leaving the analysis result unchanged or only changed slightly. If each feature is considered as a variable, reducing redundancy of a feature set is actually performing unsupervised feature selection. To look for the redundant variables, there are two main types of methods: clustering based methods, and PCA based methods.

Grouping features into clusters is a way to remove redundant features. Under the assumption that similar features are redundant, one representative feature is selected from each cluster, and the other features within the same cluster are removed. Different measures of similarity in clustering algorithms were proposed. Mitra, et al. [7] used maximal information compression index to measure the similarity and considered k-nearest-neighbors to find clusters, while the hierarchical clustering was used with a degree-based distance as the measure of similarity in [10]. The main deficiency of clustering-based methods is that only the pair wise relation of features is considered because usually the pair wise similarity or distance is used to build the hierarchical tree and to group the features iteratively.

The well-known PCA [17] method provides a way to describe data by a set of important axes called principal components. Each principal component is a combination of original variables with some coefficients, and each coefficient can be regarded as the relatedness between this variable and the component. As a result, a variable with a larger loading in a more important component will be considered as a significant one in which more information of data is kept; a variable with a larger loading in a less important component can be discarded as a redundant one by which no useful information is provided [18]. Rather than directly looking into each principal component, other works analyzed the variances in principal components. The variances of selected variables should cover the variance of the original data as much as possible [11]. Noting that the methods discussed above measure the structural difference of data represented by the original variables and selected variables indirectly, Krzanowski [19] proposed that it is better to compare each point. Given that the dimension is reduced to K by PCA, each point described by the complete feature space and the selected feature space will have two different projections to two sets of K principal components obtained by applying PCA to these two spaces. If the selected variables model the variation of data well, then the structures of both projections should be quite similar. Considering that the structure is only under some distortion

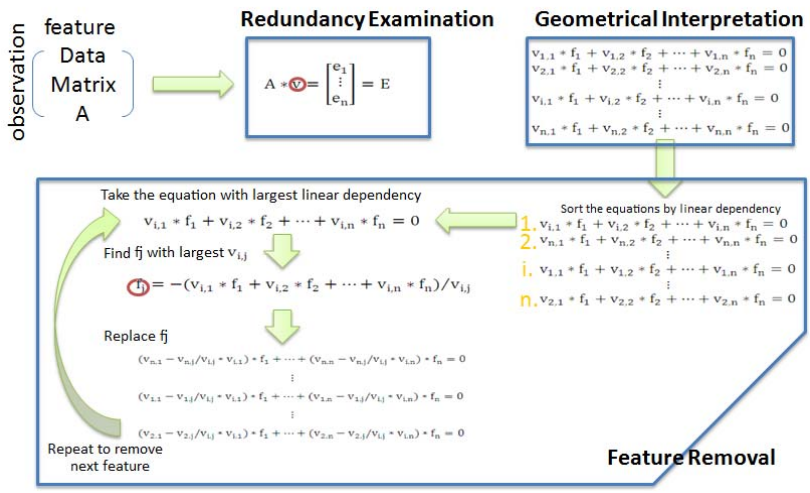

Figure 1: The flowchart of the proposed method

of translation, rotation and reflection, Procrustes analysis can measure the difference between the two data matrixes. However, the complexity becomes a concern for many PCA operations on different feature sets when the number of features increases. A recent work [12] refocused on measuring the importance of variables via the information of loading and variance provided by each principal component, but with a thorough estimation which evaluates the average relatedness to all the components and feeds back the effect of selection at each iteration. Although PCA-based methods can analyze all the features at a time, greedy search strategy is usually applied through a backward or forward selection due to the high complexity of PCA. There is one problem with greedy strategy: the effect of one selection or removal on the remaining features is not considered, while it was shown in [12] that an effective method was proposed to feed back this effect via recalculating the partial covariance matrix of the n remaining features given the removed feature with the complexity of $O(n^2)$. However, in their method, the importance of a feature was measured by sum of the product of loading in each component and the corresponding eigenvalue. The result of summing over the eigenvalues may be dominated by larger eigenvalues, and therefore can cause misjudgment of the importance of features in some cases. To cope with the problem, we propose a new unsupervised feature selection method as to remove features that have large loadings in small eigenvalues for better performance in real-world datasets. Our backward feature selection algorithm also considers the effect of each removal with the complexity of $\max(O(nm), O(n^2))$ to reweigh the n remaining features in the case of m observations. Moreover, the theoretical and geometrical interpretations are presented to support this method.

III. METHODOLOGY

The main concept of our method is the mapping of redundancy to dependence. A dependent feature can be expressed as the linear combination of some independent features. The removal of the dependent feature results in no loss of information since the information is preserved by those independent features. Our method finds a set of coefficients with which the linear combination of features (we denote it as E) is close to zero, implying the existence of dependent features. Geometrically, our method finds a hyperplane to which most observations lie close. Then, the feature with largest absolute coefficient is replaced by the other features, and the effect of its removal is updated. This process is iterated until all the remaining E's are smaller than a threshold set by the user. The flowchart of our method is shown in Fig. 1. In the following sections, each component in the flowchart is furthered explained. The definition of redundancy and the way to model it is introduced in Section III. A, and its geometrical meaning is presented in Section III. B. In Section III. C, the backward-selection process for removing the redundancy of features is proposed. We bring up the connection between our method and the PCA-based methods in Section III. D.

A. Redundancy Examination

Suppose we are given an m-by-n data matrix A. Each row of A represents an observation $x \in R^n$ and each column represents a feature $f \in R^m$. The redundancy of features is defined as the existence of dependent relations between each feature. A dependent relation implies that one feature can more or less be expressed by other features. In this case, discarding this feature results in no loss of information, since other features can make up its removal. If only linear relation is considered, then this notion can be thought as to check whether there is a non-zero column vector v with the length n of the number of features such that

$$A \cdot v = \vec{0} \tag{1}$$

The vector v describes the linear relationship among features. When dealing with real-world data, the truly linear dependency between features may not exist, and the equation becomes

$$A \cdot v = \begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix} = E \tag{2}$$

where each e_i in equation (2) is the result of the linear combination of each feature of an observation x_i, and possibly contains non-zero value. For each v, the corresponding vector E can be thought as the error vector to describe the linear dependency of features. If the error vector E is close to a zero vector, the linear dependency of features described by v is supposed to exist to some extent. This type of dependency, or one can call it near linear dependency, is needed to be considered when modeling the redundancy of features. Therefore, finding v which minimizes the following function is considered instead:

$$\left\| A \cdot v - \vec{0} \right\|_2^2 = \left\| Av \right\|_2^2 = \left\| E \right\|_2^2 \tag{3}$$

The minimization of function (3) implies that the linear dependency constraint is loosened to find some v such that Avcan be close to a zero vector. Note that the ℓ_2 norm of Av is taken to measure how different it is between Av and a zero vector. The vector v which produces the minimal value can be used to represent the relationship which best describes the near linear dependency of features. If the assumption that $||v||^2 = v^t v=1$ is used, by applying Lagrange multiplier, the minimization of function (3) can be written as:

$$\min_v \left\| Av \right\|_2^2 - \lambda(v^t v - 1) = \min_v v^t A^t A v - \lambda(v^t v - 1) \tag{4}$$

In order to solve the minimization problem, one can take the derivative of equation (4) with respect to v, and set it equal to zero to obtain

$$A^t A v = \lambda v \tag{5}$$

It implies that v is one of the eigenvectors of A^tA. Moreover, the corresponding eigenvalue λ is the ℓ_2 norm of Av since

$$\left\| Av \right\|_2^2 = v^t A^t A v = \lambda \left\| v \right\|_2^2 = \lambda \tag{6}$$

Through the eigen decomposition of A^tA, n eigenvectors and their corresponding eigenvalues can be obtained, where n is the number of features. Because all the eigenvectors satisfy equation (5), these eigenvectors orthogonally describe n linear dependency of features, with the degree of linearity expressed by the corresponding eigenvalues.

B. Geometrical Interpretation

In Equation (3), we treat the error vector E as the difference between a zero vector and the multiplication of the data matrix A and an eigenvector v. Since this multiplication can be thought as projecting the observations in A on the v, the difference measured is actually along the direction of v. This idea corresponds to estimate the distance from each observation to the hyperplane with normal vector v that passes through the zero point in the feature space. If we adjust the mean of each column in A to zero and its variance to one (the distribution of observations is centered to zero and the effect of the scale of features is eliminated), the hyperplanes which pass through the zero point can be used to describe the observations linearly. As shown in Fig. 2, when E is close to a zero vector, the observations will be all located at the positions near the hyperplane, which means this hyperplane can linearly describe the distribution of the observations. In this case, the corresponding eigenvalue, which is equal to $||E||^2$, is small. On the contrary, when the number of entries with large value in E becomes larger, as shown in Fig. 3, the observations will spread around the hyperplane rather than gather close to it. While such a hyperplane is used to represent the linear distribution of the observations, the error induced will be large, which can be easily captured by the corresponding $||E||^2$ or the eigenvalue. From those figures, the geometric relationship between the error vector E and the hyperplane with normal vector v can describe the linear dependency of features.

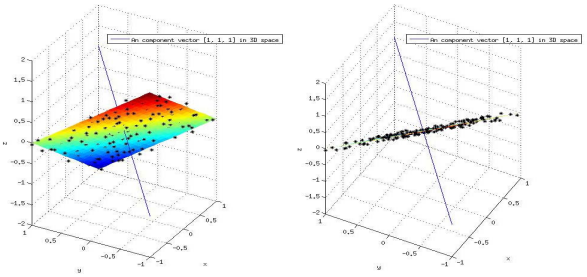

Figure 2. Left plot: the data points are clustered around a hyperplane with normal vector (1,1,1). Right plot: the same distribution of data points, but with a different view.

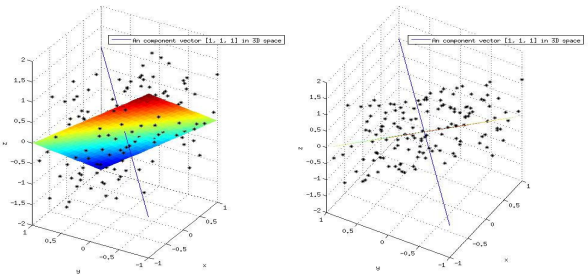

Figure 3. Left plot: the data points are located far away from a hyperplane with normal vector (1,1,1). Right plot: the same distribution of data points, but with a different view.

C. Feature Removal

In the previous sections, we have described how the dependency vector v can be obtained from eigenvector analysis. The next step is to utilize the information provided in v to evaluate the degree of redundancy of each feature so as to conduct the feature selection process. Considering that the greedy strategy taken in stepwise algorithms of PCA-based methods is not robust [20] without considering the effect of removal, we propose a method to efficiently model the effect induced at each iteration of selection to facilitate better decision of features to be removed.

Ideally if there exists perfect dependency between features, each eigenvector v can be seen as the normal vector of a hyperplene that presents such dependency, and consequently it is possible to obtain this following equation:

$$v_1 \cdot f_1 + v_2 \cdot f_2 + \ldots + v_n \cdot f_n = 0 \quad (8)$$

where f_i is the i^{th} column vector of A, and v_i is the i^{th} element of v. However, observations do not usually lie on the hyperplane perfectly. In the near linear-dependent case, an error vector shall be introduced to represent the distance of each observation to the hyperplane:

$$v_1 \cdot f_1 + v_2 \cdot f_2 + \ldots + v_n \cdot f_n = E \quad (9)$$

For an m-by-n data matrix A (n features for one observation and there are m observations), there will be n equations of hyperplanes obtained from n eigenvectors:

$$\begin{aligned}
v_{1,1} \cdot f_1 + v_{1,2} \cdot f_2 + \ldots + v_{1,n} \cdot f_n &= E_1 \\
v_{2,1} \cdot f_1 + v_{2,2} \cdot f_2 + \ldots + v_{2,n} \cdot f_n &= E_2 \\
&\vdots \\
v_{i,1} \cdot f_1 + v_{i,2} \cdot f_2 + \ldots + v_{i,n} \cdot f_n &= E_i \\
&\vdots \\
v_{n,1} \cdot f_1 + v_{n,2} \cdot f_2 + \ldots + v_{n,n} \cdot f_n &= E_n
\end{aligned}$$

Here, the idea of Gaussian Elimination is utilized to remove features in turn. At each iteration, we eliminate a feature with the smallest $||E||^2$, which reflects the most dependent relation. $||E||^2$ measures the degree of linear dependency characterized by the corresponding equation, and the equation with small $||E||^2$ represents a hyperplane with more points lying close to it. Suppose that we have the i^{th} equation whose $||E||_i^2$ is the smallest. In order to remove the dependency in the equation, one feature, for example, f_j is substituted by rewriting it as

$$f_j = -\left(v_{i,1} \cdot f_1 + v_{i,2} \cdot f_2 + \ldots + v_{i,n} \cdot f_n - E_i\right) / v_{i,j} \quad (10)$$

In order to preserve as much original information as possible, we prefer removing a feature with minimum effects on the other E's of other equations. Considering the case of substitution of f_j by equation (10), the remaining equations become

$$\begin{aligned}
v_{1,1}f_1 + v_{1,2}f_2 + \ldots - v_{1,j}\left(v_{i,1}f_1 + \ldots + v_{i,n}f_n - E_i\right)/v_{i,j} + \ldots + v_{1,n}f_n &= E_1 \\
v_{2,1}f_1 + v_{2,2}f_2 + \ldots - v_{2,j}\left(v_{i,1}f_1 + \ldots + v_{i,n}f_n - E_i\right)/v_{i,j} + \ldots + v_{2,n}f_n &= E_2 \\
&\vdots \\
v_{n,1}f_1 + v_{n,2}f_2 + \ldots - v_{n,j}\left(v_{i,1}f_1 + \ldots + v_{i,n}f_n - E_i\right)/v_{i,j} + \ldots + v_{n,n}f_n &= E_n
\end{aligned}$$

After transposition we obtain

$$\begin{aligned}
(v_{1,1} - v_{1,j} \cdot v_{i,1}/v_{i,j})f_1 + \ldots + (v_{1,n} - v_{1,j} \cdot v_{i,n}/v_{i,j})f_n &= E_1 - v_{1,j} \cdot E_i/v_{i,j} \\
(v_{2,1} - v_{2,j} \cdot v_{i,1}/v_{i,j})f_1 + \ldots + (v_{2,n} - v_{2,j} \cdot v_{i,n}/v_{i,j})f_n &= E_2 - v_{2,j} \cdot E_i/v_{i,j} \\
&\vdots \\
(v_{n,1} - v_{n,j} \cdot v_{i,1}/v_{i,j})f_1 + \ldots + (v_{n,n} - v_{n,j} \cdot v_{i,n}/v_{i,j})f_n &= E_n - v_{n,j} \cdot E_i/v_{i,j}
\end{aligned} \quad (11)$$

It is apparent that if f_j is the feature with largest $|v_{i,j}|$, substituting it will cause less effect on the other E's, since the term $E_i/v_{i,j}$ is smaller. Therefore, it is reasonable to choose the features with largest absolute coefficient in the selected equation for substitution. Now with the updated n-1 equations, the degree of the linear dependency needs to be re-estimated by ranking the equations above based on the updated error vector. To remove the next dependent feature, we can repeat the process by replacing the feature with largest absolute coefficient in the equation of the smallest newly-updated $||E||^2$. This substitution reduces the size of equations from n-1 to n-2. By repeating the process, the number of features continues to decrease. Note that each substitution requires to update all the coefficients of the remaining equations plus each E. Since the length of E is m, the inclusion of updating E's or not determines the complexity of a substitution to be O(nm) or $O(n^2)$, where m is the number of observation (fixed during the selection process); and n is the number of the features left, which is equivalent to the number of the remaining equations. We design the algorithm as an iterative procedure, and the pseudo code of it is described in Fig. 4. One can set the lower bound of the degree of linear dependency to decide when to stop. We present this constraint in our algorithm as a threshold of $||E||^2$. Those eigenvectors whose $||E||^2$ are smaller than or equal to the threshold are selected as a set of hyperplanes which describe dependent relations of features in the procedure of selection. As a result, the number of the features to be eliminated will depend on this threshold.

```
Function ufs(A, threshold)
// unsupervised feature selection
// A is a data matrix where each row contains an
// observation, and each column represents a feature
// threshold is set to the largest tolerant error (||E||_2^2)

// zero mean and unit standard deviation for each feature
A = normalize(A);
eigenvectors = get_eigenvectors(A^T A) ;
Eset = A * eigenvectors;

//suppose eigenvalues are arranged in ascending order
For each iteration
   If the smallest ||E||_2^2 of some E in Eset > threshold
      break;
   else
      find the eigenvector v with the smallest ||E||_2^2
      find the feature f with the largest absolute weight in v
      remove f by substitution and update Eset and the
         remaining eigenvectors
   end
end

return those remaining feature indexes
```

Figure 4. The backward unsupervised feature selection algorithm.

D. *Connection to PCA-based methods*

In this section, we provide a proof showing the connection between our proposed method and PCA-based methods. After standardizing A (with zero mean and unit variance for each column f_i of A), we can get the connection between our proposed method and PCA-based methods. Assume that the number of feature is n.

$$A^t A = \begin{bmatrix} f_1^t f_1 & f_1^t f_2 & \cdots \\ f_2^t f_1 & \vdots & \vdots \\ \vdots & \cdots & f_n^t f_n \end{bmatrix}$$

$$
= \begin{bmatrix}
\frac{\sum_{j=1}^{n}(f_{1j}-0)(f_{j1}-0)}{1\cdot 1} & \frac{\sum_{j=1}^{n}(f_{1j}-0)(f_{j2}-0)}{1\cdot 1} & \cdots \\
\frac{\sum_{j=1}^{n}(f_{2j}-0)(f_{j1}-0)}{1\cdot 1} & \ddots & \vdots \\
\vdots & \cdots & \frac{\sum_{j=1}^{n}(f_{nj}-0)(f_{jn}-0)}{1\cdot 1}
\end{bmatrix}
$$

$$
= \begin{bmatrix}
\frac{\sum_{j=1}^{n}(f_{1j}-\bar{f}_1)(f_{j1}-\bar{f}_1)}{std(f_1)\cdot std(f_1)} & \frac{\sum_{j=1}^{n}(f_{1j}-\bar{f}_1)(f_{j1}-\bar{f}_2)}{std(f_1)\cdot std(f_2)} & \cdots \\
\frac{\sum_{j=1}^{n}(f_{2j}-\bar{f}_2)(f_{j1}-\bar{f}_1)}{std(f_2)\cdot std(f_1)} & \ddots & \vdots \\
\vdots & \cdots & \frac{\sum_{j=1}^{n}(f_{nj}-\bar{f}_n)(f_{jn}-\bar{f}_n)}{std(f_n)\cdot std(f_b)}
\end{bmatrix}
$$

$$
= (n-1)\begin{bmatrix}
\rho_{f_1f_1} & \rho_{f_1f_2} & \cdots \\
\rho_{f_2f_1} & \ddots & \vdots \\
\vdots & \cdots & \rho_{f_nf_n}
\end{bmatrix} \tag{12}
$$

Equations (6) and (12) show that finding such near linear dependency relation is equivalent to doing eigen-decomposition on the correlation matrix of features of A. This is how PCA performs for analyzing the feature space of A.

The above observation exhibits that the eigenvectors used as the normal vectors of the hyperplanes to describe the observations are identical to the principal components in the feature space found by PCA (given features are transformed to zero mean and unit variance before PCA analysis). Considering that each eigenvector is equivalent to a principal component, we conduct a similar procedure as the method mentioned in [18] to choose the feature with the largest loading on the least important component to be removed. However, the major disadvantage of that method lies in the fact that they did not model the effect of each removal because to do that it is required to perform PCA operation in every iteration, which takes a lot of time. Consequently, as can be seen in the experimental section, their selection process does not perform as well as ours. Viewing the eigen-decomposition process from the viewpoint of dependency-modeling (rather the maximizing-variance point of view as PCA does) allows us to come up with a theoretically more sound solution in the feature removal process.

IV. EXPERIMENTS

Our proposed method is tested on four datasets. The first experiment is to test the ability of removing the near linear dependency in an artificial dataset, and then we try our method on three real-world datasets. For our methods, we experiment on two different settings: one uses the updating equation (11) and one does not. The one without updating is similar to the method motioned in [18] where features with largest loadings in least important principal components are removed sequentially. We also compare our method with a PCA-based method [12], and a clustering-based method [10].

A. Aritificial Dataset

This artificial dataset contains 500 rows of observations and 1000 columns of features. The first 500 columns are designed to be linearly independent and are randomly (but not necessarily equally) divided into 10 groups. As a result, these 10 groups have different number of columns of features, but nevertheless independent to each other. The next 500 columns are created by the linear combinations of columns within each group plus a small amount of noise. Thus for every group, there are some near linear dependent features. We would like to evaluate whether the dependent features can be removed first in the process of selection. To achieve such goal, we want to count how many of the first 500 removed features are independent to the remaining features. The best result a method can reach in this experiment is 0, which means all of the dependent features are removed before any independent feature is removed. We repeat this experiment 10 times and the experiment results are shown in Table 2. We can see that our method and PCA-based method can perform almost perfectly on this test, while Clustering-based method and our method without updating are not able to guarantee optimal solution in removing dependency.

	1	2	3	4	5	6	7	8	9	10	Avg
w/o updating	143	146	78	78	129	124	84	91	102	143	111.8
w/ updating	0	0	0	0	0	3	0	0	1	0	0.4
PCA	0	0	0	0	0	0	0	0	0	0	0
Clustering	123	95	146	77	117	60	127	47	164	119	107.5

Table 2. Number of remaining independent features after 500 removals in 10 experiments.

B. Real-World Datasets

We use three real-world datasets from UCI machine learning repository for experiments. The first two datasets are Arcene and Gisette, which were used in NIPS 2003 feature selection challenge. The last one is Arrhythmia. Originally, it contains 16 classes to specify which type of arrhythmia an observation belongs to, but we simplify the classification problem as a binary decision problem to only label each observation as "present" or "absent". Since there are some missing values in this dataset, we simply assign the mean of each dimension (feature) to those missing values. The number of observations and number of features about these datasets can be seen in Table 3.

	Number of Observations	Number of features	Note
Arcene	200	10000	2-fold cross validation
Gisette	7000	5000	5-fold cross validation
Arrhythmia	450	279	5-fold cross validation

Table 3. Real-world datasets

Different from the artificial datasets, in the real datasets it is not possible to identify completely independent feature sets for evaluation. Therefore in this experiment, we would like to evaluate whether the information can be retained after removing features. Since the goal of unsupervised feature selection methods is to retain as much information as possible while removing some redundant features, ideally its classification performance should not change too much after redundant features are removed. To demonstrate this, the performance under different number of selected features is drawn to exhibit how the performance changes as the number of selected features varies. Fig. 5 shows the results of two-fold cross validation of different feature selection methods on Arcene dataset, while Fig. 6 and Fig. 7 show the five-fold cross validation result of Gisette and Arrhythmia datasets. Note that the classifier package LibLinear [21] is used as the learning algorithm for classification.

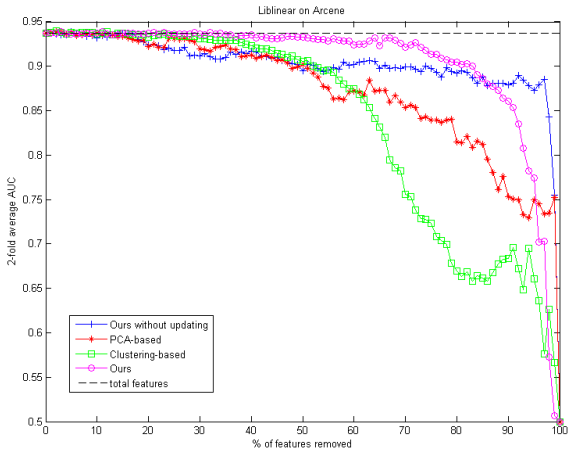

Figure 5. The two-fold average AUC curve of Arcene (blue + line: our method w/o updating, red * line: PCA-based, green square line: Clustering-based, pink circle line: our method w/ updating, and black dotted line: all features used)

According to the criterion that as the number of removed feature increases, the classification performance should not vary too much, the better method shall have the following properties: (1) have performance close to the original performance (all features are used), represented by the dotted line in the figures, (2) prevent a huge drop or improvement of performance (which indicates some non-replaceable features are just removed), and (3) hold such a trend (close to the dotted line) as long as possible while more features are removed. As shown in the figures, the green curve for clustering-based method falls much earlier than the other methods. The red curve for PCA-based method and our methods (blue for method without updating and pink for method with updating) have performances very close to the dotted line as the number of removed features is small. However, most of the cases our method stays closer to the dotted line for longer period than the other methods. Unlike clustering-based method, the performance of our method without updating even changes smoothly without any huge drops. Note that in our experiment, we use area under receiver operating characteristic (ROC) curve (AUC) as a measure of performance.

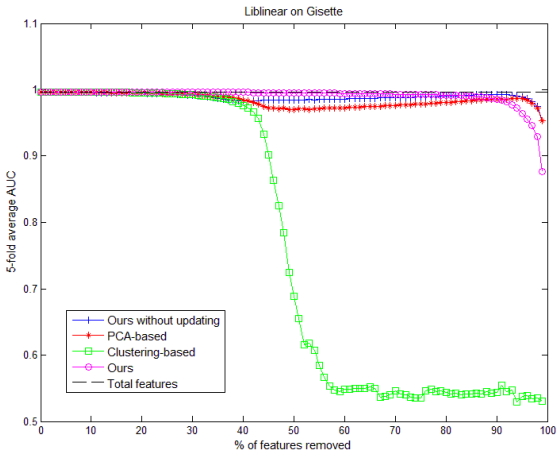

Figure 6. The five-fold average AUC curve of Gisette

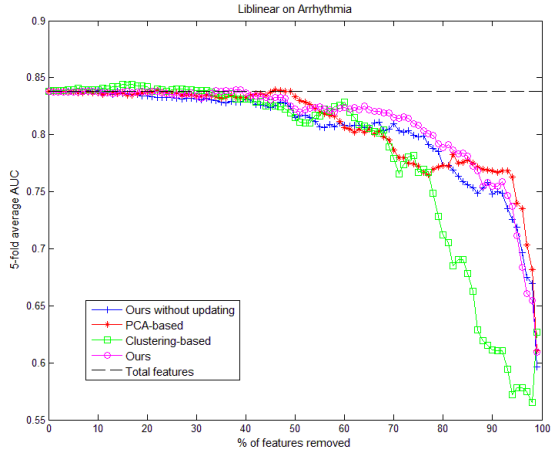

Figure 7. The five-fold average AUC of Arrhythmia.

Given the above criteria, the performance can be measured by area under offset curve of such chart. The method with smaller area is considered as a better one with respect to the criteria. We calculate the area covered by a given curve and the dotted line (i.e. performance without removal). To have better understanding of the results, we define Average Change of Area (ACA) as follows:

$$ACA(f,g) = \frac{\int |f(x) - g(x)|\, dx}{Range(x)}$$

ACA measures the area between two performance curves f(x) and g(x), and normalizes the area by the range of x. In our setting, f(x) is the AUC curve of one particular method, g(x) is the AUC curve when no feature is removed, and x is the percentage of removed features.

Some statistics are shown in Table 4. We consider the case when there is 0.01 to 0.04 change of performance (e.g. the AUC changes from 0.85 to 0.86 when considering 0.01 change of performance), the number of features are removed (the more the better) and the average change of area (the smaller the better). Smaller average change of area implies the performance curve for the method satisfies the desired properties mentioned above. As shown in Table 4, generally our method can select a subset of features without affecting the original performance significantly comparing with the other methods. That is, the average change of area is small for most of the cases. Also, the percentage of features removed given same performance change for our method is generally larger than those of the other methods. This indicates that our unsupervised feature selection method with updating can perform better in removing near-dependent features. Note that our method without updating can still attain comparable or even better results than PCA-based method in Arcene and Giestte datasets.

performance change	0.01		0.02		0.03		0.04		Avg	
Arcene										
	PFR	ACA	PFR	ACA	PFR	ACA	PFR	ACA	PFR	ACA
w/o updating	19%	**0.0023**	28%	0.0066	46%	0.0139	50%	0.0154	35.75%	0.0096
w/ updating	**60%**	0.0028	**75%**	**0.0046**	**79%**	**0.0057**	**84%**	**0.0074**	**74.50%**	**0.0051**
PCA-based	20%	0.0029	32%	0.0062	45%	0.0108	48%	0.0121	36.25%	0.0080
Clustering-based	38%	0.003	45%	0.005	49%	0.0067	54%	0.0092	46.50%	0.0060
Gisette										
w/o updating	35%	0.0027	**98%**	0.0071	**99%**	0.0075	**99%**	0.0075	82.75%	0.0062
w/ updating	**90%**	**0.0021**	94%	**0.0028**	95%	**0.0031**	96%	**0.0035**	**94%**	**0.0029**
PCA-based	39%	0.0027	44%	0.0043	**99%**	0.0129	**99%**	0.0129	70.25%	0.0082
Clustering-based	37%	0.0025	40%	0.0036	42%	0.0048	44%	0.0069	40.75%	0.0045
Arrhythmia										
w/o updating	36%	0.0036	50%	0.0058	55%	0.0075	77%	0.0147	54.50%	0.0079
w/ updating	49%	**0.0021**	**70%**	0.0063	**76%**	0.008	**79%**	0.0092	**69%**	0.0064
PCA-based	**53%**	0.0025	57%	**0.0036**	60%	**0.0048**	69%	**0.0089**	60%	**0.0050**
Clustering-based	44%	0.0032	50%	0.0047	64%	0.0083	69%	0.0104	57%	0.0067

Table 4. The statistics of the performance curves of different methods when there is 0.01 to 0.04 change of performance. (PFR: Percentage of Features Removed. ACA: Average Change of Area).

V. CONCLUSION

We propose an unsupervised feature selection method to remove the dependent features. Experiment results show that our method performs significantly better than the other competitive methods in terms of removing the dependent (or redundant) features and retaining the information. Geometrically, our proposed method tries to find a hyperplane with normal vector equal to the principal component of a data matrix (when features in the data matrix are transformed to zero mean and unit variance) such that most data points lie close to the hyperplane, implying the existence of dependent or redundant features. We also demonstrate the connection between our method and PCA-based method. Moreover, our backward feature selection method feeds back the effect of each removal of feature on the remaining features with complexity about max(O(nm), $O(n^2)$), which outperforms the PCA-based methods under some circumstances.

REFERENCES

[1] I. Guyon, and A. Elisseeff, "An Introduction to Variable and Feature Selection," in The Journal of Machine Learning Research, 2003.

[2] D. Koller, and M. Sahami, "Toward Optimal Feature Selection," in 13th International Conference on Machine Learning, 1996, pp. 284-292.

[3] K. Kira, and L. A. Rendell, "The Feature Selection Problem: Traditional Methods and a New Algorithm," in In Proc. AAAI-92, San Jose, CA, 1992, pp. 129-134.

[4] L. Yu, and H. Liu, "Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution," in Proc. 20th International Conference on Machine Learning, 2003, pp. 856-863.

[5] J. G. Dy, and C. E. Brodley, "Feature Selection for Unsupervised Learning," in Journal of Machine Learning Research, 2004, pp. 845-889.

[6] P. H., L. F., and D. C., "Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy," in IEEE Transactions on Pattern and Machine Intelligence, 2005, pp. 1226-1238.

[7] P. Mitra, C. A. Murthy, and S. K. Pal, "Unsupervised Feature Selection Using Feature Similarity," in IEEE Transcations on Pattern Analysis and Machine Intelligence, 2002, pp. 301-312.

[8] N. S.-. Madsen, C. Thomsen, and J. M. Pena, "Unsupervised Feature Subset Selection," in Workshop on Probabilistic Graphical Models for Classification, pp. 71-82.

[9] C. Ding, and H. C. Peng, "Minimum Redundancy Feature Selection from Microarray Gene Expression Data," in Proc. Second IEEE Computational Systems Bioinformatics Conf., 2003, pp. 523-528.

[10] P. Krizek, J. Kittler, and V. Hlavac, "Feature Condensing Algorithm for Feature Selection," in International Conference on Pattern Recognition, 2008.

[11] G. P. McCabe, "Principal Variables," in Technometrics, 1984, pp. 137-144.

[12] J. A. Cumming, and D. A. Wooff, "Dimension Reduction via Principal Variables," in Computational Statistics & Data Analysis, 2007, pp. 550-565.

[13] M. A. Hall, "Correlation-Based Feature Selection for Discrete and Numeric Class Machine Learning," in Proc. 17th International Conference on Machine Learning, 2000, pp. 359-366.

[14] R. Kohavi, and G. H. John, "Wrappers for Feature Subset Selection," in Artificial Intelligence, 1997, pp. 273-324.

[15] D. W. Aha, and R. L. Bankert, "A Comparative Evaluation of Sequential Feature Selection Algorithms," in Proc. 5th International Workshop on Aritificial Intelligence and Statistics, 1995, pp. 1-7.

[16] L. R., R. R., and T. M., "Genetic Algorithms as a Strategy for Feature Selection," in J. Chemometrics, 1992, pp. 267-281.

[17] S. Wold, K. Esbensen, and P. Geladi, "Principal Component Analysis," in Chemometrics and Intell. Lab. Sys., 1987.

[18] I. T. Jolliffe, "Discarding Variables in a Principal Component Analysis I: Artificial Data," in Applied Statistics, Journal of the Royal Statistical Society, 1972, pp. 160-163.

[19] W. J. Krzanowski, "Selection of Variables to Preserve Multivariate Data Structure Using Principal Components," in Applied Statistics, 1987, pp. 22-33.

[20] J. Cadima, and I. T. Jolliffe, "Loading and Correlations in the Interpretation of Principal Components," in J. Appl. Statist, 1995, pp. 203-214.

[21] E.-E. Fan, K.-W. Chang, C.-J. Hsieh *et al.*, "LIBLINEAR: A Library for Large Linear Classification," in Journal of Machine Learning Research, 2008, pp. 1871-1874.