Constraint Score: A New Filter Method for Feature Selection with Pairwise Constraints

Daoqiang Zhang[1], Songcan Chen[1] and Zhi-Hua Zhou[2]

[1] Department of Computer Science and Engineering

Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

[2] National Key Laboratory for Novel Software Technology

Nanjing University, Nanjing 210093, China

Abstract

Feature selection is an important preprocessing step in mining high-dimensional data. Generally, supervised feature selection methods with supervision information are superior to unsupervised ones without supervision information. In the literature, nearly all existing supervised feature selection methods use class labels as supervision information. In this paper, we propose to use another form of supervision information for feature selection, i.e. pairwise constraints, which specifies whether a pair of data samples belong to the same class (*must-link* constraints) or different classes (*cannot-link* constraints). Pairwise constraints arise naturally in many tasks and are more practical and inexpensive than class labels. This topic has not yet been addressed in feature selection research. We call our pairwise constraints guided feature selection algorithm as Constraint Score and compare it with the well-known Fisher Score and Laplacian Score algorithms. Experiments are carried out on several high-dimensional UCI and face data sets. Experimental results show that, with very few pairwise constraints, Constraint Score achieves similar or even higher performance than Fisher Score with full class labels on the whole training data, and significantly outperforms Laplacian Score.

Keywords: Feature selection; Pairwise constraints; Filter method; Constraint score; Fisher score; Laplacian score

1 Introduction

With the rapid accumulation of high-dimensional data such as digital images, financial time series and gene expression microarrays, feature selection has been an important preprocessing step to machine learning and data mining. In many real-world applications, feature selection has shown very effective in reducing dimensionality, removing irrelevant and redundant features, increasing learning accuracy, and enhancing learning comprehensibility [11][16][20]. Typically, feature selection methods can be categorized into two groups, i.e., 1) filter methods [20] and 2) wrapper methods [15]. The filter methods evaluate the goodness of features by using the intrinsic characteristics of the training data and are independent on any learning algorithm. On the contrary, the wrapper methods directly use predetermined learning algorithms to evaluate the features. Generally, the wrapper methods outperform the filter methods in terms of accuracy, but the former are computationally more expensive than the latter. When dealing with data with huge number of features, the filter methods are usually adopted due to their computational efficiency. In this paper, we are particularly interested in the filter methods.

According to whether the class labels are used, feature selection methods can be divided into supervised feature selection [11] and unsupervised feature selection [7-8]. The former evaluates feature relevance by the correlation between features and class labels, while the latter evaluates feature relevance by the capability of keeping certain properties of the data, e.g., the variance or the locality preserving ability [12-13]. When labeled data are sufficient, supervised feature selection methods usually outperform unsupervised feature selection methods [3]. However, in many cases obtaining class labels is expensive and the amount of labeled training data is often very limited. Most traditional supervised feature selection methods may fail on such ‘small labeled-sample problem’ [14]. A recent important advance on this direction is to use both labeled and unlabeled data for feature selection, i.e. semi-supervised feature selection [23], which introduces the popular semi-supervised learning technique [25] into feature selection research. However, like in supervised feature selection, the supervision information used in semi-supervised feature selection is still class labels.

In fact, besides class labels, there exist other forms of supervision information, e.g. the *pairwise constraints*, which specifies whether a pair of data samples belongs to the same class

(*must-link* constraints) or different classes (*cannot-link* constraints) [1-2][21]. Pairwise constraints arise naturally in many real-world tasks, e.g. image retrieval [1]. In those applications, considering the pairwise constraints is more practical than trying to obtain class labels, because the true labels may be unknown *a priori*, while it can be easier for a user to specify whether some pairs of examples belong to the same class or not, i.e. similar or dissimilar. Besides, the pairwise constraints can be derived from labeled data but not vice versa. Finally, unlike class labels, the pairwise constraints can sometimes be automatically obtained without human intervention. For those reasons, pairwise constraints have been widely used in distance metric learning [19] and semi-supervised clustering [1-2][25]. In one of our recent work, we have proposed to use pairwise constraints for dimension reduction [21].

It's worthy to note that one should neither confuse the pairwise constraints mentioned in this paper with the pairwise similarity or dissimilarity value used in spectral graph based algorithms [6][18][22][24], nor with some class pairwise methods [9]. In spectral graph based algorithms, one first computes the pairwise similarity or dissimilarity between samples to form the similarity or dissimilarity matrix, and then perform subsequent operations on it. On the other hand, in class pairwise methods, e.g. class pairwise feature selection [9], one takes the subsets of features which are the most effective in discriminating between all possible pairs of classes. Apparently, both are very different from the pairwise constraints mentioned in this paper.

In this paper, we propose to use pairwise constraints for feature selection. To the best of our knowledge, we haven't noticed any similar work on this topic before. We devise two novel score functions based on pairwise constraints to evaluate the feature goodness and name the corresponding algorithms as *Constraint Score*. Experiments are carried out on several high-dimensional UCI and face data sets to compare the proposed algorithm with established feature selection methods such as Fisher Score [3] and Laplacian Score [12], etc. Experimental results show that, with a few pairwise constraints, Constraint Score achieves similar or even higher performance than Fisher Score with full class labels on the whole training data, and significantly outperforms Laplacian Score.

The rest of this paper is organized as follows. Section 2 first introduces the background of this paper and briefly shows several existing score functions used in supervised and unsupervised feature selection. Then we present the Constraint Score algorithm in Section 3. Section 4 reports

on the experimental results. Finally, Section 5 concludes this paper with some future work.

2 Background

In this section, we briefly introduce several score functions popularly used in feature selection methods, including Variance [3], Laplacian Score [12] and Fisher Score [3]. Among them, Variance and Laplacian Score are unsupervised, while Fisher Score is supervised.

Variance might be the simplest unsupervised evaluation of the features. It uses the variance along a dimension to reflect its representative power and those features with the maximum variance are selected. Let f_{ri} denote the *r*-th feature of the *i*-th sample $\mathbf{x}_i$, *i* = 1,…,*m*; *r* = 1,…,*n*. Define $\mu_r = \frac{1}{m}\sum_i f_{ri}$. Then, the Variance score of the *r*-th feature V_r, which should be maximized, is computed as follows [3]:

$$V_r = \frac{1}{m}\sum_{i=1}^{m}\left(f_{ri} - \mu_r\right)^2 \tag{1}$$

Another unsupervised feature selection method, i.e. Laplacian Score, makes a further step on Variance. It not only prefers to those features with larger variances which have more representative power, but also prefers to selecting features with stronger locality preserving ability. A key assumption in Laplacian Score is that data from the same class are close to each other. The Laplacian score of the *r*-th feature L_r, which should be minimized, is computed as follows [12]:

$$L_r = \frac{\sum_{i,j}\left(f_{ri} - f_{rj}\right)^2 S_{ij}}{\sum_i \left(f_{ri} - \mu_r\right)^2 D_{ii}} \tag{2}$$

Where D is a diagonal matrix with $D_{ii} = \sum_j S_{ij}$, and S_{ij} is defined by the neighborhood relationship between samples $\mathbf{x}_i$ (i=1,…,*m*) as follows:

$$S_{ij} = \begin{cases} e^{-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{t}}, & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are neighbors} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Where *t* is a constant to be set, and '$\mathbf{x}_i$ and $\mathbf{x}_j$ are neighbors' means that either $\mathbf{x}_i$ is among *k* nearest neighbors of $\mathbf{x}_j$, or $\mathbf{x}_j$ is among *k* nearest neighbors of $\mathbf{x}_i$.

In contrast to Variance and Laplacian score, Fisher Score is supervised with class labels and it

seeks features with best discriminant ability. Let n_i denote the number of samples in class *i*. Let μ_r^i and $(\sigma_r^i)^2$ be the mean and variance of class *i*, *i* = 1,…,*c*, corresponding to the *r*-th feature. The Fisher score of the *r*-th feature F_r, which should be maximized, is computed as follows [3]:

$$F_r = \frac{\sum_{i=1}^{c} n_i \left(\mu_r^i - \mu_r\right)^2}{\sum_{i=1}^{c} n_i (\sigma_r^i)^2} \tag{4}$$

3 Constraint Score

In this paper, we formulate the pairwise constraints guided feature selection as follows: Given a set of data samples $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m]$, and some supervision information in the form of pairwise must-link constraints $M = \left\{ \left(\mathbf{x}_i, \mathbf{x}_j\right) \middle| \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ belong to the same class} \right\}$ and pairwise cannot-link constraints $C = \left\{ \left(\mathbf{x}_i, \mathbf{x}_j\right) \middle| \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ belong to different classes} \right\}$, use the supervision information in pairwise constraints in M and C to find the most relevant feature subsets from the original *n* features of $\mathbf{X}$.

Let f_{ri} denote the *r*-th feature of the *i*-th sample $\mathbf{x}_i$, *i* = 1,…,*m*; *r* = 1,…,*n*. To evaluate the score of the *r*-th feature using the pairwise constraints in C and M, we define two different score functions by minimizing C_r^1 and C_r^2:

$$C_r^1 = \frac{\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in M} \left(f_{ri} - f_{rj}\right)^2}{\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in C} \left(f_{ri} - f_{rj}\right)^2} \tag{5}$$

$$C_r^2 = \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in M} \left(f_{ri} - f_{rj}\right)^2 - \lambda \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in C} \left(f_{ri} - f_{rj}\right)^2 \tag{6}$$

The intuition of Eqs.(5) and (6) is simple and natural. That is, we want to select features with the best constraint preserving ability. More concrete, if there is a must-link constraint between two data samples, a ‘good’ feature should be the one on which those two samples are close to each other; on the other hand, if there is a cannot-link constraint between two data samples, a ‘good’ feature should be the one on which those two samples are far away from each other. Both Eq. (5) and Eq. (6) realize feature selection according to features’ constraint preserving ability. In Eq. (6),

there is a regularization coefficient λ, whose function is to balance the contributions of the two terms in Eq. (6). Since the distance between samples in the same class is typically smaller than that in different classes, we set $\lambda < 1$ in this paper.

In the rest of this paper, we call feature selection algorithms based on the score functions in Eqs (5) and (6) as **Constraint Score**. To be specific, we denote algorithm using Eq. (5) as **Constraint Score-1**, and denote algorithm using Eq. (6) as **Constraint Score-2**. The whole procedure of the proposed Constraint Score algorithm is summarized in **Algorithm 1** as below.

Algorithm 1: Constraint Score

Input: Data set $\mathbf{X}$, pairwise constraints set M and C, λ (for **Constraint Score-2** only)

Output: The ranked feature list

Step 1: For each of the *n* features, compute its constraint score using Eq. (5) (for **Constraint Score-1**) or Eq. (6) (for **Constraint Score-2**);

Step 2: Rank the features according to their constraint scores in ascending order.

We can also give an alternative explanation on the constraint score functions in Eqs. (5) and (6) from the spectral graph theory [5]. First, we construct two graphs G^M and G^C both with *m* nodes, using the pairwise constraints in M and C, respectively. In both graphs, the *i*-th node corresponds to the *i*-th sample $\mathbf{x}_i$. For graph G^M, we put an edge between node *i* and *j* if there is a must-link constraint between samples $\mathbf{x}_i$ and $\mathbf{x}_j$ in M. Similarly, for graph G^C, we put an edge between node *i* and *j* if there is a cannot-link constraint between samples $\mathbf{x}_i$ and $\mathbf{x}_j$ in C. Once the graphs G^M and G^C are constructed, their weight matrices, denoted by S^M and S^C, respectively, can be defined as:

$$S_{ij}^M = \begin{cases} 1, & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in M \text{ or } (\mathbf{x}_j, \mathbf{x}_i) \in M \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$$S_{ij}^C = \begin{cases} 1, & \text{if } (\mathbf{x}_i, \mathbf{x}_j) \in C \text{ or } (\mathbf{x}_j, \mathbf{x}_i) \in C \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Define $\mathbf{f}_r = [f_{r1}, f_{r2}, ..., f_{rm}]^T$, and let D^M and D^C be diagonal matrices with $D_{ii}^M = \sum_j S_{ij}^M$ and $D_{ii}^C = \sum_j S_{ij}^C$. Then, compute the Laplacian matrices [5] as $L^M = D^M - S^M$ and $L^C = D^C - S^C$. According to Eqs. (7) and (8), we get

$$\begin{aligned}
&\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in M} \left(f_{ri} - f_{rj} \right)^2 = \sum_{i,j} \left(f_{ri} - f_{rj} \right)^2 S_{ij}^M \\
&= \sum_{i,j} \left(f_{ri}^2 + f_{rj}^2 - 2 f_{ri} f_{rj} \right)^2 S_{ij}^M \\
&= \sum_{i,j} f_{ri}^2 S_{ij}^M + \sum_{i,j} f_{rj}^2 S_{ij}^M - 2 \sum_{i,j} f_{ri} S_{ij}^M f_{rj} \\
&= 2 \mathbf{f}_r^T D^M \mathbf{f}_r - 2 \mathbf{f}_r^T S^M \mathbf{f}_r \\
&= 2 \mathbf{f}_r^T L^M \mathbf{f}_r
\end{aligned} \tag{9}$$

Similarly, we have

$$\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in C} \left(f_{ri} - f_{rj} \right)^2 = \sum_{i,j} \left(f_{ri} - f_{rj} \right)^2 S_{ij}^C = 2 \mathbf{f}_r^T L^C \mathbf{f}_r \tag{10}$$

From Eqs. (9) and (10), neglecting the constant 2, Eqs. (5) and (6) change into:

$$C_r^1 = \frac{\mathbf{f}_r^T L^M \mathbf{f}_r}{\mathbf{f}_r^T L^C \mathbf{f}_r} \tag{11}$$

$$C_r^2 = \mathbf{f}_r^T L^M \mathbf{f}_r - \lambda \mathbf{f}_r^T L^C \mathbf{f}_r \tag{12}$$

The detailed procedure of the spectral graph based Constraint Score algorithm is summarized in **Algorithm 2** as below.

Algorithm 2: Constraint Score (spectral graph version)

Input: Data set $\mathbf{X}$, pairwise constraints set M and C, λ (for **Constraint Score-2** only)

Output: The ranked feature list

Step 1: Construct the graphs G^M and G^C from M and C respectively;

Step 2: Calculate the weight matrices S^M and S^C using Eqs. (7) and (8), and compute the Laplacian matrices L^M and L^C;

Step 3: Compute the constraint score of the *r*-th feature using Eq. (11) (for **Constraint Score-1**) or Eq. (12) (for **Constraint Score-2**);

Step 4: Rank the features according to their constraint scores in ascending order.

It is noteworthy that although Algorithm 2 outputs the same result as Algorithm 1, introducing the spectral graph theory can bring some additional advantages. First, it brings us a unified framework from which we can connect Constraint Score with other feature selection methods, e.g. Laplacian Score. Second, we can easily extend Algorithm 2 for semi-supervised feature selection which uses unlabeled data together with pairwise constraints for feature selection, by defining appropriate graphs and corresponding weight matrices in Algorithm 2. The detailed description regarding semi-supervised feature selection is beyond the scope of this paper and we will discuss it in another paper.

Now we analyze the time complexity of both Algorithm 1 and Algorithm 2. First, we assume the number of pairwise constraints used in Constraint Score is l, which is bounded by $0 < l < \mathrm{O}(m^2)$. Algorithm 1 has two parts: (1) Step 1 evaluates the *n* features needing $\mathrm{O}(nl)$ operations; (2) Step 2 ranks *n* features needing $\mathrm{O}(n \log n)$ operations. Hence, the overall time complexity of Algorithm 1 is $\mathrm{O}(n \max(l, \log n))$. Algorithm 2 has three parts: (1) Steps 1-2 build the graph matrices using pairwise constraints, requiring $\mathrm{O}(m^2)$ operations; (2) Step 3 evaluates the *n* features based on the graphs, requiring $\mathrm{O}(nm^2)$ operations; (3) Step 4 ranks features in ascending order in terms of constraint scores, requiring $\mathrm{O}(n \log n)$ operations. Thus, the overall time complexity of Algorithm 2 is $\mathrm{O}(n \max(m^2, \log n))$. When the number of constraints is very large, i.e. $l = \mathrm{O}(m^2)$, both algorithms have the same time complexity. However, in practice, usually only a few constraints are sufficient, i.e. $l << \mathrm{O}(m^2)$, then Algorithm 1 will be more efficient than Algorithm 2.

4 Experiments

In this section, we evaluate the performance of our proposed Constraint Score algorithms on several high-dimensional UCI repository of machine learning databases [4] including *Ionoshpere*, *Sonar*, *Soybean*, and *Wine*, and on two well-known face databases: ORL [17] and YaleB [10]. For each data set, we choose the first half of samples from each class as the training data, and the remaining data for testing. In our experiments, we simulated the generation of pairwise constraints

as follows: We randomly select pairs of samples from the training data and create must-link or cannot-link constraints depending on whether the underlying classes of the two samples are the same or different. The results of Constraint Score are averaged over 100 runs with different generation of constraints. Algorithm 1 is adopted to compute the constraint score. The parameter in Constraint Score-2 is always set to $\lambda = 0.1$ if without extra explanations. We empirically find that under that value, Constraint Score-2 can achieve satisfying performance in most data sets used in this paper.

We compare Constraint Score-1 and Constraint Score-2 with existing unsupervised feature selection methods Variance and Laplacian Score, as well as supervised feature selection method Fisher Score. Note that Variance and Laplacian Score learn the feature scores without using the class labels of training data, while Fisher Score use the full class labels on training data. As a comparison, Constraint Score is between Laplacian Score and Fisher Score. That is, it use supervision information a bit more than Laplacian Score, but much fewer than Fisher Score. The performances of all algorithms are measured by the classification accuracy using selected features on testing data. In all experiments, the nearest neighborhood (1-NN) classifier with Euclidean distance is employed for classification, after feature selection with the above algorithms. In our experiments, feature selection is performed by selecting the first *d* features from the ranking list of features generated by different algorithms, where *d* is the desired number of selected features specified by user.

Table 1. Statistics of the UCI data sets

Data sets	Size	Dimension	# of classes
Ionosphere	351	34	2
Sonar	208	60	2
Soybean	47	35	4
Wine	178	13	3

4.1 Results on UCI data sets

First, we test the five algorithms on the four UCI data sets, whose statistics are given in Table 1. Figure 1 shows the plots for accuracy vs. different numbers of selected features. For each data set, a total of 10 pairwise constraints including 5 must-link and 5 cannot-link constraints are used

in both Constraint Score-1 and Constraint Score-2. This amount of constraints is very small compared with the total amount of possible constraints that can be generated from training data. Taking *Ionosphere* as an example, there are over 10, 000 possible pairwise constraints that can be generated from training data, while our algorithms only use 10 constraints to learn the feature scores. The other three algorithms evaluate features using the whole training data, which is computationally more expensive than our algorithms.

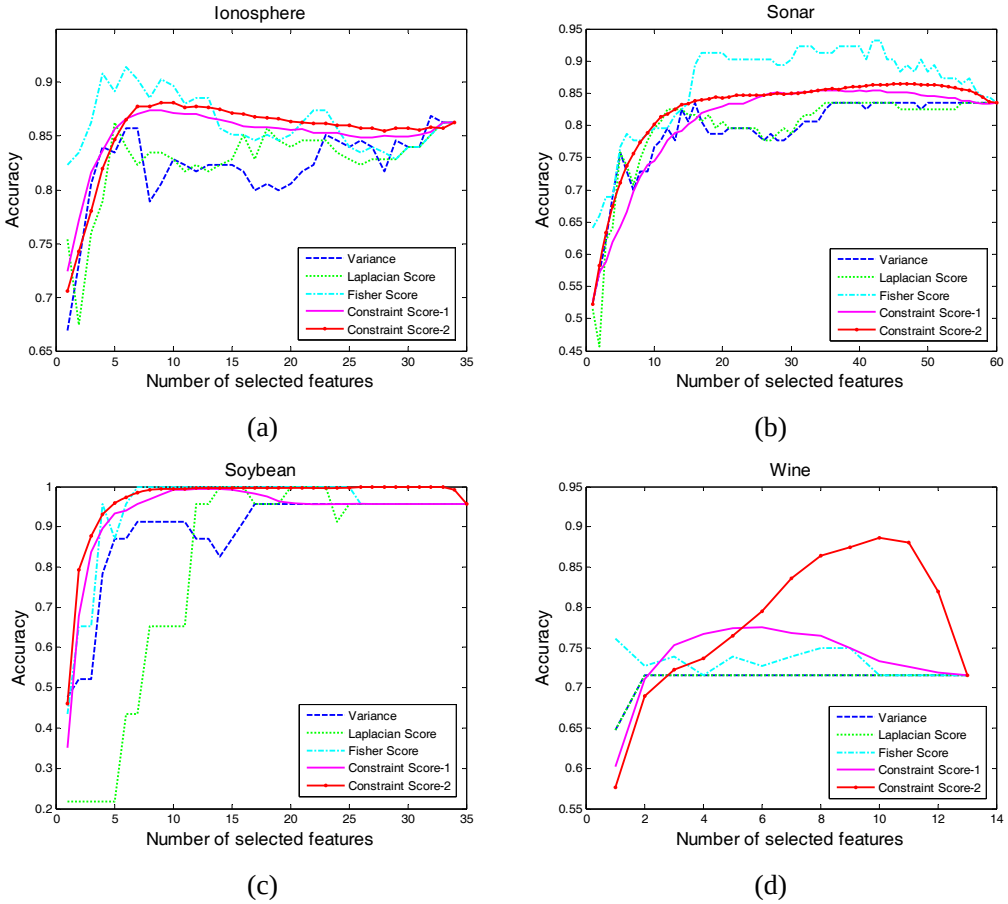


Fig. 1. Accuracy vs. different numbers of selected features on 4 UCI data sets: (a) on *Ionosphere*, (b) on *Sonar*, (c) on *Soybean*, (d) on *Wine*.

Table 2. Averaged accuracy of different algorithms on UCI data sets

Data sets	Variance	Laplacian Score	Fisher Score	Constraint Score-1	Constraint Score-2
Ionosphere	82.2 ± 3.8	82.6 ± 3.6	86.3 ± 2.5	85.1 ± 2.9	85.4 ± 3.8
Sonar	79.3 ± 6.3	79.5 ± 7.2	86.4 ± 6.9	80.7 ± 7.8	82.5 ± 6.9
Soybean	88.9 ± 12.7	79.4 ± 28.4	94.5 ± 12.1	93.5 ± 11.6	96.7 ± 9.7
Wine	71.1 ± 1.9	71.1 ± 1.9	73.2 ± 1.6	73.5 ± 4.6	78.2 ± 9.2

Figure 1 indicates that, in most cases, the performances of Constraint Score-1 and Constraint Score-2 are comparable to Fisher Score, both significantly better than that of Variance and Laplacian Score. This verifies that supervision information is very useful in learning feature scores. Table 2 compares the averaged accuracy under different number of selected features. Here the values after the symbol '±' denote the standard deviation. From Table 2 and Fig. 1 we can find that, the performance of Constraint Score-2 is almost always better than that of Constraint Score-1 and is comparable with Fisher Score. More specifically, Constraint Score-2 is superior to Fisher Score on *Soybean* and *Wine*, and is inferior on *Ionosphere* and *Sonar*. It is surprising to see that on *Wine*, as the number of selected features increases, Constraint Score-2 achieves significantly better performance than other algorithms.

To uncover the underlying reason, we reinvestigate the *Wine* data carefully. As shown in Table 1, the *Wine* data has 13 dimensions, and a further observation is that each sample has a much larger value on the 13th dimension (feature) than on other dimensions (features). For example, the first sample of the *Wine* data is [14.23,1.71,2.43,15.6,127,2.8,3.06,0.28,2.29,5.64,1.04,3.92,1065]. Since we adopt the 1-NN classifier with Euclidean distance, if the 13th feature and the other features are simultaneously selected, the final result will be dominated by the 13th feature. So the key is its order appearing in the ranking list of features. We find that Variance, Laplacian Score and Fisher Score always rank the 13th feature at the head place, while Constraint Score-2 nearly always ranks the 13th feature at the rear place. That is, the 13th feature is nearly always selected by Variance, Laplacian Score and Fisher Score no matter what the desired number of selected features is, while it is selected by Constraint Score-2 only when the desired number of selected features is near to the number of dimensions of the data. This partially explains the results whosn in Fig. 1(d).

To investigate the influence of the numbers of constraints on the performances of the algorithms, we replace the 10 constraints in Constraint Score-1 and Constraint Score-2 with 4 constrains and 40 constraints respectively, and the results are plotted in Fig. 2. As shown in the figure, generally, increasing the number of constraints improves the accuracy. But the performances of both algorithms don't drop rapidly when reducing the number of constraints. It is impressive to see from Fig. 2 that both algorithms can still remain a good accuracy with only 4 constraints. Fig. 2 also shows that under the same number of constraints, Constraint Score-2 is

superior to Constraint Score-1. The averaged accuracy under different number of selected features of Constraint Score with different numbers of constraints is summarized in Table 3.

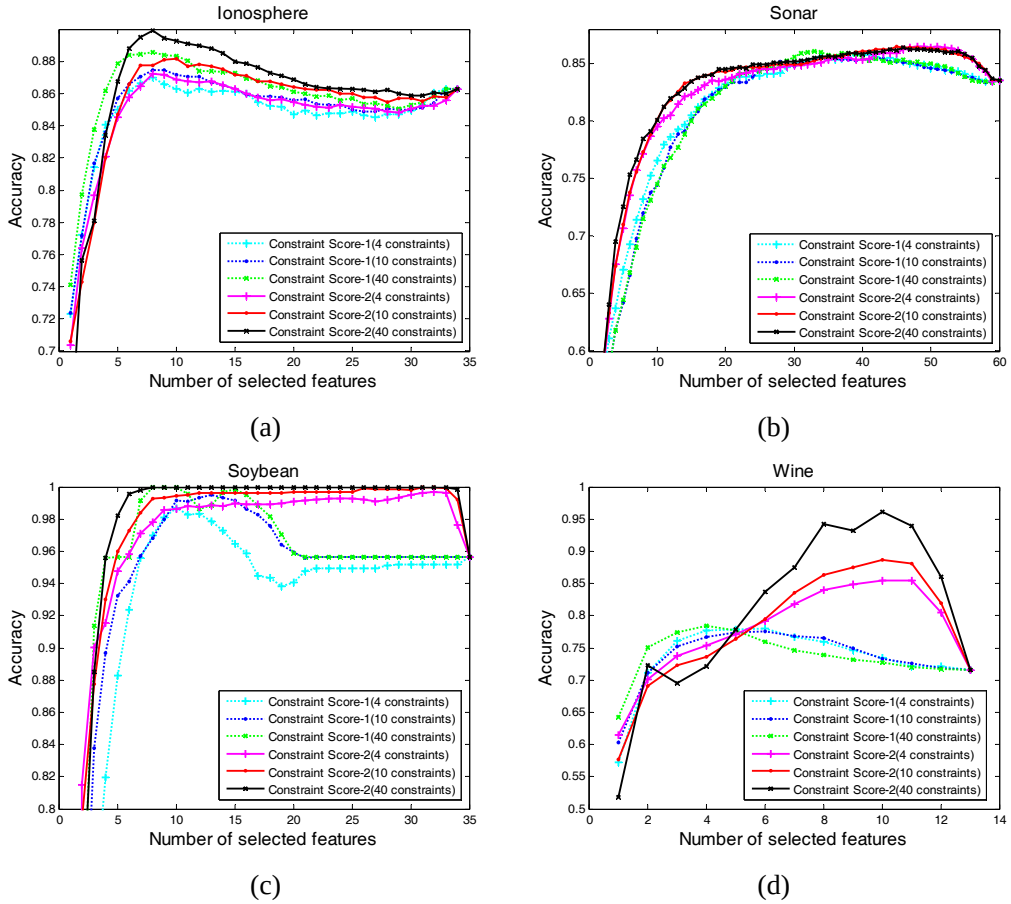


Fig. 2. Accuracy vs. different numbers of selected features and different levels of constraints on 4 UCI data sets: (a) on *Ionosphere*, (b) on *Sonar*, (c) on *Soybean*, (d) on *Wine*.

Table 3. Averaged accuracy of Constraint Score with different numbers of constraints on UCI data sets

Data sets	Constraint Score-1			Constraint Score-2		
	4 constraints	10 constraints	40 constraints	4 constraints	10 constraints	40 constraints
Ionosphere	84.8 ± 2.8	85.1 ± 2.9	86.0 ± 2.7	84.8 ± 3.3	85.4 ± 3.8	86.0 ± 4.7
Sonar	81.1 ± 7.3	80.7 ± 7.8	80.7 ± 8.1	82.2 ± 6.9	82.5 ± 6.9	82.6 ± 6.7
Soybean	91.9 ± 11.7	93.5 ± 11.6	94.4 ± 11.5	96.3 ± 8.6	96.7 ± 9.7	97.0 ± 10.5
Wine	73.4 ± 5.5	73.5 ± 4.6	73.8 ± 3.7	77.7 ± 7.2	78.2 ± 9.2	80.8 ± 13.0

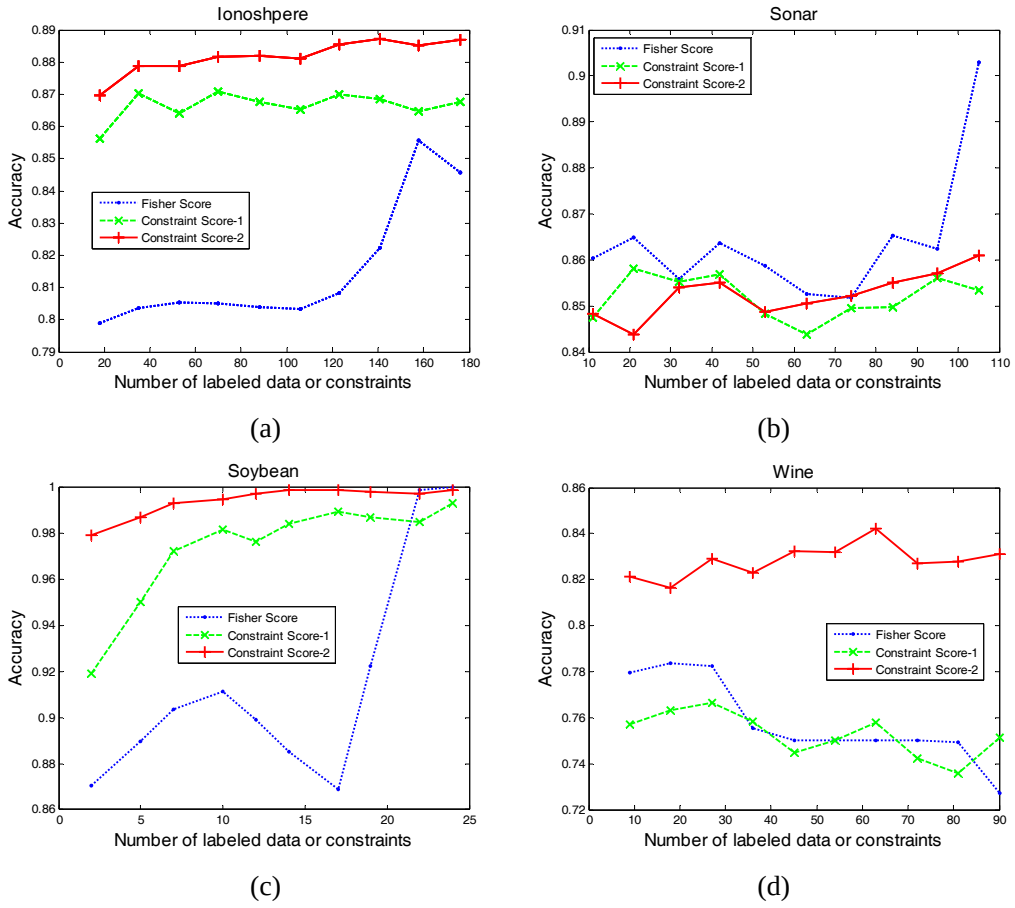


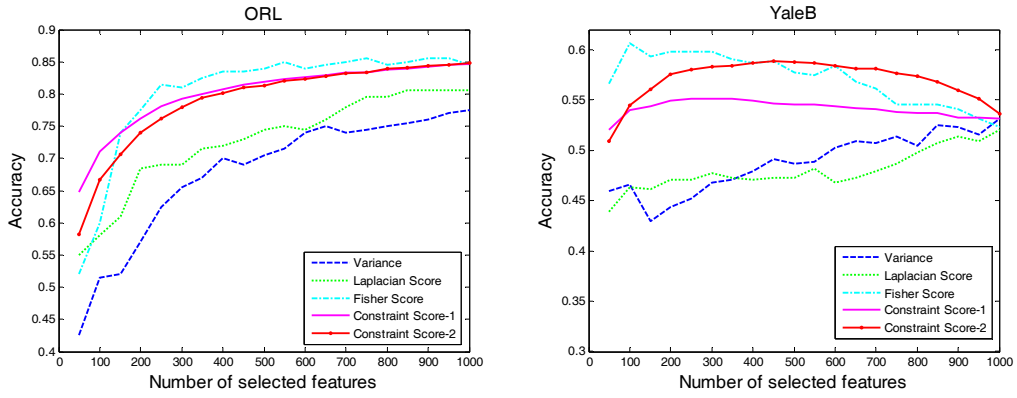
Fig. 3. Accuracy vs. different numbers of labeled data (for Fisher Score) or pairwise constraints (for Constraint Score) on 4 UCI data sets: (a) on *Ionosphere*, (b) on *Sonar*, (c) on *Soybean*, (d) on *Wine*.

Then, we compare the performances of Constraint Score-1 and Constraint Score-2 with that of Fisher Score when different levels of supervision are used. Fig. 3 shows the plots for accuracy under desired number of selected features vs. different numbers of labeled data (for Fisher Score) or pairwise constraints (for Constraint Score) on the 4 UCI data sets. Here the desired number of selected features is chosen as half of the original dimension of samples. For Fisher Score with a certain number of labeled data, we randomly sample the training data and the results are averaged over 100 runs. As shown in Fig. 3, except on *Sonar*, Constraint Score-2 is much better than the other two algorithms especially when only a few labeled data or constraints are used. Constraint Score-1 is superior to Fisher score on *Ionosphere* and *Soybean*. On *Sonar*, both Constraint Score-1 and Constraint Score-2 are inferior to Fisher Score. A closer study on Fig. 3 reveals that, generally, the accuracy of Constraint Score-2 (and partially Constraint Socre-1) increases fast in the beginning (with few constraints) and slows down at the end (with relatively more constraints). It

implies that too many constraints won't help too much to further boost the accuracy, and only a few constraints are required in Constraint Score. While Fisher Score typically requires relatively more labeled data to obtain a satisfying accuracy, as is shown in Fig. 3.

4.2 Results on face databases

In this section, we test the proposed Constraint Score algorithms on face databases with huge number of features, usually over 1, 000 features. Specifically, we do experiments on two well-known face databases, ORL [17] and YaleB [10]. The ORL database contains images from 40 individuals, each providing 10 different images. The YaleB database contains a total of 640 images including 64 frontal pose images of 10 different subjects. We crop the original images of ORL and YaleB to the size of 64x64 and 32x32 respectively. We use the original pixel intensity values as the features, so a face in ORL and YaleB databases is respectively represented by a 4096-dimensional and a 1024-dimensional feature vector. Also, we use a total of 10 pairwise constraints including 5 must-link and 5 cannot-link constraints in both Constraint Score-1 and Constraint Score-2.



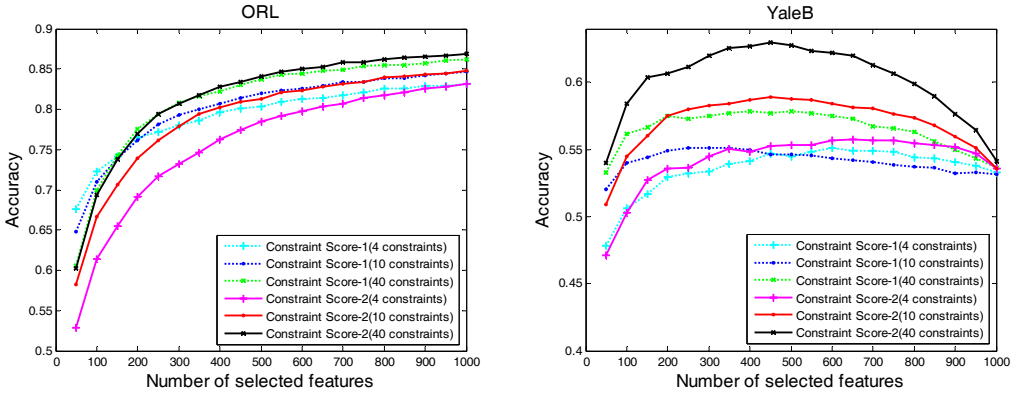
(a) (b)

Fig.4. Accuracy vs. different numbers of selected features on ORL (a) and on YaleB (b) face databases.

Table 4. Averaged accuracy of different algorithms on face databases

Data sets	Variance	Laplacian Score	Fisher Score	Constraint Score-1	Constraint Score-2
ORL	67.9 ± 9.9	72.8 ± 7.6	80.4 ± 8.9	80.2 ± 5.2	79.0 ± 6.9
YaleB	48.8 ± 2.9	48.0 ± 2.0	57.1 ± 2.5	54.2 ± 0.8	57.0 ± 2.1

Figure 4 shows the plots for accuracy vs. different numbers of selected features on ORL and on YaleB face databases. Table 4 gives the averaged accuracy under different numbers of selected features of different algorithms on face databases. As shown in Fig. 4 and Table 4, Constraint Score-2 and Constraint Score-1 have approximate performance to Fisher Score and both significantly outperform Variance and Laplacian Score. That verifies again the usefulness of supervision information in feature selection.



(a) (b)

Fig.5. Accuracy vs. different numbers of selected features and different levels of constraints on ORL (a) and on YaleB (b) face databases.

Table 5. Averaged accuracy of Constraint Score with different numbers of constraints on face databases

Data sets	Constraint Score-1			Constraint Score-2		
	4 constraints	10 constraints	40 constraints	4 constraints	10 constraints	40 constraints
ORL	79.3±4.1	80.2±5.2	81.3±6.5	75.7±8.1	79.0±6.9	81.6±6.9
YaleB	53.6±1.8	54.2±0.8	56.5±1.4	54.2±2.1	57.0±2.1	60.2±2.8

Figure 5 shows the plots for accuracy vs. different numbers of selected features and different levels of constraints on ORL and on YaleB face databases. Table 5 summarizes the averaged accuracy under different number of selected features of Constraint Score with different numbers of constraints. Like before, there are 3 levels of constraints, i.e. 4 constraints, 10 constraints and 40 constraints. Comparing Fig. 5 with Fig. 2, we notice the same tendency in two figures. That is, increasing the number of constraints improves the accuracy and vice versa, which is more apparent on YaleB than on ORL. Fig. 5 also shows that no matter under what number of

constraints, Constraint Score-2 is always superior to Constraint Score-1 on YaleB. But on ORL, it is inferior to Constraint Score-1 when there are only a few constraints, but outperform the latter when the number of constraints increases.

4.3 Further discussion

In this section, we discuss some properties of Constraint Score-1 and Constraint Score-2 further with respect to feature normalization, unbalanced number of constraints and parameter selection.

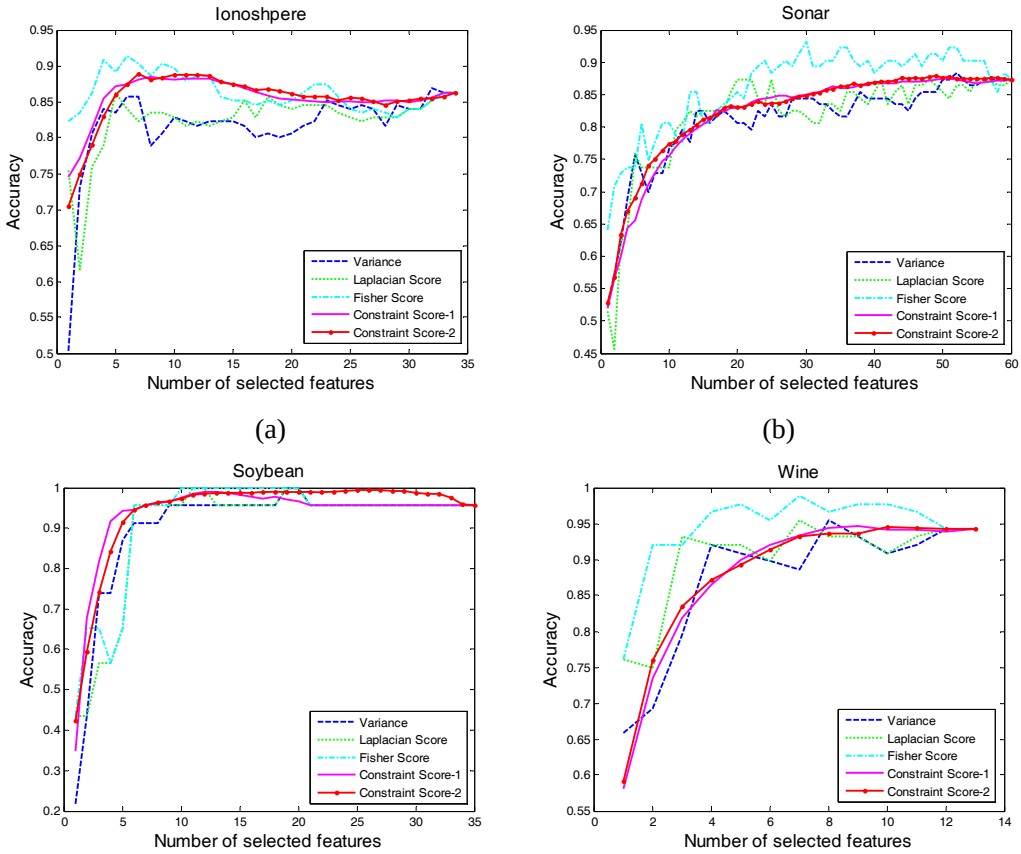
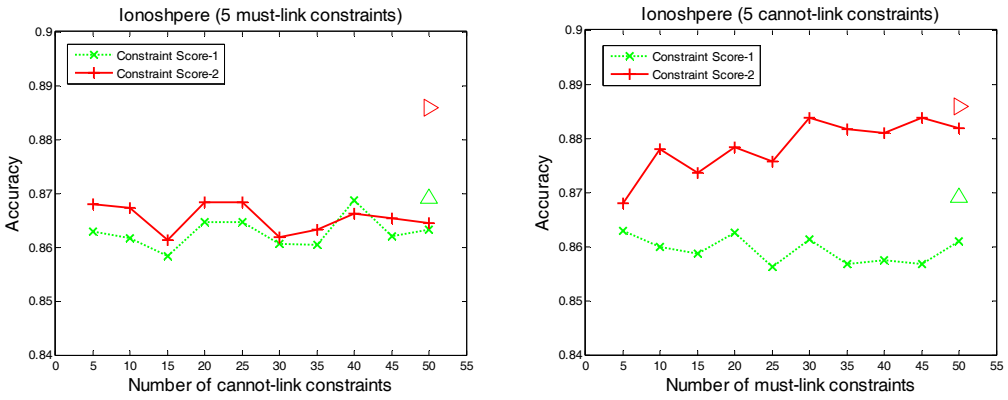


Fig. 6. Accuracy vs. different numbers of selected features on 4 normalized UCI data sets: (a) on *Ionosphere*, (b) on *Sonar*, (c) on *Soybean*, (d) on *Wine*.

First, we study the influence of feature normalization on feature selection algorithms. In previous experiments we have shown the performances of different algorithms on the original data sets without normalization, e.g. in Figs. 1 and 4. For comparison, Fig. 6 presents the plots for accuracy vs. the number of selected features of different algorithms on 4 normalized UCI data sets.

Here the 'normalization' is performed through dividing each feature value by the maximum value of all samples on that feature. Comparing Fig. 6 with Fig. 1, we can see that when differences between scales of feature values are large (e.g. on *Wine*), normalization significantly improves most algorithms' accuracies. On the other hand, when feature values have similar scales, the differences between algorithms with and without normalization are not so much, as shown in Fig. 6 (a)-(c). For ORL and YaleB face data sets, because all feature values are with the same scale, i.e., pixel gray intensity from 0 to 255, normalization is not needed. Fig. 6 also indicates that no matter without or with feature normalization, both Constraint Score-1 and Constraint Score-2 achieve competitive performances in most cases. Generally, in contrast to other methods, Constraint Score is more robust to the difference in the scale of features.

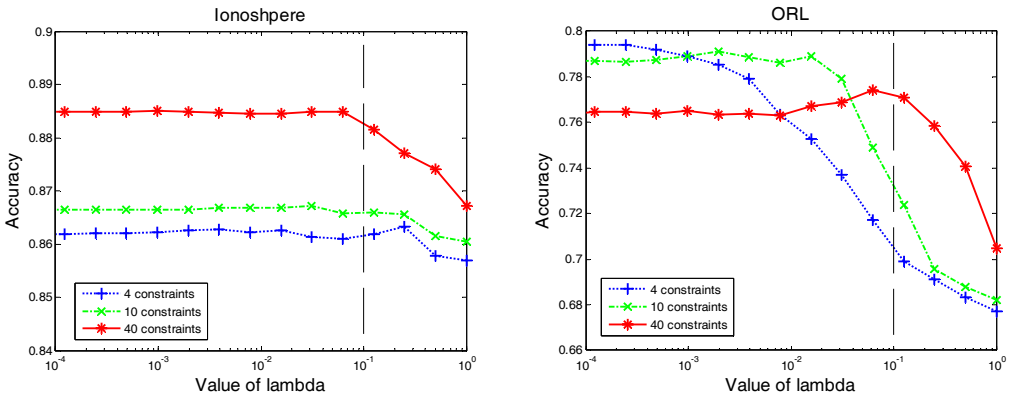


(a) (b)

Fig. 7. Performance on *Ionosphere* under unbalanced number of constraints: (a) The number of cannot-link constraints is more than that of must-link constraints; (b) The number of must-link constraints is more than that of cannot-link constraints. In both figures, the red right triangle and the green up triangle denote the respective accuracies of Constraint Score-2 and Constraint Score-1 with both 50 must-link constraints and 50 cannot-link constraints.

Then, we study the influence of unbalanced number of constraints, i.e., the number of must-link constraints is not equal to the number of cannot-link constraints, on the performances of Constraint Score-1 and Constraint Score-2. Fig. 7 shows a typical result on *Ionosphere*. Specifically, Fig. 7 (a) plots the accuracy vs. different number of cannot-link constraints (from 5 to 50) with fixed number of must-link constraints (5), while Fig. 7 (b) plots the accuracy vs. different number of must-link constraints (from 5 to 50) with fixed number of cannot-link constraints (5). We also show the accuracies of Constraint Score-1 and Constraint Score-2 with both 50 must-link

constraints and 50 cannot-link constraints for reference. Fig. 7 (a) shows that with fixed number of must-link constraints, increasing the number of cannot-link constraints only could not improve the accuracies of both Constraint Score-1 and Constraint Score-2. On the other hand, Fig. 7 (b) indicates that with fixed number of cannot-link constraints, increasing the number of must-link constraints only could not improve accuracy of Constraint Score-1 but could improve accuracy of Constraint Score-2. This suggests that for Constraint Score-2, must-link constraints are more important than cannot-link constraints.



(a) (b)

Fig. 8. Accuracy vs. different values of lambda in Constraint Score-2 on *Ionosphere* (a) and on ORL (b) data sets.

Finally, we study the influence of the value of the parameter λ on the performance of Constraint Score-2. In previous experiments, λ is always set to 0.1 for simplicity. Under that value, Constraint Score-2 has shown competitive performance in most cases. However, it is expected that an appropriate choice of λ can further improve the performance of Constraint Score-2. Fig. 8 plots the accuracy vs. λ values on *Ionosphere* and ORL data sets. As what was expected, $\lambda = 0.1$ is not the optimal value. For example, Fig. 8 (b) shows that for Constraint Score-2 with 10 constraints, more than 5 percent increase on accuracy could be achieved if some other settings such as $\lambda = 0.01$ is adopted. However, choosing the appropriate values for λ is in general difficult, as it depends on not only the data set but also the number of constraints, as revealed by Fig. 8.

5 Conclusion

In this paper, we propose a new filter method for feature selection based on pairwise constraints. To the best of our knowledge, this may be the first work to introduce pairwise constraints for feature selection. Two new score functions are proposed to evaluate features based on their constraints preserving power. Experimental results on four UCI data sets and two face databases show that with only a small number of constraints, both proposed algorithms achieve approximate performances to Fisher Score using full labeled data, and significantly outperforms Laplacian Score. Also, the proposed algorithms don't have to access the whole training data and have computational advantage on large-size data sets. Finally, because in many real applications obtaining pairwise constraints is much easier than obtaining class labels, our algorithms may have great potentials in those applications.

The main concern of this paper is to investigate the usefulness of pairwise constraints in feature selection. At the current stage, we learn feature scores using only the constraints. It is interesting to see whether we can further improve accuracy by introducing unlabeled data, similar as in semi-supervised feature selection where both labeled and unlabeled data are used for feature selection. In our experiments, the pairwise constraints are randomly generated from training data and have no contradiction, so investigating actively obtaining more informative constraints and testing our algorithms with inconsistent constraints are also interesting future work.

Acknowledgements

We want to thank the anonymous reviewers for their helpful comments and suggestions. This work is supported by National Science Foundation of China under Grant Nos. 60473035, 60505004 and 60635030, the Jiangsu Science Foundation under Grant No. BK2006521, the Foundation for the Author of National Excellent Doctoral Dissertation of China under Grant No. 200343, and the National High Technology Research and Development Program of China under Grant No. 2007AA01Z169.

References

[1] A. Bar-Hillel, T. Hertz, N. Shental and D. Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6:937–965, 2005.

[2] S. Basu. *Semi-supervised Clustering: Probabilistic Models, Algorithms and Experiments*. PhD thesis, Department of Computer Sciences, University of Texas at Austin, 2005.

[3] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[4] C. Blake, E. Keogh, and C.J. Merz. *UCI repository of machine learning databases*. [http://www.ics.uci.edu/~mlearn/MLRepository.html], Department of Information and Computer Science, University of California, Irvine, 1998.

[5] Fan R. K. Chung. *Spectral graph theory*. AMS, 1997.

[6] T. Cour, F. Benezit and J. Shi. Spectral Segmentation with Multiscale Graph Decomposition. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:1124-1131, 2005.

[7] J. G. Dy, C. E. Brodley, A. C. Kak, L. S. Broderick, and A. M. Aisen. Unsupervised feature selection applied to content-based retrieval of lung images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:373-378, 2003.

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

[9] S. Essid, G. Richard and B. David. Muscial Instrument Recognition by pairwise classification strategies. *IEEE Transactions on Audio, Speech and Language Processing*, 14(4):1401-1412, 2006

[10] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23: 643–660, 2001.

[11] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157-1182, 2003.

[12] X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. In: *Advances in Neural Information Processing Systems*, 17, MIT Press, Cambridge, MA, 2005.

[13] X. He, and P. Niyogi. Locality preserving projections. In: *Advances in Neural Information Processing Systems*, 16, MIT Press, Cambridge, MA, 2004.

[14] A. Jain and D. Zongker. Feature selection: Evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153-158, 1997.

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

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

[17] F. Samaria, and A. Harter. Parameterisation of a Stochastic Model for Human Face Identification. In: *Proceedings of 2nd IEEE Workshop on Applications of Computer Vision*, Sarasota FL, December 1994

[18] J. Shi and Malik J. Normalized cuts and image segmentation. *IEEE Trans Pattern Analysis and Machine Intelligence*, 22(8): 888-905, 2000.

[19] E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In: *Advances in Neural Information Processing Systems*, 15:505-512, MIT Press, Cambridge, MA, 2003.

[20] L. Yu and H. Liu. Feature selection for high-dimensional data: a fast correlation-based filter solution. In: *Proceedings of the 20th International Conferences on Machine Learning*, Washington DC, 2003.

[21] D. Zhang, Z.H. Zhou and S. Chen. Semi-supervised dimensionality reduction. In: *Proceedings of the 7th SIAM International Conference on Data Mining*, Minneapolis, MN, 2007.

[22] Y. Zhao, T. Wang, P. Wang, and Y. Du. Scene Segmentation and Categorization Using NCuts. In: *Proceedings of 2nd International Workshop on Semantic Learning Applications in Multimedia*, In association with CVPR 2007, Minneapolis, MN, 2007.

[23] Z. Zhao and H. Liu. Semi-supervised feature selection via spectral analysis. In: *Proceedings of the 7th SIAM International Conference on Data Mining*, Minneapolis, MN, 2007.

[24] Z. Zheng, F. yang, W. Tan, J. Jia and J. Yang. Gabor feature-based face recognition using supervised locality preserving projection. *Signal Processing*, 87(10): 2473-2483, 2007.

[25] X. Zhu. *Semi-supervised learning literature survey*. Tech. Report 1530, Department of Computer Sciences, University of Wisconsin at Madison, Madison, WI, 2006. http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf.