Data Management Support via Spectrum Perturbation-based Subspace Classification in Collaborative Environments

Chao Chen and Mei-Ling Shyu
Department of Electrical and Computer Engineering
University of Miami
Coral Gables, FL, USA
Email: c.chen15@umiami.edu, shyu@miami.edu

Shu-Ching Chen
School of Computing and Information Sciences
Florida International University
Miami, FL, USA
Email: chens@cs.fiu.edu

***Abstract*—Data management support to enable effective and efficient information sharing in collaborative environments is critical, especially in semantics based search and retrieval. In this paper, a novel spectrum perturbation-based subspace classification is proposed to mine semantics and other useful information from a large-scale dataset by utilizing a lower-dimensional subspace to discriminate different classes of the dataset. Among the existing subspace-based approaches, the principal component (PC) subspace is the most prevailing one and has been well studied. After investigating previous work related to PC subspace, we found that none of them had considered the perturbation on spectrum when building the subspace learning models. However, such perturbation is of certain importance and is able to provide discriminant information that helps improve classification performance by measuring the closeness of each testing data instance towards a subspace model by a closeness score based on the spectrum perturbation. Each testing data instance is assigned to its closest class by searching the smallest closeness score. Experiments are conducted to evaluate our proposed subspace classifier using data sets from three different sources, and the experimental results show that it achieves promising results and outperforms comparative subspace classifiers as well as some other commonly used classifiers.**

***Index Terms*—Collaborative environment, Principal component (PC) subspace, spectrum perturbation, classification, closeness score.**

I. Introduction

One of the advantages of a collaborative environment, where people are working closely together towards a common goal, lies in its efficiency that an individual environment may not be able to offer. In a large scale collaborative information system, people may search, gather, analyze, and share information frequently from text documents, images, and/or videos. Many issues arise when searching and sharing information from these data. First, due to people's subjectivity, the content is described and represented differently by the end users. Therefore, the sharing of information stays mainly at the raw data level, such as text, picture, and video, and it is hard to share information at a higher level, such as the semantics information within the images or videos. Second, intensive human effort is usually required before searching and sharing the information, e.g., the indexing and annotation of images and videos. There is a demand to find a way to efficiently search and retrieve desired information to be shared within a collaborative environment.

In response to such demands, data mining techniques including classification, association rule mining, decision tree, regression, clustering, etc. have shown their potentials [1][2][3][4][5][6][7][8][9]. The idea of applying data mining techniques is to automatically discover knowledge from a large amount of data so as to reduce expensive human efforts. For example, when indexing a large collection of images or videos, classification methods are of great help to efficiently retrieve the interested content. Some data mining techniques have already been successfully deployed in collaborative environments [10]. An example is collaborative filtering which can be regarded as a classification/regression task. In [11], a classification method was used to recommend products according to the user's past purchasing behavior. [4] proposed a collaborative approach for document clustering. In this paper, our focus is on developing an effective and efficient classification framework to support data management within a collaborative environment.

With the advance and prosperity in the areas of data mining and machine learning as well as the successful deployment in practical real-world applications, numerous classification algorithms have been proposed, including Support Vector Machine, Neural Network, Decision Tree, Bayes Network, etc. [12]. Among them, the subspace classification has also been studied and developed. Although the analysis on subspace modeling can be traced back to as early as in [13], it is not until recent decades that more and more subspace classifiers were developed and utilized in various applications like letter recognition face recognition, etc. [14][15][16][17].

As far as the subspace classification approach is concerned, each class is represented and modeled by a subspace. The subspace of each class may overlap with each other or can be mutually independent. The dimension of the subspace is usually much lower than that of the original space. The commonly adopted classification rule of these subspace classification algorithms is to assign the label of a class whose

COLLABORATECOM 2011, October 15-18, Orlando, United States

DOI 10.4108/icst.collaboratecom.2011.247202

subspace has the smallest distance to a testing data instance. Since each class is modeled in a lower-dimensional subspace, subspace classification suffers less from the so-called "curse of dimensionality" problem that has been commonly seen when dealing with high-dimensional datasets.

One of the most prevailing subspaces adopted for modeling and classification is the principal component (PC) subspace. The PC subspace has a significant property that the bases of the PC subspace are orthogonal to each other and the attributes in the PC subspace are mutually uncorrelated. Moreover, by only removing zero eigenvalues and corresponding eigenvectors, the information within the original space will not be lost while reducing the dimension of the data. Please note that the eigenvectors and the bases of principal component subspace are actually the same. Therefore, principal component subspace can compactly represent the data instances and thus a classification algorithm could potentially benefit from such a compact data representation. However, the eigenvectors are quite sensitive to the data from which they are derived, so any outliers can easily change the principal components. Previous studies mostly focus on the outliers' negative impacts on Principal Component Analysis (PCA) [18]. Indeed, the variance caused by the outliers will significantly change the eigenvectors and may affect the performance of the subspace classifiers. Nevertheless, with regard to multi-class classification, the diverse extents of sensitivities to outliers that are inherent in the eigenvectors of different classes may be served as the key information for the classification purpose.

This motivates us to develop a novel SPectrum peRtUrbation based ClassifiEr called SPRUCE in this paper. The proposed classifier uses the perturbation information on the spectra of all classes to build the subspace models and perform the classification task. Experimental results based on cross-validation evaluation on datasets from three different sources reveal that SPRUCE outperforms some other comparative classification algorithms. The effectiveness of SPRUCE enables efficient search, retrieval, and sharing of semantics information in a collaborative environment.

The paper is organized as follows. Related work is reviewed in Section II. The detailed and formal introduction of the spectrum perturbation-based subspace classifier is presented in Section III. Section IV shows the experimental results and analyses. Finally, Section V concludes the paper and discusses some future work.

II. Related Work

There are a few branches in linear PC subspace classification. Please note that the nonlinear PC subspace is out of the scope of this paper. The first branch investigates the reconstruction error of an input instance in terms of different subspaces and searches for the class to which the input instance has the minimum reconstruction error. A typical representative algorithm of this branch is Class-Dependent PCA or CD-PCA [19]. Suppose $\Phi^{(j)} = \{\Phi_1^{(j)}, ..., \Phi_\eta^{(j)}\}$ is the transformation matrix and $\overline{\mu}^{(j)} = \{\overline{\mu}_1^{(j)}, ..., \overline{\mu}_\nu^{(j)}\}$ is the center of class j, and the reconstruction error for instance $y = \{y_1, ..., y_\nu\}$ is defined as shown in Equation (1).

$$\Delta^{(j)} = \|y - y \cdot \Phi^{(j)} \cdot \Phi^{(j)^T}\|. \quad (1)$$

The classification rule is to assign the label of the class for which $\Delta^{(j)}$ is minimum. To improve the classification performance, PC selection method can be applied to decrease the value of η in transformation matrix. Bischof et al. [20] further proposed a robust version to cope with missing features and/or outliers in the dataset.

Since PCA is essentially an unsupervised learning algorithm and is not built for the classification purpose, it is not surprising to notice that the result of classification accuracy is poor. To overcome such a weakness, some PCA algorithms [21], such as those that will be mentioned in the second and third branches, incorporate class information in their learning models in order to improve the classification performance.

The second branch takes PCA as a preprocessing step to reduce the dimension of the data instances and later other learning and classification algorithms are employed to perform the classification task. Zhao et al. [22] proposed a method that combines PCA and LDA (Linear Discriminant Analysis) in the face recognition area. While not using PCA plus LDA, Park et al. [23] used a two-stage PCA to include the class label information into the features. After the two-stage transformation, any classification algorithm, such as Nearest neighbor classifier and Support Vector machine, can be employed to perform the classification task. The proposed Class-Augmented PCA algorithm (CA-PCA) in [23] first applied PCA on the training data $X_{original}$ using Equation (2).

$$X = X_{original} \cdot W_{PCA}. \quad (2)$$

Later, the class information $C(X)$ is augmented into X by constructing a larger matrix $X^a = [X, C(X)]$. Another transformation matrix W_a is derived from X^a and it can be written as $W_a = [W_{input}, W_{class}]$. The composite transformation matrix is then defined as shown in Equation (3).

$$W_{CA-PCA} = W_{PCA} \cdot W_{input}. \quad (3)$$

The transformed data $X^{'}$ using the composite transformation matrix, as shown in Equation (4), will resort to some classification algorithms to predict the class labels.

$$X^{'} = X_{original} \cdot W_{CA-PCA}. \quad (4)$$

The third branch of linear PC subspace classification algorithms rely on Gaussian latent variable models to obtain a probabilistic formulation of PCA [24]. The representative algorithm of this branch is supervised probabilistic PCA (SPPCA) proposed in [25]. In SPPCA, each observed data is represented by (x, y), where x is a data instance with features and $y \in \{+1, -1\}$ is the class label. (x, y) is generated from the latent variable model as follows.

$$x = W_x z + \mu_x + \varepsilon_x \quad (5)$$

$$y = f(z, \Theta) + \varepsilon_y \quad (6)$$

where W_x is the transformation matrix, $f(z,\Theta)$ is the deterministic function with parameter Θ, $z \sim \mathcal{N}(0,\mathbf{I})$, $\varepsilon_x \sim \mathcal{N}(0,\sigma_x^2\mathbf{I})$, and $\varepsilon_y \sim \mathcal{N}(0,\sigma_y^2\mathbf{I})$. An EM (Expectation-Maximization) Learning algorithm was proposed for SPPCA model to retrieve the parameters and latent variables. The projected data in the PC subspace is then applied some classification algorithms, just like CA-PCA.

Different from the second and third branches, our proposed classification algorithm does not treat the class label as one attribute during model learning. The class label only acts as an indicator of a class and does not have any numeric or ordinal meaning. In addition, the proposed algorithm does not depend on other classification algorithms to perform the classification task. As will be seen in the next section, it has good scalability and flexibility when deployed in various cases.

III. Spectrum perturbation-based subspace classifier

Our proposed spectrum perturbation-based subspace classifier utilizes perturbation on the spectra to discriminate classes in a dataset. Some important definitions and useful corollary are first introduced.

A. Definition and Corollary

Suppose class S has k data instances and m attributes so that $S = \{X_1,...,X_k\}^T$. Each data instance X_i is a vector of m-dimension, $X_i=\{x_{i1},...,x_{im}\}$, $i \in [1,k]$. The center of class S is denoted by $\mu=\{\mu_1,...,\mu_m\}$. Let $\mathbf{1}$ be a column vector full of 1s. Please note that we are not going to discriminate $\mathbf{1}$ with different lengths unless necessary. The eigenvectors, which are related to positive eigenvalues $\lambda = \{\lambda_1,...,\lambda_p | \lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p > 0, p \leq m\}$ of $(S-\mathbf{1}\cdot\mu)^T(S-\mathbf{1}\cdot\mu)$, are $EV=\{EV_1,...,EV_p\}$.

Definition 1. *The spectra of S in terms of EV and μ are defined as:*

$$SPA(S,\mu,EV) = diag\{EV^T \cdot (S-\mathbf{1}\cdot\mu)^T \cdot (S-\mathbf{1}\cdot\mu)\cdot EV\} \quad (7)$$

where $diag\{.\}$ function returns a row vector of the diagonal elements of a matrix. It is easy to see that the spectra of S here are just the same as eigenvalues λ.

Definition 2. *If there is a new data instance $\theta=\{\theta_1,...,\theta_m\}$ that has the same attributes as S, the* spectrum perturbation *of θ on S is defined as :*

$$PTB(\theta,S,\mu,EV) = SPA(S\cup\{\theta\},\mu,EV) - SPA(S,\mu,EV) \quad (8)$$

Usually, class S is quite large, and a direct calculation of spectrum perturbation using Definition 2 is rather expensive. Therefore, an efficient way of calculating the perturbation is proposed in *Corollary* 1.

Corollary 1. *If there is a new data instance $\theta=\{\theta_1, ..., \theta_m\}$ that has the same attributes as S, the perturbation of θ can also be rewritten as:*

$$PTB(\theta,S,\mu,EV) = SPA(\theta,\mu,EV). \quad (9)$$

Proof:

PTB(θ, S, μ, EV)
= $diag\{EV^T \cdot (S\cup\theta - \mathbf{1}\cdot\mu)^T \cdot (S\cup\theta - \mathbf{1}\cdot\mu)\cdot EV$ - $EV^T\cdot(S-\mathbf{1}\cdot\mu)^T\cdot(S-\mathbf{1}\cdot\mu)\cdot EV\}$
$= diag\{EV^T\cdot((S\cup\theta-\mathbf{1}\cdot\mu)^T)(S\cup\theta-\mathbf{1}\cdot\mu) - (S-\mathbf{1}\cdot\mu)^T\cdot(S-\mathbf{1}\cdot\mu))\cdot EV\}$

Note that $(S\cup\theta-\mathbf{1}\cdot\mu) = [S-\mathbf{1}\cdot\mu, \theta-\mathbf{1}\cdot\mu]^T$, we can get $(S\cup\theta-\mathbf{1}\cdot\mu)^T\cdot(S\cup\theta-\mathbf{1}\cdot\mu) = (S-\mathbf{1}\cdot\mu)^T\cdot(S-\mathbf{1}\cdot\mu) + (\theta-\mu)^T\cdot(\theta-\mu)$. Use this equation, we can get PTB(θ, S, μ, EV)
= $diag\{EV^T\cdot(\theta-\mu)^T\cdot(\theta-\mu)\cdot EV\}$
= SPA(θ, μ, EV).

■

As can be seen from *Corollary* 1, S no longer participates in the calculation of PTB(θ, S, μ, EV). Therefore, PTB(θ, S, μ, EV) can be rewritten as PTB(θ, μ, EV).

Corollary 2. *A new data instance $\gamma=\{\gamma_1,...,\gamma_m\}$ has the same attributes as S. Let $\mu=\{\mu_1,...,\mu_m\}$ be the center of S and EV is defined in the same way as Definition 1. If the following condition is satisfied:*

$$max\|(\gamma-\mu)\cdot EV_j\| \leq \frac{1}{m^\kappa}, j \in [1,p], \kappa > 1 \quad (10)$$

then even the attribute number m increases to an infinite dimension, $\|PTB(\gamma, \mu, EV)\|_{m->\infty} \leq constant$

Proof:

According to *Corollary* 1, $\|PTB(\gamma, S, \mu, EV)\|_{m->\infty}$
$= \|diag\{((\gamma-\mu)\cdot EV)^T(\gamma-\mu)\cdot EV\}\|_{m->\infty}$
$= \|diag\{((\gamma-\mu)\cdot[EV_1,..,EV_p])^T(\gamma-\mu)\cdot[EV_1,..,EV_p]\}\|$ (11)

By applying condition (10) to (11):

$$\|PTB(\gamma, S, \mu, EV)\|_{m->\infty} \leq \sum\tfrac{1}{m^{\kappa\cdot p}}|_{m->\infty,\kappa>1} = \zeta(\kappa\cdot p).$$

The last function is a Riemann zeta function evaluated at $\kappa\cdot p$. It converges to a constant since $\kappa\cdot p > 1$.

■

Actually, *Corollary* 2 gives an upper bound of $\|PTB(\gamma, \mu, EV)\|$ when m approaches to infinity. In real applications, the condition in (10) may be too restricted to be satisfied and the following *Corollary* 3 is more suitable for practical use.

Corollary 3. *Suppose there are ω attributes of γ that do not satisfy the condition in (10). As long as the following condition holds:*

$$max\|(\gamma_i-\mu_i)\cdot EV_j\| \leq T \quad (12)$$

$$\omega < p-1 \quad (13)$$

$\|PTB(\gamma,\ \mu,\ EV)\|_{m->\infty}$ *still has an upper bound.*

Proof:

By applying both conditions (10) and (12) to (11), we can get

$$\begin{aligned}&\|PTB(\gamma,\ S,\ \mu,\ EV)\|_{m->\infty}\\&\leq \omega\cdot T\ +\leq\sum\frac{1}{m^{\kappa\cdot(p-\omega)}}|_{m->\infty,\kappa>1}\\&=\omega\cdot T\ +\zeta(\kappa\cdot(p-\omega))|_{m->\infty,\kappa>1}\end{aligned}$$

It converges to a constant since $\kappa\cdot(p-\omega)>1$.

■

CODE 1: LEARNING (PARALLEL MODE)

Input:
(1) A set of training classes $\{S^{(1)},...,S^{(N)}\}$ with labels.
(2) A set of validation instances VS with class labels.
Output:
(1) Center of each class $\{\mu^{(1)},...,\mu^{(N)}\}$,
(2) Positive eigenvalues of each class $\{\lambda^{(1)},...,\lambda^{(N)}\}$,
(3) Corresponding eigenvectors $\{EV^{(1)},...,EV^{(N)}\}$,
(4) Number of data instances of each class $\{|S^{(1)}|,...,|S^{(N)}|\}$,
(5) β_{opt}.

Calculate the center $\mu^{(i)}$ of class $S^{(i)}$, $i\in[1,N]$.
Calculate the eigenvalues $\lambda^{(i)}$ and corresponding eigenvectors $EV^{(i)}$ of class $S^{(i)}$, $i\in[1,N]$.
Calculate the number of data instances $|S^{(i)}|$ of class $S^{(i)}$, $i\in[1,N]$.
for each data instance $Ts\in VS$
  calculate $SPA(Ts,\mu^{(i)},EV^{(i)})$, $i\in[1,N]$ in parallel
  calculate $\Gamma(Ts,\mu^{(i)},EV^{(i)},\lambda^{(i)})$, $i\in[1,N]$ in parallel
end
for $\beta\leftarrow$ *init_value* to *end_value* with step s
  for each data instance $Ts\in VS$
    calculate Weighted Closeness Score $WSC^{(i)}=|S^{(i)}|^{\beta}\cdot\Gamma(Ts,\mu^{(i)},EV^{(i)},\lambda^{(i)})$
    predict class label of Ts as c using $class\_label(Ts)=c=\underset{\phi}{\operatorname{argmin}}\{WSC^{(\phi)}\}$, $\phi\in[1,N]$
  end
    calculate accuracy ACC_{β} of classification according to authentic class label.
end
Find β_{opt} which corresponds to the maximum value of ACC_{β} :
$\beta_{opt}=\underset{\beta}{\operatorname{argmax}}\{ACC_{\beta}\}$, $\beta\in[init\_value,end\_value]$

B. SPRUCE Framework

Two kinds of frameworks that can be deployed in different situations are proposed. The parallel framework of the proposed subspace classifier is shown in Figure 1 and the sequential framework is shown in Figure 2. The parallel framework can be adopted in a parallel computing environment in which the computation power is sufficient to calculate the Weighted Closeness Score (WCS) (such as $WCS^{(1)}$ and $WCS^{(2)}$) for an input instance with regard to different classes simultaneously. The sequential framework, on the other hand, considers the situation when the computational power is limited and only a proportion of WCS can be calculated at the same time. In Figure 2, we display the extreme case that only one WSC can be calculated at one time.

The pseudo codes of learning and classification of SPRUCE with regard to the parallel mode are shown in Code 1 and Code 2. Within the code, class i is identified by the superscript (i). The spectrum perturbation of an input data instance Ts can be calculated using Equation (9) according to Corollary 1.

$\Gamma(Ts',\mu^{(i)},EV^{(i)},\lambda^{(i)})$ (as shown in Equation (14)) is used as the Closeness Score (CS) in our proposed framework to measure the closeness that a data instance is to a PC subspace.

$$\Gamma(\delta,\mu,EV,\tau)=\sum_i\tau_i\cdot((\delta-\mu)\cdot EV_i)^2. \tag{14}$$

CODE 2: CLASSIFICATION (PARALLEL MODE)

Input:
(1) A set of testing data instances TS,
(2) center of each class $\{\mu^{(1)},...,\mu^{(N)}\}$,
(3) positive eigenvalues of each class $\{\lambda^{(1)},...,\lambda^{(N)}\}$,
(4) Corresponding eigenvectors $\{EV^{(1)},...,EV^{(N)}\}$,
(5) Number of data instances of each class $\{|S^{(1)}|,...,|S^{(N)}|\}$,
(6) $\beta=\beta_{opt}$ from learning process
Output: Class labels of TS

---

for each data instance $Ts'\in TS$
  calculate $SPA(Ts',\mu^{(i)},EV^{(i)})$, $i\in[1,N]$ in parallel
  calculate $\Gamma(Ts',\mu^{(i)},EV^{(i)},\lambda^{(i)})$, $i\in[1,N]$ in parallel
  calculate Weighted Closeness Score $\overline{WSC}^{(i)}=|S^{(i)}|^{\beta}\cdot\Gamma(Ts',\mu^{(i)},EV^{(i)},\lambda^{(i)})$
  predict class label of Ts' as c' using $class\_label(Ts')=c'=\underset{\phi}{\operatorname{argmin}}\{\overline{WSC}^{(\phi)}\}$, $\phi\in[1,N]$
end

We believe this closeness score is intuitively reasonable to estimate the closeness that a given input data instance to the subspace of a class. According to Equation (14), the closer a testing instance δ is towards the class center in terms of μ, the smaller closeness score δ holds. However, there is one issue related to the comparison between the closeness of a given input data instance towards different subspaces. Empirical study shows that a large-size class usually has more large eigenvalues than a small-size class. Intuitively, the reason to explain for this phenomenon is that a large-size class usually has larger variances. Note that Principal Component Analysis (PCA) does not add but may deduct a small amount of variance from total variance after converting data instances from the original space to the PC subspace because some tiny eigenvalues could be discarded. Therefore,

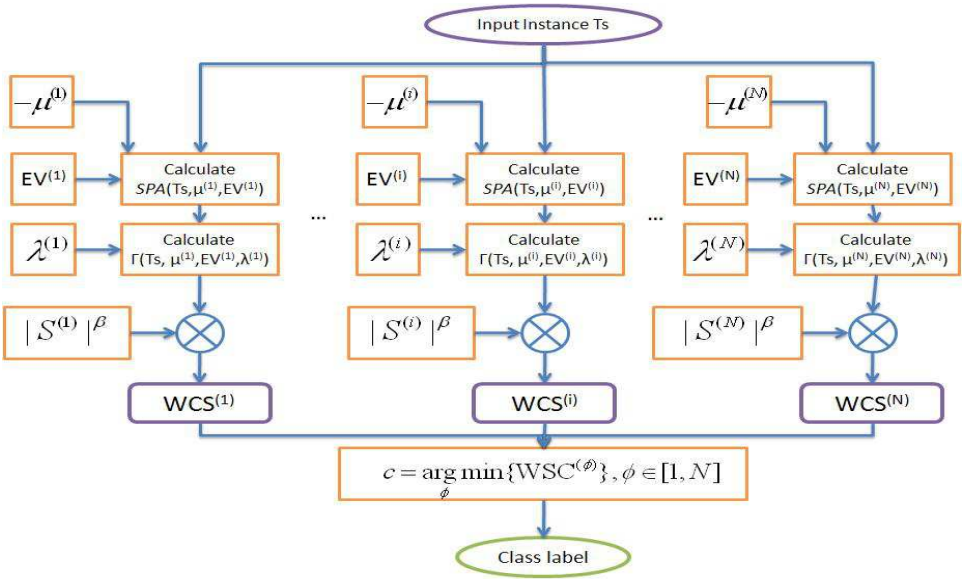

Fig. 1. The parallel framework of SPRUCE

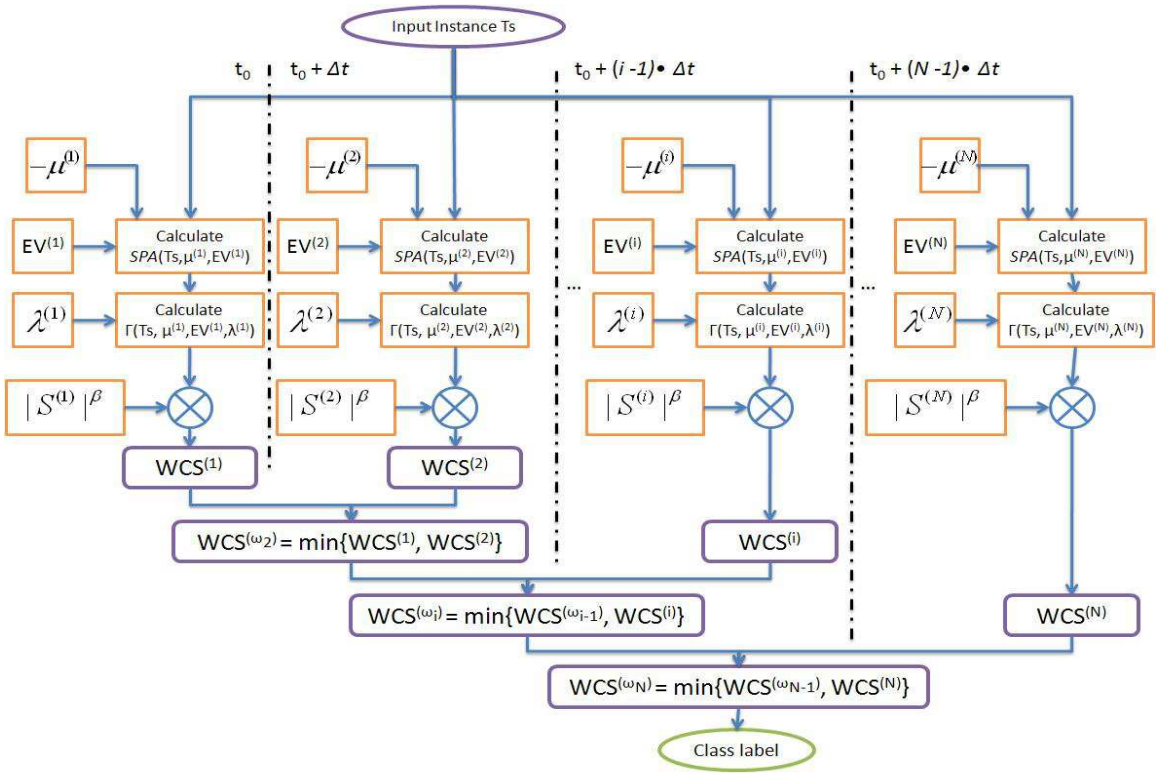

Fig. 2. The sequential framework of SPRUCE

after sorting eigenvalues in a descending order, the following equation usually holds:

$$P(\lambda_i^{(large-size)} > \lambda_i^{(small-size)}) > P(\lambda_i^{(small-size)} > \lambda_i^{(large-size)}), \quad (15)$$

where i is the index of the position in the descending sequence of λ and $P(.)$ denotes the probability.

CODE 3: LEARNING (SEQUENTIAL MODE)

Input:
(1) A set of training classes $\{S^{(1)},...,S^{(N)}\}$ with labels.
(2) A set of validation data instances VS with labels.
Output:
(1) Center of each class $\{\mu^{(1)},...,\mu^{(N)}\}$,
(2) Positive eigenvalues of each class $\{\lambda^{(1)},...,\lambda^{(N)}\}$
(3) Corresponding eigenvectors $\{EV^{(1)},...,EV^{(N)}\}$,
(4) Number of data instances of each class $\{|S^{(1)}|,...,|S^{(N)}|\}$,
(5) β_{opt}

Calculate the center $\mu^{(i)}$ of class $S^{(i)}$, $i \in [1,N]$.
Calculate the eigenvalues $\lambda^{(i)}$ and corresponding eigenvectors $EV^{(i)}$ of class $S^{(i)}$, $i \in [1,N]$.
Calculate the number of data instances $|S^{(i)}|$ of class $S^{(i)}$, $i \in [1,N]$.
for each data instance $Ts \in VS$
  for $i \leftarrow 1$ to N
    calculate $SPA(Ts, \mu^{(i)}, EV^{(i)})$, $i \in [1,N]$
    calculate $\Gamma(Ts, \mu^{(i)}, EV^{(i)}, \lambda^{(i)})$, $i \in [1,N]$
  end
end
for $\beta \leftarrow init\_value$ to end_value with step s
  for each instance $Ts \in VS$
    calculate Weighted Closeness Score $WSC^{(i)} = |S^{(i)}|^{\beta} \cdot \Gamma(Ts, \mu^{(i)}, EV^{(i)}, \lambda^{(i)})$
    predict class label of Ts as c using $class\_label(Ts) = c = \underset{\phi}{\operatorname{argmin}}\{WSC^{(\phi)}\}$, $\phi \in [1,N]$
  end
    calculate accuracy ACC_{β} of classification according to authentic class label.
end
Find β_{opt} which corresponds to the maximum value of ACC_{β} : $\beta_{opt} = \underset{\beta}{\operatorname{argmax}}\{ACC_{\beta}\}, \beta \in [init\_value, end\_value]$

Then an issue may arise from this phenomenon that the large-size class will dominate the small-size class since the eigenvalues of the large-size class is larger and therefore the CS of the large-size class is smaller. To make a balance between the large-size class and small-size class, a weight value $|S^{(i)}|^{\beta}$ is compensated to $SC^{(i)}$ to get a Weighted Closeness Score (WSC) for class i, $WSC^{(i)}$. $|S^{(i)}|$ is the number of data instances of $S^{(i)}$ and β is a scalar factor. In the learning process, this β is derived by the process to search the maximum accuracy. In a real situation, cross-validation could be utilized and β can adopt the mean value β_{opt} with regard to different folds.

For the sequential mode, the pseudo codes of the learning and classification steps are shown in Code 3 and Code 4. The difference from the parallel mode in the learning step only lies in the part of calculating $SPA(Ts, \mu^{(i)}, EV^{(i)})$ and $\Gamma(Ts, \mu^{(i)}, EV^{(i)}, \lambda^{(i)})$ because of the insufficiency of computation power. $SPA(Ts', \mu^{(i)}, EV^{(i)})$ and $\Gamma(Ts', \mu^{(i)}, EV^{(i)}, \lambda^{(i)})$ are only allowed to be computed one by one. The sequential mode in the classification step also requires computing $SPA(Ts', \mu^{(i)}, EV^{(i)})$ and $\Gamma(Ts', \mu^{(i)}, EV^{(i)}, \lambda^{(i)})$ one by one. The newly generated $WSC^{(i)}$ in each time slot Δt will be compared with $WSC^{(min)}$ to decide if the current class label requires updating.

CODE 4: CLASSIFICATION (SEQUENTIAL MODE)

Input:
(1) A set of testing data instances TS,
(2) center of each class $\{\mu^{(1)},...,\mu^{(N)}\}$,
(3) positive eigenvalues of each class $\{\lambda^{(1)},...,\lambda^{(N)}\}$
(4) Corresponding eigenvectors $\{EV^{(1)},...,EV^{(N)}\}$,
(5) Number of data instances of each class $\{|S^{(1)}|,...,|S^{(N)}|\}$,
(6) $\beta = \beta_{opt}$ from learning process
Output: Class labels of TS

---

for each data instance $Ts' \in TS$
  for $i \leftarrow 1$ to N
    calculate $SPA(Ts', \mu^{(i)}, EV^{(i)})$, $i \in [1,N]$
    calculate $\Gamma(Ts', \mu^{(i)}, EV^{(i)}, \lambda^{(i)})$, $i \in [1,N]$
    calculate Weighted Closeness Score $\overline{WSC}^{(i)} = |S^{(i)}|^{\beta} \cdot \Gamma(Ts', \mu^{(i)}, EV^{(i)}, \lambda^{(i)})$
    if i equals 1
      $class\_label(Ts') = 1$,
      $\overline{WSC}^{(min)} = \overline{WSC}^{(1)}$
    else if $\overline{WSC}^{(i)} < \overline{WSC}^{(min)}$
      $class\_label(Ts') = i$,
      $\overline{WSC}^{(min)} = \overline{WSC}^{(i)}$
    end
  end
  predict class label of Ts' as $c'(class\_label(Ts') = c')$ where $\overline{WSC}^{(c')} = min\{\overline{WSC}^{(i)}\}, i \in [1,N]$
end

IV. EXPERIMENTS AND RESULTS

To show the effectiveness of SPRUCE, comparative experiments with other linear PC subspace classification algorithms are conducted. The setup of the experiments is introduced in Section IV-A and the experimental results and analyses are shown in IV-B.

A. Experiment setup

The datasets used in the experiments are from three different sources: UCI machine learning dataset repository [26], Statlog datasets [27], and Trecvid 2008 semantic indexing training set [28]. Five are from UCI machine learning dataset repository: Wine, Waveform Database Generator (Version 1), Iris, Haberman's survival, and SPECTF Heart dataset. Two are from Statlog: Vehicle and Heart. Please note that some datasets in Statlog, like Vehicle, can also be found at UCI machine learning repository, and Heart from Statlog and SPECTF Heart from UCI machine learning repository are completely two different datasets. Three other datasets are built from TRECVID 2008 semantic indexing training set: Dog (Concept ID: 4), Kitchen (Concept ID: 5), and Harbor (Concept ID: 11). TRECVID 2008 semantic indexing training set only has raw data, such as videos and keyframes and the distribution of classes is rather imbalanced. The concept class (CO) versus non-concept class (NC) are too imbalanced with the average CO/NC ratio of $1:100$. Therefore, we extract features from the raw data and balance the classes by keeping concept class and subsampling non-concept class so that the CO/NC ratio is close to $1:3$. The following step summarizes the process to build the datasets from TRECVID2008:

PROCESS TO BUILD DATASETS FROM TRECVID2008

1 Extract audio visual features from videos.
2 Extract visual features from keyframes.
3 Merge features from previous two steps.
4 Apply feature selection to remove trivial features.
5 Subsample the non-concept class about 3 times the size of concept class.
6 Merge non-concept class with concept class to form a new dataset.

The performance of different classification algorithms is evaluated in terms of accuracy. Each data set is evaluated using 10 times 3-fold cross validation. CD-PCA applies the exhausted PC selection method in principal component subspace to reach the best classification accuracy. All the other algorithms in the experiments simply discard the trivial principal components which correspond to zero eigenvalues to reduce the dimension. CA-PCA and SPPCA are evaluated using the settings in their original papers. Furthermore, since SPPCA applies multi-class classification class by class and its performance is evaluated in a different way from the evaluation method adopted in this paper. We choose the maximum accuracy among these classes as the final accuracy shown in the experimental results so as to make it possible for comparison. The parameter β involved in SPRUCE adopts the mean value of optimal β value of different folds through cross-validation. Suppose the training data applies 3-fold cross-validation. Therefore, each fold produces one β_{opt}. The mean value of the three β_{opt} values will be adopted as the parameter β in classification.

The mean and standard deviation of the accuracy values from 10 times 3-fold cross validation further go through the significant test using Equation (16) to see how significantly SPRUCE outperforms those comparative linear PC subspace classification algorithms (CD-PCA, CA-PCA, and SPPCA) as well as some other types of classification algorithms (such as Decision Tree (DTree), Nearest Neighbor(NN), and K-Nearest Neighbor(K-NN)).

$$t = \frac{ACC_{SPRUCE} - ACC_{target}}{\sqrt{\frac{STD^2_{SPRUCE}}{DF_{SPRUCE}} + \frac{STD^2_{target}}{DF_{target}}}}. \tag{16}$$

Here, ACC_{SPRUCE} is the mean accuracy of SPRUCE, and ACC_{target} is the maximum value of mean accuracy of all comparative algorithms. STD_{SPRUCE} and STD_{target} are the standard deviation values of SPRUCE and the selected comparative algorithms, respectively. DF_{SPRUCE} is the degree of freedom of SPRUCE while DF_{target} is the degree of freedom of the selected comparative algorithms. In the experiments, 10 times 3-fold cross validation is adopted. Therefore, $DF_{SPRUCE} = DF_{target} = 9$. The algorithms of CD-PCA, CA-PCA and SP-PCA are implemented by ourselves according to the original papers; while the implementations of DTree, NN and K-NN available in Weka [29] are used.

B. Experimental results and analyses

The experimental results of classification accuracy against subspace classification methods and the corresponding result of significant test are shown in Table 1 and Table 3, respectively. As can be seen from Table 1, SPRUCE outperforms the other comparative classification algorithms on all data sets. Table 3 further reveals that SPRUCE can beat other comparative subspace classification algorithms significantly for 11 out of 12 datasets. It is easy to notice that CD-PCA renders the best classification accuracy among all three comparative classification algorithms for 7 datasets. However, this is due to the deployment of the exhausted PC selection method on CD-PCA and this deployment will definitely increase the accuracy though it is rather expensive to apply this deployment in practical use. For the datasets built from TRECVID2008 training set, the performance of SPRUCE is close to CA-PCA and SPPCA. This may be due to the fact that the concept class and non-concept class in the three TRECVID2008 datasets have a larger intra-class variance and/or smaller inter-class variance than those of the other datasets. Therefore, it increases the difficulty in building a robust spectrum perturbation subspace model for each class. Finally, it needs to be mentioned here that CA-PCA, SPPCA, and SPRUCE can benefit from the sophisticated PC selection process to increase the accuracy of the classification.

The effectiveness of SPRUCE against some other commonly used classification algorithms can be seen from Table 2 and Table 4, respectively. As can be seen from the two tables, SPRUCE can render better classification accuracy, though K-NN seems to be a strong competitor. Compare Table 1 with Table 2, it is noticeable that the 3 comparative subspace methods did not render stable performance against the other selected classification algorithms in Table 2. For example, SPPCA

TABLE I
ACCURACY COMPARISON AGAINST SUBSPACE METHODS ON DATASETS WITH STANDARD DEVIATION

Data source	Date set	CD-PCA	CA-PCA	SPPCA	SPRUCE
UCI	Wine	89.01 ± 1.10	96.18 ± 1.21	78.52 ± 5.05	**98.99 ± 0.69**
UCI	Waveform	81.23 ± 0.18	67.30 ± 0.84	78.92 ± 0.74	**84.82 ± 0.18**
UCI	Iris	95.39 ± 0.75	92.53 ± 1.83	79.76 ± 5.79	**97.39 ± 0.52**
UCI	Haberman's survival	71.60 ± 1.29	67.29 ± 2.74	73.99 ± 1.07	**74.12 ± 0.51**
UCI	SPECTF Heart	77.87 ± 2.54	73.33 ± 2.20	79.40 ± 0.00	**81.84 ± 2.19**
Statlog	Vehicle	77.64 ± 0.83	74.98 ± 0.83	67.02 ± 0.66	**83.50 ± 0.96**
Statlog	Heart	79.56 ± 0.74	74.89 ± 1.30	58.63 ± 4.73	**80.67 ± 0.92**
TRECVID 2008	dog	71.10 ± 0.71	70.68 ± 0.70	75.34 ± 0.44	**75.99 ± 0.12**
TRECVID 2008	kitchen	66.95 ± 0.80	70.21 ± 1.00	73.78 ± 0.29	**74.91 ± 0.04**
TRECVID 2008	harbor	67.30 ± 1.12	70.62 ± 0.52	73.66 ± 0.42	**75.00 ± 0.04**

TABLE II
ACCURACY COMPARISON AGAINST OTHER SELECTED CLASSIFICATION METHODS ON DATASETS WITH STANDARD DEVIATION

Data source	Date set	DTree	NN	K-NN	SPRUCE
UCI	Wine	90.63 ± 1.89	95.45 ± 0.72	96.19 ± 0.83	**98.99 ± 0.69**
UCI	Waveform	76.20 ± 0.40	77.46 ± 0.35	82.97 ± 0.30	**84.82 ± 0.18**
UCI	Iris	93.90 ± 1.41	95.54 ± 0.85	95.58 ± 0.89	**97.39 ± 0.52**
UCI	Haberman's survival	71.70 ± 1.06	67.19 ± 2.66	71.93 ± 1.24	**74.12 ± 0.51**
UCI	SPECTF Heart	80.19 ± 1.44	76.74 ± 1.72	80.60 ± 1.10	**81.84 ± 2.19**
Statlog	Vehicle	70.99 ± 0.78	68.89 ± 0.77	70.01 ± 0.90	**83.50 ± 0.96**
Statlog	Heart	77.74 ± 2.38	76.11 ± 2.09	79.19 ± 1.13	**80.67 ± 0.92**
TRECVID 2008	dog	71.25 ± 1.93	67.35 ± 0.68	74.19 ± 0.61	**75.99 ± 0.12**
TRECVID 2008	kitchen	72.05 ± 1.25	70.66 ± 0.67	74.49 ± 0.73	**74.91 ± 0.04**
TRECVID 2008	harbor	72.86 ± 1.68	69.45 ± 0.72	74.37 ± 0.50	**75.00 ± 0.04**

TABLE III
SIGNIFICANCE TEST AGAINST SUBSPACE METHODS ON ALL DATASETS

Data source	Date set	Comparative Algorithm	Comparative Accuracy	SPRUCE	p-value
UCI	Wine	CA-PCA	96.18 ± 1.21	98.99 ± 0.69	0.00
UCI	Waveform	CD-PCA	81.23 ± 0.18	84.82 ± 0.18	0.00
UCI	Iris	CD-PCA	95.39 ± 0.75	97.39 ± 0.52	0.00
UCI	Haberman's survival	SPPCA	73.99 ± 1.07	74.12 ± 0.51	0.37
UCI	SPECTF Heart	SPPCA	79.40 ± 0.00	81.84 ± 2.19	0.00
Statlog	Vehicle	CD-PCA	77.64 ± 0.83	83.50 ± 0.96	0.00
Statlog	Heart	CD-PCA	79.56 ± 0.74	80.67 ± 0.92	0.01
TRECVID 2008	dog	SPPCA	75.34 ± 0.44	75.99 ± 0.12	0.00
TRECVID 2008	kitchen	SPPCA	73.78 ± 0.29	74.91 ± 0.04	0.00
TRECVID 2008	harbor	SPPCA	73.66 ± 0.42	75.00 ± 0.04	0.00

TABLE IV
SIGNIFICANCE TEST AGAINST OTHER SELECTED CLASSIFICATION METHODS ON ALL DATASETS

Data source	Date set	Comparative Algorithm	Comparative Accuracy	SPRUCE	p-value
UCI	Wine	K-NN	96.19 ± 0.83	98.99 ± 0.69	0.00
UCI	Waveform	K-NN	82.97 ± 0.30	84.82 ± 0.18	0.00
UCI	Iris	K-NN	95.58 ± 0.89	97.39 ± 0.52	0.00
UCI	Haberman's survival	K-NN	71.93 ± 1.24	74.12 ± 0.51	0.00
UCI	SPECTF Heart	K-NN	80.60 ± 1.10	81.84 ± 2.19	0.08
Statlog	Vehicle	DTree	70.99 ± 0.78	83.50 ± 0.96	0.00
Statlog	Heart	K-NN	79.19 ± 1.13	80.67 ± 0.92	0.01
TRECVID 2008	dog	K-NN	74.19 ± 0.44	75.99 ± 0.12	0.00
TRECVID 2008	kitchen	K-NN	74.49 ± 0.29	74.91 ± 0.04	0.00
TRECVID 2008	harbor	K-NN	74.37 ± 0.42	75.00 ± 0.04	0.00

could beat K-NN on datasets dog and Haberman's survival but it is worse than K-NN for the rest of the datasets. However, the proposed SPRUCE is able to provide a more stable behavior that always outperforms K-NN for the selected datasets. There are two reasons to explain this behavior. First, it utilizes a supervised learning process to build the subspace and therefore the subspace reflects the characteristics of each class, which makes it easy to differentiate instances belonging to different classifiers. Second, the proposed supervised subspace method does not rely on any assumption, such as normal distribution of data and label patters. It does not rely on the other classifier to do the classification jobs. Therefore, it can be adapted to different situations.

V. Conclusion and Future work

In this paper, a new spectrum perturbation-based subspace classification algorithm called SPRUCE is introduced. SPRUCE takes into consideration the spectrum perturbation of each class when building subspace learning model and utilizes the perturbation on the spectra of classes to measure the closeness of an input instance towards those classes. Experimental results reveal that it outperforms comparative linear PC subspace classification algorithms and some other commonly used classifiers in terms of classification accuracy. The promising results demonstrate that SPRUCE can effectively and efficiently enable data management support for semantics-level information search, retrieval, and sharing in a collaborative environment.

Nevertheless, there are still a few issues that need to be discussed in the future, for example, how to

- handle the curse of dimensionality problem for large dimensional datasets; and
- extend SPRUCE to nonlinear domains.

For the first issue, we will explore the development of a fast algorithm to derive eigenvectors of the covariance matrix, which enables the deployment of SPRUCE in applications involving large datasets. For the second issue, we plan to investigate the possibility of applying kernel trick on SPRUCE to extend it to nonlinear domain, which may make it capable of handling nonlinear cases.

Acknowledgment

For Shu-Ching Chen, this work is supported in part by the U.S. Department of Homeland Security under grant Award Number 2010-ST-062-000039, the U.S. Department of Homeland Security's VACCINE Center under Award Number 2009-ST-061-CI0001, and NSF HRD-0833093.

References

[1] A. Ceglar and J. F. Roddick, "Association mining," *ACM Computing Surveys (CSUR)*, vol. 38, no. 2, pp. 5–es, July 2006.

[2] S.-C. Chen, M.-L. Shyu, C. Zhang, and M. Chen, "A multimodal data mining framework for soccer goal detection based on decision tree logic," *International Journal of Computer Applications in Technology, Special Issue on Data Mining Applications*, vol. 27, no. 4, pp. 312–323, 2006.

[3] S.-C. Chen, M.-L. Shyu, M. Chen, and C. Zhang, "A decision tree-based multimodal data mimimg framework for soccer goal detection," in *IEEE International Conference on Multimedia and Expo (ICME04)*, June 2004, pp. 265–268.

[4] K. Hammouda and M. Kamel, "Collaborative document clustering," in *SIAM Conference on Data Mining*, April 2006, pp. 453–463.

[5] L. Lin, C. Chen, M.-L. Shyu, and S.-C. Chen, "Weighted subspace filtering and ranking algorithms for video concept retrieval," *IEEE Multimedia*, vol. 18, no. 3, pp. 32–43, 2011.

[6] K.-H. Liu, M.-F. Weng, C.-Y. Tseng, Y.-Y. Chuang, and M.-S. Chen, "Association and temporal rule mining for post-filtering of semantic concept detection in video," *IEEE Transactions on Multimedia*, vol. 10, no. 2, pp. 240–251, February 2008.

[7] M.-L. Shyu, C. Chen, and S.-C. Chen, "Multi-class classification via subspace modeling," *International Journal of Semantic Computing*, vol. 5, no. 1, pp. 55–78, 2011.

[8] F. Thabtah, "Challenges and interesting research directions in associative classification," in *IEEE International Conference on Data Mining Workshops (ICDMW06)*, December 2006, pp. 785–792.

[9] D. Zhou and C. J. C. Burges, "Spectral clustering and transductive learning with multiple views," in *ACM International Conference on Machine Learning (ICML07)*, January 2007, pp. 1159–1166.

[10] X. Su and T. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 2009, no. 4, pp. 4:2–4:2 (19 pages), January 2009.

[11] J.-S. Lee, C. Jun, J. Lee, and S. Kim, "Classification-based collaborative filtering using market basket data," *Expert Systems with Applications*, vol. 29, no. 3, pp. 700–704, 2005.

[12] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," in *Proceeding of the 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, May 2007, pp. 3–24.

[13] H. Hotelling, "Analysis of a complex of statistical variable into principal components," *Journal of Educational Psychology*, vol. 24, no. 7, pp. 498–520, October 1933.

[14] J. Laaksonen, "Subspace classifiers in recognition of handwritten digits," Department of Computer Science and Engineering, May 1997.

[15] R. Cappelli, D. Maio, and D. Maltoni, "Subspace classification for face recognition," in *Proceedings of the International ECCV 2002 Workshop on Biometric Authentication*, June 2002, pp. 133–142.

[16] H. Murata and T. Onoda, "Applying kernel based subspace classification to a non-intrusive monitoring for household electric appliances," in *Proceedings of the International Conference on Artificial Neural Networks*, ser. ICANN '01. London, UK: Springer-Verlag, August 2001, pp. 692–698.

[17] D. Zhora, "Financial forecasting using random subspace classifier," in *Proceedings of 2004 IEEE International Joint Conference on Neural Networks*, July 2004, pp. 2735–2740.

[18] I. T. Jolliffe, *Principal Component Analysis*. Springer-Verlag, 2002.

[19] A. Sharma, K. Paliwal, and G. Onwubolu, "Class-dependent PCA, MDC and LDA: A combined classifier for pattern classification," *Pattern Recognition*, vol. 39, no. 7, pp. 1215–1229, 2006.

[20] H. Bischof, A. Leonardis, and F. Pezzei, "A robust subspace classifier," in *Proceedings of Fourteenth International Conference on Pattern Recognition*, August 1998, pp. 114–116.

[21] R. Santiago-Mozos, J. M. Leiva-Murillo, F. Perez-Cruz, and A. Artes-Rodriguez, "Supervised-PCA and SVM classifiers for object detection in infrared images," in *Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance*, July 2003, pp. 122–127.

[22] W. Zhao, R. Chellappa, and A. Krishnaswamy, "Discriminant analysis of principal components for face recognition," in *Proceedings of the 3rd. International Conference on Face and Gesture*, April 1998, pp. 336–341.

[23] M. Park and J. Choi, "Theoretical analysis on feature extraction capability of class-augmented PCA," *Pattern Recognition*, vol. 42, no. 11, pp. 2353–2362, 2009.

[24] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society, Series B*, vol. 61, pp. 611–622, 1999.

[25] S. Yu, K. Yu, V. Tresp, H. Kriegel, and M. Wu, "Supervised probabilistic principal component analysis," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, August 2006, pp. 464–473.

[26] A. Asuncion and D. J. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[27] R. D. King, C. Feng, and A. Sutherland, "Statlog: Comparison of classification algorithms on large real-world problems," *Applied Artificial Intelligence*, vol. 9, no. 3, pp. 289–333, 1995.

[28] A. F. Smeaton, P. Over, and W. Kraaij, "Evaluation campaigns and TRECVid," in *ACM International Workshop on Multimedia Information Retrieval (MIR06)*, October 2006, pp. 321–330.

[29] Weka, *WEKA*. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/