Knowledge-Based Systems xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Feature selection for noisy variation patterns using kernel principal component analysis

Anshuman Sahu [a,*], Daniel W. Apley [b], George C. Runger [a]

[a] School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85287, USA
[b] Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60208, USA

ARTICLE INFO

Article history:
Received 12 February 2014
Received in revised form 5 August 2014
Accepted 29 August 2014
Available online xxxx

Keywords:
Nonlinear PCA
Kernel feature space
Preimages
Variation patterns
Feature ensembles

ABSTRACT

Kernel Principal Component Analysis (KPCA) is a technique widely used to understand and visualize non-linear variation patterns by inverse mapping the projected data from a high-dimensional feature space back to the original input space. Variation patterns often occur in a small number of relevant features out of the overall set of features that are recorded in the data. It is, therefore, crucial to discern this set of relevant features that define the pattern. Here we propose a feature selection procedure that augments KPCA to obtain importance estimates of the features given the noisy training data. Our feature selection strategy involves projecting the data points onto sparse random vectors for calculating the kernel matrix. We then match pairs of such projections, and determine the preimages of the data with and without a feature, thereby trying to identify the importance of that feature. Thus, preimages' differences within pairs are used to identify the relevant features. An advantage of our method is it can be used with any suitable KPCA algorithm. Moreover, the computations can be parallelized easily leading to significant speedup. We demonstrate our method on several simulated and real data sets, and compare the results to alternative approaches in the literature.

1. Introduction

Advances in signal acquisition and computational processing coupled with cheap storage have resulted in massive multivariate data being collected in today's processes like semiconductor manufacturing, automobile-body assemblies, inspection systems, etc. The data can be in form of spatial profiles, time series or images where the measurements are recorded over several features. These features are affected by different sources of variation which result in variation patterns in the data. The goal, therefore, is to identify these sources of variation based on the process data collected. Moreover, the variation pattern may be present in only a small subset of the process variables that are collected. Finding this relevant subset of features is, therefore, critical to understand the process, and is the focus of our work presented in this paper.

Principal Component Analysis (PCA) is a common technique to identify variation pattern in data by projecting along the directions of maximum variability in the data. However, PCA can only identify linear relationships among features in the data. Kernel Principal Component Analysis (KPCA) extends PCA to the case where data contain non-linear patterns as shown by Schölkopf et al. [1]. KPCA identifies non-linear patterns in data by mapping the data from input space to a high-dimensional (possibly infinite) feature space, and performing PCA in the feature space. This is achieved by employing the kernel trick [2]. Thus, only calculations in terms of dot products in the input space are required, without an explicit mapping to the feature space. KPCA is widely used for nonlinear process monitoring [3–5], fault detection and diagnosis [6–9], and anomaly detection [10,11].

To visualize the variation pattern in input space, an inverse transform is used to map the denoised data from feature space back to the input space. The exact preimage of a denoised point in feature space might not exist, so that a number of algorithms for estimating approximate preimages have been proposed [12–15]. Also, [16,17] considered meta-methods to improve the preimage results by averaging from ensembles.

Our task now is to identify the relevant subset of the original set of features over which the pattern exists (a feature selection task). The difficulty is to handle the non-linear relationships between features in input space. Because the feature space in KPCA already provides an avenue to consider higher-order interactions between features, it is more appealing to apply a feature selection procedure

* Corresponding author.
E-mail addresses: anshuman.sahu@asu.edu (A. Sahu), apley@northwestern.edu (D.W. Apley), george.runger@asu.edu (G.C. Runger).

http://dx.doi.org/10.1016/j.knosys.2014.08.027
0950-7051/

in feature space itself. However, it is not always possible to obtain the feature representation in feature space (for example, in the case of a Gaussian kernel) because the data are not explicitly mapped. Therefore, the challenge here is to perform feature selection in the feature space.

Some work has considered feature selection in feature space for supervised learning. A weighted feature approach was provided by Allen [18] where weights are assigned to features while computing the kernel. This feature weighting is incorporated into the loss function corresponding to classification or regression problem and a lasso penalty is put on the weights. The features corresponding to non-zero weights obtained after minimizing the objective (loss function with penalty) are considered the important ones. Similarly, recent work [19,20] also employed feature weighting for the cases of Support Vector Machine (SVM) classification and regression, respectively. For both the cases, an anisotropic Gaussian kernel was used to supply weights to features. Specifically, Maldonado et al. [19] provided an iterative algorithm for solving the feature selection problem by embedding the feature weighting in the dual formulation of SVM problem. The algorithm begins with an initial set of weights. At each iteration, it solves the SVM problem for the given set of feature weights, updates the weights using the gradient of the objective function, and removes the features that are below a certain given threshold. This procedure is repeated till convergence. Finally, the features obtained with non-zero weights are considered important.

Since KPCA is unsupervised, we next consider feature selection in feature space for unsupervised learning. One common aspect of all these algorithms, similar to their counterparts in supervised setting, is they involve some kind of feature weighting mechanism, and the relevant features are obtained by regularizing (shrinking) the weights of irrelevant features using some criteria. A method for feature selection in Local Learning-Based Clustering [21] was proposed by Zeng and ming Cheung [22]. The feature selection is achieved by regularizing the weights assigned to features. A method to measure variable importance in KPCA was suggested by Muniz et al. [23]. They computed the kernel between two data points as weighted sum of individual kernels where each individual kernel is computed on a single feature of each of the two data points, and the weights assigned to each kernel serve as a measure of importance of the feature involved in computing the kernel. They formulated a loss function where a lasso penalty was imposed on the weights to determine the non-zero weights (and the corresponding relevant features). In addition to feature selection in feature space for unsupervised learning, there exist several other feature selection procedures for unsupervised learning that operate in the input space. Laplacian Score (LS) was proposed by He et al. [24] for each feature to estimate its ability to preserve local structure. The authors construct a nearest neighbor graph, and identify the important features as those which maintain this graph structure. Multi-Cluster Feature Selection (MCFS) proposed by Cai et al. [25] used spectral analysis to select the features that preserve the multi-cluster structure of the data. The authors compute the nearest neighbors graph, define weights on edges in the graph, construct the graph Laplacian, and solve the generalized eigen-problem [26] to obtain the top K eigenvectors. For each eigenvector, the contribution of each feature is found by solving a L1-regularized regression. Each feature now has K contribution values, and the maximum of it is assigned as the MCFS score of the feature. The features with higher MCFS scores are important. Unsupervised Discriminative Feature Selection (UDFS) proposed by Yang et al. [27] aims to select the most discriminative features which preserve the local structure of the data (via manifold) while simultaneously accounting for feature correlation. The authors assume the existence of a linear classifier that classifies each data point to a class. They propose learning the classifier that maximizes their local discriminative score. To this end, they propose a regularized optimization problem by inducing $\ell_{2,1}$ norm on the coefficients of the classifier. Note that each coefficient of the linear classifier corresponds to a feature in the dataset. They also propose an iterative algorithm to solve this optimization problem. The top features are determined based on sorting the ℓ_2 norm of the coefficient vectors over all iterations in descending order.

The approaches provided in the literature focus on the case when noise-free training data are available. However, this is not the case in areas like manufacturing variation analysis. In practice, the data are corrupted with noise and has a lot of irrelevant features. Thus, we work with a noisy data set from which we need to find the relevant subset of the features over which the patterns in the data exist. To this end, we propose our novel approach.

As pointed out previously, an innovative way to do feature selection in high-dimensional feature space is to assign weights to features in input space. By using such an approach, we can compute the kernel using all the features instead of iteratively computing it using a subset of features at a time. The goal next is to identify the weights (by some regularization criterion) so that the non-zero weights correspond to the relevant features. We propose an alternative approach for this feature weighting mechanism. Instead of trying to determine the feature weights through a regularization approach, we multiply the features by sparse random vectors whose entries are independent and identically distributed drawn from a distribution (such as Gaussian). After projecting data points onto random subsets of features, we measure feature importance from differences in preimages, where preimages are computed with and without a feature. Therefore, more important features are expected to result in greater differences. The process is repeated iteratively with different sparse random vectors and the differences are averaged to estimate the final feature importance. Our approach above provides robustness to irrelevant features in the data by being able to project only on a small random subset of features at a time, and calculating the final mapped data matrix in input space from an ensemble of feature subsets. Another advantage of our approach is it works with any KPCA preimage algorithm.

We organize the remaining part of our paper as follows. Section 2 provides a brief description of different methods used to visualize the variation patterns in KPCA. For our feature selection method, we can consider any one of them as the base algorithm. Section 3 presents a mathematical description of our methodology. Section 4 shows the results of implementing our algorithm on several simulated and real datasets. Finally Section 5 provides conclusions.

2. Background on preimages in KPCA

KPCA is equivalent to PCA in feature space [1]. Let $\mathbf{X}$ denote the data set with N instances and F features where the instances are denoted by $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$. Similar to PCA, we want to find the eigenvalues and eigenvectors of the covariance matrix $\mathbf{C}$ in feature space. If the corresponding set of points mapped in the feature space $\varphi(\mathbf{x}_i),\ i = 1, 2, \ldots, N$ are assumed to be centered, $\mathbf{C}$ can be calculated by

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}\varphi(\mathbf{x}_i)\varphi(\mathbf{x}_i)' \tag{1}$$

The eigenvalues λ and eigenvectors $\mathbf{v}$ of matrix $\mathbf{C}$ are given by

$$\mathbf{C}\mathbf{v} = \lambda\mathbf{v} \tag{2}$$

It can be shown that an eigenvector corresponding to non-zero eigenvalue of $\mathbf{C}$ can be written as a linear combination of $\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_N)$. Using this simplification reduces the original

Please cite this article in press as: A. Sahu et al., Feature selection for noisy variation patterns using kernel principal component analysis, Knowl. Based Syst. (2014), http://dx.doi.org/10.1016/j.knosys.2014.08.027

problem of finding eigenvalues and eigenvectors of **C** to finding the corresponding eigenvalues and eigenvectors of the kernel matrix **K** with entries

$$\mathbf{K}_{ij} := (\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)) \tag{3}$$

The product $\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$ is evaluated using the kernel trick Aizerman et al. [2] without explicitly computing the mapping $\varphi(\cdot)$.

Training data are used to obtain a reliable estimate of the principal component subspace in feature space onto which the test data can be projected. The procedure for visualizing variation pattern in test data can, thus, be summarized in four steps. The first step is to map the training data from input space to feature space via the kernel trick [2]. The second step is to calculate the principal component directions of the training data in feature space as shown by Schölkopf et al. [1]. The third step is to map the test data x to feature space and then project onto the space spanned by a small subset of the principal component directions found above. This projected test data (denoted by $P_{\varphi(x)}$) is also called the denoised data in feature space. In order to observe the pattern in input space, the denoised data are mapped back from feature space to input space in the fourth step. This last step is also referred to as obtaining the preimage $\hat{x}$ in KPCA literature. The above steps can be seen in Fig. 1.

The preimage can be used to visualize the variation pattern of the data in input space. As mentioned, in general, such an inverse mapping from feature space to input space may not exist, and the preimage cannot always be determined exactly [12]. Hence, several algorithms have been proposed to estimate the preimage. Mika et al. [12] proposed a gradient descent approach to numerically estimate the preimage matrix which, when mapped to the feature space, is closest (in terms of Euclidean distance) to the denoised matrix in feature space. Since the objective function (Euclidean distance) to minimize is non-convex, this approach is sensitive to initial starting solution. Kwok and Tsang [14] used the relationship between distance in input space and the feature space, and estimated the preimage of a test point as a linear combination of the training data points whose projections in feature space are closest to the denoised data point in feature space. Kwok and Tsang [14] chose only a few nearest training data points in order to reduce the computational burden. Bakir et al. [13] applied kernel regression to the preimage problem where the inverse mapping from feature space to input space is posed as a regression problem. Both approaches by Kwok and Tsang [14] and Bakir et al. [13] favor noise-free training data.

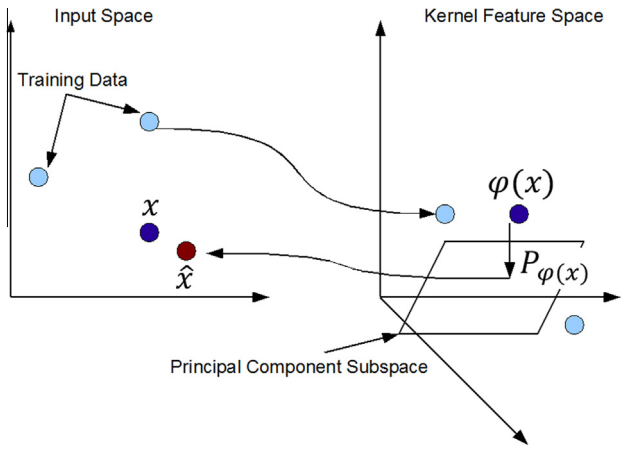


Fig. 1. KPCA and the preimage problem. Training data are transformed to feature space and used to learn a principal component plane. A test point x is transformed and projected to the plane as $P\varphi(x)$. The inverse transform of $P\varphi(x)$ may not exist, and an approximate preimage $\hat{x}$ is computed.

More recently, Sahu et al. [17] and Shinde et al. [16] considered meta-methods to estimate the final preimage. Since the training data is noisy, we expect the principle component subspace estimation to be unreliable. Thus, instead of estimating a single preimage from the training data, we resample the training data multiple times and estimate the final preimage by averaging the preimages obtained from each sample.

3. Feature selection using sparse random vectors with matched pairs

We now present a unique approach to learn the preimage as well as understand the contribution of a feature towards the variation pattern in the data. We utilize the idea of random projections where we project onto a small subset of features in each iteration. Essentially we try to capture the effect of that subset of features in feature space onto to which we randomly project. By repeating this procedure over a number of iterations, we create a diversified ensemble of feature subsets which account for the possible interactions between features that give rise to the variation pattern in the data. We explain this concept by an example. Let the dataset have features $f_1, f_2, f_3, f_4, f_5, f_6$. Suppose we choose the number of iterations to be three. In the first iteration, we randomly project onto $\{f_1, f_2\}$. In the second iteration, we randomly project onto $\{f_2, f_3, f_4\}$. In the third iteration, we randomly project onto $\{f_1, f_4\}$. The ensemble of feature subsets is given by $\{\{f_1, f_2\}, \{f_2, f_3, f_4\}, \{f_1, f_4\}\}$. We note that here we are creating an ensemble of feature subsets to estimate the preimage as well as identify the features that are relevant for the preimage, whereas previously we created an ensemble of data points by resampling the training data set [17,16] to estimate the primage. Matched pairs of projections are created for each feature to estimate the effect of the feature on the variation pattern. We then calculate the difference in the preimage as a result of excluding the feature. Thus, important features are expected to result in high differences. We refer to our procedure as MPFS, and describe it mathematically as follows.

Let $\mathbf{w}$ be a sparse random vector of dimension F where $\lfloor \gamma F \rfloor$ entries are non-zero. Here γ is a parameter that controls sparseness. The entries in the sparse random vector are independently sampled from a distribution (such as Gaussian). Let B be a fixed number of iterations. Let **K** be the kernel matrix obtained from instances in the input space. Let **X** denote the data matrix with N rows, N being the total number of data points. Let $\mathbf{x}_i$ and $\mathbf{x}_j$ denote two instances in input space. Assume that we are using a Gaussian kernel. The ijth entry in **K** is calculated as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|_F^2}{\sigma}\right) \tag{4}$$

For the purpose of MPFS, we modify **K** to $\mathbf{K}_w$ where we obtain the corresponding ijth entry in $\mathbf{K}_w$ as

$$k_w(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-(\mathbf{w}^T\mathbf{x}_i - \mathbf{w}^T\mathbf{x}_j)^2}{\sigma}\right) \tag{5}$$

We also normalized $\mathbf{w}$ to unit length in Eq. (5). Preliminary experiments, however, did not show meaningful differences in results obtained from normalized and nonnormalized $\mathbf{w}$.

For each $f = 1, 2, \ldots, F$ in each iteration b ($b = 1, 2, \ldots, B$), we generate a sparse random vector $\mathbf{w}_b$. To create matched pairs, we transform $\mathbf{w}_b$ to $\mathbf{w}_b^*$ by the following mechanism. Denote fth entry of $\mathbf{w}_b$ by $\mathbf{w}_b[f]$ and the corresponding entry in $\mathbf{w}_b^*$ as $\mathbf{w}_b^*[f]$. Then, for each b, we set

$$\mathbf{w}_b^*[f] = \begin{cases} 0 & \text{if } \mathbf{w}_b[f] \neq 0 \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

Please cite this article in press as: A. Sahu et al., Feature selection for noisy variation patterns using kernel principal component analysis, Knowl. Based Syst. (2014), http://dx.doi.org/10.1016/j.knosys.2014.08.027

Thus, for every feature f at each iteration b, we generate matched pairs $\mathbf{w}_b$ and $\mathbf{w}_b^*$ which differ only at the fth entry. We use $\mathbf{w}_b$ to obtain kernel matrix $\mathbf{K}_{wb}$ by substituting $\mathbf{w}$ with $\mathbf{w}_b$ in Eq. (5) and then use $\mathbf{K}_{wb}$ and $\mathbf{X}$ in a preimage estimation algorithm to obtain $\widehat{\mathbf{X}}_b$ at iteration b for each f. Similarly, we use $\mathbf{w}_b^*$ to obtain $\mathbf{K}_{wb}^*$ and then use $\mathbf{K}_{wb}^*$ along with $\mathbf{X}$ to obtain $\widehat{\mathbf{X}}_b^f$ at iteration b for each f.

The importance of feature f, denoted by imp_f, is calculated as

$$imp_f = \sum_{b=1}^{B} \frac{||\widehat{\mathbf{X}}_b - \widehat{\mathbf{X}}_b^f||_F}{B} \tag{7}$$

where the Frobenius norm of the matrix is used. The Frobenius norm of a matrix is defined as the square root of the sum of squares of elements of the matrix. Thus, if Frobenius norm of $(\widehat{\mathbf{X}}_b - \widehat{\mathbf{X}}_b^f)$ is small, there is not much difference in the preimage estimated with and without feature f. Thus, imp_f becomes small and feature f is not so important. Similarly, if Frobenius norm of $(\widehat{\mathbf{X}}_b - \widehat{\mathbf{X}}_b^f)$ is high, imp_f is high and feature f becomes important.

We summarize MPFS in Algorithm 1. $g(\cdot)$ denotes a preimage estimation function. Note that the function $g(\cdot)$ takes $\mathbf{K}_{wb}$ (or $\mathbf{K}_{wb}^*$) and $\mathbf{X}$ as input, and outputs $\widehat{\mathbf{X}}_b$ (or $\widehat{\mathbf{X}}_b^f$) respectively at each iteration b for each feature f.

Algorithm 1. MPFS: Feature Selection Algorithm

Initialize $b = 1,\ f = 1,\ \widehat{\mathbf{M}} = \mathbf{0}$
Initialize feature importance vector *imp* with F zeros indexed by $imp_f,\ f = 1, 2, \ldots, F$
for $b = 1 \rightarrow B$ **do**
  for $f = 1 \rightarrow F$ **do**
    Generate sparse random vector $\mathbf{w}_b$
    Use $\mathbf{w}_b$ to calculate $\mathbf{K}_{wb}$
    $\widehat{\mathbf{X}}_b \leftarrow g(\mathbf{K}_{wb}, \mathbf{X})$
    if $\mathbf{w}[f] == 0$ **then**
      Set $\mathbf{w}[f] = 1$ to generate $\mathbf{w}_b^*$
    else
      Set $\mathbf{w}[f] = 0$ to generate $\mathbf{w}_b^*$
    end if
    Use $\mathbf{w}_b^*$ to obtain $\mathbf{K}_{wb}^*$
    $\widehat{\mathbf{X}}_b^f \leftarrow g(\mathbf{K}_{wb}^*, \mathbf{X})$
    $imp_f \leftarrow imp_f + ||\widehat{\mathbf{X}}_b - \widehat{\mathbf{X}}_b^f||_F$
    $\widehat{\mathbf{M}} \leftarrow \widehat{\mathbf{M}} + \widehat{\mathbf{X}}_b$
    $f \leftarrow f + 1$
  end for
  $b \leftarrow b + 1$
end for
$\widehat{\mathbf{X}} = \frac{\widehat{\mathbf{M}}}{B \times F}$
for $f = 1 \rightarrow F$ **do**
  $imp_f \leftarrow \frac{imp_f}{B}$ {importance of fth feature is given by imp_f}
end for

For each iteration b ($b = 1, 2, \ldots, B$), when we use $\mathbf{w}_b$, we are essentially capturing the interaction between that subset of features via kernel matrix $\mathbf{K}_{wb}$. Because the entries in $\mathbf{w}_b$ in each iteration are sparse and independently sampled from a distribution, we work with a diversified ensemble of feature subsets. An advantage is this tends to be more robust towards noisy and irrelevant features in the data. This is important in our case because we do not have noise-free training data for our algorithm. Moreover, this also enables us to work with any existing preimage estimation algorithm for KPCA in the literature. Finally, we also note that in addition to feature selection via MPFS, we also obtain the preimage $\widehat{\mathbf{X}}$ as a result of our algorithm. This suggests that MPFS can also be thought of as a "meta" approach for estimating preimage in KPCA.

4. Experimental results

We evaluate our method on several simulated and real data sets to show its efficacy. Since we know the relevant features in simulated data sets beforehand, we can verify if our algorithm is able to identify them in different scenarios. We first present the results on simulated data sets, and then show the results on real data sets.

4.1. Results on simulated data

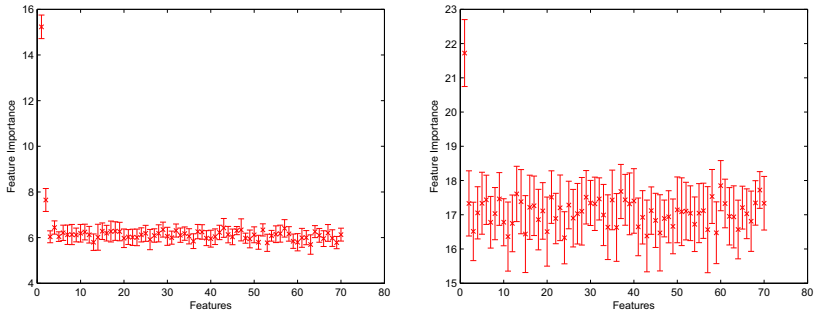
We generate several simulated data sets where each data set has a pattern (linear or non-linear) embedded into it. The pattern is only over a subset of relevant features out of the total set of features, and we want to find those relevant features. Our feature selection methodology can work with any KPCA algorithm. For the purpose of this paper, we use the algorithm proposed in Sahu et al. [17] as the base algorithm. Preliminary experiments did not show sensitivity to B. We set $B = 50$ for all the experiments. For the experiments we set the Gaussian kernel parameter $\sigma = 1$, and the sparseness parameter $\gamma = 1/\sqrt{F}$, where F is the total number of features in the data. However, we conduct some sensitivity experiments to evaluate the role of these parameters. We also vary the noise level in the data through the standard deviation of added Gaussian noise σ_G. We experimented with polynomial kernels, and found the conclusions of the experiments to be the same as those obtained using Gaussian kernel.

The first data set is the *Line2* data set which refers to the fact that the pattern is linear only over two features out of the total set of features. More specifically, the data set consisting of 50 instances and 70 features is generated as follows: $x_1 = 0.1t$ for $t = 1, 2, \ldots, 50$, $x_2 = 0.5(1 - x_1)$, and $x_3, x_4, \ldots, x_{70}$ are independent Gaussian noise with mean 0 and variance σ_G^2, Independent Gaussian noise with mean 0 and variance σ_G^2 are also added to x_1 and x_2. Fig. 2 shows the feature importance as a function of the feature index, along with standard errors for feature importance values obtained by applying MPFS on 10 independent replicates of the data set.

The second data set *Plane5* refers to the fact that the pattern is a plane over five features. The data set consists of 50 instances and 70 features generated as follows: $x_1 = 0.1t,\ t = 1, 2, \ldots, 50$, x_2, x_3, x_4 are independently, Gaussian distributed with mean 0 and variance 1, $x_5 = 1 - 0.2x_1 + 3x_2 + 2x_3 + 0.5x_4$, and $x_6, x_7, \ldots, x_{70}$ are independent, Gaussian noise with mean 0 and variance σ_G^2. Independent Gaussian noise with mean 0 and variance σ_G^2 are added to x_1, x_2, x_3, x_4 and x_5. The results are shown in Fig. 3 (standard errors for feature importance values are obtained from generating different x_2, x_3, x_4 10 times).

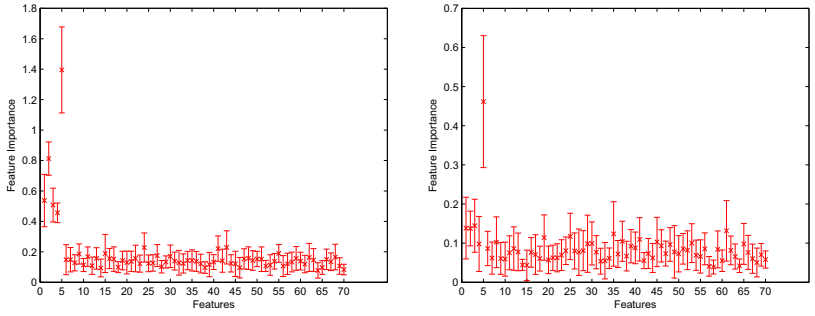
The third data set *Curve3* refers to the fact that the pattern is a curve over three features. The data set consists of 50 data points and 70 features generated as follows: $x_1 = 0.1t,\ t = 1, 2, \ldots, 50$, x_2 is Gaussian distributed with mean 0 and variance 1, $x_3 = x_2^2/x_1$, and $x_4, x_5, \ldots, x_{70}$ are independent, Gaussian noise with mean 0 and variance σ_G^2. Independent Gaussian noise with $\mu = 0$ and variance σ_G^2 are added to x_1, x_2, x_3. Fig. 4 shows the results (standard errors for feature importance values are obtained from generating different x_2 10 times).

The fourth data set *Sphere3* refers to the fact that the pattern is spherical over three features. The data set consists of 50 data points and 70 features generated as follows. The pattern is of the form $x_1^2 + x_2^2 + x_3^2 = 25$ where $x_1 = 5\sin(t)\cos(t)$, $x_2 = 5\sin(t)$



(a) $\sigma_G = 0.9$ (b) $\sigma_G = 3$

Fig. 2. Feature importance plots for MPFS applied to the *Line2* data set with for selected values of noise σ_G. The procedure is replicated 10 times to obtain standard errors for feature importance. Under moderate noise ($\sigma_G = 0.9$), x_1 and x_2 can be identified as relevant features visually.



(a) $\sigma_G = 0.9$ (b) $\sigma_G = 3$

Fig. 3. Feature importance plots for MPFS applied to the *Plane5* data set for selected values of noise σ_G. The procedure is replicated 10 times to obtain standard errors for feature importance. Under moderate noise ($\sigma_G = 0.9$), $x_1,\ x_2,\ x_3,\ x_4,\ x_5$ can be identified as relevant features visually.

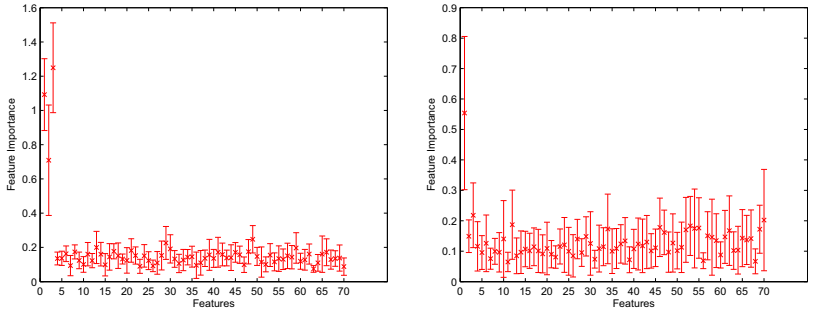
$\sin(t)$, $x_3 = 5\cos(t)$, for $t = 1,2,\ldots,50$, and $x_4, x_5, \ldots, x_{70}$ are independent, Gaussian noise with mean 0 and variance σ_G^2. Independent, Gaussian noise noise with mean 0 and variance σ_G^2 are added to x_1, x_2, x_3. Fig. 5 shows the results (standard errors for feature importance values are obtained from 10 replicates).

To evaluate the sensitivity of our results to the parameters involved (σ and γ), we conducted the above experiments on *Line2* and *Sphere3* datasets setting ($\sigma_G \in \{0.9, 3\}$). Specifically, we took $\sigma \in 0.1, 10$ and $\gamma \in \{\frac{2}{\sqrt{70}}, \frac{3}{\sqrt{70}}\}$. Figs. 6–9 show the results. To see the effect of intermediate noise levels on *Line2* and *Sphere3* datasets, we chose ($\sigma_G \in \{1.5, 2\}$) keeping $\sigma = 1$ and $\gamma = \frac{1}{\sqrt{70}}$. Fig. 10 shows the results (standard errors are obtained from 10 replicates). We see that for all datasets corrupted with a medium level of noise, our algorithm is able to detect the important features. However, when we increase the noise level to high ($\sigma_G = 3$), the algorithm cannot detect all the relevant features. Thus, our algorithm works well for cases with moderate noise levels.

To compare our approach, we tested the algorithm in Muniz et al. [23] on the *Line2* and *Sphere3* data sets with $\sigma_G = 0.9$. Fig. 11 shows the results. In both cases, it is not able to identify all the relevant features. We also found that the algorithm in Muniz et al. [23] is able to identify the relevant features in the data sets when noise is not added. However, even in the presence of moderate amount of noise, its performance deteriorates as shown in Fig. 11.

4.2. Results on real data sets

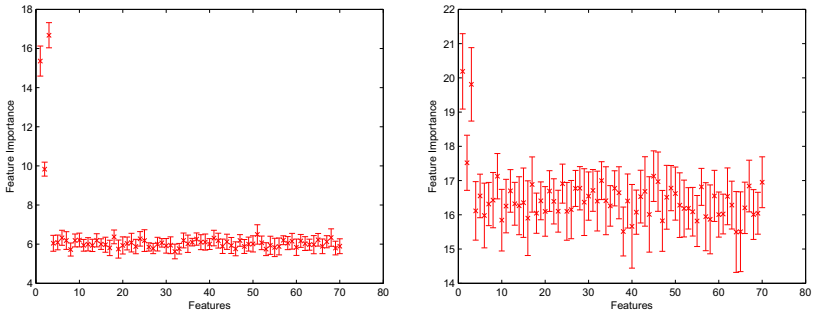
We also evaluated MPFS on several real data sets available in UCI Machine Learning Repository Bache and Lichman [28] (the



(a) $\sigma_G = 0.9$ (b) $\sigma_G = 3$

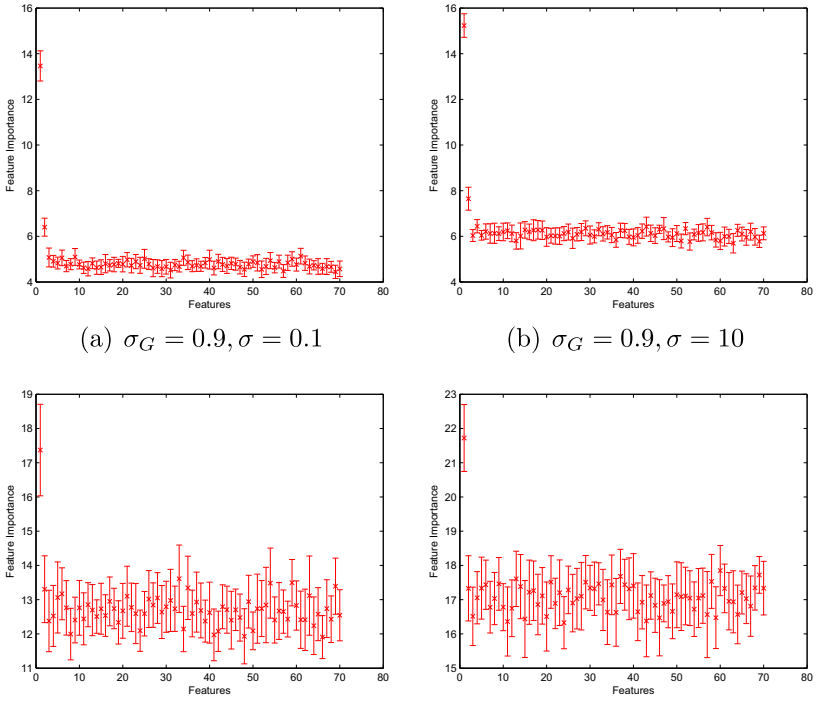
Fig. 4. Feature importance plots for our algorithm applied to the *Curve3* data set for selected values of noise σ_G. The procedure is replicated 10 times to obtain standard errors for feature importance. Under moderate noise ($\sigma_G = 0.9$), $x_1,\ x_2,\ x_3$ can be identified as relevant features visually.

Please cite this article in press as: A. Sahu et al., Feature selection for noisy variation patterns using kernel principal component analysis, Knowl. Based Syst. (2014), http://dx.doi.org/10.1016/j.knosys.2014.08.027



(a) $\sigma_G = 0.9$ (b) $\sigma_G = 3$

Fig. 5. Feature importance plots for our algorithm applied to the *Sphere3* dataset for selected values of noise σ_G. The procedure is replicated 10 times to obtain standard errors for feature importance. Under moderate noise ($\sigma_G = 0.9$), $x_1,\ x_2,\ x_3$ can be identified as relevant features visually.



(a) $\sigma_G = 0.9, \sigma = 0.1$ (b) $\sigma_G = 0.9, \sigma = 10$

(c) $\sigma_G = 3, \sigma = 0.1$ (d) $\sigma_G = 3, \sigma = 10$

Fig. 6. Feature importance plots to illustrate the sensitivity of our algorithm to the kernel parameter σ applied to the *Line2* data set with selected values of noise σ_G. Under moderate noise ($\sigma_G = 0.9$), $x_1,\ x_2$ can be identified as relevant features visually for different values of σ.

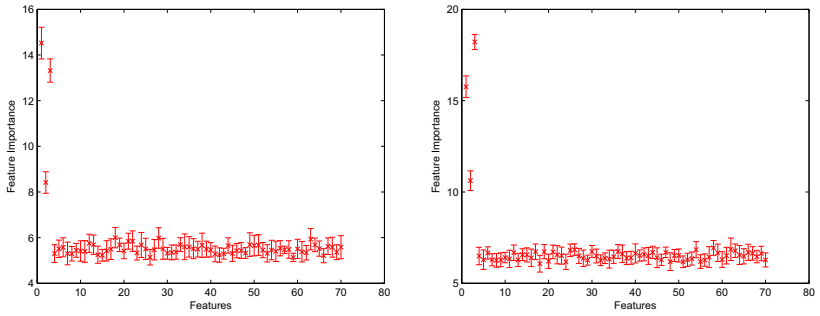
address of the website is http://archive.ics.uci.edu/ml/). For our experiments, we chose *Wine*, *Ionosphere*, *Sonar*, and *Hill Valley* data sets. The details of these datasets as well as results obtained from our experiments are described subsequently. We compare MPFS to baseline filter based feature selection approaches (*T*-test, Information Gain Ratio (IGR)) as well as sophisticated unsupervised feature selection procedures described before like LS, MCFS, and UDFS. For each data set in our experiment, we first standardize all the features so that each feature has zero mean and unit variance. Gaussian noise with mean 0 and variance $\sigma_G^2 = 0.81$ is then added to the data set. Different feature selection procedures are applied with the appropriate parameter settings described by the authors of the corresponding papers, and the selected features are fed to a classifier (logistic regression) where the results are computed with 10-fold cross-validation. The entire procedure is repeated 10 times, and average values of salient metrics for classification like *Accuracy*, *True Positive Rate*(*TPR*), *False Positive Rate* (*FPR*), *Precision*, *F-Measure* and *Area under Curve*(*AUC*) are reported (*Recall* is equivalent to *TPR*, and hence not reported). We also show the average values of these metrics when all the features are used for classification.

4.2.1. Results for wine data set

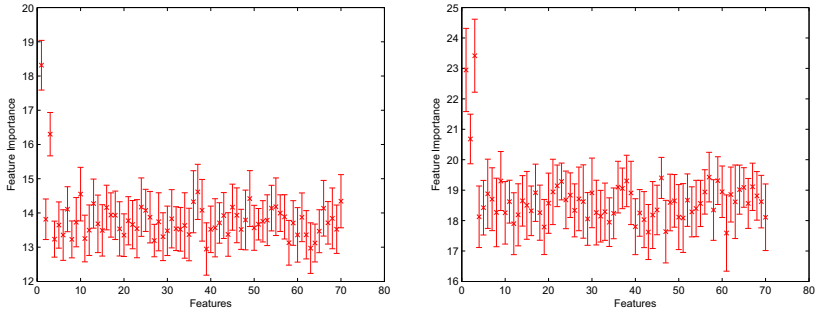
Wine data set has 178 instances, 13 features, and 3 classes. For each instance belonging to a type of wine, each of the features corresponds to the quantities of a particular constituent for the class of wine based on chemical analysis. The data is obtained from three different cultivars in Italy. Since there are 3 classes, we used a multinomial logistic regression. Here we used Chi-squared based feature selection instead of *T*-test as a baseline filter because of 3 classes.

The results of our experiments are shown in Table 1. We see that MPFS obtained the same results as UDFS which is the best result among different feature selection methods in terms of all the classification metrics. It is noteworthy that even if both these algorithms are unsupervised in nature, the classification metrics

Please cite this article in press as: A. Sahu et al., Feature selection for noisy variation patterns using kernel principal component analysis, Knowl. Based Syst. (2014), http://dx.doi.org/10.1016/j.knosys.2014.08.027

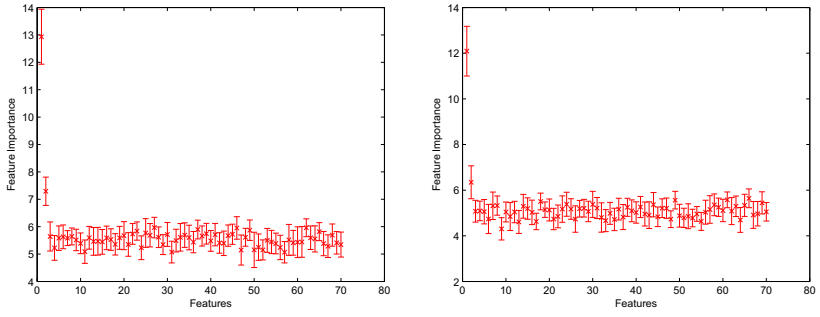


(a) $\sigma_G = 0.9, \sigma = 0.1$ (b) $\sigma_G = 0.9, \sigma = 10$

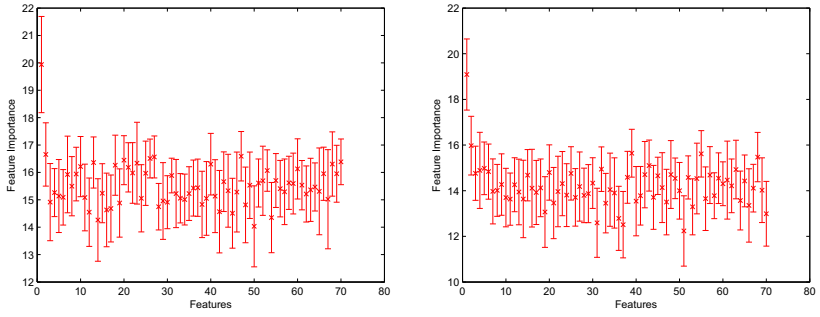


(c) $\sigma_G = 3, \sigma = 0.1$ (d) $\sigma_G = 3, \sigma = 10$

Fig. 7. Feature importance plots to illustrate the sensitivity of our algorithm to the kernel parameter σ applied to the *Sphere3* data set with selected noise σ_G. Under moderate noise ($\sigma_G = 0.9$), $x_1,\ x_2,\ x_3$ can be identified as relevant features visually for different values of σ.



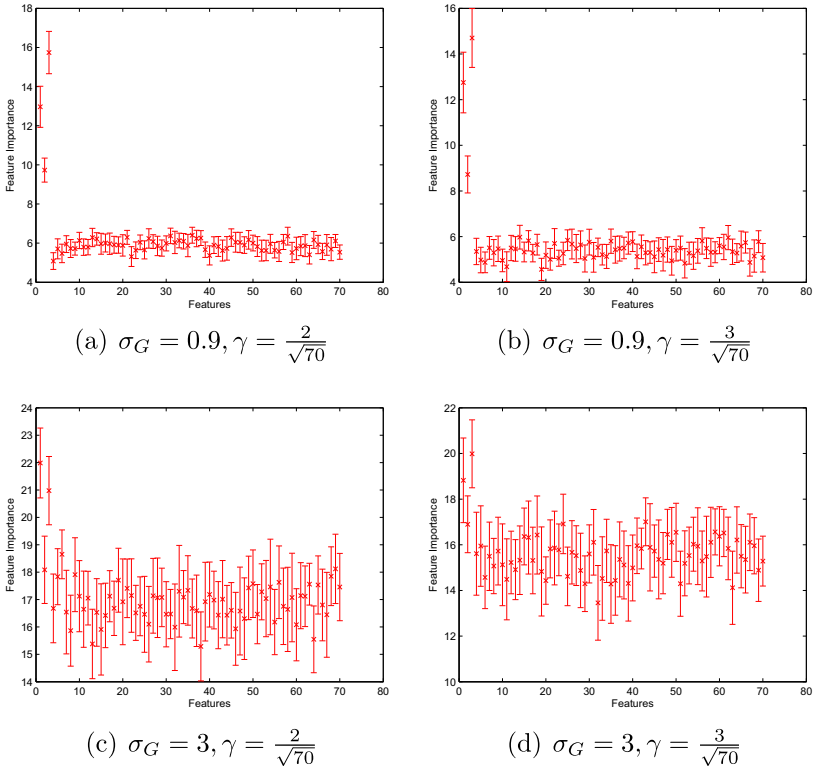
(a) $\sigma_G = 0.9, \gamma = \frac{2}{\sqrt{70}}$ (b) $\sigma_G = 0.9, \gamma = \frac{3}{\sqrt{70}}$



(c) $\sigma_G = 3, \gamma = \frac{2}{\sqrt{70}}$ (d) $\sigma_G = 3, \gamma = \frac{3}{\sqrt{70}}$

Fig. 8. Feature importance plots to illustrate the sensitivity of our algorithm to the sparseness parameter γ applied to the *Line2* data set with selected noise σ_G. Under moderate noise ($\sigma_G = 0.9$), $x_1,\ x_2$ can be identified as relevant features visually for different values of γ.

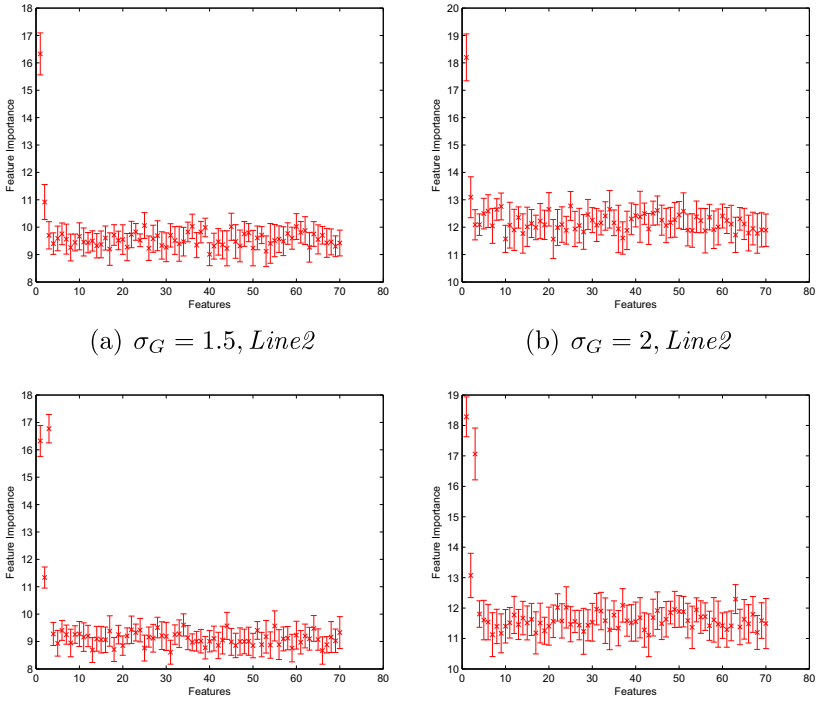
Please cite this article in press as: A. Sahu et al., Feature selection for noisy variation patterns using kernel principal component analysis, Knowl. Based Syst. (2014), http://dx.doi.org/10.1016/j.knosys.2014.08.027



(a) $\sigma_G = 0.9, \gamma = \frac{2}{\sqrt{70}}$ (b) $\sigma_G = 0.9, \gamma = \frac{3}{\sqrt{70}}$

(c) $\sigma_G = 3, \gamma = \frac{2}{\sqrt{70}}$ (d) $\sigma_G = 3, \gamma = \frac{3}{\sqrt{70}}$

Fig. 9. Feature importance plots to illustrate the sensitivity of our algorithm to the sparseness parameter γ applied to the *Sphere3* data set with selected values of noise σ_G. Under moderate noise ($\sigma_G = 0.9$), $x_1,\ x_2,\ x_3$ can be identified as relevant features visually for different values of γ.

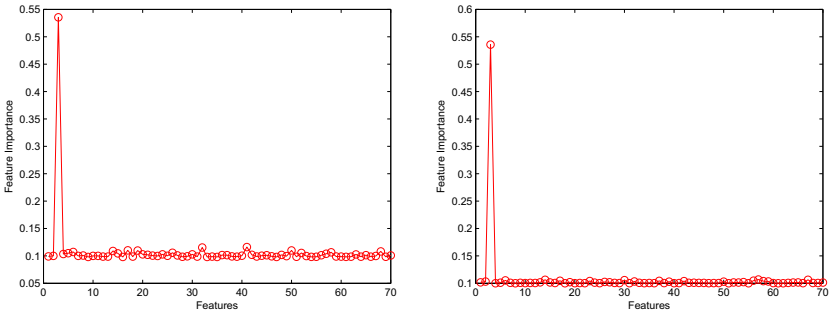


(a) $\sigma_G = 1.5, Line2$ (b) $\sigma_G = 2, Line2$

(c) $\sigma_G = 1.5, Sphere3$ (d) $\sigma_G = 2, Sphere3$

Fig. 10. Feature importance plots for our algorithm for selected values of σ_G added to the data sets. As the level of added noise (σ_G) increases, it becomes difficult to identify all the relevant features.

Please cite this article in press as: A. Sahu et al., Feature selection for noisy variation patterns using kernel principal component analysis, Knowl. Based Syst. (2014), http://dx.doi.org/10.1016/j.knosys.2014.08.027



(a) *Line2* dataset (b) *Sphere3* dataset

Fig. 11. Feature importance plots for *Line2* and *Sphere3* data sets based on the algorithm by Muniz et al. [23]. Based on the feature importance values, it is not able to identify all the relevant features when the data sets are corrupted with moderate noise ($\sigma_G = 0.9$).

Table 1
Classification results for different feature selection methods for *Wine* dataset. "All" refers to the fact that all features were used by the classifier (no feature selection), and entries for this column are italicized. "CHI" refers to Chi-squared based feature selection method. The best entries for each metric among the feature selection methods only are marked in bold.

Metric	All	CHI	IGR	LS	MCFS	UDFS	MPFS
Accuracy	*0.971*	0.949	0.949	0.955	0.966	**0.971**	**0.971**
TPR	*0.972*	0.949	0.949	0.955	0.966	**0.972**	**0.972**
FPR	*0.014*	0.027	0.027	0.023	0.017	**0.013**	**0.013**
Precision	*0.972*	0.95	0.95	0.956	0.966	**0.973**	**0.973**
F − Measure	*0.972*	0.949	0.949	0.955	0.966	**0.972**	**0.972**
AUC	*0.999*	0.986	0.986	0.974	0.987	**0.996**	**0.996**

Table 3
Classification results for different feature selection methods for *Sonar* dataset. "All" refers to the fact that all features were used by the classifier (no feature selection), and entries for this column are italicized. The best entries for each metric among the feature selection methods only are marked in bold.

Metric	All	*T*-test	IGR	LS	MCFS	UDFS	MPFS
Accuracy	*0.730*	0.721	0.735	0.735	0.754	0.711	**0.759**
TPR	*0.731*	0.721	0.736	0.736	0.755	0.712	**0.76**
FPR	*0.273*	0.284	0. 269	0.266	**0.249**	0.295	**0.249**
Precision	*0.731*	0.721	0.735	0.736	0.755	0.711	**0.76**
F − Measure	*0.731*	0.721	0.735	0.736	0.755	0.711	**0.758**
AUC	*0.763*	0.779	0.8	0.811	0.818	0.749	**0.822**

are very close to those obtained using all the features. We also see that CHI and IGR methods gave similar results, and MPFS performed better than these baseline filter methods which utilize the information about the class label in the dataset.

4.2.2. Results for ionosphere data set

Ionosphere data set has 351 instances, 34 features, and 2 classes. The instances belonging to "Good" class convey the fact there is evidence of some type of structure in the ionosphere whereas the instances belonging to "Bad" class do not show the evidence of any structure. The data is obtained from Goose Bay, Labrador. Since there are 2 classes, we used two-class logistic regression for classification.

The results of our experiments are shown in Table 2. We can see that overall MPFS performed better in all metrics (except *F − measure* and *AUC* where it is slightly worse) among different feature selection methods. We also note that using the total set of features provided the best results.

4.2.3. Results for sonar data set

Sonar data set has 208 instances, 60 features, and 2 classes out of which 111 instances belong to the "Mine" class and rest 97 instances belong to "Rock" class. The features for "Mine" class are obtained by recording sonar signals bouncing off a metal cylinder at different angles whereas the features for "Rock" class are obtained under similar conditions from rocks. We used two-class logistic regression for classification.

The results of our experiments are shown in Table 3. We can see that MPFS performed the best in all metrics among different feature selection methods. Moreover, the performance of MPFS is slightly better than MCFS. Thus, both MPFS and MCFS deliver the best results. It is noteworthy that although MPFS and MCFS are unsupervised, the features provided by them perform better classification than the case when all features are selected.

4.2.4. Results for hill valley data set

Hill Valley data set has 606 instances, 100 features, and 2 classes. Essentially, each instance is a sequence of 100 points classified as "Hill" if a "bump" is observed when plotting the points on a two-dimensional plot or classified as "Valley" if a dip is observed. For our purpose, we took the dataset with noise where the bump/dip is not trivial to observe. Again we used two-class logistic regression for classification.

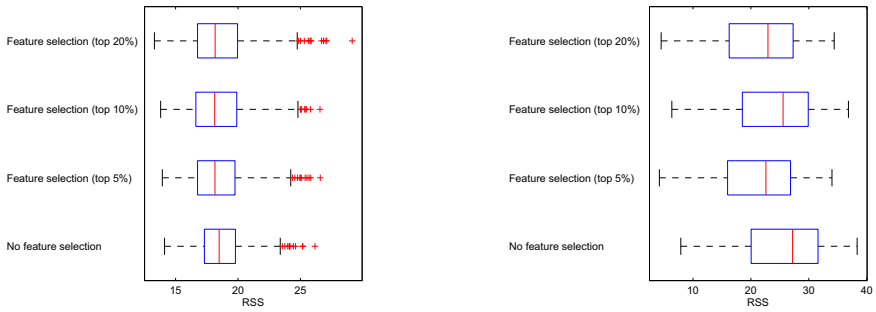
Table 2
Classification results for different feature selection methods for *Ionosphere* dataset. "All" refers to the fact that all features were used by the classifier (no feature selection), and entries for this column are italicized. The best entries for each metric among the feature selection methods only are marked in bold.

Metric	All	*T*-test	IGR	LS	MCFS	UDFS	MPFS
Accuracy	*0.888*	0.712	0.811	0.806	0.814	0.803	**0.817**
TPR	*0.889*	0.712	0.812	0.806	0.815	0.803	**0.818**
FPR	*0.153*	0.5	0.259	0.301	0.264	0.302	**0.284**
Precision	*0.889*	0.761	0.81	0.812	0.814	0.808	**0.824**
F − Measure	*0.887*	0.653	0.807	0.795	**0.809**	0.793	0.808
AUC	*0.87*	0.535	**0.843**	0.74	0.799	0.719	0.818

Table 4
Classification results for different feature selection methods for *Hill Valley* dataset. "All" refers to the fact that all features were used by the classifier (no feature selection), and entries for this column are italicized. The best entries for each metric among the feature selection methods only are marked in bold.

Metric	All	*T*-test	IGR	LS	MCFS	UDFS	MPFS
Accuracy	*0.526*	0.486	0.511	0.615	**0.625**	0.608	0.622
TPR	*0.526*	0.487	0.512	0.616	**0.625**	0.609	0.622
FPR	*0.474*	0.522	0.489	0.392	**0.382**	0.399	0.385
Precision	*0.526*	0.468	0.511	0.657	**0.687**	0.664	0.666
F − Measure	*0.526*	0.422	0.511	0.585	0.589	0.57	**0.593**
AUC	*0.524*	0.447	0.494	0.645	0.665	0.652	**0.67**

Please cite this article in press as: A. Sahu et al., Feature selection for noisy variation patterns using kernel principal component analysis, Knowl. Based Syst. (2014), http://dx.doi.org/10.1016/j.knosys.2014.08.027



(a) RSS results for KPCA algorithm by Kwok and Tsang (2004)

(b) RSS results for KPCA algorithm by Zheng et al. (2010)

Fig. 12. RSS obtained with and without feature selection for two popular KPCA preimage estimation algorithms. Feature selection shows statistically significant improvement in RSS.

The results of our experiments are shown in Table 4. We can see that the performance of MPFS was very competitive with MCFS. It was slightly worse than MCFS in terms of *Accuracy*, *TPR*, *FPR*, *Precision* but was better in $F-measure$, *AUC*. It is noteworthy that MCFS and MPFS select features that result in much better classification metrics than those obtained using all the features.

Overall we can see that MPFS resulted in the best (very close to best in case of *Hill Valley* dataset) classification performance. Although MPFS is unsupervised, it outperformed the baseline filter based feature selection procedures which make use of the information about class label.

4.3. Results on face image data

A real-world application of KPCA is denoising images. For our purpose, we chose a real face data set available at http://isomap.stanford.edu/datasets.html. There are 698 data points and the dimensionality of each data point is 4096. We took all 698 images, added independent Gaussian noise ($\mu = 0$ and $\sigma_G = 0.5$) to the images to create the noisy data set, and subsequently denoised the test set by obtaining preimages. We chose the algorithms proposed by Kwok and Tsang [14] and Zheng et al. [15] to obtain preimages. To evaluate the efficacy of our feature selection procedure, we computed residual error (RSS) for each image with and without feature selection. RSS is computed as follows: Let $\mathbf{x}_i,\ i = 1,\ldots,N,\ \mathbf{x}_k \in R^p$ represent a set of N observations in the input space. We estimate the preimage residual root sum of squared error (RSS) for a method by calculating the Euclidean distance between the obtained preimage $\hat{\mathbf{x}}$ and its true image $\mathbf{t}$ as shown in Eq. (8).

$$\mathrm{RSS} = \sqrt{\sum_{i=1}^{N} (\hat{\mathbf{x}}_i - \mathbf{t}_i)^2} \tag{8}$$

A smaller value of RSS shows better performance. We note that Eq. (8) can be used to calculate the RSS for each image by setting $N = 1$ for that particular image. Also we decided to choose three levels – top 5%, top 10% and top 20% – of the features when performing feature selection. We then computed the RSS first without performing feature selection and then selecting features at the three levels described previously. Fig. 12 shows the boxplot results of RSS obtained for both algorithms under the scenario described above.

We computed the difference between the RSS value obtained without feature selection, and then selected features at each of the three levels for each of the 698 instances for both the algorithms. We then performed a one-sided Wilcoxon signed-rank test (the alternate hypothesis is that the median difference is greater than zero). The p-values obtained for all of the tests were smaller than 0.001. Thus, MPFS on the preimage estimation algorithms provides statistically significant improvement over the results obtained from without using MPFS.

5. Conclusion and further study

A new feature selection algorithm (MPFS) for KPCA for the case of noisy training data is presented. The data points are projected onto multiple sparse random subsets of features, and then a feature importance measure is calculated by denoising the data matrix using matched pairs of projections (with and without a feature). An advantage of working with an ensemble of feature subsets is they tend to be more robust towards noisy and irrelevant features in the data. Also, our feature selection methodology can used with any suitable KPCA algorithm available in the literature. Finally, we also note that in addition to feature selection via MPFS, we can also obtain the preimage. This suggests that MPFS can also be thought of as a "meta" approach for estimating preimage in KPCA. We demonstrate the effectiveness of our algorithm on several simulated and real data sets.

Since the focus of this paper is on feature selection, we did not investigate fully the "meta" approach ability of MPFS to estimate the preimage for KPCA. Given the fact that preimage estimation is important for many real applications like image denoising, manufacturing variation analysis, etc., we plan to work on it in future.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 0825331. We also thank the reviewers for their constructive feedback.

References

[1] B. Schölkopf, A.J. Smola, K.R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319.
[2] M. Aizerman, E. Braverman, L. Rozonoer, Theoretical foundations of the potential function method in pattern recognition learning, Automat. Rem. Contr. 25 (1964) 821–837.
[3] S.W. Choi, I.-B. Lee, Nonlinear dynamic process monitoring based on dynamic kernel PCA, Chem. Eng. Sci. 59 (24) (2004) 5897–5908.
[4] X. Liu, U. Kruger, T. Littler, L. Xie, S. Wang, Moving window kernel PCA for adaptive monitoring of nonlinear processes, Chemometr. Intell. Lab. Syst. 96 (2) (2009) 132–143.
[5] Z. Ge, C. Yang, Z. Song, Improved kernel PCA-based monitoring approach for nonlinear processes, Chem. Eng. Sci. 64 (9) (2009) 2245–2255.
[6] J.-M. Lee, C. Yoo, I.-B. Lee, Fault detection of batch processes using multiway kernel principal component analysis, Comput. Chem. Eng. 28 (9) (2004) 1837–1847.

[7] S.W. Choi, C. Lee, J.-M. Lee, J.H. Park, I.-B. Lee, Fault detection and identification of nonlinear processes based on kernel PCA, Chemometr. Intell. Lab. Syst. 75 (1) (2005) 55–67.
[8] P. Cui, J. Li, G. Wang, Improved kernel principal component analysis for fault detection, Expert Syst. Appl. 34 (2) (2008) 1210–1219.
[9] R. Sun, F. Tsung, L. Qu, Evolving kernel principal component analysis for fault diagnosis, Comput. Ind. Eng. 53 (2) (2007) 361–371.
[10] H. Hoffmann, Kernel PCA for novelty detection, Pattern Recogn. 40 (3) (2007) 863–874.
[11] Y. Gu, Y. Liu, Y. Zhang, A selective kernel PCA algorithm for anomaly detection in hyperspectral imagery, in: IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 2, II–II, 2006.
[12] S. Mika, B. Schölkopf, A.J. Smola, K.R. Müller, M. Scholz, G. Rätsch, Kernel PCA and de-noising in feature spaces, in: NIPS, 1998, pp. 536–542.
[13] G.H. Bakir, J. Weston, B. Schölkopf, Learning to find pre-images, in: NIPS, 2003.
[14] J. Kwok, I. Tsang, The pre-image problem in kernel methods, IEEE Trans. Neural Netw. 15 (2004) 1517–1525.
[15] W.S. Zheng, J. Lai, P.C. Yuen, Penalized preimage learning in kernel principal component analysis, IEEE Trans. Neural Netw. 21 (4) (2010) 551–570.
[16] A. Shinde, A. Sahu, D. Apley, G. Runger, Preimages for variation patterns from kernel PCA and bagging, IIE Trans. 46 (5) (2014) 429–456.
[17] A. Sahu, G. Runger, D. Apley, Image denoising with a multi-phase kernel principal component approach and an ensemble version, in: IEEE Applied Imagery Pattern Recognition Workshop (AIPR), 2011, pp. 1–7.
[18] G.I. Allen, Automatic feature selection via weighted kernels and regularization, J. Comput. Graph Stat. 22 (2) (2013) 284–299.
[19] S. Maldonado, R. Weber, J. Basak, Simultaneous feature selection and classification using kernel-penalized support vector machines, Inf. Sci. 181 (1) (2011) 115–128.
[20] S. Maldonado, R. Weber, Feature selection for support vector regression via Kernel penalization, in: IJCNN, 2010, pp. 1–7.
[21] M. Wu, B. Schölkopf, A local learning approach for clustering, in: NIPS, 2006, pp. 1529–1536.
[22] H. Zeng, Y. ming Cheung, Feature selection and kernel learning for local learning-based clustering, IEEE Trans. Pattern Anal. Mach. Intell. 33 (8) (2011) 1532–1547.
[23] V. Muniz, J.V. Horebeek, R. Ramos, Measuring the importance of variables in kernel PCA, in: COMPSTAT, 2008, pp. 517–524.
[24] X. He, D. Cai, P. Niyogi, Laplacian score for feature selection, in: Y. Weiss, B. Schölkopf, J. Platt (Eds.), Advances in Neural Information Processing Systems, vol. 18, MIT Press, 2006, pp. 507–514. <http://papers.nips.cc/paper/2909-laplacian-score-for-feature-selection.pdf>.
[25] D. Cai, C. Zhang, X. He, Unsupervised feature selection for multi-cluster data, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2010, pp. 333–342.
[26] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in: NIPS, vol. 14, 2001, pp. 585–591.
[27] Y. Yang, H.T. Shen, Z. Ma, Z. Huang, X. Zhou, $l_{2,1}$-Norm regularized discriminative feature selection for unsupervised learning, in: IJCAI, 2011, pp. 1589–1594.
[28] K. Bache, M. Lichman, UCI Machine Learning Repository, 2013. <http://archive.ics.uci.edu/ml>.

Please cite this article in press as: A. Sahu et al., Feature selection for noisy variation patterns using kernel principal component analysis, Knowl. Based Syst. (2014), http://dx.doi.org/10.1016/j.knosys.2014.08.027