Non-Linear Variation Patterns and Kernel Preimages

by

Anshuman Sahu

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved November 2013 by the
Graduate Supervisory Committee:

George C. Runger, Chair
Teresa Wu
Rong Pan
Ross Maciejewski

ARIZONA STATE UNIVERSITY

December 2013

ABSTRACT

Identifying important variation patterns is a key step to identifying root causes of process variability. This gives rise to a number of challenges. First, the variation patterns might be non-linear in the measured variables, while the existing research literature has focused on linear relationships. Second, it is important to remove noise from the dataset in order to visualize the true nature of the underlying patterns. Third, in addition to visualizing the pattern (preimage), it is also essential to understand the relevant features that define the process variation pattern.

This dissertation considers these variation challenges. A base kernel principal component analysis (KPCA) algorithm transforms the measurements to a high-dimensional feature space where non-linear patterns in the original measurement can be handled through linear methods. However, the principal component subspace in feature space might not be well estimated (especially from noisy training data). An ensemble procedure is constructed where the final preimage is estimated as the average from bagged samples drawn from the original dataset to attenuate noise in kernel subspace estimation. This improves the robustness of any base KPCA algorithm.

In a second method, successive iterations of denoising a convex combination of the training data and the corresponding denoised preimage are used to produce a more accurate estimate of the actual denoised preimage for noisy training data. The number of primary eigenvectors chosen in each iteration is also decreased at a constant rate. An efficient stopping rule criterion is used to reduce the number of iterations.

A feature selection procedure for KPCA is constructed to find the set of relevant features from noisy training data. Data points are projected onto sparse random vectors. Pairs of such projections are then matched, and the differences in variation patterns within pairs are used to identify the relevant features. This approach provides robustness to irrelevant features by calculating the final variation pattern from an ensemble of feature subsets.

Experiments are conducted using several simulated as well as real-life data sets. The proposed methods show significant improvement over the competitive methods.

To my beloved

ACKNOWLEDGMENTS

I want to express my utmost gratitude to my advisor, Professor George Runger, for his strong belief in me. He instilled a spirit to discover practical solutions to problems. I am deeply grateful to him for his kind support as well as insightful discussions through out my study at ASU that have shaped me as a researcher.

My sincere thanks to the esteemed members of my dissertation committee-Professor Teresa Wu, Professor Rong Pan, and Professor Ross Maciejewski-for providing valuable comments and sharing their nuggets of wisdom for both my dissertation as well as my future life.

I owe everything to my parents for raising me with strong values. Whatever good is in me is because of them. I will always be indebted to them for their unconditional love. They have given me their best gift in the form of my brother who strongly believes in me. His unflinching support gives me strength. I cannot ask for a better brother than him. I always look forward to the affections showered on me by my grandma. Also I miss my grandpa from whom I would have learnt a lot.

I am blessed with a wonderful lady whose elegance is superlative. She has brought a constant influx of togetherness into my life. You are mine and I am yours sweetheart.

My labmates in ASU have been very supportive. I am grateful to my current labmates Bahar and Erik for useful discussions on a lot of topics. My former labmates Houtao, Mustafa, Anirudh, and Amit have provided many useful insights. Also Houtao and Kabir have given me invaluable advice. It has been a nice experience to be a part of the gang. It is difficult to enlist the names of all of my friends in ASU with whom I spent quality time. I am thankful to everyone of them. I will always remember my enjoyable trip to Sonora with Billi. I am also thankful to Brint for providing prompt technical support for software installation as well as troubleshooting whenever the need arose. Outside academic circle, Priyesh, Sree, Minith, and Ninad shared fun-filled hours of camaraderie. Special thanks to my friends outside Arizona- Sheru, Maddy, Yash, Payoj, and Deepak- with whom I spent memorable time during my vacations. All these people conspired to make my stay worthwhile.

Finally, my obeisance to my God who is always with me. He never forgets me, and blesses me with everything I achieve.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

CHAPTER 1

INTRODUCTION

Faced with the challenge of rapidly evolving customer demand, manufacturing companies strive to improve upon their existing line of products. This requires a fundamental understanding of different operations involved in manufacturing products. There is considerable interest in reducing the variance in the dimensions of final products so as to ensure a higher proportion of them confirm to the desired specifications and are defect-free. Thus, the need is to identify the sources of variation in the products.

Automation in manufacturing processes was introduced to help achieve mass production of goods at an economical rate. With more and more customers demanding a high level of performance from the products they use, it has become imperative to improve our manufacturing processes which in turn neccesitates the use of automated monitoring and inspection systems. Many organizations now collect massive amounts of in-process data with the foresight of potentially using information hidden in it to identify root causes of product or process variation. Advances in measurement and data storage technologies have made it possible to track hundreds, or even thousands of dimensional characteristics with a 100% sample rate. Datasets in the form of a spatial or time series as well as images are common in modern manufacturing. Semiconductor processing, for example, involves inputs from multiple variables; each represented by a finite time series. Thus, each run is characterized by thousands of measurements.

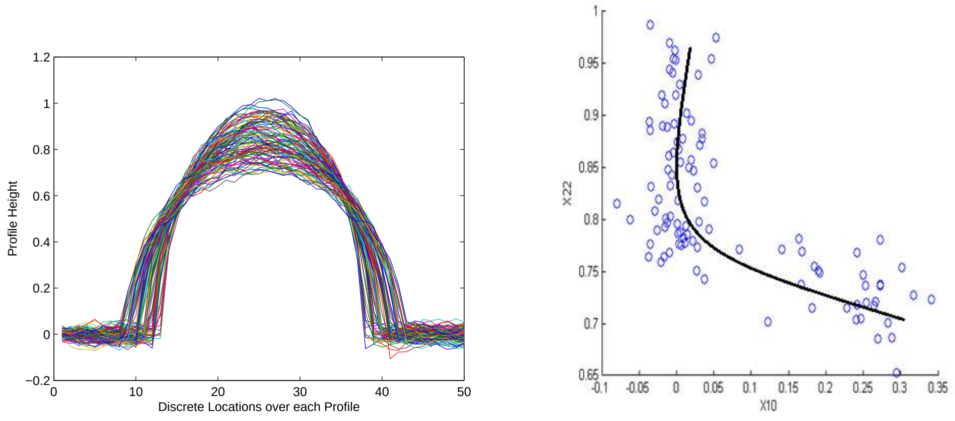
Machine-vision systems (MVS) are widely used in many industries including medical, transportation, construction, and other industrial applications. In particular, [12] provide an in-depth survey on industrial applications which include identification of structural, surface, and operational defects. A key aspect of any MVS is acquisition and analysis of images. [15] discuss about the acquisition of grayscale or binary images for MVS. Our proposed methodology deals with analysis of such images.

More recently, [25] show how 3D laser scanners have become popular in scanning complex manufactured part geometries. The data generated from such scanners is referred to as point cloud data. The point cloud represent a set of points measured in three-dimensional cartesian coordinate system. The point cloud data provide an accurate representation of the scanned object.

In addition to image and point cloud data, profile data is also widely prevalent in manufacturing industry, especially in paper processing and autobody assemblies. The profiles represent measurements taken at several points along each part in a two-dimensional plot. We present an example of the profile data in manufacturing automotive engine gaskets. One critical-to-quality feature is a bead on the gasket, the purpose of which is to create a tight seal. Figure 1(a) shows a set of profiles measured across 100 gasket beads. Each profile is obtained by scanning a stylus across the gasket bead. Each profile in Figure 1 has been discretized into 50 points evenly spaced over the horizontal axis. Hence, the measurement vector **x** for each part consists of the vertical axis profile heights at the 50 locations. We can see that the raw data collected by sensors is inherently noisy, but buried in the noise is a pronounced systematic part-to-part variation pattern, by which the gasket bead is flattening and elongating by varying amounts on each part. Here, the pattern in the data (as represented by the relationships between the different elements of **x**) is non-linear. For example, consider the scatterplot between the sensor recordings at position 10 and position 22 (denoted by $x10$ and $x22$, respectively) as shown in Figure 1(b). It is clear from the plot that there are non-linear relationships between the sensor recordings (features) in the data. For simplicity, we showed the plot between two features, but the actual non-linear pattern can involve more than only two features.

The gasket example shows a pattern when a single source of variation (corresponding to bead flattening/elongation) is present in the process. In practice, multiple variation sources and their corresponding effects on process and product characteristics are embedded in the collected data. Each variation source and its interaction with other sources results in a unique non-linear pattern in the data. Identifying and segregating important variation patterns from the irrelevant ones is then a key step to identifying root causes of process variability. Thus, in order to diagnose and control process or product quality problems, a method is required to identify the sources of variability. Also irrelevant features, in addition to noise in the data, tend to make it harder for the underlying pattern to be recognized and visualized. Therefore, we face two challenges. First it is important to remove noise from the dataset in order to visualize the true nature of the underlying patterns. Second in addition to visualizing the pattern, it is also essential to understand the relevant process features. More specifically, the process inherently manifests itself in a small number of features out of the overall set of features that are recorded. It is, therefore, crucial to discern this set of relevant features that define the process variation pattern.

Principal component analysis (PCA) [10] is a widely used technique in identifying patterns in the data. Given a p-dimensional random vector $\mathbf{x}$, signals often tend to locate on some d-dimensional manifold in p-dimensional space ($d < p$) while noise tends to be less structured. PCA partitions the p-dimensional space into a d-dimensional signal space and a



(a) Profiles (b) Scatterplot

Figure 1. Left: profile measurements for 100 gaskets. Each gasket height is measured at 50 equally-spaced positions. Right: scatterplot between sensor recordings at positions 10 and 22 (denoted by $x10$ and $x22$, respectively)

$(p-d)$ dimensional noise space. A drawback of PCA is in its assumption that the variation pattern is formed by linear combination of variables. PCA tends to lose its effectiveness in identifying patterns when the resultant pattern is nonlinear in nature [22]. One approach to nonlinear PCA is based on principal curves [7]. Apley *et al* [3] used principal curves to identify and visualize nonlinear patterns in data. Schölkopf *et al* [22] extended the linear PCA framework to account for nonlinear structures in the data set through kernel principal component analysis (KPCA). KPCA works on the principle of mapping the data in the input space to a higher dimensional feature space via a nonlinear map $\varphi : R^p \rightarrow R^m$ where m is the number of features and the feature space is also denoted as F. Linear PCA is applied to the mapped points in the feature space to extract components that are nonlinear in the original input space. Because the feature space can have a very large number of dimensions, such explicit mapping can be computationally expensive. Linear PCA in the feature space depends on the data only through inner products of the feature vectors. Schölkopf *et al* used kernel functions to compute the inner products in the feature space without carrying out the mapping explicitly [1]. Similar to linear PCA, projections onto a smaller subset of principal component directions in the feature space can be used to denoise signals.

To interpret the denoised signal, it is valuable to visualize it in the original input space. However, because the projections in feature space are on a subspace that might not correspond to the manifold of the original space, such an inverse transformation from the feature space to the input space does not typically exist [16]. Directly visualizing an exact pattern denoised in the feature space is therefore not possible. Instead an approximate preimage is typically sought. In other words, the preimage approximates the reverse mapping of $P\phi(x)$ to the original space where x is a test point in input space. This is referred to as the preimage problem in KPCA literature and this is the focus of our research. The preimage problem is represented graphically in Figure 2.

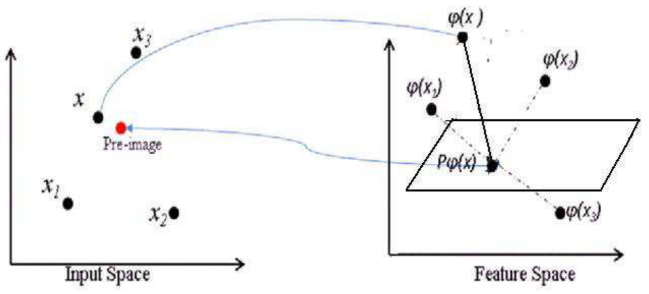


Figure 2. A test point x is transformed to feature space as $\phi(x)$ and projected to the PCA plane as $P\phi(x)$. The preimage approximates the reverse transform of $P\phi(x)$ to the original space.

Previous work by Mika *et al* [16] defined this as a nonlinear optimization problem and approached it using standard gradient descent. A drawback of using standard gradient descent methods is convergence to local minima. As a result, solutions obtained using this method are sensitive to choices of initial starting values. Kwok and Tsang [11] used the relationship between the feature space distance and the input space distance derived by Williams [26] for commonly used kernels along with multidimensional scaling (MDS) to find approximate preimages. This method was an improvement over the one suggested by Mika *et al*. Kernel regression was applied by [4] to the preimage problem where the inverse mapping from feature space to input space is posed as a regression problem. The problem of estimating a better preimage by adding penalty terms to the preimage learning process was discussed by [30]. Recently, [8] pointed out limitations inherent in the orthogonal projection operation in any KPCA algorithm , and proposed to modify it by incorporating information about the local geometry in the neighborhood of a test point in feature space so that the projection of the corresponding preimage remains closer to the full manifold. The full manifold was defined by [8] to be the set of all points in feature space that have exact preimages. Another approach suggested by [9] was to estimate the preimage of a point by doing a line search in input space along the steepest descent direction of the objective loss function evaluated at that point. The loss function is defined as the squared distance between the projected point on the principal component subspace in feature space and the preimage point mapped to the feature space.

The above approaches in literature assume that a noise-free training data set is available for learning. This is not true in many settings for instance in manufacturing variation analysis which we are working on. In our research, we propose three methodologies to address the above issues.

In Chapter 3, we apply a procedure similar to bagging [5] to improve the estimate of the preimage. The PCA plane in feature space might not be well estimated from the training data (especially with noisy training data). Instead of a single estimate of the preimage from one single training dataset, we resample the training set and apply a base KPCA algorithm to each sample. Thus, we estimate the final preimage as the average from bagged samples drawn from the original dataset to attenuate noise in kernel subspace estimation. We expect to improve the estimate from an average over several samples. We also found that the improvement is most pronounced when the parameters differ from those that minimize the error rate. Consequently, our approach improves the robustness of any base KPCA algorithm.

In Chapter 4, we propose another method to tackle the problem of handling noisy training data. The idea is that the initial estimate of the actual denoised test set obtained by a KPCA preimage estimation algorithm may not be accurate; hence, successive iterations of denoising a convex combination of the test set and the corresponding denoised set can lead us to a more accurate estimate of the actual denoised test set. We also decrease the number of top eigenvectors chosen in each iteration at a constant rate. The intuition is that we initially retain all eigenvectors so as not to loose any information about the pattern in data, and as we approach towards the final denoised preimage, we only retain the top most eigenvectors that will account for the structure in data and get rid of the noise. We also propose a simple and efficient stopping rule criteria to obtain the desirable preimage in fewer number of iterations. Our approach can easily be applied to any KPCA algorithm.

In addition to handling noise in training data, we also need to take care of the fact that there are many irrelevant features collected in the training data. Thus, we need to find the set of features relevant to the pattern in training data. In Chapter 5, we propose a feature selection procedure that augments KPCA to obtain importance estimates of the features given noisy training data. Our feature selection strategy involves projecting the data points onto sparse random vectors. 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. 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. Thus, 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.

This dissertation is arranged as follows. Chapter 1 provides the introduction. Chapter 2 provides the background behind the methodologies discussed in this dissertation. Chapter 3 discusses our approach of applying resampling (bootstrapping) technique to improve the preimage estimation. Chapter 4 provides a serial approach to estimate the preimage. Chapter 5 provides a feature selection methodology to identify the relevant subset of features. Finally, we conclude in Chapter 6.

CHAPTER 2

BACKGROUND

In this chapter, we provide the background behind the methodologies discussed in the dissertation. We also provide the relevant background in details in each of the subsequent chapters.

1 Understanding Variation Patterns

Measurement data on multiple process variables contain potential information about the sources of variation that result in complex, non-linear variation patterns. The task here is to visualize the non-linear patterns in the noisy data. PCA [10] is a widely used technique in manufacturing process literature. However, it can identify linear patterns in the data. Subsequently, we briefly discuss about PCA pointing out its limitation in handling non-linear patterns, and discuss how KPCA was developed to handle this issue. We next discuss about visualizing non-linear patterns in data using preimage in KPCA.

1.1 Brief Review of PCA

Principal component analysis (PCA) [10] finds the directions along which the projected data results in largest variance. Let $\mathbf{X}$ be a $N \times p$ data matrix with N data points and each column representing a variable. Let $\mathbf{w}_1 \in R^p$ be the column vector for projection. Assume that $\mathbf{w}_1$ is normalized to length 1 implying $\mathbf{w}_1^T \mathbf{w}_1 = 1$. The variance of the data after projecting onto $\mathbf{w}_1$ is $\mathbf{w}_1^T \mathbf{A}^T \mathbf{A} \mathbf{w}_1$. Thus, the problem reduces to finding $\mathbf{w}_1$ that maximizes this variance.

Note that the above finds the primary direction for maximizing variance. In general, PCA finds a series of direction vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d$ that maximize the variance after projection where each direction vector is orthogonal to the rest of the direction vectors. It can be shown that these direction vectors correspond to the top eigenvectors of the covariance matrix $\mathbf{A}^T\mathbf{A}$. Also the direction vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d$ form an orthonormal basis reducing the dimensionality of the original data matrix from p to d.

As can be seen previously, PCA involves projecting the data matrix onto direction vectors. Thus, it can only identify patterns in the data that correspond to linear combination of the variables. To overcome this limitation of PCA, KPCA was proposed by [22]. Essentially KPCA involves transforming the data from the original input space to a high-dimensional feature space and performing PCA in the feature space.

1.2 Brief Review of KPCA

Let $\mathbf{x}_i$, $i = 1, \ldots, N$, $\mathbf{x}_k \in R^p$ represent a set of N centered observations in the input space i.e.$\sum_{i=1}^{N} \mathbf{x}_i = 0$. To handle nonlinear structures, [22] suggested using a nonlinear map φ to transform the data from input space $\mathbf{x}$ to a higher-dimensional feature space F, $\varphi : \mathbf{R}^p \to F$. Also assume that the set of points mapped in the feature space $\varphi(\mathbf{x}_i)$ are centered. Let $\mathbf{C}$ represent the covariance matrix of $\varphi(\mathbf{x}_i)$

$$\mathbf{C} = \frac{1}{N} \sum_{i=1}^{N} \varphi(\mathbf{x}_i)\varphi(\mathbf{x}_i)' \tag{2.1}$$

A new coordinate system is obtained by the eigen decomposition of the covariance matrix $\mathbf{C}$. Here, we find the eigenvalues λ and eigenvectors $\mathbf{v} \in F$ of matrix $\mathbf{C}$, where F is the kernel feature space satisfying

$$\lambda \mathbf{v} = \mathbf{C}\mathbf{v} \tag{2.2}$$

The above equation (2.2) can be equivalently written as

$$\lambda < \varphi(\mathbf{x}_j) \cdot \mathbf{v} >=< \varphi(\mathbf{x}_j) \cdot \mathbf{C}\mathbf{v} > \tag{2.3}$$

for all $j = 1, \dots, N$. For a non-zero eigenvalue the corresponding eigenvector $\mathbf{v}$ lies in the span of $\varphi(\mathbf{x}_1), \dots, \varphi(\mathbf{x}_N)$. Thus, there exist coefficients α_i ($i = 1, \dots, N$) such that

$$\mathbf{v} = \sum_{i=1}^{N} \alpha_i \varphi(\mathbf{x}_i) \tag{2.4}$$

Substitute equation (2.4) into equation (2.3), for all $j = 1, \dots, N$, and simplify by introducing an $N \times N$ matrix $\mathbf{K}$ whose entries are given by

$$\mathbf{K}_{ij} :=< \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j) > \tag{2.5}$$

The final equation can be written in matrix form as

$$N\lambda\mathbf{K}\boldsymbol{\alpha} = \mathbf{K}^2\boldsymbol{\alpha} \tag{2.6}$$

where $\boldsymbol{\alpha}$ denotes a column vector with entries $\alpha_1, \dots, \alpha_N$. Also, because $\mathbf{K}$ is a positive, semi-definite matrix, all eigenvalues of $\mathbf{K}$ are non-negative. Let $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N \geq 0$ denote the eigenvalues and $\boldsymbol{\alpha}^1, \dots, \boldsymbol{\alpha}^N$ the corresponding set of eigenvectors of $\mathbf{K}$ in equation (2.6). We select the greatest l non-zero eigenvalues and their corresponding eigenvectors which account for most of the variance in the data. In practice, l is a tunable parameter. We also assume, without loss of generality, that each $\boldsymbol{\alpha}^k$, for $k = 1, \dots, l$ is normalized. The normalization conditions for $\boldsymbol{\alpha}^k$ are

$$\begin{aligned} 1 &= \sum_{i,j=1}^{N} \alpha_i^k \alpha_j^k < \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j) > \\ &= \sum_{i,j=1}^{N} \alpha_i^k \alpha_j^k \mathbf{K}_{ij} =< \boldsymbol{\alpha}^k \cdot \mathbf{K}\boldsymbol{\alpha}^k >= \lambda_k < \boldsymbol{\alpha}^k \cdot \boldsymbol{\alpha}^k > \end{aligned}$$

1.2.1 Using Kernels Explicitly mapping the input space points into a higher dimensional space can prove to be computationally expensive. Also, note that each element of the kernel matrix $\mathbf{K}$ is computed as the inner product of $< \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j) >$. [1] showed that inner products in the feature space can be computed from the original input space data points (known as the kernel trick). For example, let $\mathbf{x}_i$ and $\mathbf{x}_j$ represent two data points in input space, $\mathbf{x}_i, \mathbf{x}_j \in R^p$. Let $\varphi(\mathbf{x}_i)$ and $\varphi(\mathbf{x}_j)$ represent their corresponding map in the feature space. Using a kernel function k, we can obtain the inner product in the feature space by computing inner product in input space. This usually holds for feature spaces defined in terms of some positive definite kernel. One particular type of kernel function is the polynomial kernel of order s, expressed as

$$k(\mathbf{x}_i, \mathbf{x}_j) =< \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j) >= (< \mathbf{x}_i \cdot \mathbf{x}_j > +1)^s \tag{2.7}$$

This kernel function implicitly maps the input space into C_s^{p+s}. For example, using the above kernel with degree two $(s = 2)$, a two dimensional input space $(p = 2)$ will be mapped into a six dimensional feature space. The corresponding feature space map is $(1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$. This kernel was used to demonstrated the circle example earlier. Another commonly used kernel function is the Gaussian kernel of the form

$$k(\mathbf{x}_i, \mathbf{x}_j) =< \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j) >= \exp\left(\frac{-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|_F^2}{\sigma}\right) \tag{2.8}$$

where σ is a parameter related to the width of the kernel.

We have assumed that we are dealing with a set of data points that are centered in the feature space. Because we never explicitly map to the feature space, it is difficult to compute the mean of the mapped observations in the feature space. However, the kernel matrix can be modified to provide the inner product of centered mapped observations. This matrix, say $\tilde{\mathbf{K}}$ can be defined in terms of $\mathbf{K}$ as follows

$$\tilde{\mathbf{K}} = \mathbf{K - OK - KO + O\,K\,O} \qquad (2.9)$$

where $\mathbf{O}$ is a matrix with all elements $1/N$

1.3 Preimage Definition

Given a projected point on the principal component subspace in feature space, the task is to learn the point it would map back to in the input space. This is called preimage learning in KPCA literature. Let $\mathbf{x}$ be a test point in input space with a corresponding centered map $\varphi(\mathbf{x})$ in the feature space. In order to extract nonlinear principal components for the $\varphi-$image of a test point $\mathbf{x}$, we compute its projections on the k^{th} component for $k = 1, \ldots, l$ as follows

$$\beta_k = < \mathbf{v}^k \cdot \varphi(\mathbf{x}) > = \sum_{i=1}^{N} \alpha_i^k < \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}) > = \sum_{i=1}^{N} \alpha_i^k k(\mathbf{x}, \mathbf{x}_i) \qquad (2.10)$$

where the last equality follows from the definition of a kernel function.

Each β_k is the length of the projection onto the normalized eigenvector and equals the k^{th} score for data instance $\mathbf{x}$. As in linear PCA, each nonlinear score obtained using equation (2.10) represents a unique measure of variation in the data. The importance of each nonlinear score (variable) and its corresponding pattern can be measured by its associated eigenvalue which describes the amount of variation explained. Using this information, an appropriate number of variables can be used to summarize the data. Theoretically, we can compute as many scores as there are dimensions in the feature space. However, practically, this is limited to the rank of the kernel matrix $\mathbf{K}$. Similar to linear PCA, we project $\varphi(\mathbf{x})$ onto a subspace spanned by the top l eigenvectors. This projected point exists in the feature space. In order to interpret this point, it is valuable to visualize it in the original input space. This necessitates an inverse mapping from the feature space to the input space. As mentioned, such an inverse map is not always defined [16]. To illustrate this, consider an input space point (1,1). Using a polynomial kernel of degree 2, it can be mapped to the feature space as $(1, \sqrt{2}, \sqrt{2}, 1, 1, \sqrt{2})$. This feature space point can be inverse mapped into the input space point (1,1). However, consider a feature space point $(1, \sqrt{2}, \sqrt{2}, 5, 5, 7)$. There is no exact preimage for this point. Thus, we need to settle for an approximate preimage $\hat{\mathbf{x}}$, where $\varphi(\hat{\mathbf{x}}) \cong P_l\varphi(\mathbf{x})$. This is referred to as the preimage problem and is represented graphically in Figure 2.

2 Existing Methods for Finding Preimages

A brief overview of the more popular methods to obtain preimages is provided. To estimate the preimage of $P_l\varphi(\mathbf{x})$, [16] proposed minimizing the squared distance

$$\begin{aligned} \rho(\hat{\mathbf{x}}) &= \|\varphi(\hat{\mathbf{x}}) - P_l\varphi(\mathbf{x})\|^2 \\ &= \|\varphi(\hat{\mathbf{x}})\|^2 - 2P_l\varphi(\mathbf{x})'\varphi(\hat{\mathbf{x}}) + \Omega \end{aligned} \tag{2.11}$$

where Ω represents all terms independent of $\hat{\mathbf{x}}$. Equation (2.11) is minimized using standard gradient descent. An extremum can be obtained by setting the derivative of Equation (2.11) to zero. Because this method uses standard gradient descent, a drawback is that one can converge to a local minima. Hence, the preimage obtained is sensitive to starting values. Also, the iteration scheme can fail to converge in certain experiments even after choosing different starting values [11].

In another approach, [11] computed the Euclidean distance between $P_l\varphi(\mathbf{x})$ and all feature space training points $\varphi(\mathbf{x}_i)$. Then, n-nearest neighbors in the feature space are identified based on this distance metric. For commonly used kernels such as Gaussian and polynomial kernel, there exists a relationship between distance in the feature space and distance in the input space [26]. Using this relationship, corresponding input space distances between the desired preimage $\hat{\mathbf{x}}$ and the n-nearest input space points $\mathbf{x}_i s$ are computed. These input space distances are preserved when $P_l\varphi(\mathbf{x})$ is embedded back into the input space. [11] then proposed using multi-dimensional scaling (MDS) [6] as a tool to visualize the preimage (and denoted the method as KMDS). Given distances between points in a high-dimensional feature space, MDS attempts to find a lower dimensional approximation of the data so as to preserve the pairwise distances as much as possible. A new coordinate system is defined in input space by singular value decomposition of the n-nearest neighbors. MDS is then used to project $P_l\varphi(\mathbf{x})$ into this new coordinate system. Approximate preimages are found using eigenvectors of the new coordinate system.

For a $N \times p$ input matrix, the computational complexity of SVD is $O(cN^2p + c'p^3)$, where c and c' are constants. Therefore, [11] proposed using n-nearest neighbors to reduce computational time.

A penalized strategy to guide the preimage learning process was presented by [30]. The preimage is modeled by a weighted combination of the observed samples where the weights are learned by an optimization function. Under this framework, a penalized methodology is developed by integrating two types of penalties. First, a convexity constraint is imposed for learning the combination weights to generate a well-defined preimage. Second, a penalized function is used as part of the optimization to guide the preimage learning process.

Recently, [8] pointed out limitations inherent in the orthogonal projection operation in any KPCA algorithm , and proposed to modify it by incorporating information about the local geometry in the neighborhood of a test point in feature space so that the projection of the corresponding preimage remains closer to the full manifold. The full manifold was defined by [8] to be the set of all points in feature space that have exact preimages. Another approach suggested by [9] was to estimate the preimage of a point by doing a line search in input space along the steepest descent direction of the objective loss function evaluated at that point. The loss function is defined as the squared distance between the projected point on the principal component subspace in feature space and the preimage point mapped to the feature space.

3 Feature Selection in Kernel Feature Space

The task here 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 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. [2] provided a weighted feature approach 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 ([14] and [13]) 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, [14] 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.

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 [27] was proposed by [29]. This involved regularizing the weights assigned to features.

A method to measure variable importance in KPCA was proposed by [17]. 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).

We discuss the approach given by [17]. Let **a** denote the direction of maximum variance, and **b** denote the vector of feature weights that shows the importance of each feature. We assume that each entry of **b** is non-negative, and **b** is normalized to length 1. We do the following

$$\max_{\mathbf{a},\mathbf{b}} \quad \mathbf{b}^T\mathbf{P}\mathbf{b} - \lambda \|\mathbf{b}\|_1 \tag{2.12}$$

where $\mathbf{P}_{i,j} = \frac{1}{N}\mathbf{a}^T\tilde{\mathbf{K}}_i\tilde{\mathbf{K}}_j^T\mathbf{a}$, $\tilde{\mathbf{K}}$ is the centered kernel matrix. λ is a small positive constant for regularization defined by the user. Each entry of $\tilde{\mathbf{K}}$ is calculated by computing the kernel between two data points as sum of individual kernels where each individual kernel is computed on a single feature of each of the two data points. This involves optimization over parameters **a** and **b** for which [17] suggested an alternating approach (optimizing one parameter while keeping the other fixed till convergence).

The above approaches 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, our approach works 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.

CHAPTER 3

PREIMAGES FOR VARIATION PATTERNS FROM KERNEL PCA AND BAGGING

1 Introduction

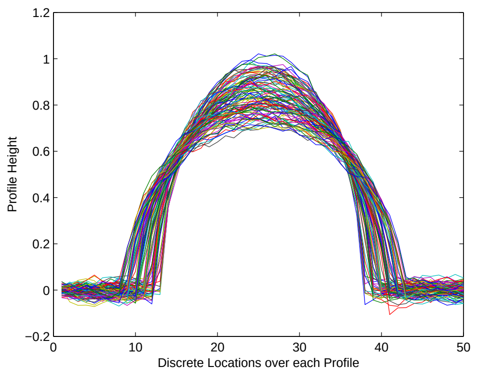
Many manufacturing organizations collect massive amounts of in-process data with the foresight of potentially using information hidden in it to identify root causes of product or process variation. Advances in measurement and data storage technologies have made it possible to track hundreds, or even thousands of dimensional characteristics with a 100% sample rate. Datasets in the form of spatial or time series, as well as images are common in modern manufacturing. Semiconductor processing, for example, involves inputs from multiple variables; each represented by a finite time series. Thus, each run is characterized by thousands of measurements. We can utilize this data to provide visual insights into the process which is crucial for engineers to make decisions.

Machine-vision systems (MVS) are widely used in many industries including medical, transportation, construction, and other industrial applications. In particular, [12] provide an in-depth survey on industrial applications which include identification of structural, surface, and operational defects. A key aspect of any MVS is acquisition and analysis of images. [15] discuss about the acquisition of grayscale or binary images for MVS. Our proposed methodology deals with analysis of such images.

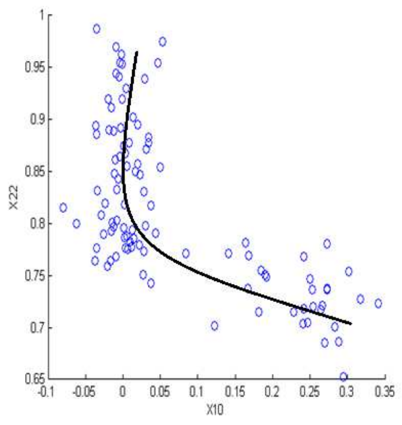
More recently, [25] show how 3D laser scanners have become popular in scanning complex manufactured part geometries. The data generated from such scanners is referred to as point cloud data. The point cloud represent a set of points measured in three-dimensional cartesian coordinate system. The point cloud data provide an accurate representation of the scanned object. Our proposed methodology is useful in visualizing the point cloud data.

In addition to image and point cloud data, profile data is also widely prevalent in manufacturing industry, especially in paper processing and autobody assemblies. The profiles are obtained as functional relationships between the response variable and independent variable(s). We present an example of the profile data in manufacturing automotive engine gaskets. One critical-to-quality feature is a bead on the gasket, the purpose of which is to create a tight seal. Figure 3(a) shows a set of profiles measured across 100 gasket beads. Each profile is obtained by scanning a stylus across the gasket bead. Each profile in Figure 1 has been discretized into 50 points evenly spaced over the horizontal axis. Hence, the measurement vector $\mathbf{x}$ for each part consists of the vertical axis profile heights at the 50 locations. We can see that the raw data collected by sensors is inherently noisy, but buried in the noise is a pronounced systematic part-to-part variation pattern, by which the gasket bead is flattening and elongating by varying amounts on each part. Here, the pattern in the data (as represented by the relationships between the different elements of $\mathbf{x}$) is non-linear. For example, consider the scatterplot between the sensor recordings at position 10 and position 22 (denoted by $x10$ and $x22$, respectively) as shown in Figure 3(b). It is clear from the plot that there are non-linear relationships between the sensor recordings (features) in the data. For simplicity, we showed the plot between two features, but the actual non-linear pattern can involve more than only two features.

The gasket example shows a pattern when a single source of variation (corresponding to bead flattening/elongation) is present in the process. In practice, multiple variation sources and their corresponding effects on process and product characteristics are embedded in the collected data. Each variation source and its interaction with other sources results in a unique pattern in the data. Identifying and segregating important variation patterns from the irrelevant ones is then a key step to identifying root causes of process variability in order to diagnose and control process or product quality problems. Noise tends to make it harder for the underlying pattern to be recognized and visualized. Therefore, it is important to remove noise from the dataset in order to visualize the true nature of the underlying patterns.



(a) Profiles



(b) Scatterplot

Figure 3. Left: profile measurements for 100 gaskets. Each gasket height is measured at 50 equally-spaced positions. Right: scatterplot between sensor recordings at positions 10 and 22 (denoted by $x10$ and $x22$, respectively)

Given a p-dimensional random vector $\mathbf{x}$, signals often tend to locate on some d-dimensional manifold in p-dimensional space ($d < p$) while noise tends to be less structured. Singular value decomposition (SVD) and principal component analysis (PCA) work on partitioning the p-dimensional space into a d-dimensional signal space and a $(p-d)$ dimensional noise space. Signals can be denoised by projecting data points onto the retained subset of principal component directions.

A drawback of PCA is in its assumption that the variation pattern is formed by linear combination of variables. PCA tends to lose its effectiveness in identifying patterns when the resultant pattern is nonlinear in nature [22]. One approach to nonlinear PCA is based on principal curves [7]. [3] used principal curves to identify and visualize nonlinear patterns in data. [22] extended the linear PCA framework to account for nonlinear structures in the data set through kernel principal component analysis (KPCA). KPCA works on the principle of mapping the data in the input space to a higher dimensional feature space via a nonlinear map $\varphi : R^p \to R^m$ where m is the number of features and the feature space is denoted as F. Linear PCA is applied to the mapped points in the feature space to extract components that are nonlinear in the original input space. Because the feature space can have a very large number of dimensions, such explicit mapping can be computationally expensive. Linear PCA in the feature space depends on the data only through inner products of the feature vectors. Schölkopf *et al* used kernel functions to compute the inner products in the feature space without carrying out the mapping explicitly as shown by [1]. Similar to linear PCA, projections onto a smaller subset of principal component directions in the feature space can be used to denoise signals.

To interpret the denoised signal, it is valuable to visualize it in the original input space. However, because the projections in feature space are on a subspace that might not correspond to the manifold of the original space, such an inverse transformation from the feature space to the input space does not typically exist as shown by [16]. Directly visualizing an exact pattern denoised in the feature space is therefore not possible. Instead an approximate preimage is typically sought. This is referred to as the preimage problem in KPCA and this is the focus of this research. Previous work by [16] defined this as a nonlinear optimization problem and approached it using standard gradient descent. A drawback of using standard gradient descent methods is convergence to local minima. As a result, solutions obtained using this method are sensitive to choices of initial starting values. [11] used the relationship between the feature space distance and the input space distance derived by [26] for commonly used kernels along with multidimensional scaling (MDS) to find approximate preimages. This method was an improvement over the one suggested by [16]. [18] extended the KPCA framework to handle noise, outlier and missing data. [30] addressed the problem of estimating a better preimage by adding penalty terms to the preimage learning process. [19] considered feature selection in kernel PCA with sparse random vectors. This approach can be applied prior to the preimage method discussed here to reduce the dimensionality of the problem.

The previous methods to denoise for KPCA assume that the training data are noise-free. In practice, many cases (such as manufacturing variation analysis) fail to meet this assumption. To improve the estimate of the preimage, we apply a procedure similar to bagging developed by [5]. Instead of a single estimate of the preimage from one single training dataset, we resample the training set and apply a base algorithm to each sample. The PCA plane in feature space might not be well estimated from the training data (especially with noisy training data). We expect to improve the estimate from an average over several samples, and we also improve the robustness of a base algorithm to parameter settings. We refer to this method as BKPCA to indicate a bagged KPCA approach. The remainder of this chapter is organized as follows. Section 2 offers a brief review of PCA and KPCA. Section 3 formally introduces the problem of finding pre-images. Section 4 reviews ex-

TABLE 1. Eigenvalues of covariance matrix in input space.

Eigenvalue	12.979	11.952
Proportion	0.521	0.479
Cumulative	0.521	1.000

isting methods for computing preimages. Section 5 introduces the proposed methodology. Section 6 provides experimental results and Section 7 provides conclusions.

2 Brief Review of KPCA

Let $\mathbf{x}_i$, $i = 1, \ldots, N$, $\mathbf{x}_k \in R^p$ represent a set of N centered observations in the input space i.e.$\sum_{i=1}^{N} \mathbf{x}_i = 0$. To handle nonlinear structures, [22] suggested linearizing the distribution by using a nonlinear map φ to transform the data from the input space $\mathbf{x}$ to a higher-dimensional feature space F, $\varphi : \mathbf{R}^p \to F$.

Consider the following example with p=2, in which $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]'$ was generated as uniformly distributed over a circle in two dimensional space, as illustrated in Figure 4. Results from linear PCA are summarized in Table 1 and Table 2. Eigenvectors provide the nature of the linear relationship between $\mathbf{x}_1$ and $\mathbf{x}_2$. Since PCA was carried out in input space, the true structure of the relationship between $\mathbf{x}_1$ and $\mathbf{x}_2$ i.e. $\mathbf{x}_1^2 + \mathbf{x}_2^2 = constant$ is not captured by the eigenvectors. Now, consider a nonlinear map, say $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)' \to \varphi(\mathbf{x}) =$

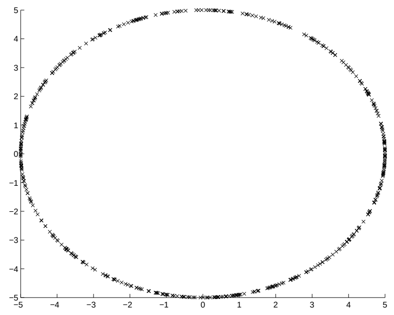


Figure 4. Scatter plot of input space variables

TABLE 2. Eigenvectors of covariance matrix in input space.

Variable	PC_1	PC_2
x_1	0.845	0.535
x_2	-0.535	0.845

TABLE 3. Eigenvalues of covariance matrix in feature space

Eigenvalue	168.14	144.79	25.95	23.50	0.000
Proportion	0.464	0.400	0.072	0.065	0.000
Cumulative	0.464	0.864	0.935	1.000	1.000

$(\mathbf{x}_1^2, \mathbf{x}_2^2, \sqrt{2}\mathbf{x}_1, \sqrt{2}\mathbf{x}_2, \sqrt{2}\mathbf{x}_1\mathbf{x}_2)$. The significance of this mapping will be discussed in a later section. Table 3 and Table 4 shows the eigenvalues and eigenvectors of the covariance matrix in the feature space respectively. Note that the eigenvector corresponding to the smallest eigenvalue correctly captures the true nature of the relationship that describes the circle. More specifically, the equation for the circle coincides with setting to zero, the variance of the linear combination of features represented by the eigenvector (i.e., constraining the linear combination to equal a constant). Refer to Table 7 for a list of symbols used throughout the chapter.

TABLE 4. Eigenvectors of covariance matrix in feature space

Variable	PC1	PC2	PC3	PC4	PC5
$\mathbf{x}_1^2$	-0.705	-0.042	-0.002	0.037	0.707
$\mathbf{x}_2^2$	0.705	0.042	0.002	-0.037	0.707
$\sqrt{2}\mathbf{x}_1$	-0.025	-0.008	-0.851	-0.525	-0.000
$\sqrt{2}\mathbf{x}_2$	-0.046	-0.001	0.526	-0.849	0.000
$\sqrt{2}\mathbf{x}_1\mathbf{x}_2$	-0.059	0.998	-0.006	-0.002	0.000

2.1 Nonlinear PCA

We refer to [22] for more details regarding the following discussion. Assume that the set of points mapped in the feature space $\varphi(\mathbf{x}_i)$ are centered. Let $\mathbf{C}$ represent the covariance matrix of $\varphi(\mathbf{x}_i)$

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}\varphi(\mathbf{x}_i)\varphi(\mathbf{x}_i)' \tag{3.1}$$

As in linear PCA, a new coordinate system is obtained by the eigen decomposition of the covariance matrix $\mathbf{C}$. Here, we find the eigenvalues λ and eigenvectors $\mathbf{v} \in F$ of matrix $\mathbf{C}$, where F is the kernel feature space satisfying

$$\lambda\mathbf{v} = \mathbf{C}\mathbf{v} \tag{3.2}$$

The above equation (5.2) can be equivalently written as

$$\lambda < \varphi(\mathbf{x}_j)\cdot\mathbf{v} > = < \varphi(\mathbf{x}_j)\cdot\mathbf{C}\mathbf{v} > \tag{3.3}$$

for all $j = 1,\ldots,N$. For a non-zero eigenvalue the corresponding eigenvector $\mathbf{v}$ is

$$\mathbf{v} = \sum_{i=1}^{N}\alpha_i\varphi(\mathbf{x}_i) \tag{3.4}$$

Substitute equation (4.3) into equation (3.3), for all $j = 1,\ldots,N$, and simplify by introducing an $N\times N$ matrix $\mathbf{K}$ whose entries are given by

$$\mathbf{K}_{ij} := < \varphi(\mathbf{x}_i)\cdot\varphi(\mathbf{x}_j) > \tag{3.5}$$

The final equation can be written in matrix form as

$$N\lambda\mathbf{K}\boldsymbol{\alpha} = \mathbf{K}^2\boldsymbol{\alpha} \tag{3.6}$$

where $\boldsymbol{\alpha}$ denotes a column vector with entries $\alpha_1,\ldots,\alpha_N$. Also, because $\mathbf{K}$ is a positive, semi-definite matrix, all eigenvalues of $\mathbf{K}$ are non-negative. Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N \geq 0$ denote the eigenvalues and $\boldsymbol{\alpha}^1,\ldots,\boldsymbol{\alpha}^N$ the corresponding set of eigenvectors of $\mathbf{K}$ in equation

(3.6). We select the greatest l non-zero eigenvalues and their corresponding eigenvectors which account for most of the variance in the data. In practice, l is a tunable parameter. We also assume, without loss of generality, that each $\boldsymbol{\alpha}^k$, for $k = 1,\ldots,l$ is normalized. The normalization conditions for $\boldsymbol{\alpha}^k$ are

$$\begin{aligned} 1 &= \sum_{i,j=1}^{N} \alpha_i^k \alpha_j^k < \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j) > \\ &= \sum_{i,j=1}^{N} \alpha_i^k \alpha_j^k \mathbf{K}_{ij} =< \boldsymbol{\alpha}^k \cdot \mathbf{K}\boldsymbol{\alpha}^k >= \lambda_k < \boldsymbol{\alpha}^k \cdot \boldsymbol{\alpha}^k > \end{aligned}$$

2.2 Using Kernels

Explicitly mapping the input space points into a higher dimensional space can prove to be computationally expensive. Also, note that each element of the kernel matrix $\mathbf{K}$ is computed as the inner product of $< \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j) >$. [1] showed that inner products in the feature space can be computed from the original input space data points (known as the kernel trick). For example, let $\mathbf{x}_i$ and $\mathbf{x}_j$ represent two data points in input space, $\mathbf{x}_i, \mathbf{x}_j \in R^p$. Let $\varphi(\mathbf{x}_i)$ and $\varphi(\mathbf{x}_j)$ represent their corresponding map in the feature space. Using a kernel function k, we can obtain the inner product in the feature space by computing inner product in input space. This usually holds for feature spaces defined in terms of some positive definite kernel. One particular type of kernel function is the polynomial kernel of order s, expressed as

$$k(\mathbf{x}_i, \mathbf{x}_j) =< \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j) >= (< \mathbf{x}_i \cdot \mathbf{x}_j > +1)^s \tag{3.7}$$

This kernel function implicitly maps the input space into C_s^{p+s}. For example, using the above kernel with degree two $(s = 2)$, a two dimensional input space $(p = 2)$ will be mapped into a six dimensional feature space. The corresponding feature space map is $(1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$. This kernel was used to demonstrated the circle example earlier. Another commonly used kernel function is the Gaussian kernel of the form

$$k(\mathbf{x}_i, \mathbf{x}_j) =< \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j) >= \exp\left(\frac{-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|_F^2}{\sigma}\right) \tag{3.8}$$

where σ is a parameter related to the width of the kernel.

We have assumed that we are dealing with a set of data points that are centered in the feature space. Because we never explicitly map to the feature space, it is difficult to compute the mean of the mapped observations in the feature space. However, the kernel matrix can be modified to provide the inner product of centered mapped observations. This matrix, say $\tilde{\mathbf{K}}$ can be defined in terms of $\mathbf{K}$ as follows

$$\tilde{\mathbf{K}} = \mathbf{K - OK - KO + O\,K\,O} \tag{3.9}$$

where $\mathbf{O}$ is a matrix with all elements $1/N$

3 Problem Definition

Let $\mathbf{x}$ be a test point in input space with a corresponding centered map $\varphi(\mathbf{x})$ in the feature space. In order to extract nonlinear principal components for the $\varphi-$image of a test point $\mathbf{x}$, we compute its projections on the k^{th} component for $k = 1, \ldots, l$ as follows

$$\beta_k =< \mathbf{v}^k \cdot \varphi(\mathbf{x}) >= \sum_{i=1}^{N} \alpha_i^k < \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}) >= \sum_{i=1}^{N} \alpha_i^k k(\mathbf{x}, \mathbf{x}_i) \tag{3.10}$$

where the last equality follows from the definition of a kernel function.

Each β_k is the length of the projection onto the normalized eigenvector and equals the k^{th} score for data instance $\mathbf{x}$. As in linear PCA, each nonlinear score obtained using equation (4.5) represents a unique measure of variation in the data. The importance of each nonlinear score (variable) and its corresponding pattern can be measured by its associated eigenvalue which describes the amount of variation explained. Using this information, an appropriate number of variables can be used to summarize the data. Theoretically, we can

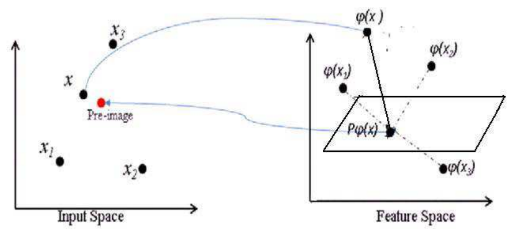


Figure 5. A test point x is transformed to feature space as $\phi(x)$ and projected to the PCA plane as $P\phi(x)$. The preimage approximates the reverse transform of $P\phi(x)$ to the original space.

compute as many scores as there are dimensions in the feature space. However, practically, this is limited to the rank of the kernel matrix **K**. Similar to linear PCA, a denoised image can be obtained by projecting $\varphi(\mathbf{x})$ onto a subspace spanned by the top l eigenvectors

$$P_l\varphi(\mathbf{x}) = \sum_{k=1}^{l} \beta_k \mathbf{v}^k \tag{3.11}$$

This denoised image exists in the feature space. In order to interpret this image, it is valuable to visualize it in the original input space. This necessitates an inverse mapping from the feature space to the input space. As mentioned, such an inverse map is not always defined [16]. To illustrate this, consider an input space point (1,1). Using a polynomial kernel of degree 2, it can be mapped to the feature space as $(1,\sqrt{2},\sqrt{2},1,1,\sqrt{2})$. This feature space point can be inverse mapped into the input space point (1,1). However, consider a feature space point $(1,\sqrt{2},\sqrt{2},5,5,7)$. There is no exact preimage for this point. Thus, we need to settle for an approximate preimage $\hat{\mathbf{x}}$, where $\varphi(\hat{\mathbf{x}}) \cong P_l\varphi(\mathbf{x})$. This is referred to as the preimage problem and is represented graphically in Figure 5.

4 Existing Methods for Finding Preimages

A brief overview of the more popular methods to obtain preimages is provided. [16] proposed to estimate the preimage of $P_l\varphi(\mathbf{x})$ by minimizing the squared distance

$$\begin{aligned}\rho(\hat{\mathbf{x}}) &= \|\varphi(\hat{\mathbf{x}}) - P_l\varphi(\mathbf{x})\|^2 \\ &= \|\varphi(\hat{\mathbf{x}})\|^2 - 2P_l\varphi(\mathbf{x})'\varphi(\hat{\mathbf{x}}) + \Omega \qquad (3.12)\end{aligned}$$

where Ω represents all terms independent of $\hat{\mathbf{x}}$. Equation (3.12) is minimized using standard gradient descent. An extremum can be obtained by setting the derivative of Equation (3.12) to zero. Because this method uses standard gradient descent, a drawback is that one can converge to a local minima. Hence, the preimage obtained is sensitive to starting values. Also, the iteration scheme can fail to converge in certain experiments even after choosing different starting values [11].

In another approach, [11] computed the Euclidean distance between $P_l\varphi(\mathbf{x})$ and all feature space training points $\varphi(\mathbf{x}_i)$. Then, n-nearest neighbors in the feature space are identified based on this distance metric. For commonly used kernels such as Gaussian and polynomial kernel, there exists a relationship between distance in the feature space and distance in the input space [26]. Using this relationship, corresponding input space distances between the desired preimage $\hat{\mathbf{x}}$ and the n-nearest input space points $\mathbf{x}_i s$ are computed. These input space distances are preserved when $P_l\varphi(\mathbf{x})$ is embedded back into the input space. [11] then proposed using multi-dimensional scaling (MDS) [6] as a tool to visualize the preimage (and denoted the method as KMDS). Given distances between points in a high-dimensional feature space, MDS attempts to find a lower dimensional approximation of the data so as to preserve the pairwise distances as much as possible. A new coordinate system is defined in input space by singular value decomposition of the n-nearest neighbors. MDS is then used to project $P_l\varphi(\mathbf{x})$ into this new coordinate system. Approximate preimages are found using eigenvectors of the new coordinate system.

For a $N \times p$ input matrix, the computational complexity of SVD is $O(cN^2p + c'p^3)$, where c and c' are constants. Therefore, [11] proposed using n-nearest neighbors to reduce computational time.

More recently [30] presented a penalized strategy to guide the preimage learning process. The preimage is modeled by a weighted combination of the observed samples where the weights are learned by an optimization function. Under this framework, a penalized methodology is developed by integrating two types of penalties. First, a convexity constraint is imposed for learning the combination weights to generate a well-defined preimage. Second, a penalized function is used as part of the optimization to guide the preimage learning process. An issue with this approach is that the observed samples (training set) should be noise-free. In case of noisy training set, the preimage obtained from this model is inherently noisy.

5 Preimages from Bagging

As stated in the previous section, most of the methods assume that the training data are noise-free. Some applications meet this assumption, but in practice many other cases (such as manufacturing variation analysis) do not. Our objective is to improve upon the previous approaches. We improve the estimate of the preimage through a procedure similar to bagging [5]. Instead of a single estimate of the preimage from one single training dataset, resample the training set B times with replacement, with each sample size equal to that of the original data. Use each bootstrap sample to complete any of the previous methods and obtain an estimated preimage for a test point. Finally, estimate the final preimage of each test point as the average of the obtained B points. Let $\hat{\mathbf{X}}(b)$ denote the preimage obtained from the b^{th} sample. The final estimated preimage $\hat{\mathbf{X}}$ is given by

$$\hat{\mathbf{X}} = \sum_{b=1}^{B} \frac{\hat{\mathbf{X}}(b)}{B} \tag{3.13}$$

We refer to this method as BKPCA to indicate a bagged KPCA approach. In bagging one averages over several models. The intuition behind our approach is that the PCA plane in feature space might not be well estimated from the training data; thus, some improvement might be expected from an average over several estimates. We provide subsequent experiments to illustrate these comments. Moreover, this also makes our method more robust to noisy instances in the training data set. The detailed steps in our method can be summarized in the following algorithm

```
1: Given a training data set $D_0$ and a test data set $D_{test}$. Fix values for the kernel parameter
   (such as $\sigma$ in a Gaussian kernel or the degree $s$ in a polynomial kernel), number of
   bootstrap samples $B$, other parameters as defined for each method by the corresponding
   authors. The objective is to estimate the denoised test data set.
2: for each test data point $i$ in $D_{test}$ do
3:     for $b = 1$ to $B$ do
4:         Select a bootstrap sample $D_b$ from $D_0$ with replacement.
5:         Generate the $l$ eigenvectors in kernel feature space from $D_b$ (where we use a Gaus-
           sian kernel with kernel parameter $\sigma$)
6:         Transform $i^{\text{th}}$ point in $D_{test}$ to feature space and project it onto the subspace
           spanned by chosen $l$ eigenvectors.
7:         Choose from $D_b$, $n$ nearest points in feature space to the projected $i^{\text{th}}$ point in
           $D_{test}$.
```

```
8:          Estimate the denoised data point for the $i^{\text{th}}$ test point from its $n$ nearest neighbors
            in $D_b$.
9:      end for
10:     Estimate the final denoised point for the $i^{\text{th}}$ test point as the average across $B$ de-
        noised points obtained above.
11: end for
```

Note that if we have no distinct training and test sets, but instead a single set that we want to denoise, then the algorithm still applies directly with D_0 the same as D_{test}.

Because the training data is noisy, a single KPCA subspace fit from the full training data may not provide a reliable estimate. In order to evaluate the stability of a subspace estimated from noisy training data, we consider an appropriate metric called subspace distance that was developed by [24] to measure difference between two subspaces. Specifically, the subspace distance d_{SU} between two subspaces is calculated as

$$d_{SU} = \sqrt{l - \sum_{i=1}^{l} \sum_{j=1}^{l} (\mathbf{u}_i{}'\mathbf{v}_j)^2} \tag{3.14}$$

where $\mathbf{u}_i$'s and $\mathbf{v}_j$'s are each a set of orthonormal bases (eigenvectors) spanning the two subspaces, respectively, and l is the number of leading eigenvectors chosen. [24] considered two subspaces to be similar if $d_{SU} < \sqrt{l}/2$.

We show in our experiments that a single subspace estimated from the noisy training data can be unstable. Since the preimage is learned from the subspace, we expect variablity in the preimage. One approach to reduce this variability is to average across preimages learnt from different subspaces.

In practice, however, we are provided with a single realization of the noisy training data. Therefore, we draw bootstrap samples to obtain multiple realizations of the training data. As shown in the experiments, each bootstrap sample of the training data set results in a different kernel principal component subspace in the feature space, and each subspace results in a different preimage learned in the input space. Essentially, bootstrapping tries to obtain a representation of the true distribution of the preimages in input space by drawing several samples of the training set. Since the training set is noisy, we expect variability in the preimage learned from the full training data. Thus, by averaging across all preimages learned from different bootstrap samples, we try to smooth out variations from different preimages. We visually demonstrate this in Figure 6. Also using a bootstrap sample tends to down-weight the influence of noisy instances in determining the kernel principal component subspace thereby improving the robustness of a base KPCA algorithm.

The performance of our method against others is evaluated using the Euclidean distance metric. The preimage residual root sum of squared error (RSS) for all the methods was estimated by using the Euclidean distance between the obtained preimage $\hat{\mathbf{x}}$ and its true image $\mathbf{t}$ by

$$RSS = \sqrt{\sum_{i=1}^{N} (\hat{\mathbf{x}}_i - \mathbf{t}_i)^2}. \tag{3.15}$$

In our experiments, the true image is known beforehand to which we add noise to generate the noisy test images. The true image is only used for comparison purpose and not used in any of the calculations in our algorithm.

6 Experimental Results

Images provide high-dimensional inputs and are useful to visualize the success of de-noising. Consequently, we experiment on databases with images of handwritten digits and faces. We currently include image examples that, even though are not from manufacturing, are similar to manufacturing image data. We consider two scenarios: one when we have training data which is noise-free (called the low-noise case), and a second without a noise-free data set for training which results in learning on the noisy data set (called the high-noise case). In both the cases, we have to denoise the given test data based on our learning on the training data. In practice, we usually encounter the high-noise case, and our motivation comes from this fact.

There are several parameters involved in different preimage estimation algorithms for KPCA. The parameters common to all algorithms are σ (Gaussian kernel parameter) and l (the number of leading eigenvalues). Next there are some parameters specific to the given algorithm such as the number of nearest neighbors n in KMDS algorithm and the penalty parameter λ in case of penalized learning algorithm with ridge penalty. In addition to these parameters in any base algorithm, our BKPCA algorithm involves an additional parameter B which is the number of bootstrap samples.

According to [16], σ is set to rp times the average component variance where p is the dimensionality of input space, and r is a constant whose value is usually set to two. We used the value of σ suggested by [16] for our experiments. Additionally, we conducted some experiments shown in Figures 7 through 14 where we chose different values of σ to see how RSS changes. Finally we didn't find substantial difference in results while using the values of σ suggested by [16] and one suggested by [11].

[30] recommended that the value of l be chosen to preserve 95% of the energy of the training data. For Figures 19 through 22, we also experimented with several values of the % energy to be preserved (chosen from $\{$ 70, 80, 90, 95$\}$).

We now discuss about some of the parameters specific to a given preimage estimation algorithm. According to [11], the number of nearest neighbors n is set to 10. [30] discuss about choosing the parameters for different penalty functions depending on the application involved. Based on their results, we chose the ridge penalty with $\lambda = 0.001$.

For our BKPCA procedure, we have an additional parameter B. For our experiments in Figures 19 through 22, we chose $B \in \{50, 100, 200, 500\}$.

We carry out our experiments to study the behavior of the algorithms extensively. We compared our BKPCA meta-method with different base algorithms proposed in literature.

6.1 Subspace Stability Evaluation

The USPS dataset at http://yann.lecun.com/exdb/mnist/ consists of 16×16 gray scale images of zip code digits (0-9), automatically scanned from envelopes by the U.S.Postal Service. Example images are seen in the first row of Figure 29.

First we conducted the following experiments to show the instability of KPCA subspace learned from noisy training data. We chose digits 7, 5, and 4 (from the USPS digits dataset) for the following experiments. We generated 100 pairs of noisy data samples for each digit where each noisy data sample was obtained by adding independent Gaussian noise with mean $\mu = 0$ and $\sigma_G = 1$ to the original data set. The kernel parameter σ was set to $2p$ times the variance of data where p is the dimensionality of input space. We chose $l \in \{50, 100\}$. When $l = 50$, we calculated the subspace distance for each pair for different digits, and found the average subspace distance to be 6.33 (maximum value was 6.36, minimum value was 6.29, and standard deviation was 0.012) which is higher than 3.53 (= $\sqrt{50}/2$). Similarly when $l = 100$, the average subspace distance was found to be 8.74 (maximum value was 8.77, minimum value was 8.71, and standard deviation was 0.011) which is greater than 5 (= $\sqrt{100}/2$) . Since the distance between kernel principal component subspaces is large, they are not similar.

In order to observe the difference in KPCA subspace learned from different bootstrap samples, we used the parameter settings from the previous experiments except we added independent Gaussian noise with mean $\mu = 0$ and $\sigma_G = 1$ to each digit data set, and generated 100 different pairs of bootstrap samples from a data set. We compared the subspace distance between different KPCA subspaces learned from different bootstrap samples. When $l = 50$, we calculated the subspace distance for each pair for different digits, and found the average subspace distance to be 5.57 (maximum value was 6.13, minimum value was 5.19, and standard deviation was 0.15) which is higher than 3.53. Similarly when $l = 100$, the average subspace distance was found to be 7.02 (maximum value was 7.41, minimum value was 6.68, and standard deviation was 0.14) which is greater than 5. Thus, similar to results from the experiments on noisy data sets, we see that the KPCA subspaces are different.

Each subspace is expected to result in a different preimage learned in input space. To visually illustrate the variations in preimages learned from different bootstrap samples, we show the results for the gasket data in Figure 6. Each profile has been discretized into 200 points, and the preimages for a profile learned from each of the $B = 50$ bootstrap samples are shown. Figure 6 shows that averaging over preimages learned from several bootstrap samples of training data can reduce the variability of the final preimage.

6.2 Digit Images Denoised

We now demonstrate our approach by denoising each of the the digits. For the low-noise case, we randomly chose 300 images for training. Another set of mutually exclusive 300 images was taken as true data and Gaussian noise with mean $\mu = 0$ and standard deviation σ_G with values of $\sigma_G = 0.5$ and $\sigma_G = 1$ were added to the true data to produce the noisy test data which are to be subsequently denoised. For the high-noise case, for all digits we

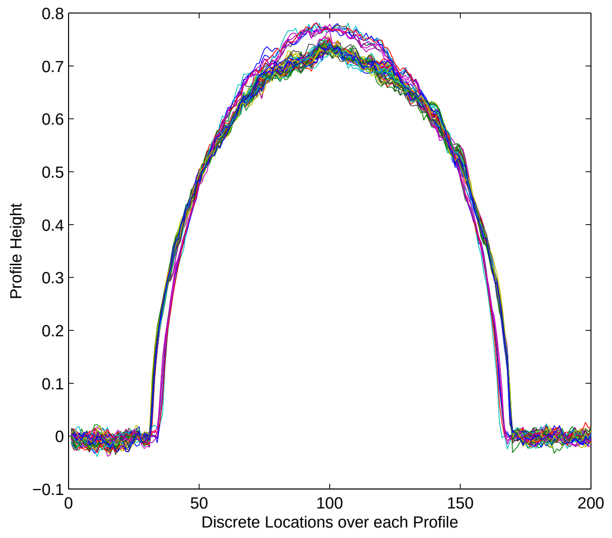
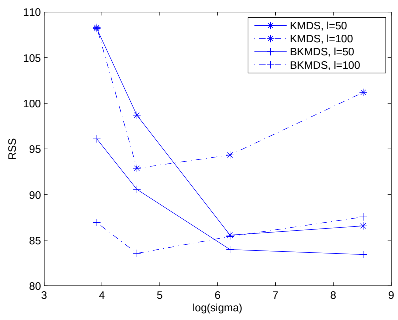


Figure 6. Plot showing variation in preimages for a profile learned from each of the $B = 50$ bootstrap samples from the gasket data. Each profile has been discretized into 200 points.

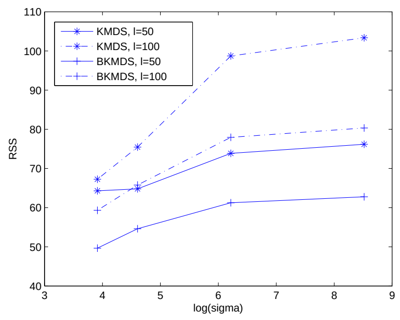
randomly chose 300 images as true data and added Gaussian noise $\mu = 0$ and σ_G with values of 0.5 and 1 to the true data to produce the noisy test data which are to be subsequently denoised. Note that the noisy test set itself is the training data here.

First consider the KMDS algorithm. When we apply BKPCA to the base KMDS algorithm we denote the procedure as BKMDS. We consider the RSS values for different parameter settings for both KMDS and BKMDS for the high-noise case. Figure 7 and Figure 8 show how the RSS varies for each algorithm for different parameter settings of σ and l with $n = 10$ and with Gaussian noise $\mu = 0$ and $\sigma_G = 0.5$. The horizontal axis scale is $\log(\sigma)$ and the piece-wise linear curves illustrate KMDS and BKMDS for either $l = 50$ or 100. Digits 0 through 4 are shown in Figure 7, and digits 5 through 9 are shown in Figure 8.

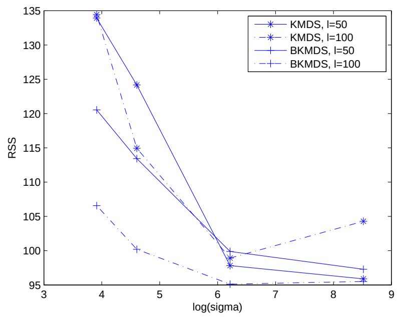
To further explore parameter settings in the high-noise case, Figures 9 through 11 consider digits 7 and 9 only. Figure 9 shows how the RSS varies for $n = 10$ with Gaussian



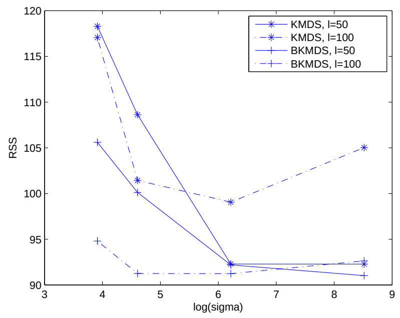
(a) Digit 0



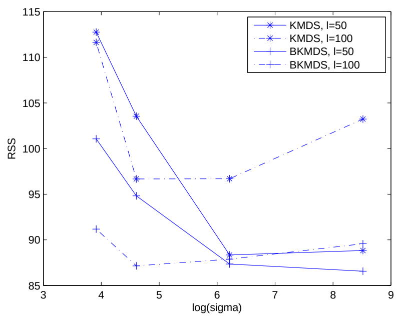
(b) Digit 1



(c) Digit 2



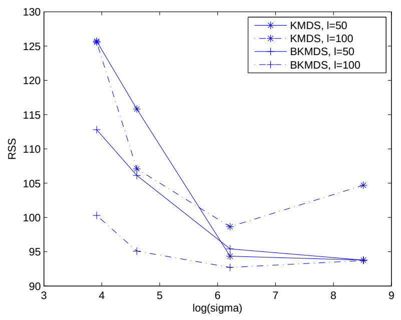
(d) Digit 3



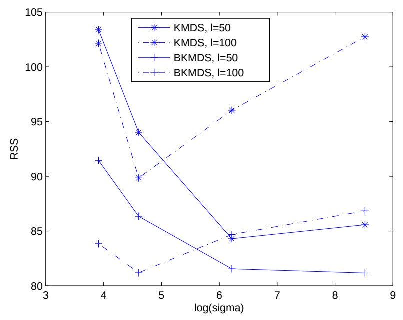
(e) Digit 4

Figure 7. RSS values of KMDS versus bagging (denoted as BKMDS) for digits 0-4 for the high-noise case with the number of leading eigenvectors $l \in \{50, 100\}$, 10 nearest neighbors and noise $\sigma_G = 0.5$. The RSS is shown for different parameter settings of σ where the horizontal axis scale is $\log(\sigma)$.

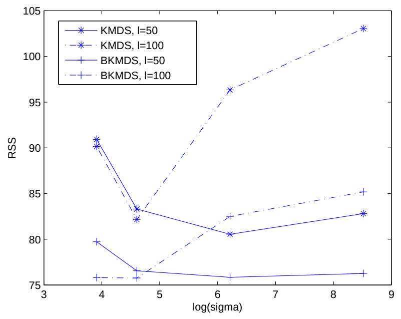
noise $\mu = 0$ and $\sigma_G = 1$. Figure 10 shows how the RSS varies for $n = 25$ with Gaussian noise $\mu = 0$ and $\sigma_G = 0.5$. Figure 11 shows how the RSS varies for $n = 25$ with Gaussian noise $\mu = 0$ and $\sigma_G = 1$.



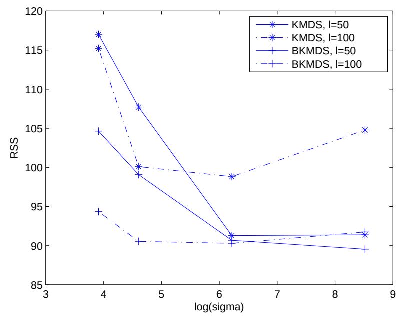
(a) Digit 5



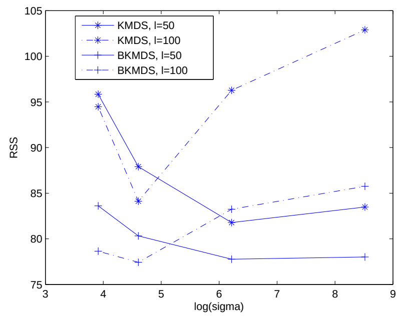
(b) Digit 6



(c) Digit 7



(d) Digit 8



(e) Digit 9

Figure 8. RSS values of KMDS versus bagging (denoted as BKMDS) for digits 5-9 for the high-noise case with the number of leading eigenvectors $l \in \{50, 100\}$, 10 nearest neighbors and noise $\sigma_G = 0.5$. The RSS is shown for different parameter settings of σ where the horizontal axis scale is $\log(\sigma)$.

We clearly see from the figures that BKPCA improves the KMDS algorithm for the chosen parameter values for the high-noise case. The improvement is most pronounced when the parameters differ from those that minimize the RSS. Importantly, BKPCA improves the robustness of the base algorithm.

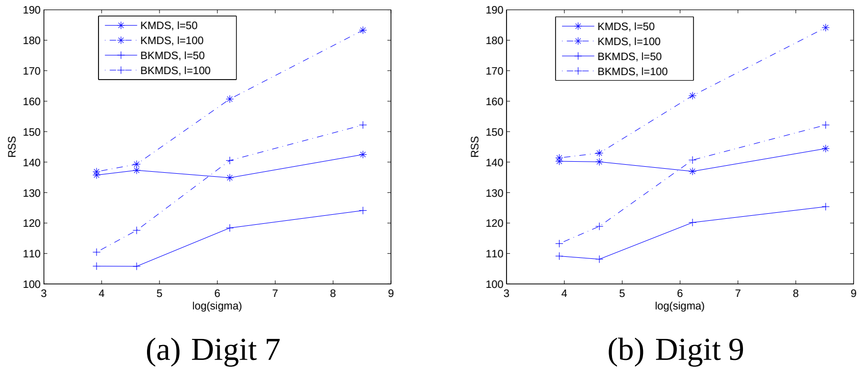


Figure 9. RSS values of KMDS versus bagging (denoted as BKMDS) for digits 7 and 9 for the high-noise case with the number of leading eigenvectors $l \in \{50, 100\}$, 10 nearest neighbors, noise $\sigma_G = 1$. The RSS is shown for different parameter settings of σ where the horizontal axis scale is $\log(\sigma)$.

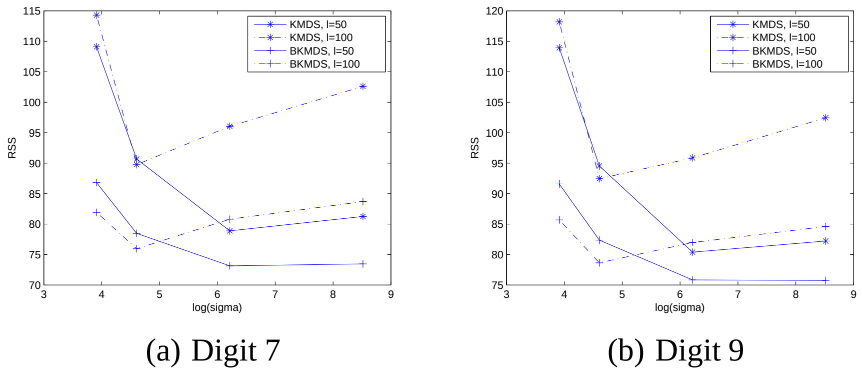
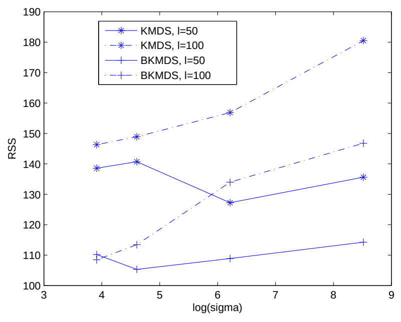


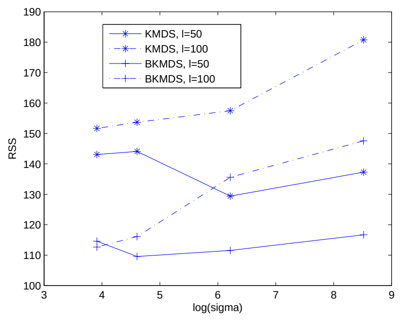
Figure 10. RSS values of KMDS versus bagging (denoted as BKMDS) for digits 7 and 9 for the high-noise case with the number of leading eigenvectors $l \in \{50, 100\}$, 25 nearest neighbors, noise $\sigma_G = 0.5$. The RSS is shown for different parameter settings of σ where the horizontal axis scale is $\log(\sigma)$.

For the low-noise case, we also report the RSS values for digits 7 and 9 for different parameter and noise settings. Figure 12 shows how the RSS varies for $n = 25$ with Gaussian noise $\mu = 0$ and $\sigma_G = 0.5$. Figure 13 and Figure 14 show the RSS varies for $n = 10, 25$, respectively, with Gaussian noise $\mu = 0$ and $\sigma_G = 1$.

We see from the figures that both KMDS and BKPCA perform comparably in the low-noise case, thus, confirming the fact that BKPCA performs at least as good as KMDS in the

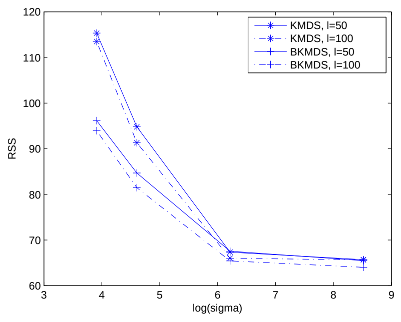


(a) Digit 7

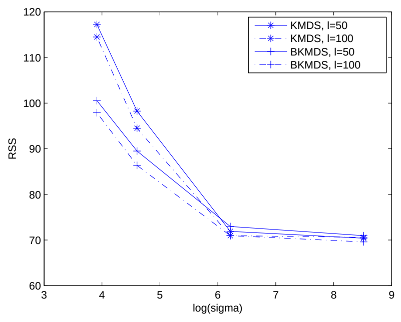


(b) Digit 9

Figure 11. RSS values of KMDS versus bagging (denoted as BKMDS) for digits 7 and 9 for the high-noise case with the number of leading eigenvectors $l \in \{50, 100\}$, 25 nearest neighbors, noise $\sigma_G = 1$. The RSS is shown for different parameter settings of σ where the horizontal axis scale is $\log(\sigma)$.



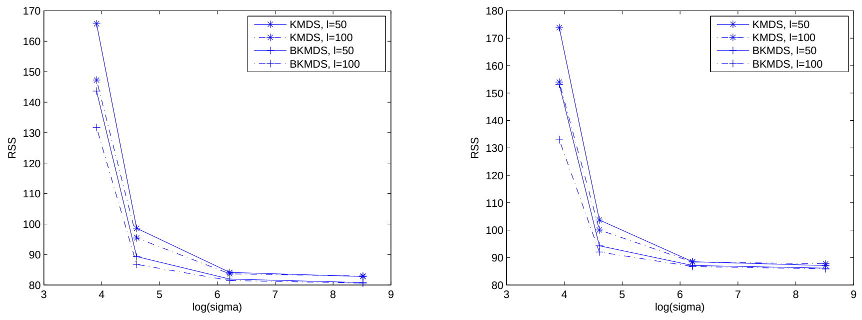
(a) Digit 7



(b) Digit 9

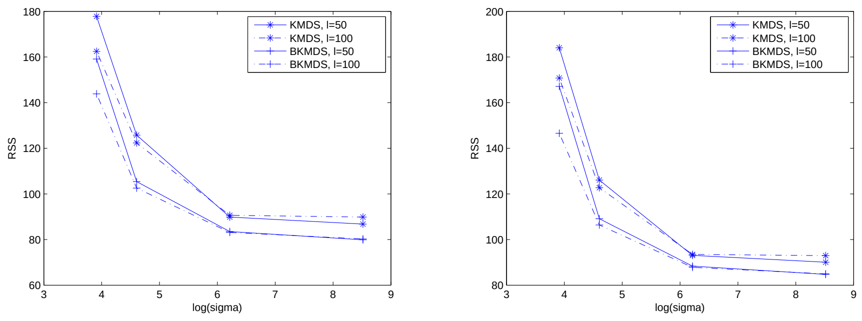
Figure 12. RSS values of KMDS versus bagging (denoted as BKMDS) for digits 7 and 9 for the low-noise case with the number of leading eigenvectors $l \in \{50, 100\}$, 25 nearest neighbors, noise $\sigma_G = 0.5$. The RSS is shown for different parameter settings of σ where the horizontal axis scale is $\log(\sigma)$.

best possible scenario when training data are noise-free. However, for applications where only noisy training data are available, high-noise experiments illustrate that BKPCA can substantially improve upon KMDS.



(a) Digit 7 (b) Digit 9

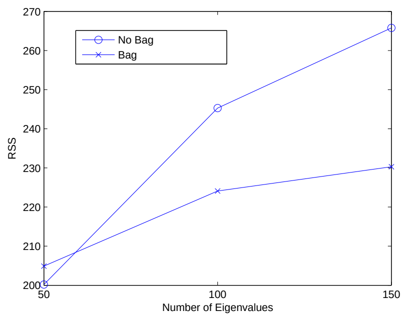
Figure 13. RSS values of KMDS versus bagging (denoted as BKMDS) for digits 7 and 9 for the low-noise case with the number of leading eigenvectors $l \in \{50, 100\}$, 10 nearest neighbors, noise $\sigma_G = 1$. The RSS is shown for different parameter settings of σ where the horizontal axis scale is $\log(\sigma)$.



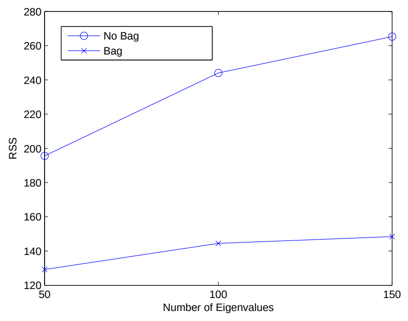
(a) Digit 7 (b) Digit 9

Figure 14. RSS values of KMDS versus bagging (denoted as BKMDS) for digits 7 and 9 for the low-noise case with the number of leading eigenvectors $l \in \{50, 100\}$, 25 nearest neighbors, noise $\sigma_G = 1$. The RSS is shown for different parameter settings of σ where the horizontal axis scale is $\log(\sigma)$.

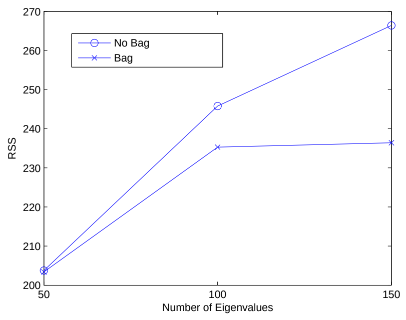
We also applied our algorithm on the gradient descent method [16] and on the penalized preimage approach [30] for the digits data in the high-noise case. Figures 15 and 16 show the results for the gradient descent approach. Figures 17 and 18 show the results for the penalized preimage approach approach. In both the cases, BKPCA reduces the RSS.



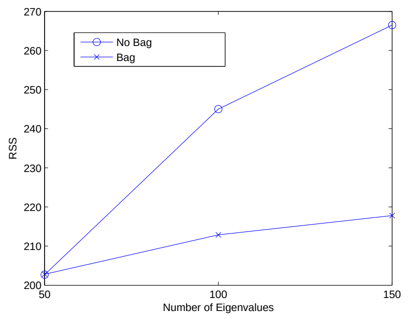
(a) Digit 0



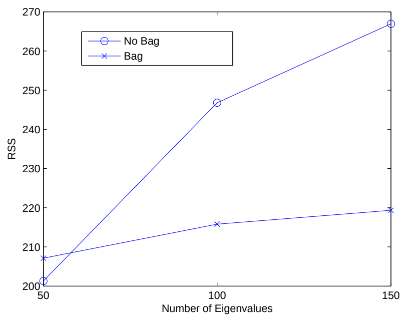
(b) Digit 1



(c) Digit 2

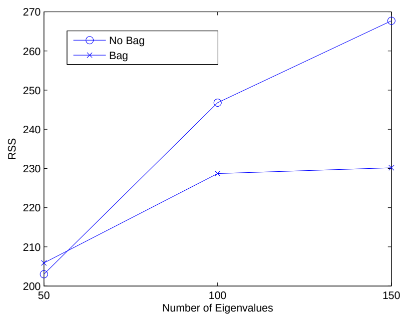


(d) Digit 3

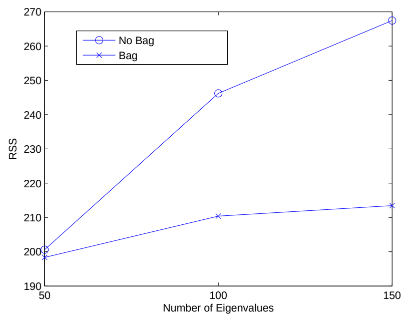


(e) Digit 4

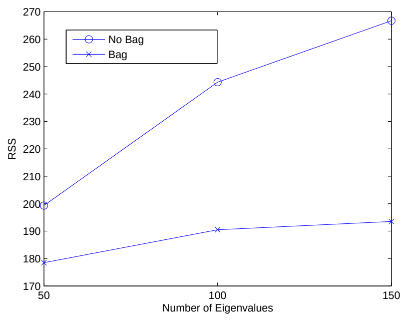
Figure 15. RSS values of the gradient descent approach versus bagged approach for digits 0-4 for the high-noise case with noise $\sigma_G = 1$.



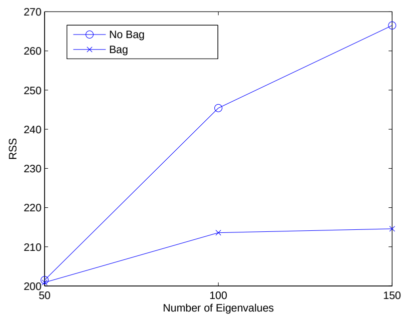
(a) Digit 5



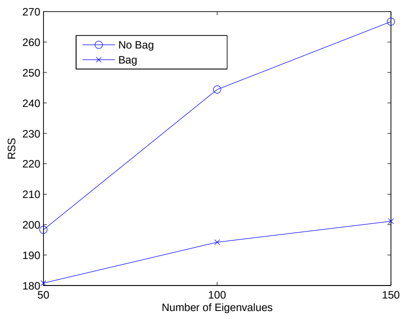
(b) Digit 6



(c) Digit 7



(d) Digit 8



(e) Digit 9

Figure 16. RSS values of the gradient descent approach versus bagged approach for digits 5-9 for the high-noise case with noise $\sigma_G = 1$.

We also experimented with several values of the % energy to be preserved (chosen from $\{$ 70, 80, 90, 95$\}$). We chose to experiment on digits 7 and 9 (300 instances chosen from each digit). The noisy digit images were generated by adding independent Gaussian noise $\mu = 0$ and $\sigma_G = 1$. For each of the values of % energy preserved, we compare the penalized preimage approach by [30] to our BKPCA approach with values of $B \in \{50, 100, 200, 500\}$.

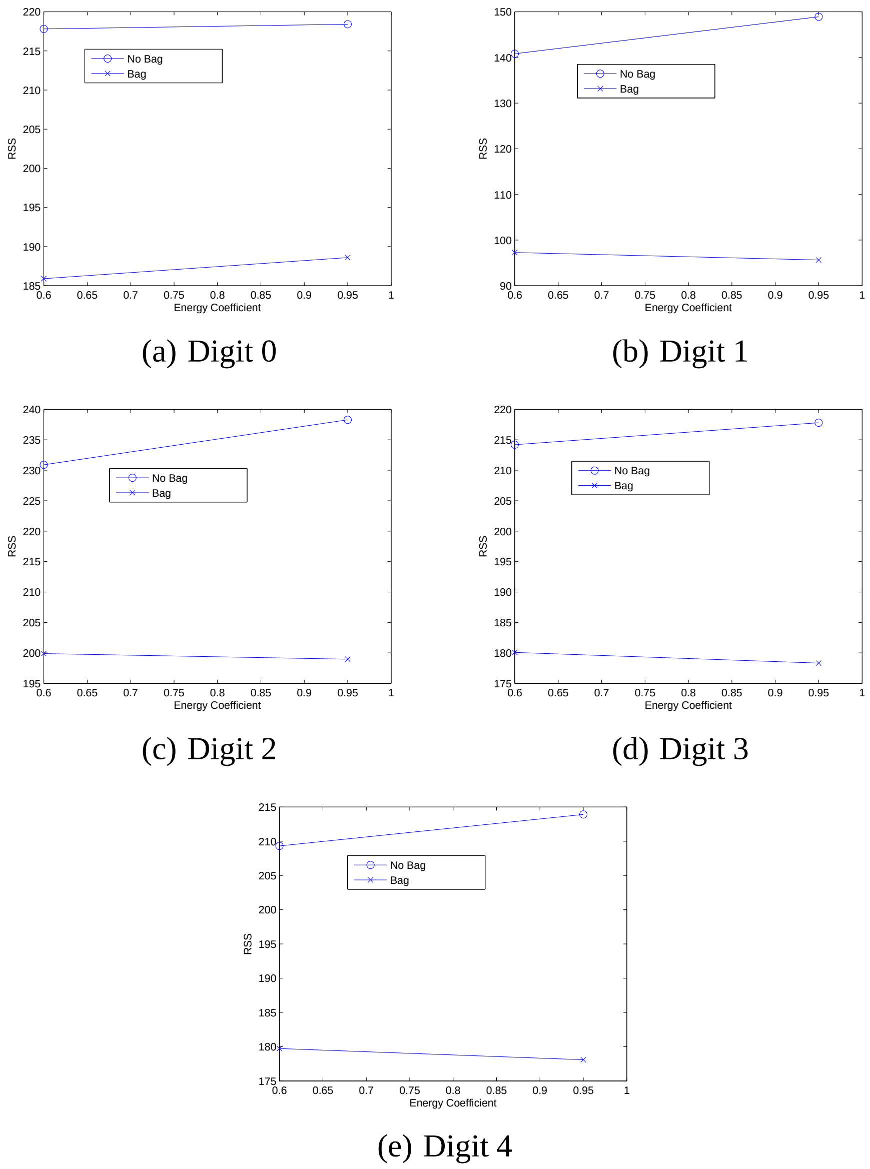
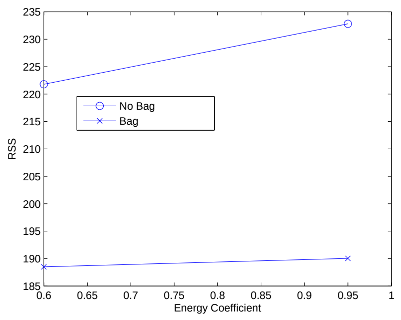

(a) Digit 0 (b) Digit 1 (c) Digit 2 (d) Digit 3 (e) Digit 4

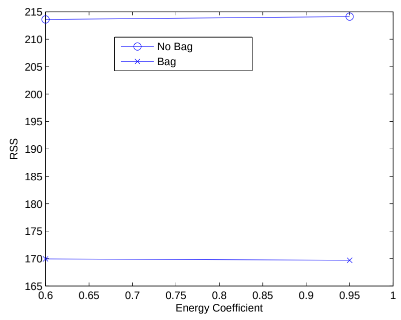
Figure 17. RSS values of penalized preimage approach versus bagged approach for digits 0-4 for the high-noise case with noise $\sigma_G = 1$.

The results are shown in Figures 19 through 20. We see that our BKPCA method performs significantly better than the penalized preimage approach for each value of B.

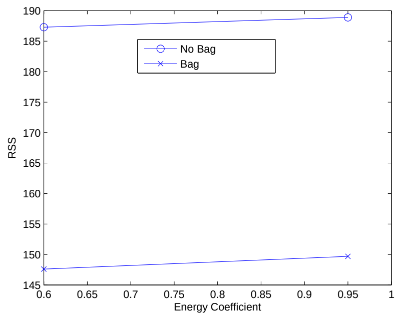
For the gradient descent approach suggested by [16], we find the value of l that corresponds to % of energy to be preserved. Thus, for our experiments, $l \in \{75, 100, 135, 160\}$. For each of the values of l, we compare the gradient descent approach by [16] to our



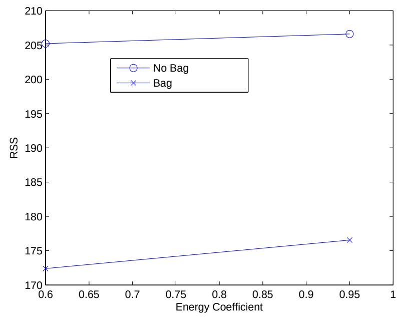
(a) Digit 5



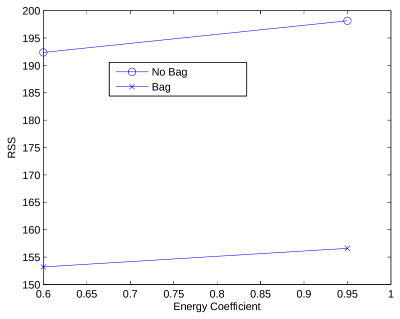
(b) Digit 6



(c) Digit 7



(d) Digit 8

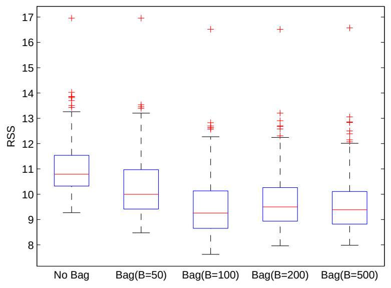


(e) Digit 9

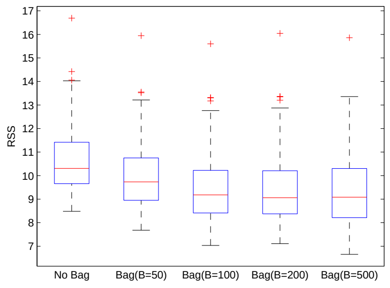
Figure 18. RSS values of penalized preimage approach versus bagged approach for digits 5-9 for the high-noise case with noise $\sigma_G = 1$.

BKPCA approach with values of $B \in \{50, 100, 200, 500\}$. The results are shown in Figures 21 through 22. We see that our BKPCA method performs significantly better than the gradient descent approach by [16] for each value of B.

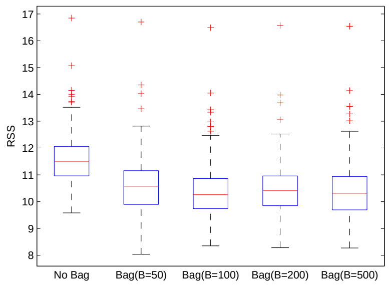
We next select digits 7 and 9, and show the boxplot of RSS for all the data points. We use the Gaussian kernel for denoising with the kernel parameter σ set to the value of $2p$



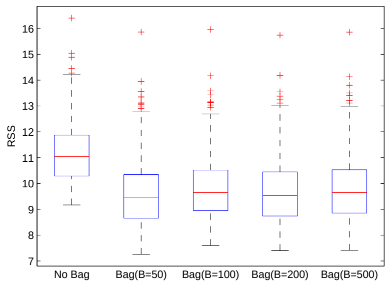
(a) Energy=70%



(b) Energy=80%



(c) Energy=90%

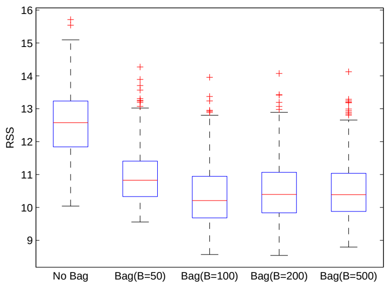


(d) Energy=95%

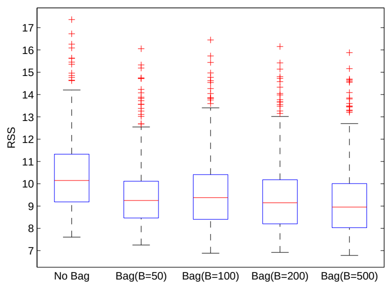
Figure 19. Boxplot of RSS values for digit 7 for BKPCA approach versus penalized preimage approach for different values of energy to be preserved and B. Independent Gaussian noise $\mu = 0$ and $\sigma_G = 1$ was added.

times the average component variances as specified by Schölkopf *et al*. The other parameters for each algorithm are set to the levels as discussed by the authors of the corresponding chapters, except we experimented with the number of eigenvalues selected (or the energy to be preserved in the penalized preimage case). We also considered two scenarios for the above cases where we added Gaussian noise with $\mu = 0$ and $\sigma_G = \{0.7, 1\}$. Figures 23 to 24 show the results for the penalized preimage algorithm. Figures 25 to 26 show the results for the gradient descent algorithm. Figures 27 to 28 show the results for KMDS algorithm. The BKPCA algorithm consistently shows better performance for RSS.

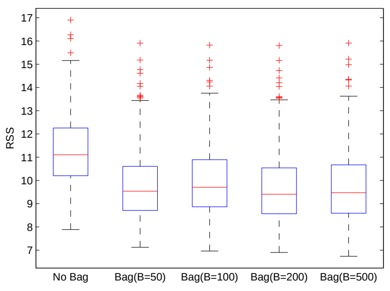
For the plots shown in Figures 19 through 22, we compute the difference between RSS obtained through the original method and our BKPCA method for each of the 300



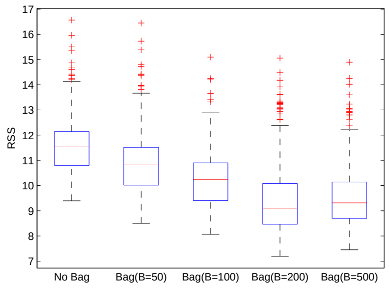
(a) Energy=70%



(b) Energy=80%



(c) Energy=90%

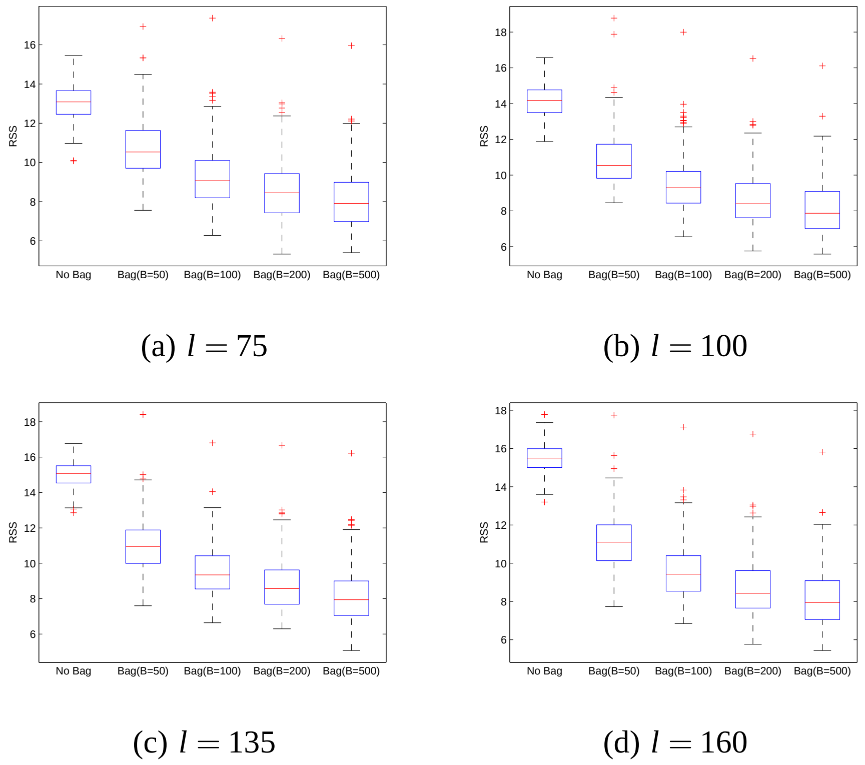


(d) Energy=95%

Figure 20. Boxplot of RSS values for digit 9 for BKPCA approach versus penalized preimage approach for different values of energy to be preserved and B. Independent Gaussian noise $\mu = 0$ and $\sigma_G = 1$ was added.

instances for different values of B. We then performed one-sided Wilcoxon signed-rank test (the alternate hypothesis being the median difference is greater than zero). The p-values obtained for all the tests on all the plots were extremely small (smaller than 0.0001). Thus, our method provides statistically significant improvement over the results obtained from other methods. For other plots, similar to ones shown in Figures 19 through 22, we obtain similar results from the one-sided Wilcoxon signed-rank test.

We also apply the methods to the digit dataset for the high-noise case with Gaussian noise $\mu = 0$ and $\sigma_G = 1$ and show the visual preimages in Figure 29 for KMDS and BKPCA methods where the preimages were obtained from the parameter settings in the experiments which resulted in the minimum RSS. For reference, we also show the noiseless images (first



(a) $l = 75$ (b) $l = 100$

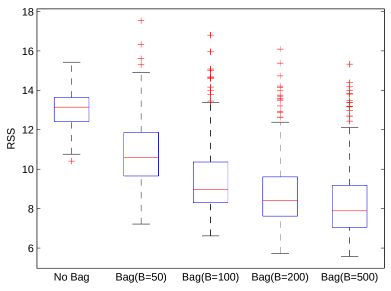
(c) $l = 135$ (d) $l = 160$

Figure 21. Boxplot of RSS values for digit 7 for BKPCA approach versus gradient descent approach for different values of l and B. Independent Gaussian noise $\mu = 0$ and $\sigma_G = 1$ was added.

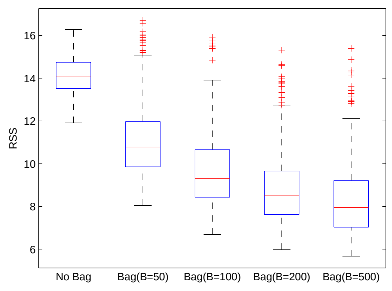
row) as well as the noisy test images (second row). We can clearly see that BKPCA method visually improves denoised preimages from the KMDS method.

6.3 *Face Data*

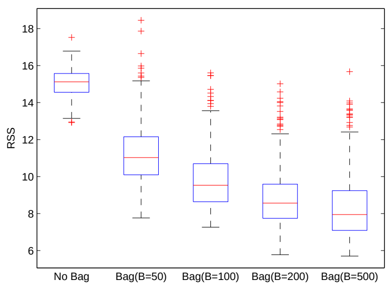
We use the face data set available at http://isomap.stanford.edu/datasets.html. There are 698 samples and the dimensionality of each sample is 4096. For our purpose, we took all 698 images, added independent Gaussian noise ($\mu = 0$ and $\sigma_G = 1$) to the images to create the noisy test set, and subsequently denoised the test set. For evaluation purposes, we compute RSS for each image. Example images are seen in the first row of Figure 33



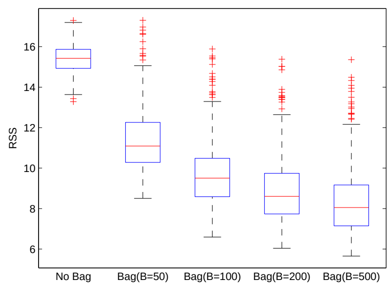
(a) $l = 75$



(b) $l = 100$



(c) $l = 135$

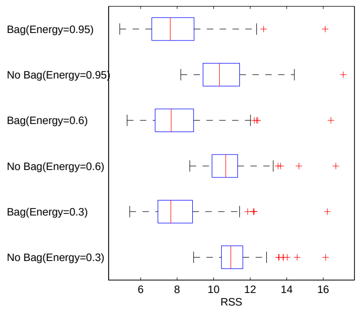


(d) $l = 160$

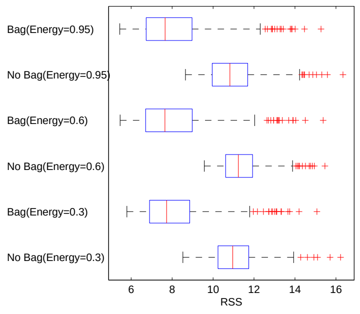
Figure 22. Boxplot of RSS values for digit 9 for BKPCA approach versus gradient descent approach for different values of l and B. Independent Gaussian noise $\mu = 0$ and $\sigma_G = 1$ was added.

Boxplots of RSS are shown for all images for the three denoising algorithms with and without BKPCA applied. Figure 30 shows the results for the penalized preimage algorithm. Note that the penalized preimage algorithm also allows for a weakly-supervised penalty term in addition to the ridge and Laplacian penalty. In our applications in manufacturing settings, information for the weakly-supervised penalty may not be available and, hence, we only use the ridge penalty for our case. Figure 31 shows the results for gradient descent algorithm. Figure 32 shows the results for KMDS algorithm.

The parameters for each algorithm were set to the levels as discussed by the authors of the corresponding chapters, except we experimented with the number of eigenvalues selected (or the energy to be preserved in penalized preimage approach). As specified

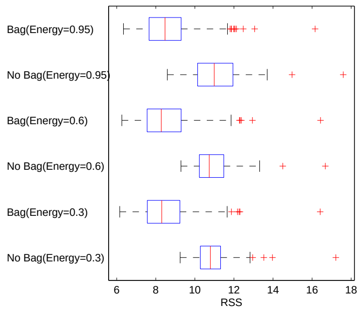


(a) Digit 7

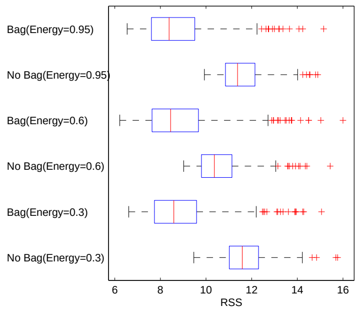


(b) Digit 9

Figure 23. RSS values of penalized preimage approach on digits 7 and 9 for different energy levels with added Gaussian noise $\sigma_G = 1$. For each energy levels, results are shown with and without bagging.



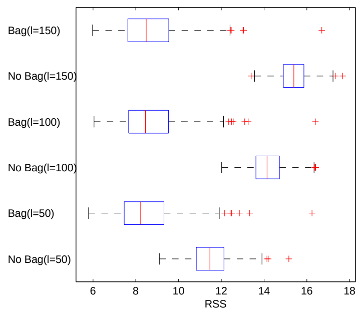
(a) Digit 7



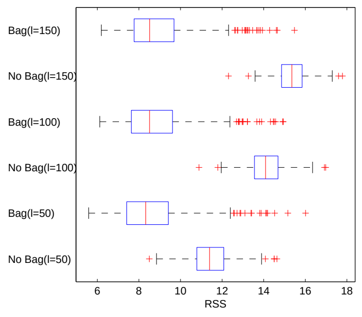
(b) Digit 9

Figure 24. RSS values of penalized preimage approach on digits 7 and 9 for different energy levels with added Gaussian noise $\sigma_G = 0.7$. For each energy levels, results are shown with and without bagging.

before, in addition to the parameters for each algorithm, we set at $B = \{50, 100\}$ for this experimental purpose. Overall, our bagged version performs much better than the original methods as can be seen from the plots. Results are not sensitive to the value selected for B.

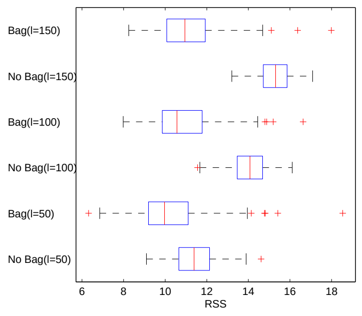


(a) Digit 7

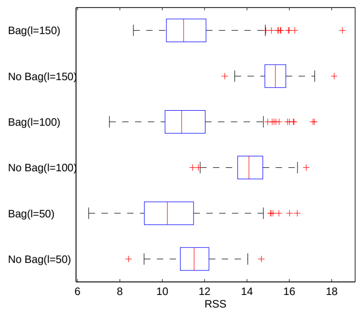


(b) Digit 9

Figure 25. RSS values of the gradient descent algorithm on digits 7 and 9 for different numbers of eigenvalues (l) with added Gaussian noise $\sigma_G = 1$. For each l, results are shown with and without bagging.

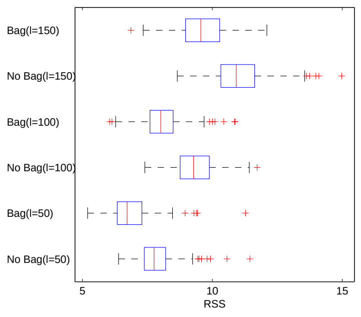


(a) Digit 7

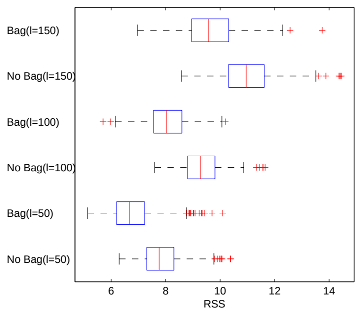


(b) Digit 9

Figure 26. RSS values of the gradient descent algorithm on digits 7 and 9 for different numbers of eigenvalues (l) with added Gaussian noise $\sigma_G = 0.7$. For each l, results are shown with and without bagging.

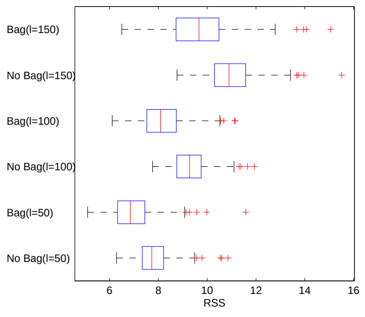


(a) Digit 7

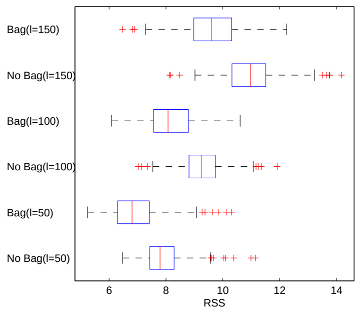


(b) Digit 9

Figure 27. RSS values of KMDS algorithm on digits 7 and 9 for different numbers of eigenvalues (l) with added Gaussian noise $\sigma_G = 1$. For each l, results are shown with and without bagging.



(a) Digit 7



(b) Digit 9

Figure 28. RSS values of KMDS algorithm on digits 7 and 9 for different numbers of eigenvalues (l) with added Gaussian noise $\sigma_G = 0.7$. For each l, results are shown with and without bagging.

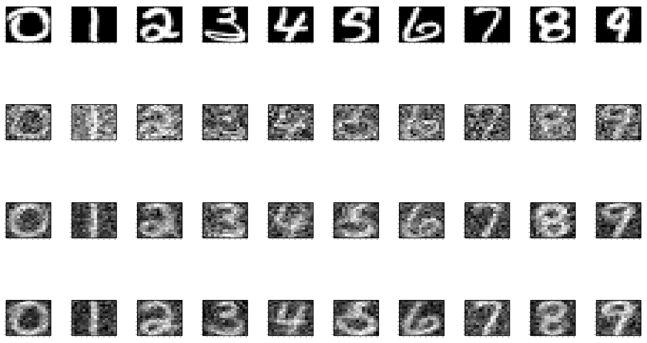
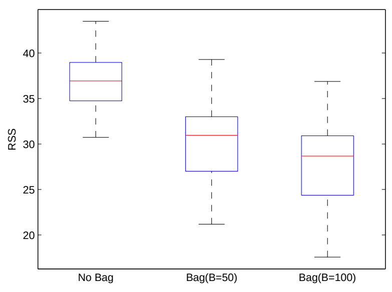
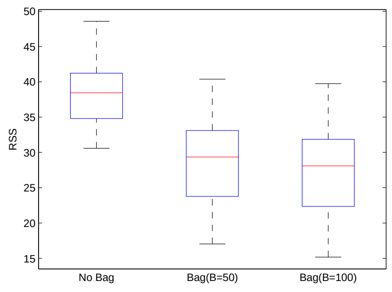


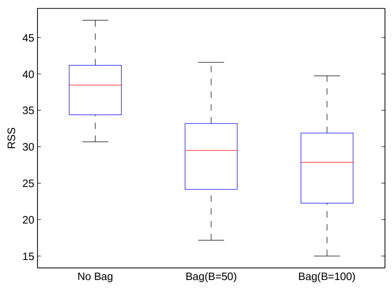
Figure 29. Results for digits 0 through 9 with $l = 50$ and $n = 10$. First row shows noiseless reference images. Second row shows noisy test images (Gaussian noise $\mu = 0$ and $\sigma_G = 1$). Third row shows denoised preimages from the KMDS method. Fourth row shows denoised preimages from our BKPCA method. The improved results from BKPCA can be seen.



(a) Energy=0.95

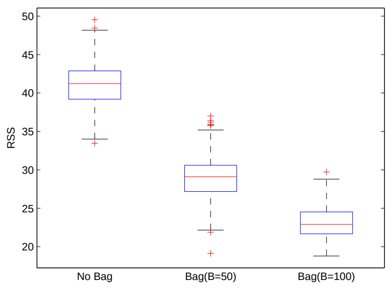


(b) Energy=0.6

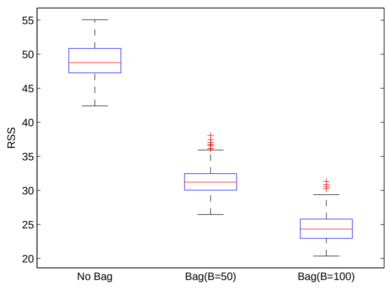


(c) Energy=0.3

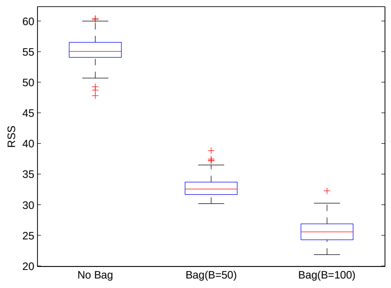
Figure 30. Face data RSS values of penalized preimage approach versus the bagged approach for different energy levels with added Gaussian noise ($\mu = 0$ and $\sigma_G = 1$).



(a) $l = 150$

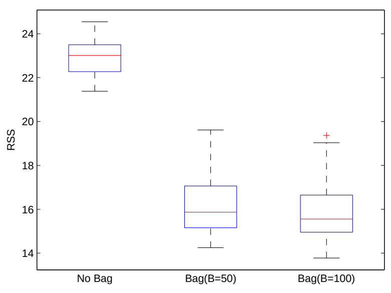


(b) $l = 300$

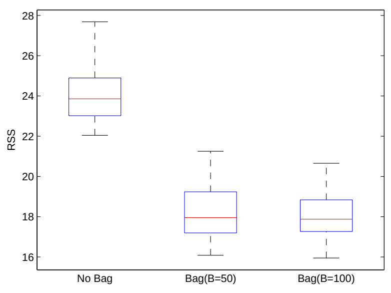


(c) $l = 400$

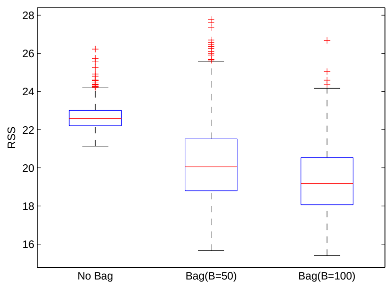
Figure 31. Face data RSS values of the gradient descent algorithm versus the bagged approach for different numbers of eigenvalues (l) with added Gaussian noise ($\mu = 0$ and $\sigma_G = 1$).



(a) $l = 150$



(b) $l = 300$



(c) $l = 400$

Figure 32. Face data RSS values of KMDS algorithm versus the bagged approach for different numbers of eigenvalues (l) with added Gaussian noise ($\mu = 0$ and $\sigma_G = 1$).

We also show the visual results of denoising the face image dataset by applying KMDS algorithm as well as the bagged version BKMDS in Figure 33. The preimages were obtained from the parameter settings in the experiments which resulted in the minimum RSS. The first row shows uncorrupted face images. The second row shows noisy faces obtained from added Gaussian noise $\mu = 0$ and $\sigma = 0.7$. We can see that visually the figures obtained from BKMDS (fourth row) are clearer than those obtained from the KMDS method (third row).

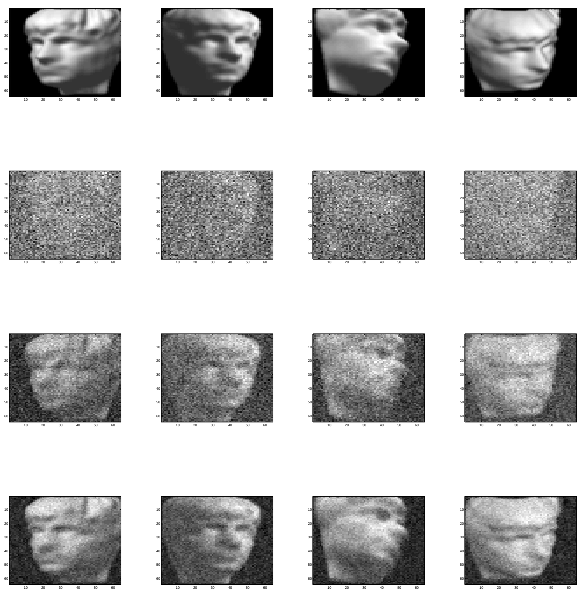


Figure 33. Results for selected face images with $l = 150$ and $n = 10$. First row shows noiseless reference images. Second row shows noisy test images (Gaussian noise $\mu = 0$ and $\sigma_G = 0.7$). Third row shows denoised preimages from the KMDS method. Fourth row shows denoised preimages from our BKPCA method. The improved results from BKPCA can be seen.

We also calculated the computational time involved in running the original methods as well as our BKPCA procedure. The parameter settings were selected to provide the minimum RSS value. We chose $B \in \{50, 100, 200, 500\}$. We used an Intel (R) Core (TM)2 Quad CPU Q6600 computer with 2.4 GHz clock speed and 4.00 GB RAM. For the exper-

TABLE 5. Average time in seconds required to denoise a data point from the USPS digits data set. The standard error is reported within parenthesis.

Method	Original	$B = 50$	$B = 100$	$B = 200$	$B = 500$
Gradient Descent	0.14 (0.01)	4.19 (0.08)	8.32 (0.13)	16.3 (0.11)	40.5 (0.21)
KMDS	0.30 (0.01)	11.8 (0.05)	23.3 (0.09)	46.8 (0.57)	117.1 (0.8)

TABLE 6. Average time in seconds required to denoise a data point from the face image data set. The standard error is reported within parenthesis.

Method	Original	$B = 50$	$B = 100$	$B = 200$	$B = 500$
Gradient Descent	0.28 (0.01)	23.9 (0.18)	47.0 (0.1)	93.5 (0.3)	235.5 (0.64)
KMDS	0.34 (0.01)	15.5 (0.08)	30.4 (0.2)	60.9 (0.2)	152.3 (0.23)

iments, the average time in seconds required to denoise a single data point (with standard error) is reported for different cases in Tables 5 through 6. We observe that the time taken by BKPCA procedure is more than the time taken by other methods. However, the improvement in RSS is significant. In practice, we can use a modest value of B ($B = 50$) to obtain a reasonable tradeoff between the computational time involved and the desired decrease in RSS. Moreover, the BKPCA procedure can be parallelized easily lowering their computational time which is not significantly greater than the computational time for other procedures.

7 Conclusions

A new method to approximate the preimage of a denoised signal is provided that uses bagging to compensate for noisy training data. For applications such as manufacturing analysis and improvement it is important to interpret and visualize results so that the preimage problem is an important element to extend analytical methods in these domains. However, noise-less training data can be problematic. Our BKPCA method substantially improves the original methods in our experimental results on the datasets here. The improvement is most pronounced when the parameters differ from those that minimize the RSS. Consequently, BKPCA improves the robustness of the base algorithm. Visual comparison with the true images provide evidence that the model selected was able to identify the underlying variation structure. Although BKPCA is slightly more computationally expensive due to the bootstrap replicates, the algorithm still ran quickly in our experiment. Furthermore, the bagging approach easily lends itself to a parallel implementation that can increase the speed of computations.

We currently propose an ensemble approach for estimating the preimage by averaging over several preimage estimates obtained from different bootstrap samples from the training data. Each bootstrap sample consists of observations which are randomly drawn from the original training set with replacement. In future, we would like to investigate whether sampling without replacement has an effect on our preimage estimate. Finally, in addition to randomly selecting instances from training data, we would like to randomly select features from the training set and estimate preimages from an ensemble of feature subsets. This is also expected to provide robustness to the noise in data.

TABLE 7. List of symbols used in Chapter 1

N	Number of data points
p	Dimension of each data point in input space
F	Kernel feature space
$\mathbf{x}_i$	Data point i in input data matrix, $i = 1, 2, \cdots, N$
$\mathbf{X}$	Input data matrix
$\hat{\mathbf{x}}_i$	Denoised i^{th} point, $i = 1, 2, \cdots, N$
$\hat{\mathbf{X}}$	Denoised matrix
$\mathbf{x}$	Test point
$\hat{\mathbf{x}}$	Estimated preimage of test point
$\mathbf{t}$	True image of test point $\mathbf{x}$
$\mathbf{O}$	Matrix with all elements $1/N$
B	Number of bootstrap samples
l	Number of top nonzero eigenvalues chosen
$\mathbf{v}$	Eigenvectors of $\mathbf{C}$
$\mathbf{C}$	Covariance matrix in kernel feature space F
$\boldsymbol{\alpha}$	Eigenvectors of $\mathbf{K}$
$\mathbf{K}$	Kernel matrix
$\hat{\mathbf{X}}(b)$	Denoised matrix from b^{th} bootstrapped sample, $b = 1, 2, \ldots, B$
$\varphi(\cdot)$	Mapping from input space to kernel feature space F
$\tilde{\mathbf{K}}$	Modified kernel matrix
β_k	Projections on the k^{th} component for $k = 1, 2, \ldots, l$
$\mathbf{t}_i$	i^{th} data point in true data matrix, $i = 1, 2, \ldots, N$
d	Dimension of manifold in which true data reside
$P_l\varphi(\mathbf{x})$	Denoised point in F
D_0	Training data set in BKPCA algorithm
D_{test}	Test data set in BKPCA algorithm
σ	Parameter for Gaussian kernel
s	Parameter for Polynomial kernel
n	Number of nearest neighbors
D_b	Bootstrap sample, $b = 1, 2, \ldots, B$
$\rho(\hat{\mathbf{x}})$	Squared distance between $\varphi(\hat{\mathbf{x}})$ and $P_l\varphi(\mathbf{x})$
Ω	Terms independent of $\hat{\mathbf{x}}$ in calculating $\rho(\hat{\mathbf{x}})$

CHAPTER 4

A SERIAL APPROACH TO VARIATION PATTERNS IN KERNEL PCA

1 Introduction

Massive amount of in-process data is ubiquitous in many manufacturing organizations. With advances in data acquisition and storage technologies, it is now possible to track several thousands of dimensional characteristics. Datasets collected are in the form of spatial profiles, images or time series. Analyzing this data to glean useful information is necessary to identify root causes of product or process variation. This can provide actionable insights to engineers to make decisions.

Consider an example for manufacturing automotive engine gaskets. A critical component on the gasket is a bead which is used to create a seal. Using a beaded gasket distributes the load on the gasket to the areas where the bead is applied and often removes the need to reconfigure the flange. The data is collected by scanning a stylus across the gasket bead to obtain a profile where each profile can be discretized into a fixed number of points. For each part, the measurement vector (data) consists of measuring the vertical axis profile heights at each of the points. A signal is observed by the flattening and elongation of bead on each part. This signal is the result of a systematic part-to-part variation pattern due to application of load on the gasket. However, this signal can be obfuscated by noise. We also note that the pattern (represented by the relationships between different elements of the measurement vectors for all parts) is nonlinear.

In the above example, the non-linear pattern obtained when a single source of variation (corresponding to bead flattening/elongation) is present in the process. In practice, multiple sources of variation interact with each other resulting in a unique pattern in the data. A key step to identifying root causes of process variability is ,thus, identifying the important variation patterns. Noise makes it harder for the underlying pattern to be recognized in real conditions. Therefore, it is important to remove noise from the dataset in order to visualize the underlying patterns.

There are many techniques proposed in literature to remove noise. Principal component analysis (PCA) is a widely used technique in manufacturing control literature. PCA provides a simple way to identify the variation pattern. However, PCA tends to lose its effectiveness in identifying patterns when the resultant pattern is nonlinear in nature as shown by [22]. They extended the linear PCA framework to account for nonlinear structures in the data set through kernel principal component analysis (KPCA). KPCA maps the data in the input space to a higher dimensional (possibly infinite) feature space via a nonlinear map $\varphi : R^p \to R^m$ where m is the number of features and the feature space is denoted as F. Linear PCA is then applied to the mapped points in the feature space to extract components that are nonlinear in the original input space. Kernel trick shown by [1] is used to compute the inner products in the feature space, thus avoiding the computational burden required to explicitly do the same in the large dimensional feature space. Projections onto a smaller subset of principal component directions in the feature space can be used to denoise signals.

To interpret the denoised signal, it is valuable to visualize it in the original input space. This process of obtaining the inverse transformation is referred to as obtaining the preimage in KPCA literature. However, such an inverse transformation from the feature space to the input space does not typically exist as shown by [16]. Instead an approximate preimage is typically sought which is the focus of this research. Previous work by [16] defined this as a nonlinear optimization problem and approached it using standard gradient descent. A drawback of using standard gradient descent methods is their convergence to local minima. Also this method is sensitive to choices of initial starting values. [11] used an algebraic approach to find approximate preimages by exploiting the relationship between the distance in feature space and the corresponding distance in input space derived by [26] for commonly used kernels along with multidimensional scaling (MDS). This method was an improvement over the one suggested by [16]. [4] applied kernel regression approach to the preimage problem by formulating the inverse mapping from feature space to input space as a regression problem. [18] extended the KPCA framework to handle noise, outlier and missing data. [30] addressed the problem of estimating a better preimage by adding penalty terms to the preimage learning process. [23] and [20] considered meta-method to improve the preimage results through bagging. Recently [8] pointed out limitations inherent in the orthogonal projection operation in any KPCA algorithm , and proposed to modify it by incorporating information about the local geometry in the neighborhood of a test point in feature space so that the projection of the corresponding preimage remains closer to the full manifold. The full manifold was defined by [8] to be the set of all points in feature space that have exact preimages. Finally [19] considered feature selection in kernel PCA with sparse random vectors. This approach can be applied prior to the preimage methods discussed here to reduce the dimensionality of the problem.

Most of the previous methods to denoise for KPCA assume that the training data are noise-free. In practice, many cases (such as manufacturing variation analysis) fail to meet this assumption. We are only given a noisy training set to learn from and subsequently denoise to observe the variation pattern. To improve the estimate of the preimage in such cases, we provide a new meta approach. The idea is that the initial estimate of the actual denoised test set obtained by a KPCA preimage estimation algorithm may not be accurate because of the inherent noise in the data; hence, successive iterations of denoising a convex combination of the test set and the corresponding denoised set can lead us to a more accurate estimate of the actual denoised test set. We also consider another variant of the above approach where we decrease the number of top eigenvectors chosen in each iteration at a constant rate. The intuition is that we initially retain all eigenvectors so as not to loose any information about the pattern in data and as we approach towards the final denoised preimage, we only retain the top most eigenvectors that will account for the structure in data and get rid of the noise. The remainder of this chapter is organized as follows. Section 2 offers a brief review of KPCA and the ensuing preimage estimation algorithms. Section 3 discusses our proposed methodology. Section 4 provides experimental results and Section 5 provides conclusions.

2 Background on Preimages in KPCA

KPCA is equivalent to PCA in feature space ([22]). Let $\mathbf{X}$ denote the data set with N instances and F features where the instances are denoted by $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$. 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 $\varphi(\mathbf{x}_i), i = 1, 2, \cdots, N$ are assumed to be centered, $\mathbf{C}$ can be calculated by

$$\mathbf{C} = \frac{1}{N} \sum_{i=1}^{N} \varphi(\mathbf{x}_i) \varphi(\mathbf{x}_i)' \tag{4.1}$$

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

$$\mathbf{C}\mathbf{v} = \lambda \mathbf{v} \tag{4.2}$$

It can be shown that an eigenvector corresponding to non-zero eigenvalue of $\mathbf{C}$ can be written as a linear combination of $\varphi(\mathbf{x}_1), \cdots, \varphi(\mathbf{x}_N)$. Let there be coefficients α_i ($i = 1, \ldots, N$) such that

$$\mathbf{v} = \sum_{i=1}^{N} \alpha_i \varphi(\mathbf{x}_i) \tag{4.3}$$

Using the above simplification reduces the original problem of finding eigenvalues and eigenvectors of $\mathbf{C}$ to finding the corresponding eigenvalues and eigenvectors of the kernel matrix $\mathbf{K}$ with entries

$$\mathbf{K}_{ij} := (\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)) \tag{4.4}$$

The product $\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$ is evaluated using the kernel trick [1] without explicitly computing the mapping $\varphi(.)$.

Let $\mathbf{x}$ be a test point in input space with a corresponding centered map $\varphi(\mathbf{x})$ in the feature space. In order to extract nonlinear principal components for the $\varphi-$image of the test point $\mathbf{x}$, we compute its projections on the k^{th} component for $k = 1, \cdots, l$ as follows

$$\beta_k = (\mathbf{v}^k \cdot \varphi(\mathbf{x})) = \sum_{i=1}^{N} \alpha_i^k (\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x})) = \sum_{i=1}^{N} \alpha_i^k k(\mathbf{x}, \mathbf{x}_i) \tag{4.5}$$

where $\boldsymbol{\alpha}^k$ denotes a column vector with entries $\alpha_1, \ldots, \alpha_N$ for $k = 1, \ldots, l$. The dot products are evaluated using a kernel function.

The denoised image in feature space can be obtained by projecting $\varphi(\mathbf{x})$ onto a subspace spanned by the top l eigenvectors

$$P_l \varphi(\mathbf{x}) = \sum_{k=1}^{l} \beta_k \mathbf{v}^k \tag{4.6}$$

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 overall procedure for obtaining the 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. The second step is to calculate the principal component directions of the training data in feature space as shown by [22]. 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 Figure 34.

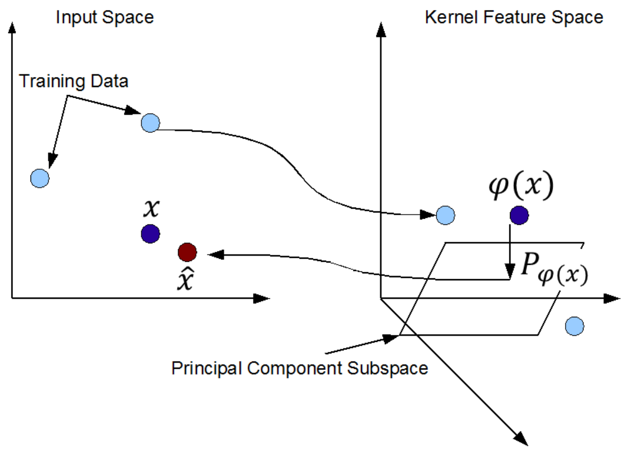


Figure 34. KPCA and the preimage problem. Training data are transformed to feature space and used to learn a principal component plane. A test point x is transformed and projected to the plane as $P\varphi(x)$. The inverse transform of $P\varphi(x)$ may not exist, and an approximate preimage $\hat{x}$ is computed.

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 [16]. Hence, several algorithms have been proposed to estimate the preimage. [16] 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. [11] 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. [11] chose only a few nearest training data points in order to reduce the computational burden. We refer to the method used by [11] as KMDS. [4] 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 [11] and [4] favor noise-free training data.

3 Preimages from Serial Denoising

As discussed before, we want to improve upon the previous approaches proposed in literature to handle the case when training data is noisy. The overall idea is that since the training data is noisy, the principal component subspace is not effectively learned. Thus, we take an approach to serially learn a reliable estimate of the principal component subspace. We present a method called Serial denoising and a variant of it called Eigen denoising based on the above concept.

3.1 Serial Denoising

We consider the original test set $\mathbf{X}_0$. Let us denote the denoising function (any KPCA preimage estimation algorithm) applied to $\mathbf{X}_{i-1}$ as $g(\mathbf{X}_{i-1})$ for $i = 1, 2, \cdots, N_{max}$. The denoised data at iteration i is calculated by

$$\mathbf{X}_i = (\alpha)\mathbf{X}_{i-1} + (1-\alpha)g(\mathbf{X}_{i-1}), i = 1, 2, \cdots, N_{max} \tag{4.7}$$

Here $\mathbf{X}_i$ is considered the denoised data at iteration i because $g(\mathbf{X}_{i-1})$ is considered as a large step for denoising. We describe the procedure below in Algorithm 1.

Algorithm 1 Algorithm for serial Denoising

```
We consider the original test set $\mathbf{X}_0$. Fix values for the number of eigenvectors $l$, a user-defined constant $\alpha$, kernel parameter $\sigma$, $iter = 1$. Fix any other KPCA algorithm specific parameters (for instance number of nearest neighbors $n$ in KMDS algorithm). We have to estimate the denoised preimage.
repeat
    Denoise $\mathbf{X}_{iter-1}$ by computing $g(\mathbf{X}_{iter-1})$
    $\mathbf{X}_{iter} = (\alpha)\mathbf{X}_{iter-1} + (1-\alpha)g(\mathbf{X}_{iter-1})$
    The denoised matrix at step $iter$ is denoted by $\mathbf{X}_{iter}$
    $iter \leftarrow iter + 1$
until $iter = N_{max}$
```

The final denoised matrix is obtained at the iteration where RSS is the minimum, the RSS in each iteration being defined as

$$RSS_i = \|\mathbf{X}_i - \mathbf{X}^*\|_F \,, i = 1, 2, \cdots, N_{max} \tag{4.8}$$

where $\mathbf{X}^*$ is the true data matrix. This is easy to calculate when we know the true data matrix. In practice, however, the true data matrix is not known beforehand, and thus, we need a stopping rule to determine the final denoised matrix. Sometimes signal-to-noise ratio (SNR) is also used for evaluation purpose where SNR (in dB) for denoised matrix $\mathbf{X}$ is defined as

$$SNR = -10\log_{10}\frac{\|\mathbf{X} - \mathbf{X}^*\|_F^2}{\|\mathbf{X}^*\|_F^2} \tag{4.9}$$

It can be easily verified that low values of RSS will correspond to high values of SNR.

3.1.1 Stopping Rule for serial Denoising We define the stopping rule as follows. Let

$$L_i = \|\mathbf{X}_i - \mathbf{X}_{i-1}\|_F \,, i = 1, 2, \cdots, N_{max} \tag{4.10}$$

We define ε as a small positive constant. For serial denoising, we stop at iteration i when $L_{i-1} - L_i \leq \varepsilon$; else we stop at iteration $i = N_{max}$.

3.2 Eigen Denoising

We consider another variant of serial denoising called Eigen denoising (EKPCA) where we decrease the number of eigenvectors at a constant rate in each iteration. The eigenvectors are recomputed from the kernel matrix in each iteration. Also let $g(.)$ denote any KPCA preimage estimation function applied to the data. EKPCA is described in Algorithm 2.

Algorithm 2 Algorithm for Eigen Denoising

```
We consider the original test set $\mathbf{X}_0$. Fix values for the initial fixed number of eigenvectors $l$, a user-defined constant $\alpha$, kernel parameter $\sigma$, a user- defined constant $\delta$, $iter = 1$. Fix any other parameters used in a KPCA preimage estimation algorithm (for instance number of nearest neighbors $n$ in KMDS algorithm). We estimate the denoised data set as follows.
repeat
    $l^* \leftarrow l - \delta(iter - 1)$
    Denoise $\mathbf{X}_{iter-1}$ by computing $g(\mathbf{X}_{iter-1})$ using eigenvectors corresponding to top $l^*$ eigenvalues
    $\mathbf{X}_{iter} = (\alpha)\mathbf{X}_{iter-1} + (1-\alpha)g(\mathbf{X}_{iter-1})$
    The denoised matrix at step $iter$ is denoted by $\mathbf{X}_{iter}$
    $iter \leftarrow iter + 1$
until $l^* \leq 1$
```

Note that by setting $\delta = 0$, eigen denoising becomes equivalent to serial denoising. The final denoised matrix is obtained at the iteration where RSS is the minimum, the RSS being defined as

$$RSS_i = \|\mathbf{X}_i - \mathbf{X}^*\|_F \,, i = 1, 2, \cdots, N_{max} \tag{4.11}$$

where $\mathbf{X}^*$ is the true data matrix. This is easy to calculate when we know the true data matrix. In practice, however, the true data matrix is not known beforehand, and thus, we need a stopping rule to estimate the final denoised matrix.

3.2.1 Stopping Rule for Eigen Denoising We define the stopping rule as follows. Let

$$L_i = \|\mathbf{X}_i - \mathbf{X}_{i-1}\|_F \,, i = 1, 2, \cdots, N_{max} \tag{4.12}$$

We define ε as a small positive constant. For eigen denoising, we stop at iteration i when $L_{i-1} - L_i \leq \varepsilon$; else we stop when $l \leq 1$, where l is the number of eignevectors.

Basically, we can think of our parameters α for serial denoising (with δ for EKPCA) providing some form of shrinkage that allows us to serially approach the final denoised preimage. At each step, we re-estimate our prinicpal component subspace beased on the denoising at the previous step. This is expected to perform better than estimating the principal component subspace only once especially in situations when we have noisy training data. Our stopping rule also ensures that we obtain the desired preimage in a fewer number of iterations. Another advantage of our approach is it works with any KPCA algorithm in literature.

4 Experimental Results

We consider two datasets- a classical hand-written digits dataset and a face dataset for the purpose of our experiments. We evaluate our serial denoising procedure as well as its variant (EKPCA) on the data sets. We use a Gaussian kernel for our experiments given by the following equation

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|_F^2}{\sigma}\right) \tag{4.13}$$

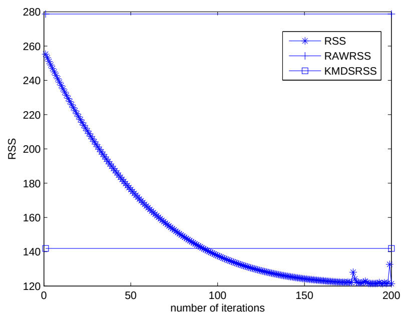
where σ is a parameter related to the width of the kernel between data points $\mathbf{x}_i, \mathbf{x}_j$.

4.1 Experiment Results for Serial Denoising on USPS Digits Dataset

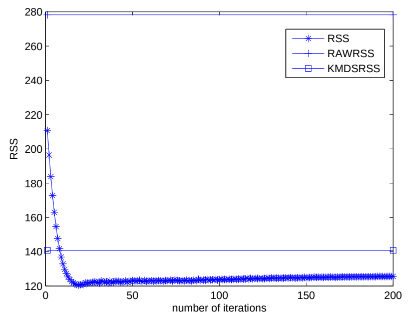
We consider the USPS digits dataset at http://yann.lecun.com/exdb/mnist/. It consists of 16×16 gray scale images of zip code digits (0-9) automatically scanned from envelopes by the U.S.Postal Service. We initially investigate the effects of different parameters on our results. To see the effect of α, we chose digit 9 and set $\sigma = 50$, $l = 100$, $n = 10$, and $N_{max} = 200$. [1] We added Gaussian noise with $\mu = 0$ and $\sigma_G = 1$. We varied $\alpha \in \{0.9, 0.7, 0.5, 0.3, 0.1, 0\}$. Figure 35 shows the plots of RSS against the number of iterations for different values of α as specified below each subfigure. For reference, we also show RSS obtained using $\mathbf{X}_0$ ($\|\mathbf{X}_0 - \mathbf{X}^*\|_F$) as the acronym "RAWRSS", and the RSS obtained from the KMDS method ($\|g(\mathbf{X}_0) - \mathbf{X}^*\|_F$) as the acronym"KMDSRSS".

Based on the experiments, we found that the minimum value of RSS achieved under different α values is not significantly different. However, we also found that for high values of α ($\alpha = 0.9$), the RSS decreases slowly over the iterations whereas for small values of α, RSS decreases initially to the lowest value and then shows an upward trend. In practice, therefore, we will use the stopping rule for small values of α to obtain the denoised data matrix as described previously. In order to show the efficacy of the stopping rule, we chose to experiment on all digits 0-9. We chose $\sigma = 50$; $l \in \{50, 100\}$; $\alpha = 0.1$; $n = 10$. We set $N_{max} = 200$ which is large enough. We added Gaussian noise with $\mu = 0$ and $\sigma_G = 1$. Figures 36-37 show the results. The acronym "KMDSRSS" shows the RSS achieved by the KMDS method; the acronym "ORACLERSS" stands for the true minimum RSS achieved theoretically by our method; the acronym "STOPRULERSS" stands for the RSS achieved practically using the stopping rule. We would like to mention that the RSS cannot be calculated in practice becasue we won't know the true data matix. We used the true matrix here only to show that our stopping rule produces RSS ("STOPRULERSS") which is close enough (slightly higher than) to the true minimum RSS ("ORACLERSS") while significantly lower than the RSS achieved by the KMDS method. Also the stopping rule is simple enough to be implemented in practice. For $\alpha \in \{0.1, 0\}$ and all the digits, we show

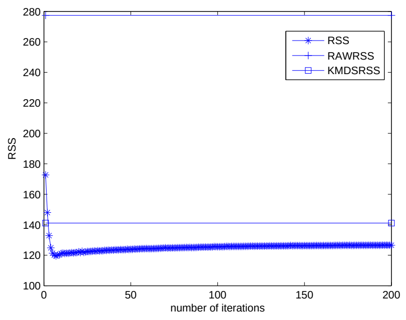
[1]similar results were obtained for other digits and other parameter settings also



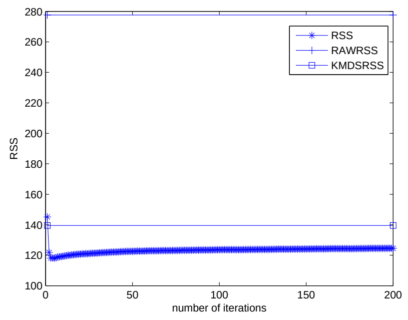
(a) $\alpha = 0.9$



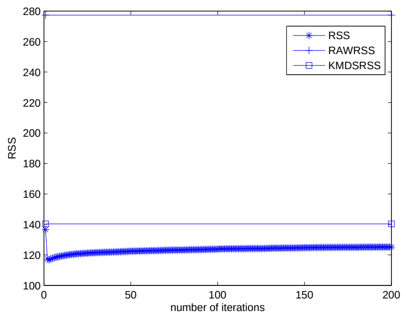
(b) $\alpha = 0.7$



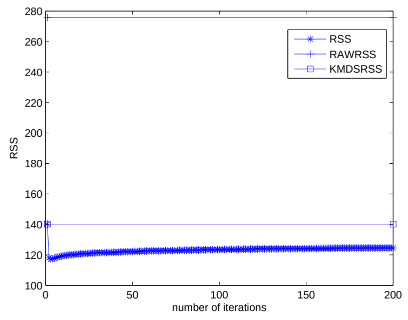
(c) $\alpha = 0.5$



(d) $\alpha = 0.3$



(e) $\alpha = 0.1$

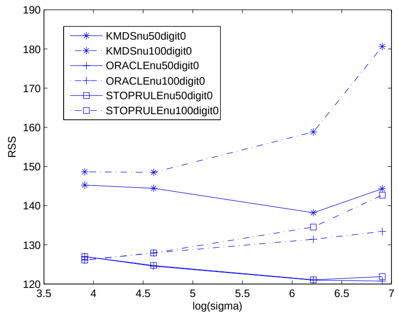


(f) $\alpha = 0$

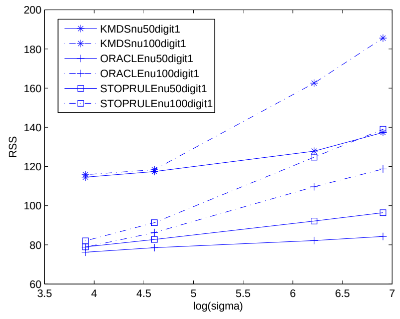
Figure 35. Effect of α on serial denoising of digit 9 with $\sigma = 50$, $l = 100$, $n = 10$, $N_{max} = 200$, and $\sigma_G = 1$

the plot of the iteration number at which the ORACLERSS is attained versus the iteration number at which the serial denoising stops due to stopping rule in Figure 38.

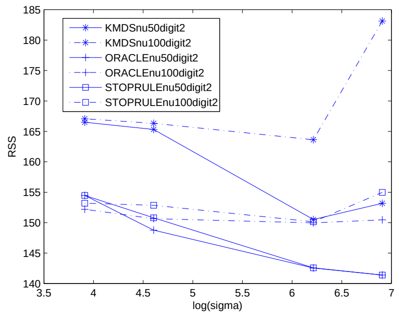
We did some more experiments on digits 7 and 9 by setting $n = 25$ and rest other parameters as described previously. The results shown in Figure 39 are similar to the results obtained in the previous figures.



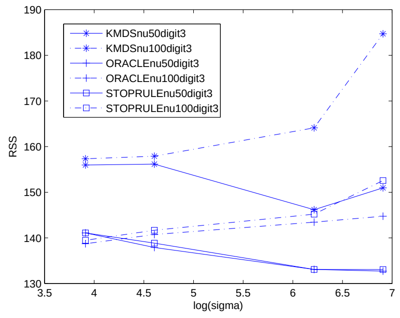
(a) Digit 0



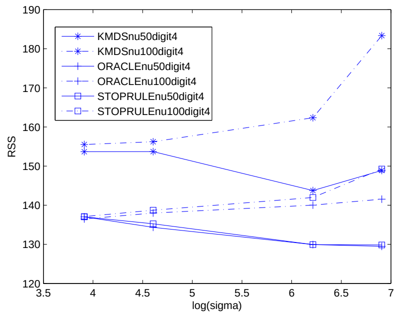
(b) Digit 1



(c) Digit 2



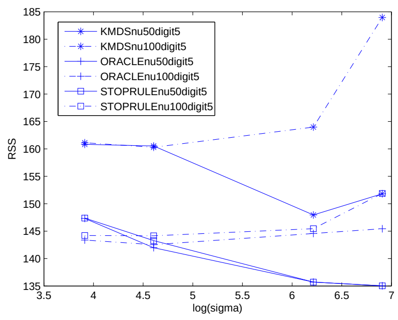
(d) Digit 3



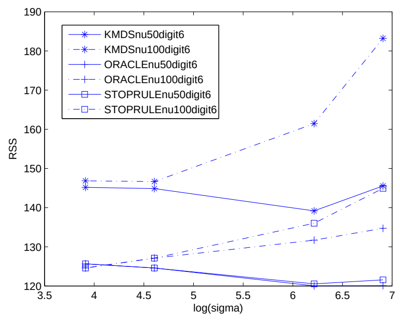
(e) Digit 4

Figure 36. Comparison of RSS values achieved by KMDS method with serial denoising for digits 0-4 setting $\sigma \in \{50, 100, 500, 1000\}$; $l \in \{50, 100\}$; $\alpha = 0.1$; $n = 10$; $N_{max} = 200$; Gaussian noise with $\mu = 0$ and $\sigma_G = 1$

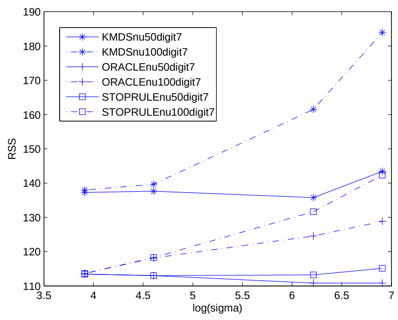
We also experimented on the digits dataset with the gradient descent algorithm. Figure 40 shows the results of the experiments on digits 6 and 8. It can be clearly seen that the serial denoising procedure improves upon the base algorithm. Also we see that the



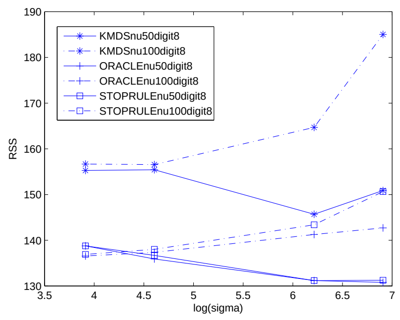
(a) Digit 5



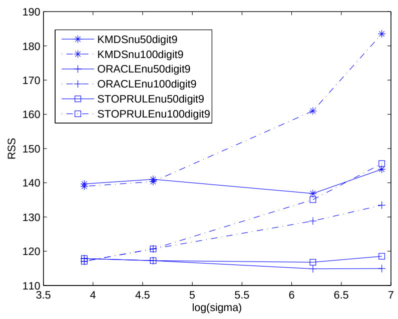
(b) Digit 6



(c) Digit 7



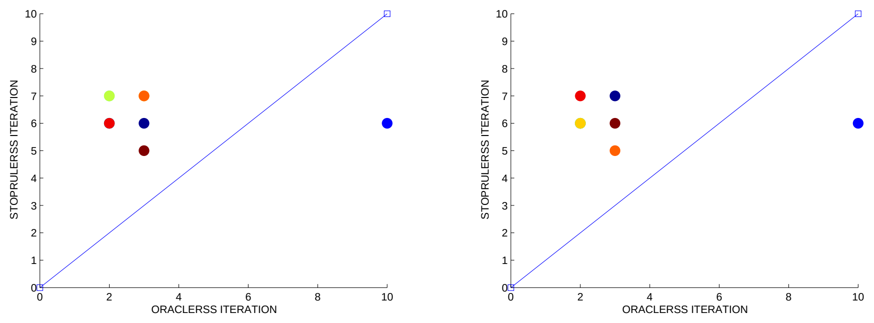
(d) Digit 8



(e) Digit 9

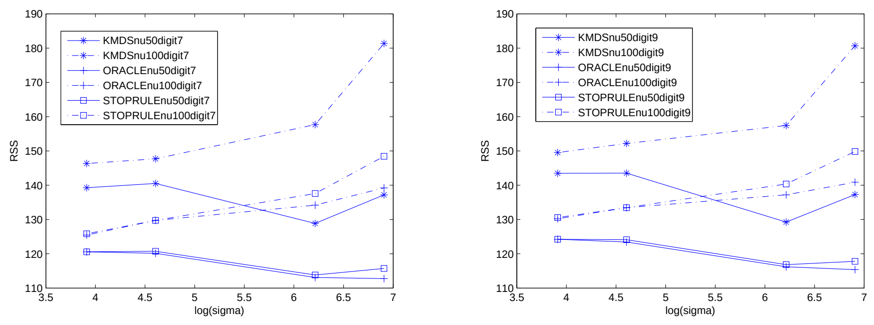
Figure 37. Comparison of RSS values achieved by KMDS method with serial denoising for digits 5-9 setting $\sigma \in \{50, 100, 500, 1000\}$; $l \in \{50, 100\}$; $\alpha = 0.1$; $n = 10$; $N_{max} = 200$; Gaussian noise with $\mu = 0$ and $\sigma_G = 1$

minimum values of RSS obtained under different values of α are not significantly different.



(a) $\alpha = 0.1$ (b) $\alpha = 0$

Figure 38. Plot of the iteration number at which the ORACLERSS is attained versus the iteration number at which the serial denoising stops due to stopping rule for all digits 0-9 setting $\sigma = 50$, $l = 100$, $n = 10$, $N_{max} = 10$, $\alpha = 0.1$



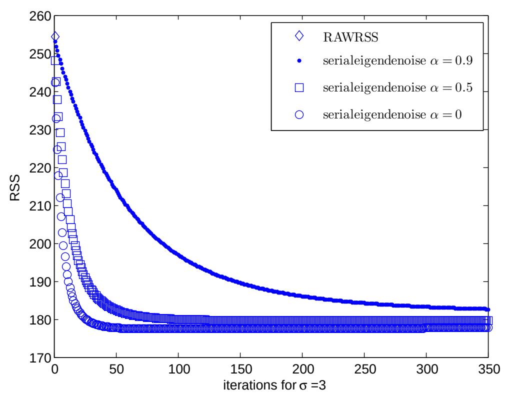
(a) Digit 7 (b) Digit 9

Figure 39. Comparison of RSS values achieved by KMDS method with serial denoising for digits 7 and 9 setting $\sigma \in \{50, 100, 500, 1000\}$; $l \in \{50, 100\}$; $\alpha = 0.1$; $n = 25$; $N_{max} = 200$; Gaussian noise with $\mu = 0$ and $\sigma_G = 1$

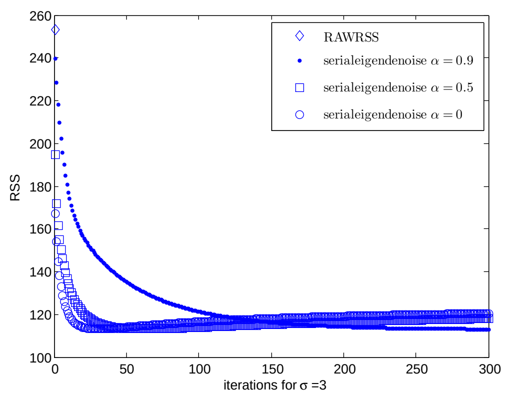
4.2 Results of Serial Denoising on Face Data

We also applied our serial procedure on the face image data set available at http://isomap.stanford.edu/datasets.html.

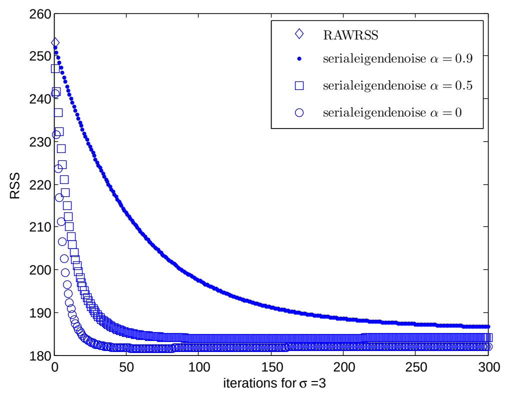
The data set contains 698 samples and dimensionality of each sample is 4096. For our purpose, we randomly took 300 samples, added independent Gaussian noise ($\mu = 0$ and $\sigma_G = 0.7$) to the samples to create the noisy test set. We used KMDS and gradient descent algorithm for denoising all images, and compare it with the respective serial denoising pro-



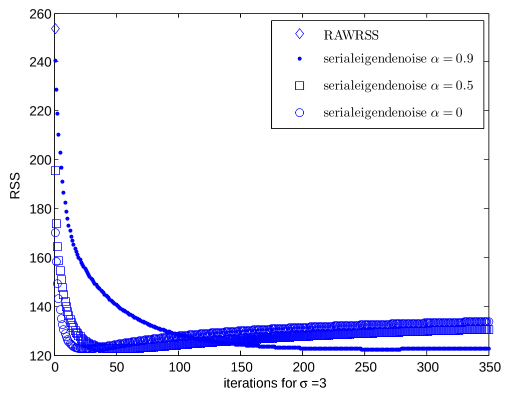
(a) Digit 6, $l = 200$



(b) Digit 6, $l = 50$



(c) Digit 8, $l = 200$

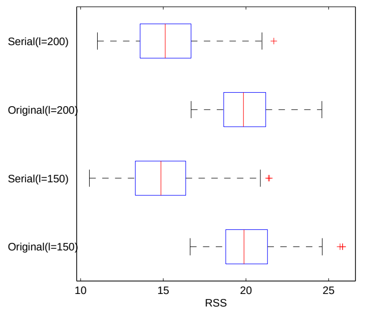


(d) Digit 8, $l = 50$

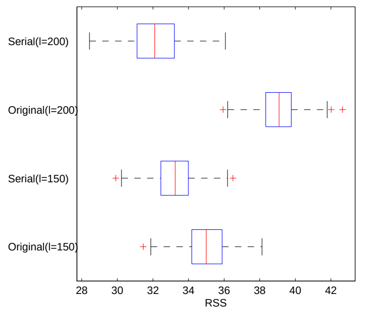
Figure 40. Comparison of RSS values achieved by gradient descent algorithm with the serial denoising procedure for digits 6 and 8 for $l \in \{50, 200\}$; Gaussian noise with $\mu = 0$ and $\sigma_G = 1$

cedures. Figure 41 shows the boxplot results of RSS obtained for each method. It is clear that the serial denoising procedure is better than the base algorithm.

For the plots shown in Figure 41, we compute the difference between RSS obtained through the original method and our serial denoising method for each of the 300 instances. We then performed one-sided Wilcoxon signed-rank test (the alternate hypothesis being the median difference is greater than zero). The p-values obtained for all the tests on all the plots were extremely small (smaller than 0.0001). Thus, our method provides statistically significant improvement over the results obtained from other methods.



(a) KMDS algorithm versus serial denoising



(b) Gradient descent algorithm versus serial denoising

Figure 41. Comparison of RSS values for the face image data set with Gaussian noise ($\mu = 0$ and $\sigma_G = 0.7$). The parameters other than l (the number of eigenvalues) is set to the values as described in the respective references

We also show the visual results of denoising the face image dataset by applying the gradient descent algorithm as well as applying the serial denoising procedure in Figure 42. The first row shows uncorrupted face images. The second row shows noisy faces obtained from added Gaussian noise $\mu = 0$ and $\sigma = 0.7$. We can see that visually the figures obtained from serial denoising (fourth row) are clearer than those obtained from the gradient descent method (third row).

We also calculated the computational time involved in running the original methods as well as our serial denoising procedure (without the stopping rule criterion). The parameter settings were selected to provide the minimum RSS value. We chose $N_{max} \in \{50, 100, 200, 500\}$. We used an Intel (R) Core (TM)2 Quad CPU Q6600 computer with 2.4 GHz clock speed and 4.00 GB RAM. For the experiments, the average time in seconds required to denoise a single data point (with standard error) is reported for different cases in Tables 8 through 9.

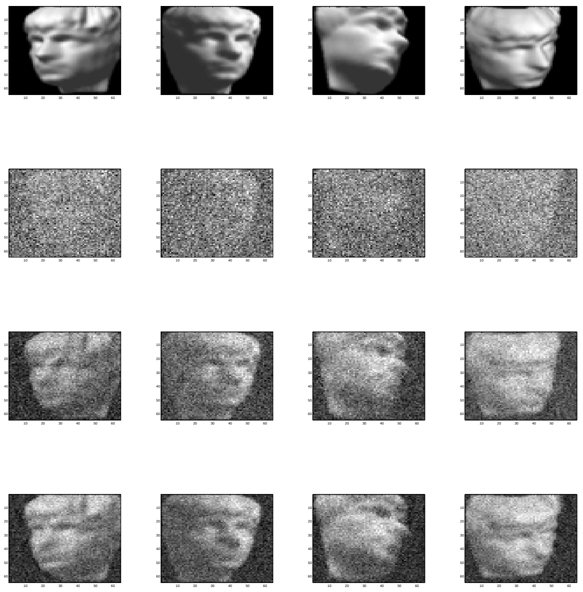


Figure 42. Results for selected face images with $l = 150$. First row shows noiseless reference images. Second row shows noisy test images (Gaussian noise $\mu = 0$ and $\sigma_G = 0.7$). Third row shows denoised preimages from the gradient descent method. Fourth row shows denoised preimages from our serial denoising method. The improved results from serial denoising can be seen.

TABLE 8. Average time in seconds required to denoise a data point from the USPS digits data set. The standard error is reported within parenthesis.

Method	Original	$N_{max} = 50$	$N_{max} = 100$	$N_{max} = 200$	$N_{max} = 500$
Gradient Descent	0.15 (0.01)	5.5 (0.08)	10.9 (0.13)	21.5 (0.15)	53.8 (0.21)
KMDS	0.31 (0.01)	13.1 (0.06)	25.9 (0.11)	52.1 (0.59)	130.2 (0.85)

4.3 Experiment Results for Eigen Denoising

Now we show our results on the data sets using Eigen denoising procedure. We fix $\delta = 1$ so that l is reduced serially (this is the serialest possible rate of reduction). We demonstrate the usefulness of the stopping rule in Figures 43-44 when we conduct experiments on all the digits 0-9 setting $\sigma \in \{50, 100, 500, 1000\}$; $l \in \{50, 100\}$; $\alpha = 0.1$; $n = 10$; $\delta = 1$. The acronym “ORACLERSS” stands for the true minimum RSS achieved theoretically by our

TABLE 9. Average time in seconds required to denoise a data point from the face image data set. The standard error is reported within parenthesis.

Method	Original	$N_{max} = 50$	$N_{max} = 100$	$N_{max} = 200$	$N_{max} = 500$
Gradient Descent	0.28 (0.01)	25.1 (0.18)	49.9 (0.1)	99.5 (0.3)	250.5 (0.64)
KMDS	0.34 (0.01)	16.8 (0.08)	33.0 (0.2)	67.8 (0.2)	165.3 (0.23)

method; the acronym "STOPRULERSS" stands for the RSS achieved practically using the stopping rule; the acronym "KMDSdeduct" refers to the fact that the KMDS method is evaluated at the number of eigenvalues at which the Eigen Denoising stopped (in other words, deducting an amount ($\delta \times iteration$) from the initial number of eigenvalues chosen where $iteration$ is the iteration number when Eigen Denoising stops).

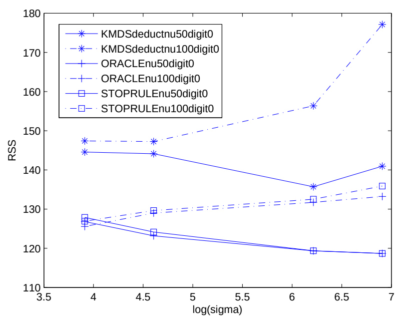
We did some more experiments on digits 7 and 9 by setting $n = 25$ and rest other parameters as described previously. The results shown in Figure 45 are similar to the results obtained in the previous figures.

We also conducted some experiments on digits 4, 6, and 8 using the gradient descent algorithm. We begin with 200 eigenvectors initially and decrease it by one at each iteration. Figure 46 shows the results of our experiments.

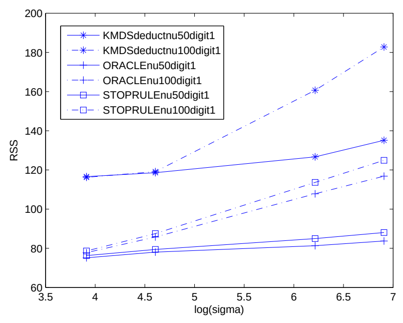
5 Conclusion

A new method to approximate the preimage of a denoised signal is provided that uses a serial approach for estimating preimages.

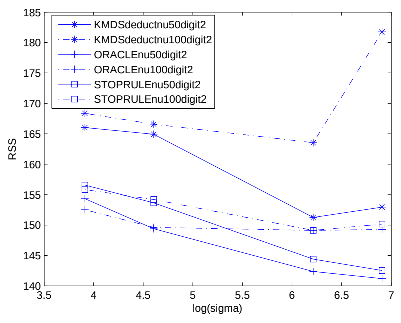
Furthermore, a variant to the above approach is also proposed that gradually discards the irrelevant eigenvectors. We also design and employ a simple stopping rule which ensures that we obtain the final preimage within an acceptable threshold, and the procedure is completed in fewer iterations. Our method improves upon the original methods in the experimental results shown on the datasets here.



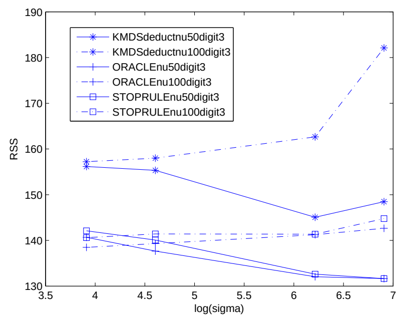
(a) Digit 0



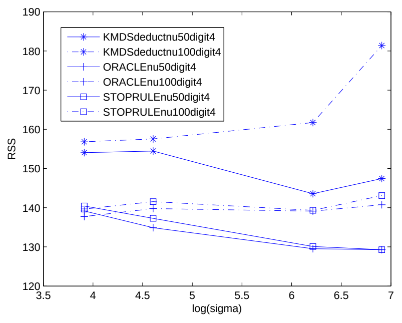
(b) Digit 1



(c) Digit 2

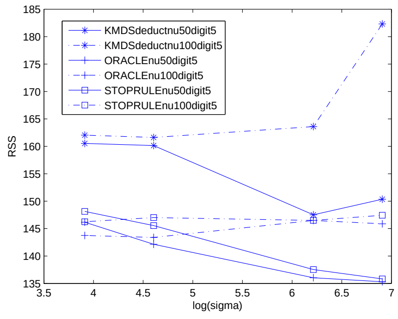


(d) Digit 3

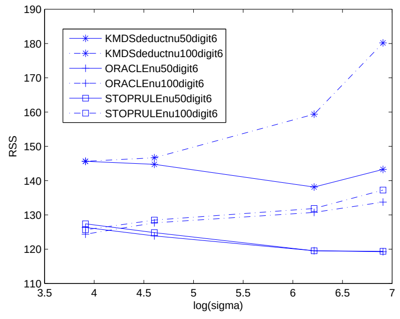


(e) Digit 4

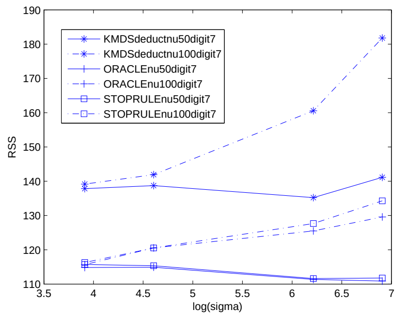
Figure 43. Comparison of RSS values achieved by KMDS method with eigen denoising for digits 0-4 setting $\sigma \in \{50, 100, 500, 1000\}$; $l \in \{50, 100\}$; $\alpha = 0.1$; $n = 10$; $\delta = 1$; Gaussian noise with $\mu = 0$ and $\sigma_G = 1$



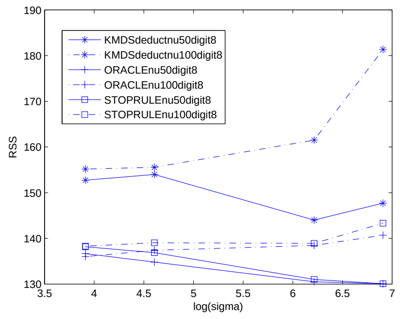
(a) Digit 5



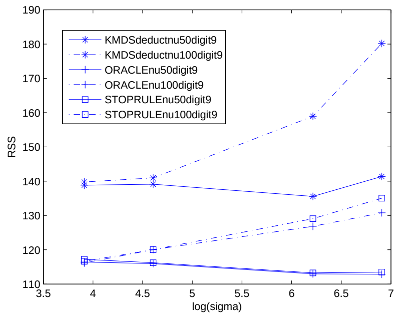
(b) Digit 6



(c) Digit 7

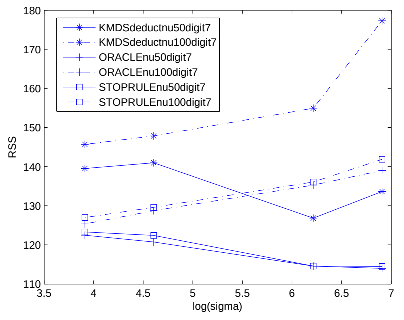


(d) Digit 8

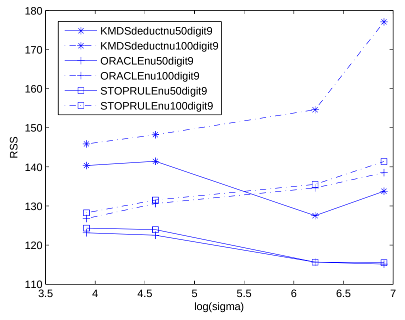


(e) Digit 9

Figure 44. Comparison of RSS values achieved by KMDS method with eigen denoising for digits 5-9 setting $\sigma \in \{50, 100, 500, 1000\}$; $l \in \{50, 100\}$; $\alpha = 0.1$; $n = 10$; $\delta = 1$; Gaussian noise with $\mu = 0$ and $\sigma_G = 1$

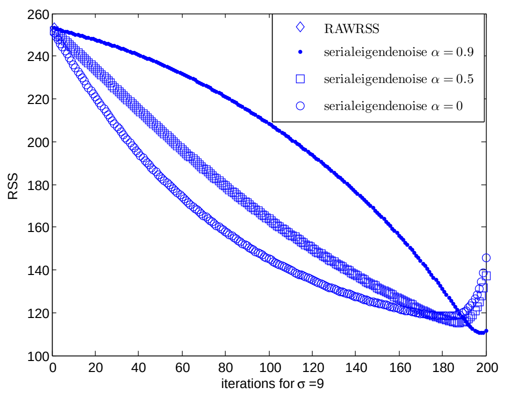


(a) Digit 7

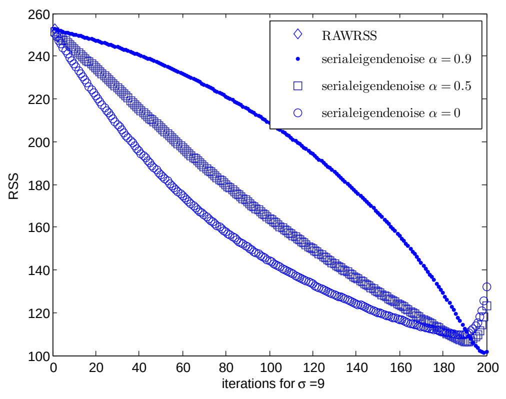


(b) Digit 9

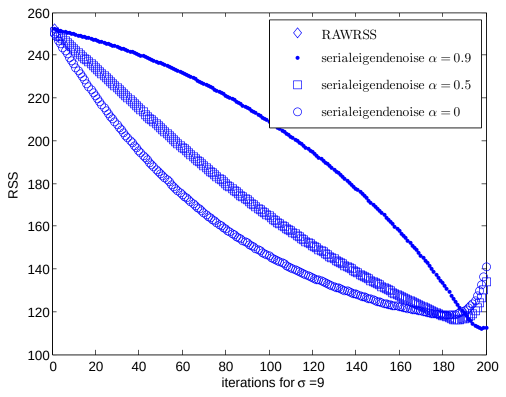
Figure 45. Comparison of RSS values achieved by KMDS method with eigen denoising for digits 7 and 9 setting $\sigma \in \{50, 100, 500, 1000\}$; $l \in \{50, 100\}$; $\alpha = 0.1$; $n = 25$; $\delta = 1$; Gaussian noise with $\mu = 0$ and $\sigma_G = 1$



(a) Digit 4



(b) Digit 6



(c) Digit 8

Figure 46. Comparison of RSS values achieved by gradient descent method with serial denoising for digits 4, 6, and 8 setting $\delta = 1$; Gaussian noise with $\mu = 0$ and $\sigma_G = 1$

CHAPTER 5

FEATURE SELECTION FOR KERNEL PRINCIPAL COMPONENT ANALYSIS

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. 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 chapter.

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) developed by [22] extends PCA to the case where data contain non-linear patterns. 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 ([1]). Thus, only calculations in terms of dot products in the input space are required, without an explicit mapping to the feature space.

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 ([16], [11], [30]). A meta-method to improve the preimage results through bagging was considered by [23]. A sequential procedure to obtain preimage was developed by [21].

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 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. [2] provided a weighted feature approach 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 ([14] and [13]) 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, [14] 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.

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. [29] proposed a method for feature selection in Local Learning-Based Clustering [27] by regularizing the weights assigned to features. [17] dealt with measuring variable importance in KPCA. 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).

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 alternate 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 chapter 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 datasets. We also compare the results of our approach to the results obtained from the methodology described by [17]. Finally Section 5 provides conclusions.

2 Background on Preimages in KPCA

KPCA is equivalent to PCA in feature space ([22]). Let $\mathbf{X}$ denote the data set with N instances and F features where the instances are denoted by $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$. 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 $\varphi(\mathbf{x}_i), i = 1, 2, \cdots, N$ are assumed to be centered, $\mathbf{C}$ can be calculated by

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N} \varphi(\mathbf{x}_i)\varphi(\mathbf{x}_i)' \tag{5.1}$$

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

$$\mathbf{C}\mathbf{v} = \lambda\mathbf{v} \tag{5.2}$$

It can be shown that an eigenvector corresponding to non-zero eigenvalue of $\mathbf{C}$ can be written as a linear combination of $\varphi(\mathbf{x}_1), \cdots, \varphi(\mathbf{x}_N)$. Using this simplification reduces the original problem of finding eigenvalues and eigenvectors of $\mathbf{C}$ to finding the corresponding eigenvalues and eigenvectors of the kernel matrix $\mathbf{K}$ with entries

$$\mathbf{K}_{ij} := (\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)) \tag{5.3}$$

The product $\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$ is evaluated using the kernel trick [1] without explicitly computing the mapping $\varphi(.)$.

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 [1]. The second step is to calculate the principal component directions of the training data in feature space as shown in [22]. 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 Figure 47.

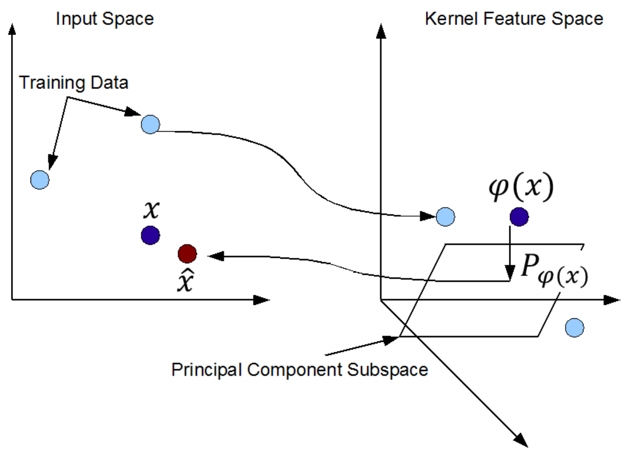


Figure 47. KPCA and the preimage problem. Training data are transformed to feature space and used to learn a principal component plane. A test point x is transformed and projected to the plane as $P\varphi(x)$. The inverse transform of $P\varphi(x)$ may not exist, and an approximate preimage $\hat{x}$ is computed.

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 [16]. Hence, several algorithms have been proposed to estimate the preimage. [16] 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. [11] 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. [11] chose only a few nearest training data points in order to reduce the computational burden. [4] 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 [11] and [4] favor noise-free training data.

3 Feature Selection Using Sparse Random Vectors with Matched Pairs

The main idea of our approach is to understand the contribution of a feature towards the variation pattern in the data. When we project onto a small subset of features at a time using sparse random projections, we essentially try to capture the effect of that subset of features in feature space. 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. Matched pairs of projections are created for each feature to estimate the effect of the feature on the variation pattern. We calculate the difference in the preimage as a result of excluding the feature. Thus, important features are expected to result in high differences.

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 $\mathbf{K}$ be the kernel matrix obtained from instances in the input space. Let $\mathbf{x}_i$ and $\mathbf{x}_j$ denote two instances in input space. Assume that we are using a Gaussian kernel. The ij^{th} entry in $\mathbf{K}$ is calculated as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|_F^2}{\sigma} \right). \tag{5.4}$$

For the purpose of our feature selection procedure, we modify $\mathbf{K}$ to $\mathbf{K}_w$ where we obtain the corresponding ij^{th} entry in $\mathbf{K}_w$ as

$$k_w(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-(\mathbf{w}^T\mathbf{x}_i - \mathbf{w}^T\mathbf{x}_j)^2}{\sigma}\right). \tag{5.5}$$

We also normalized $\mathbf{w}$ to unit length in equation 5.5. Preliminary experiments, however, didn't show meaningful differences in results obtained from normalized and nonnormalized $\mathbf{w}$.

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

$$\mathbf{w}^*(f) = \begin{cases} 0 & \text{if } \mathbf{w}(f) \neq 0 \\ 1 & \text{otherwise} \end{cases} \tag{5.6}$$

Thus, for every feature f at each iteration b, we generate matched pairs $\mathbf{w}$ and $\mathbf{w}^*$ which differ only at the f^{th} entry. We use $\mathbf{w}$ to obtain $\mathbf{K}_w$ as shown in the previous subsection and then use $\mathbf{K}_w$ and $\mathbf{X}$ in the preimage algorithm to obtain $\hat{\mathbf{X}}_b$ at iteration b. Similarly, we use $\mathbf{w}^*$ to obtain $\mathbf{K}_w^*$ and then use $\mathbf{K}_w^*$ along with $\mathbf{X}$ to obtain $\hat{\mathbf{X}}_b(f)$ at iteration b.

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

$$imp_f = \sum_{b=1}^{B} \frac{||\hat{\mathbf{X}}_b - \hat{\mathbf{X}}_b(f)||_F}{B} \tag{5.7}$$

where the Frobenious norm of the matrix is used. We summarize the above procedure in Algorithm 3. In Algorithm 3 $g(.)$ denotes a preimage estimation function. Note that the function $g(.)$ takes $\mathbf{K}_w$ (or $\mathbf{K}_w^*$) and $\mathbf{X}$ as input, and outputs $\hat{\mathbf{X}}_b$ (or $\hat{\mathbf{X}}_b(f)$) at iteration b for feature f.

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. This is important in our case

Algorithm 3 Feature Selection Algorithm

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

because we don't have noise-free training data for our algorithm. This enables us to work with any preimage estimation algorithm for KPCA in the literature.

4 Experimental Results

To evaluate our method, we generate several simulated data sets. 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. For actual data, relevant features are not usually known. Consequently, we use simulated data to construct such features. Our feature selection methodology can work with any KPCA algorithm. For the purpose of this chapter, we use the algorithm proposed in [11] as the base algorithm. The number of leading eigenvectors is chosen according to the criterion proposed by [11]. 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. We also vary the noise level in the data through the standard deviation σ_G of added Gaussian noise.

The first data set is the *Line2* data set which refers to the fact that the pattern is linear only over two features. More specifically, the data set consists of 50 instances and 70 features generated as follows: $x_1 = 0.1t$ for $t = 1, 2, \cdots, 50$, $x_2 = 0.5(1 - x_1)$, and $x_3, x_4, \cdots, x_{70}$ 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. Figure 48 shows the variable importance as a function of the variable index, along with standard error bars obtained by repeating the feature selection procedure 10 times.

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: $x_1 = 0.1t$, $t = 1, 2, \cdots, 50$, 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 $x_6, x_7, \cdots, x_{70}$ 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 Figure 49 (standard errors 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: $x_1 = 0.1t$, $t = 1, 2, \cdots, 50$, x_2 is Gaussian distributed with mean 0 and variance 1, $x_3 = x_2^2/x_1$, and $x_4, x_5, \cdots, x_{70}$ are independent, Gaussian noise with mean 0 and variance σ_G^2. Independent

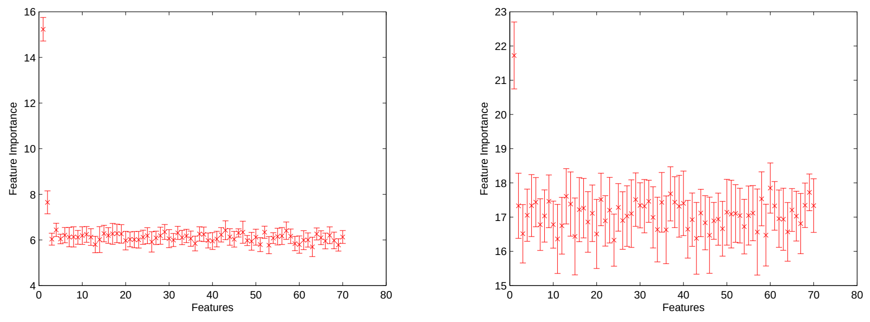
Gaussian noise with $\mu = 0$ and variance σ_G^2 are added to x_1, x_2, x_3. Figure 50 shows the results (standard errors from generating different x_2 10 times).

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 $x_1 = 5\sin(t)\cos(t)$, $x_2 = 5\sin(t)\sin(t)$, $x_3 = 5\cos(t)$, for $t = 1, 2, \cdots, 50$, and $x_4, x_5, \cdots, x_{70}$ 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. Figure 51 shows the results (standard errors from 10 replicates).

We see that for almost 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.

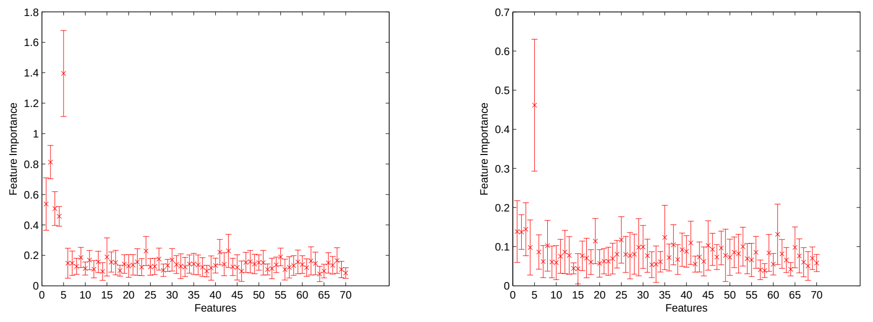
We also conduct some experiments to evaluate the sensitivity of our results to the parameters involved (σ and γ). We conducted experiments on *Line2* and *Sphere3* datasets setting ($\sigma_G \in \{1, 2, 3\}$), $\sigma \in \{0.1, 1, 5, 10, 50\}$ and $\gamma \in \{\frac{2}{\sqrt{70}}, \frac{3}{\sqrt{70}}, \frac{5}{\sqrt{70}}\}$. Figures 52-61 in the appendix show the results. We see that our algorithm is clearly able to detect the important features under small to medium noise levels over a wide range of parameters. However, as noise level increases, the ability to detect these features diminishes.

We show the results from experiments conducted to study the sensitivity of our feature selection algorithm to different parameters.All the plots show the relative importance scores of the relevant features compared to the noise features under several values of noise σ_G. The relevant features are designated as V1, V2, and V3 in case of *Sphere3* dataset, and V1, and V2 in case of *Line2* dataset. The noise features are designated by “others” in all the cases. Furthermore, the mean importance of all noise features is set to zero (baseline), and



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

Figure 48. Feature importance plots for our algorithm applied to the *Line2* data set for selected values of noise σ_G.



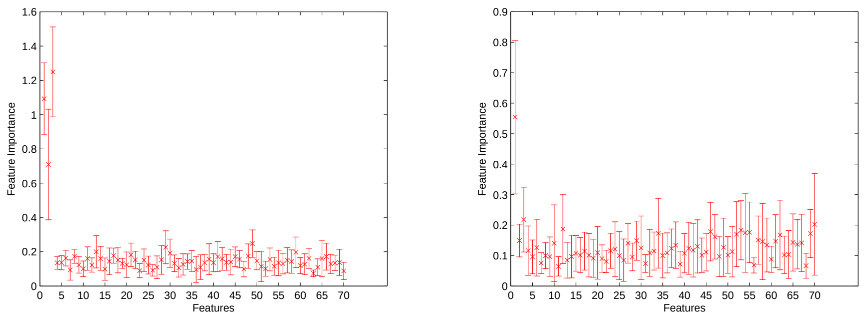
(a) $\sigma_G = 0.9$ (b) $\sigma_G = 3$

Figure 49. Feature importance plots for our algorithm applied to the *Plane5* data set for selected values of noise σ_G.

the relative importance scores of the other features are calculated by subtracting the mean importance of all the noise features.

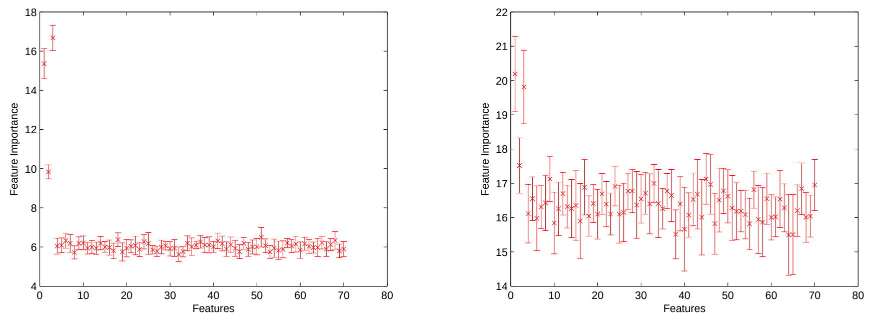
To compare our approach, we tested the algorithm in [17] on the *Line2* and *Sphere3* data sets with $\sigma_G = 0.9$. Figure 62 shows the results. In both cases, it is not able to identify the relevant features.

We also calculated the computational time involved in running our feature selection procedure. We chose $B \in \{50, 100\}$. We used an Intel (R) Core (TM)2 Quad CPU Q6600 computer with 2.4 GHz clock speed and 4.00 GB RAM. For $B = 50$, we found that the



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

Figure 50. Feature importance plots for our algorithm applied to the *Curve3* data set for selected values of noise σ_G.



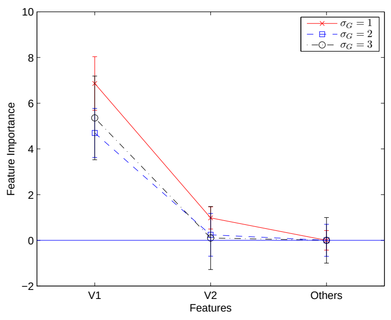
(a) $\sigma_G = 0.9$ (b) $\sigma_G = 3$

Figure 51. Feature importance plots for our algorithm applied to the *Sphere3* dataset for selected values of noise σ_G.

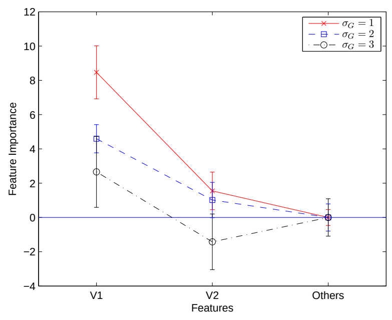
average time was 271.17 seconds with a standard error of 0.249. For $B = 100$, the average time taken was 542.68 seconds with a standard error of 0.23.

5 Conclusion

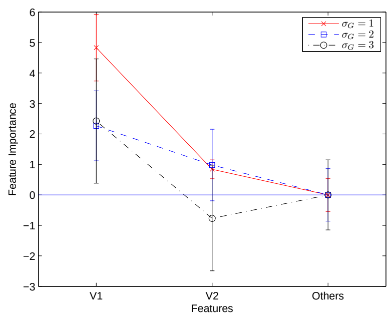
A new feature selection algorithm for KPCA for the case of noisy training data are presented. The data points are projected onto multiple sparse random subsets of features, and then a feature importance measure is calculated by denoising the data matrix using matched pairs of projections (with and without a feature). An advantage of working with an ensemble of feature subsets is they tend to be more robust towards noisy and irrelevant



(a) $\gamma = \frac{2}{\sqrt{70}}, \sigma = 0.1$



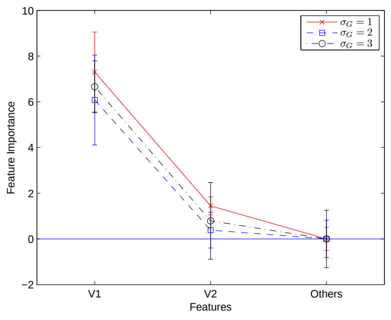
(b) $\gamma = \frac{3}{\sqrt{70}}, \sigma = 0.1$



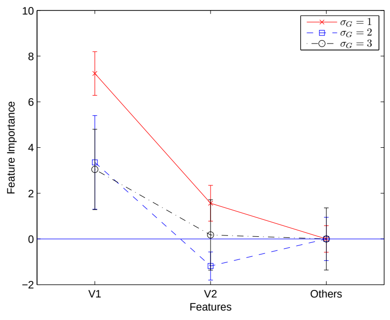
(c) $\gamma = \frac{5}{\sqrt{70}}, \sigma = 0.1$

Figure 52. Feature importance plots to illustrate the sensitivity of our algorithm to sparseness paramter γ with kernel parameter $\sigma = 0.1$ applied to the *Line2* data set for different values of noise σ_G. The mean importance of all noise features is set to zero (baseline), and the relative importance scores of the other features are calculated by subtracting the mean importance of all the noise features.

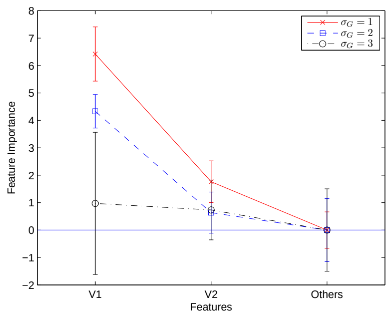
features in the data. Also, our feature selection methodology can used with any suitable KPCA algorithm available in the literature.



(a) $\gamma = \frac{2}{\sqrt{70}}, \sigma = 1$

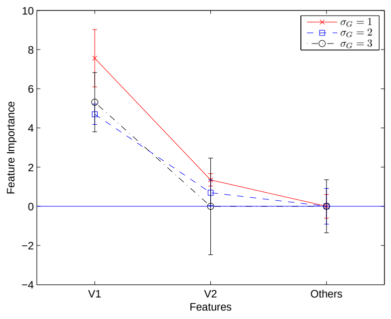


(b) $\gamma = \frac{3}{\sqrt{70}}, \sigma = 1$

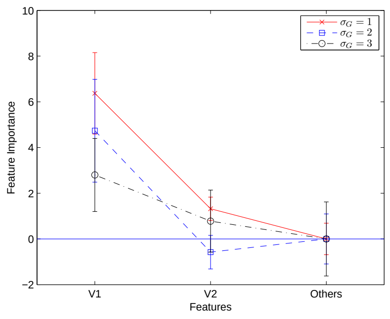


(c) $\gamma = \frac{5}{\sqrt{70}}, \sigma = 1$

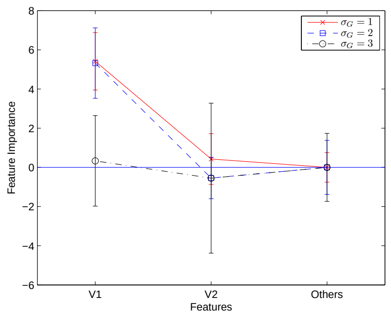
Figure 53. Feature importance plots to illustrate the sensitivity of our algorithm to sparseness paramter γ and kernel parameter $\sigma = 1$ applied to the *Line2* data set for different values of noise σ_G. The mean importance of all noise features is set to zero (baseline), and the relative importance scores of the other features are calculated by subtracting the mean importance of all the noise features.



(a) $\gamma = \frac{2}{\sqrt{70}}, \sigma = 5$

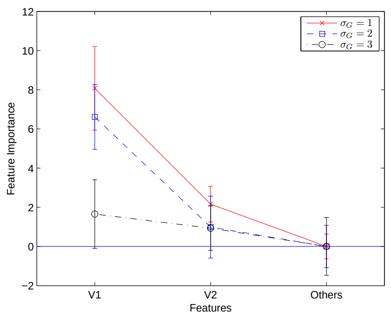


(b) $\gamma = \frac{3}{\sqrt{70}}, \sigma = 5$

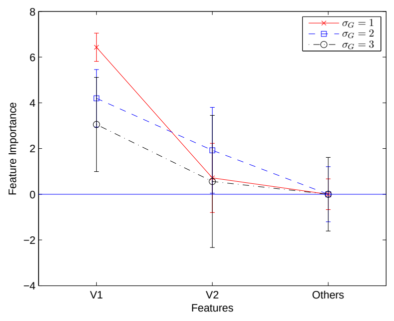


(c) $\gamma = \frac{5}{\sqrt{70}}, \sigma = 5$

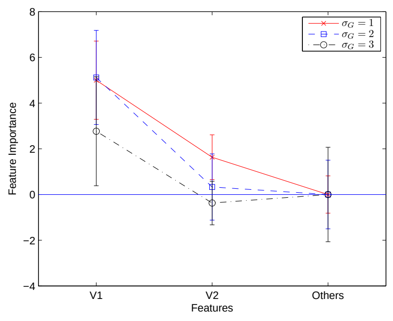
Figure 54. Feature importance plots to illustrate the sensitivity of our algorithm to sparseness paramter γ and kernel parameter $\sigma = 5$ applied to the *Line2* data set for different values of noise σ_G. The mean importance of all noise features is set to zero (baseline), and the relative importance scores of the other features are calculated by subtracting the mean importance of all the noise features.



(a) $\gamma = \frac{2}{\sqrt{70}}, \sigma = 10$

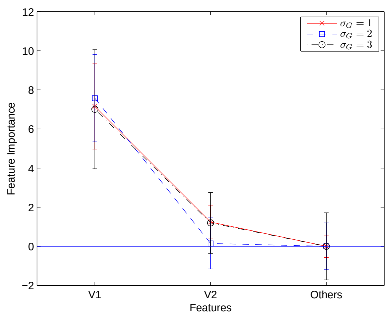


(b) $\gamma = \frac{3}{\sqrt{70}}, \sigma = 10$

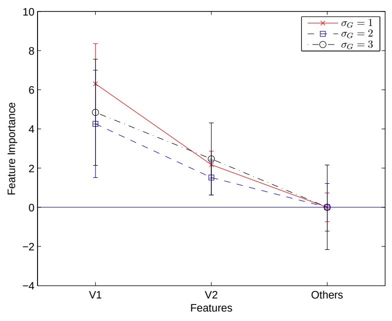


(c) $\gamma = \frac{5}{\sqrt{70}}, \sigma = 10$

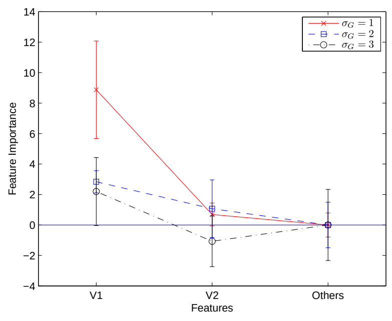
Figure 55. Feature importance plots to illustrate the sensitivity of our algorithm to sparseness paramter γ and kernel parameter $\sigma = 10$ applied to the *Line2* data set for different values of noise σ_G. The mean importance of all noise features is set to zero (baseline), and the relative importance scores of the other features are calculated by subtracting the mean importance of all the noise features.



(a) $\gamma = \frac{2}{\sqrt{70}}, \sigma = 50$

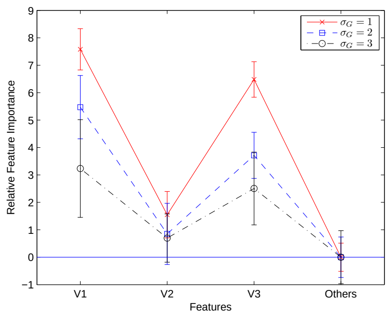


(b) $\gamma = \frac{3}{\sqrt{70}}, \sigma = 50$

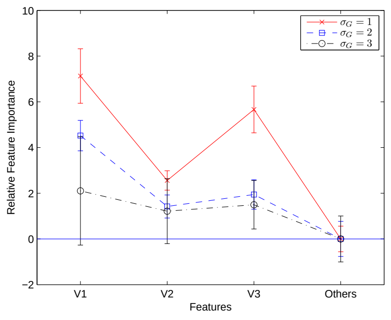


(c) $\gamma = \frac{5}{\sqrt{70}}, \sigma = 50$

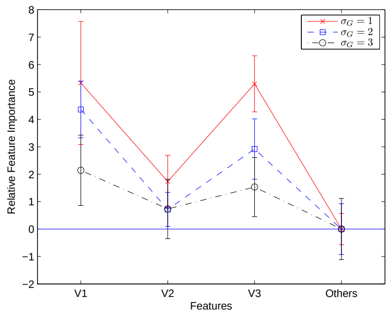
Figure 56. Feature importance plots to illustrate the sensitivity of our algorithm to sparseness paramter γ and kernel parameter $\sigma = 50$ applied to the *Line2* data set for different values of noise σ_G. The mean importance of all noise features is set to zero (baseline), and the relative importance scores of the other features are calculated by subtracting the mean importance of all the noise features.



(a) $\gamma = \frac{2}{\sqrt{70}}, \sigma = 0.1$

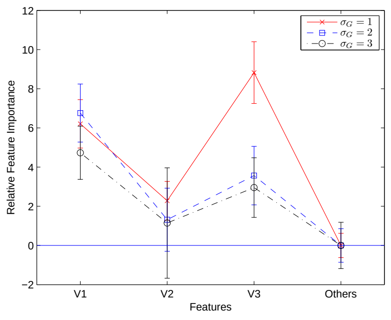


(b) $\gamma = \frac{3}{\sqrt{70}}, \sigma = 0.1$

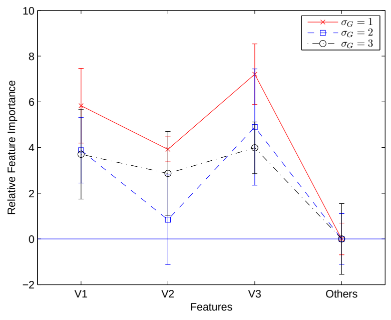


(c) $\gamma = \frac{5}{\sqrt{70}}, \sigma = 0.1$

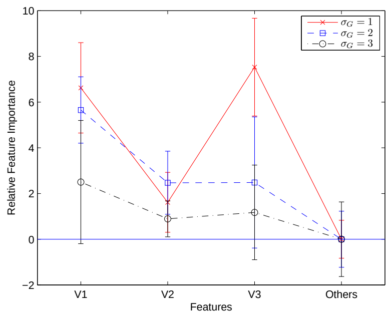
Figure 57. Feature importance plots to illustrate the sensitivity of our algorithm to sparseness paramter γ with kernel parameter $\sigma = 0.1$ applied to the *sphere3* data set for different values of noise σ_G. The mean importance of all noise features is set to zero (baseline), and the relative importance scores of the other features are calculated by subtracting the mean importance of all the noise features.



(a) $\gamma = \frac{2}{\sqrt{70}}, \sigma = 1$

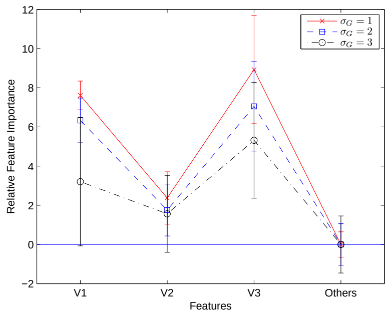


(b) $\gamma = \frac{3}{\sqrt{70}}, \sigma = 1$

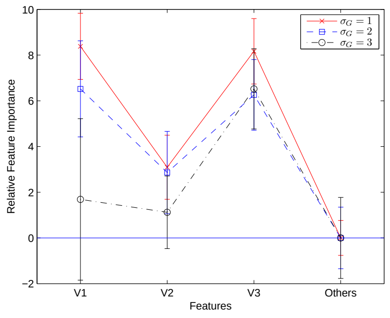


(c) $\gamma = \frac{5}{\sqrt{70}}, \sigma = 1$

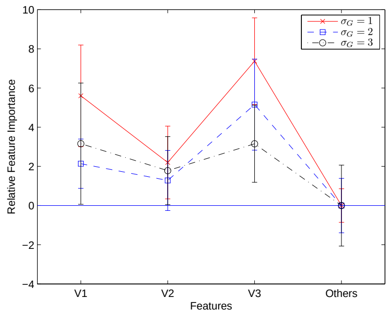
Figure 58. Feature importance plots to illustrate the sensitivity of our algorithm to sparseness paramter γ and kernel parameter $\sigma = 1$ applied to the *sphere3* data set for different values of noise σ_G. The mean importance of all noise features is set to zero (baseline), and the relative importance scores of the other features are calculated by subtracting the mean importance of all the noise features.



(a) $\gamma = \frac{2}{\sqrt{70}}, \sigma = 5$

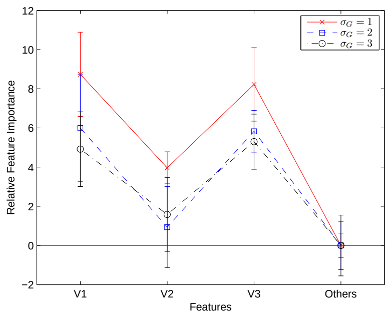


(b) $\gamma = \frac{3}{\sqrt{70}}, \sigma = 5$

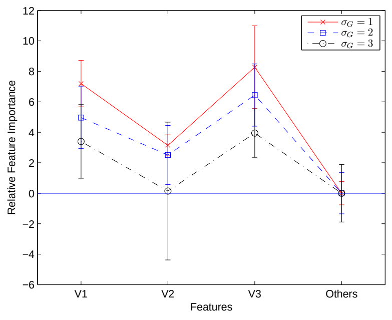


(c) $\gamma = \frac{5}{\sqrt{70}}, \sigma = 5$

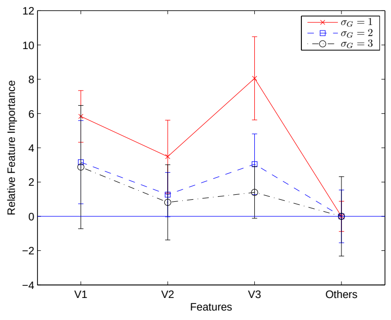
Figure 59. Feature importance plots to illustrate the sensitivity of our algorithm to sparseness paramter γ and kernel parameter $\sigma = 5$ applied to the *sphere3* data set for different values of noise σ_G. The mean importance of all noise features is set to zero (baseline), and the relative importance scores of the other features are calculated by subtracting the mean importance of all the noise features.



(a) $\gamma = \frac{2}{\sqrt{70}}, \sigma = 10$

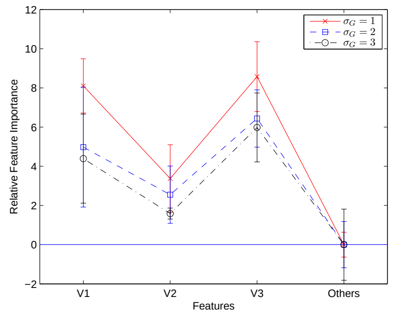


(b) $\gamma = \frac{3}{\sqrt{70}}, \sigma = 10$

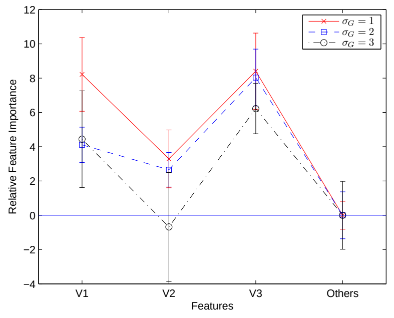


(c) $\gamma = \frac{5}{\sqrt{70}}, \sigma = 10$

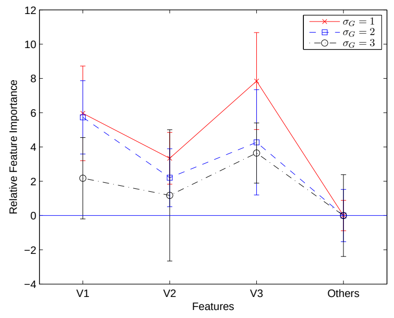
Figure 60. Feature importance plots to illustrate the sensitivity of our algorithm to sparseness paramter γ and kernel parameter $\sigma = 10$ applied to the *sphere3* data set for different values of noise σ_G. The mean importance of all noise features is set to zero (baseline), and the relative importance scores of the other features are calculated by subtracting the mean importance of all the noise features.



(a) $\gamma = \frac{2}{\sqrt{70}}, \sigma = 50$

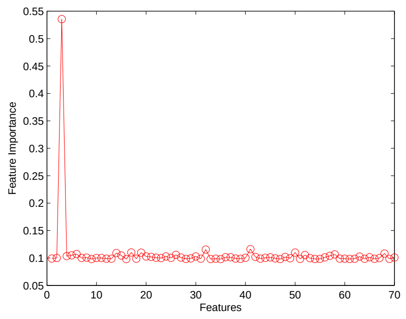


(b) $\gamma = \frac{3}{\sqrt{70}}, \sigma = 50$

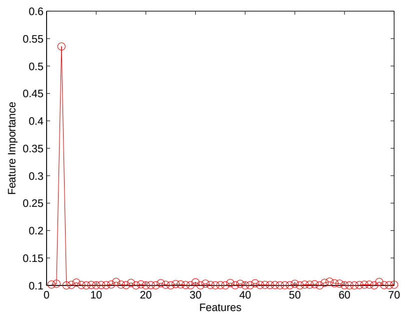


(c) $\gamma = \frac{5}{\sqrt{70}}, \sigma = 50$

Figure 61. Feature importance plots to illustrate the sensitivity of our algorithm to sparseness paramter γ and kernel parameter $\sigma = 50$ applied to the *sphere3* data set for different values of noise σ_G. The mean importance of all noise features is set to zero (baseline), and the relative importance scores of the other features are calculated by subtracting the mean importance of all the noise features.



(a) *Line2* dataset



(b) *Sphere3* dataset

Figure 62. Feature importance plots for the algorithm by [17] for *Line2* and *Sphere3* data sets.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

1 Conclusions

This dissertation proposes meta-approaches to improve upon the estimate of the preimage obtained from using KPCA algorithms in literature. In the first method, we apply a procedure similar to bagging shown in [5] to improve the estimate of the preimage. The PCA plane in feature space might not be well estimated from the training data (especially with noisy training data). Instead of a single estimate of the preimage from one single training dataset, we resample the training set and apply a base KPCA algorithm to each sample. Thus, we estimate the final preimage as the average from bagged samples drawn from the original dataset to attenuate noise in kernel subspace estimation. We expect to improve the estimate from an average over several samples. We also found that the improvement is most pronounced when the parameters differ from those that minimize the error rate. Consequently, our approach improves the robustness of any base KPCA algorithm.

We propose another method to tackle the problem of handling noisy training data. The idea is that the initial estimate of the actual denoised test set obtained by a KPCA preimage estimation algorithm may not be accurate; hence, successive iterations of denoising a convex combination of the test set and the corresponding denoised set can lead us to a more accurate estimate of the actual denoised test set. We also decrease the number of top eigenvectors chosen in each iteration at a constant rate. The intuition is that we initially retain all eigenvectors so as not to loose any information about the pattern in data, and as we approach towards the final denoised preimage, we only retain the top most eigenvectors that will account for the structure in data and get rid of the noise. We also propose a simple and efficient stopping rule criteria to obtain the desirable preimage in fewer number of iterations. Our approach can easily be applied to any KPCA algorithm.

In addition to handling noise in training data, we also need to take care of the fact that there are many irrelevant features collected in the training data. Thus, we need to find the set of features relevant to the pattern in training data. In our third study, we propose a feature selection procedure that augments KPCA to obtain importance estimates of the features given noisy training data. Our feature selection strategy involves projecting the data points onto sparse random vectors. 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. 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. Thus, 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.

2 Future Work

In future, we plan to investigate kernel principal component subspace estimation from noisy training data. We emperically investigated the distance between principal component subspaces learned from bagged samples taken from noisy training data. This served as a measure of difference between subpaces. Fundamentally, the principal component subspace is defined by the set of orthonormal eigenvectors that span it. Thus, it would be interesting to see how the eigenvectors of the principal component subspace change as a result of learning from noisy data points. To understand this analytically, we plan to extend the methods developed by [28] to our problem.

Another interesting aspect would be to understand the effect of input features in estimating kernel principal component subspace. Since the training data might contain a lot of irrelevant features, we can investigate if this adversely affects the eigenvector computation in feature space. Feature selection can then be incorporated into the analysis.

REFERENCES

[1] Aizerman, M., E. Braverman and L. Rozonoer, “Theoretical foundations of the potential function method in pattern recognition learning”, Automation and Remote Control **25**, 821–837 (1964).

[2] Allen, G., “Automatic feature selection via weighted kernels and regularization”, Journal of Computational and Graphical Statistics **22**, 2, 284–299 (2013).

[3] Apley, D. W. and F. Zhang, “Identifying and visualizing nonlinear variation patterns in multivariate manufacturing data”, IIE Transactions **39**, 6, 691–701 (2007).

[4] Bakir, G. H., J. Weston and B. Schölkopf, “Learning to find pre-images”, Advances in neural information processing systems **16**, 7, 449–456 (2004).

[5] Breiman, L., “Bagging predictors”, Machine Learning **24**, 123–140 (1996).

[6] Cox, T. and M. Cox, *Multidimensional Scaling.Monographs on Statistics and Applied Probability 88* (Chapman and Hall/CRC, 2001), second edn.

[7] Hastie, T. and W. Stuetzle, “Principal curves”, Journal of the American Statistical Association **84**, 406, 502–516 (1989).

[8] Im, J. K., D. W. Apley and G. C. Runger, “Tangent hyperplane kernel principal component analysis for denoising”, Neural Networks and Learning Systems, IEEE Transactions on **23**, 4, 644–656 (2012).

[9] Im, J. K., D. W. Apley and G. C. Runger, “Contour gradient kernel principal component analysis for denoising”, Manuscript submitted for publication (2013).

[10] Jolliffe, I., *Principal component analysis* (Wiley Online Library, 2005).

[11] Kwok, I., J.T.Y.and Tsang, “The pre-image problem in kernel methods”, IEEE Transactions on Neural Networks **15**, 1517–1525 (2004).

[12] Malamas, E. N., E. G. M. Petrakis, M. E. Zervakis, L. Petit and J.-D. Legat, “A survey on industrial vision systems, applications, tools”, Image Vision Comput. **21**, 2, 171–188 (2003).

[13] Maldonado, S. and R. Weber, “Feature selection for support vector regression via kernel penalization”, in “IJCNN”, pp. 1–7 (2010).

[14] Maldonado, S., R. Weber and J. Basak, “Simultaneous feature selection and classification using kernel-penalized support vector machines”, Inf. Sci. **181**, 1, 115–128 (2011).

[15] Megahed, F. M., W. H. Woodall and J. A. Camelio, “A review and perspective on control charting with image data”, Journal of Quality Technology **43**, 2, 83–98 (2011).

[16] Mika, S., B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz and G. Rätsch, “Kernel PCA and de-noising in feature spaces”, in “NIPS”, pp. 536–542 (1998).

[17] Muniz, V., J. V. Horebeek and R. Ramos, “Measuring the importance of variables in kernel PCA”, in “COMPSTAT”, pp. 517–524 (2008).

[18] Nguyen, M. H. and F. D. la Torre, “Robust kernel principal component analysis”, in “NIPS”, pp. 1185–1192 (2008).

[19] Sahu, A., D. Apley and G. Runger, “Feature selection for kernel principal component analysis”, Submitted for publication (2012).

[20] Sahu, A., G. Runger and D. Apley, “Image denoising with a multi-phase kernel principal component approach and an ensemble version”, in “IEEE Applied Imagery Pattern Recognition Workshop (AIPR)”, pp. 1–7 (2011).

[21] Sahu, A., G. Runger and D. Apley, “A serial approach to preimage estimation from variation patterns using kernel PCA”, Arizona State University Technical Report (2013).

[22] Schölkopf, B., A. J. Smola and K.-R. Müller, “Nonlinear component analysis as a kernel eigenvalue problem”, Neural Computation **10**, 5, 1299–1319 (1998).

[23] Shinde, A., A. Sahu, D. Apley and G. Runger, “Preimages for variation patterns from kernel PCA and bagging”, IIE Transactions (To Appear).

[24] Wang, L., X. Wang and J. Feng, “Subspace distance analysis with application to adaptive bayesian algorithm for face recognition”, Pattern Recognition **39**, 3, 456–464 (2006).

[25] Wells, L. J., F. M. Megahed, C. B. Niziolek, J. A. Camelio and W. H. Woodall, “Statistical process monitoring approach for high-density point clouds”, Journal of Intelligent Manufacturing pp. 1–13 (2012).

[26] Williams, C. K., “On a connection between kernel PCA and metric multidimensional scaling”, in “Advances in Neural Information Processing Systems 13”, pp. 675–681 (MIT Press, 2001).

[27] Wu, M. and B. Schölkopf, “A local learning approach for clustering”, in “NIPS”, pp. 1529–1536 (2006).

[28] Xu, Z., “Perturbation analysis for subspace decomposition with applications in subspace-based algorithms”, Signal Processing, IEEE Transactions on **50**, 11, 2820–2830 (2002).

[29] Zeng, H. and Y.-m. Cheung, “Feature selection and kernel learning for local learning-based clustering”, Pattern Analysis and Machine Intelligence, IEEE Transactions on **33**, 8, 1532–1547 (2011).

[30] Zheng, W. S., J. Lai and P. C. Yuen, “Penalized preimage learning in kernel principal component analysis”, IEEE Transactions on Neural Networks **21**, 4, 551–570 (2010).