IIE Transactions
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/uiie20

Preimages for Variation Patterns from Kernel PCA and Bagging

Amit Shinde [a], Anshuman Sahu [a], Daniel Apley [b] & George Runger [a]
[a] School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ, 85287, USA
[b] Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 60208, USA
Accepted author version posted online: 25 Oct 2013.

To cite this article: IIE Transactions (2013): Preimages for Variation Patterns from Kernel PCA and Bagging, IIE Transactions

To link to this article: http://dx.doi.org/10.1080/0740817X.2013.849836

Preimages for Variation Patterns from Kernel PCA and Bagging

Amit Shinde, Anshuman Sahu, Daniel Apley, George Runger [‡]

Abstract

Manufacturing industries collect massive amounts of multivariate measurement through automated inspection processes. Noisy measurements and high-dimensional, irrelevant features make it difficult to identify useful patterns in the data. Principal component analysis provides linear summaries of datasets with fewer latent variables. Kernel principal component analysis (KPCA), however, identifies nonlinear patterns. One challenge in KPCA is to inverse map the denoised signal from a high-dimensional feature space into its preimage in input space to visualize the nonlinear variation sources. However, such an inverse map is not always defined. We provide a new meta-method applicable to any KPCA algorithm to approximate the preimage. We improve upon previous work where a strong assumption was availability of noise-free training data. This is problematic for applications like manufacturing variation analysis. To attenuate noise in kernel subspace estimation we estimate the final preimage as the average from bagged samples drawn from the original dataset. 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 demonstrate the usefulness of our method by analyzing a classical hand-written digit dataset and a face dataset. Significant improvement over the existing methods is observed.

[*]A. Shinde, G. Runger, and A. Sahu are in the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ, 85287 USA e-mail: (amit.shinde@asu.edu, runger@asu.edu, anshuman.sahu@asu.edu).

[†]D. Apley is in the Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 60208 USA; email: (apley@northwestern.edu).

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, Malamas et al. (2003) 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. Megahed et al. (2011) discuss about the acquisition of grayscale or binary images for MVS. Our proposed methodology deals with analysis of such images.

More recently, Wells et al. (2012) 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 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 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.

Given a p-dimensional random vector **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 combi-

nation of variables. PCA tends to lose its effectiveness in identifying patterns when the resultant pattern is nonlinear in nature (Schölkopf et al., 1998). One approach to nonlinear PCA is based on principal curves (Hastie and Stuetzle, 1989). Apley and Zhang (2007) used principal curves to identify and visualize nonlinear patterns in data. Schölkopf et al. (1998) 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 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 Aizerman et al. (1964). 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 Mika et al. (1998). 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 Mika et al. (1998) 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 (2004) used the relationship between the feature space distance and the input space distance derived by Williams (2001) 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. (1998). Nguyen and la Torre (2008) extended the KPCA framework to handle noise, outlier and missing data. Zheng et al. (2010) addressed the problem of estimating a better preimage by adding penalty terms to the preimage learning process. Sahu et al. (2012) 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 Breiman (1996). 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 paper 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 existing 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, Schölkopf et al. (1998) 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, $\varphi : \mathbf{R}^p \rightarrow F$.

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 2. 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 $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)' \rightarrow \varphi(\mathbf{x}) = (\mathbf{x}_1^2, \mathbf{x}_2^2, \sqrt{2}\mathbf{x}_1, \sqrt{2}\mathbf{x}_2, \sqrt{2}\mathbf{x}_1\mathbf{x}_2)$. 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 5 for a list of symbols used throughout the paper.

2.1 Nonlinear PCA

We refer to Schölkopf et al. (1998) 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 $\varphi(\mathbf{x}_i)$

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

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

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

The above equation (2) can be equivalently written as

$$\lambda < \varphi(\mathbf{x}_j) \cdot \mathbf{v} > = < \varphi(\mathbf{x}_j) \cdot \mathbf{C}\mathbf{v} > \tag{3}$$

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

$$\mathbf{v} = \sum_{i=1}^{N} \alpha_i \varphi(\mathbf{x}_i) \tag{4}$$

Substitute equation (4) into equation (3), for all $j = 1, \ldots, N$, and simplify by introducing an $N \times N$ matrix $\mathbf{K}$ whose entries are given by

$$\mathbf{K}_{ij} := < \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j) > \tag{5}$$

The final equation can be written in matrix form as

$$N\lambda \mathbf{K}\boldsymbol{\alpha} = \mathbf{K}^2\boldsymbol{\alpha} \tag{6}$$

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 (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 $< \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j) >$. Aizerman et al. (1964) 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

$$k(\mathbf{x}_i, \mathbf{x}_j) =< \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j) >= (< \mathbf{x}_i \cdot \mathbf{x}_j > +1)^s \tag{7}$$

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

$$k(\mathbf{x}_i, \mathbf{x}_j) =< \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j) >= \exp\left(\frac{-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|_F^2}{\sigma}\right) \tag{8}$$

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

$$\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{OK} - \mathbf{KO} + \mathbf{O}\,\mathbf{K}\,\mathbf{O} \tag{9}$$

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

$$\beta_k =< \mathbf{v}^k \cdot \varphi(\mathbf{x}) >= \sum_{i=1}^{N} \alpha_i^k < \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}) >= \sum_{i=1}^{N} \alpha_i^k k(\mathbf{x}, \mathbf{x}_i) \tag{10}$$

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

$$P_l\varphi(\mathbf{x}) = \sum_{k=1}^{l} \beta_k \mathbf{v}^k \tag{11}$$

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 Mika et al. (1998). 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 3.

4 Existing Methods for Finding Preimages

A brief overview of the more popular methods to obtain preimages is provided. Mika et al. (1998) proposed to estimate the preimage of $P_l\varphi(\mathbf{x})$ by minimizing the squared distance

$$\begin{aligned}\rho(\hat{\mathbf{x}}) &= \|\varphi(\hat{\mathbf{x}}) - P_l\varphi(\mathbf{x})\|^2 \\ &= \|\varphi(\hat{\mathbf{x}})\|^2 - 2P_l\varphi(\mathbf{x})'\varphi(\hat{\mathbf{x}}) + \Omega \end{aligned} \quad (12)$$

where Ω represents all terms independent of $\hat{\mathbf{x}}$. Equation (12) is minimized using standard gradient descent. An extremum can be obtained by setting the derivative of Equation (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 (Kwok, 2004).

In another approach, Kwok (2004) 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 (Williams, 2001). 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. Kwok (2004) then proposed using multi-dimensional scaling (MDS) (Cox and Cox, 2001) 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, Kwok (2004) proposed using n-nearest neighbors to reduce computational time.

More recently Zheng et al. (2010) 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 (Breiman, 1996). 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

$$\hat{\mathbf{X}} = \sum_{b=1}^{B} \frac{\hat{\mathbf{X}}(b)}{B} \tag{13}$$

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 σ 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 Gaussian kernel with kernel parameter σ)
6: Transform i^{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^{th} point in D_{test}.
8: Estimate the denoised data point for the i^{th} test point from its n nearest neighbors in D_b.
9: **end for**
10: Estimate the final denoised point for the i^{th} test point as the average across B denoised 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 Wang et al. (2006) to measure difference between two subspaces. Specifically, the subspace distance d_{SU} between two subspaces is calculated as

$$d_{SU} = \sqrt{l - \sum_{i=1}^{l}\sum_{j=1}^{l}(\mathbf{u}_i'\mathbf{v}_j)^2} \tag{14}$$

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. Wang et al. (2006) 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 4. 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

$$RSS = \sqrt{\sum_{i=1}^{N} (\hat{\mathbf{x}}_i - \mathbf{t}_i)^2}. \quad (15)$$

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 denoising. 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 Mika et al. (1998), 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 Mika et al. (1998) for our experiments. Additionally, we conducted some experiments shown in Figures 5 through 12 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 Mika et al. (1998) and one suggested by Kwok (2004).

Zheng et al. (2010) recommended that the value of l be chosen to preserve 95% of the energy of the training data. For Figures 17 through 20, 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 Kwok (2004), the number of nearest neighbors n is set to 10. Zheng et al. (2010) 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 17 through 20, 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 27.

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 4. 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 4 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 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 5 and Figure 6 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 5, and digits 5 through 9 are shown in Figure 6.

To further explore parameter settings in the high-noise case, Figures 7 through 9 consider digits 7 and 9 only. Figure 7 shows how the RSS varies for $n = 10$ with Gaussian noise $\mu = 0$ and $\sigma_G = 1$. Figure 8 shows how the RSS varies for $n = 25$ with Gaussian noise $\mu = 0$ and $\sigma_G = 0.5$. Figure 9 shows how the RSS varies for $n = 25$ with Gaussian noise $\mu = 0$ and $\sigma_G = 1$.

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.

For the low-noise case, we also report the RSS values for digits 7 and 9 for different parameter and noise settings. Figure 10 shows how the RSS varies for $n = 25$ with Gaussian noise $\mu = 0$ and $\sigma_G = 0.5$. Figure 11 and Figure 12 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 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.

We also applied our algorithm on the gradient descent method (Mika et al., 1998) and on the penalized preimage approach (Zheng et al., 2010) for the digits data in the high-noise case. Figures 13 and 14 show the results for the gradient descent approach. Figures 15 and 16 show the results for the penalized preimage approach approach. In both the cases, BKPCA reduces the RSS.

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 Zheng et al. (2010) to our BKPCA approach with values of $B \in \{50, 100, 200, 500\}$. The results are shown in Figures 17 through 18. 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 Mika et al. (1998), 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 Mika et al. (1998) to our BKPCA approach with values of $B \in \{50, 100, 200, 500\}$. The results are shown in Figures 19 through 20. We see that our BKPCA method performs significantly better than the gradient descent approach by (Mika et al., 1998) 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 $2p$ 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 papers, 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 21 to 22 show the results for the penalized preimage algorithm. Figures 23 to 24 show the results for the gradient descent algorithm. Figures 25 to 26 show the results for KMDS algorithm. The BKPCA algorithm consistently shows better performance for RSS.

For the plots shown in Figures 17 through 20, we compute the difference between RSS obtained through the original method and our BKPCA method for each of the 300 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 17 through 20, 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 27 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 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 31

Boxplots of RSS are shown for all images for the three denoising algorithms with and without

BKPCA applied. Figure 28 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 29 shows the results for gradient descent algorithm. Figure 30 shows the results for KMDS algorithm.

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

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 31. 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).

Acknowledgements: This material is based upon work supported by the National Science Foundation under Grant No. 0825331.

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.

References

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

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

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24:123–140. 10.1007/BF00058655.

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

Hastie, T. and Stuetzle, W. (1989). Principal curves. *Journal of the American Statistical Association*, 84:502–516.

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

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

Megahed, F. M., Woodall, W. H., and Camelio, J. A. (2011). A review and perspective on control charting with image data. *Journal of quality technology*, 43(2):83–98.

Mika, S., Schölkopf, B., Smola, A. J., Müller, K.-R., Scholz, M., and Rätsch, G. (1998). Kernel pca and de-noising in feature spaces. In *NIPS*, pages 536–542.

Nguyen, M. H. and la Torre, F. D. (2008). Robust kernel principal component analysis. In *NIPS*, pages 1185–1192.

Sahu, A., Apley, D., and Runger, G. (2012). Feature selection for kernel principal component analysis. *Submitted for publication*.

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

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

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

Williams, C. K. (2001). On a connection between kernel pca and metric multidimensional scaling. In *Advances in Neural Information Processing Systems 13*, pages 675–681. MIT Press.

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

Amit Shinde is a Senior Operations Research Analyst at FedEx Express. He received his Ph.D. and M.S. degrees in Industrial Engineering from Arizona State University and B.E in Mechanical Engineering from the University of Mumbai. His research focuses on the development and application of hybrid simulation and data mining models for manufacturing, logistics and supply chain analytics.

Anshuman Sahu is a Ph.D. candidate in Department of Industrial Engineering in the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University. He received his B.Tech in Production and Industrial Engineering from Motilal Nehru National Institute of Technology, India. His research interests are developing and applying statistical and data mining techniques to solve practical problems in different domains such as manufacturing, e-commerce, and building energy consumption.

Daniel W. Apley is a Professor of Industrial Engineering and Management Sciences at Northwestern University, Evanston, IL. He obtained B.S., M.S., and Ph.D. degrees in Mechanical Engineering and an M.S. degree in Electrical Engineering from the University of Michigan. His research interests lie at the interface of engineering modeling, statistical analysis, and data mining, with particular emphasis on manufacturing variation reduction applications in which very large amounts of data are available. His research has been supported by numerous industries and government agencies. He received the NSF CAREER award in 2001, the IIE Transactions Best Paper Award in 2003, and the Wilcoxon Prize for best practical application paper appearing in Technometrics in 2008. He has served as Editor-in-Chief for the Journal of Quality Technology, Chair of the Quality, Statistics & Reliability Section of INFORMS, Director of the Manufacturing and Design Engineering Program at Northwestern, and Associate Editor for Technometrics.

George C. Runger, Ph.D., is a Professor in the School of Computing, Informatics and Decision

Systems Engineering and interim chair of the Department of Biomedical Informatics at Arizona State University. His research is on real-time monitoring and other data-analysis methods with a focus on large, complex data sets with applications in areas such as manufacturing and health care. He holds degrees in industrial engineering and statistics. In addition to academic work, he was a senior engineer at IBM.

Table 1: Eigenvalues of covariance matrix in input space.

Eigenvalue	12.979	11.952
Proportion	0.521	0.479
Cumulative	0.521	1.000

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

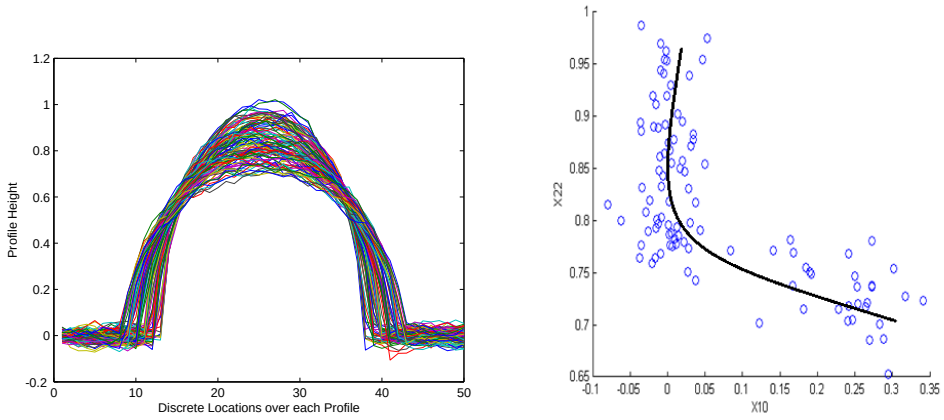
Eigenvalue	168.14	144.79	25.95	23.50	0.000
Proportion	0.464	0.400	0.072	0.065	0.000
Cumulative	0.464	0.864	0.935	1.000	1.000

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

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 $\rho(\hat{\mathbf{x}})$

Table 5: List of symbols used in the paper



(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)

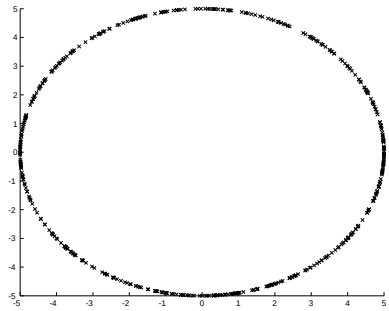


Figure 2: Scatter plot of input space variables

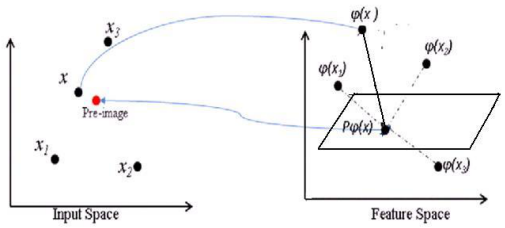


Figure 3: A test point x is transformed to feature space as $\phi(x)$ and projected to the PCA plane as $P\phi(x)$. The preimage approximates the reverse transform of $P\phi(x)$ to the original space.

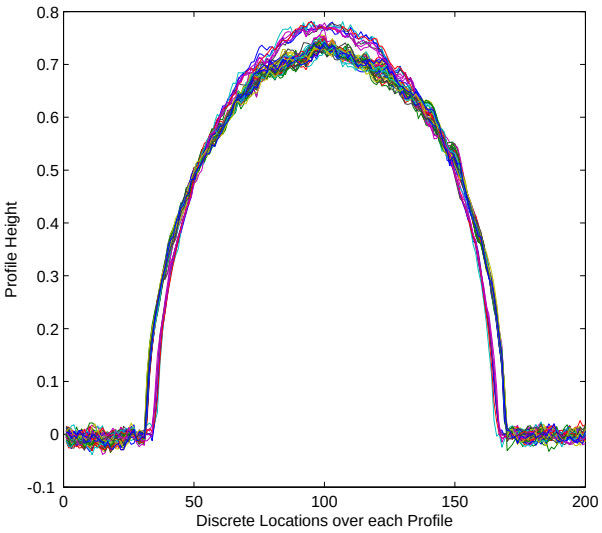
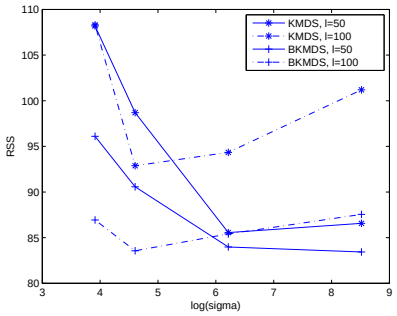
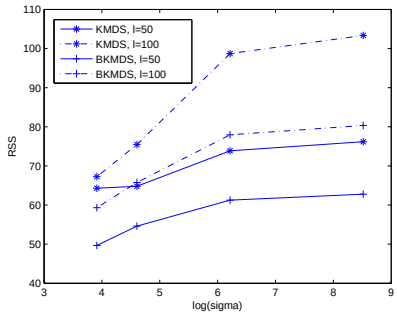


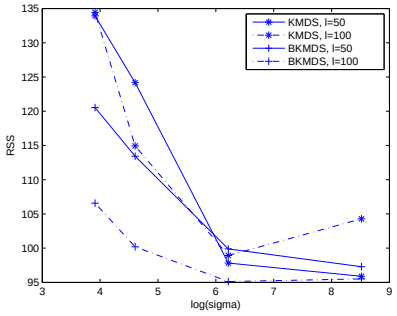
Figure 4: Plot showing variation in preimages for a profile learned from each of the $B = 50$ bootstrap samples from the gasket data. Each profile has been discretized into 200 points.



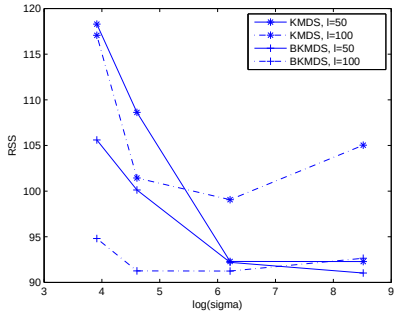
(a) Digit 0



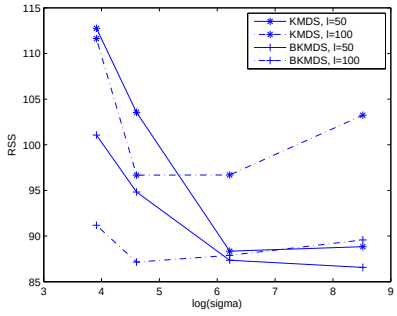
(b) Digit 1



(c) Digit 2

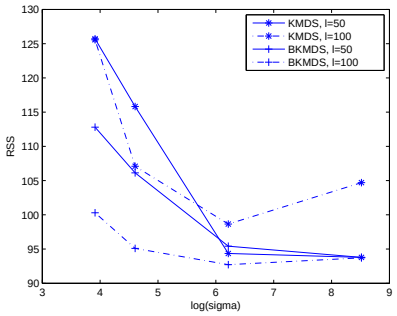


(d) Digit 3

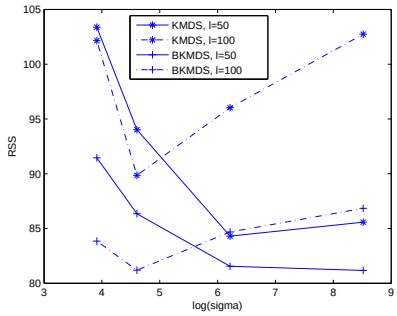


(e) Digit 4

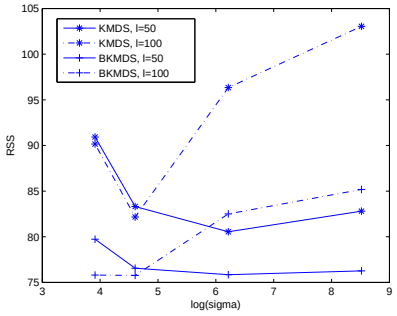
Figure 5: RSS values of KMDS versus bagging (denoted as BKMDS) for digits 0-4 for the high-noise case with the number of leading eigenvectors $l \in \{50, 100\}$, 10 nearest neighbors and noise $\sigma_G = 0.5$. The RSS is shown for different parameter settings of σ where the horizontal axis scale is $\log(\sigma)$.



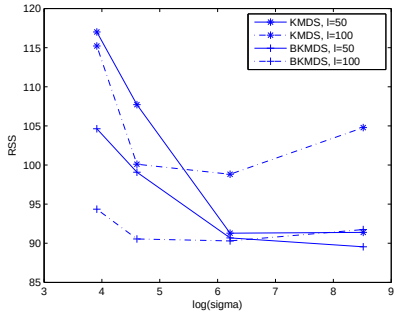
(a) Digit 5



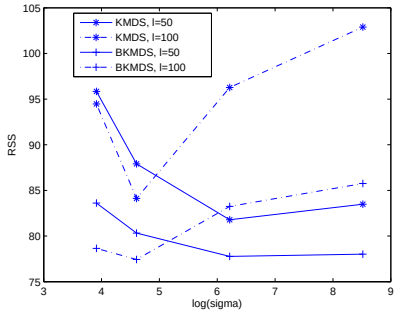
(b) Digit 6



(c) Digit 7

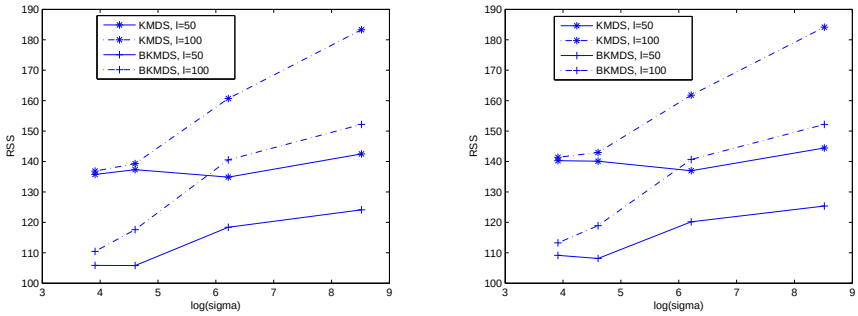


(d) Digit 8



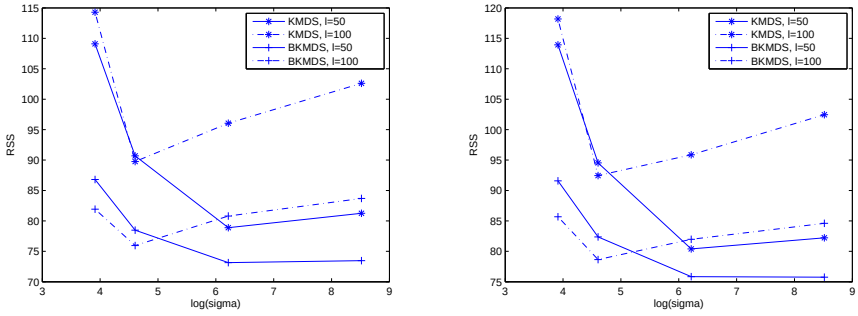
(e) Digit 9

Figure 6: 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 $\log(\sigma)$.



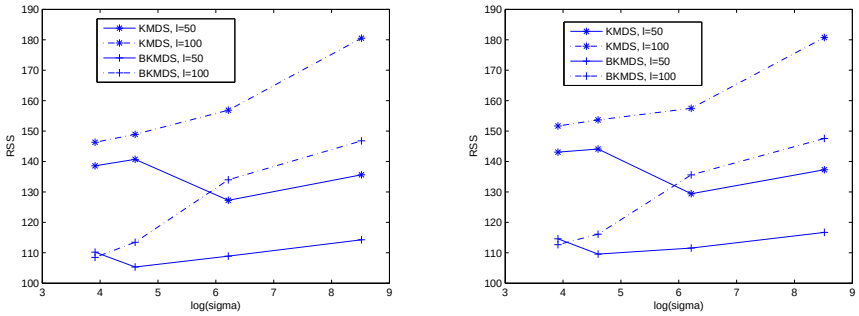
(a) Digit 7 (b) Digit 9

Figure 7: 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 $\log(\sigma)$.



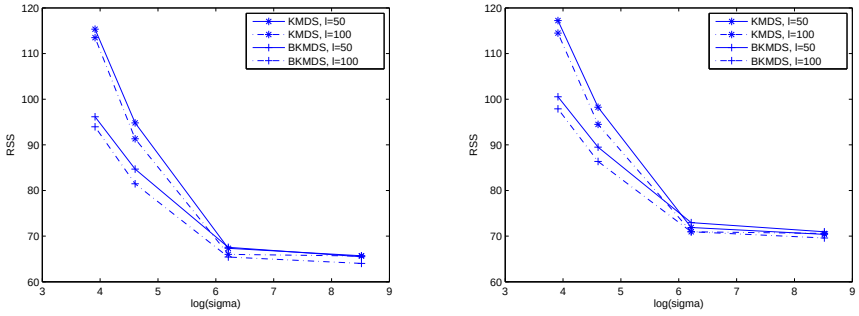
(a) Digit 7 (b) Digit 9

Figure 8: 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 $\log(\sigma)$.



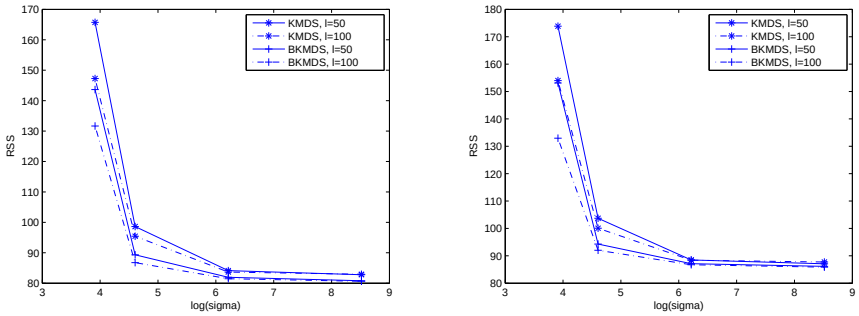
(a) Digit 7 (b) Digit 9

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 $l \in \{50, 100\}$, 25 nearest neighbors, noise $\sigma_G = 1$. The RSS is shown for different parameter settings of σ where the horizontal axis scale is $\log(\sigma)$.



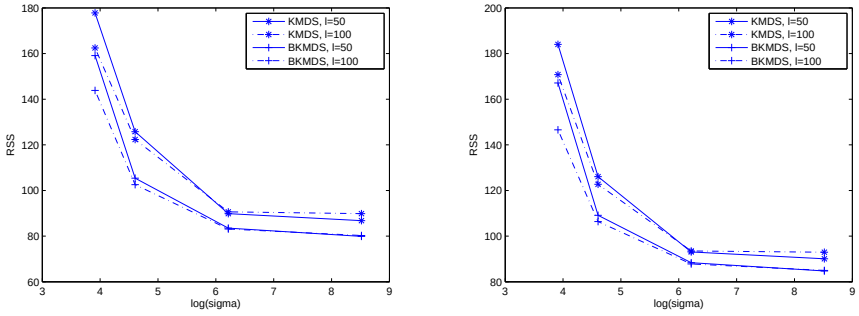
(a) Digit 7 (b) Digit 9

Figure 10: 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 $\log(\sigma)$.



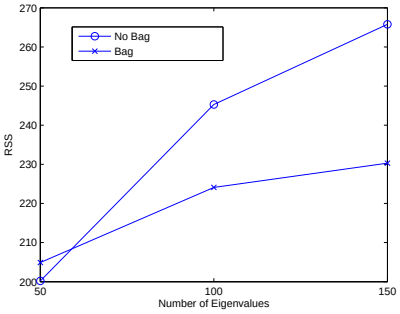
(a) Digit 7 (b) Digit 9

Figure 11: 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 $\log(\sigma)$.

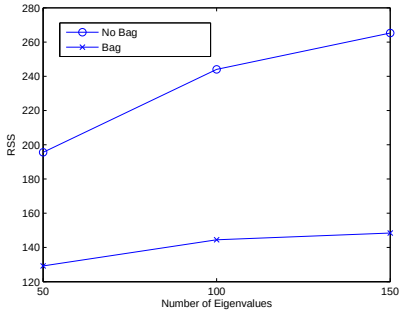


(a) Digit 7 (b) Digit 9

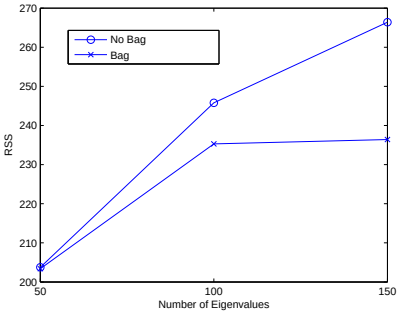
Figure 12: RSS values of KMDS versus bagging (denoted as BKMDS) for digits 7 and 9 for the low-noise case with the number of leading eigenvectors $l \in \{50, 100\}$, 25 nearest neighbors, noise $\sigma_G = 1$. The RSS is shown for different parameter settings of σ where the horizontal axis scale is $\log(\sigma)$.



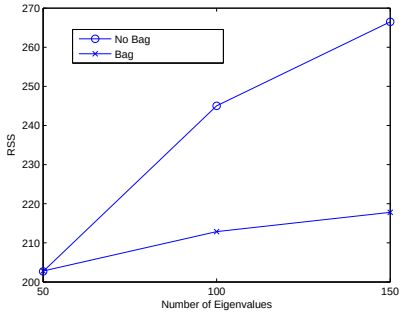
(a) Digit 0



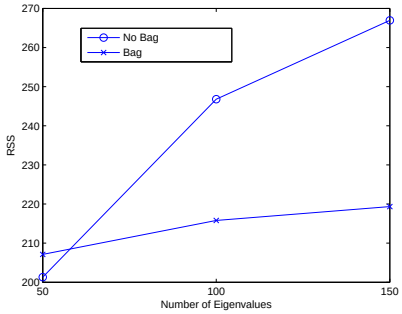
(b) Digit 1



(c) Digit 2

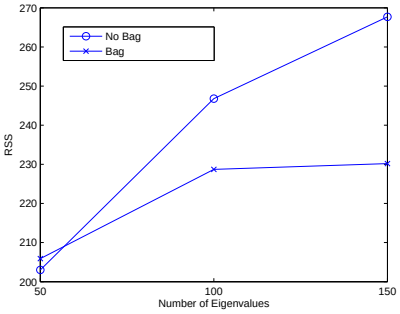


(d) Digit 3

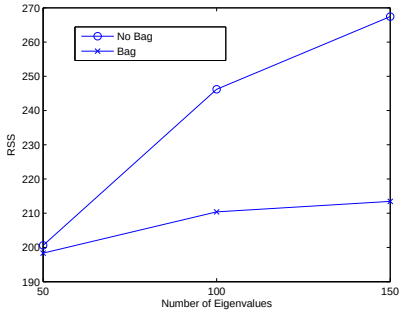


(e) Digit 4

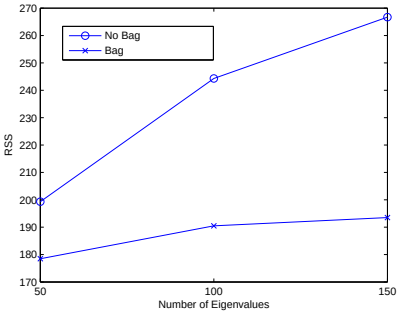
Figure 13: RSS values of the gradient descent approach versus bagged approach for digits 0-4 for the high-noise case with noise $\sigma_G = 1$.



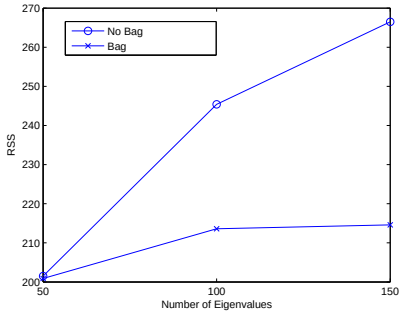
(a) Digit 5



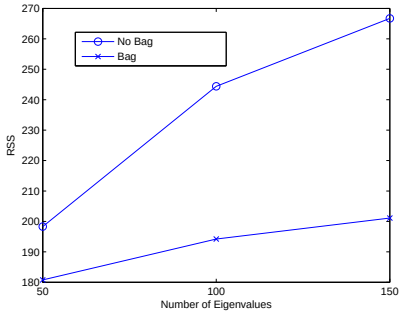
(b) Digit 6



(c) Digit 7

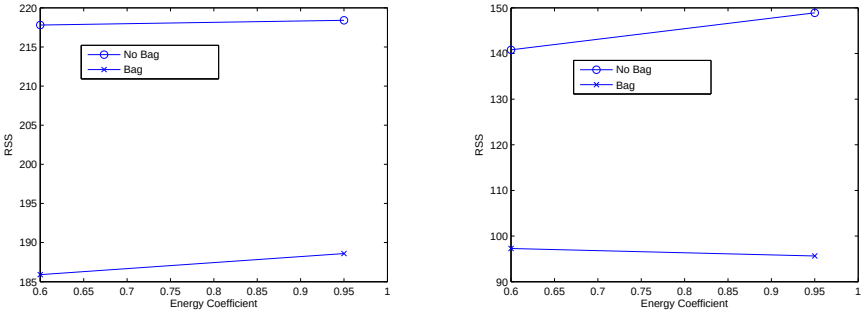


(d) Digit 8



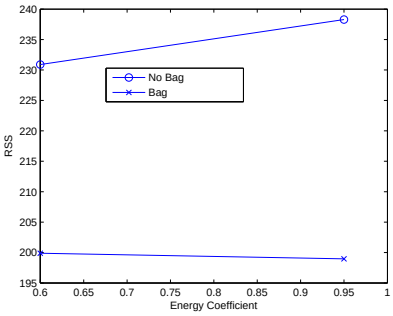
(e) Digit 9

Figure 14: RSS values of the gradient descent approach versus bagged approach for digits 5-9 for the high-noise case with noise $\sigma_G = 1$.

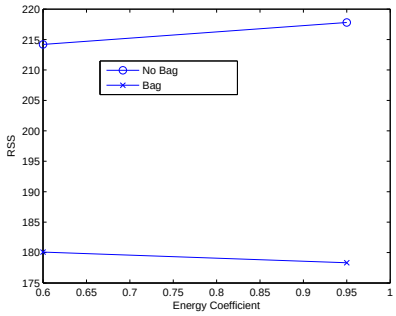


(a) Digit 0

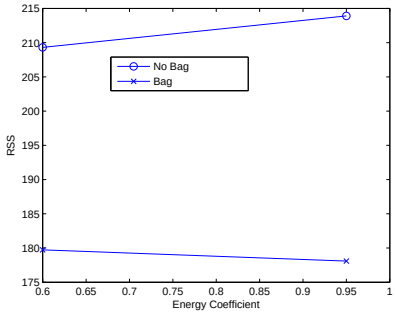
(b) Digit 1



(c) Digit 2

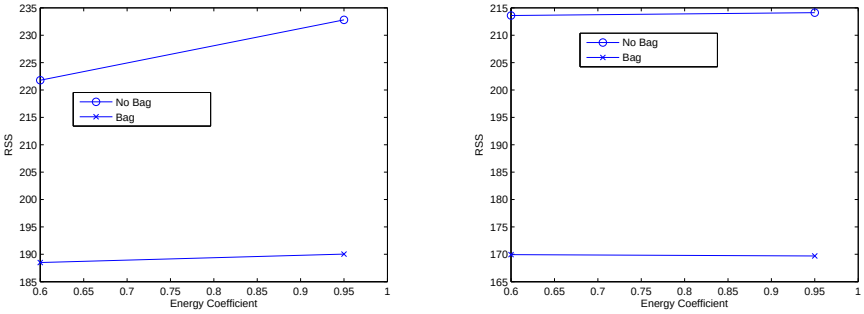


(d) Digit 3



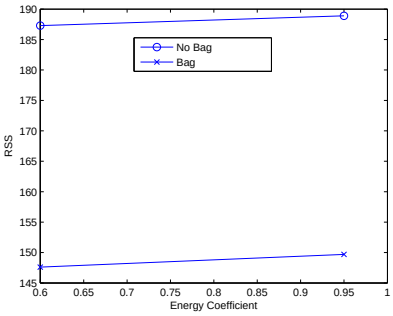
(e) Digit 4

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

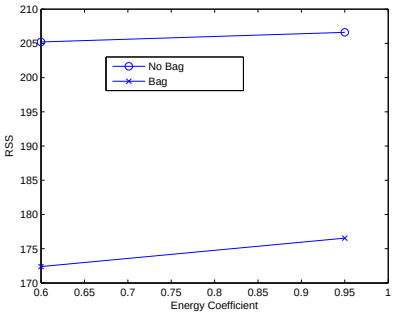


(a) Digit 5

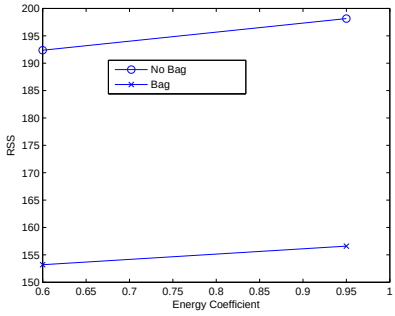
(b) Digit 6



(c) Digit 7

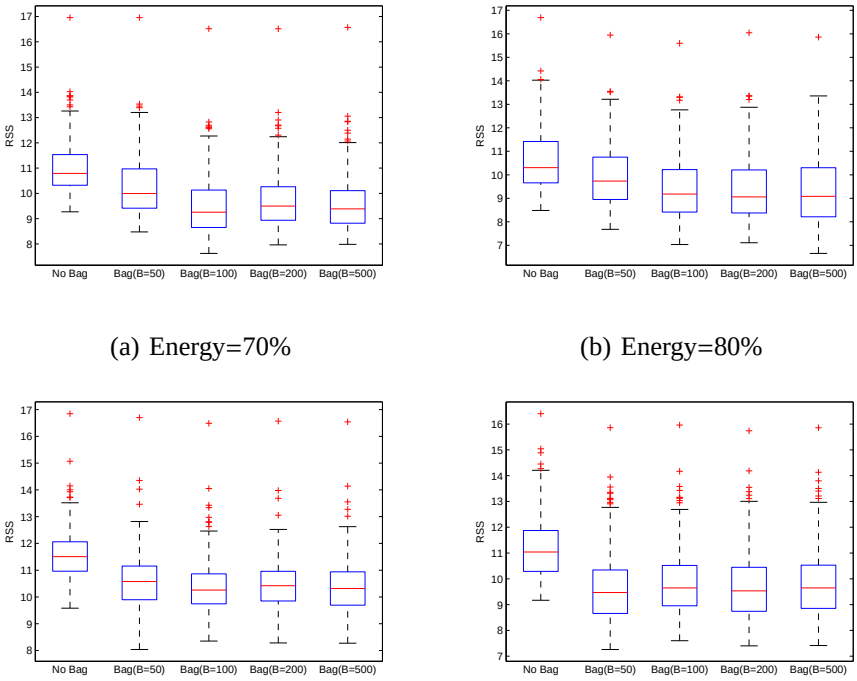


(d) Digit 8



(e) Digit 9

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



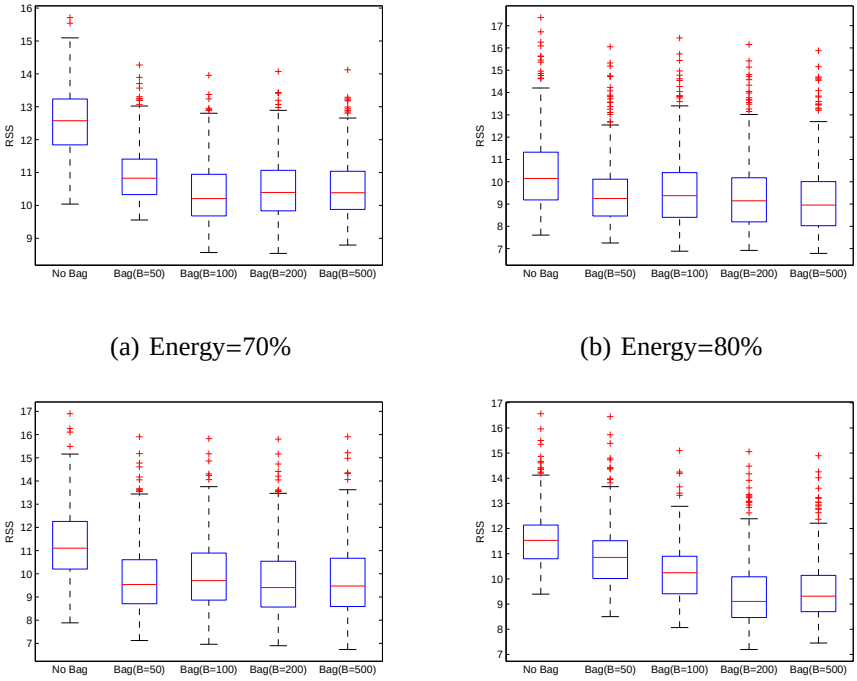
(a) Energy=70%

(b) Energy=80%

(c) Energy=90%

(d) Energy=95%

Figure 17: 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.



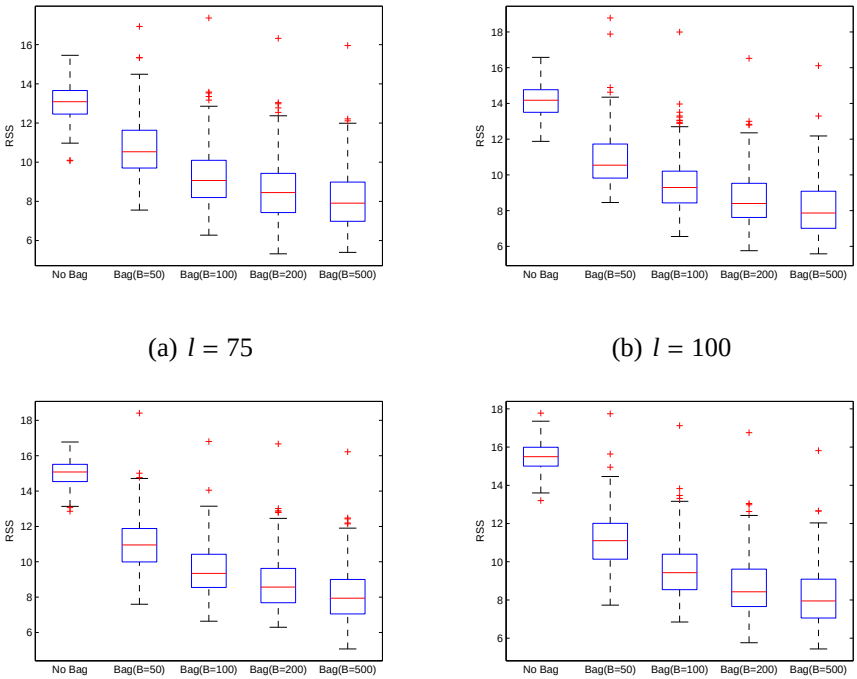
(a) Energy=70%

(b) Energy=80%

(c) Energy=90%

(d) Energy=95%

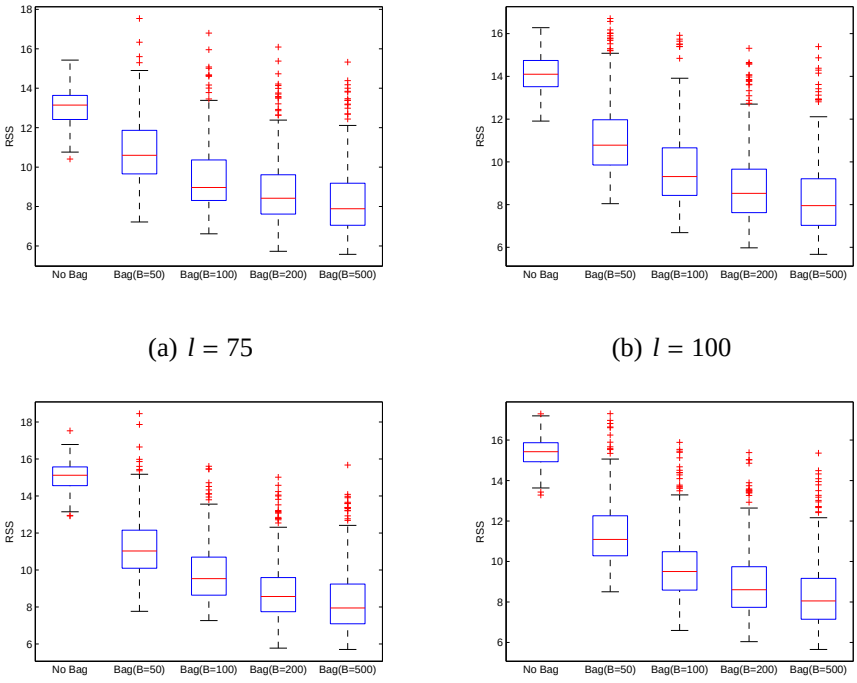
Figure 18: Boxplot of RSS values for digit 9 for BKPCA approach versus penalized preimage approach for different values of energy to be preserved and B. Independent Gaussian noise $\mu = 0$ and $\sigma_G = 1$ was added.



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

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

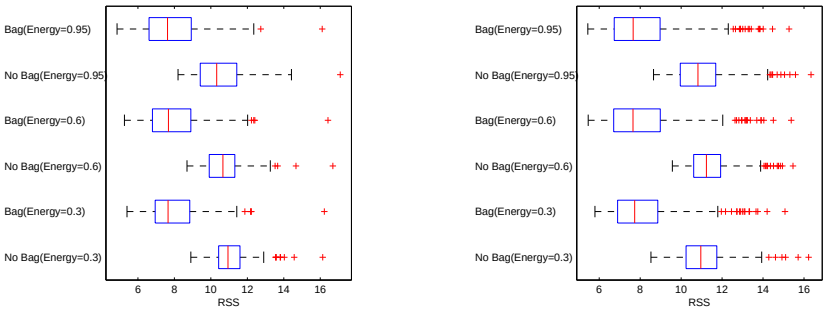
Figure 19: Boxplot of RSS values for digit 7 for BKPCA approach versus gradient descent approach for different values of l and B. Independent Gaussian noise $\mu = 0$ and $\sigma_G = 1$ was added.



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

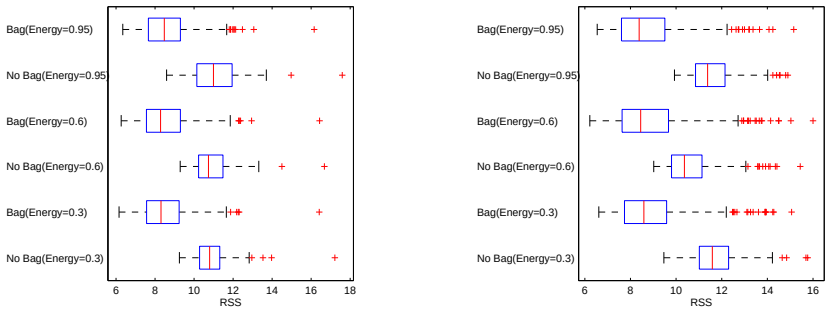
(c) $l = 135$ (d) $l = 160$

Figure 20: 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.



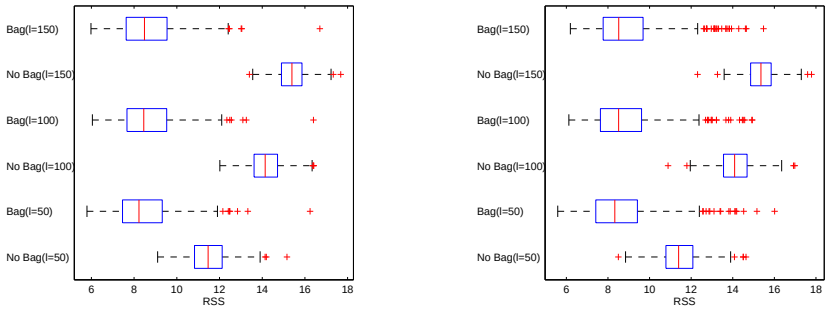
(a) Digit 7 (b) Digit 9

Figure 21: 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.



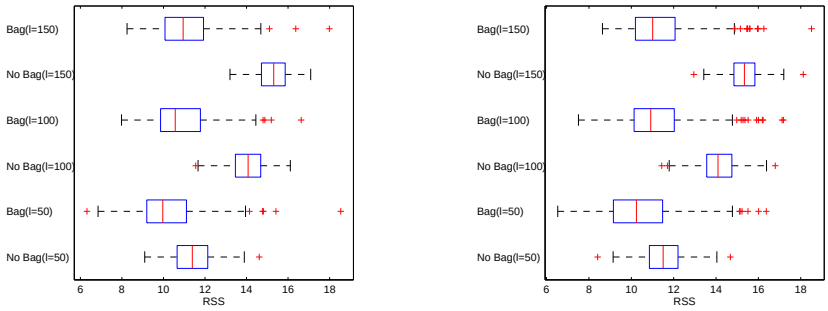
(a) Digit 7 (b) Digit 9

Figure 22: 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.



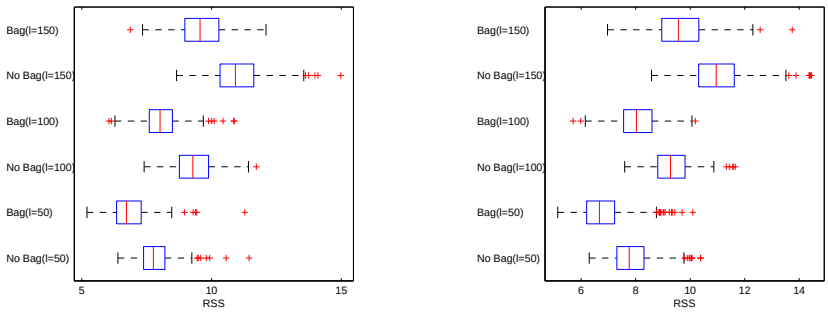
(a) Digit 7 (b) Digit 9

Figure 23: 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.



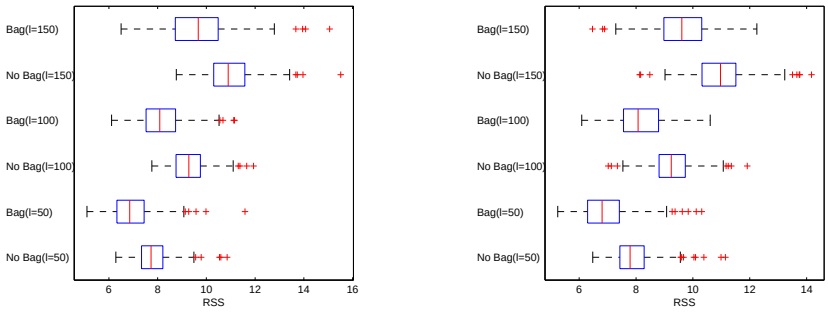
(a) Digit 7 (b) Digit 9

Figure 24: 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 25: 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 26: 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 27: 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.

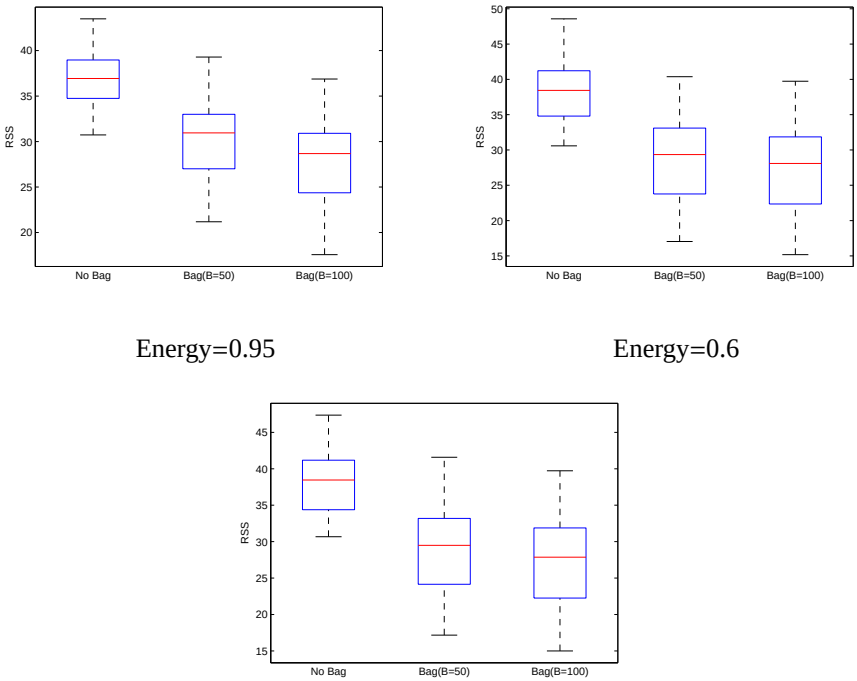
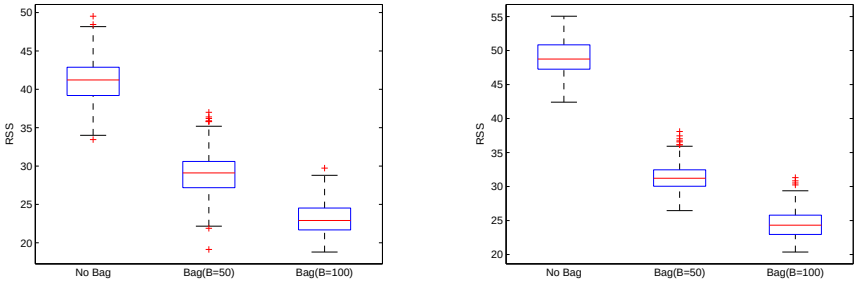
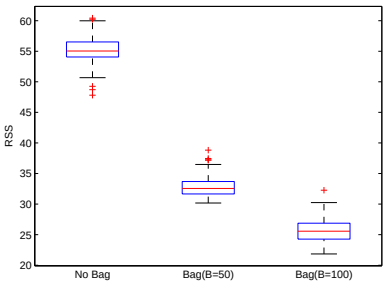


Figure 28: Face data RSS values of penalized preimage approach versus the bagged approach for different energy levels with added Gaussian noise ($\mu = 0$ and $\sigma_G = 1$).

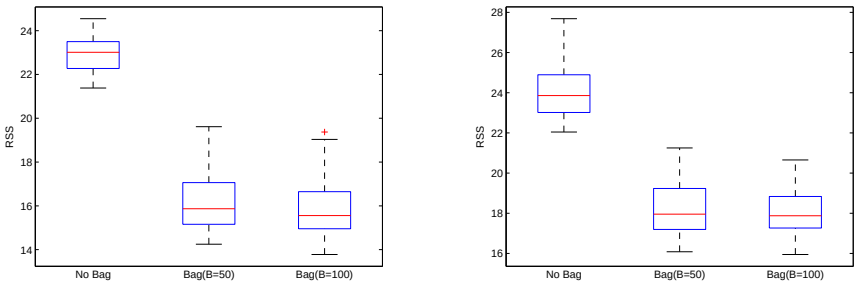


$l = 150$ $l = 300$

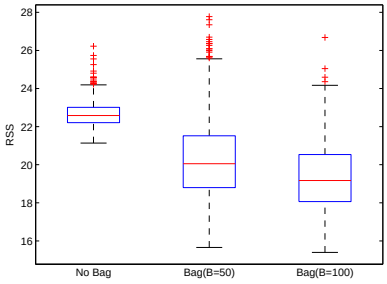


$l = 400$

Figure 29: 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 $\sigma_G = 1$).



$l = 150$ $l = 300$



$l = 400$

Figure 30: Face data RSS values of KMDS algorithm versus the bagged approach for different numbers of eigenvalues (l) with added Gaussian noise ($\mu = 0$ and $\sigma_G = 1$).

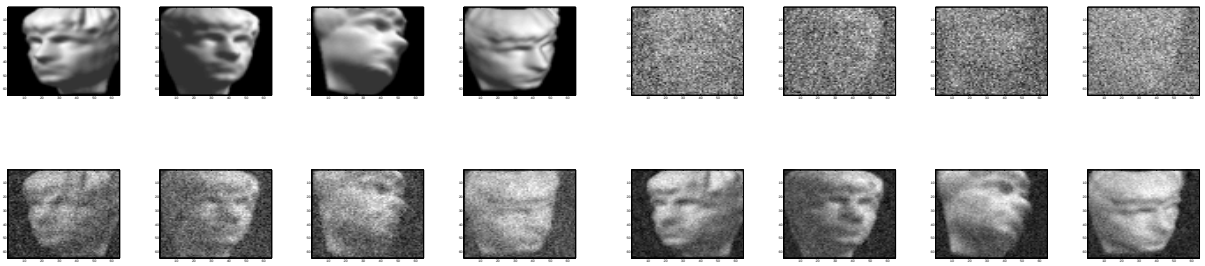


Figure 31: Results for selected face images with $l = 150$ and $n = 10$. First row shows noiseless reference images. Second row shows noisy test images (Gaussian noise $\mu = 0$ and $\sigma_G = 0.7$). Third row shows denoised preimages from the KMDS method. Fourth row shows denoised preimages from our BKPCA method. The improved results from BKPCA can be seen.