SAND2007-4317C

An Approximate Version of Kernel PCA

Shawn Martin*
Sandia National Laboratories†
PO Box 5800
Albuquerque, NM 87185-0310
smartin@sandia.gov

October 2, 2006

Abstract

We propose an analog of kernel Principal Component Analysis (kernel PCA). Our algorithm is based on an approximation of PCA which uses Gram-Schmidt orthonormalization. We combine this approximation with Support Vector Machine kernels to obtain a nonlinear generalization of PCA. By using our approximation to PCA we are able to provide a more easily computed (in the case of many data points) and readily interpretable version of kernel PCA. After demonstrating our algorithm on some examples, we explore its use in applications to fluid flow and microarray data.

1 Introduction

The goal of Principal Component Analysis (PCA) is to find a coordinate representation for a data set such that the most variance in the data is captured in the least number of coordinates. This representation is typically found by performing a linear transformation of the original data via the Singular Value Decomposition (SVD). The resulting singular vectors provide an orthonormal basis for the data while the singular values provide information on the importance of each basis vector. A review of PCA, its history, examples of applications, and information about the SVD can be found in [7], [9], [3], and [19].

In addition to the standard linear version of PCA, some nonlinear variants have been proposed. These methods include Hebbian networks [3], multi-layer perceptrons [3], Principal Curves [5], and kernel PCA [15]. Finally, there are other data analysis methods similar in spirit to PCA. These include Projection Pursuit [4], Independent Component Analysis [8], [6], Isomap [18], and Locally Linear Embedding [12].

In this paper we propose an approximate version of the nonlinear kernel version of PCA [15]. Kernel PCA combines the computation of PCA by diagonalizing the covariance matrix with nonlinear preprocessing using Support Vector Machine (SVM) kernels [2], [16]. This results in a nonlinear version of PCA which reduces to the standard linear version when using a linear kernel. We propose an approximate version of this method in order to overcome two inherent difficulties.

First, the method of nonlinear preprocessing using kernels makes it difficult to return to the original input space. In the case of kernel PCA, it is difficult to interpret the nonlinear principal components because they do not typically correspond to vectors in the input space. Second, kernel PCA is performed by diagonalizing an $m \times m$ matrix, where m is the number of data points under consideration. In the case of large m this is not feasible. Both of these problems are addressed by using our approximate version of kernel PCA. These problems are also addressed, using different approaches, in [14] and [17].

Our version of kernel PCA is based on the use of an approximation to standard PCA. In our modification of standard PCA we locate a basis using the same criterion employed by PCA but subject to an additional constraint. Our basis is required to correspond directly to a linearly independent subset of our original data. Essentially, we find an ordered linearly independent subset of our data which best approximates the PCA expansion when used with Gram-Schmidt orthonormalization. When coupled with SVM kernels we get an approximate version of kernel PCA which is easier to compute (in the case of large datasets) and more readily interpretable.

*This work was supported by the Office of Advanced Scientific Computing Research (OASCR) Mathematics, Information, and Computational Sciences (MICS) Project 547 Task 01.11.

†Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

2 Background

Our version of kernel PCA is a combination of Gram-Schmidt orthonormalization and PCA, all rewritten in terms of inner products so that SVM kernels can be used. We therefore provide some background on Gram-Schmidt, PCA, and SVM kernel functions.

2.1 Gram-Schmidt. Gram-Schmidt orthonormalization [19] is a procedure for transforming a set of linearly independent vectors $\{\mathbf{x}_1, \ldots \mathbf{x}_m\}$ into an orthonormal basis $\{\mathbf{u}_1, \ldots, \mathbf{u}_m\}$. This basis is constructed iteratively via projections. Specifically,

$$\begin{gathered}\mathbf{u}_1 = \tfrac{\mathbf{x}_1}{\|\mathbf{x}_1\|},\ \mathbf{p}_1 = (\mathbf{x}_2, \mathbf{u}_1)\mathbf{u}_1 \\ \mathbf{u}_2 = \tfrac{\mathbf{x}_2 - \mathbf{p}_1}{\|\mathbf{x}_2 - \mathbf{p}_1\|},\ \mathbf{p}_2 = (\mathbf{x}_3, \mathbf{u}_2)\mathbf{u}_2 + (\mathbf{x}_3, \mathbf{u}_1)\mathbf{u}_1 \\ \vdots \\ \mathbf{u}_m = \tfrac{\mathbf{x}_m - \mathbf{p}_{m-1}}{\|\mathbf{x}_m - \mathbf{p}_{m-1}\|},\end{gathered}$$

where we use $(\mathbf{x}, \mathbf{y})$ to denote the inner product (dot product) of $\mathbf{x}$ with $\mathbf{y}$. Now we can represent our original data $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ in the new basis as the upper triangular matrix

$$\begin{pmatrix} (\mathbf{x}_1, \mathbf{u}_1) & \cdots & (\mathbf{x}_m, \mathbf{u}_1) \\ \vdots & \ddots & \vdots \\ (\mathbf{x}_1, \mathbf{u}_m) & \cdots & (\mathbf{x}_m, \mathbf{u}_m) \end{pmatrix}.$$

We can also include linearly dependent data $\{\mathbf{x}_{m+1}, \ldots, \mathbf{x}_n\}$ as additional columns in the matrix

$$U = \begin{pmatrix} (\mathbf{x}_1, \mathbf{u}_1) & \cdots & (\mathbf{x}_m, \mathbf{u}_1) & \cdots & (\mathbf{x}_n, \mathbf{u}_1) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ (\mathbf{x}_1, \mathbf{u}_m) & \cdots & (\mathbf{x}_m, \mathbf{u}_m) & \cdots & (\mathbf{x}_n, \mathbf{u}_m) \end{pmatrix}.$$

2.2 PCA. PCA is typically formulated as an eigenvalue problem which is closely related to the SVD [9], [3]. It also typically described as a procedure for successively capturing the maximal variance in the data. The PCA eigenvectors (singular vectors) satisfy [9]

$$\begin{aligned}\mathbf{u}_1 &= \arg\max_{\mathbf{u}} \textstyle\sum_{i=1}^{n} (\mathbf{u}, \mathbf{x}_i)^2 \\ \mathbf{u}_2 &= \arg\max_{\mathbf{u}} \textstyle\sum_{i=1}^{n} (\mathbf{u}, \mathbf{x}_i - (\mathbf{x}_i, \mathbf{u}_1)\mathbf{u}_1)^2 \\ &\vdots \\ \mathbf{u}_m &= \arg\max_{\mathbf{u}} \textstyle\sum_{i=1}^{n} (\mathbf{u}, \mathbf{x}_i - \sum_{j=1}^{m-1} (\mathbf{x}_i, \mathbf{u}_j)\mathbf{u}_j)^2,\end{aligned}$$

where $\{\mathbf{u}_1, \ldots, \mathbf{u}_m\}$ are also required to be orthonormal.

PCA is often illustrated by finding the major and minor axes in a cloud of data filling an ellipse. The first eigenvector corresponds to the major axis of the ellipse while the second eigenvector corresponds to the minor axis. This example is shown later in Figure 1.

2.3 SVM Kernels. SVM kernels were orginally used in the context of integral operators, but have also been interpreted as inner products in feature spaces for use in machine learning. Kernels have been applied to Support Vector Machines [2], Principal Component Analysis [15], Fisher's Linear Discriminant [10], and a host of other algorithms.

A SVM kernel is a function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ with an associated map $\Phi : \mathbb{R}^n \to F$ such that

$$k(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}), \Phi(\mathbf{y})), \tag{2.1}$$

where F is an inner product space. The map Φ is typically nonlinear and the relation (2.1) is used to avoid explicit computation of $\Phi(\mathbf{x})$. Some simple kernels are the linear kernel $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}, \mathbf{y})$, the polynomial kernel $k(\mathbf{x}, \mathbf{y}) = ((\mathbf{x}, \mathbf{y}) + c)^d,\ d \in \mathbb{Z}$ and the gaussian radial basis function kernel $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2/2\sigma^2),\ \sigma \neq 0$.

SVM kernels are useful in machine learning because they allow the application of linear methods to nonlinear problems. In principle, an appropriate map Φ can be used to change a nonlinear problem in $\mathbb{R}^n$ into a linear problem in F. Once the problem has undergone this transformation, a linear method can be applied.

SVM kernels come into play because a given nonlinear map Φ usually results in a large (sometimes infinite) increase in the dimension of the original problem. To avoid this dimensional increase, the inner products in a linear method are replaced by kernels. This substitution yields a nonlinear method which never explicity uses the map Φ. By replacing inner products $(\mathbf{x}_i, \mathbf{x}_j)$ with kernels $k(\mathbf{x}_i, \mathbf{x}_j) = (\Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j))$ we effectively remap our problem using Φ before applying an inner product in a higher dimensional space.

Additional information on kernels and their use in machine learning can be found in [16].

3 Algorithm

Having provided the background on Gram-Schmidt, PCA, and SVM kernels, we can describe the basic strategy of our algorithm. First, a linearly independent subset of our data is chosen using the PCA criterion. Next, a kernel version of Gram-Schmidt is performed using that subset. The implementation of this strategy involves rewriting Gram-Schmidt in terms of inner products and constraining the PCA criterion so that each $\mathbf{u}_i$ corresponds directly to an actual data point. This is our approximation to PCA (APCA). By replacing inner products with kernels we get the approximate version of kernel PCA (AKPCA).

AKPCA is based on a reformulation of the Gram-Schmidt procedure. To describe this reformulation we observe that Gram-Schmidt is recursive and that we can

rewrite it as follows

$$\begin{aligned}
(\mathbf{x}_1, \mathbf{u}_1) &= \|\mathbf{x}_1\| = \tfrac{(\mathbf{x}_1,\mathbf{x}_1)}{(\mathbf{x}_1,\mathbf{u}_1)} \\
(\mathbf{x}_2, \mathbf{u}_1) &= \tfrac{(\mathbf{x}_2,\mathbf{x}_1)}{\|\mathbf{x}_1\|} = \tfrac{(\mathbf{x}_2,\mathbf{x}_1)}{(\mathbf{x}_1,\mathbf{u}_1)} \\
(\mathbf{x}_2, \mathbf{u}_2)^2 &= \|\mathbf{x}_2 - \mathbf{p}_1\|^2 = \|\mathbf{x}_2\|^2 - \|\mathbf{p}_1\|^2 \\
(\mathbf{x}_2, \mathbf{u}_2) &= \tfrac{1}{(\mathbf{x}_2,\mathbf{u}_2)} \left[(\mathbf{x}_2, \mathbf{x}_2) - (\mathbf{x}_2, \mathbf{u}_1)^2\right] \\
(\mathbf{x}_3, \mathbf{u}_1) &= \tfrac{(\mathbf{x}_3,\mathbf{x}_1)}{(\mathbf{x}_1,\mathbf{u}_1)} \\
(\mathbf{x}_3, \mathbf{u}_2) &= \tfrac{1}{(\mathbf{x}_2,\mathbf{u}_2)} \left[(\mathbf{x}_3, \mathbf{x}_2) - (\mathbf{x}_2, \mathbf{u}_1)(\mathbf{x}_3, \mathbf{u}_1)\right],
\end{aligned}$$

and in general

$$(\mathbf{x}_i, \mathbf{u}_j) = \frac{1}{(\mathbf{x}_j, \mathbf{u}_j)} \left[(\mathbf{x}_i, \mathbf{x}_j) - \sum_{k=1}^{j-1} (\mathbf{x}_j, \mathbf{u}_k)(\mathbf{x}_i, \mathbf{u}_k)\right].$$

This formula can also be used for the remainder of our (linearly dependent) data $\{\mathbf{x}_{m+1}, \ldots, \mathbf{x}_n\}$ to arrive at the matrix U in Section 2.1.

Another useful addition to Gram-Schmidt is a change of basis matrix for switching from $\{\mathbf{u}_1, \ldots, \mathbf{u}_m\}$ to $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$. This matrix, $T = (\mathbf{t}_1, \ldots, \mathbf{t}_m)$, can be computed columnwise by

$$\begin{aligned}
\mathbf{t}_1 &= \tfrac{1}{(\mathbf{x}_1,\mathbf{u}_1)} \mathbf{e}_1 \\
\mathbf{t}_2 &= \tfrac{1}{(\mathbf{x}_2,\mathbf{u}_2)} \left[\mathbf{e}_2 - (\mathbf{x}_2, \mathbf{u}_1)\mathbf{t}_1\right] \\
&\vdots \\
\mathbf{t}_m &= \tfrac{1}{(\mathbf{x}_m,\mathbf{u}_m)} \left[\mathbf{e}_m - \textstyle\sum_{i=1}^{m-1} (\mathbf{x}_m, \mathbf{u}_i)\mathbf{t}_i\right],
\end{aligned}$$

where $\{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$ are the standard basis vectors. Now TU expresses our data in terms of the basis $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$.

This formulation of Gram-Schmidt has two principal advantages over the standard formulation for use with AKPCA. First, it is expressed in terms of inner products to allow the use of SVM kernels. Second, the change of basis matrix T allows us to express our data in terms of actual examples in our data set. This will allow us to interpret the results of any nonlinear remapping of our data.

We next modify our reformulation of Gram-Schmidt by a constrained version of the PCA criterion. Specifically, we use the PCA criterion to select the linearly independent subset used in Gram-Schmidt. Abandoning our previous labeling $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ in favor of the more accurate labeling $\{\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_m}\}$, this subset is selected by

$$\begin{aligned}
\mathbf{x}_{i_1} &= \arg\max_{\mathbf{x}_i} \tfrac{1}{\|\mathbf{x}_i\|^2} \textstyle\sum_{j=1}^{n} (\mathbf{x}_i, \mathbf{x}_j)^2 \\
\mathbf{x}_{i_2} &= \arg\max_{\mathbf{x}_i} \\
&\quad \tfrac{1}{\|\mathbf{x}_i\|^2} \textstyle\sum_{j=1}^{n} \left[(\mathbf{x}_i, \mathbf{x}_j) - (\mathbf{x}_i, \mathbf{u}_1)(\mathbf{x}_j, \mathbf{u}_1)\right]^2 \\
&\vdots \\
\mathbf{x}_{i_m} &= \arg\max_{\mathbf{x}_i} \\
&\quad \tfrac{1}{\|\mathbf{x}_i\|^2} \sum_{j=1}^{n} \left[(\mathbf{x}_i, \mathbf{x}_j) - \sum_{l=1}^{m-1} (\mathbf{x}_i, \mathbf{u}_l)(\mathbf{x}_j, \mathbf{u}_l)\right]^2,
\end{aligned}$$

Figure 1. Ellipse and Parabola. In this illustration we compare the singular vectors found by PCA with those found by APCA for an ellipse (left) and a parabola (right). In both examples, a PCA singular vector is shown as a solid line, while a APCA singular vector (an actual point in the data set) is marked with an X.

where the inner products $(\mathbf{x}_\bullet, \mathbf{u}_l)$ are computed using the reformulation of the Gram-Schmidt procedure described above.

This combination of Gram-Schmidt and PCA is APCA (our approximate version of PCA). APCA provides an interesting approximation of PCA because it selects points in the actual data set with properties similar to those of PCA. In addition, APCA is readily generalized to AKPCA (our approximate version of kernel PCA) by replacing inner products with kernels. AKPCA then combines the advantages of our reformulation of Gram-Schmidt (kernel use and interpretation after remapping) with the approximate statistical properties of APCA.

4 Examples and Applications

Here we illustrate the use of AKPCA by various examples and applications. Our first example uses an ellipsoid and a parabola. We compare the singular vectors found by PCA with those found by APCA (or AKPCA using a linear kernel) on the ellipse filling data cloud mentioned previously, and on a parabola with gaussian noise. These examples are shown in Figure 1.

In our next example we use AKPCA on the parabola with different polynomial kernels. In this parabola, x-values are drawn from a uniform distribution on $[-1, 1]$ and y-values are the squares of the x-values plus Gaussian noise with a standard deviation of .2. This example was also used in [15] so provides some comparison with that work. Following [15] we use polynomial kernels with $c = 1$ and $d = 2$, 3, and 4. Our results are shown in Figure 2. In the cases $d = 2$ and $d = 3$ the first two singular vectors point in the directions of the arms of the parabola and the third singular vector points in the direction of the noise. When $d = 4$, the first two singular vectors point in the directions of the parabola arms, the second two point in

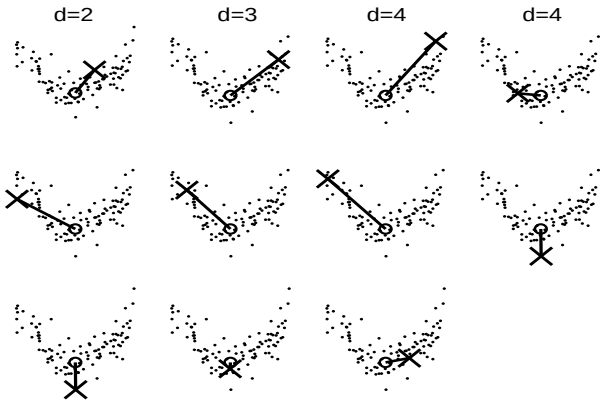

Figure 2. Parabola Revisited. Illustrated here are the singular vectors found by AKPCA using the polynomial kernel with $c = 1$ and $d = 2, 3$, and 4 for a parabola. The first column (from top to bottom) shows the first three singular vectors when $d = 2$, the second column shows the first three singular vectors when $d = 3$, and the third and fourth columns show the first five singular vectors when $d = 4$. In each plot, the origin is marked with a circle and connected via a line to a singular vector marked by an X.

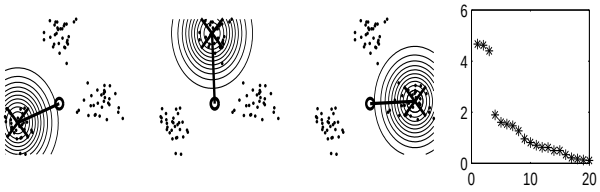

Figure 3. Clusters. Here we show singular vectors and values for three Gaussian clusters found using AKPCA with an RBF kernel. The first three plots show the singular vectors and the last plot shows the singular values. The singular vectors are displayed as in Figure 2, with the origin marked by a circle, each singular vector marked with an X, and contours are shown corresponding to hyperplanes in the remapped data space.

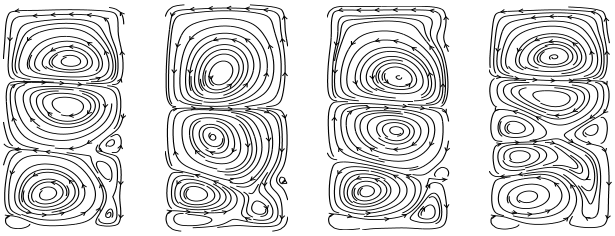

Figure 4. Taylor-Couette Flow. Shown here are the first four APCA eigenflows from left to right. Each plot shows streamlines of the flow based on the radial and axial velocities.

the directions of the noise in the arms, and the fifth points in the direction of overall noise.

The last example, found in an expanded version [13] of [15], consists of three Gaussian clusters in the region $[-1, 1] \times [-.5, 1]$. Each cluster has a standard deviation of .1. In this example we use AKPCA with an RBF kernel (following [13]) of width $\sigma = .22$. Our results are shown in Figure 3. Since we are using a Gaussian kernel which matches our Gaussian clusters, AKPCA performs center selection. This occurs because the three centers given by the first three singular vectors are most representative of the data when viewed through the eyes of our Gaussian kernel. This is illustrated by both the singular vectors, which are seen to be centers, and the singular values, which are dominated by the first three nearly equal values.

4.1 Taylor-Couette Fluid Flow. Next, we investigated the application of APCA to a larger data set. Specifically, we apply APCA to a problem in Taylor-Couette fluid flow. Taylor-Couette flow occurs in fluid trapped between concentric cylinders. When the cylinders are rotated independently, the resulting flow often contains toroidal vortices known as Taylor-Couette cells. The Taylor-Couette data we used was generated by numerical simulation [1], and consists of 799 cross-sectional snapshots of the flow as time progresses. Each snapshot contains three velocities and one pressure at points on a 49×21 dimensional grid. We reshaped each snapshot into a 4116 dimensional vector and put the data into a 4116×799 matrix. We performed APCA on this matrix to obtain the first four eigenflows shown in Figure 4.

Our first remark concerning these eigenflows is that the corresponding singular values account for 96% of the energy of the flow. In other words, the first four singular values added together make up 96% of the sum of all the singular values. This means that the eigenflows under consideration characterize the flow to a high degree of accuracy.

Next we remark that when we project the entire flow onto these eigenflows, we get periodic graphs, as shown in Figure 5. Based on our previous remark, we may conclude with high confidence that the flow under consideration is periodic.

Finally, we remark on the eigenflows themselves.

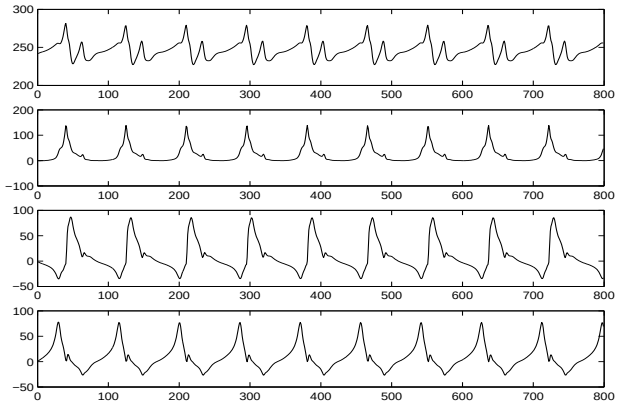

Figure 5. Taylor-Couette Projections. These plots show the projections of the entire Taylor-Couette flow onto the first four eigenflows. The top plot is for the projection onto the first eigenflow, the next plot is for the projection onto the second eigenflow, et cetera. In each plot the x-axis represents time and the y-axis the value of the projection.

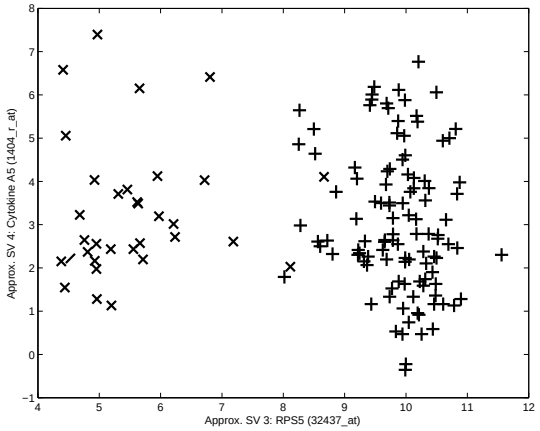

Figure 6. Anomalous Microarray Cluster. This cluster, with members marked by X's, was found by performing APCA on the variables (genes) in the dataset. The plot shows the values for each patient of the third and fourth singular vector genes.

These particular snapshots of the flow represent what the flow is doing most of the time. In this case, the flow consists of three Taylor-Couette cells. The upper cell is stable while the lower cells periodically grow and shrink, at some point even spawning and re-absorbing smaller sub-cells. All these events are captured by the eigenflows in Figure 4. In fact, by comparing Figures 4 and 5, it is almost possible to visualize the entire flow, including the appearance and disappearance of the sub-cells in the fourth eigenflow.

4.2 DNA Microarray Data. Finally, we describe an application of APCA to microarray data. In this application, we discuss a new way to assess leukemia microarray data quality. This data used was from a study of acute infant leukemia using gene expression profiling [11]. The dataset consisted of 140 acute lymphoid leukemia (ALL) and acute myeloid leukemia (AML) cases, profiled using Affymetrix U95AV2 gene chips to obtain 12,625 expression values for each patient.

APCA performed on the variables (genes) discovered a cluster of approximately 33 patients (shown in Figure 6). This cluster was apparent in the third singular vector and by using alternates to that singular vector we produced a list of genes correlated with the cluster. We discovered that most of the genes in this list were down regulated and indicative of low viability. When we found that 20 of the 33 patients in the cluster had been profiled on the same day, we became convinced that these experiments had suffered from some systematic problem (perhaps one or more bad enzymes used during processing). This successful application of APCA resulted in the identification of a list of genes that can be monitored to suggest when microarray data should be examined for other problems before analysis.

5 Discussion

5.1 Relation to PCA. AKPCA is based on an approximation of PCA. Consequently, AKPCA has approximately the same properties as PCA. Some of these properties are [9]: maximization of the statistical variance, minimization of the mean square truncation error, maximization of the mean squared projection, and minimization of entropy. In the case of APCA these properties are directly approximated. In the case of AKPCA these properties apply (approximately) after the nonlinear remapping.

In practice, PCA is often used for low dimensional representation and for visualization of data. AKPCA inherits these attributes with the additional advantage of the interpretability of the AKPCA singular vectors. This advantage was illustrated with the applications of APCA to the Taylor-Couette simulation data and to the microarray data. In the case of the Taylor-Couette data we were able to produce actual instances in the flow which best represented the entire flow. This provided a very useful visualization of the flow, especially in the case of the sub-cells in the fourth APCA eigenflow. In the case of the microarray data we were able to discover an interesting cluster and to produce a set of genes which could be used to interpret the corresponding cluster.

5.2 Comparison to Kernel PCA. Kernel PCA is very similar to AKPCA. Kernel PCA is computed by performing an eigenvalue decomposition of a kernel version of the covariance matrix. When using the linear kernel $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}, \mathbf{y})$, kernel PCA reduces to PCA. Thus kernel PCA is a direct generalization of PCA and requires only linear optimization. In fact, the only difference between kernel PCA and AKPCA is that AKPCA uses APCA to approximate PCA before using an SVM kernel. This gives AKPCA two advantages over kernel PCA. First, the nonlinear principal components found by kernel PCA do not, in general, correspond to points in the original data set. In fact, a kernel PCA component may not correspond to *any* point in the original data *space*. In comparison, each nonlinear principal component found by AKPCA corresponds directly to a point in the original data set. This is an advantage when interpreting the principal components that AKPCA finds. Second, the computation of kernel PCA requires an eigenvalue decomposition of an $n \times n$ matrix, where n is the number of data points under consideration. Although linear, this is a difficult calculation for large n. In comparison, AKPCA is matrix free. AKPCA uses the same matrix, but the explicit formulation of the matrix is not required. Even if the matrix is formed, an eigenvalue decomposition is not necessary.

5.3 Conclusion. We have introduced both an interesting appoximation to PCA and a nonlinear extension of that approximation. APCA performs PCA constrained to the original data set. This makes the singular vectors found by APCA more interpretable than the analagous singular vectors found by standard PCA. Thus APCA provides information that is difficult to obtain by PCA alone. This was illustrated using the Taylor-Couette simulation data. AKPCA provides a nonlinear generalization of APCA and so can be considered a nonlinear generalization of PCA. AKPCA provides the interpretability of APCA in a nonlinear setting.

The algorithm which performs AKPCA is also capable of data analysis on very large data sets. It is parallelizable and can be implemented to exploit computers with large memory capacities. (It can also be used with minimal memory requirements.) In addition, the AKPCA algorithm is modular and can be easily changed to perform other tasks. It can be extended for use with other algorithms such as regression and classification.

References

[1] R. Adair. *Simulations of Taylor-Couette Flow*. PhD thesis, Colorado State University, 1997.

[2] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

[3] K.I. Diamantaras and S.Y. Kung. *Principal Component Neural Networks*. John Wiley & Sons, 1996.

[4] J. H. Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, 82:249–266, 1987.

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

[6] Aapo Hyvärinen. Survey on Independent Component Analysis. *Neural Computing Surveys*, 2:94–128, 1999.

[7] I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[8] C. Jutten and J. Herault. Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture, 1991.

[9] Michael Kirby. *Geometric Data Analysis*. John Wiley & Sons, 2001.

[10] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE, 1999.

[11] M. Mosquera-Caro, J. Potter, E. Andries, S. Martin, P. Helman, R. Veroff, H. Kang, X. Wang, B. Camitta, J. Shuster, A. Carroll, M. Murphy, F. Shultz, C. Wilson, G. Dahl, R. Arceci, D. Haaland, G. Davidson, S. Atlas, and C. Willman. Gene expression profiling for molecular classification and outcome prediction in infant leukemia reveals novel biological clusters, etiologies and pathways for treatment failure. *Blood*, 100(11):254, 2002.

[12] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 22 Dec. 2000.

[13] B. Schölkopf. *Support Vector Learning*. Oldenbourg Verlag, Munich, 1997. Doktorarbeit, TU Berlin.

[14] B. Schölkopf, S. Mika, A. Smola, G. Rätsch, and K.-R. Müller. Kernel PCA pattern reconstruction *via* approximate pre-images. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, pages 147–152, Berlin, 1998. Springer Verlag.

[15] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - SV Learning*, pages 327–352. MIT Press, Cambridge, MA, 1999.

[16] B. Schölkopof and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[17] Alex J. Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proc. 17th International Conf. on Machine Learning*, pages 911–918. Morgan Kaufmann, San Francisco, CA, 2000.

[18] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 22 December 2000.

[19] L. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, Philadelphia, 1997.