

SPECTRAL CLUSTERING AND KERNEL PRINCIPAL COMPONENT ANALYSIS ARE PURSUING GOOD PROJECTIONS

VIKAS CHANDRAKANT RAYKAR
DECEMBER 15, 2004

ABSTRACT. We interpret spectral clustering algorithms in the light of unsupervised learning techniques like principal component analysis and kernel principal component analysis. We show both are equivalent up to a normalization of the dot product or the affinity matrix.

1. INTRODUCTION

Many image segmentation algorithms can be cast in the clustering framework. Each pixel based on its and its neighbors properties can be considered as a point in some high dimensional space. Image Segmentation essentially consists of finding clusters in this high dimensional space. We have two sides to the problem: How good is the embedding? and How well can you cluster these possibly nonlinear clusters?

Given a set of n data points in some d -dimensional space, the goal of clustering is to partition the data points into k clusters. Clustering can be considered as an instance of unsupervised techniques used in machine learning. If the data are tightly clustered or lie in nice linear subspaces then simple techniques like k -means clustering can find the clusters. However in most of the cases the data lie on highly nonlinear manifolds and a simple measure of Euclidean distance between the points in the original d -dimensional space gives very unsatisfactory results.

A plethora of techniques have been proposed in recent times which show very impressive results with highly nonlinear manifolds. The important among them include Kernel Principal Component Analysis (KPCA), Non-linear manifold learning (Isomap, Locally Linear Embedding (LLE)), Spectral Clustering, Random Fields, Belief Propagation, Markov Random Fields and Diffusion. All these methods though different can be shown to be very similar in spirit. The goal of the proposed project is to understand all these methods in a unified view that all these methods are doing clustering and more importantly, all are trying to do it in a very similar way which we elucidate in the next paragraph. Even though in the long goal we would like to link all the methods this report shows the link between spectral clustering and KPCA. A shadow of doubt should have already crept in when

one observes that both use the Gaussian kernel. Ng, Jordan and Weiss¹ mention in the concluding section that there are some intriguing similarities between spectral clustering and kernel PCA.

2. PRINCIPAL COMPONENT ANALYSIS (PCA)

Given N data points in d dimensions, by doing a PCA we project the data points onto p directions ($p \leq d$) which capture the maximum variance of the data. When $p \ll d$ then PCA can be used as a statistical dimensionality reduction technique. These directions correspond to the eigen vectors of the covariance matrix of the training data points. Intuitively PCA fits an ellipsoid in d dimensions and uses the projections of the data points on the first p major axes of the ellipsoid.

Consider N column vectors X_1, X_2, \dots, X_N , each representing a point in the d -dimensional space. The PCA algorithm can be summarized as follows.

- (1) Subtract the mean from all the data points $X_j \leftarrow X_j - \frac{1}{N} \sum_{i=1}^N X_i$.
- (2) Compute the $d \times d$ covariance matrix $S = \sum_{i=1}^N X_i X_i^T$.
- (3) Compute the $p \leq d$ largest eigen values of S and the corresponding eigen vectors.
- (4) Project the data points on the eigen vectors and use the projections instead of the data points.

With respect to clustering, the problem will be simplified in the sense that we are now clustering in a lower dimensional space (i.e. each point is represented by its projection on the first few principal eigen vectors). Consider Figure 1(a) which shows two linearly separable clusters. The two eigen vectors are also marked. Figure 1(a) and (c) show the first and the second principal component for each of the points. From Figure 1(b) it can be seen that the first principal component can be used to segment the points into two clusters.

PCA has very nice properties like it is the best linear representation in the MSE sense, the first p principal components have more variance than any other components etc. However PCA is still linear. It uses only second order statistics in the form of covariance matrix. The best it can do is to fit an ellipsoid around the data. These can be seen for the data set in Figure 1(b) where linear algorithms cannot find the cluster. A more natural representation would be one principal direction along the circle and another normal to it.

One way to get around this problem is to embed the data in some higher dimensional space and then do a PCA in that space. Consider the data set in Figure 2(a) and the corresponding embedding in the 3 dimensional space in Figure 2(b). Here each point (x, y) is mapped to a point $(x, y, x^2 + y^2)$. Figure 2(c)(d) and (e) show the first three principal components in this three dimensional space. Note that the third principal components can now

¹On Spectral Clustering: Analysis and an algorithm, Andrew Y. Ng, Michael Jordan, and Yair Weiss. In NIPS 14., 2002

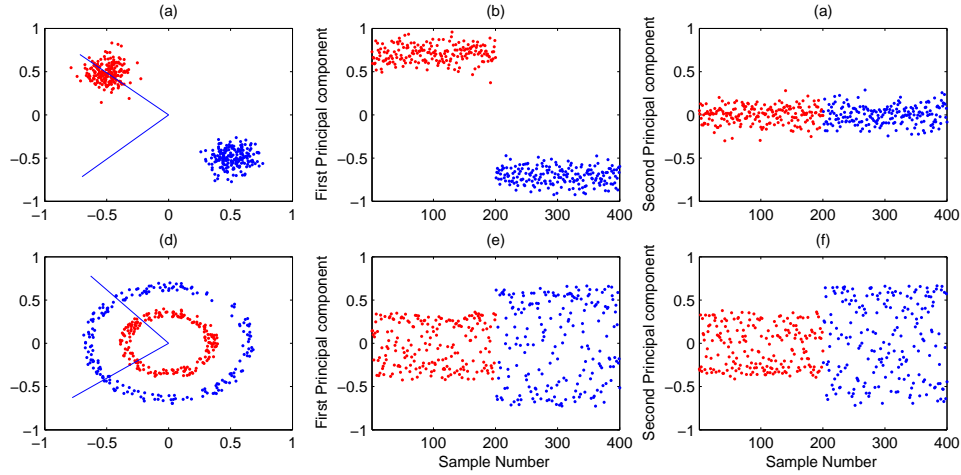


FIGURE 1. The first and the second principal components for two datasets.

separate the dataset into two clusters.

★ Does it matter which eigen values do we pick for clustering?

One thing to be noted at this point is that the eigen vectors which capture the maximum variance do not necessarily have the best discriminating power. In the previous example the third eigen vector had the best discriminatory power. This is natural since in the formulation of the PCA we did not address the issue of discrimination. We were only concerned with capturing the maximum variance. All the other methods which we discuss have the same problem. The question arises as to which eigen vectors to use. One way around this is to use as many eigen vectors as possible for clustering. The other question to be addressed is whether PCA or other methods can be formulated in such a way that discriminatory power is introduced. We can look at methods like Linear Discriminant Analysis and see if new methods of spectral clustering can be got.

★ How to decide the embedding?

However for any given dataset it is very difficult to specify what embedding will lead to a good clustering. I think that higher the dimension the clusters become more and more linear. In the limit when the data is embedded in infinite dimensional space no form of nonlinearity can exist (this is just a conjecture). However The computational complexity of performing PCA increases. The problem of finding the embedding still remains. These two problems are addressed by the nonlinear version of PCA called the Kernel PCA (KPCA).

Before that we look at PCA in a different way which naturally motivates KPCA. Solving the eigen value problem for the $d \times d$ covariance matrix S is

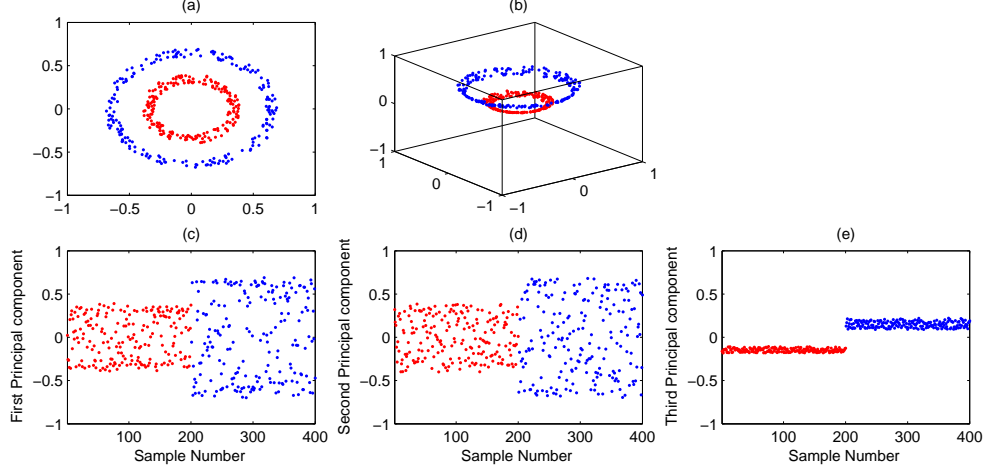


FIGURE 2. The first and the second principal components for two datasets.

of $O(d^3)$ complexity. However there are some applications where we have a few samples of very high dimensions (e.g. we have 100 256×256 face images, or we have embedded the data in a very high dimensional space). In such cases where $N < d$ we can formulate the problem in terms of an $N \times N$ matrix called as the dot product matrix. In fact this is the formulation which will naturally lead to KPCA in the next section. This method was first introduced by Kirby and Sirovitch² in 1987 and also used by Turk and Pentland³ in 1991 for face recognition. The problem can be formulated as follows. Just for notational simplicity define a new $d \times N$ matrix X where each column correspond to the d dimensional data point. Also assume that the data is centered.

According to our previous formulation for one eigen vector e corresponding to eigen value λ we have to solve,

$$\begin{aligned} Se &= \lambda e \\ XX^T e &= \lambda e \\ X^T X X^T e &= \lambda X^T e \\ K\alpha &= \lambda \alpha \end{aligned}$$

where $K = X^T X$ is called the dot product matrix or also the Gram matrix and $\alpha = X^T e$ is its eigen vector with eigen value λ . Note that K is an $N \times N$ matrix and hence we have only N eigen vectors, while d is typically

²L. Sirovitch and M. Kirby. Low dimensional procedure for the characterization of human faces. J. Opt. Soc. Am., 2:519-524, 1987.

³M. Turk and A. Pentland. Eigenfaces for recognition. Journal of Neuroscience, 3(1):71-86, 1991.

larger than N . We can compute e from the eigen vector α as follows,

$$X^T e = \alpha \Rightarrow X X^T e = X \alpha \Rightarrow S e = X \alpha \Rightarrow \lambda e = X \alpha \Rightarrow e = \frac{1}{\lambda} X \alpha$$

Note that the eigen vector e can be written as a linear combination of the data points X . Now we need the projection of all the data points on the eigen vector e . Let y be a $1 \times N$ row vector where each element is the projection of the datapoint on the eigen vector e .

$$y = e^T X = \left\{ \frac{1}{\lambda} X \alpha \right\}^T X = \frac{1}{\lambda} \alpha^T X^T X = \frac{1}{\lambda} \alpha^T K = \frac{1}{\lambda} \lambda \alpha^T = \alpha^T$$

★ The neat thing about this is that the elements of the eigen vectors of the matrix K , are the projection of the data points on the eigen vectors of the matrix S . This is the interpretation for why all the spectral clustering methods use the eigen vectors. They first form a dot product matrix (however they do some non-linear transformation and normalization) and then take the eigen vectors of the matrix. By this they are essentially projecting each data point on the eigen vectors of the covariance matrix of the datapoints in some other space (because of the gaussian kernel).

The PCA algorithm now becomes. Let $X = [X_1, X_2, \dots, X_N]$ be a $d \times N$ matrix where each column X_i is a point in some d -dimensional space.

- (1) Subtract the mean from all the data points. $X \leftarrow X(I - \frac{\mathbf{1}_{N \times N}}{N})$ where I is $N \times N$ identity matrix and $\mathbf{1}_{N \times N}$ is an $N \times N$ matrix with all entries equal to 1.
- (2) Form the dot product matrix $K = X^T X$.
- (3) Compute the N eigen vectors of the dot product matrix.
- (4) Each element of the eigen vector is the projection of the data point on the principal direction.

★ Note that by dealing with K we never have to deal with the eigen vectors e explicitly.

3. KERNEL PRINCIPAL COMPONENT ANALYSIS (KPCA)

The essential idea of KPCA ⁴ is based on the hope that if we do some non linear mapping of the data points to a higher dimensional (possibly infinite) space we can get better non linear features which are a more natural and compact representation of the data. The computational complexity arising from the high dimensionality mapping is mitigated by using the kernel trick. Consider a nonlinear mapping $\phi : R^d \rightarrow R^h$ from R^d the space of d dimensional data points to some higher dimensional space R^h . So now every point X_i is mapped to some point $\phi(X_i)$ in a higher dimensional space. Once we have this mapping KPCA is nothing but Linear PCA done on the points in

⁴A. Smola B. Scholkopf and K.-R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. Technical Report 44, Max-Planck-Institut für biologische Kybernetik, Tbingen, December 1996.

the higher dimensional space. However the beauty of KPCA is that we do not need to explicitly specify the mapping.

In order to use the kernel trick PCA should be formulated in terms of the dot product matrix as discussed in the previous section. Also as of now assume that the mapped data are zero centered (i.e $\sum_{i=1}^N \phi(X_i) = 0$). Although this is not a valid assumption as of now it simplifies our discussion. So now in our case the dot product matrix K becomes

$$[K]_{ij} = \phi(X_i) \cdot \phi(X_j).$$

In literature K is usually referred to as the Gram matrix. Solving the eigen value of K gives the corresponding N eigen vectors which are the projection of the datapoints on the eigen vectors of the covariance matrix.

As mentioned earlier we do not need the eigen vectors e explicitly (we need only the dot product). The kernel trick basically makes use of this fact and replaces the dot product by a kernel function which is more easy to compute than the dot product.

$$k(x, y) = \langle \phi(x) \cdot \phi(y) \rangle.$$

This allows us to compute the dot product without having to carry out the mapping. The question of which kernel function corresponds to a dot product in some space is answered by the Mercer's theorem which says that k is a continuous kernel of a positive integral operator then there exists some mapping where k will act as a dot product. The most commonly used kernels are the polynomial, Gaussian and the tanh kernel.

It can be shown that centering the data is equivalent to working with the new Gram matrix K' where,

$$K' = (I - \frac{\mathbf{1}_{N \times N}}{N})^T K (I - \frac{\mathbf{1}_{N \times N}}{N})$$

where I is $N \times N$ identity matrix and $\mathbf{1}_{N \times N}$ is an $N \times N$ matrix with all entries equal to 1.

The kernel PCA algorithm is as follows: Let $X = [X_1, X_2, \dots, X_N]$ be a $d \times N$ matrix where each column X_i is a point in some d -dimensional space.

- (1) Choose a appropriate kernel $k(x, y)$.
- (2) Form the $N \times N$ Gram matrix K where $[K]_{ij} = k(X_i, X_j)$.
- (3) Form the modified Gram matrix $K' = (I - \frac{\mathbf{1}_{N \times N}}{N})^T K (I - \frac{\mathbf{1}_{N \times N}}{N})$.
- (4) Compute the N eigen vectors of the dot product matrix.
- (5) Each element of the eigen vector is the projection of the data point on the principal direction in some higher dimensional transformed.

3.1. Different kernels used. The three most popular kernels are the Polynomial, Gaussian and the tanh kernel. The polynomial kernel of degree d is given by

$$k(x, y) = (x \cdot y + c)^d$$

where c is a constant whose choice depends on the range of the data. It can be seen that the polynomial kernel picks up correlations upto order d .

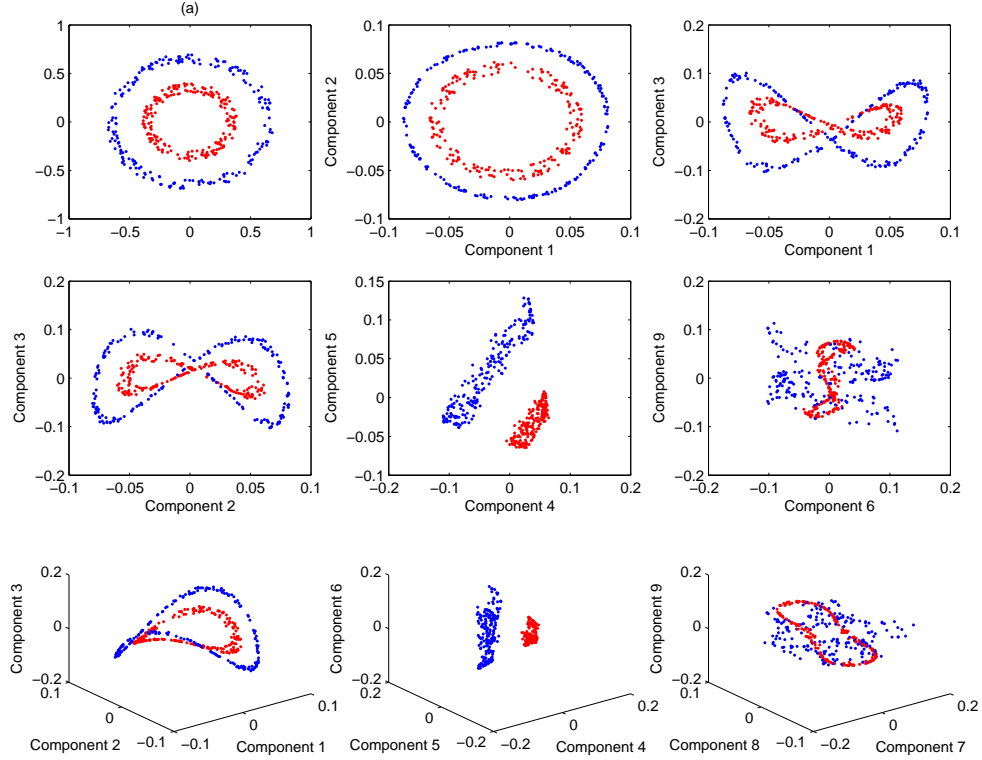


FIGURE 3. Different kernel principal components for the two circle data set. Gaussian kernel with $\sigma = 0.4$ was used.

The Gaussian or the radial basis function kernel is given by

$$k(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$$

where the parameter σ controls the support region of the kernel.

The tanh kernel which is mostly used in Neural network type application is given by

$$k(x, y) = \tanh((x \cdot y) + b)$$

This kernel is not actually positive definite but is sometimes used in practice.

★ Figure 3(a) shows our original data set and a few kernel principal components are shown. The gaussian kernel with $\sigma = 0.4$ was used. It can be seen that the fourth and the fifth have a very good discriminatory power. It should be reiterated again that the first few principal component may not have the highest discriminatory power. Ideally we would like to use as many eigen vectors as possible. It would be interesting to see if some sort of discriminatory power can be incorporated in the formulation itself.

★ The results are very sensitive to the choice of σ . This is expected since it completely changes the nature of the embedding. Methods to optimally choose σ have to be developed both for KPCA and spectral clustering. σ

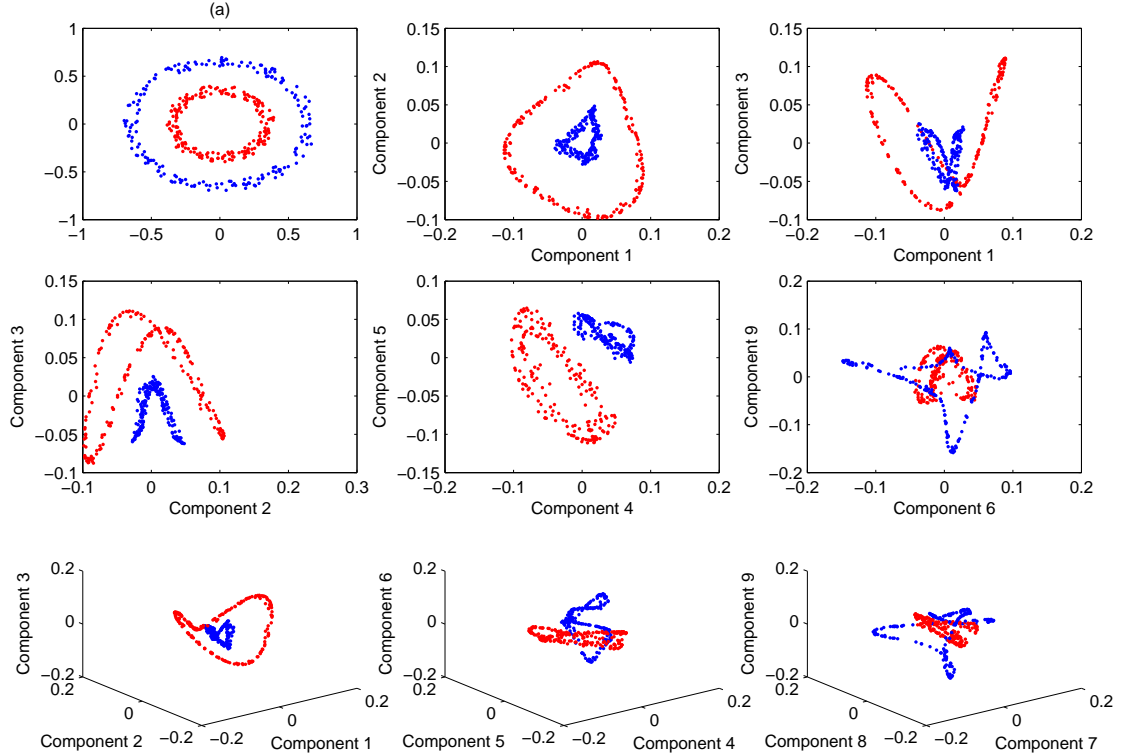


FIGURE 4. Different kernel principal components for the two circle data set. Gaussian kernel with $\sigma = 0.2$ was used.

depends on the nature of the data and the task at hand. Figure 4 shows the same for $\sigma = 0.2$. Note that the cluster shapes are very different.

★ The Gaussian kernel implicitly does the mapping. So we do not have a good feel of what kind of mapping it is doing. Also we cannot find the eigen vectors explicitly. Methods to somehow approximate the eigen vectors of the covariance matrix can be developed to get a better understanding of the nature of the mapping performed. What is the role of σ in the Gaussian kernel.

4. SPECTRAL CLUSTERING

A wide variety of spectral clustering methods have been proposed. All these methods can be summarized as follows, except that each method uses a different normalization and chooses to use the eigen vectors differently.

We have a set of N points in some d -dimensional space $X = X_1, X_2, \dots, X_N$.

- (1) For the affinity matrix A defined by $A_{ij} = e^{-\frac{\|X_i - X_j\|^2}{2\sigma^2}}$.
- (2) Normalize the affinity matrix to construct a new matrix L .

- (3) Find the k largest (or smallest depending on the type of normalization) eigen vectors e_1, e_2, \dots, e_k of L . Form the $N \times k$ matrix $Y = [e_1 \ e_2 \ \dots \ e_k]$ by stacking the eigen vectors in columns.
- (4) Normalize each row of Y to have unit length.
- (5) Treat each row of Y as a point in k -dimensional space and cluster them into k clusters via K -means or some other clustering algorithm.

The affinity matrix A is exactly the Gram matrix K which we used in the kernel PCA. The only difference we have to explore is the normalization step. So essentially spectral clustering methods are trying to embed the data points into some transformed space so that clustering will be easier in that space.

In KPCA we formed the modified gram matrix $K' = (I - \frac{1_{N \times N}}{N})^T K (I - \frac{1_{N \times N}}{N})$ in order to center the data in the transformed space. This normalization is equivalent to subtracting the row mean, the column mean and then adding the grand mean of the elements of K .

The following are some of the clustering methods we studied.

- Ng, Jordan and Weiss ⁵ : $L = D^{-1/2} A D^{-1/2}$ where D is a diagonal matrix whose $(i, i)^{th}$ -element is the sum of the A 's i^{th} row. $L_{ij} = \frac{A_{ij}}{\sqrt{D_{ii}} \sqrt{D_{jj}}}$.
- Shi and Malik ⁶: Solves the generalized eigenvalue problem $(D - W)e = \lambda De$. For segmentation uses the generalized eigen vector corresponding to the second smallest eigenvalue. If v is the eigen vector of L with eigen value λ , then $D^{-1/2}v$ is the generalized eigen vector of W with eigen value $1 - \lambda$. Hence this method is similar to the previous one expect that it uses only one eigen vector.
- Meila and Shi ⁷: The rows of A are normalized to sum to 1.
- Perona and Freeman ⁸ : Use only the first eigen vector of the affinity matrix A .

The best clustering methods uses the normalization $L_{ij} = \frac{A_{ij}}{\sqrt{D_{ii}} \sqrt{D_{jj}}}$ and uses as many eigen vectors as possible. Figure 5 and 6 show a few different eigen vectors of the matrix L using the method of Ng, Jordan and Weiss, for the same values of the Gaussian kernel used for KPCA. Note that clusters get separated, with the clusters better than that of the KPCA. This is probably because of the normalization step.

⁵On Spectral Clustering: Analysis and an algorithm, Andrew Y. Ng, Michael Jordan, and Yair Weiss. In NIPS 14,, 2002

⁶J. Shi and J. Malik. Normalized cuts and image segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'97), pages 731–737, 1997

⁷Marina Meila and Jianbo Shi. Learning segmentation by random walks. In Neural Information Processing Systems 13, 2001.

⁸P. Perona and W. T. Freeman. A factorization approach to grouping. In H. Burkhardt and B. Neumann, editors, Proc ECCV, pages 655–670, 1998.

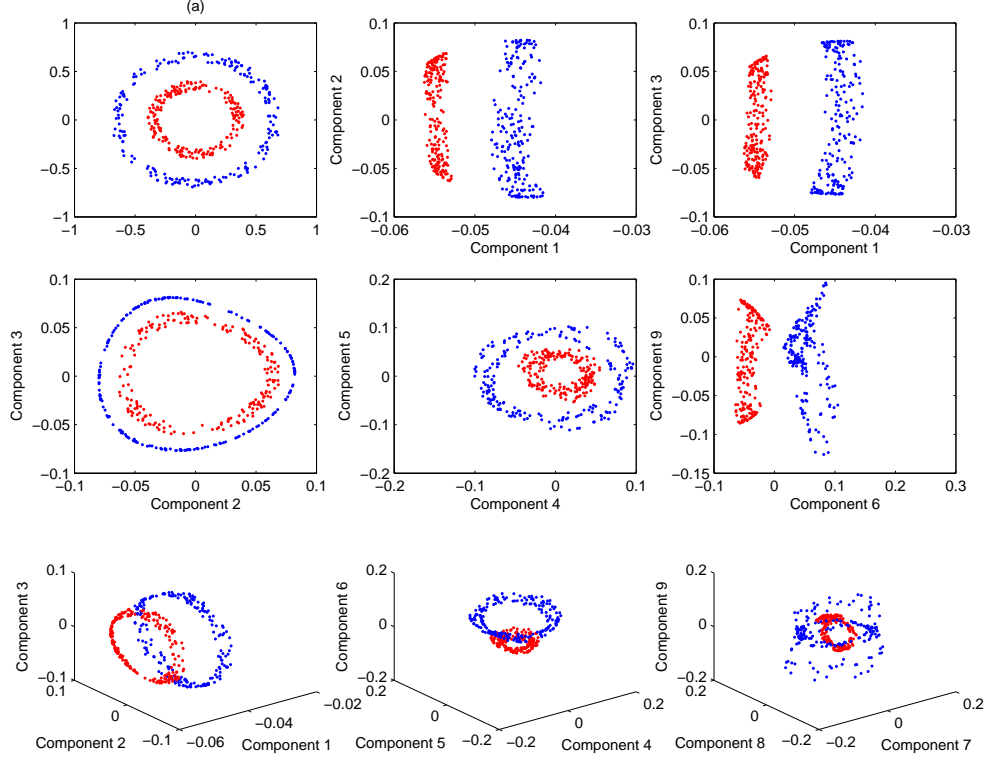


FIGURE 5. Different eigen vectors for the two circle data set. Gaussian kernel with $\sigma = 0.4$ was used.

5. SOME TOPICS TO BE PURSUED FURTHER

- It is noted that normalization can give dramatically good results. Can we explain the different normalization in terms of KPCA? Can we use these normalization for applications using KPCA?
- The algorithm is very sensitive to σ . How to choose the right scaling parameter? We can search over the scaling parameters to get the best clusters but finely should we search. The scaling parameter may not be related linearly with how tightly the clusters are clustered.
- What exactly is the nonlinear mapping performed by the Gaussian kernel? Can we explore different other kernels like in KPCA for spectral clustering?
- It is always advantageous to use as many eigen vectors as possible because we do not know which eigen vectors have the most discriminatory power. Experimentally it has been observed using more eigen vectors and directly computing a partition from all the eigen vectors

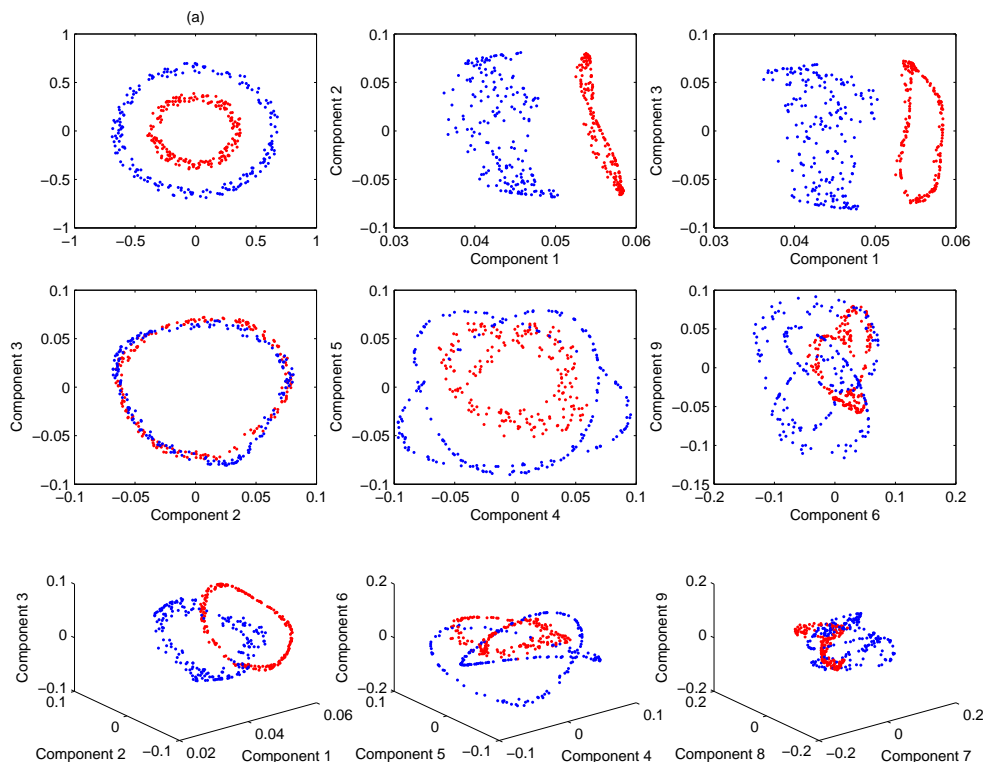


FIGURE 6. Different kernel principal components for the two circle data set. Gaussian kernel with $\sigma = 0.2$ was used.

gives better results⁹. This naturally fits in if we look at spectral clustering in terms of KPCA.

- The algorithms use k largest eigen vectors and then using K -means they find k clusters. I think the number of eigen vectors needed does not have such a direct relation with the number of clusters in the data. Nothing prevents us from using greater than or less than k eigen vectors.
- Normalized cut is formulated as solving a relaxation of a NP-hard discrete graph partitioning problem. While this is true, I think the power of nuts comes from using the gaussian kernel in defining the affinity matrix which reduces the non-linearity in the clusters.
- Relation of these methods in terms of the VC-dimension in statistical learning theory need to be studied. When exactly does mapping which take us into a higher dimensional space than the dimension of the input space provide us with greater classification power or clustering power? What is so unique about the Gaussian kernel.

⁹C. J. Alpert and S.-Z. Yao, "Spectral Partitioning: The More Eigenvectors, the Better," UCLA CS Dept. Technical Report 940036, October 1994

- One thing that puzzles me was that I found the spectral clustering methods give better performance. This is obviously due to the normalization they do. It would be nice to understand the effect of normalization in term of the kernel function.
- How are these methods related to Multidimensional Scaling and non-linear manifold learning techniques?
- In KPCA we can project any other point not belonging to the dataset. This performance on new samples is known as the generalization ability of the learner. It would a good idea to try spectral clustering on classification and recognition tasks and compare their generalization capacity with other state of the art methods.
- What are the differences in how spectral methods should be applied to different kinds of problems, such as dimensionality reduction, clustering and classification? Can we recast the problem or do we need a completely new formulation?
- How does spectral clustering methods relate to statistical methods¹⁰ like Bayesian inference, MRF etc.
- A more basic question to be understood is what are the limitations of the methods? What kind of nonlinearities each kernel can handle?
- How are these methods related to the Gibbs distribution in Markov random fields?

6. CONCLUSION

We have shown how KPCA and spectral clustering are trying to find a good nonlinear basis to the dataset, but differ in the normalization applied to the Gram matrix. However lot of open questions remain and I feel this is a good topic to pursue further research into machine learning.

E-mail address: vikas@cs.umd.edu

¹⁰Learning Segmentation with Random Walk Marina Maila and Jianbo Shi Neural Information Processing Systems, NIPS, 2001