CISC 271 Class 33

Nonlinear Separation – Kernel PCA

Hastie *et al.*, 2009 [6], pp. 547–550

kip.15in

Main Concepts:

- *Avoid directly embedding vectors*
- *Gram matrix has embedded dot products*
- *Kernel PCA: algorithm*

Sample Problem, Machine Inference: How can we separate data nonlinearly?

From previous classes, we understand that we can embed a vector $\vec{u} \in \mathbb{R}^n$ in a higher-dimensional space as $\hat{u} \in \mathbb{R}^p$, which performs $\vec{u} \hookrightarrow \hat{u}$. The transformation, or mapping, can be written as

$$\phi : \mathbb{R}^n \hookrightarrow \mathbb{R}^p \quad \text{or} \quad \hat{u} = \vec{\phi}(\vec{u}) \tag{33.1}$$

Let us recall how we have performed principal components analysis (PCA) thus far. For a data matrix $A \in \mathbb{R}^{m \times n}$, we first computed the zero-mean matrix $M \in \mathbb{R}^{m \times n}$ by finding the mean of each column as $\bar{a}_j$ and subtracting this mean from each entry of the respective column.

If we were to embed each observation in A, the result would be a matrix $A \in \mathbb{R}^{m \times p}$. It is unclear how we can use this matrix for clustering or for linear separation of the data. For example, a weight vector would be $\hat{w} \in \mathbb{R}^p$ and how we are to interpret this higher-dimensional vector may not be obvious.

Let us explore PCA in more detail. We will re-write the zero-mean matrix M, use an alternative scatter matrix that relies on observations, and then show that we can use the Gram matrix of a kernel function to elegantly avoid the computationally intensive process of vector embedding.

33.1 Principal Components Analysis And Scatter Of Observations

We can re-write the process of finding a zero-mean matrix M, using linear algebra, if we introduce a *centering* matrix G_m that depends on the number m of observations in our data.

This centering matrix can be derived by expanding the definition of the zero-mean matrix:

$$\begin{aligned}
\bar{A} &= \frac{1}{m}\vec{1}^{\,T}A \\
M &= A - \vec{1}\,\bar{A} \\
&= IA - \frac{1}{m}\vec{1}\,\vec{1}^{\,T}A \\
&= \left[I - \frac{1}{m}\vec{1}\,\vec{1}^{\,T}\right]A \\
&= G_m A \\
\Rightarrow \qquad G_m &\stackrel{\text{def}}{=} \left[I - \frac{1}{m}\vec{1}\,\vec{1}^{\,T}\right]
\end{aligned} \tag{33.2}$$

We can see that the centering matrix G_m of Equation 33.2 is symmetric and positive-semidefinite, having a rank of $m-1$ because it is the identity minus a rank-1 matrix.

The rank of the zero-mean matrix M is a number r which is no greater than the number of variables n, so we would say that

$$\text{rank}(M) = r \quad \text{where} \quad r \leq n \tag{33.3}$$

We defined the scatter matrix for PCA using the variables, which is

$$S_V = M^T M \tag{33.4}$$

The singular value decomposition (SVD) of the zero-mean matrix M can be written as

$$M = U\Sigma V^T \tag{33.5}$$

Equation 33.4 can be written, using Equation 33.5, as a spectral decomposition

$$\begin{aligned}
S_V &= V\Sigma^T\Sigma V^T \\
&= V\Lambda_V V^T
\end{aligned} \tag{33.6}$$

Combining Equation 33.6 with Equation 33.3, we know that the eigenvalue matrix Λ_V of S_V will have r non-zero leading diagonal entries, each entry being a positive real number.

In PCA, we produced each score vector $\vec{z}_j$ as the product of the zero-mean matrix M and the j^{th} loading vector $\vec{v}_j$, which is also the j^{th} right singular vector of the zero-mean matrix M, as $\vec{z}_j = M\vec{v}_j$. We can gather the score vectors into a score matrix Z that we can write concisely as

$$Z_v = MV = U\Sigma = U\Lambda_V^{1/2} \tag{33.7}$$

The eigenvalue matrix Λ_V and the singular-value matrix Σ are, from the derivation of the SVD, related as the square roots of their respective entries.

Consider the scatter matrix of the *observations*. We can write this as the "right-transpose" product of the zero-mean matrix M. This matrix $S_u \in \mathbb{R}^{m \times m}$ is the symmetric positive semidefinite matrix defined as

$$S_U = MM^T \tag{33.8}$$

Using Equation 33.8 and the SVD of the zero-mean matrix M, we can write

$$S_U = U\Lambda_U U^T \tag{33.9}$$

Because the rank of M is r, and because we have used the SVD of the zero-mean matrix M consistently, we can make a simple and remarkable observation:

The first r entries of Λ_V and Λ_U are identical

Note that the sizes of the original Λ_V and Λ_U – that is, the eigenvalue matrices including zero eigenvalues – are in general different. But, because of the SVD, the diagonal entries are the same up to the number r that is the rank of the zero-mean matrix M.

If we perform PCA by using the observation-style scatter matrix S_U, then the scores – gathered into a matrix Z_U – would be

$$Z_U = MU = U\Sigma = U\Lambda_U^{1/2} \tag{33.10}$$

Because Z_V in Equation 33.7 and Z_U in Equation 33.10 are equal, it does not matter mathematically whether we use the variable-style scatter matrix S_V to perform PCA, or the observation-style scatter matrix S_U. The resulting scores are identical.

Let us use the observation-style scatter matrix S_U and the centering matrix G_m that we derived in Equation 33.2. We can write the matrix S_U, using the centering matrix G_m, as

$$\begin{aligned} S_U &= MM^T \\ &= [G_m A][G_m A]^T \\ &= G_m[AA^T]G_m^T \end{aligned} \tag{33.11}$$

Equation 33.11 prepares us for the use of a kernel function in our data analysis.

33.2 Kernel Functions and the Gram Matrix

Now, let us return to the process of embedding a vector in a high-dimensional vector space. If we try to perform PCA using the embedding, we must:

- Embed each observation from $\mathbb{R}^n$ to $\mathbb{R}^p$
- Transform $A \hookrightarrow \hat{A}$
- Transform $M \hookrightarrow \hat{M}$
- Compute $\hat{S}_V \in \mathbb{R}^{p \times p}$

The dimension p can grow combinatorially from n, so this computation can rapidly become unwieldy. An alternative is to use the observation-style scatter matrix S_U. From Equation 33.11, we see that using S_U on our original data matrix A requires us to:

- Compute the inner-product matrix AA^T
- Center AA^T using G_m

The method of *kernel PCA* is the use of the Gram matrix for an observation-style scatter matrix S_U to perform PCA. The process that we can compute is:

- Define the kernel function $\kappa(\underline{a}_i, \underline{a}_j)$
- Compute the Gram matrix $\hat{W}_{ij} \stackrel{\text{def}}{=} \kappa(\underline{a}_i, \underline{a}_j)$ from the observations in the data matrix A
- Center the Gram matrix as the kernel observation-style scatter matrix $\hat{S}_U$, using the centering matrix G_m, as
$$\hat{S}_U = G_m W G_m^T \tag{33.12}$$
- Compute the singular values of Equation 33.12 in a matrix $\hat{\Sigma}$ and as the eigenvectors $\hat{u}_j$
- Compute the kernel PCA score vectors as
$$\hat{Z} = \hat{U}\hat{\Sigma} \tag{33.13}$$

We can see that each score vector in $\hat{Z}$ of Equation 33.13 has m entries, that is, $\hat{z}_j \in \mathbb{R}^m$. We can compute a score for each observation, but by using the Gram matrix $\hat{W}$ instead of using the observation-style scatter matrix S_U.

33.3 Example – Kernel PCA For Fisher's Iris Data

We can test our kernel PCA method on Fisher's Iris data set. First, let us try using a conventional method:

- Use PCA to find two score vectors, $\vec{z}_1$ and $\vec{z}_2$
- Use the `MATLAB` function `kmeans` to cluster the PCA scores

When we use this method, we see in Figure 33.1 that it incorrectly clusters two of the data vectors.

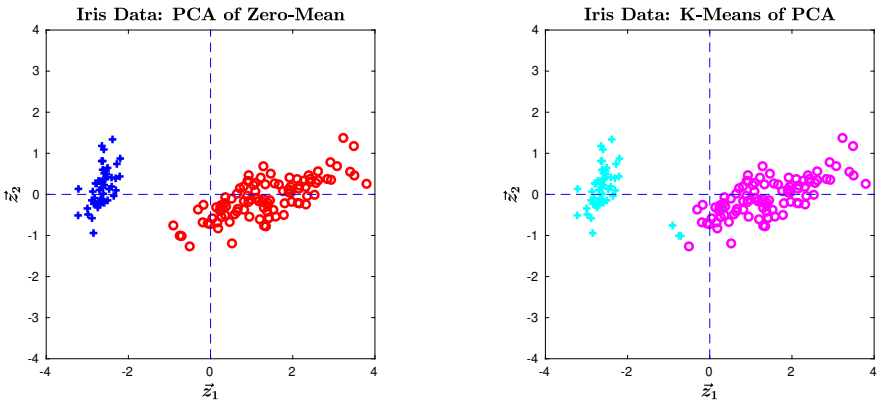

Figure 33.1: Fisher's Iris data set when processed using conventional PCA. (A) The data scored in 2D, with species *I. setosa* indicated as blue crosses and the other two species indicated as red circles. (B) Results of k-means clustering, showing that two data vectors are inappropriately clustered with the *I. setosa* data.

Next, let us try using kernel PCA. For the kernel function, we can use the Gaussian kernel. We need to select a hyper-parameter σ^2 for the Gaussian distribution; a guideline in data analysis is to first try the value and kernel that are

$$\begin{aligned} \sigma^2 &= m \\ \kappa(\underline{u}, \underline{v}) &= \exp\left(\frac{-\|\underline{u} - \underline{v}\|^2}{2\sigma^2}\right) \end{aligned} \tag{33.14}$$

We can calculate the Gram matrix $\hat{W}$ for the 4D Iris data, which will produce a 150×150 symmetric positive semidefinite matrix. The rank of the Gram matrix is substantially less than the size of the matrix, which is not unusual for a Gaussian kernel function.

We find the spectral decomposition of the observation-style scatter matrix $\hat{S}_U = G_m \hat{W} G_m$ by centering the Gram matrix $\hat{W}$; this gives us an orthogonal matrix $\hat{U}$ and eigenvalues $\hat{\lambda}_j$. To be

prudent, we can produce a scree plot of the eigenvalues. From Figure 33.2, either 2 or 3 would seem to be appropriate choices; we can select 2 as the number of scores that we will use, preferring to use the smallest dimension that effectively solves our problem.

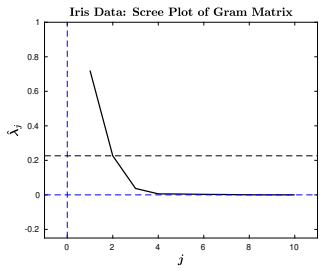

Figure 33.2: Scree plot of a Gram matrix, using a Gaussian kernel, for Fisher's Iris data set. We select 2 as the relevant number of scores for further use.

We can find the score vectors $\hat{z}_1$ and $\hat{z}_2$ from $\hat{S}_U$ and $\hat{U}$, and then repeat the plotting of Figure 33.1(A) with the same label indicators. Likewise, we can perform k-means clustering and plot the results, which are shown in Figure 33.3.

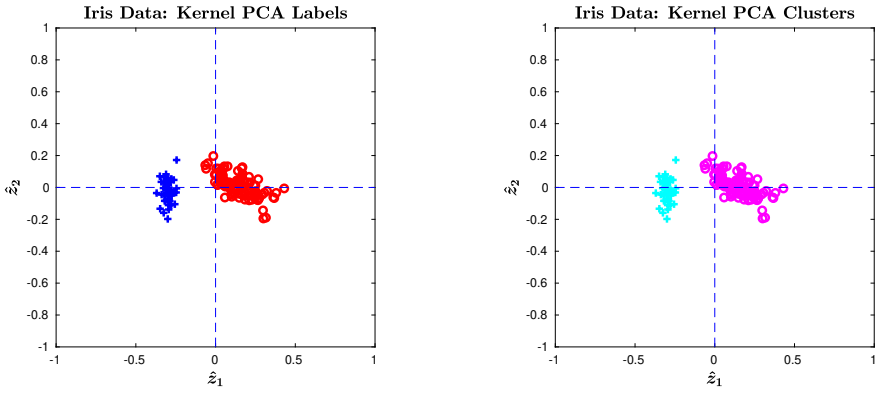

Figure 33.3: Fisher's Iris data set when processed using kernel PCA. (A) The data scored in 2D, with species *I. setosa* indicated as blue crosses and the other two species indicated as red circles. (B) Results of k-means clustering, showing that all data vectors are appropriately clustered.

We have found that using kernel PCA with a Gaussian kernel function is effective at finding clusters that match the data labels. Our analysis is now ready for us to explain to others.