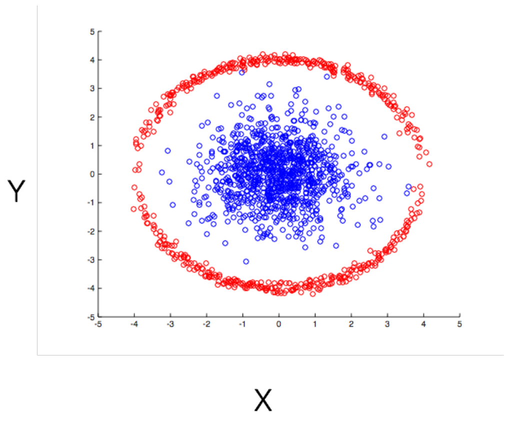
Machine Learning for Data Science (CS 4786)

Lecture 6: Kernel PCA

The text in black outlines high level ideas. The text in blue provides simple mathematical details to "derive" or get to the algorithm or method. The text in red are mathematical details for those who are interested.

1 Motivation

Consider the 2 dimensional data set distributed as follows:

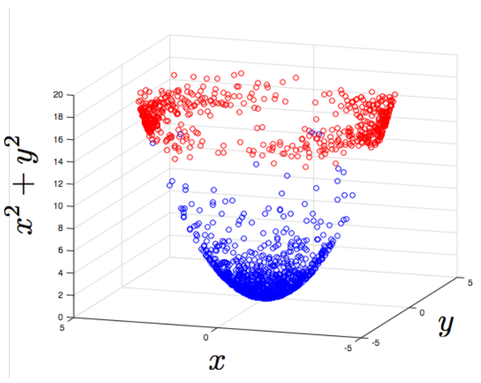


Any form of linear projection method to one dimension on this data will merge the red and blue points together thus losing information about the original dataset. This is because one needs to consider non-linear projection to one dimension to obtain separation of red and blue points.

The key idea that we shall explore in kernel methods have the following idea. Say instead of trying to reduce the original 2 dimensional points to one dimension via linear projection, we first write the data points as points in higher dimensional space. As an example, for this example, say we write every point $\mathbf{x}_t = (X_t, Y_t)$ into a 3 dimensional point given by mapping Φ as

$$\Phi(\mathbf{x}_t) = (X_t, Y_t, X_t^2 + Y_t^2)$$

In this 3 dimensional space, the set of points look like



In this case if instead of doing PCA on the original dataset we perform PCA on $\Phi(\mathbf{x}_1),\ldots,\Phi(\mathbf{x}_n)$, then PCA would pick direction that separates red and blue points. The basic idea in kernel PCA is to take the original data set, map them (implicitly) to higher dimensional space using mapping Φ and then perform PCA on this space which is linear projection in this higher dimensional space that already captures non-linearities.

2 The Kernel Trick

Consider a non-linear dimensionality reduction where the mapping Φ maps a vector $\mathbf{x}_t$ to higher dimension that captures all r^{th} order polynomials. As an example consider an example where $\mathbf{x}_t = (X_t, Y_t)$ is in 2 dimensions. If we want to represent all 3^{rd} degree polynomials for example, one needs to consider

$$\Phi(\mathbf{x}_t) = (X_t^3,\ Y_t^3,\ X_t^2 Y_t,\ Y_t^2 X_t,\ X_t^2,\ Y_t^2,\ X_t Y_t,\ X_t,\ Y_t,\ 1)$$

This mapped vector in higher dimension is often referred to as feature space. More generally, to capture all r^{th} degree polynomial of $\mathbf{x}_t$'s in d dimensions we need $\Phi(\mathbf{x}_t)$ to be order d^K dimensional. Further, in many interesting cases we might want to have Φ to possibly even map to infinite dimensions. In this case, first enumerating $\Phi(\mathbf{x}_t)$'s is not feasible. The kernel trick works on one simple idea:

Rewrite the algorithm/method so that it only needs access to inner products between data points. Next, replace these inner products with inner products in feature space.

Kernel trick relies on the fact that while we may not be able to explicitly write out the mappings $\Phi(\mathbf{x}_t)$, for any two $\mathbf{x}_t, \mathbf{x}_s$ we can compute using a kernel function k the inner product as $k(\mathbf{x}_t, \mathbf{x}_s) = \Phi(\mathbf{x}_t)^\top \Phi(\mathbf{x}_s)$.

Examples of kernel functions are $k(\mathbf{x}, \mathbf{x}') = \exp(-\sigma\|\mathbf{x}-\mathbf{x}'\|^2)$ or $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}')^p$.

As a specific example let us consider the case where $\mathbf{x}_t = (X_t, Y_t)$ and p some integer

$$k(\mathbf{x}_t, \mathbf{x}_s) = (\mathbf{x}_t^\top \mathbf{x}_s)^p = (X_t X_s + Y_t Y_s)^p = \sum_{i=0}^{p} \binom{p}{i} (X_t X_s)^i (Y_t Y_s)^{p-i}$$

where the last equality is using binomial theorem. Now notice that if one defines

$$\Phi((X,Y)) = \left(X^p, pYX^{p-1}, \ldots, \binom{p}{i} Y^i X^{p-i}, \ldots, Y^p\right)$$

then we have that

$$\Phi(\mathbf{x}_t)^\top \Phi(\mathbf{x}_s) = \sum_{i=0}^{p} \binom{p}{i} (X_t X_s)^i (Y_t Y_s)^{p-i} = k(\mathbf{x}_t, \mathbf{x}_s)$$

Now of course to perform Kernel PCA what we need to do is to write PCA such that it only depends on inner products.

3 Rewriting PCA in Terms of inner products

In this section we rewrite PCA completely in terms of inner product. Specifically we show how to compute lower dimensional representation $\mathbf{y}_1, \ldots, \mathbf{y}_n$ only in terms of inner products between data point.

3.1 PCA for Centered Data

We start this subsection assuming that data is centered. That is $\frac{1}{n}\sum_{t=1}^{n} \mathbf{x}_t = \mathbf{0}$. Recall that PCA finds projection matrix W such that the k^{th} column of W (say W_k) is the k^{th} eigen vector of the covariance matrix. Hence,

$$\begin{aligned}
\lambda_k W_k &= \Sigma W_k \\
&= \left(\frac{1}{n}\sum_{t=1}^{n} \mathbf{x}_t \mathbf{x}_t^\top\right) W_k \\
&= \left(\frac{1}{n}\sum_{t=1}^{n} \mathbf{x}_t \mathbf{x}_t^\top W_k\right) \\
&= \frac{1}{n}\sum_{t=1}^{n} \left(\mathbf{x}_t^\top W_k\right) \mathbf{x}_t
\end{aligned}$$

Hence if we let $\alpha_k[t] = \frac{1}{\lambda_k n}\mathbf{x}_t^\top W_k$, then we see that,

$$W_k = \sum_{t=1}^{n} \alpha_k[t] \mathbf{x}_t$$

that is a linear combination of the (centered) data points. Now using the above we notice that

$$\mathbf{y}_s[k] = \mathbf{x}_s^\top W_k = \mathbf{x}_s^\top \left(\sum_{t=1}^{n} \alpha_k[t] \mathbf{x}_t\right) = \sum_{t=1}^{n} \alpha_k[t] \mathbf{x}_s^\top \mathbf{x}_t$$

Form this we conclude that if data were centered and we knew α_k's then we can compute the lower dimensional projections only using inner products. Now let us turn our attention to computing α_k's. To do this we start with our earlier observations that $\alpha_k[s] = \frac{1}{\lambda_k n}\mathbf{x}_s^\top W_k$ and subsequently plug in the fact that $W_k = \sum_{t=1}^{n} \alpha_k[t]\mathbf{x}_t$. Hence

$$\begin{aligned}\alpha_k[s] &= \frac{1}{\lambda_k n}\mathbf{x}_s^\top W_k \\ &= \frac{1}{\lambda_k n}\mathbf{x}_s^\top \sum_{t=1}^{n} \alpha_k[t]\mathbf{x}_t \\ &= \frac{1}{\lambda_k n}\sum_{t=1}^{n} \alpha_k[t]\mathbf{x}_s^\top \mathbf{x}_t\end{aligned}$$

If we let $\tilde{K}$ be the matrix such that $\tilde{K}_{t,s} = \mathbf{x}_s^\top \mathbf{x}_t$ then we see that,

$$\alpha_k[s] = \frac{1}{\lambda_k n}\sum_{t=1}^{n} \alpha_k[t]\tilde{K}_{s,t} = \frac{1}{\lambda_k n}\alpha_k^\top \tilde{K}_s$$

Hence we can conclude that

$$\alpha_k = \frac{1}{\lambda_k n}\tilde{K}\alpha_k$$

That is each α_k is in the direction of the k^{th} eigen vector of the matrix $\tilde{K}$ (that it is k^{th} requires us to observe that the non-zero eigenvalues of $\tilde{K}$ and Σ match). This determines the direction. Now to get a handle on magnitude we note that $\|W_k\|^2 = 1$ as its an eigenvector. Hence

$$1 = \|W_k\|^2 = W_k^\top W_k = \left(\sum_{t=1}^{n} \alpha_k[t]\mathbf{x}_t\right)^\top \sum_{t=1}^{n} \alpha_k[t]\mathbf{x}_t = \alpha_k^\top \tilde{K}\alpha_k$$

However since α_k is in the direction of the k^{th} eigenvector of $\tilde{K}$ whose corresponding eigenvalue is say γ_k, we have that

$$1 = \alpha_k^\top \tilde{K}\alpha_k = \gamma_k \alpha_k^\top \alpha_k$$

Hence we conclude that $\|\alpha_k\| = \frac{1}{\sqrt{\gamma_k}}$.

Thus for centered data we can compute α_k's by computing the top K eigenvectors of $\tilde{K}$ and normalizing the k^{th} eigenvector by dividing through by $1/\sqrt{\gamma_k}$. Subsequently we compute the lower dimensional projections as:

$$\mathbf{y}_s[k] = \sum_{t=1}^{n} \alpha_k[t]\mathbf{x}_s^\top \mathbf{x}_t = \alpha_k^\top \tilde{K}_s$$

3.2 Centering Data

In the above section we showed that for already centred data, we can perform PCA with only knowing the matrix $\tilde{K}$. However the crucial assumption there was that we had centered data. To perform PCA only based on inner products we need to be able to show that we can compute $\tilde{K}$ matrix only based on inner products. Here we use the following simple calculation.

$$\begin{aligned}
\tilde{K}_{s,t} &= \left(\mathbf{x}_t - \frac{1}{n}\sum_{u=1}^{n}\mathbf{x}_u\right)^{\top}\left(\mathbf{x}_s - \frac{1}{n}\sum_{u=1}^{n}\mathbf{x}_u\right) \\
&= \mathbf{x}_t^{\top}\mathbf{x}_s - \left(\frac{1}{n}\sum_{u=1}^{n}\mathbf{x}_u\right)^{\top}\mathbf{x}_s - \left(\frac{1}{n}\sum_{u=1}^{n}\mathbf{x}_u\right)^{\top}\mathbf{x}_t \\
&\quad + \frac{1}{n^2}\left(\sum_{u=1}^{n}\mathbf{x}_u\right)^{\top}\left(\sum_{v=1}^{n}\mathbf{x}_v\right) \\
&= \mathbf{x}_t^{\top}\mathbf{x}_s - \frac{1}{n}\sum_{u=1}^{n}\mathbf{x}_u^{\top}\mathbf{x}_s - \frac{1}{n}\sum_{u=1}^{n}\mathbf{x}_u^{\top}\mathbf{x}_t + \frac{1}{n^2}\sum_{u=1}^{n}\sum_{v=1}^{n}\mathbf{x}_u^{\top}\mathbf{x}_v \quad = \mathrm{Kern}_{t,s} - \frac{1}{n}\sum_{u=1}^{n}\mathrm{Kern}_{u,s} - \frac{1}{n}\sum_{u=1}^{n}\mathrm{Kern}_{u,t} + \frac{1}{n^2}\sum_{u=1}^{n}\sum
\end{aligned}$$

where $\mathrm{Kern}_{s,t} = \mathbf{x}_t^{\top}\mathbf{x}_s$ where $\mathbf{x}$'s this time are not centered. Thus we can see that once can compute the centered inner product matrix $\tilde{K}$ based only on the uncentered inner product matrix K. The above equation can we succinctly written in matrix form as

$$\tilde{K} = \mathrm{Kern} - \frac{(\mathbf{1}_{n\times n} \times \mathrm{Kern})}{n} - \frac{(\mathrm{Kern} \times \mathbf{1}_{n\times n})}{n} + \frac{(\mathbf{1}_{n\times n} \times \mathrm{Kern} \times \mathbf{1}_{n\times n})}{n^2}$$

where $\mathbf{1}_{n\times n}$ is teh all ones matrix.

4 Kernel PCA

Now to perform Kernel PCA is simple we simply replace inner products with kernel functions. That is $\mathrm{Kern}_{s,t} = k(\mathbf{x}_t, \mathbf{x}_s)$ and then perform PCA only based on inner products as shown in the slides for kernel PCA.