# Kernel Principal Components Analysis

**Max Welling**
Department of Computer Science
University of Toronto
10 King's College Road
Toronto, M5S 3G5 Canada
*welling@cs.toronto.edu*

## Abstract

This is a note to explain kPCA.

## 1 PCA

Let's fist see what PCA is when we do not worry about kernels and feature spaces. We will always assume that we have centered data, i.e. $\sum_i \mathbf{x}_i = 0$. This can always be achieved by a simple translation of the axis.

Our aim is to find meaningful projections of the data. However, we are facing an unsupervised problem where we don't have access to any labels. If we had, we should be doing Linear Discriminant Analysis. Due to this lack of labels, our aim will be to find the subspace of largest variance, where we choose the number of retained dimensions beforehand. This is clearly a strong assumption, because it may happen that there is interesting signal in the directions of small variance, in which case PCA in not a suitable technique (and we should perhaps use a technique called independent component analysis). However, usually it is true that the directions of smallest variance represent uninteresting noise.

To make progress, we start by writing down the sample-covariance matrix $C$,

$$C = \frac{1}{N}\sum_i \mathbf{x}_i \mathbf{x}_i^T \tag{1}$$

The eigenvalues of this matrix represent the variance in the eigen-directions of data-space. The eigen-vector corresponding to the largest eigenvalue is the direction in which the data is most stretched out. The second direction is orthogonal to it and picks the direction of largest variance in that orthogonal subspace etc. Thus, to reduce the dimensionality of the data, we project the data onto the retained eigen-directions of largest variance:

$$U\Lambda U^T = C \quad \Rightarrow \quad C = \sum_a \lambda_a \mathbf{u}_a \mathbf{u}_a^T \tag{2}$$

and the projection is given by,

$$\mathbf{y}_i = U_k^T \mathbf{x}_i \quad \forall i \tag{3}$$

where $U_k$ means the $d \times k$ sub-matrix containing the first $k$ eigenvectors as columns. As a side effect, we can now show that the projected data are de-correlated in this new basis:

$$\frac{1}{N}\sum_i \mathbf{y}_i \mathbf{y}_i^T = \frac{1}{N}\sum_i U_k^T \mathbf{x}_i \mathbf{x}_i^T U_k = U_k^T C U_k = U_k^T U \Lambda U^T U_k^T = \Lambda_k \tag{4}$$

where $\Lambda_k$ is the (diagonal) $k \times k$ sub-matrix corresponding to the largest eigenvalues.

Another convenient property of this procedure is that the reconstruction error in $L_2$-norm between from $\mathbf{y}$ to $\mathbf{x}$ is minimal, i.e.

$$\sum_i ||\mathbf{x}_i - \mathcal{P}_k \mathbf{x}_i||^2 \tag{5}$$

where $\mathcal{P}_k = U_k U_k^T$ is the projection onto the subspace spanned by the columns of $U_k$, is minimal.

Now imagine that there are more dimensions than data-cases, i.e. some dimensions remain unoccupied by the data. In this case it is not hard to show that the eigen-vectors that span the projection space must lie in the subspace spanned by the data-cases. This can be seen as follows,

$$\lambda_a \mathbf{u}_a = C\mathbf{u}_a = \frac{1}{N}\sum_i \mathbf{x}_i \mathbf{x}_i^T \mathbf{u}_a = \frac{1}{N}\sum_i (\mathbf{x}_i^T \mathbf{u}_a)\mathbf{x}_i \;\Rightarrow\; \mathbf{u}_a = \sum_i \frac{(\mathbf{x}_i^T \mathbf{u}_a)}{N\lambda_a}\mathbf{x}_i = \sum_i \alpha_i^a \mathbf{x}_i \tag{6}$$

where $\mathbf{u}_a$ is some arbitrary eigen-vector of $C$. The last expression can be interpreted as: “every eigen-vector can be exactly written (i.e. losslessly) as some linear combination of the data-vectors, and hence it must lie in its span”. This also implies that instead of the eigenvalue equations $C\mathbf{u} = \lambda\mathbf{u}$ we may consider the $N$ projected equations $\mathbf{x}_i^T C\mathbf{u} = \lambda \mathbf{x}_i^T \mathbf{u}\ \forall i$. From this equation the coefficients $\alpha_i^a$ can be computed efficiently a space of dimension $N$ (and not $d$) as follows,

$$\begin{aligned}
\mathbf{x}_i^T C\mathbf{u}_a &= \lambda_a \mathbf{x}_i^T \mathbf{u}_a \quad \Rightarrow \\
\mathbf{x}_i^T \frac{1}{N}\sum_k \mathbf{x}_k \mathbf{x}_k^T \sum_j \alpha_j^a \mathbf{x}_j &= \lambda_a \mathbf{x}_i^T \sum_j \alpha_j^a \mathbf{x}_j \quad \Rightarrow \\
\frac{1}{N}\sum_{j,k} \alpha_j^a [\mathbf{x}_i^T \mathbf{x}_k][\mathbf{x}_k^T \mathbf{x}_j] &= \lambda_a \sum_j \alpha_j^a [\mathbf{x}_i^T \mathbf{x}_j]
\end{aligned} \tag{7}$$

We now rename the matrix $[\mathbf{x}_i^T \mathbf{x}_j] = K_{ij}$ to arrive at,

$$K^2 \boldsymbol{\alpha}^a = N\lambda_a K\boldsymbol{\alpha}^a \quad \Rightarrow \quad K\boldsymbol{\alpha}^a = (\tilde{\lambda}_a)\boldsymbol{\alpha}^a \quad \text{with} \quad \tilde{\lambda}_a = N\lambda_a \tag{8}$$

So, we have derived an eigenvalue equation for $\boldsymbol{\alpha}$ which in turn completely determines the eigenvectors $\mathbf{u}$. By requiring that $\mathbf{u}$ is normalized we find,

$$\mathbf{u}_a^T \mathbf{u}_a = 1 \quad \Rightarrow \quad \sum_{i,j} \alpha_i^a \alpha_j^a [\mathbf{x}_i^T \mathbf{x}_j] = \boldsymbol{\alpha}_a^T K \boldsymbol{\alpha}_a = N\lambda_a \boldsymbol{\alpha}_a^T \boldsymbol{\alpha}_a = 1 \quad \Rightarrow \quad ||\boldsymbol{\alpha}_a|| = 1/\sqrt{N\lambda_a} \tag{9}$$

Finally, when we receive a new data-case $\mathbf{t}$ and we like to compute its projections onto the new reduced space, we compute,

$$\mathbf{u}_a^T \mathbf{t} = \sum_i \alpha_i^a \mathbf{x}_i^T \mathbf{t} = \sum_i \alpha_i^a K(\mathbf{x}_i, \mathbf{t}) \tag{10}$$

This equation should look familiar, it is central to most kernel methods.

Obviously, the whole exposition was setup so that in the end we only needed the matrix $K$ to do our calculations. This implies that we are now ready to kernelize the procedure by replacing $\mathbf{x}_i \rightarrow \Phi(\mathbf{x}_i)$ and defining $K_{ij} = \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_j)^T$, where $\Phi(\mathbf{x}_i) = \Phi_{ia}$.

## 2 Centering Data in Feature Space

It is in fact very difficult to explicitly center the data in feature space. But, we know that the final algorithm only depends on the kernel matrix, so if we can center the kernel matrix we are done as well. A kernel matrix is given by $K_{ij} = \Phi_i \Phi_j^T$. We now center the features using,

$$\Phi_i = \Phi_i - \frac{1}{N} \sum_k \Phi_k \tag{11}$$

Hence the kernel in terms of the new features is given by,

$$\begin{aligned} K_{ij}^c &= (\Phi_i - \frac{1}{N} \sum_k \Phi_k)(\Phi_j - \frac{1}{N} \sum_l \Phi_l)^T && (12) \\ &= \Phi_i \Phi_j^T - [\frac{1}{N} \sum_k \Phi_k] \Phi_j^T - \Phi_i [\frac{1}{N} \sum_l \Phi_l^T] + [\frac{1}{N} \sum_k \Phi_k][\frac{1}{N} \sum_l \Phi_l^T] && (13) \\ &= K_{ij} - \boldsymbol{\kappa}_i \mathbf{1}_j^T - \mathbf{1}_i \boldsymbol{\kappa}_j^T + k \mathbf{1}_i \mathbf{1}_j^T && (14) \\ \text{with} \quad & \boldsymbol{\kappa}_i = \frac{1}{N} \sum_k K_{ik} && (15) \\ \text{and} \quad & k = \frac{1}{N^2} \sum_{ij} K_{ij} && (16) \end{aligned}$$

Hence, we can compute the centered kernel in terms of the non-centered kernel alone and no features need to be accessed.

At test-time we need to compute,

$$K_c(\mathbf{t}_i, \mathbf{x}_j) = [\Phi(\mathbf{t}_i) - \frac{1}{N} \sum_k \Phi(\mathbf{x}_k)][\Phi(\mathbf{x}_j) - \frac{1}{N} \sum_l \Phi(\mathbf{x}_l)]^T \tag{17}$$

Using a similar calculation (left for the reader) you can find that this can be expressed easily in terms of $K(\mathbf{t}_i, \mathbf{x}_j)$ and $K(\mathbf{x}_i, \mathbf{x}_j)$ as follows,

$$K_c(\mathbf{t}_i, \mathbf{x}_j) = K(\mathbf{t}_i, \mathbf{x}_j) - \boldsymbol{\kappa}(\mathbf{t}_i) \mathbf{1}_j^T - \mathbf{1}_i \boldsymbol{\kappa}(\mathbf{x}_j)^T + k \mathbf{1}_i \mathbf{1}_j^T \tag{18}$$