Machine Learning for Data Science (CS4786)
Lecture 7

Non-Linear Dimensionality Reduction

Feb 23, 2016

Course Webpage :
http://www.cs.cornell.edu/Courses/cs4786/2016sp/

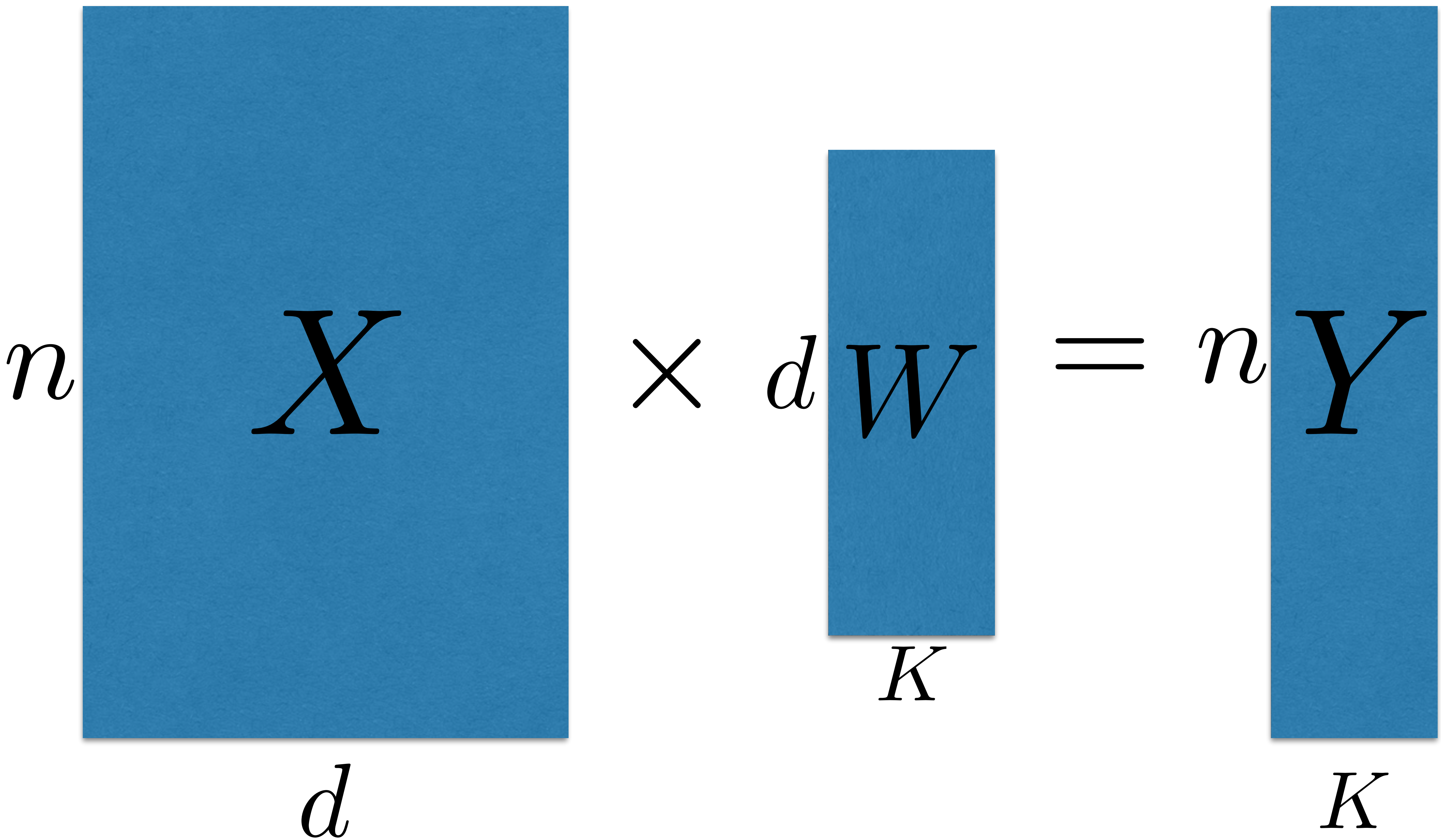
Announcement

- Students added to CMS according to assignment 0.
  - If you submitted Assignment 0 and were enrolled but not added to CMS email either me or TA's
  - Collect commented assignments back from the hand-back room.

Announcement

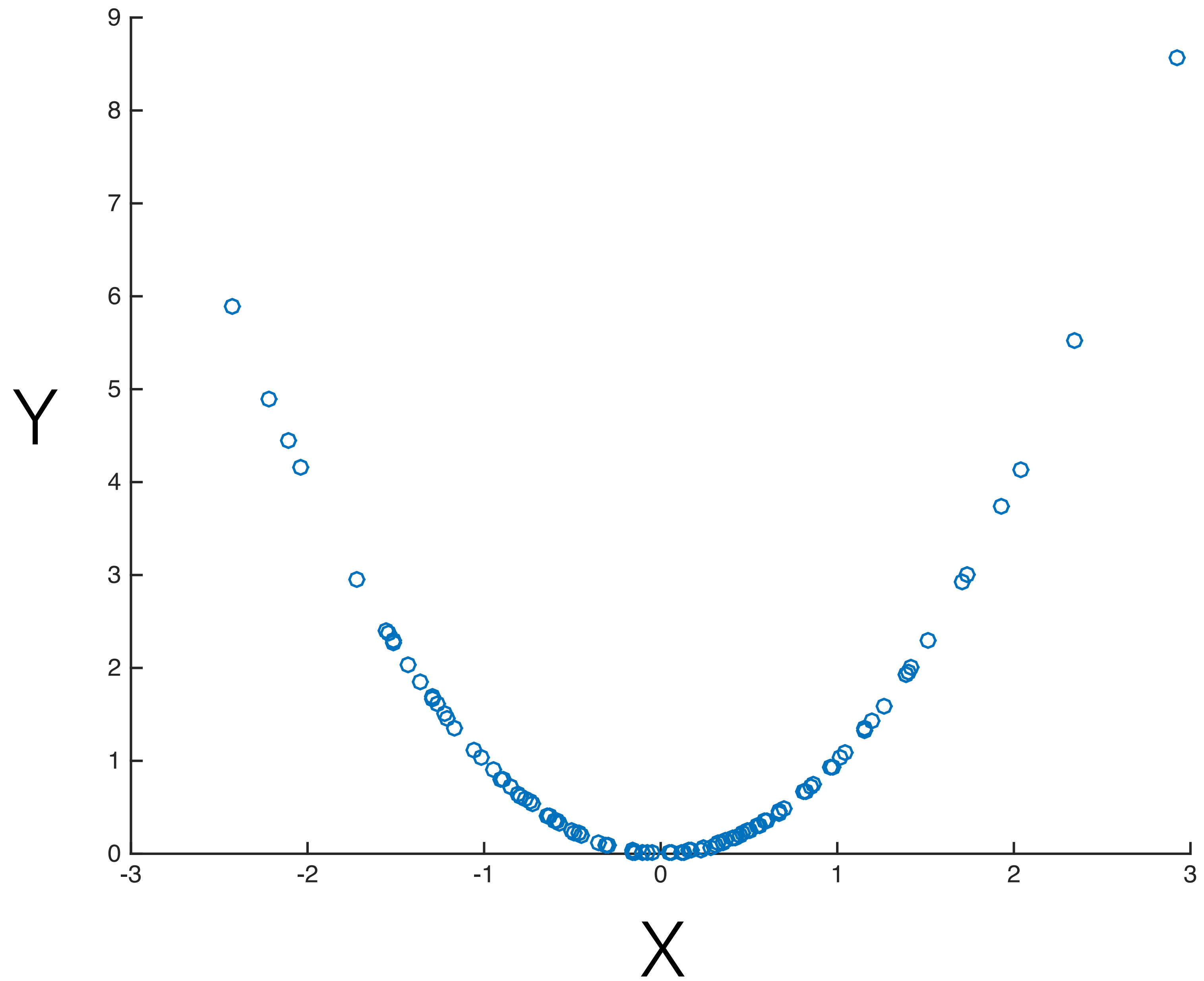
- Assignment 1 is out. Due on 7th March, 2016 at 11:59PM.
- Form groups early in CMS and get started early!
- Three questions, one on PCA, one on CCA and one on random projection Vs PCA.
- `Hw1.pdf` has the questions and instructions
- `a1.zip` has all the data files.
- ...

Linear Projections



Works when data lies in a low dimensional linear sub-space

EXAMPLE



Data is indeed one dimensional!

Linear Projections (Right Co-ordinates)

Demo

A First Cut

- Given $\mathbf{x}_t \in \mathbb{R}^d$, vector

$$\Phi(\mathbf{x}_t) = (\mathbf{x}_t[1], \ldots, \mathbf{x}_t[d], \mathbf{x}_t[1] \cdot \mathbf{x}_t[1], \mathbf{x}_t[1] \cdot \mathbf{x}_t[2], \ldots, \mathbf{x}_t[d] \cdot \mathbf{x}_t[d], \ldots)^\top$$

- Enumerating products up to order K (ie. products of at most K coordinates) we can get degree K polynomial non-linearity.

- However dimension blows up as d^K

- Is there a way to do this without enumerating Φ?

Kernel Trick

- Essence of Kernel trick:
  - **If** we can write down an algorithm only in terms of $\Phi(\mathbf{x}_t)^\top \Phi(\mathbf{x}_s)$ for data points $\mathbf{x}_t$ and $\mathbf{x}_s$
  - **Then** we don't need to explicitly enumerate $\Phi(\mathbf{x}_t)$'s but instead, compute $k(\mathbf{x}_t, \mathbf{x}_s) = \Phi(\mathbf{x}_t)^\top \Phi(\mathbf{x}_s)$ (even if Φ maps to infinite dimensional space)
- Example: RBF kernel $k(\mathbf{x}_t, \mathbf{x}_s) = \exp(-\sigma \|\mathbf{x}_t - \mathbf{x}_s\|_2^2)$
- Kernel function measures similarity between points.

Lets Rewrite PCA

- k^{th} column of W is eigenvector of covariance matrix
  That is, $\lambda_k W_k = \Sigma W_k$. Rewriting, for centered X

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

W_k's can be written as linear combination of $\mathbf{x}_t$'s, as

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

where $\alpha_k[t] = \frac{1}{\lambda_k n}\left(\mathbf{x}_t^\top W_k\right)$

Lets Rewrite PCA

- First note that $\mathbf{y}_i[k] = \mathbf{x}_i^\top W_k = \sum_{t=1}^n \alpha_k[t]\mathbf{x}_i^\top \mathbf{x}_t$
- For any $i, k \in [n]$, $\lambda_k \mathbf{x}_i^\top W_k = \mathbf{x}_i^\top \Sigma W_k$ and so,

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

LETS REWRITE PCA

- Let $\tilde{K}$ be an $n \times n$ matrix such that

$$\tilde{K}_{s,t} = \mathbf{x}_s^\top \mathbf{x}_t$$

- Rewriting from previous slide,

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

In other words, $\lambda_k \tilde{K} \boldsymbol{\alpha}_k = \frac{1}{n}\left(\tilde{K} \times \tilde{K}\right) \boldsymbol{\alpha}_k$ and so, $\lambda_k \boldsymbol{\alpha}_k = \frac{1}{n} \tilde{K} \boldsymbol{\alpha}_k$. That is, $\boldsymbol{\alpha}_k$ is in the direction of the eigen vector of $\tilde{K}$.

Lets Rewrite PCA

- Further, since W_k is unit norm,

$$1 = \|W_k\|_2^2 = \left(\sum_{t=1}^{n} \boldsymbol{\alpha}_k[t]\mathbf{x}_t\right)^{\top}\left(\sum_{s=1}^{n} \boldsymbol{\alpha}_k[s]\mathbf{x}_s\right) = {\boldsymbol{\alpha}_k}^{\top}\tilde{K}\boldsymbol{\alpha}_k = n\lambda_k{\boldsymbol{\alpha}_k}^{\top}\boldsymbol{\alpha}_k$$

Hence $\|\boldsymbol{\alpha}_k\|^2 = \frac{1}{n\lambda_k}$ which is the inverse of eigen value of $\tilde{K}$

- Finally notice that

$$\mathbf{y}_i[k] = \mathbf{x}_i^{\top}W_k = \sum_{t=1}^{n} \boldsymbol{\alpha}_k[t]\tilde{K}_{t,i}$$

PCA REWRITTEN

- If we only need to compute projections of data points, its enough to have access to matrix $\tilde{K}$ (a $n \times n$ matrix)
- Projection computed by computing top eigen vectors of $\tilde{K}$
- Rescaling them by inverse of the square-root of corresponding eigen values
- Projection obtained by taking product of the top eigen vectors with matrix $\tilde{K}$

Kernel PCA

- If we want to port PCA to kernel PCA, we need to be able to write $\tilde{K}$ in terms of kernel functions.
- We assumed centered data, so

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

- Knowing kernel function, we can perform Kernel PCA even when Φ maps to infinite dimensional space!

Kernel PCA

1. $\underbrace{\tilde{K}}_{n}\; n = \mathrm{compute}\left(k, X\right)$

2. $\left[A, \lambda\right] = \mathrm{eigs}\left(\tilde{K}, K\right)$

3. $P = \left[\dfrac{A_1}{\sqrt{\lambda_1}} \cdots \dfrac{A_K}{\sqrt{\lambda_K}}\right]$

Kernel PCA

4. $Y = \tilde{K} \times P$

Demo