Canonical Correlation Analysis

+ Kernel PCA
Audio might have background sounds uncorrelated with video

Video might have lighting changes uncorrelated with audio

Redundant information between two views: the speech
Example II: Combining Feature Extractions

- Method A and Method B are both equally good feature extraction techniques.
Method A and Method B are both equally good feature extraction techniques.

Concatenating the two features blindly yields large dimensional feature vector with redundancy.
Method A and Method B are both equally good feature extraction techniques.

Concatenating the two features blindly yields large dimensional feature vector with redundancy.

Applying techniques like CCA extracts the key information between the two methods.
Method A and Method B are both equally good feature extraction techniques.

Concatenating the two features blindly yields large dimensional feature vector with redundancy.

Applying techniques like CCA extracts the key information between the two methods.

Removes extra unwanted information.
Canonical Correlation Analysis
Canonical Correlation Analysis

Age
+ Gender
+ Candies per week
+ Favorite Cartoon
Data comes in pairs \((x_1, x'_1), \ldots, (x_n, x'_n)\) where \(x_t\)'s are \(d\) dimensional and \(x'_t\)'s are \(d'\) dimensional.

Goal: Compress say view one into \(y_1, \ldots, y_n\) that are \(K\) dimensional vectors.

Retain information redundant between the two views.

Eliminate "noise" specific to only one of the views.
Data comes in pairs \((x_1, x'_1), \ldots, (x_n, x'_n)\) where \(x_t\)'s are \(d\) dimensional and \(x'_t\)'s are \(d'\) dimensional.

Goal: Compress say view one into \(y_1, \ldots, y_n\), that are \(K\) dimensional vectors.

- Retain information redundant between the two views.
- Eliminate “noise” specific to only one of the views.
Which Direction to Pick?

View I

View II
Which Direction to Pick?

View I

View II
Which Direction to Pick?

View I

View II
Which Direction to Pick?

View I

View II
Which Direction to Pick?

PCA direction
Which Direction to Pick?

Direction has large covariance
Say $\mathbf{w}_1$ and $\mathbf{v}_1$ are the directions we choose to project in views 1 and 2 respectively we want these directions to maximize,

\[
\frac{\frac{1}{n} \sum_{t=1}^{n} (\mathbf{y}_t[1] - \frac{1}{n} \sum_{t=1}^{n} \mathbf{y}_t[1]) \cdot (\mathbf{y}_t'[1] - \frac{1}{n} \sum_{t=1}^{n} \mathbf{y}_t'[1])}{\sqrt{\frac{1}{n} \sum_{t=1}^{n} (\mathbf{y}_t[1] - \frac{1}{n} \sum_{t=1}^{n} \mathbf{y}_t[1])^2 \sqrt{\frac{1}{n} \sum_{t=1}^{n} (\mathbf{y}_t'[1] - \frac{1}{n} \sum_{t=1}^{n} \mathbf{y}_t'[1])}}}.
\]
Say \( w_1 \) and \( v_1 \) are the directions we choose to project in views 1 and 2 respectively we want these directions to maximize,

\[
\frac{1}{n} \sum_{t=1}^{n} \left( y_t[1] - \frac{1}{n} \sum_{t=1}^{n} y_t[1] \right) \cdot \left( y_t'[1] - \frac{1}{n} \sum_{t=1}^{n} y_t'[1] \right)
\]

where \( y_t[1] = w_1^\top x_t \) and \( y_t'[1] = v_1^\top x_t' \)
**Maximizing Correlation Coefficient**

Say $w_1$ and $v_1$ are the directions we choose to project in views 1 and 2 respectively we want these directions to maximize,

$$\frac{1}{n} \sum_{t=1}^{n} \left( y_t[1] - \frac{1}{n} \sum_{t=1}^{n} y_t[1] \right) \cdot \left( y_t'[1] - \frac{1}{n} \sum_{t=1}^{n} y_t'[1] \right)$$

s.t. $\frac{1}{n} \sum_{t=1}^{n} \left( y_t[1] - \frac{1}{n} \sum_{t=1}^{n} y_t[1] \right)^2 = \frac{1}{n} \sum_{t=1}^{n} \left( y_t'[1] - \frac{1}{n} \sum_{t=1}^{n} y_t'[1] \right) = 1$

where $y_t[1] = w_1^\top x_t$ and $y_t'[1] = v_1^\top x_t'$
Hence we want to solve for projection vectors $w_1$ and $v_1$ that

$$\text{maximize} \quad \frac{1}{n} \sum_{t=1}^{n} w_1^\top (x_t - \mu) \cdot v_1^\top (x'_t - \mu')$$

subject to

$$\frac{1}{n} \sum_{t=1}^{n} (w_1^\top (x_t - \mu))^2 = \frac{1}{n} \sum_{t=1}^{n} (v_1^\top (x'_t - \mu'))^2 = 1$$

where $\mu = \frac{1}{n} \sum_{t=1}^{n} x_t$ and $\mu' = \frac{1}{n} \sum_{t=1}^{n} x'_t$
Hence we want to solve for projection vectors $\mathbf{w}_1$ and $\mathbf{v}_1$ that

maximize $\mathbf{w}_1^\top \Sigma_{1,2} \mathbf{v}_1$

subject to $\mathbf{w}_1^\top \Sigma_{1,1} \mathbf{w}_1 = \mathbf{v}_1^\top \Sigma_{2,2} \mathbf{v}_1 = 1$
Hence we want to solve for projection vectors $\mathbf{w}_1$ and $\mathbf{v}_1$ that maximize $\mathbf{w}_1^\top \Sigma_{1,2} \mathbf{v}_1$
subject to $\mathbf{w}_1^\top \Sigma_{1,1} \mathbf{w}_1 = \mathbf{v}_1^\top \Sigma_{2,2} \mathbf{v}_1 = 1$

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} = \text{COV}(\mathbf{X} \mathbf{X}')$$
Write $\bar{x}_t = \bar{x}_t \bar{x}'_t$ the $d + d'$ dimensional concatenated vectors.

Calculate covariance matrix of the joint data points $\mathbf{\Sigma} = \mathbf{\Sigma}_1,1,1 \mathbf{\Sigma}_1,2,2 \mathbf{\Sigma}_2,1,1 \mathbf{\Sigma}_2,2,2$. The top $K$ eigen vectors of this matrix give us projection matrix for view I.

Calculate $\mathbf{\Sigma}_1,2,2 \mathbf{\Sigma}_2,1,1 \mathbf{\Sigma}_1,1,1 \mathbf{\Sigma}_2,2,2$. The top $K$ eigen vectors of this matrix give us projection matrix for view II.
Write \( \tilde{x}_t = x_t x'_t \) the \( d + d' \) dimensional concatenated vectors.

Calculate covariance matrix of the joint data points

\[ \Sigma = \begin{pmatrix} \Sigma_1,1 & \Sigma_1,2 \\ \Sigma_2,1 & \Sigma_2,2 \end{pmatrix} \]

Calculate \( \Sigma_{-1,1} x_1,1 \Sigma_{1,2} \Sigma_{-1,2} x_2,1 \). The top \( K \) eigen vectors of this matrix give us projection matrix for view I.

Calculate \( \Sigma_{-1,2} x_2,2 \Sigma_{2,1} \Sigma_{-1,1} x_1,2 \). The top \( K \) eigen vectors of this matrix give us projection matrix for view II.

\[ X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \]
CCA Algorithm

1. \( X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \)

2. \( \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} = \text{cov} \begin{pmatrix} X \end{pmatrix} \)
1. \[ X = \begin{pmatrix} \mathbf{X}_1 & \mathbf{X}_2 \end{pmatrix} \]

2. \[ \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} = \text{cov}(\mathbf{X}) \]

3. \[ W = \text{eigs}(\Sigma_{11}^{-1} \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21}, K) \]
CCA Algorithm

1. \( X = \begin{pmatrix} n & X_1 & X_2 \\ d_1 & & \end{pmatrix} \)

2. \( \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} = \text{cov}(X) \)

3. \( W = \text{eigs}(\Sigma_{11}^{-1} \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21}, K) \)

4. \( Y_1 = (X_1 - \mu_1) \times W \)
LINEAR PROJECTIONS

\( n \)

\( \mathbf{X} \)

\( d \)
Linear Projections

\[ n \times d \quad \rightarrow \quad n \times K \]
LINEAR PROJECTIONS

\[ n \times d \times W = n \times Y \]
LINEAR PROJECTIONS

$X \times d W = Y$

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

- Lift to higher dimensions to introduce non-linearity
  - Linear in high dim = non-linear in lower dim
- Project to lower dimension using PCA

Examples:
- RBF kernel: $k(x_t, x_s) = \exp(-\frac{(x_t - x_s)^2}{2})$
- Polynomial kernel: $k(x_t, x_s) = (x_t^Ty_t)^p$
Given $\mathbf{x}_t \in \mathbb{R}^d$, the feature space vector is given by mapping

$$
\Phi(\mathbf{x}_t) = (\mathbf{x}_t[1] \cdot \mathbf{x}_t[1], \mathbf{x}_t[1] \cdot \mathbf{x}_t[2], \ldots, \mathbf{x}_t[d] \cdot \mathbf{x}_t[d], \ldots)^\top
$$
Given $x_t \in \mathbb{R}^d$, the feature space vector is given by mapping

$$\Phi(x_t) = (x_t[1], \ldots, x_t[d], x_t[1] \cdot x_t[1], x_t[1] \cdot x_t[2], \ldots, x_t[d] \cdot x_t[d], \ldots)^\top$$

Enumerating products up to order $K$ (ie. products of at most $K$ coordinates) we can get degree $K$ polynomials.
Given $x_t \in \mathbb{R}^d$, the feature space vector is given by mapping

$$
\Phi(x_t) = (x_t[1], \ldots, x_t[d], x_t[1] \cdot x_t[1], x_t[1] \cdot x_t[2], \ldots, x_t[d] \cdot x_t[d], \ldots)^T
$$

Enumerating products up to order $K$ (ie. products of at most $K$ coordinates) we can get degree $K$ polynomials.

However dimension blows up as $d^K$
Given $x_t \in \mathbb{R}^d$, the feature space vector is given by mapping

$$
\Phi(x_t) = (x_t[1], \ldots, x_t[d], x_t[1] \cdot x_t[1], x_t[1] \cdot x_t[2], \ldots, x_t[d] \cdot x_t[d], \ldots)^\top
$$

Enumerating products up to order $K$ (i.e., products of at most $K$ coordinates) we can get degree $K$ polynomials.

However dimension blows up as $d^K$

Is there a way to do this without enumerating $\Phi$?
Essence of Kernel trick:

If we can write down an algorithm only in terms of $(x_t \cdot x_s)$ for data points $x_t$ and $x_s$, then we don't need to explicitly enumerate $(x_t)$'s but instead, compute $k(x_t, x_s) = (x_t \cdot x_s)$ (even if maps to infinite dimensional space).

Example: RBF kernel

$$k(x_t, x_s) = \exp\left(-\frac{(x_t - x_s)^2}{2}\right),$$

polynomial kernel

$$k(x_t, x_s) = (x_t^y)_t$$

Kernel function measures similarity between points.
Essence of Kernel trick:

- If we can write down an algorithm only in terms of $\Phi(x_t)^\top \Phi(x_s)$ for data points $x_t$ and $x_s$

Example: RBF kernel $k(x_t, x_s) = \exp(-\frac{(x_t - x_s)^2}{2})$,
polynomial kernel $k(x_t, x_s) = \sum_i x_t^i y_t^i$. 

Kernel function measures similarity between points.
Essence of Kernel trick:

- **If** we can write down an algorithm only in terms of $\Phi(x_t)\mathbf{^T} \Phi(x_s)$ for data points $x_t$ and $x_s$

- **Then** we don’t need to explicitly enumerate $\Phi(x_t)$’s but instead, compute $k(x_t, x_s) = \Phi(x_t)\mathbf{^T} \Phi(x_s)$ (even if $\Phi$ maps to infinite dimensional space)
Essence of Kernel trick:

- **If** we can write down an algorithm only in terms of $\Phi(x_t)\top\Phi(x_s)$ for data points $x_t$ and $x_s$

- **Then** we don’t need to explicitly enumerate $\Phi(x_t)$’s but instead, compute $k(x_t, x_s) = \Phi(x_t)\top\Phi(x_s)$ (even if $\Phi$ maps to infinite dimensional space)

Example: RBF kernel $k(x_t, x_s) = \exp(-\sigma\|x_t - x_s\|_2^2)$, polynomial kernel $k(x_t, x_s) = (x_t\top y_t)^p$
**Kernel Trick**

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

Essence of Kernel trick:

If we can write down an algorithm only in terms of

\[(x_t^T y_t)^p\]

for data points \(x_t\) and \(x_s\),
then we don't need to explicitly enumerate \(x_t^T\)'s but instead compute

\[k(x_t, x_s) = (x_t^T y_t)^p\]

Even if maps to infinite dimensional space.

Example: RBF kernel

\[k(x_t, x_s) = \exp(-\frac{(x_t - x_s)^2}{2})\]

Polynomial kernel

\[k(x_t, x_s) = (x_t^T y_t)^p\]
Kernel Trick

\[ (x_t^T y_t)^p = \sum_{k_1+k_2+\ldots+k_d=p} \binom{c}{k_1, k_2, \ldots, k_d} \prod_{j=1}^{d} (x_t[j] y_t[j])^{k_j} \]
**Kernel Trick**

Essence of Kernel trick:

If we can write down an algorithm only in terms of $(x_t)^\top (x_s)$

for data points $x_t$ and $x_s$.

Then we don't need to explicitly enumerate $(x_t)^\top$'s but instead,
compute $k(x_t, x_s) = (x_t)^\top (x_s)$ (even if $\text{maps to infinite dimension space}$)

Example: RBF kernel

\[ k(x_t, x_s) = \exp(-\frac{1}{2}\|x_t - x_s\|^2) \]

polynomial kernel

\[ k(x_t, x_s) = (x_t)^\top y_t \]

Kernel function measures similarity between points.
Kernel Trick

\[(x_t^\top y_t)^p = \sum_{k_1+k_2+\ldots+k_d=p} \left( \prod_{j=1}^{d} (x_t[j]y_t[j])^{k_j} \right) \left( \prod_{j=1}^{d} x_t[j]^{k_t} \right) \cdot \left( \prod_{j=1}^{d} y_t[j]^{k_j} \right) \]

\[
\Phi(x)^\top = \left( \ldots, \sqrt{\prod_{j=1}^{d} x_t[j]^{k_t}}, \ldots \right) \quad \text{for } k_1+k_2+\ldots+k_d=p
\]
The $k$th column of $W$ is eigenvector of covariance matrix. That is,

$$W_k = \mathbf{\Lambda} W_k.$$ 

Rewriting, for centered $X$,

$$X_k W_k = \frac{1}{n} \sum_{t=1}^{n} x_t x_t' W_k = \frac{1}{n} x_t' \mathbf{\Lambda} x_t = \frac{1}{n} x_t' x_t W_k$$

where

$$\mathbf{\Lambda}_k[t] = \frac{1}{n} x_t' x_t W_k.$$
Let's start with the assumption that Data is centered! (i.e. Sum of $x_t$'s is 0)
Let's start with the assumption that Data is centered! (i.e. Sum of $x_t$'s is 0)

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

$$\lambda_k W_k = \frac{1}{n} \left( \sum_{t=1}^{n} x_t x_t^\top \right) W_k = \frac{1}{n} \sum_{t=1}^{n} (x_t^\top W_k) x_t$$
Let's start with the assumption that Data is centered! (i.e. Sum of $x_t$'s is 0)

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

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

But $x_t^\top W_k = y_t[k]$
Let's rewrite PCA

Lets start with the assumption that Data is centered! (i.e. Sum of \( x_t \)'s is 0)

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

  \[
  \lambda_k W_k = \frac{1}{n} \left( \sum_{t=1}^{n} x_t x_t^\top \right) W_k = \frac{1}{n} \sum_{t=1}^{n} (x_t^\top W_k) x_t
  \]

  But \( x_t^\top W_k = y_t[k] \)

  \[
  \lambda_k W_k = \frac{1}{n} \sum_{t=1}^{n} y_t[k] x_t
  \]
\[ y_s[k] = W_k^T x_s \]
\[ y_s[k] = W_k^\top x_s \]

\[ = \frac{1}{\lambda_k} \left( \frac{1}{n} \sum_{t=1}^{n} y_t[k] x_t \right)^\top x_s \]
Let's Rewrite PCA

\[ y_s[k] = W_k^\top x_s \]

\[ = \frac{1}{\lambda_k} \left( \frac{1}{n} \sum_{t=1}^{n} y_t[k] x_t \right)^\top x_s \]

\[ = \frac{1}{n\lambda_k} \sum_{t=1}^{n} y_t[k] x_t^\top x_s \]
\[ y_s[k] = W_k^\top x_s \]

\[
\begin{align*}
\frac{1}{\lambda_k} \left( \frac{1}{n} \sum_{t=1}^{n} y_t[k] x_t \right) ^\top x_s \\
\frac{1}{n \lambda_k} \sum_{t=1}^{n} y_t[k] x_t^\top x_s \\
\frac{1}{n \lambda_k} \sum_{t=1}^{n} y_t[k] \tilde{K}_{s,t}
\end{align*}
\]
\[
\mathbf{y}_s[k] = W_k^\top \mathbf{x}_s \\
= \frac{1}{\lambda_k} \left( \frac{1}{n} \sum_{t=1}^{n} \mathbf{y}_t[k] \mathbf{x}_t \right)^\top \mathbf{x}_s \\
= \frac{1}{n \lambda_k} \sum_{t=1}^{n} \mathbf{y}_t[k] \mathbf{x}_t^\top \mathbf{x}_s \\
= \frac{1}{n \lambda_k} \sum_{t=1}^{n} \mathbf{y}_t[k] \tilde{K}_{s,t}
\]

Where \( \tilde{K}_{s,t} = \mathbf{x}_t^\top \mathbf{x}_s \) is the kernel matrix for centered data.
• Hence, the k’th column on Y matrix is such that

\[ y[k] = \frac{1}{n\lambda_k} y[k] \tilde{K} \]
• Hence, the k’th column on Y matrix is such that

\[ y[k] = \frac{1}{n\lambda_k} y[k] \tilde{K} \]

Also we have, \( 1 = \|W_k\|^2 = \frac{1}{\lambda_k^2 n^2} \left( \sum_{t=1}^{n} y_t[k] x_t \right)^\top \left( \sum_{s=1}^{n} y_s[k] x_s \right) \)
Hence, the k’th column on Y matrix is such that

\[ y[k] = \frac{1}{n\lambda_k} y[k] \tilde{K} \]

Also we have, \( 1 = ||W_k||^2 = \frac{1}{\lambda_k^2 n^2} \left( \sum_{t=1}^{n} y_t[k] x_t \right)^\top \left( \sum_{s=1}^{n} y_s[k] x_s \right) \)

\[ = \frac{1}{\lambda_k^2 n^2} \sum_{t=1}^{n} \sum_{s=1}^{n} y_s[k] x_s^\top x_t y_t[k] \]
• Hence, the $k$'th column on $Y$ matrix is such that

$$y[k] = \frac{1}{n\lambda_k} y[k] \tilde{K}$$

Also we have, $1 = ||W_k||^2 = \frac{1}{\lambda_k^2 n^2} \left( \sum_{t=1}^{n} y_t[k] x_t \right)^\top \left( \sum_{s=1}^{n} y_s[k] x_s \right)$

$$= \frac{1}{\lambda_k^2 n^2} \sum_{t=1}^{n} \sum_{s=1}^{n} y_s[k] x_s^\top x_t y_t[k]$$

$$= \frac{1}{\lambda_k^2 n^2} y[k] \tilde{K} y[k]^\top$$
• Hence, the k’th column on Y matrix is such that

\[ \mathbf{y}[k] = \frac{1}{n\lambda_k} \mathbf{y}[k] \tilde{K} \]

Also we have, \( 1 = \|W_k\|^2 = \frac{1}{\lambda_k^2 n^2} \left( \sum_{t=1}^{n} \mathbf{y}_t[k] \mathbf{x}_t \right)^\top \left( \sum_{s=1}^{n} \mathbf{y}_s[k] \mathbf{x}_s \right) \)

\[ = \frac{1}{\lambda_k^2 n^2} \sum_{t=1}^{n} \sum_{s=1}^{n} \mathbf{y}_s[k] \mathbf{x}_s^\top \mathbf{x}_t \mathbf{y}_t[k] \]

\[ = \frac{1}{\lambda_k^2 n^2} \mathbf{y}[k] \tilde{K} \mathbf{y}[k]^\top = \frac{1}{n\lambda_k} \|\mathbf{y}[k]\|^2 \]
• Hence, the k'th column on Y matrix is such that

\[
y[k] = \frac{1}{n\lambda_k} y[k] \tilde{K}
\]

Also we have, 

\[
1 = \|W_k\|^2 = \frac{1}{\lambda_k^2 n^2} \left( \sum_{t=1}^{n} y_t[k] x_t \right) \top \left( \sum_{s=1}^{n} y_s[k] x_s \right)
\]

\[
= \frac{1}{\lambda_k^2 n^2} \sum_{t=1}^{n} \sum_{s=1}^{n} y_s[k] x_s \top x_t y_t[k]
\]

\[
= \frac{1}{\lambda_k^2 n^2} y[k] \tilde{K} y[k] \top = \frac{1}{n\lambda_k} \|y[k]\|^2
\]

Hence \( P_k = y[k]/\sqrt{n\lambda_k} \) is an eigenvector of \( \tilde{K} \) with eigen value \( \gamma_k = n\lambda_k \)
We assumed centered data, what if its not,

\[
\tilde{K}_{s,t} = \left( x_t - \frac{1}{n} \sum_{u=1}^{n} x_u \right)^\top \left( x_s - \frac{1}{n} \sum_{u=1}^{n} x_u \right)
\]

\[
= x_t^\top x_s - \left( \frac{1}{n} \sum_{u=1}^{n} x_u \right)^\top x_s - \left( \frac{1}{n} \sum_{u=1}^{n} x_u \right)^\top x_t
\]

\[
+ \frac{1}{n^2} \left( \sum_{u=1}^{n} x_u \right)^\top \left( \sum_{v=1}^{n} x_v \right)
\]

\[
= x_t^\top x_s - \frac{1}{n} \sum_{u=1}^{n} x_u^\top x_s - \frac{1}{n} \sum_{u=1}^{n} x_u^\top x_t + \frac{1}{n^2} \sum_{u=1}^{n} \sum_{v=1}^{n} x_u^\top x_v
\]
Equivalently, if $\mathbf{Kern}$ is the matrix ($\mathbf{Kern}_{t,s} = x_t^T x_s$),

$$\hat{\mathbf{K}} = \mathbf{Kern} - \frac{(\mathbf{1}_{n \times n} \times \mathbf{Kern})}{n} - \frac{\mathbf{Kern} \times \mathbf{1}_{n \times n}}{n} + \frac{(\mathbf{1}_{n \times n} \times \mathbf{Kern} \times \mathbf{1}_{n \times n})}{n^2}$$
All we need to be able to compute, to perform PCA are $x^t x$.

Replace $x^t x$ with $(x^t)(x^s) = k(x^t, x^s)$ to perform PCA in feature space.
Kernel PCA

All we need to be able to compute, to perform PCA are $x_t^\top x_s$
All we need to be able to compute, to perform PCA are $x_t^\top x_s$

Replace $x_t^\top x_s$ with $\Phi(x_t)^\top \Phi(x_s) = k(x_t, x_s)$ to perform PCA in feature space
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

\[
\tilde{K}_s, t = \frac{1}{n} \sum_{u=1}^{n} (x_t - \frac{1}{n} \sum_{v=1}^{n} x_u) (x_s - \frac{1}{n} \sum_{v=1}^{n} x_v) + \frac{1}{n^2} \sum_{u=1}^{n} \sum_{v=1}^{n} (x_u - \frac{1}{n} \sum_{v=1}^{n} x_v) (x_v - \frac{1}{n} \sum_{v=1}^{n} x_v) = k(x_t, x_s)
\]

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

$$\tilde{K}_{s,t} = \frac{1}{n} \sum_{u=1}^{n} (x_u - \frac{1}{n} \sum_{v=1}^{n} x_v) (x_s - \frac{1}{n} \sum_{v=1}^{n} x_v) \frac{1}{n} \sum_{v=1}^{n} (x_u - \frac{1}{n} \sum_{v=1}^{n} x_v) \frac{1}{n} \sum_{v=1}^{n} (x_s - \frac{1}{n} \sum_{v=1}^{n} x_v)$$

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

$\text{Kern} = \begin{bmatrix}
    k(x_1, x_1) & k(x_1, x_2) & \ldots & k(x_1, x_n) \\
    k(x_2, x_1) & k(x_2, x_2) & \ldots & k(x_2, x_n) \\
    \vdots & \vdots & \ddots & \vdots \\
    k(x_{n-1}, x_1) & k(x_{n-1}, x_2) & \ldots & k(x_{n-1}, x_n) \\
    k(x_n, x_1) & k(x_n, x_2) & \ldots & k(x_n, x_n)
\end{bmatrix}$
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

$$\tilde{K}_{s,t} = \frac{1}{n} \sum_{u=1}^{n} (x_t - \bar{x}_t)(x_s - \bar{x}_s) = \mathbf{k}(x_t, x_s)$$

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

$$\tilde{K} = \text{Kern} - \frac{1}{n} \left( \mathbf{1} \text{Kern} + \text{Kern} \mathbf{1} \right) + \frac{1}{n^2} \mathbf{1} \text{Kern} \mathbf{1}$$
Kernel PCA

\[ P_1 \sqrt{\lambda_1} \quad \cdots \quad P_K \sqrt{\lambda_K} \]
Kernel PCA

3. \[ \begin{bmatrix} P & , \gamma \end{bmatrix}^n_K = \text{eigs}(\begin{bmatrix} \tilde{K} \end{bmatrix}, K) \]
Kernel PCA

3. \[ \begin{bmatrix} P \\ \gamma \end{bmatrix} = \text{eigs} \left( \begin{pmatrix} \tilde{K} \\ K \end{pmatrix} \right) \]

4. \[ n \begin{bmatrix} Y \\ \gamma \end{bmatrix} = \begin{bmatrix} P_k \gamma_k \\ P_k \gamma_k \end{bmatrix} \]