ADVANCED MACHINE LEARNING

Kernel PCA
Overview Today’s Lecture

• Brief Recap of Classical Principal Component Analysis (PCA)

• Derivation of kernel PCA

• Brief introduction to Canonical Correlation Analysis (CCA)

• Extension to kernel CCA
Principal Component Analysis: Overview

Take samples of two classes (yellow and pink classes)

Each image is a high-dimensional vector

$$x \in \mathbb{R}^{320 \times 240 \times 3 = 230400}$$
Principal Component Analysis: Overview

Project the images onto a lower dimensional space

\[ y \in \mathbb{R}^2 \text{ through matrix } \mathbf{A} \in \mathbb{R}^{2 \times 230400} : \ y = \mathbf{A}x \]
Principal Component Analysis: Overview

Project the images onto a lower dimensional space

\[ y \in \mathbb{R}^2 \text{ through matrix } A \in \mathbb{R}^{2 \times 230400} : \quad y = Ax \]

What is A?
PCA discovers the matrix A
Principal Component Analysis: Overview

Infinite number of choices for projection matrix $A$  
$\rightarrow$ need criteria to reduce the choice

1: minimum information loss (minimal reconstruction error)

What is the 2D to 1D projection that minimizes the reconstruction error?
Principal Component Analysis: Overview

Infinite number of choices for projection matrix $A$ → need criteria to reduce the choice

1: minimum information loss (minimal reconstruction error)
2: equivalent to finding the direction with maximum variance

What is the 2D to 1D projection that minimizes the reconstruction error?
Principal Component Analysis: Overview

Dataset $X = \begin{bmatrix} x^1 & x^2 & \ldots & x^M \end{bmatrix}$ (data is centered - $E\{X\} = 0$)

Compute covariance matrix of dataset $X$ : $C = E\{XX^T\}$

Find eigenvalue decomposition: $C = V\Lambda V^T$

$V = \begin{bmatrix} e^1 & \ldots & e^N \end{bmatrix}$: matrix of eigenvectors

$\Lambda$ : Diagonal matrix of eigenvalues

Order $e^1, \ldots, e^N$, s.t. $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_N$

The eigenvectors form a basis of the space.

$e^1$ is aligned with the axis of maximum variance.

Project data onto eigenvectors

Remove projections with low $\lambda$ (noise)
PCA for Data Compression

Original image is encoded in \( x \in \mathbb{R}^N \).

Compressed image is \( y \in \mathbb{R}^{p=0.1N} \)
\[
y = A_p x,
\]

Rows of \( A_p \) contains 1st \( p \) eigenvectors

Original Image

Image compressed 90%
PCA for Feature Extraction

Results of decomposition with Principal Component Analysis: eigenvectors

Encapsulate main differences across groups of images (in the first eigenvectors)

Detailed features (glasses) get encapsulated next (in the following eigenvectors)
Principal Component Analysis: Pros & Cons

**Advantages:**

a) Projection ensures minimal reconstruction error

b) Projects does not distort the space (rotation in space)

→ Results often in the features in the projection being interpretable visually

**Limitations:**

a) PCA assumes a linear transformation

→ With centering of data, can do only a rotation in space

b) Fails at finding directions that require a non-linear transformation
Revisiting the hypotheses of PCA

PCA assumed a *linear* transformation

→ **Non-linear** PCA *(Kernel PCA)*: find a non-linear embedding of the data and then perform linear PCA.
Recall: Principle of kernel Methods
Going back to linearity

Find a non-linear transformation that send the data in a space where linear computation is again feasible.
Kernel PCA: Principle

Determine a transformation which brings out features of the data so as to make subsequent computation easier.

Example above: Data becomes *linearly* separable when using a rbf kernel and projecting onto first 2 PC-s of kernel PCA.
Kernel PCA: Principle

Idea: Send the data $X$ into a feature space $H$ through the nonlinear map $\phi$.

$$X = \left\{ x^i \in \mathbb{R}^N \right\}_{i=1}^{M} \mapsto \phi(X) = \left( \phi(x^1), \ldots, \phi(x^M) \right)$$

Perform linear pca in feature space and project into set of eigenvectors in feature space

Scholkopf et al, Neural Computation, 1998
Kernel PCA: Principle

Idea: Send the data $X$ into a *feature space* $H$ through the nonlinear map $\phi$.

In feature space, perform classical *linear* computation

\[ X = \{ x^i \in \mathbb{R}^N \}_{i=1}^{M} \mapsto \phi(X) = (\phi(x^1), \ldots, \phi(x^M)) \]

Performs linear PCA in feature space

Original Space

$X$ in original space

In feature space

$H$
Kernel-Induced Feature Space

Idea: Send the data $X$ into a feature space $H$ through the nonlinear map $\phi$.

$X = \{ x^i \in \mathbb{R}^N \}^{i=1...M} \mapsto \phi(X) = \left( \phi(x^1), \ldots, \phi(x^M) \right)$

While the dimension of the original space is $N$, the dimension of the feature space may be greater than $N$! $\Rightarrow X$ is lifted onto $H$

Determining $\phi$ is difficult $\Rightarrow$ Kernel Trick
The Kernel Trick

In most cases, determining the transformation $\phi$ may be difficult.

Linear PCA computes an inner product across pairs of observations:

$\left\langle x^i, x^j \right\rangle$

No need to compute the transformation $\phi$, if one expresses everything as a function of the inner product in feature space

$\rightarrow$ the kernel function:

$k : X \times X \rightarrow \mathbb{R}$

$k(\mathbf{x}^i, \mathbf{x}^j) \rightarrow \left\langle \phi(\mathbf{x}^i), \phi(\mathbf{x}^j) \right\rangle$. Metric of similarity across datapoints

May extract some features
Popular Kernels

• Gaussian / RBF Kernel (translation-invariant):

\[ k(x, x') = e^{-\frac{||x-x'||}{2\sigma^2}}, \quad \sigma \in \mathbb{R}. \]

• Homogeneous Polynomial Kernels:

\[ k(x, x') = \langle x, x' \rangle^p, \quad p \in \mathbb{N}; \]

• Inhomogeneous Polynomial Kernels:

\[ k(x, x') = (\langle x, x' \rangle + c)^p, \quad p \in \mathbb{N}, \quad c \geq 0 \]
From Linear PCA to Kernel PCA

Rewriting PCA in terms of dot products:
Each eigenvector $e^1, ..., e^N$ found by linear PCA can be expressed as a linear combination of the datapoints:

Using

$$C e^i = \frac{1}{M} \sum_{j=1}^{M} x^j (x^j)^T e^i$$

with

$$C e^i = \lambda_i e^i$$

we obtain,

$$e^i = \frac{1}{\lambda_i M} \sum_{j=1}^{M} x^j (x^j)^T e^i$$
From Linear PCA to Kernel PCA

Rewriting PCA in terms of dot products:
Each eigenvector $e^1, \ldots, e^N$ found by linear PCA can be expressed as a linear combination of the datapoints:

Using $Ce^i = \frac{1}{M} \sum_{j=1}^{M} x^j (x^j)^T e^i$ with $Ce^i = \lambda_i e^i$

we obtain, $e^i = \frac{1}{\lambda_i M} \sum_{j=1}^{M} x^j \underbrace{(x^j)^T e^i}_{\alpha_j} = \frac{1}{\lambda_i M} \sum_{j=1}^{M} \alpha^i_j x^j$. 

Scalar
Linear PCA in Feature Space

Sending the data in feature space through $\phi$:

$$\phi : X \rightarrow H \quad x \mapsto \phi(x)$$

Assume that, in feature space $H$, the data are centered:

$$\sum_{i=1}^{M} \phi(x^i) = 0$$

The covariance matrix in the feature space is:

$$C_\phi = \frac{1}{M} FF^T$$

The columns $i = 1 \ldots M$ of $F$ are composed of $\phi(x^i)$. 
Linear PCA in Feature Space

As in the original space, in feature space, the covariance matrix can be diagonalized and we have now to find the eigenvalues \( \lambda_i \geq 0 \), satisfying:

\[
C_\phi v^i = \lambda_i v^i
\]

\[
\Rightarrow \left< \phi(x^j), C_\phi v^i \right> = \lambda_i \left< \phi(x^j), v^i \right>, \quad \forall i, j = 1,...,M
\]

All solutions \( v \) with \( \lambda \) different of zero lie in the span of the \( \phi(x^1),...,\phi(x^M) \), and we can thus write:

\[
\lambda_i v^i = \lambda_i \sum_{j=1}^{M} \alpha^i \phi(x^j), \quad \alpha^i = [\alpha^i_1 \ldots \alpha^i_M]
\]
Linear PCA in Feature Space

\[ C_\phi v^i = \lambda_i \sum_{j=1}^{M} \alpha^i \phi(x^j) \quad \langle \phi(x^j), C_\phi v^i \rangle = \lambda_i \langle \phi(x^j), v^i \rangle \]

\[ \Rightarrow \frac{1}{M} \sum_{i=1}^{M} \alpha^i \left( \sum_{j=1}^{M} \alpha^j \phi(x^i) \right) \langle \phi(x^i), \phi(x^j) \rangle = \lambda \sum_{j=1}^{M} \alpha^j \langle \phi(x^i), \phi(x^j) \rangle \]

Given that: \[ K_{ij} = \langle \phi(x^i), \phi(x^j) \rangle \] Kernel Trick

\[ \Rightarrow \text{eigenvalue problem of the form:} \]

\[ \alpha^i K = M \lambda_i \alpha^i, \quad M: \text{number of datapoints} \]

Dual eigenvalue problem of finding the eigenvectors \( v \) of \( C_\phi \).
The solutions to the dual eigenvalue problem are given by all the eigenvectors $\alpha^1, ..., \alpha^M$ with non-zero eigenvalues $\lambda_1, ..., \lambda_M$.

Asking that the eigenvectors $v$ of $C_\phi$ be normalized, i.e. $\langle v^i, v^i \rangle = 1 \quad \forall i = 1, ..., M$

is equivalent to asking that the dual eigenvectors $\alpha^1, ..., \alpha^M$ are such that: $1 / \lambda^i = \|\alpha^i\|$.

Kernel PCA finds at most $M$ eigenvectors $M$: number of datapoints $M >> N$ dimension of each datapoint
Constructing the kPCA projections

We cannot see the projection in feature space!
We can only compute the projections of each point onto each eigenvector.

Projection of query point $x$ onto eigenvector $v^i$:

$$\langle v^i, \phi(x) \rangle = \sum_{j=1}^{M} \alpha_j^i \langle \phi(x^j), \phi(x) \rangle = \sum_{j=1}^{M} \alpha_j^i k(x^j, x)$$

Contour lines group points with equal projection:

All points $x$, s.t. $\langle v^i, \phi(x) \rangle = cst$. 
Contour linear in linear PCA are straight lines. In kPCA, these appear curvy in original space, while straight in feature space.
Popular Kernels

• Gaussian / RBF Kernel (translation-invariant):
  \[ k(x, x') = e^{-\frac{|x-x'|^2}{2\sigma^2}}, \quad \sigma \in \mathbb{R}. \]

• Homogeneous Polynomial Kernels:
  \[ k(x, x') = \langle x, x' \rangle^p, \quad p \in \mathbb{N}; \]

• Inhomogeneous Polynomial Kernels:
  \[ k(x, x') = (\langle x, x' \rangle + c)^p, \quad p \in \mathbb{N}, \quad c \geq 0 \]
Figure 4: Two-dimensional toy example with three data clusters (gaussians with standard deviation 0.1, depicted region: $[-1, 1] \times [-0.5, 1]$): first eight nonlinear principal components extracted with $k(x, y) = \exp(-\frac{|x-y|^2}{0.1})$. Note that the first two principal components (top left) nicely separate the three clusters. Components 3–5 split up the clusters into halves. Similarly, components 6–8 split them again, in a way orthogonal to the above splits. Thus, the first eight components divide the data into 12 regions.
Kernel PCA: Examples

From Schölkopf & Smola, 2002

Figure 5: Two-dimensional toy example with three data clusters (gaussians with standard deviation 0.1, depicted region: $[-1, 1] \times [-0.5, 1]$): first three nonlinear principal components extracted with $k(x, y) = \tanh \left( 2(x \cdot y) + 1 \right)$. The first two principal components (top left) are sufficient to separate the three clusters, and the third component splits the clusters into halves.
Kernel PCA: Examples

MLDEMONS Two sets of “circle” datapoints

Original Data
Kernel PCA: Examples

MLDEMOS Gaussian Kernel

Projections onto first two eigenvectors
Kernel PCA: Examples

MLDEMOS “Pair of Glasses” datapoints

Original Data
Kernel PCA: Examples

MLDEMONS  Gaussian Kernel, kernel width=0.9

Projections onto first two eigenvectors
Kernel PCA: Examples

MLDEMOS Two sets of “circle” datapoints

Original Data
Kernel PCA: Examples

MLDEMOS  Polynomial Kernel order $p=20$

Projections onto first two eigenvectors
Kernel PCA: Examples

MLDEMOS  Polynomial Kernel order p=20

Points clusters here

Projections onto first two eigenvectors
Curse of Dimensionality

Kernel PCA is very intensive computationally.

Computation of the eigenvectors requires eigenvalue decomposition of the Gram matrix (Kernel Matrix is $M \times M$) which grows quadratically with the number of data points $M$.

Computation of each projection in original space grows linearly with $M$ too.

→ A variety of sparse methods have been proposed in the literature
Sparse $k$PCA with $L_p$ penalty

Kernel PCA can be rephrased as a constrained maximization problem:

$F$ is the set of vectors $v$, satisfying:

$$ F = \left\{ v \mid v = \sum_{i=1}^{M} \alpha_i \phi(x^i) \text{ with } \|v\|^2 = \sum_{i,j=1}^{M} \alpha_i \alpha_j k(x^i, x^j) \leq 1 \right\} $$

The first eigenvector can be found by optimizing for:

$$ v^1 = \arg \max_{v \in F_p} \left( \frac{1}{M} \sum_{i=1}^{M} \left\langle v, \phi(x^i) - \sum_{j=1}^{M} \phi(x^j) \right\rangle^2 \right) $$

This is difficult to optimize. One must add further constraints, which can also help reduce the number of solutions.
Sparse kPCA with $L_p$ penalty

The set $F_p$ of possible vectors $v$, contained in a $L_p$ ball:

$$F_p = \left\{ v \mid v = \sum_{i=1}^{M} \alpha_i \phi(x^i) \text{ with } \sum_{i=1}^{M} |\alpha_i|_p \leq 1 \text{ and } \|v\|_p = 1 \right\}$$

The principal eigenvector is then found by optimizing for:

$$v_p = \arg \max_{v \in F_p} \left( \frac{1}{M} \sum_{i=1}^{M} \left( v, \phi(x^i) - \sum_{j=1}^{M} \phi(x^j) \right) \right).$$

Usually $p=1$ or $p=2$ and the optimal solution is found by finding the non-zero $\alpha_i$ that correspond to patterns $\phi(x^i)$ that lie on the vertex of the $L_p$ ball.
Summary

- Kernel PCA offers an alternative to standard PCA to determine projections of the data while not assuming a linear transformation.

- It exploits the principle of the kernel trick, by replacing the inner product between pair of datapoints in the standard PCA (to compute the Covariance matrix) by the kernel function.

- The computation of kernel PCA is simple, yet it is expensive as it requires an eigenvalue decomposition of a very large matrix.

- Current research develops sparse versions of the algorithm.