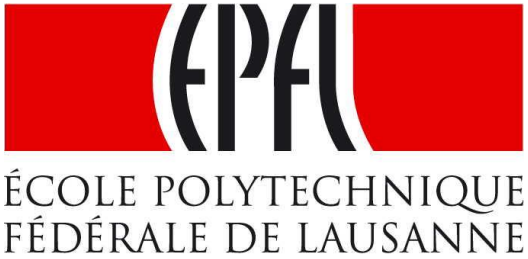
ADVANCED MACHINE LEARNING

Kernel PCA



Overview Today's Lecture

- Brief Recap of Classical Principal Component Analysis (PCA)
- Derivation of kernel PCA
- Exercises to develop a geometrical intuition of kernel PCA

Principal Component Analysis: Overview

Take samples of two classes (yellow and pink classes)

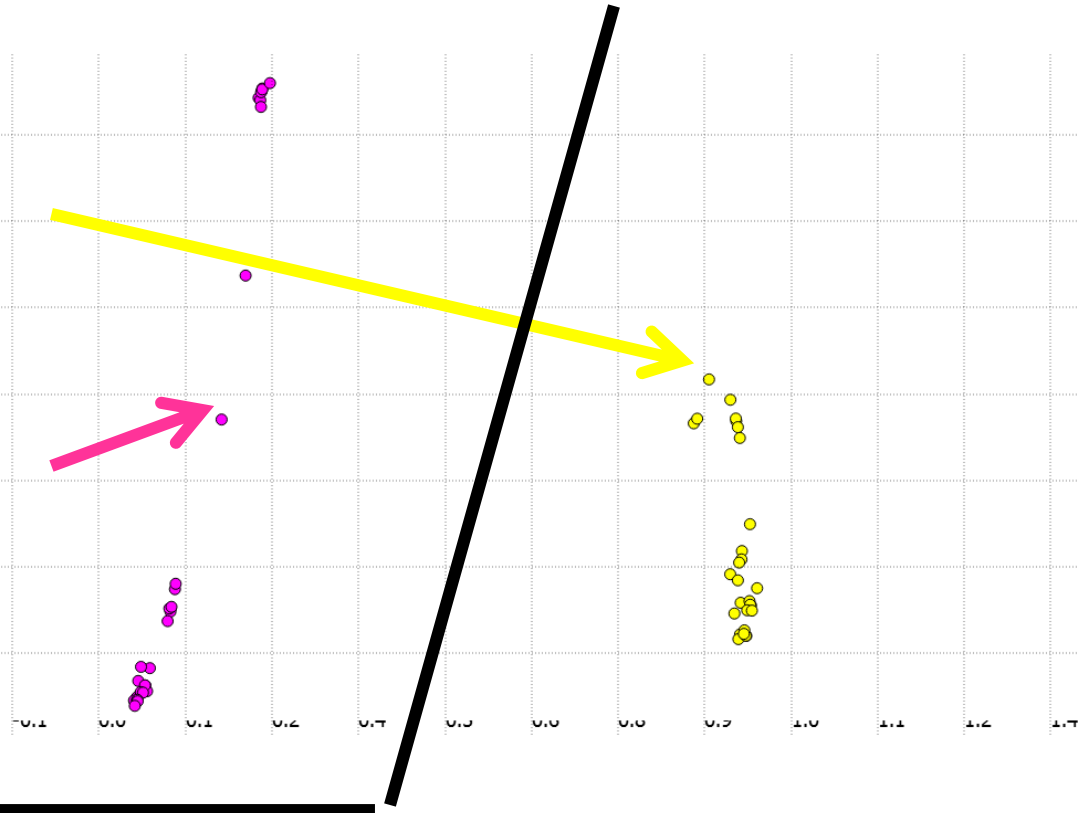
Each image is a high-dimensional vector

$x \in \mathbb{R}^{320 \cdot 240 \cdot 3 = 230400}$

Principal Component Analysis: Overview

Project the images onto a lower dimensional space

$y \in \mathbb{R}^2$ through matrix $A \in \mathbb{R}^{2\times 230400}$: $y = Ax$



Principal Component Analysis: Overview

Project the images onto a lower dimensional space

$y \in \mathbb{R}^2$ through matrix $A \in \mathbb{R}^{2\times 230400}$: $y = Ax$

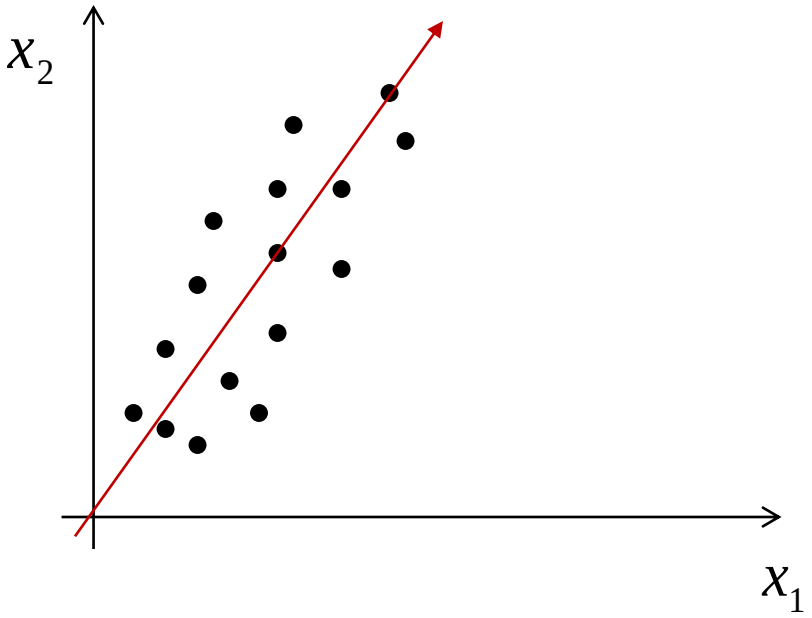
What is A?
PCA discovers the matrix A

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)



What is the 2D to 1D projection that minimizes the reconstruction error?

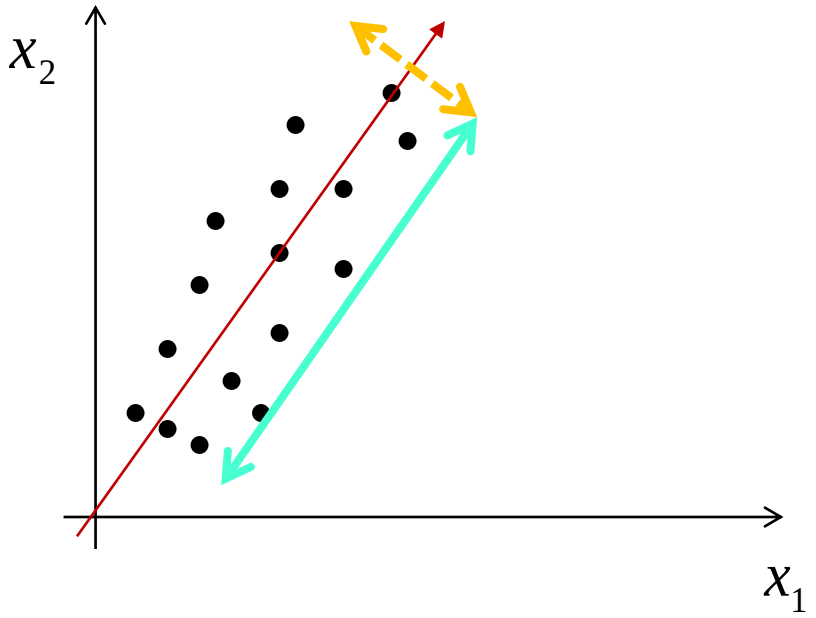
Principal Component Analysis: Overview

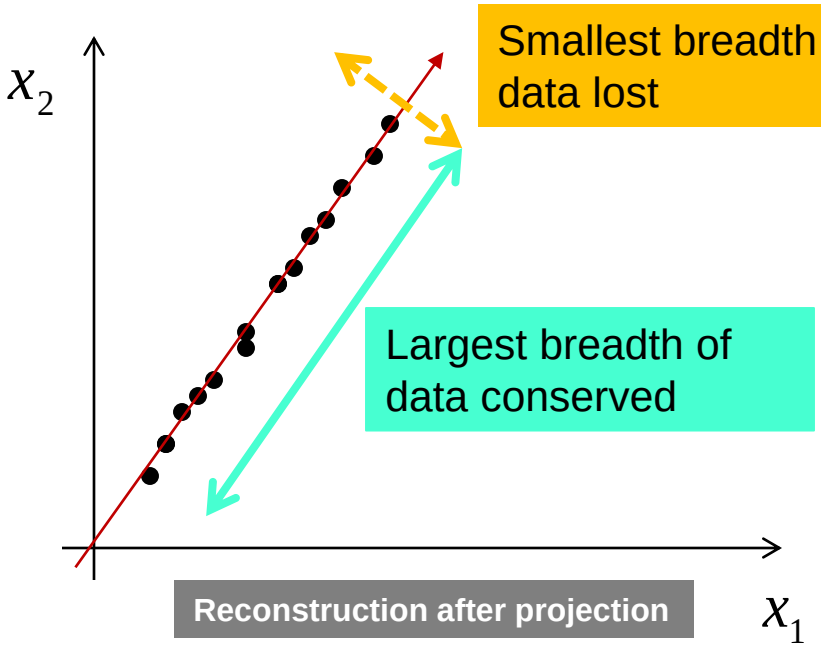
Infinite number of choices for projection matrix A

→ need criteria to reduce the choice

1: minimum information loss(minimal reconstruction error)

2: equivalent to finding the direction with maximum variance



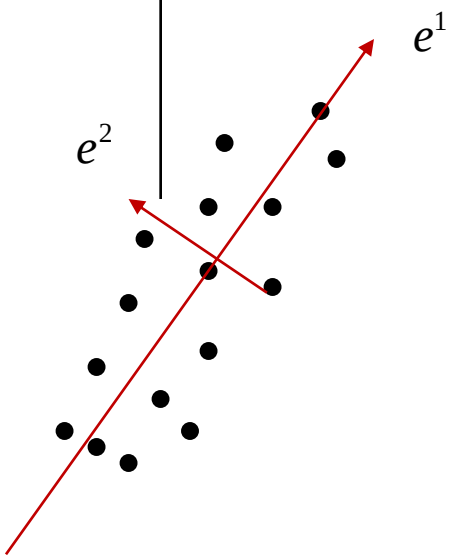


What is the 2D to 1D projection that minimizes the reconstruction error?

Principal Component Analysis: Overview

Dataset $X = \left[x^1 \; x^2 \; .....x^M \right]$ (data is centered, i.e. $E\{X\} = 0$)

Compute covariance matrix of dataset X : $C = \frac{1}{M} E\left\{ XX^T \right\}$



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

$V = \left[e^1 .... e^N \right]$: matrix of eigenvectors

Λ : Diagonal matrix of eigenvalues

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

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 λ (noise).

PCA for Data Compression

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

Compressed image is $y \in \mathbb{R}^{p=0.1N}$

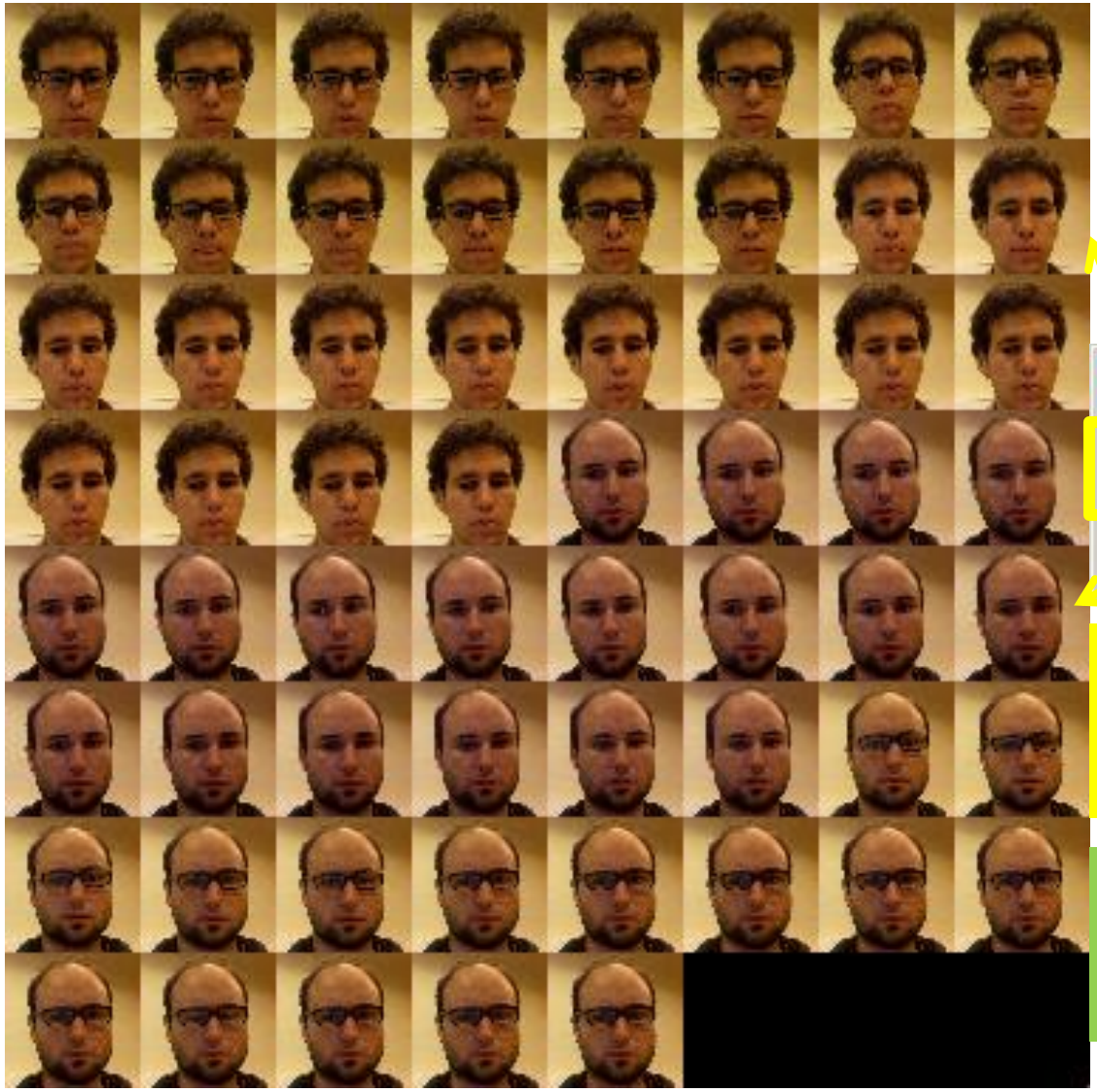
$y = A_p x,$

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) The projection through PCA ensures minimal reconstruction error.

b) The projection does not distort the space (rotation in space).

→ Ease of visualization/interpretation: The features that appear in the projections are often interpretable visually.

Limitations:

a) PCA assumes a linear transformation:

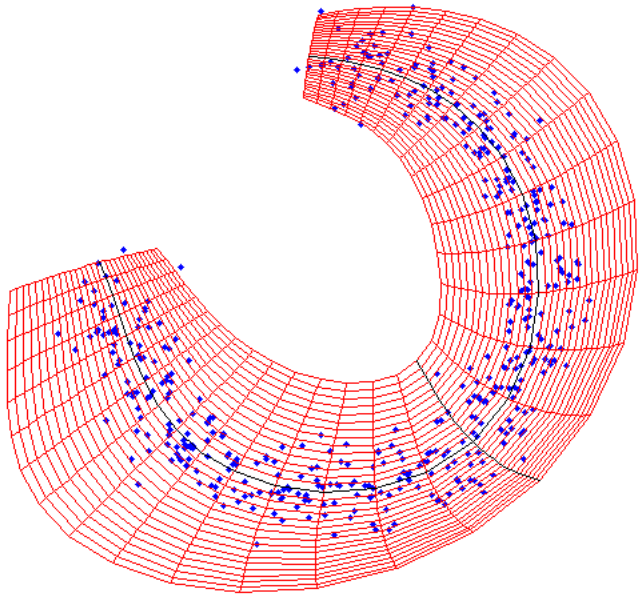
→ With centering of data, one can only do a rotation in space.

b) It fails at finding directions that require a non-linear transformation.

Revisiting the hypotheses of PCA

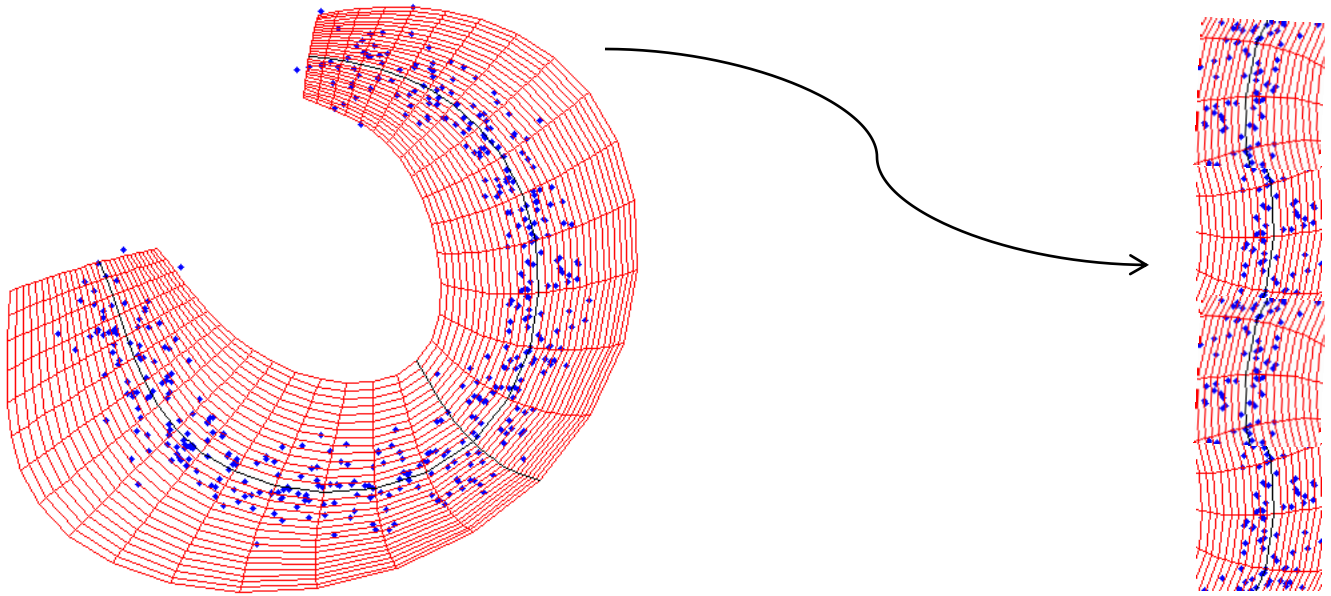
PCA assumed a ***linear*** transformation

→ ***Non-linear*** PCA (Kernel PCA): find a non-linear embedding of the data and then perform linear PCA.



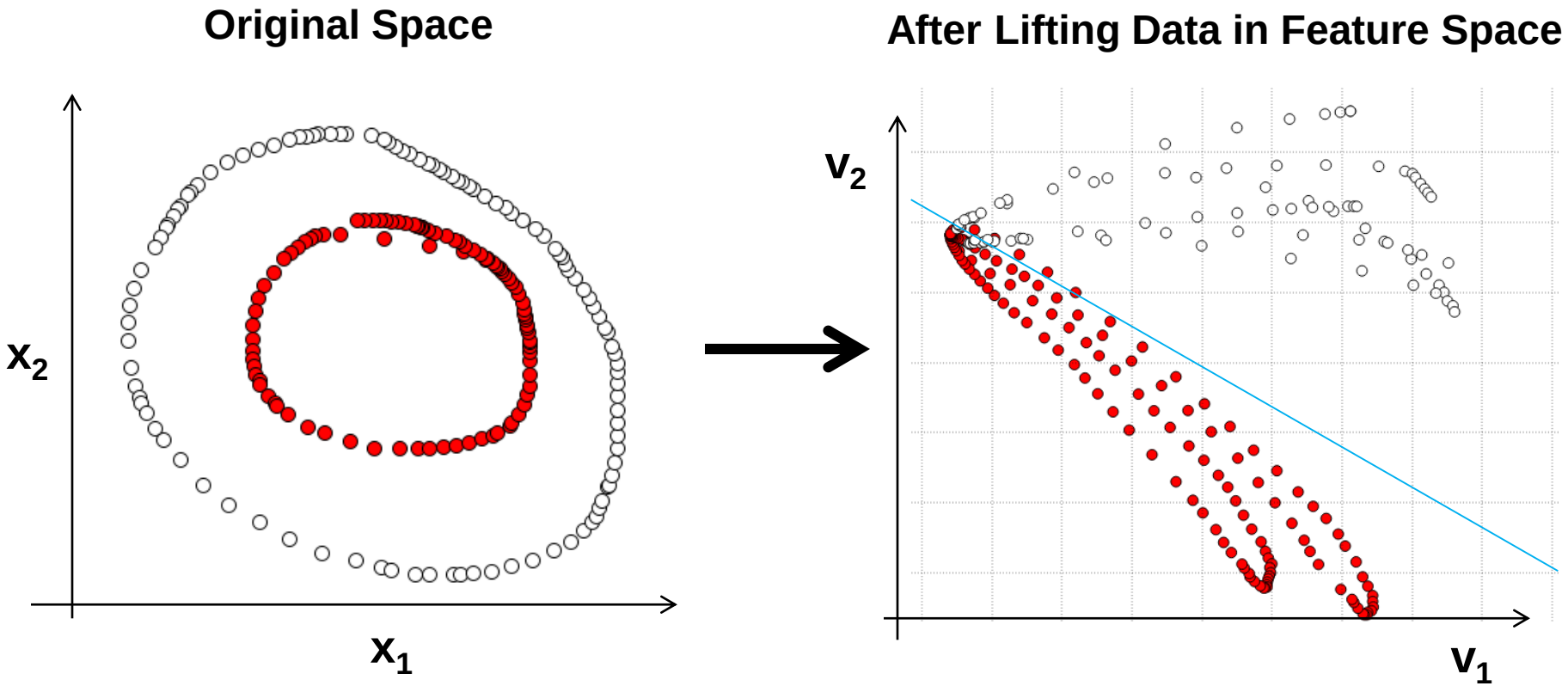
Recall: Principle of kernel Methods
Going back to linearity

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.



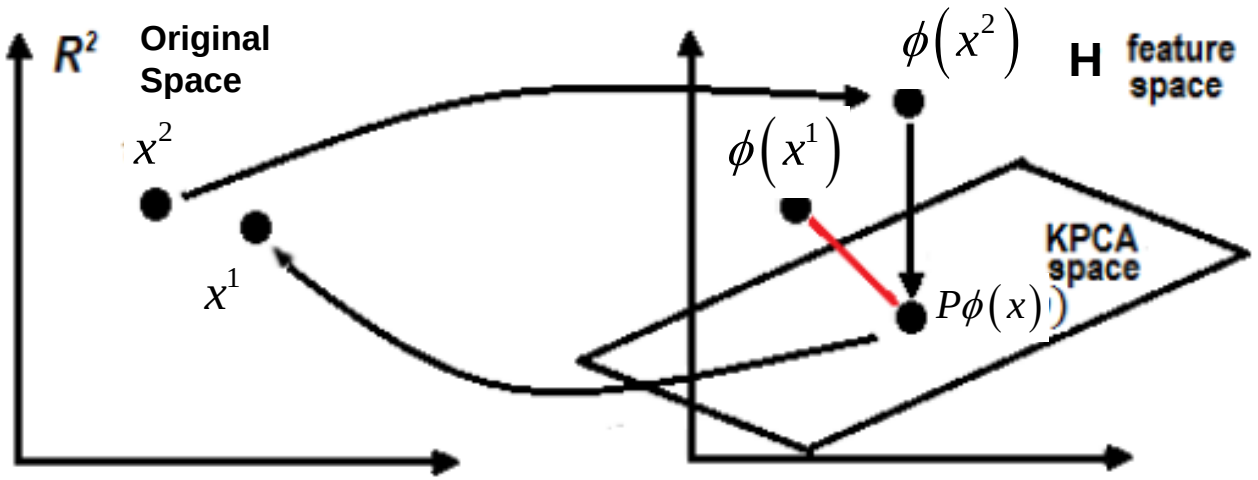
Example above: Data becomes *linearly* separable when using a rbf kernel and projecting onto first 2 PC-s of kernel PCA.

Kernel PCA: Principle

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

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

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



Scholkopf et al, Neural Computation, 1998

Kernel PCA: Principle

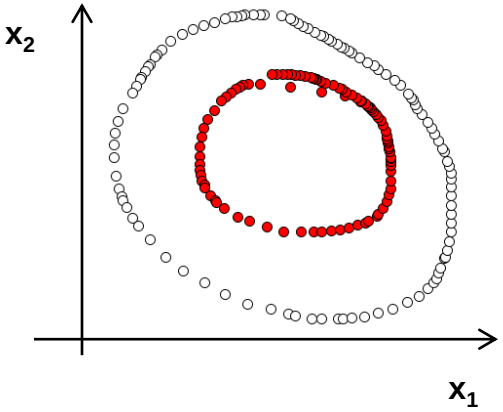
Idea: Send the data *X* into a *feature* space *H* through the nonlinear map ϕ.

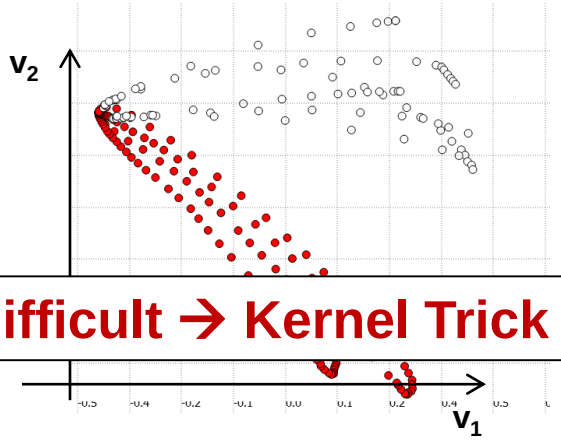
$$X = \left\{ x^i \in \mathbb{R}^N \right\}^{i=1...M} \mapsto \phi(X) = \left(\phi\left(x^1\right), ..., \phi\left(x^M\right) \right)$$

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

Data projected onto the two first principal components in feature space

Original Space





Determining ϕ is difficult → Kernel Trick

Linear PCA in Feature Space

Sending the data in feature space through ϕ:

$$\phi : X \to H \qquad x \mapsto \phi(x)$$

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

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

The covariance matrix in the feature space is:

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

The columns $i = 1...M$ of F are composed of $\phi\left(x^i\right)$.

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 $\lambda_i >= 0$, satisfying:

$$C_\phi v^i = \lambda_i v^i$$

Primal eigenvalue problem: Finding the eigenvectors v of C_ϕ

Not possible in feature space!

=> Formulate everything as a dot product and use kernel trick!

From Linear PCA to Kernel PCA

Each eigenvector $v^1,...,v^M$ can be expressed as a linear combination of the images of the datapoints:

Rewriting PCA in terms of dot products:

Using $C_\phi v^i = \frac{1}{M}\sum_{j=1}^{M}\phi\left(x^j\right)\phi\left(x^j\right)^T v^i$ with $C_\phi v^i = \lambda_i v^i$

we obtain, $v^i = \frac{1}{\lambda_i M}\sum_{j=1}^{M}\phi\left(x^j\right)\underbrace{\phi\left(x^j\right)^T v^i}_{\alpha_j^i} = \frac{1}{\lambda_i M}\sum_{j=1}^{M}\alpha_j^i\phi\left(x^j\right).$

Scalar

Linear PCA in Feature Space

Multiplying the equation:

$$C_\phi v^i = \lambda_i v^i$$

by $\phi\left(x^j\right)^T$, on both sides, we have:

$$\left\langle \phi\left(x^j\right), C_\phi v^i \right\rangle = \lambda_i \left\langle \phi\left(x^j\right), v^i \right\rangle, \ \forall i, j = 1, \ldots M$$

$$C_\phi v^i = \frac{1}{M} \sum_{k=1}^{M} \phi\left(x^k\right) \phi\left(x^k\right)^T v^i \qquad v^i = \frac{1}{\lambda_i M} \sum_{l=1}^{M} \alpha_l^i \phi\left(x^l\right)$$

Linear PCA in Feature Space

$$\Rightarrow \frac{1}{\lambda_i M^2} \sum_{k=1}^{M} \Big\langle \phi(x^j), \phi(x^k) \Big\rangle \sum_{l=1}^{M} \alpha_l^i \Big\langle \phi(x^k), \phi(x^l) \Big\rangle = \frac{1}{M} \sum_{l=1}^{M} \alpha_l^i \Big\langle \phi(x^j), \phi(x^l) \Big\rangle$$

$\langle \phi(x^j), \phi(x^k) \rangle$: $\sum_k K_{jk}$ — $\langle \phi(x^k), \phi(x^l) \rangle$: K_{kl} — $\langle \phi(x^j), \phi(x^l) \rangle$: K_{jl}

Use the kernel trick: $k(x^i, x^j) := K_{ij} = \langle \phi(x^i), \phi(x^j) \rangle$

→ Eigenvalue problem of the form:

$$K\alpha^i = M\lambda_i \alpha^i, \qquad K: \text{ Gram Matrix}$$

Dual eigenvalue problem: Finding the dual eigenvectors α^i.

Linear PCA in Feature Space

The solutions to the dual eigenvalue problem:

are given by all the eigenvectors $\alpha^1,...,\alpha^M$ with non-zero eigenvalues $\lambda_1,...,\lambda_M$.

Kernel PCA finds at most M eigenvectors
M: number of datapoints
M>>N dimension of each datapoint

→ Eigenvalue problem of the form:

$$K\alpha^i = M\lambda_i\alpha^i, \qquad K: \text{ Gram Matrix}$$

Linear PCA in Feature Space

Request that the eigenvectors v of C_ϕ be normalized,

i.e. $\left\langle v^i, v^i \right\rangle = 1 \quad \forall i = 1,...,M$

is equivalent to asking that the dual eigenvectors $\alpha^1,...,\alpha^M$

are such that: $1/\lambda^i = \left\| \alpha^i \right\|$.

Constructing the kPCA projections

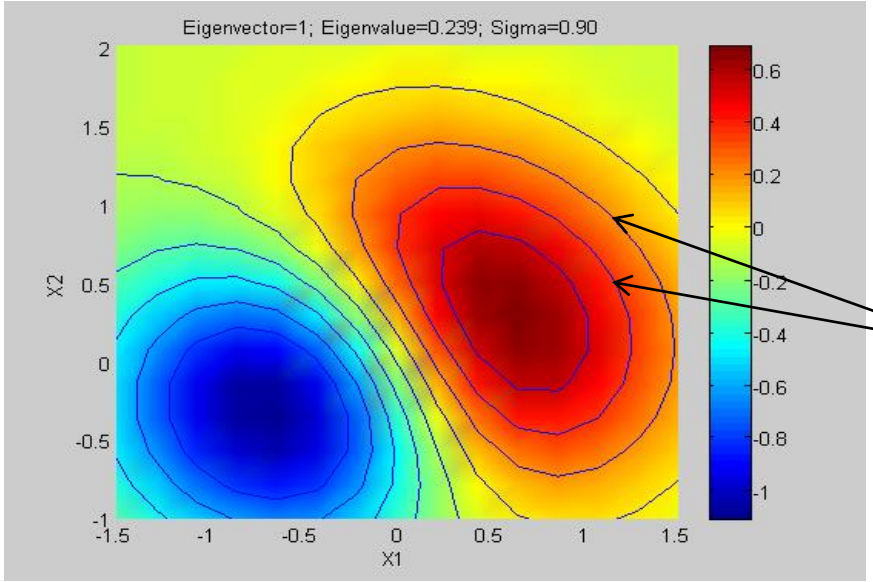
We cannot see the projection in feature space!

We can only compute the projections of each point onto each eigenvector

Projection of query point x onto eigenvector v^i:

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

Sum over all training points



Isolines group points with equal projection:

All points $x, s.t.: \left\langle v^i, \phi(x) \right\rangle = cst.$

kPCA projections: Exercise

Recall: projection of query point x onto eigenvector v^i:

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

where the α^i are the dual eigenvectors,

solutions of the eigenvalue decomposition of K.

Consider a $2-$dimensional data $-$space, with two datapoints,

and the RBF kernel:

$$k\left(x, x'\right) = e^{-\frac{\|x-x'\|^2}{\sigma^2}}$$

a) How many dual eigenvectors do you have and what is their dimension?

b) Compute the eigenvectors and draw the isolines for the projections

on each eigenvector.

c) Repeat (b) for a homogeneous polynomial kernel with p=2: $k\left(x, x'\right) = \left\langle x, x' \right\rangle^2$

kPCA projections: Exercise

Recall: projection of query point x onto eigenvector v^i:

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

where the α^i are the dual eigenvectors,

solutions of the eigenvalue decomposition of K.

Consider a $2-$dimensional data $-$space, with three equidistant datapoints,

and the RBF kernel:

$$k\left(x, x'\right) = e^{-\frac{\|x-x'\|^2}{\sigma^2}}$$

a) How many dual eigenvectors do you have and what is their dimension?

b) Compute the eigenvectors and draw the isolines for the projections

on each eigenvector.

c) Repeat (b) for a homogeneous polynomial kernel with p=2: $k\left(x, x'\right) = \left\langle x, x' \right\rangle^2$

d) What happens if you take 3 non-equidistant datapoints?

Curse of Dimensionality

Kernel PCA is very intensive computationally.

Computation of the eigenvectors requires eigenvalue decomposition of the Gram matrix (Kernel Matrix is *M x 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

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.