

# ***MACHINE LEARNING***

**Methods for feature extraction and  
reduction of dimensionality:**

**Probabilistic PCA and kernel PCA**

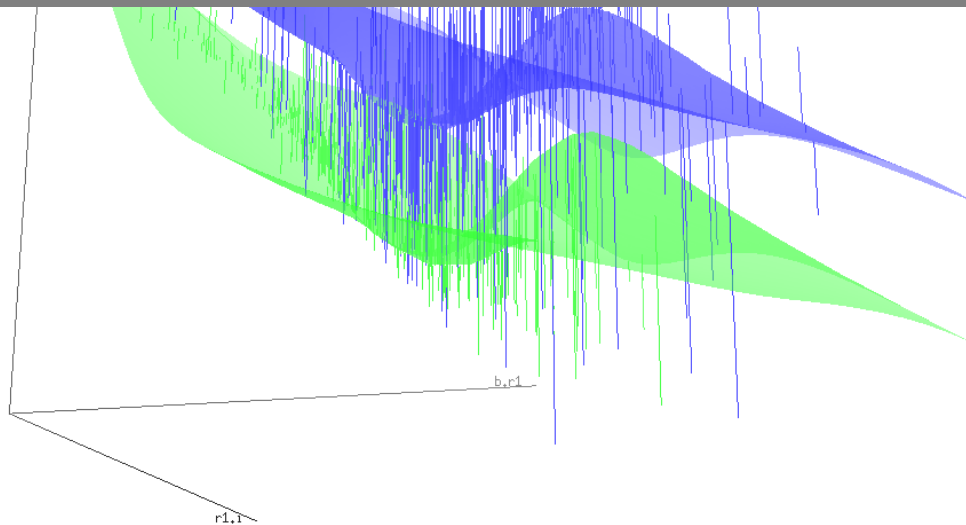


ÉCOLE POLYTECHNIQUE  
FÉDÉRALE DE LAUSANNE

# *Why reducing the data dimensionality?*

**Reducing the dimensionality** of the dataset at hand so that computation afterwards is more tractable

Idea: only a few of the dimensions matter, the projections of the data along the residual dimensions do not contain informative structure of the data (already a form of generalization)



# *Why reducing the data dimensionality?*

**The curse of dimensionality** refers to the exponential growth of volume covered by the parameters' values to be tested as the dimensionality increases.

In ML, analyzing high dimensional data is made particularly difficult because:

- ❑ Often one does not have enough observations to get good estimates (i.e. to sample enough all parameters).
- ❑ Adding more dimensions (hence more features) can increase the noise, and hence the error.

# *Principal Component Analysis*

- ❑ Principal Component Analysis is a method widely used in Engineering and Science.
- ❑ Its principle is based on statistical analysis of the **correlations** underpinning the dataset at hand
- ❑ Its popularity is due to the fact that:
  - ❑ Its computation is simple and tractable with an analytical solution
  - ❑ Its result can be easily visualized through usually a 2 or 3 dimensional graph.

## Co-Variance, Correlation

The covariance and correlation are a measure of the dependency between two variables. Given two variables  $x$  and  $y$  (assuming that  $x$  and  $y$  are both zero mean):

$$\text{cov}(x, y) = E\{x, y\} - E\{x\}E\{y\},$$

$$\text{corr}(x, y) = \frac{\text{cov}(x, y)}{\text{var}(x)\text{var}(y)}.$$

$x$  and  $y$  are said to be *uncorrelated*, if their covariance is zero:

$$\text{corr}(x, y) = 0 \text{ and } \text{cov}(x, y) = 0.$$

# Co-Variance Matrix

If  $X = \left\{ x_j^i \right\}_{j=1 \dots N}^{i=1 \dots M}$  is a multidimensional dataset containing  $M$   $N$ -dimensional datapoints, the covariance matrix  $C$  of  $X$  is given by:

$$C = E \{ XX^T \} = \begin{bmatrix} \text{cov}(X_1, X_1) & \dots & \text{cov}(X_1, X_N) \\ \dots & & \\ \text{cov}(X_1, X_N) & \dots & \text{cov}(X_N, X_N) \end{bmatrix}$$

$C$  is diagonal when the data  $X$  is decorrelated along each dimension.

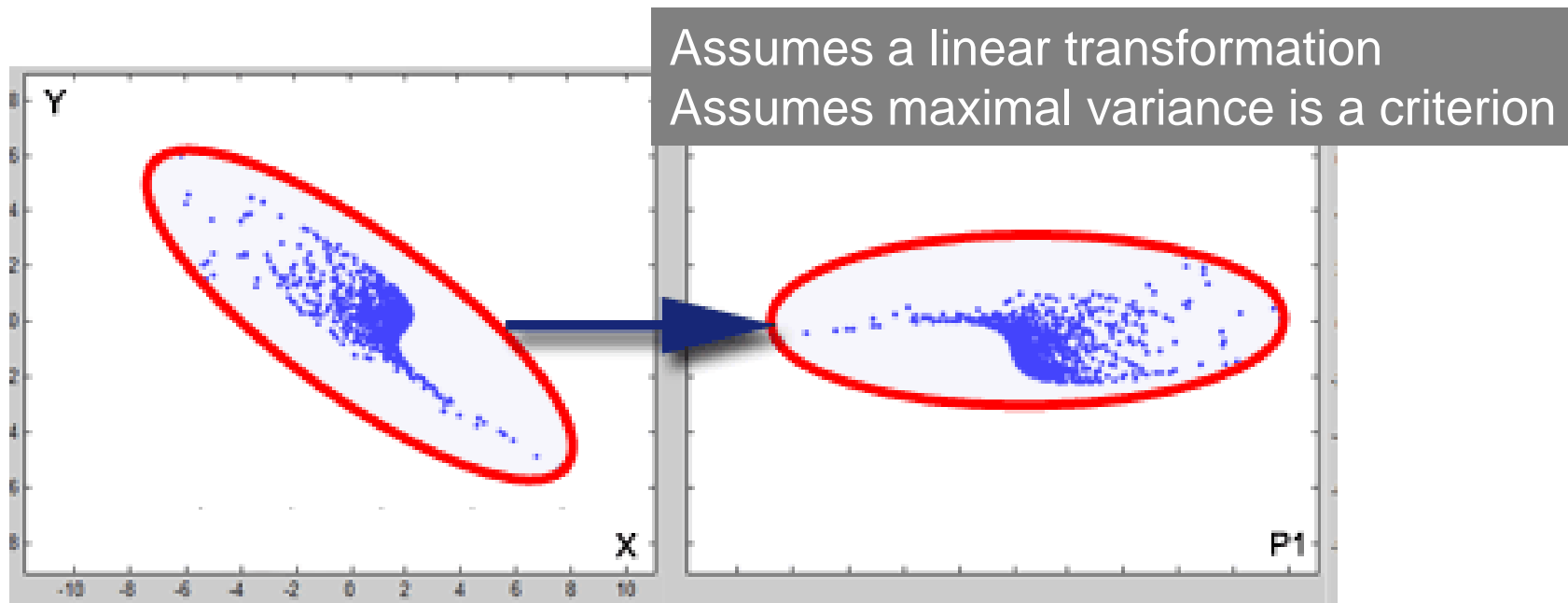
The rows  $X_j, j = 1 \dots N$ , represent the coordinate of each datapoint with respect to the  $j$ -th basis vector.

The column of  $X$  contain the  $M$  datapoints.

$C = E \{ XX^T \} \sim XX^T$ , since expectation is only a normalization factor.

# Purpose of PCA

Goal: to **find a better representation** of the dataset at hand so as to simplify computation afterwards



Raw 2D dataset

Projected onto two first Principal components

# PCA: principle

## PRINCIPLE:

- ❑ Define a low dimensional manifold in the original space.
- ❑ Represent each data point  $X$  by its projection  $Y$  onto this manifold.

---

## FORMALISM:

Consider a data set of  $M$   $N$ -dimensional data points

$$\mathbf{X} = \left\{ x_j^i \right\}_{j=1, \dots, N}^{i=1, \dots, M} \quad \text{and} \quad \mathbf{x}^i \in \mathbb{R}^N, i = 1, \dots, M :$$

PCA aims at finding a linear map  $A$ , s.t

$$A: \mathbb{R}^N \xrightarrow{A} \mathbb{R}^q, \quad q \leq N$$

$$Y = AX, \quad Y = \{y^1, \dots, y^M\} \quad \text{and each } y^i \in \mathbb{R}^q$$

# *PCA: principle*

There are three equivalent methods for performing PCA:

1. **Maximize the variance of the projection** (Hotelling 1933). In other words, this method tries to maximize the spread of the projected data.
2. **Minimize the reconstruction error** (Pearson 1901), i.e. to minimize the squared distance between the original data and its "estimate" in a low dimensional manifold.
3. Mean Least Error of the parameter in a latent variable (Tipping and Bishop 1996)

# Standard PCA: **Variance Maximization** through Eigenvalue Decomposition

## Algorithm:

1. Determine the direction (vector) along which the variance of the data is maximal.
2. Determine an orthonormal basis composed of the direction obtained in 1. The projection of the data onto each axis are uncorrelated.

## Standard PCA: **Variance Maximization** through Eigenvalue Decomposition

### Algorithm:

1) Zero mean:  $X \rightarrow X' = X - E\{X\}$

2) Compute Covariance matrix:  $C = E\{X'(X')^T\}$

3) Compute eigenvalues using  $|C - \lambda_i I| = 0, i = 1 \dots N$

4) Compute eigenvectors using  $Ce^i = \lambda_i e^i$

5) Choose first  $q < N$  eigenvectors:  $e^1, \dots, e^q$  with  $\lambda_1 \geq \lambda_2 \geq \dots \lambda_q$

6) Project data onto new basis:  $X' \rightarrow Y = A_q X', A_q = \begin{pmatrix} e_1^1 & \dots & e_N^1 \\ \dots & & \dots \\ e_1^q & \dots & e_N^q \end{pmatrix}$

# Standard PCA: **Variance Maximization** through Eigenvalue Decomposition



Demo PCA for Face Classification

# Principal Component Analysis

LIMITATION OF STANDARD and MSQ PCA:

The variance-covariance matrix needs to be calculated:

- Can be very computation-intensive for large datasets with a high # of dimensions
- Does not deal properly with missing data
- Incomplete data must either be discarded or imputed using ad-hoc methods
- Outliers can unduly affect the analysis

→ Probabilistic PCA addresses some of the above limitations

# Probabilistic PCA

The data  $X = \{x_j^i\}_{j=1\dots N}^{i=1\dots M}$  are samples of the distribution of the random variable  $x$ .

$x$  is generated by the latent variable  $z$  following:

$$x = W^T z + \mu + \varepsilon$$

The **latent variable  $z$**  corresponds to the *unobserved* variable. It is a *lower dimensional* representation of the data and their *dependencies*.

In Probabilistic PCA, the latent variable model consists then of:

- **$x$** : observed variables (*Dimension  $N$* )
- **$z$** : latent variables (*Dimension  $q$* )

with  $q < N$

Less dimensions results in more parsimonious models.

# Probabilistic PCA

The data  $X = \{x_j^i\}_{j=1\dots N}^{i=1\dots M}$  are samples of the distribution of the random variable  $x$ .

$x$  is generated by the latent variable  $z$  following:

$$x = W^T z + \mu + \varepsilon$$

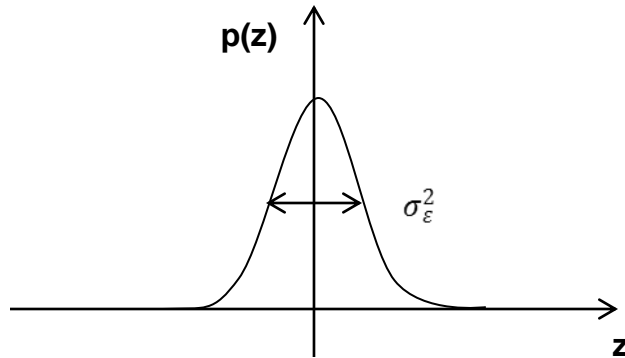
Assumptions:

- The latent variable  $z$  are centered and white, i.e.  $z = \mathbf{N}(0, I)$
- $\mu$  is a parameter (usually the mean of the data  $x$ )
- $\varepsilon$  the noise follows a zero mean Gaussian distribution  $\varepsilon = \mathbf{N}(0, \sigma_\varepsilon^2)$
- $W^T$  is a  $N \times q$  matrix.

Variance of the noise is diagonal

- conditional independence on the observables *given* the latent variables;
- $z$  encapsulates all correlations across original dimensions.

# Probabilistic PCA



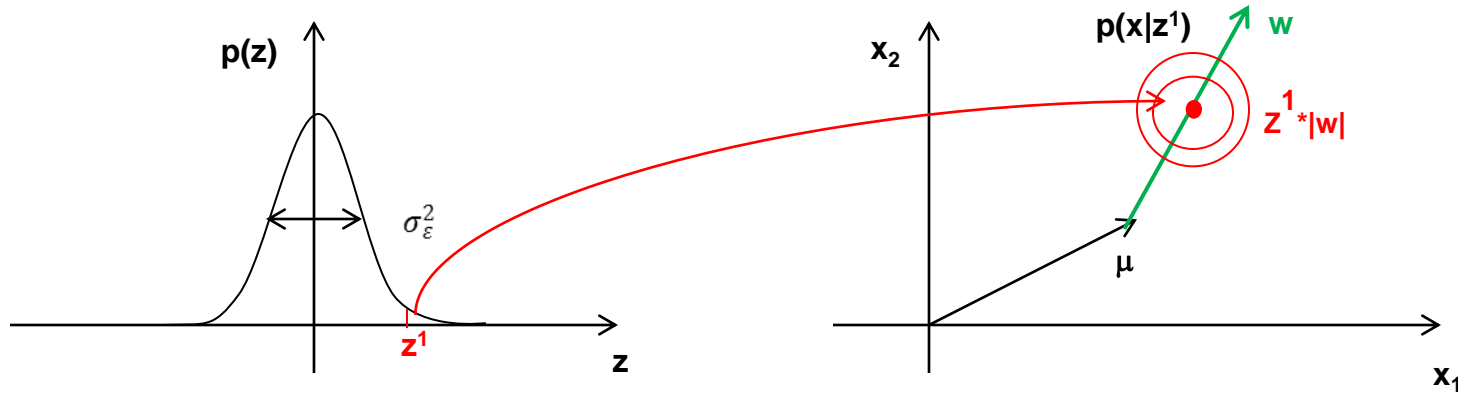
Assumptions:

- The latent variable  $z$  are centered and white, i.e.  $z = \mathbf{N}(0, I)$
- $\mu$  is a parameter (usually the mean of the data  $x$ )
- $\varepsilon$  the noise follows a zero mean Gaussian distribution  $\varepsilon = \mathbf{N}(0, \sigma_\varepsilon^2)$
- $W^T$  is a  $N \times q$  matrix.

Variance of the noise is diagonal

- conditional independence on the observables *given* the latent variables;
- $z$  encapsulates all correlations across original dimensions.

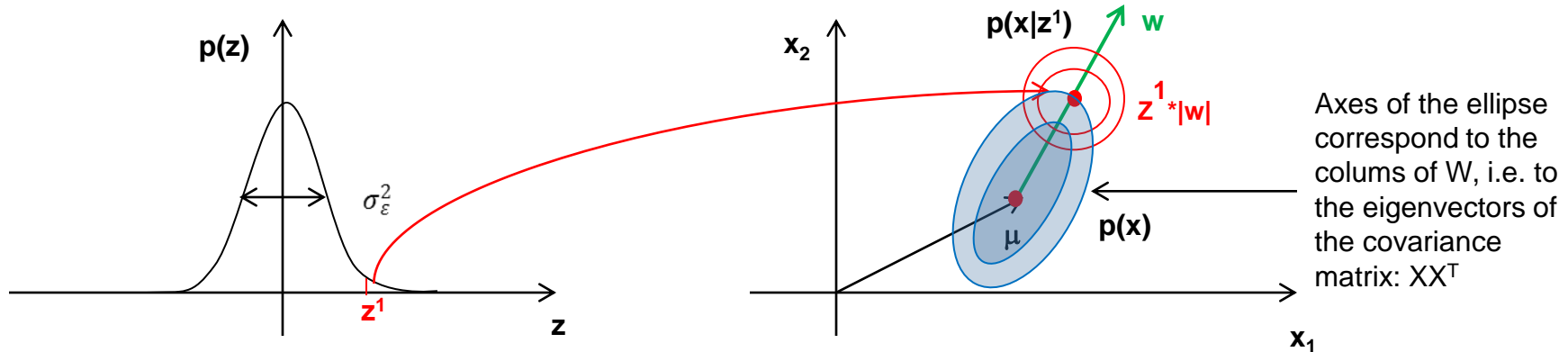
# Probabilistic PCA



Assuming further an isotropic Gaussian noise model  $N(0, \sigma_\varepsilon^2 \mathbf{I})$   
 $\rightarrow$  conditional probability distribution of the observables given  
 the latent variables  $p(x | z)$  is given by:

$$p(x | z) = N(W^T z + \mu, \sigma_\varepsilon^2 \mathbf{I})$$

# Probabilistic PCA



The marginal distribution can be computed by integrating out the latent variable and is then:

$$p(x) = N(\mu, W^T W + \sigma_\varepsilon^2 I)$$

Open parameters; can be learned through maximum likelihood

# Probabilistic PCA through Maximum Likelihood

If we set  $B = W^T W + \sigma_\varepsilon^2 I$ , one can then compute the log-likelihood:

$$L(B, \sigma_\varepsilon, \mu) = -\frac{M}{2} \left\{ N \ln(2\pi) + \ln |B| + \text{tr}(B^{-1}C) \right\}$$

where  $C$  is the sample covariance matrix of the complete set of  $M$  datapoints  $X = \{x^1, \dots, x^M\}$ .

The maximum likelihood estimate of  $\mu$  is the mean of the dataset  $X$ .

The parameters  $B$  and  $\sigma_\varepsilon$  are estimated through E-M.

See lecture notes for values of  $B$  and  $\sigma$  + exercises for derivation

# Probabilistic PCA through Maximum Likelihood

The use E-M to estimate the variables of PPCA offers a natural approach to the estimation of the principal axes in cases where some, or indeed all, of the data vectors  $X$  exhibit one or more *missing* values.

→ Exploit E-M approach to estimate the latent variables.

---

- Compute complete likelihood of the dataset (dataset is  $X$  and  $Z$ ):

$\log p(X, Z | \mu, W, \sigma_\varepsilon)$  and treat  $X$  as the missing data!

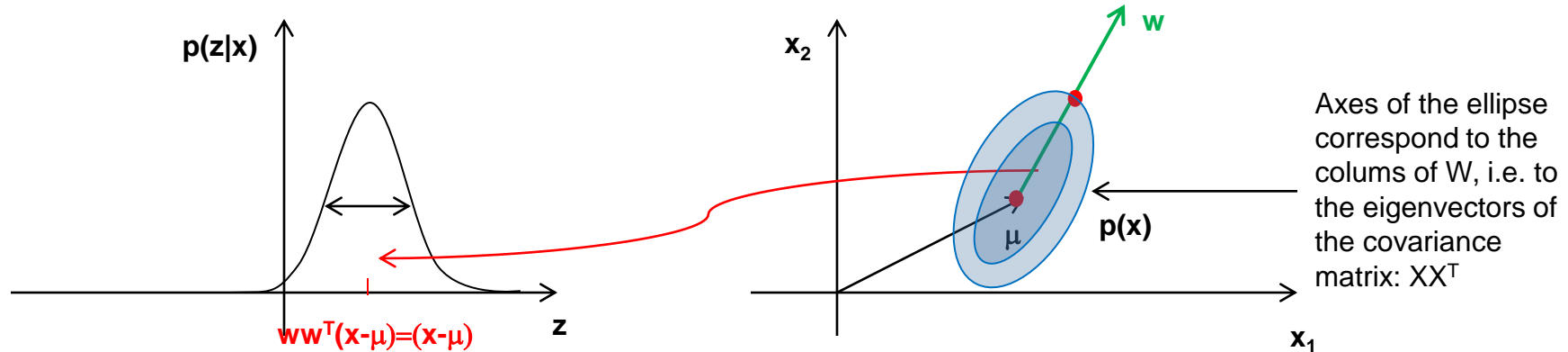
- E-step: compute expectation of complete log likelihood using estimate of  $p(z | x)$  and current parameters

- M-step: maximize parameters  $W, \sigma_\varepsilon$

- Iterate until likelihood no longer increases

← Allows on-line learning (incremental update)

# Probabilistic PCA



The conditional distribution of the latent variable over the data is given by:

**Is again Gaussian!**

$$p(z|x) = \mathcal{N}\left(B^{-1}W(x-\mu), B^{-1}\sigma_\varepsilon^2\right), \quad B = W^T W + \sigma_\varepsilon^2 I$$

In the absence of noise, one recovers standard PCA, as

$\left((W)^T W\right)^{-1} W(x-\mu)$  is an orthogonal projection of  $x$  onto the latent space.

# ***Probabilistic PCA: Dimensionality Reduction***

Reduction of the dimensionality is obtained by looking at the latent variable and estimating its distribution.

Reduce dimensionality by projecting onto a subset  $q$  of the dimensions

$$p(z | x) = \mathbf{N}\left(B^{-1}W(x - \mu), B^{-1}\sigma_{\varepsilon}^2\right), \quad B = W_q^T W_q + \sigma_{\varepsilon}^2 I$$

In the absence of noise, one recovers standard PCA, as

$\left((W)^T W\right)^{-1} W(x - \mu)$  is an orthogonal projection of  $x$  onto the latent space.

# Probabilistic PCA: Summary

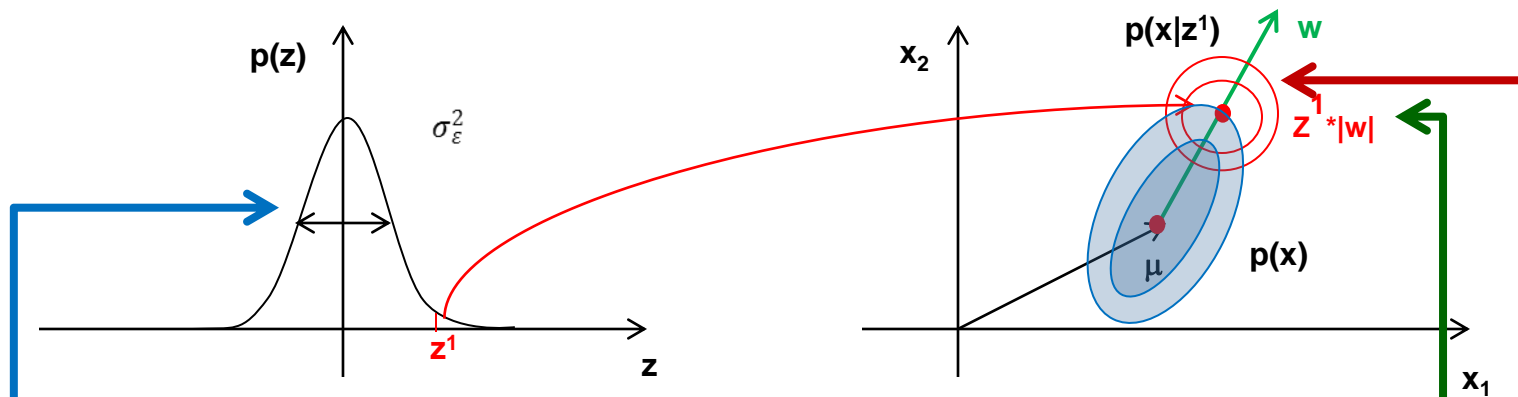
Idea: Assume that the data  $X$  were generated by a Gaussian latent variable model, Probabilistic PCA consists then in estimating the density of the latent variable through maximum likelihood.

Probabilistic PCA is then PCA through projection on a latent space.

Advantages of expressing PCA in probabilistic form:

- It can easily be extended to estimation from mixtures of PCA models.
- The estimated density can easily be used for classification and other Bayesian computation afterwards.

# Probabilistic PCA: Summary



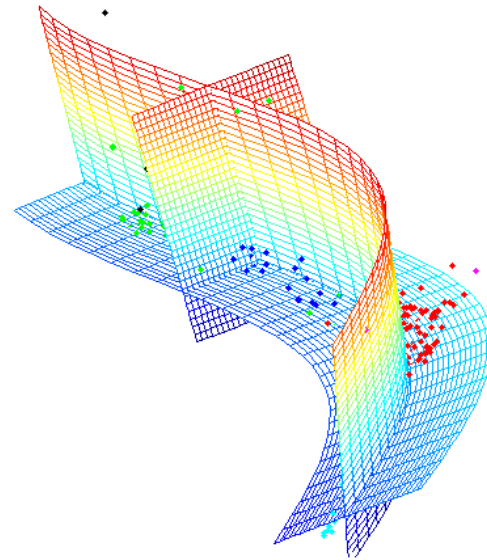
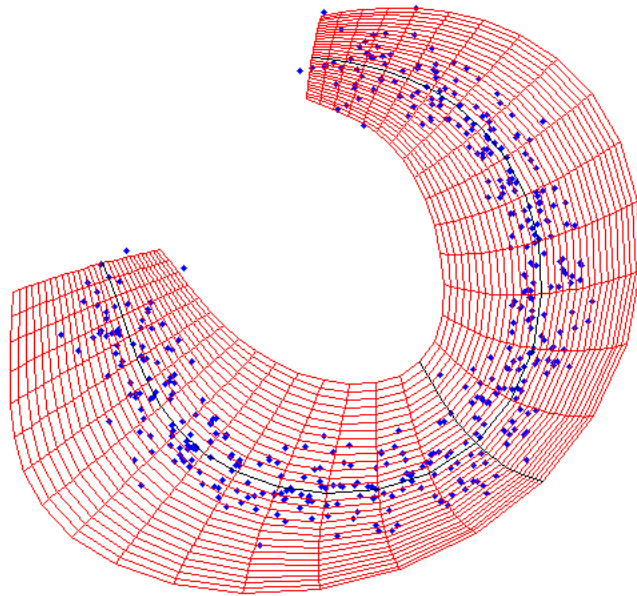
## Assumptions:

- underlying latent variable has a Gaussian distribution
- linear relationship between latent and observed variables
- isotropic Gaussian noise in observed dimensions

# Revisiting the hypotheses of PCA

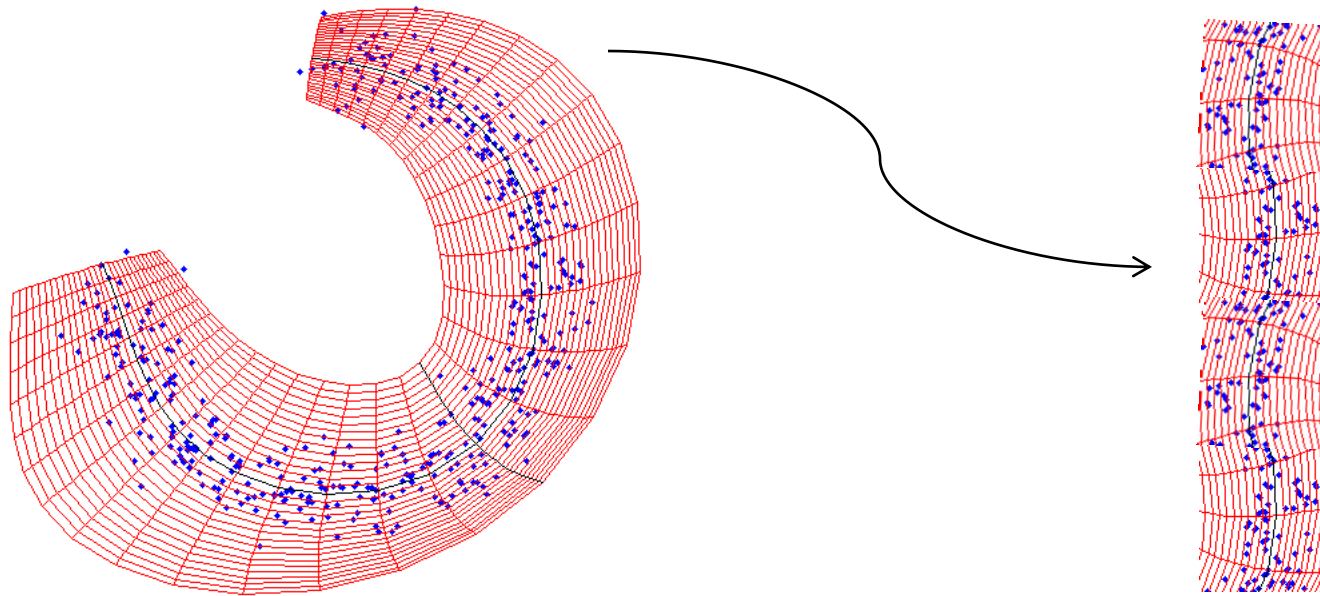
PCA assumed a *linear* transformation

→ *Non-linear* PCA ([Kernel PCA](#)): to find a non-linear embedding of the data



# Going back to linearity

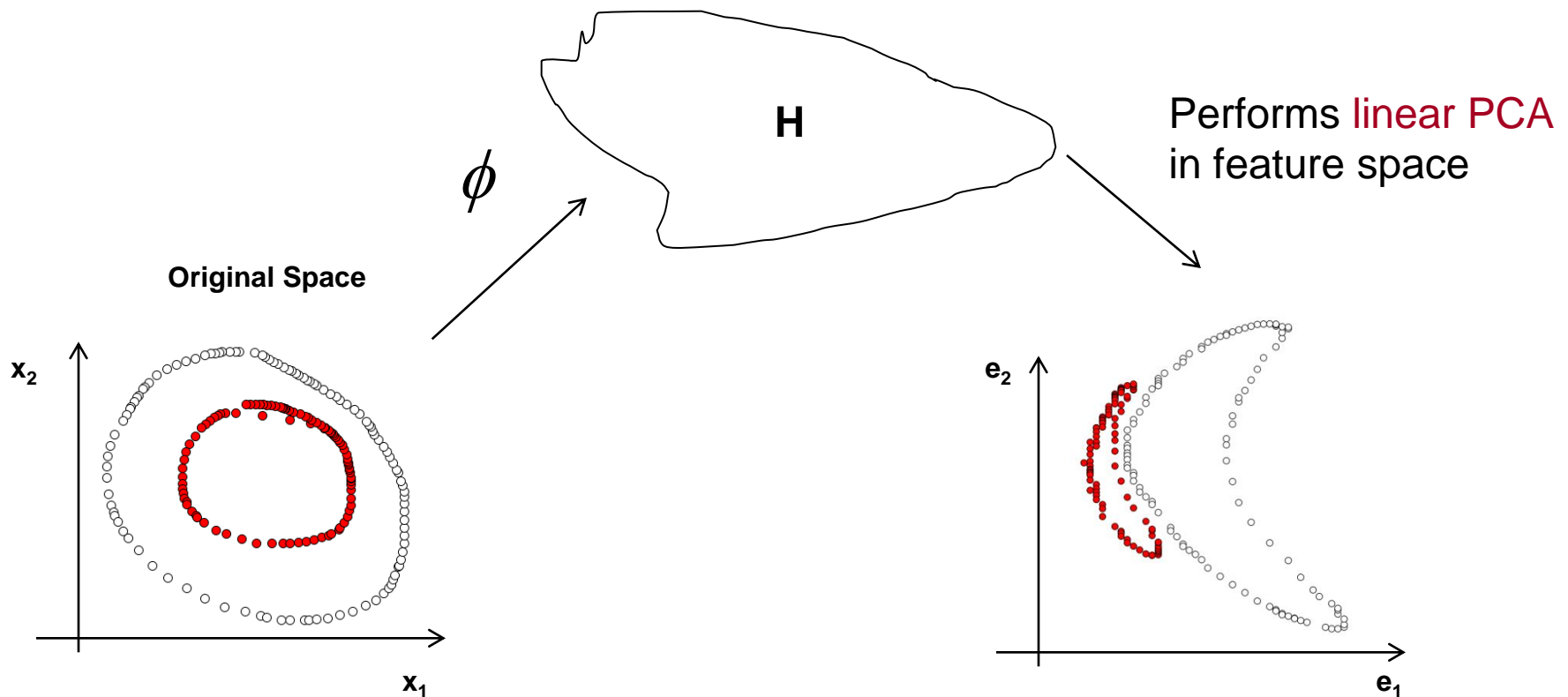
Find a non-linear transformation that send the data in a space where linear computation is again feasible.



# Kernel-Induced Feature Space

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

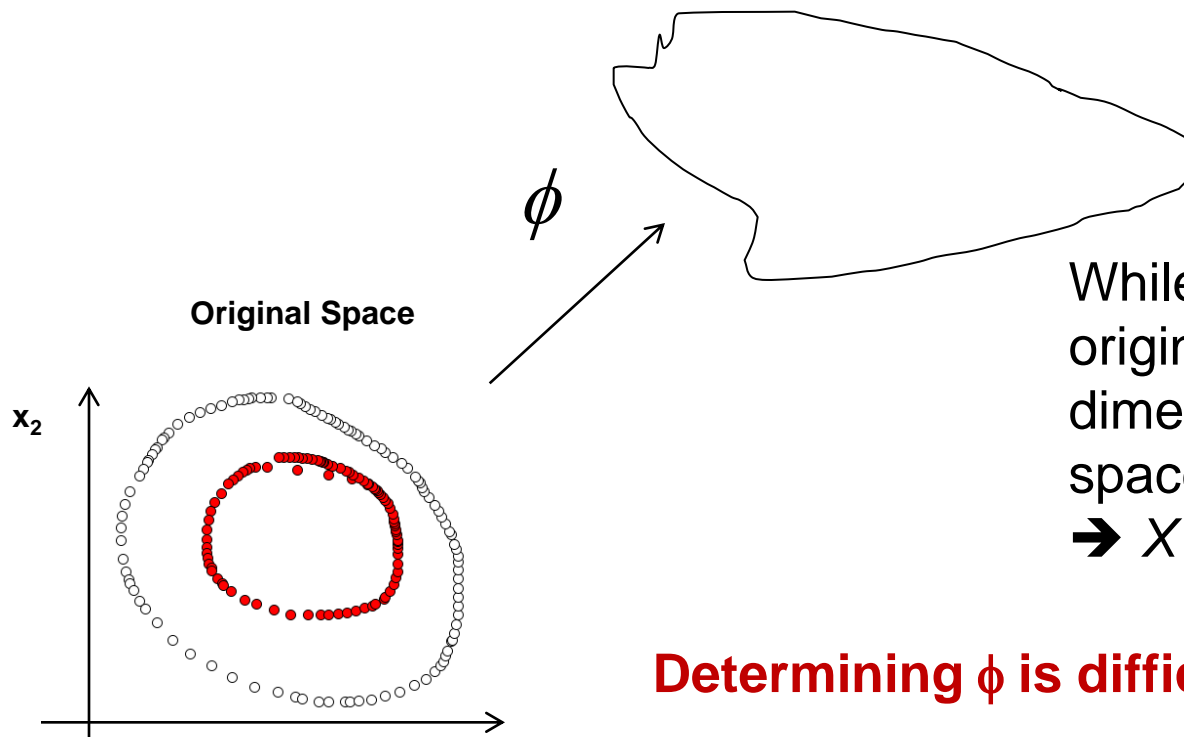
$$X = \{x^i \in \mathbb{R}^N\}_{i=1\dots M} \mapsto \phi(X) = (\phi(x^1), \dots, \phi(x^M))$$



# Kernel-Induced Feature Space

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

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



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:

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

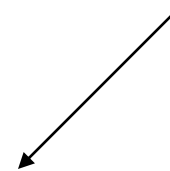
No need to compute the transformation  $\phi$ , if one expresses everything as a function of the inner product in feature space

→ the *kernel function*:

**Metric of similarity across datapoints**  
**May extract some features**

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

$$k(x^i, x^j) \rightarrow \langle \phi(x^i), \phi(x^j) \rangle.$$



# From Linear PCA to Kernel PCA

Rewriting PCA in terms of dot products:

Each eigenvector  $e^1, \dots, e^N$  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,  $e^i = \frac{1}{\lambda_i M} \sum_{j=1}^M x^j (x^j)^T e^i$


# From Linear PCA to Kernel PCA

Rewriting PCA in terms of dot products:

Each eigenvector  $e^1, \dots, e^N$  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,  $e^i = \frac{1}{\lambda_i M} \sum_{j=1}^M x^j \underbrace{(x^j)^T e^i}_{\alpha_j^i} = \frac{1}{\lambda_i M} \sum_{j=1}^M \alpha_j^i x^j.$


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 Correlation matrix in the feature space is:

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

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

# Linear PCA in Feature Space

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

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

$$\Rightarrow \langle \phi(x^j), C_\phi v^i \rangle = \lambda_i \langle \phi(x^j), v^i \rangle, \quad \forall i, j = 1, \dots, M$$

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

$$C_\phi v^i = \lambda_i v^i = \lambda_i \sum_{j=1}^M \alpha^j \phi(x^j)$$

# Linear PCA in Feature Space

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

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

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

→ eigenvalue problem of the form:

$$K \alpha^i = M \lambda_i \alpha^i, \quad M \in \mathbb{N}: \text{number of datapoints}$$

Dual eigenvalue problem of finding the eigenvectors  $v$  of  $C_{\phi}$ .

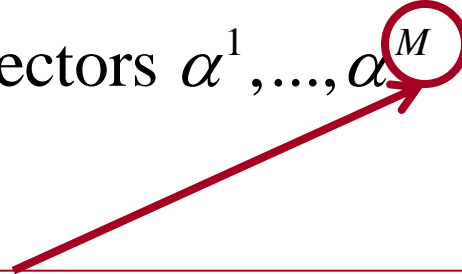
# Linear PCA in Feature Space

The solutions to the dual eigenvalue problem :  
are given by all the eigenvectors  $\alpha^1, \dots, \alpha^M$  with  
non-zero eigenvalues  $\lambda_1, \dots, \lambda_M$ .

Asking that the eigenvectors  $v$  of  $C_\phi$  be normalized,

$$\text{i.e. } \langle v^i, v^i \rangle = 1 \quad \forall i = 1, \dots, M$$

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

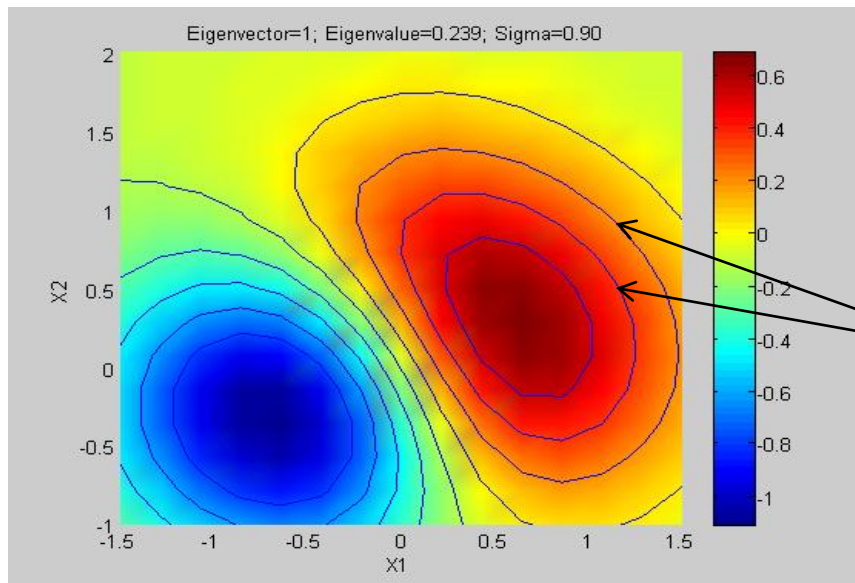
We cannot see the projection in feature space!

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

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

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

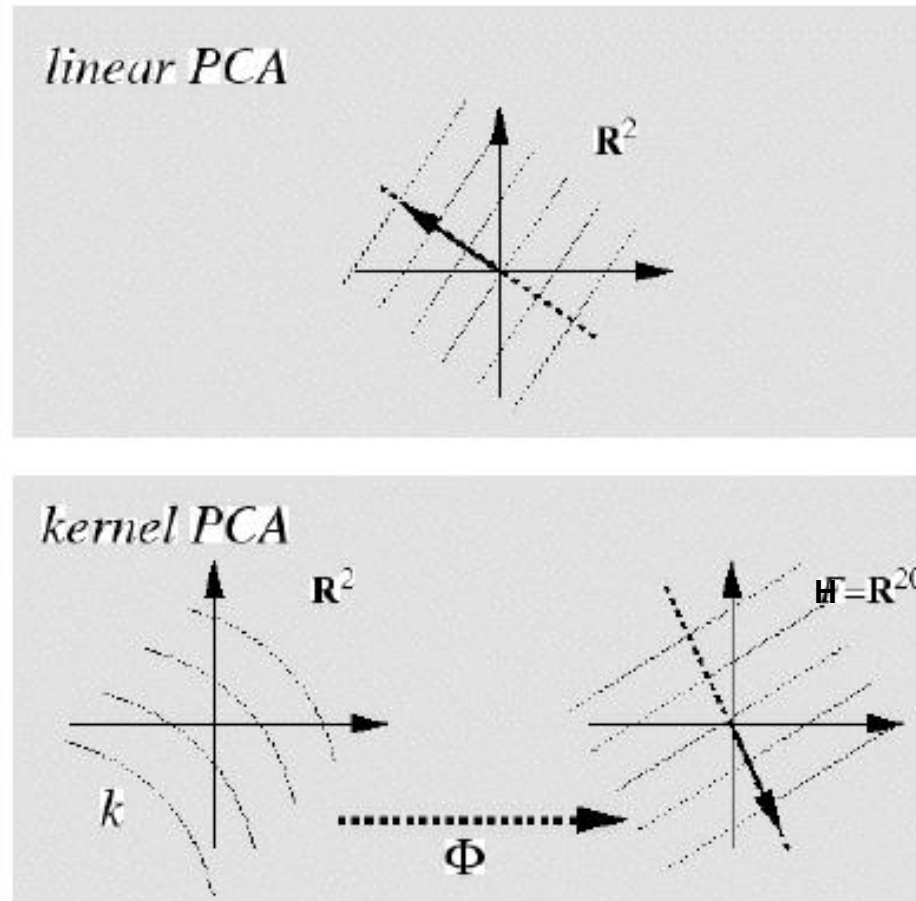
Sum over all training points



Contour lines group points with equal projection:

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

## Kernel PCA vs regular (linear) PCA



From Scholkopf & Smola, 2002

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):**

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

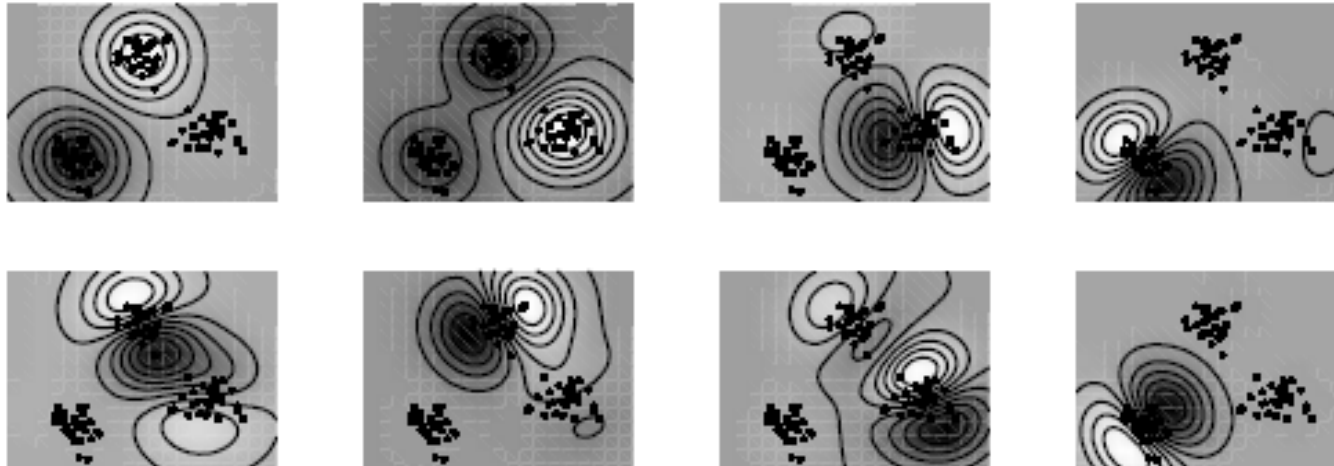
- **Homogeneous Polynomial Kernels:**

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

- **Inhomogeneous Polynomial Kernels:**

$$k(x, x') = (\langle x, x' \rangle + c)^p, \quad p \in \mathbb{N}, c \geq 0$$

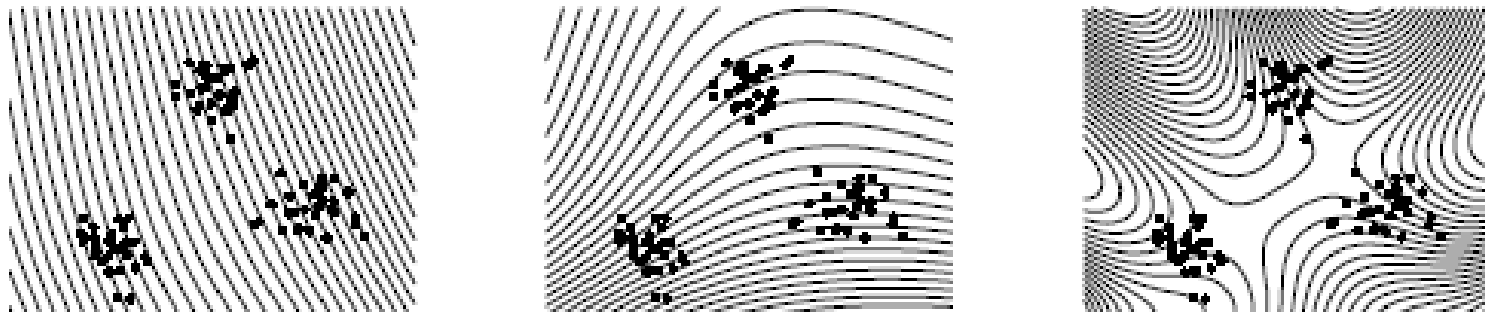
# Kernel PCA: Examples



From Scholkopf & Smola, 2002

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. The Matlab code used for generating this figure

# Kernel PCA: Examples

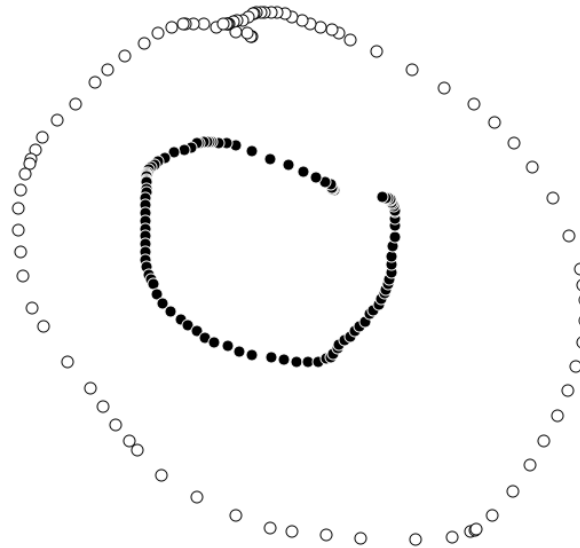


From Scholkopf & Smola, 2002

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  $k(\mathbf{x}, \mathbf{y}) = \tanh(2(\mathbf{x} \cdot \mathbf{y}) + 1)$ . 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

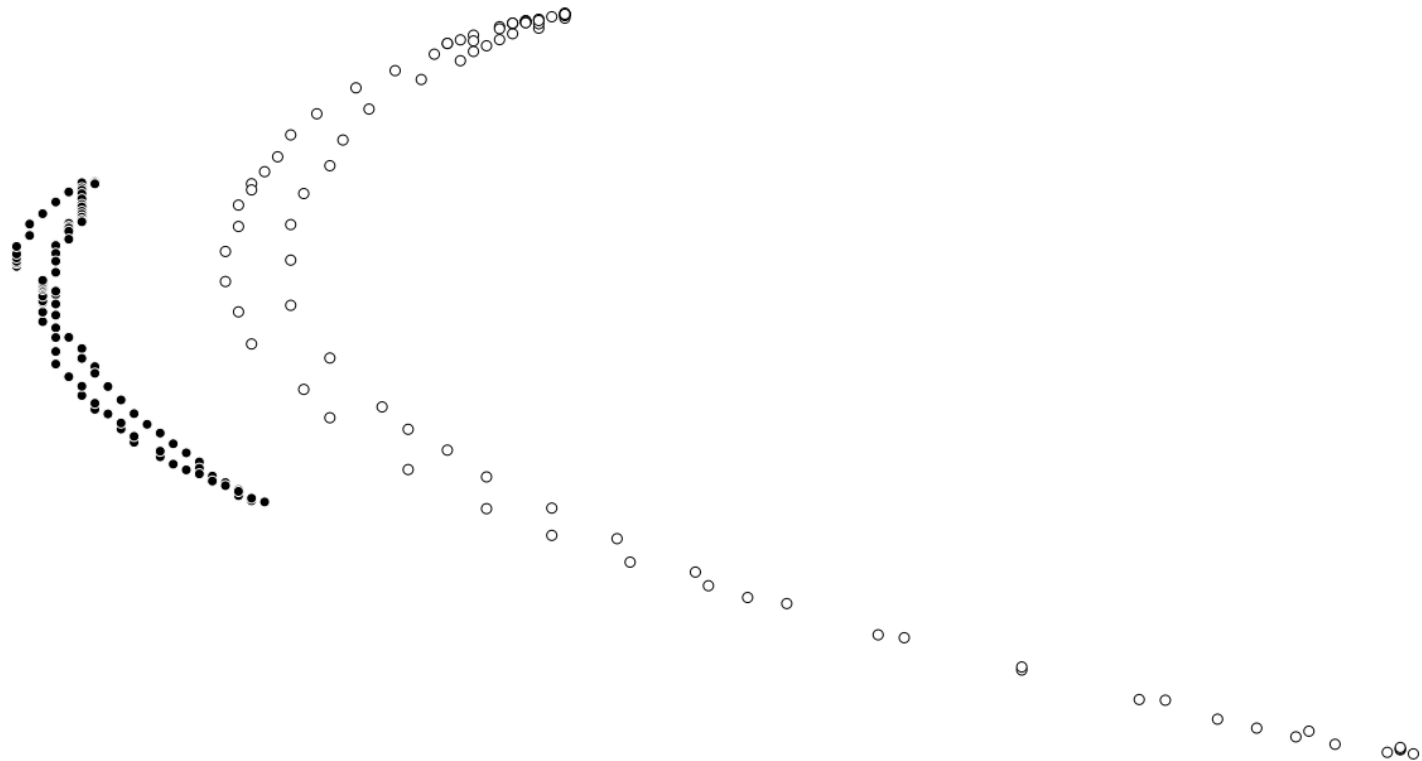
MLDEMOS Two sets of “circle” datapoints



**Original Data**

# Kernel PCA: Examples

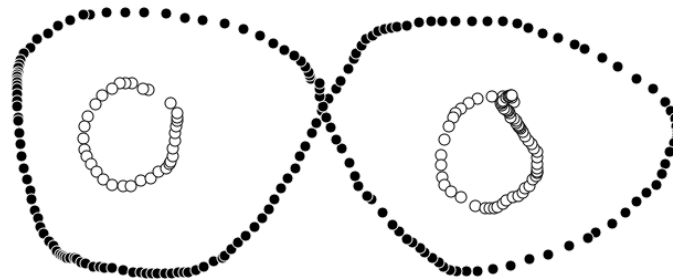
MLDEMOS Gaussian Kernel



**Projections onto first two eigenvectors**

# Kernel PCA: Examples

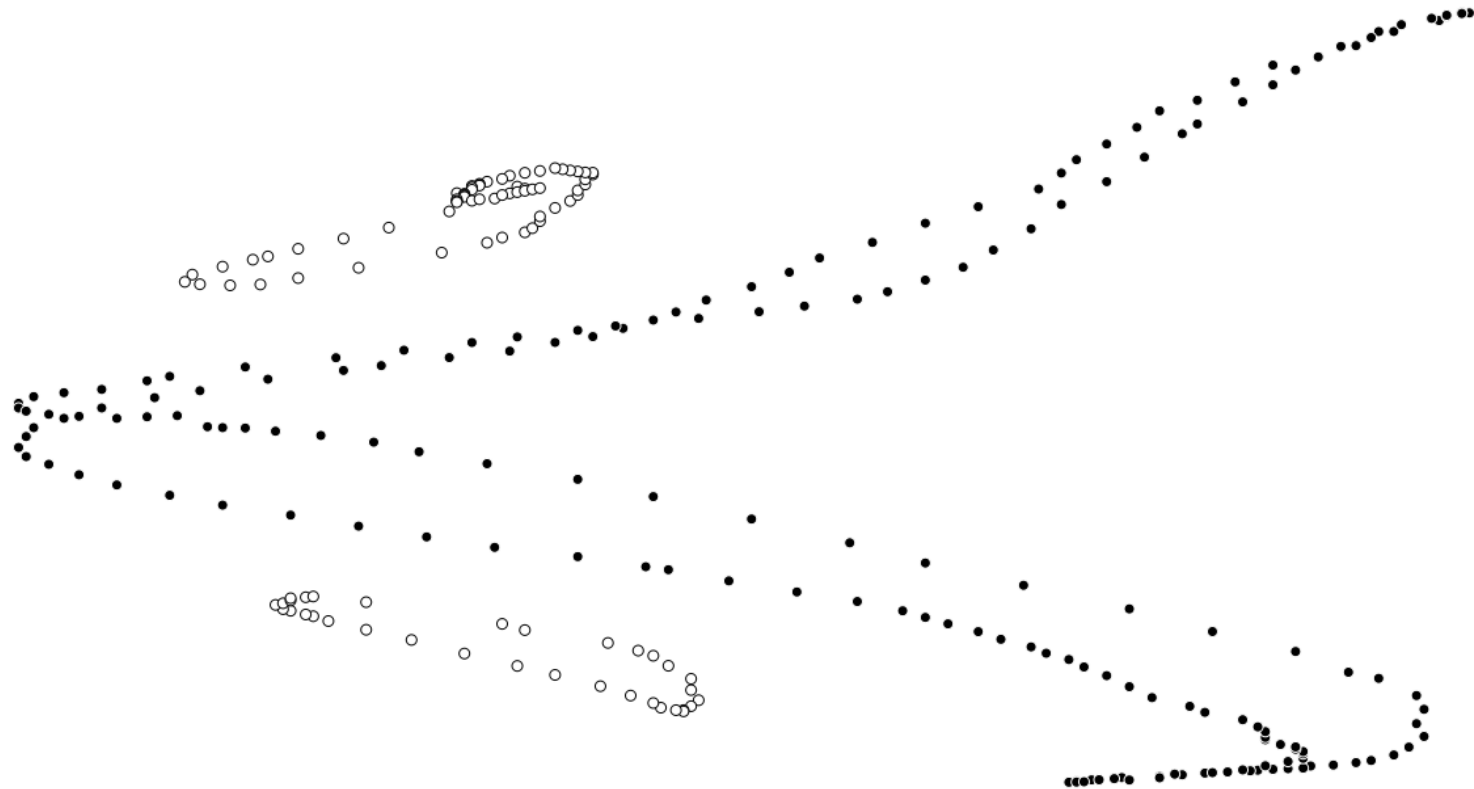
MLDEMOS “Pair of Glasses” datapoints



**Original Data**

# Kernel PCA: Examples

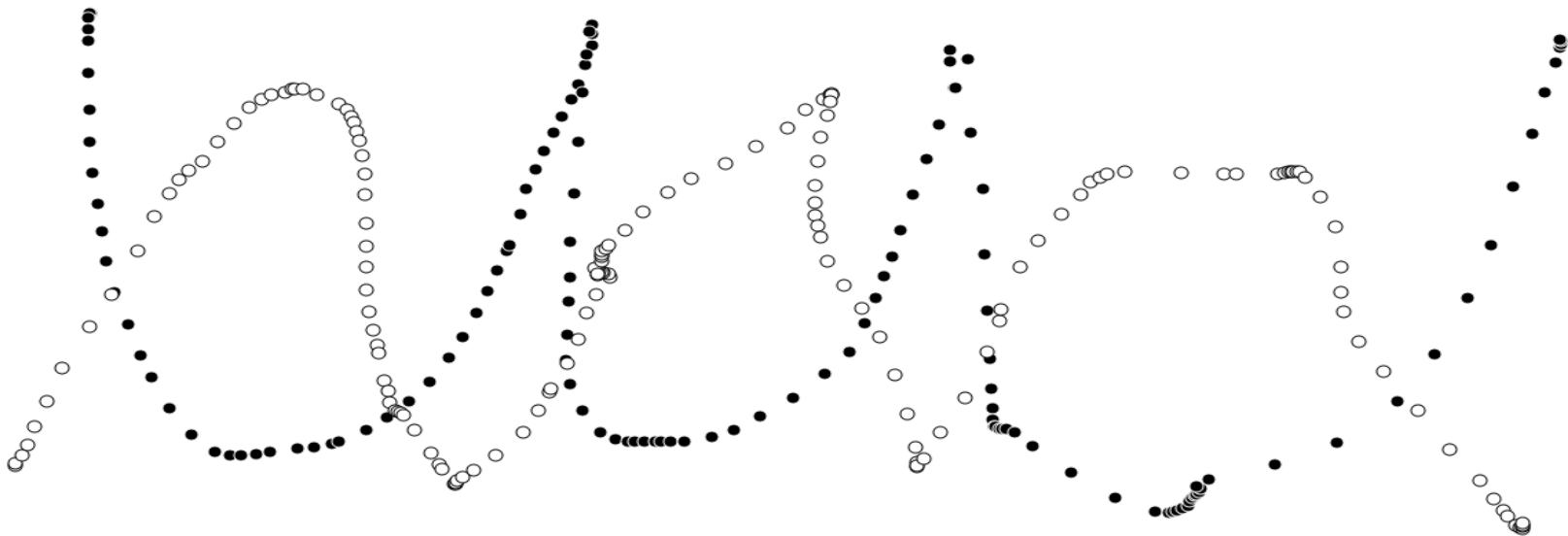
MLDEMOS Gaussian Kernel. kernel width=0.9



**Projections onto first two eigenvectors**

# Kernel PCA: Examples

MLDEMOS Two sets of “circle” datapoints



Original Data

# Kernel PCA: Examples

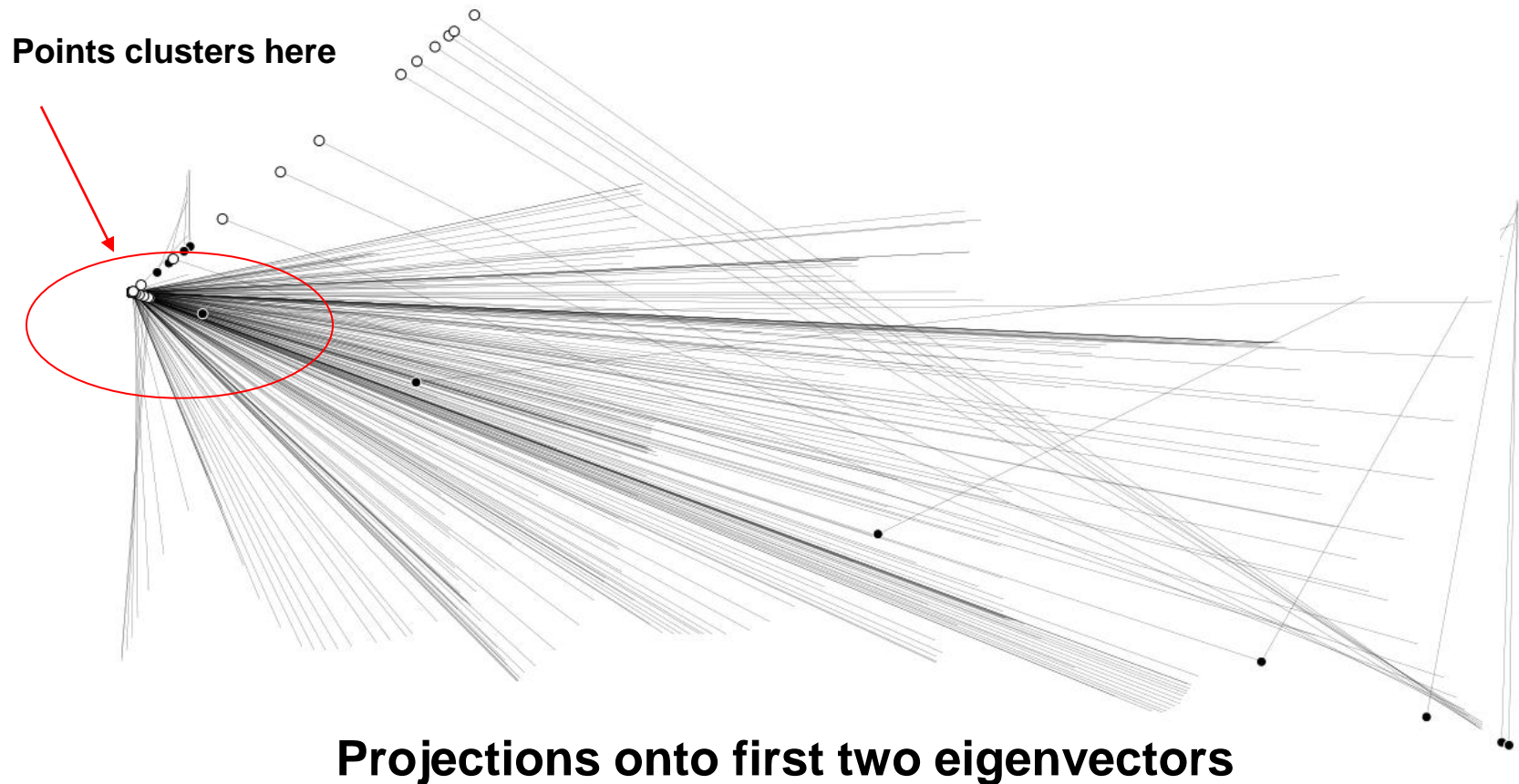
MLDEMOS Polynomial Kernel order  $p=20$



**Projections onto first two eigenvectors**

# Kernel PCA: Examples

MLDEMOS Polynomial Kernel order  $p=20$



# 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 quadratic ally with the number of data points  $M$ .

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

→ Sparse methods

## *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:

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

This is difficult to optimize. One must add further constraints, which can also help reduce the number of solutions.

## *Sparse $k$ PCA 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:

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

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.

# How to choose kernels?

- There is no rule for choosing the right kernel; each kernel must be adapted to a particular problem.
- Do a grid search over values of kernel parameters and perform crossvalidation for each choice.
- Some considerations are important:
  - Kernel parameters are often related to geometrical properties of data; e.g kernel width in rbf kernel relates to size of the variance of the data
  - Experimentally, there is some “robustness” in the choice, if the chosen kernels provide an acceptable trade-off between
    - simpler and more efficient structure (e.g. linear separability), which requires some “explosion”
    - Information structure preserving, which requires that the “explosion” is not too strong.