EE769 Intro to ML
Dimension Reduction

Amit Sethi
Faculty member, IIT Bombay

Learning objectives

- Write the advantages of dimension reduction
- Write steps for principal component analysis
- Extend PCA to nonlinear methods using the kernel trick

Dimensionality reduction: what & why?

$x \in \mathbb{R}^{100}$ $\quad y \in \mathbb{R}^{5}$

$X \in \mathbb{R}^{N \times D}$

- Objective:

  $= i$, find mapping $f : x_i \rightarrow y_i,\ y_i \in {}^{d},\ d<D$

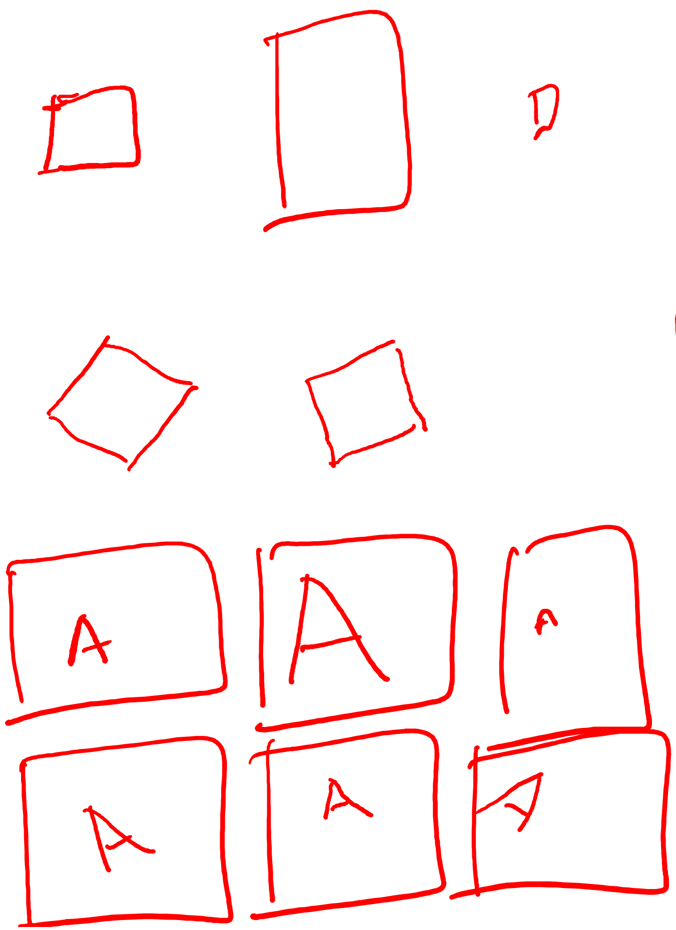
  - Such that, there exists f^{-1} that can (almost) reconstruct $= g : y \rightarrow \hat{x}_i$

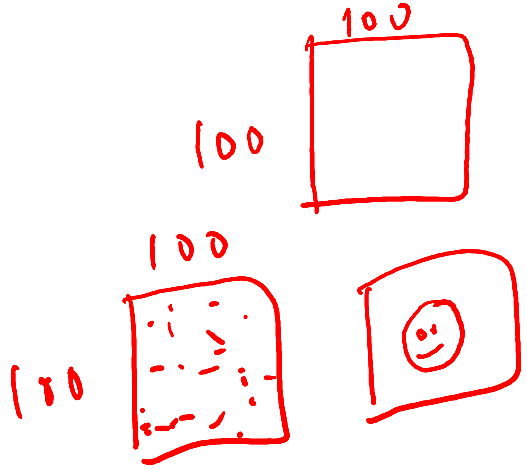
$\hat{x}_i \in \mathbb{R}^{D}$

$\| \hat{x}_i - x_i \|_2^2$

- Advantages:
  - Less redundancy, easier classification
  - Smaller storage, faster search
  - **Unravel meaningful latent variables**

Example: Only two latent variables





10,000

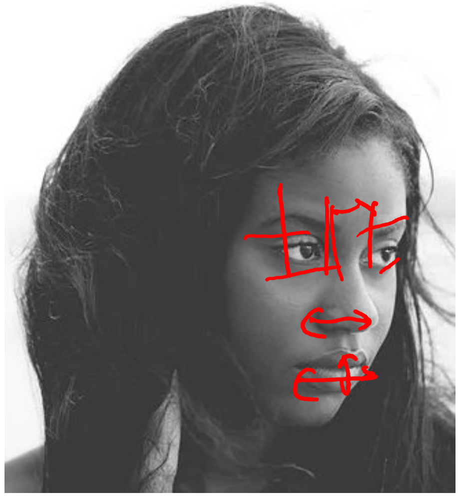
$x_i \in \mathbb{R}^{10,000}$

$y \equiv (\text{Size}, \text{Rotation})$

$y \in \mathbb{R}^2$

Example 2: Many latent variables, but still less than "pixels"

100



100

100 100

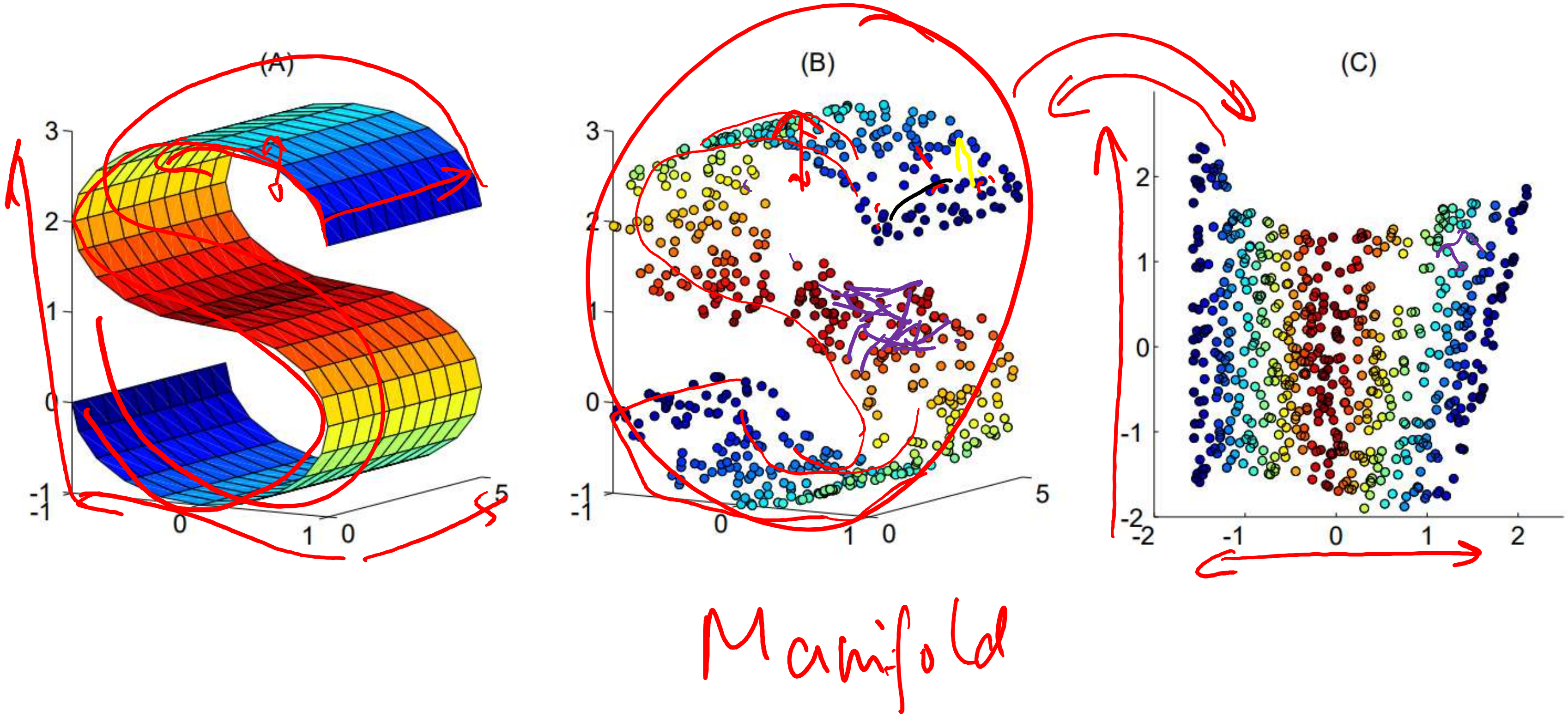
100

100

- What defines the shape of a face?
- Add pose and expression…
- Still a lot less variables than the pixels of a face image

Image source: Pixabay.com

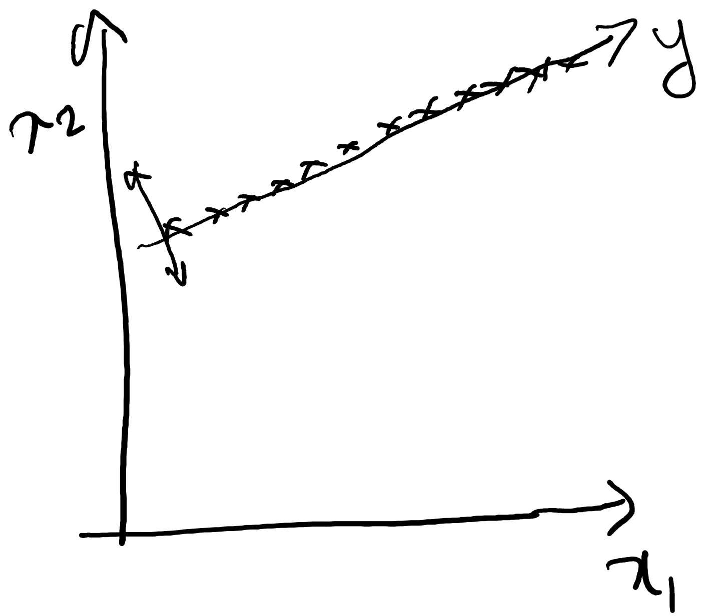
Visualizing manifolds



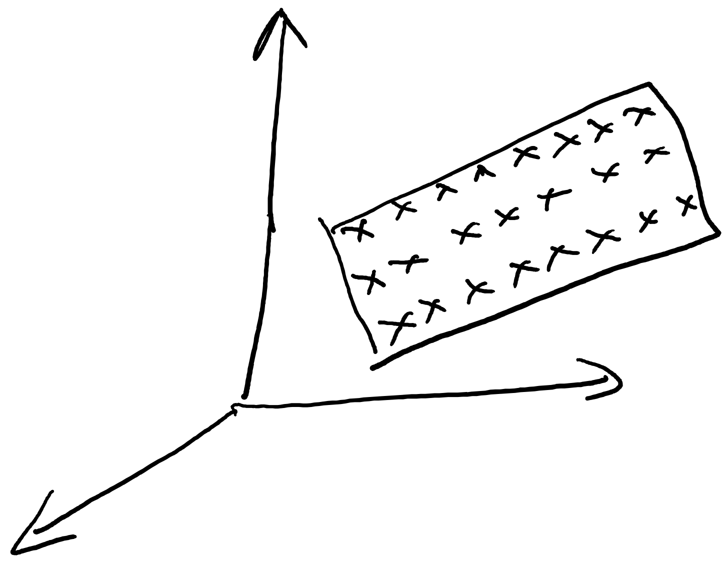
“An Introduction to Locally Linear Embedding,” Lawrence K. Saul

Inherent linear dimension of data

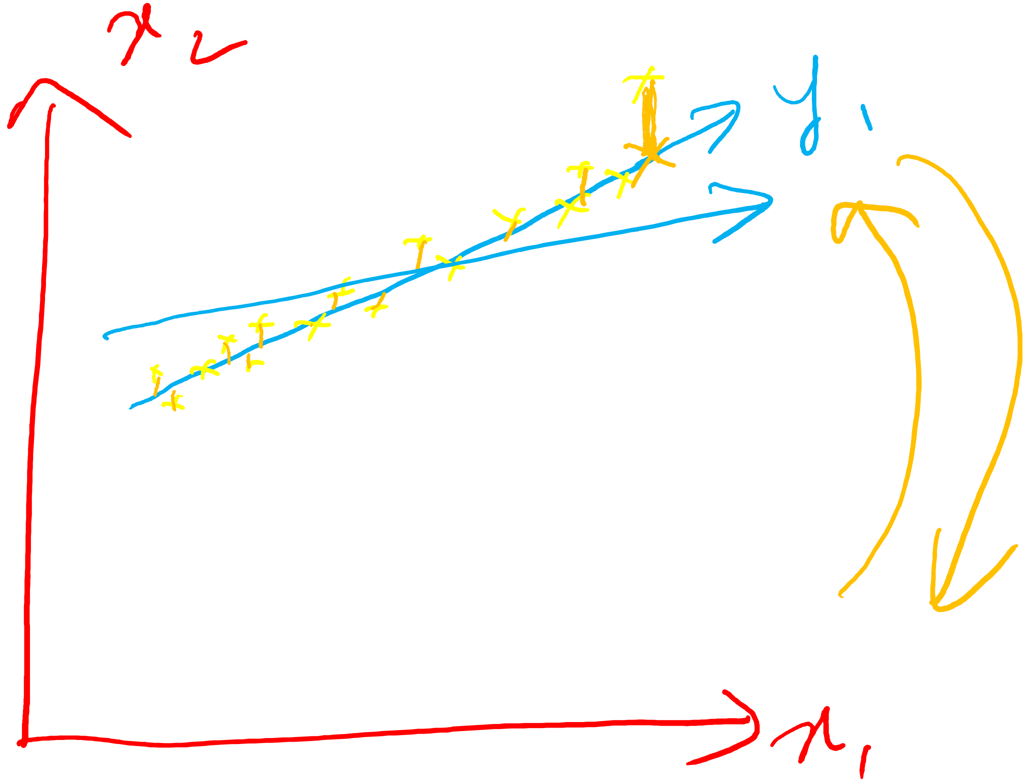
Data along a line on a 2-D plane



Data along a line or hyperplane in 3-D



Deviations from the underlying hyperplane



$y_i = w_1 x_{1i} + w_2 x_{2i}$

Principal component analysis

Objective:

- Find a way to select top $d < D$ orthogonal directions that explain the maximum possible variance of the data

Outline:

- Mean-center the data
- Loop
  - Find direction of maximum variance (that is orthogonal to all previously found directions)
- Until desired percent of variance is captured or number of dimensions

Complete algorithm PCA

$X \in \mathbb{R}^{N \times D}$

- Mean centering: $\boldsymbol{z}_i = \boldsymbol{x}_i - \boldsymbol{\mu}_{\boldsymbol{x}}$
- Covariance: $N\,\boldsymbol{C} = \boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z}$ $\quad C \in \mathbb{R}^{D \times D}$ $\quad$ C is P.S.D.
- Eigen decomposition: $\boldsymbol{C} = \boldsymbol{U}\,\boldsymbol{\Lambda}\,\boldsymbol{U}^{\mathrm{T}}$; $\lambda_j \boldsymbol{u}_j = \boldsymbol{C}\,\boldsymbol{u}_j$ $\rightarrow$ unit vector
- Selection: $\boldsymbol{C}_d = \boldsymbol{U}\,\boldsymbol{\Lambda}_d\,\boldsymbol{U}^{\mathrm{T}} = \boldsymbol{U}_d\,\boldsymbol{\Lambda}_d\,\boldsymbol{U}_d^{\mathrm{T}},\ d<D$ $\quad [u_1, u_2\ \boxed{\dots u_D}] = U$
  - Drop dimensions: Set lower $D-d$ eigenvalues to zero
- Projection: $\boldsymbol{Y} = \boldsymbol{Z}\,\boldsymbol{U}_d$

$N \times d \qquad N \times D \quad D \times d \qquad \lambda_{d+1} \dots \lambda_D$

$$\begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \boxed{\ddots\ \lambda_D} \end{bmatrix} = \Lambda$$

- Reconstruction: $\boldsymbol{Z}_d = \boldsymbol{Y}\,\boldsymbol{U}_d^{\mathrm{T}} = \boldsymbol{Z}\,\boldsymbol{U}_d\,\boldsymbol{U}_d^{\mathrm{T}}$ $\quad N \times D$
- Reconstruction error: $\|\boldsymbol{Z} - \boldsymbol{Z}_d\|_2^2 \propto |\boldsymbol{\Lambda} - \boldsymbol{\Lambda}_d|$

$\lambda_1 \geq \lambda_2 \geq \dots \lambda_D \geq 0$

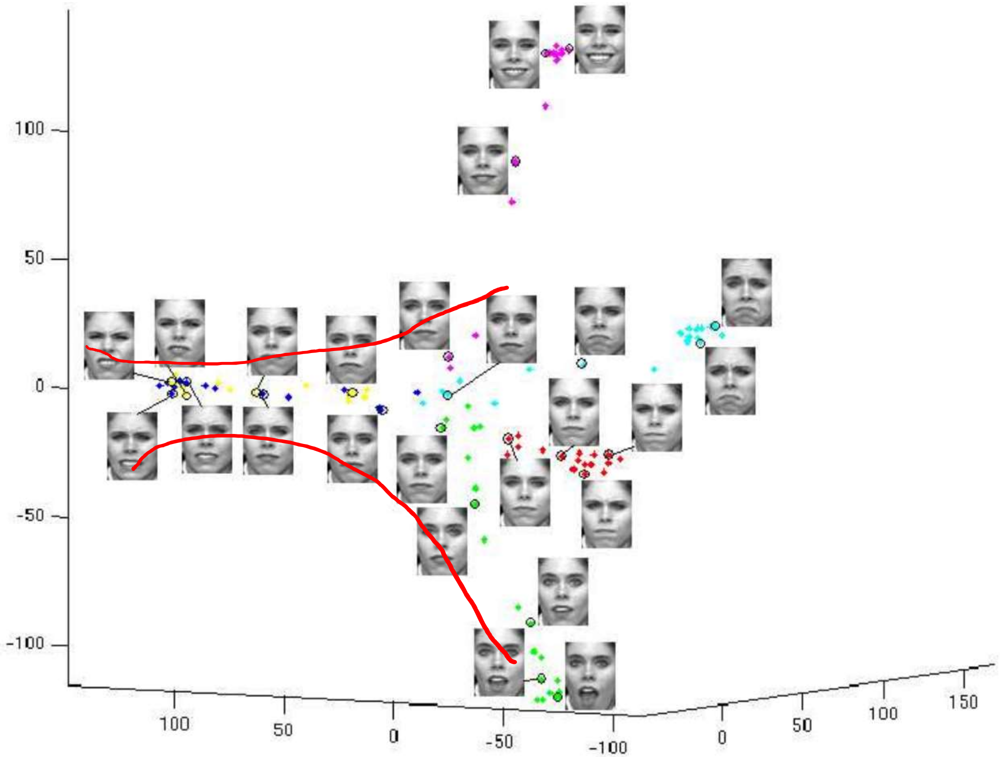
$= \sum_{i=d+1}^{D} \lambda_i$

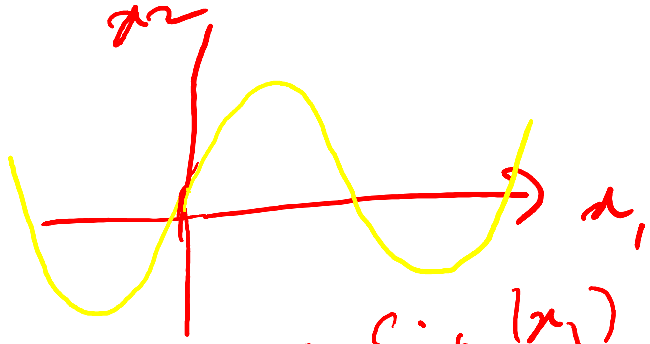
Example: Eigenfaces

7 P.C. of face images

Turk, Matthew, and Alex Pentland. "Eigenfaces for recognition." *Journal of cognitive neuroscience* 3.1 (1991): 71-86.

Data often lies around a nonlinear manifold





$x_2 = \sin(x_1)$

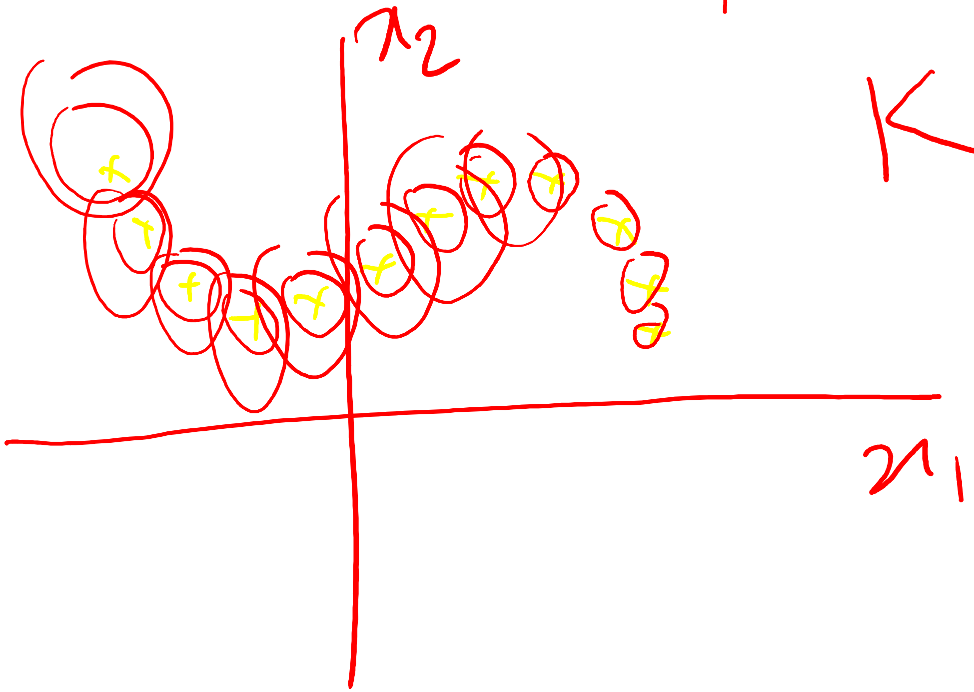
$y = f(x_1)$

Shan, Caifeng, Shaogang Gong, and Peter W. McOwan. "Appearance manifold of facial expression." *International Workshop on Human-Computer Interaction*. Springer, Berlin, Heidelberg, 2005.

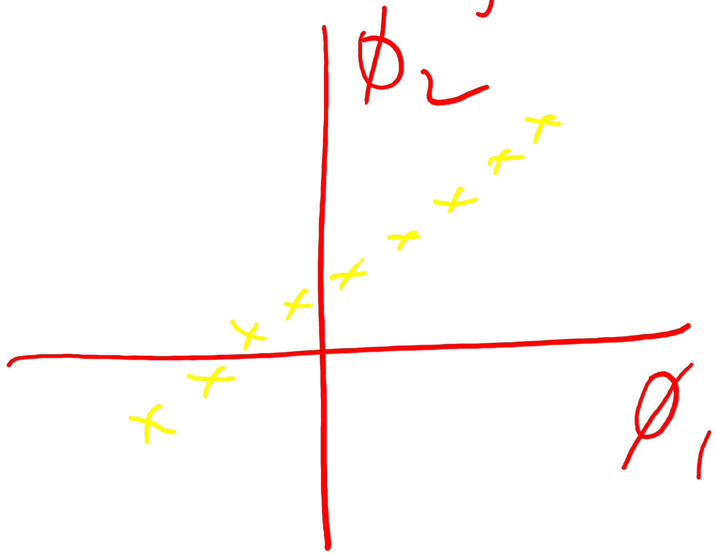
Kernel PCA introduces nonlinearity by mapping data to a feature space

$\underline{x}_i^T \underline{x}_j \qquad \phi(x_i)^T \phi(x_j) \qquad k(x_i, x_j)$

k_{ij}



K



Kernel PCA avoids calculating features directly by using the kernel trick

- Recall PCA: $\boldsymbol{C}\,\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i\,;\; N\,\boldsymbol{C} = \boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z}$
- Substituting features: $\sum_{\mathrm{n}} \boldsymbol{\varphi}(\boldsymbol{z}_{\mathrm{n}})\,\boldsymbol{\varphi}(\boldsymbol{z}_{\mathrm{n}})^{\mathrm{T}}\,\boldsymbol{u}_{\mathrm{i}} = N\,\boldsymbol{\lambda}_i \boldsymbol{u}_i$
- Substitute: $u_i = \sum_{\mathrm{n}} a_{in}\,\phi(z_{\mathrm{n}})$
- This means: $\sum_n k_{l,n} \sum_m a_{im}\,k_{n,m} = N\,\lambda_i \sum_n a_{in}\,k_{l,n}$
- In matrix form: $\boldsymbol{K}^2\,\boldsymbol{a}_{\mathrm{i}} = \lambda_{\mathrm{i}}\,N\,\boldsymbol{K}\,\boldsymbol{a}_{\mathrm{i}}$
- Solve eigenvalue problem: $\boldsymbol{K}\,\boldsymbol{a}_{\mathrm{i}} = \lambda_{\mathrm{i}}\,N\,\boldsymbol{a}_{\mathrm{i}}$

$K_{i,j}$

$K \in \mathbb{R}^{N \times N}$ $\quad N \times N \quad = \phi(z_i)^T \phi(z_j)$

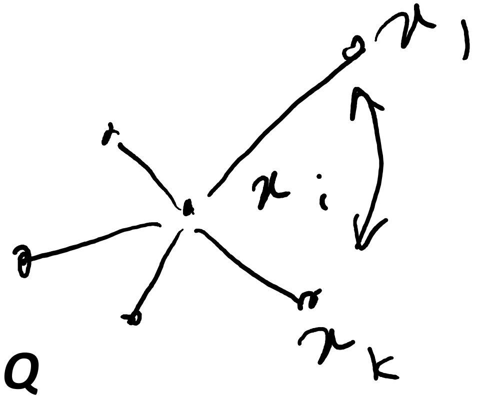
- There is a mean-centering step:
  - Ref: https://www.ics.uci.edu/~welling/classnotes/papers_class/Kernel-PCA.pdf

Schölkopf, Bernhard (1998). "Nonlinear Component Analysis as a Kernel Eigenvalue Problem". *Neural Computation*. **10** (5): 1299–1319.

Stochastic Neighbor Embedding

t-SNE

- $p_{j|i}$ = *softmax* $(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / 2\sigma_i^2)$
- $q_{j|i}$ = $\cancel{\text{softmax}\,(-\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)}$
- Objective: Reduce KL divergence between ***P*** and ***Q***

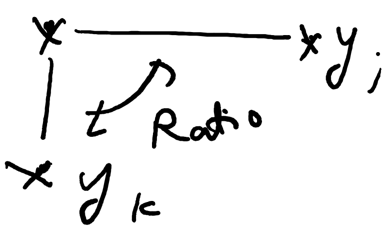


$$\frac{e^{-\|x_i - x_j\|/2\sigma_i^2}}{\sum_{k \neq i} e^{-\|x_i - x_k\|/2\sigma_i}}$$

N×N   N×N

$$\frac{(1+\|y_i - y_j\|)^{-1}}{\sum_k \cdots}$$

$$\sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$



"Stochastic Neighbor Embedding," Geoffrey Hinton and Sam Roweis

Auto-encoder bottlenecks for dimension reduction