Machine Learning for Robotics
Intelligent Systems Series
Lecture 4

Georg Martius
Slides adapted from Christoph Lampert, IST Austria

MPI for Intelligent Systems, Tübingen, Germany

May 15, 2017

Unsupervised Learning
Clustering

Clustering

Given: data

$$X = \{x^1, \dots, x^m\} \subset \mathbb{R}^d$$

Clustering – Transductive

Task: partition the point in X into **clusters** $S_1, \dots, S_K$.

Idea: elements within a cluster are similar to each other, elements in different clusters are dissimilar

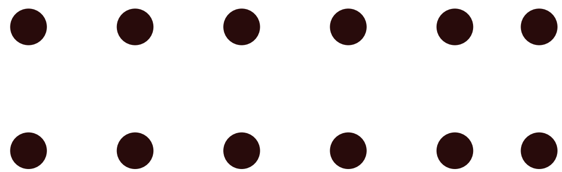
Clustering – Inductive

Task: define a partitioning function $f : \mathbb{R}^d \to \{1, \dots, K\}$ and set $S_k = \{\ x \in X : f(x) = k\ \}$.

(allows assigning a cluster label also to new points, $x \neq X$: "out-of-sample extension")

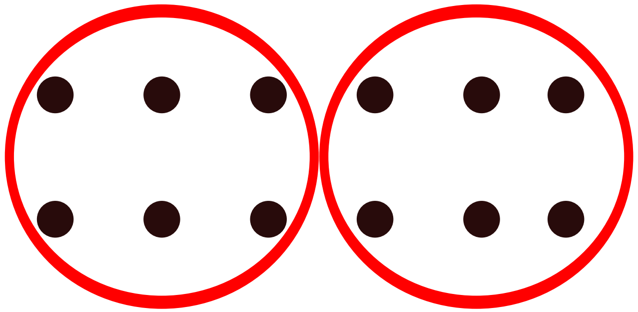
Clustering

Clustering is fundamentally problematic and subjective



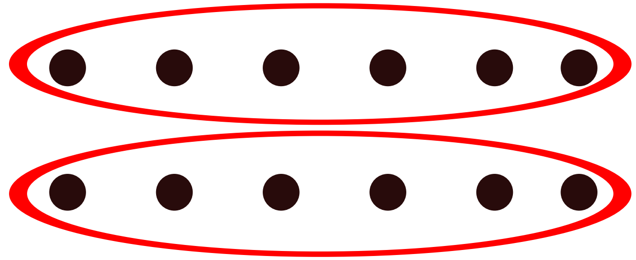
Clustering

Clustering is fundamentally problematic and subjective



Clustering

Clustering is fundamentally problematic and subjective



Clustering – Linkage-based

General framework to create a **hierarchical partitioning**

- initialize: each point x_i is it's own cluster, $S_i = \{i\}$
- repeat
  - take two most similar clusters and merge into a single new cluster
- until K clusters left

Open question: how to define similarity between clusters?

Clustering – Linkage-based

Given: similarity between individual points $d(x_i, x_j)$

Single linkage clustering

Smallest distance between any cluster elements

$$d(S, S') = \mathbf{min}_{i\in S, j\in \mathbb{S}'}\, d(x_i, x_j)$$

Average linkage clustering

Average distance between all cluster elements

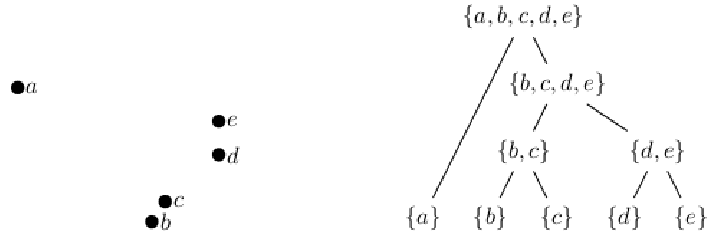
$$d(S, S') = \frac{1}{|S||S'|} \sum_{i\in S, j\in \mathbb{S}'} d(x_i, x_j)$$

Max linkage clustering

Largest distance between any cluster elements

$$d(S, S') = \mathbf{max}_{i\in S, j\in \mathbb{S}'}\, d(x_i, x_j)$$

Example: Single linkage clustering



Theorem
The edges of a single linkage clustering forms a minimal spanning tree.

Show Jupyter notebook

Clustering – centroid-based clustering

Let $c_1, \ldots, c_K \in \mathbb{R}^d$ be K **cluster centroids**. Then a distance-based clustering function, $c : \mathcal{X} \to \{1, \ldots, K\}$, is given by the assignment

$$f(x) = \underset{k=1,\ldots,K}{\mathbf{argmin}} \|x - c_i\| \qquad \text{(arbitrary tie break)}$$

(similar to K-means with training set $\{(c_1, 1), \ldots, (c_K, K)\}$)

Clustering – centroid-based clustering

K-means objective
Find $c_1, \ldots, c_K \in \mathbb{R}^d$ by minimizing the total Euclidean error

$$\sum_{i=1}^{m} \|x_i - c_{f(x_i)}\|^2$$

Clustering – centroid-based clustering

K-means objective

Find $c_1, \dots, c_K \in \mathbb{R}^d$ by minimizing the total Euclidean error

$$\sum_{i=1}^{m} \|x_i - c_{f(x_i)}\|^2$$

Lloyd's algorithm

- Initialize $c_1, \dots, c_K$ (random subset of X, or smarter)
- repeat
  - set $S_k = \{i : f(x_i) = k\}$ (current assignment)
  - $c_k = \frac{1}{|S_k|} \sum_{i \in S_k} x_i$ (mean of points in cluster)
- until no more changes to S_k

Demo: `http://shabal.in/visuals/kmeans/6.html`

Clustering – centroid-based clustering

Alternatives:

- k-mediods: like k-means, but centroids must be datapoints
  update step chooses mediod of cluster instead of mean
- k-medians: like k-means, but minimize $\sum_{i=1}^{m} \|x_i - c_{f(x_i)}\|$
  update step chooses median of each coordinate with each cluster

Clustering – graph-based clustering

For $x_1, \dots, x_m$ form a graph $G = (V, E)$ with vertex set $V = \{1, \dots, m\}$ and edge set E. Each **partitioning of the graph defines a clustering** of the original dataset.

Choice of edge set

ϵ-nearest neighbor graph

$$E = \{(i, j) \subset V \times V : \|x_i - x_j\| < \epsilon\}$$

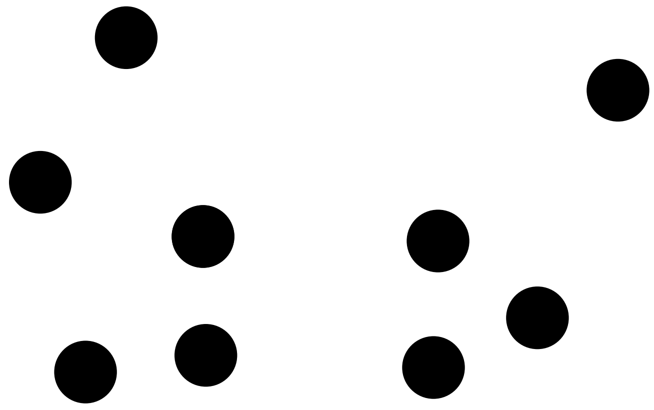
k-nearest neighbor graph

$$E = \{(i, j) \subset V \times V : x_i \text{ is a } k\text{-nearest neighbor of } x_j \}$$

Weighted graph

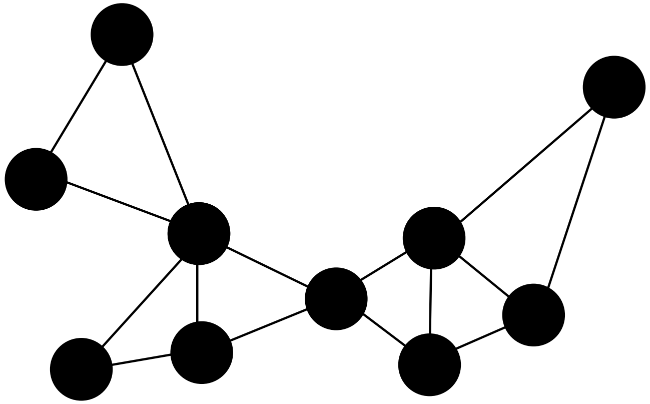
Fully connected, but define edge weights $w_{ij} = \exp(-\lambda \|x_i - x_j\|^2)$.

Example: Graph-based Clustering



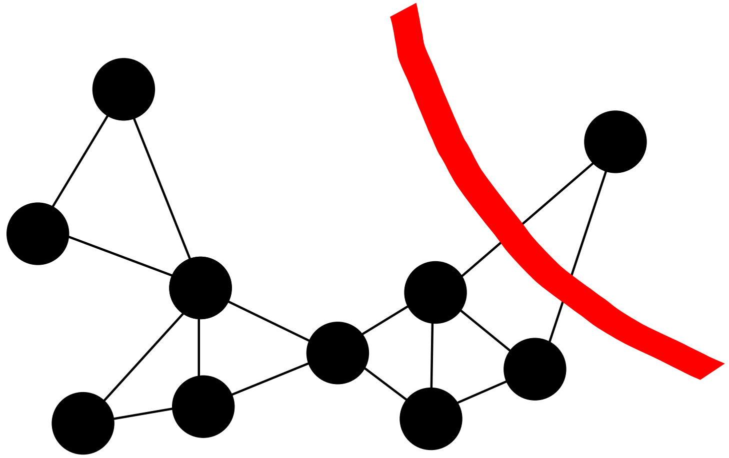
Data set

Example: Graph-based Clustering



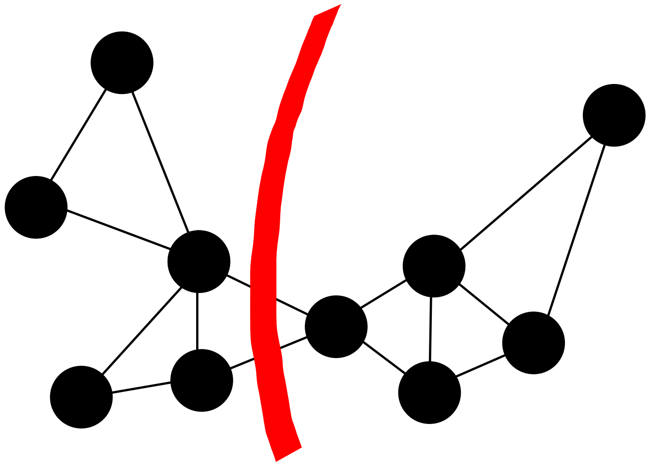
Neighborhood Graph

Example: Graph-based Clustering



Min Cut: biased towards small clusters

Example: Graph-based Clustering



Normalized Cut: balanced weight of cut edges and volume of clusters

Spectral Clustering

Approximate solution to *Normalized Cut*

Spectral Clustering

- Input: weight matrix $W \in \mathbb{R}^{m\times m}$
- compute graph Laplacian $L = W - D$,
  for $D = \text{diag}(d_1, \ldots, d_m)$ with $d_i = \sum_j w_{ij}$.
- let $v \in \mathbb{R}^m$ be the eigenvector of L corresponding to the second smallest eigenvalue (the smallest is 0, since L is singular)
- assign x_i to cluster 1 if $v_i \geq 0$ and to cluster 2 otherwise.

To obtain more than 2 clusters apply recursively, each time splitting the largest remaining cluster.

Clustering Axioms [Kleinberg, "An Impossibility Theorem for Clustering", NIPS 2002]

Scale-Invariance
For any distance d and any $\alpha > 0$, $f(d) = f(\alpha \cdot d)$

Richness
Range(f) is the set of all partitions of $\{1, \ldots, m\}$

Consistency
Let d and d' be two distance functions. If $f(d) = \Gamma$, and d' is a Γ-transform of d, then $f(d') = \Gamma$.

Definition: d' is a Γ-transform of d, iff for any i, j in the same cluster $d'(i,j) \leq d(i,j)$ and for i, j in different clusters, $d'(i,j) \geq d(i,j)$.

Clustering Axioms [Kleinberg, "An Impossibility Theorem for Clustering", NIPS 2002]

Scale-Invariance
For any distance d and any $\alpha > 0$, $f(d) = f(\alpha \cdot d)$

Richness
Range(f) is the set of all partitions of $\{1, \ldots, m\}$

Consistency
Let d and d' be two distance functions. If $f(d) = \Gamma$, and d' is a Γ-transform of d, then $f(d') = \Gamma$.

Definition: d' is a Γ-transform of d, iff for any i, j in the same cluster $d'(i,j) \leq d(i,j)$ and for i, j in different clusters, $d'(i,j) \geq d(i,j)$.

Theorem: "Impossibility of Clustering". For each $m \geq 2$, there is no clustering function f that satisfies all three axioms at the same time.

Clustering Axioms [Kleinberg, "An Impossibility Theorem for Clustering", NIPS 2002]

Scale-Invariance
For any distance d and any $\alpha > 0$, $f(d) = f(\alpha \cdot d)$

Richness
Range(f) is the set of all partitions of $\{1, \ldots, m\}$

Consistency
Let d and d' be two distance functions. If $f(d) = \Gamma$, and d' is a Γ-transform of d, then $f(d') = \Gamma$.

Definition: d' is a Γ-transform of d, iff for any i, j in the same cluster $d'(i,j) \leq d(i,j)$ and for i, j in different clusters, $d'(i,j) \geq d(i,j)$.

Theorem: "Impossibility of Clustering". For each $m \geq 2$, there is no clustering function f that satisfies all three axioms at the same time.

(but not all hope lost: "Consistency" is debatable...)

Unsupervised Learning
Dimensionality Reduction

Dimensionality Reduction

Given: data

$$X = \{x^1, \dots, x^N\} \subset \mathbb{R}^d$$

Dimensionality Reduction – Transductive

Task: Find a lower-dimensional representation

$$Y = \{y^1, \dots, y^N\} \subset \mathbb{R}^n$$

with $n \ll d$, such that Y "represents X well"

Dimensionality Reduction – Inductive

Task: find a function $\phi : \mathbb{R}^d \to \mathbb{R}^n$ and set $y_i = \phi(x_i)$

(allows computing $\phi(x)$ for $x \neq X$: "out-of-sample extension")

Linear Dimensionality Reduction

Choice 1: $\phi : \mathbb{R}^d \to \mathbb{R}^n$ is **linear** or **affine**.

Choice 2: "Y represents X well" means:

There's a $\psi : \mathbb{R}^n \to \mathbb{R}^d$ such that $\sum_{i=1}^{N} \|x_i - \psi(y_i)\|^2$ is small.

Linear Dimensionality Reduction

Choice 1: $\phi : \mathbb{R}^d \to \mathbb{R}^n$ is **linear** or **affine**.

Choice 2: "Y represents X well" means:

There's a $\psi : \mathbb{R}^n \to \mathbb{R}^d$ such that $\sum_{i=1}^{N} \|x_i - \psi(y_i)\|^2$ is small.

Principal Component Analysis

Given $X = \{x^1, \dots, x^N\} \subset \mathbb{R}^d$, find function $\phi(x) = Wx$ and $\psi(y) = Uy$ by solving

$$\min_{\substack{U \in \mathbb{R}^{n \times d} \\ W \in \mathbb{R}^{d \times n}}} \sum_{i=1}^{N} \|x_i - UWx_i\|^2$$

Principal Component Analysis (PCA)

$$U, W = \operatorname*{argmin}_{U \in \mathbb{R}^{n \times d}, W \in \mathbb{R}^{d \times n}} \sum_{i=1}^{N} \|x_i - UWx_i\|^2 \qquad \text{(PCA)}$$

Lemma

If U, W are minimizers of the above PCA problem, then the column of U are orthogonal, and $W = U^\top$.

Principal Component Analysis (PCA)

$$U, W = \underset{U \in \mathbb{R}^{n \times d}, W \in \mathbb{R}^{d \times n}}{\textbf{argmin}} \sum_{i=1}^{N} \|x_i - UWx_i\|^2 \qquad \text{(PCA)}$$

Lemma
If U, W are minimizers of the above PCA problem, then the column of U are orthogonal, and $W = U^\top$.

Theorem
Let $C = \sum_{i=1}^N x_i x_i^\top$ and let $u_1, \ldots, u_n$ be n eigenvectors of A that correspond to the largest n eigenvalues of C. Then $U = \left(u_1 | u_2 | \cdots | u_n\right)$ and $W = U^\top$ are minimizers of the PCA problem.

- C has orthogonal eigenvectors, since it is symmetric positive definite.
- U can also be obtained by singular value decomposition, $X = USV$.

Principal Component Analysis (PCA)

$$U, W = \underset{U \in \mathbb{R}^{n \times d}, W \in \mathbb{R}^{d \times n}}{\textbf{argmin}} \sum_{i=1}^{N} \|x_i - UWx_i\|^2 \qquad \text{(PCA)}$$

Lemma
If U, W are minimizers of the above PCA problem, then the column of U are orthogonal, and $W = U^\top$.

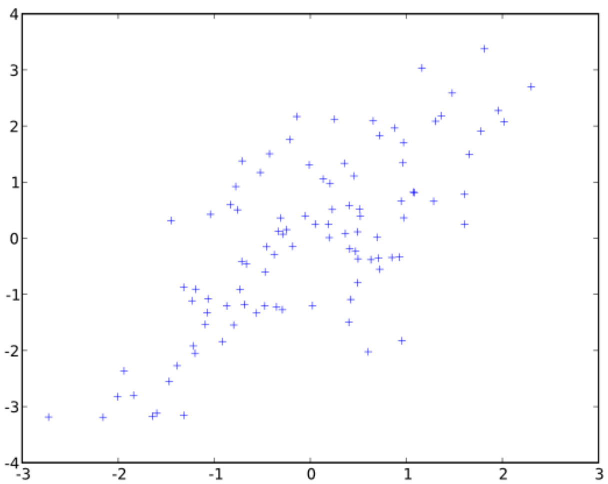
Theorem
Let $C = \sum_{i=1}^N x_i x_i^\top$ and let $u_1, \ldots, u_n$ be n eigenvectors of A that correspond to the largest n eigenvalues of C. Then $U = \left(u_1 | u_2 | \cdots | u_n\right)$ and $W = U^\top$ are minimizers of the PCA problem.

- C has orthogonal eigenvectors, since it is symmetric positive definite.
- U can also be obtained by singular value decomposition, $X = USV$.

Typically data is zero-meaned before: $x'_i = x_i - \frac{1}{N}\sum_{j=1}^N x_j$ and thus C is *Covariance matrix*. (Affine PCA)

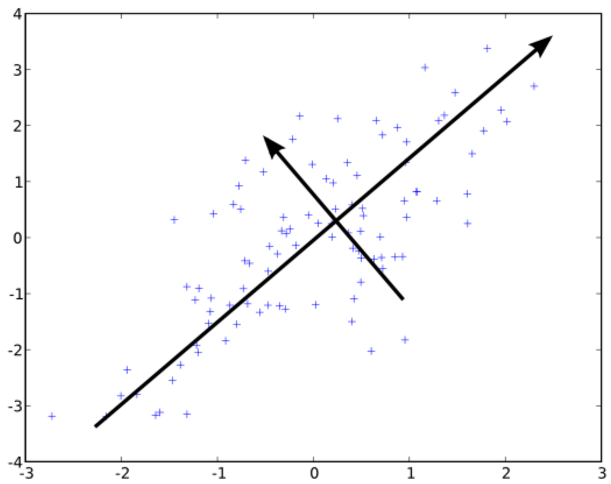
Principal Component Analysis – Visualization

Data



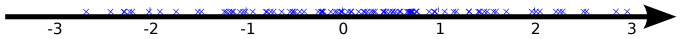
Principal Component Analysis – Visualization

PCA



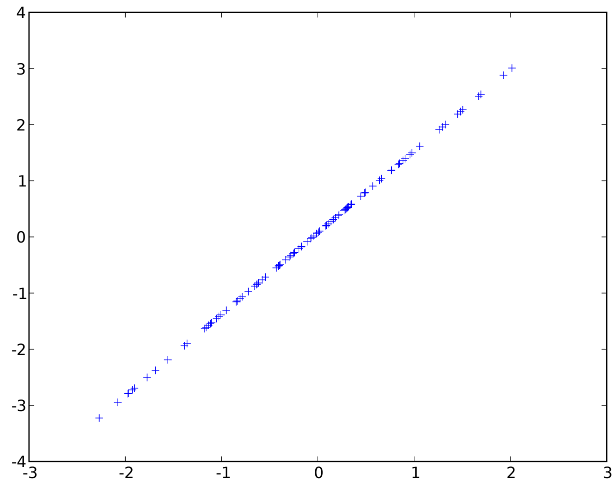
Principal Component Analysis – Visualization

Projected onto first component



Principal Component Analysis – Visualization

Reconstructed from first component



Principal Component Analysis – Alternative Views

There's (at least) one more way to interpret the PCA procedure:

The following to goals are equivalent:

- find subspace such that projecting to it orthogonally results in the **smallest reconstruction error**
- find subspace such that projecting to it orthogonally results **preserves most of the data variance**

Principal Component Analysis – as Variance maximization projection

Goal:

find direction $u_1 \in \mathbb{R}^d$ where the data has largest variance
Projection: $u_1^\top x_i$. **Variance in projected space:**

$$\frac{1}{N}\sum_{i=1}^{N}(u_1^\top x_i - u_1^\top \bar{x}) = u^\top S u$$

with $S = \sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T$ (Covariance matrix).

Principal Component Analysis – as Variance maximization projection

Goal:

find direction $u_1 \in \mathbb{R}^d$ where the data has largest variance
Projection: $u_1^\top x_i$. **Variance in projected space:**

$$\frac{1}{N}\sum_{i=1}^{N}(u_1^\top x_i - u_1^\top \bar{x}) = u^\top S u$$

with $S = \sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T$ (Covariance matrix).
Maximize with constraint $u^\top u = 1$:

$$u_1 = \underset{u}{\mathbf{argmax}}\, u^\top S u + \lambda(1 - u^\top u)$$

Derivative w.r.t. u: $Su = \lambda u$ (Eigenvalue problem)
Variance is given by: $u^\top S u = \lambda$ (use $u^\top u = 1$)

The Eigenvector corresponding to the largest Eigenvalue is the direction of largest projected variance.
All PCA components are given by the Eigenvectors with decreasing Eigenvalues.

Principal Component Analysis – Applications

Data Visualization

If the original data is high-dimensional, use PCA with $n = 2$ or $n = 3$ to obtain low-dimensional representation that can be visualized.

Data Compression

If the original data is high-dimensional, use PCA to obtain a lower-dimensional representation that requires less RAM/storage.

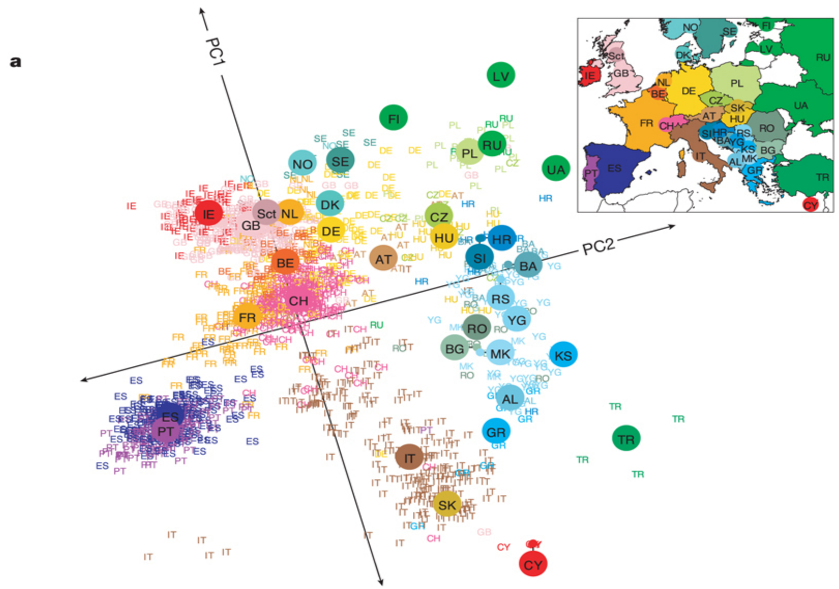
n typically chosen such that 95% or 99% of variance are preserved.

Data Denoising

If the original data is noisy, apply PCA and reconstruction to obtain a less noisy representation.

n depends on noise level if known, otherwise as for compression.

Genes mirror geography in Europe [Novembre et al, Nature 2008]



Canonical Correlation Analysis (CCA) [Hotelling, 1936]

Given: paired data

$$X_1 = \{x_1^1, \ldots, x_1^N\} \subset \mathbb{R}^d \qquad X_2 = \{x_2^1, \ldots, x_2^N\} \subset \mathbb{R}^{d'}$$

for example (after some preprocessing):

- *DNA expression* and *gene expression*
- *images* and *text captions*.

Canonical Correlation Analysis (CCA)

Find projections $\phi_1(x_1) = U_1 x_1$ and $\phi_2(x_2) = U_2 x_2$ with $U_1 \in \mathbb{R}^{d \times n}$ and $U_2 \in \mathbb{R}^{d' \times n}$ such that after projection X_1 and X_2 are **maximally correlated**.

Canonical Correlation Analysis (CCA)

One dimension: find directions $u_1 \in \mathbb{R}^d$, $u_2 \in \mathbb{R}^{d'}$, such that

$$\max_{u_1 \in R^d, u_2 \in \mathbb{R}^{d'}} \text{corr}(u_1^\top X_1, u_2^\top X_2).$$

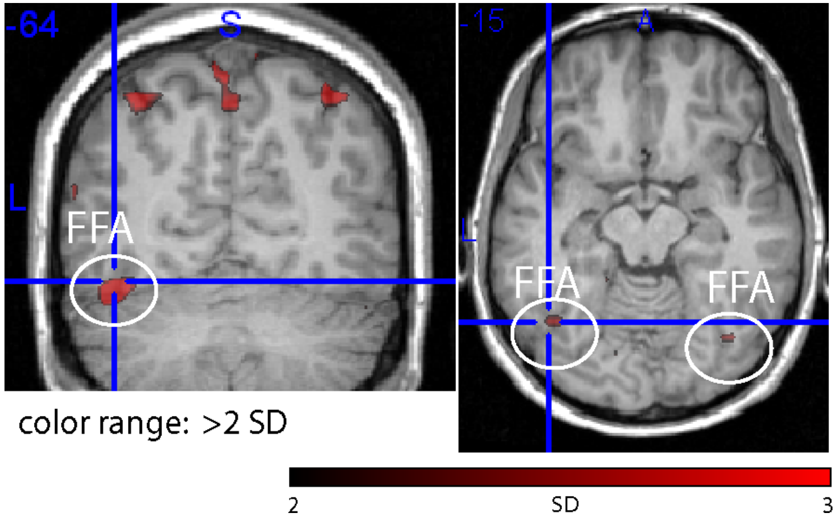
With $C_{11} = \text{cov}(X_1, X_1)$, $C_{22} = \text{cov}(X_2, X_2)$ and $C_{12} = \text{cov}(X_1, X_2)$,

$$\max_{u_1 \in R^d, u_2 \in \mathbb{R}^{d'}} \frac{u_1^\top C_{12} u_2}{\sqrt{u_1^\top C_{11} u_1} \sqrt{u_2^\top C_{22} u_2}}$$

Find u_1, u_2 by solving **generalized eigenvalue problem** for maximal λ:

$$\begin{pmatrix} \mathbf{0} & C_{12} \\ C_{12}^\top & \mathbf{0} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \lambda \begin{pmatrix} C_{11} & \mathbf{0} \\ \mathbf{0} & C_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

Example: Canonical Correlation Analysis for fMRI Data



data 1: video sequence data 2: fMRI signal while watching

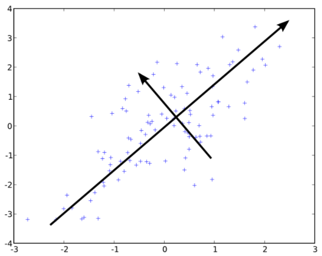
Kernel Principle Component Analysis (Kernel-PCA)

Reminder: given samples $x_i \in \mathbb{R}^d$, PCA finds the directions of maximal covariance. Assume $\sum_i x_i = \mathbf{0}$ (e.g. by first subtracting the mean).

- The PCA directions $u_1, \dots, u_n$ are the *eigenvectors* of the covariance matrix

  $$C = \frac{1}{m} \sum_{i=1}^{m} x_i x_i^\top$$

  sorted by their eigenvalues.



- We can express x_i in PCA-space by $P(x_i) = \sum_{j=1}^{n} \langle x_i, u_j \rangle u_j$.
- Lower-dim. coordinate mapping: $x_i \mapsto \begin{pmatrix} \langle x_i, u_1 \rangle \\ \langle x_i, u_2 \rangle \\ \dots \\ \langle x_i, u_n \rangle \end{pmatrix} \in \mathbb{R}^n$

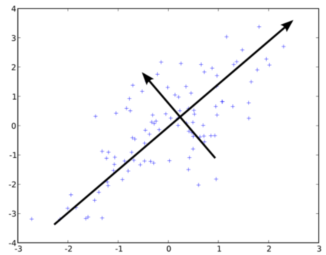
Kernel-PCA

Given samples $x_i \in \mathcal{X}$, kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with an implicit feature map $\phi : \mathcal{X} \to \mathcal{H}$. **Do PCA in the (implicit) feature space $\mathcal{H}$.**

- The kernel-PCA directions $u_1, \dots, u_n$ are the eigenvectors of the covariance operator

  $$C = \frac{1}{N} \sum_{i=1}^{N} \phi(x_i)\phi(x_i)^\top$$

  sorted by their eigenvalue.

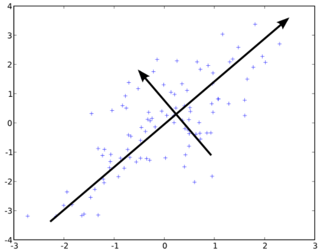


- Lower-dim. coordinate mapping: $x_i \mapsto \begin{pmatrix} \langle \phi(x_i), u_1 \rangle \\ \langle \phi(x_i), u_2 \rangle \\ \dots \\ \langle \phi(x_i), u_n \rangle \end{pmatrix} \in \mathbb{R}^n$

Kernel-PCA

Given samples $x_i \in \mathcal{X}$, kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with an implicit feature map $\phi : \mathcal{X} \to \mathcal{H}$. **Do PCA in the (implicit) feature space $\mathcal{H}$.**

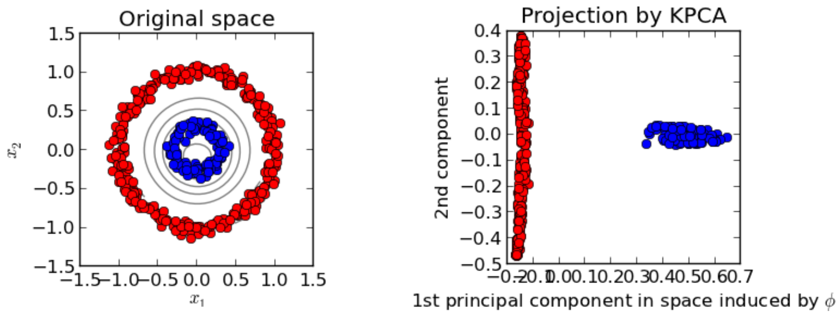
- Equivalently, we can use the eigenvectors u'_j and eigenvalues λ_j of $K \in \mathbb{R}^{N \times N}$, with $K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle = k(x_i, x_j)$



- Coordinate mapping: $x_i \mapsto \left(\sqrt{\lambda_1} u'^i_1, \dots, \sqrt{\lambda_n} u'^i_n \right)$.

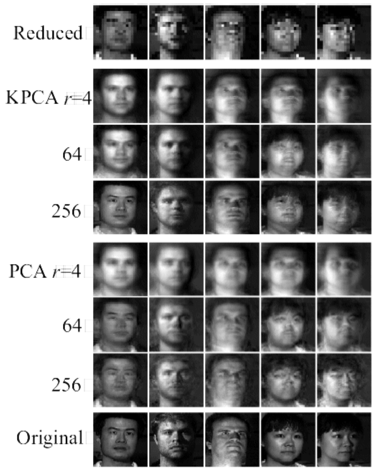
Kernel-PCA

Example: Non-linear dimensionality reduction with Kernel PCA



Application – Image Superresolution

- Collect high-res face images
- Use KernelPCA with Gaussian kernel to learn non-linear projections
- For new low-res image:
  - scale to target high resolution
  - project to closest point in face subspace



reconstruction in r dimensions

[Kim, Jung, Kim, "Face recognition using kernel principal component analysis", Signal Processing Letters, 2002.]

Multidimensional Scaling (MDS)

Given: data $X = \{x^1, \dots, x^m\} \subset \mathbb{R}^d$

Task: find embedding $y^1, \dots, y^m \subset \mathbb{R}^n$ that **preserves pairwise distances** $\Delta_{ij} = \|x^i - x^j\|$.

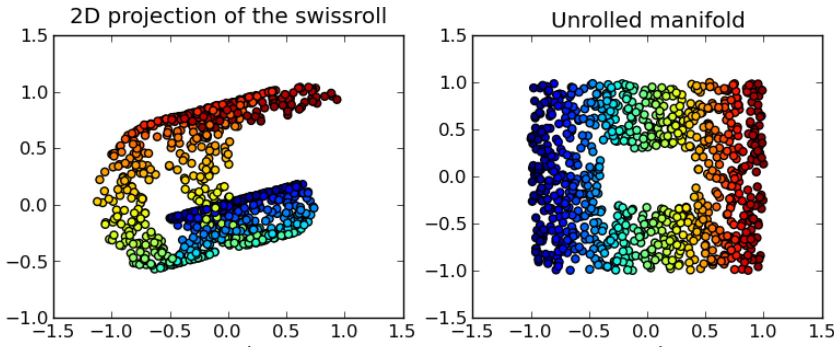
Solve, e.g., by gradient descent on

$$\sum_{i,j} \left(\|y^i - y^j\|^2 - \Delta_{ij}^2 \right)^2$$

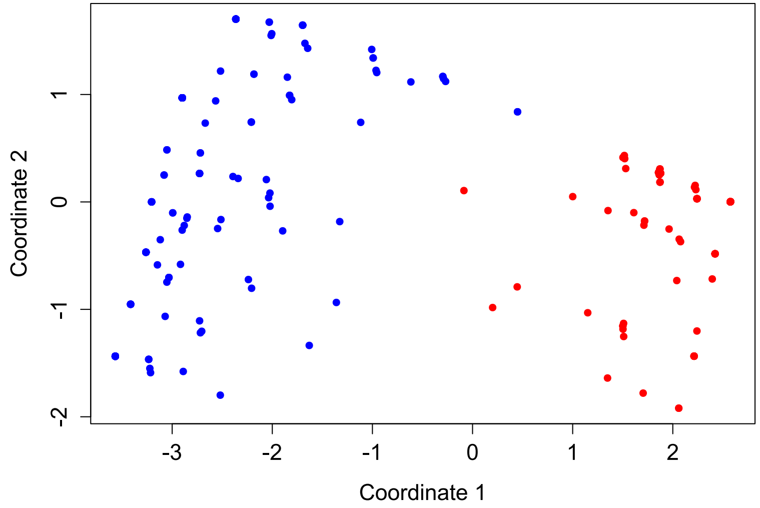
Multiple extensions:

- non-linear embedding
- take into account geodesic distances (e.g. IsoMap)
- arbitrary distances instead of Euclidean

Multidimensional Scaling (MDS)



Multidimensional Scaling (MDS)



2D embedding of *US Senate Voting behavior*

Other methods for dimensionality reduction and manifold learning