Statistical Machine Learning

Christoph Lampert

IST AUSTRIA

Institute of Science and Technology

Spring Semester 2015/2016 // Lecture 12

Unsupervised Learning
Dimensionality Reduction

Dimensionality Reduction

Given: data

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

Dimensionality Reduction – Transductive

Task: Find a lower-dimensional representation

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

with $m \ll d$, 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")

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 $\sum_{i=1}^{m} \|x_i - \psi(y_i)\|^2$ is small.

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 $\sum_{i=1}^{m} \|x_i - \psi(y_i)\|^2$ is small.

Principal Component Analysis

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

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

Principal Component Analysis (PCA)

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

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)

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

Lemma

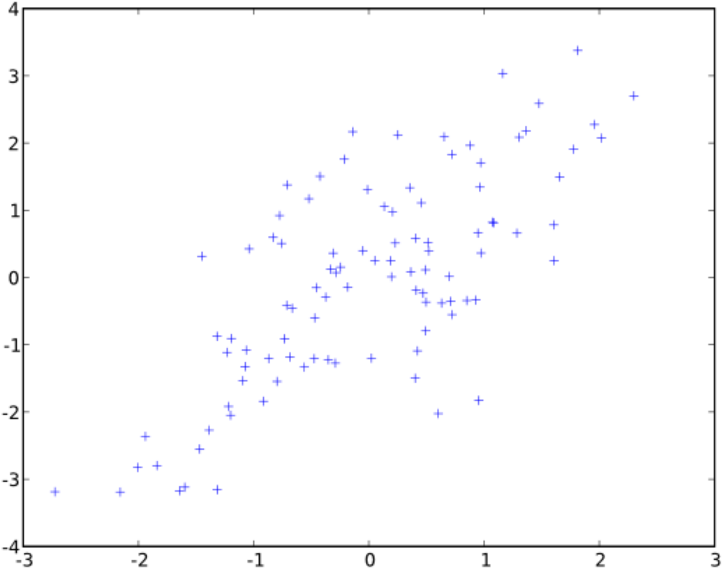
If U, W are minimizers of the above PCA problem, then the column of U are orthogonal, and $W = U^\top$.

Theorem

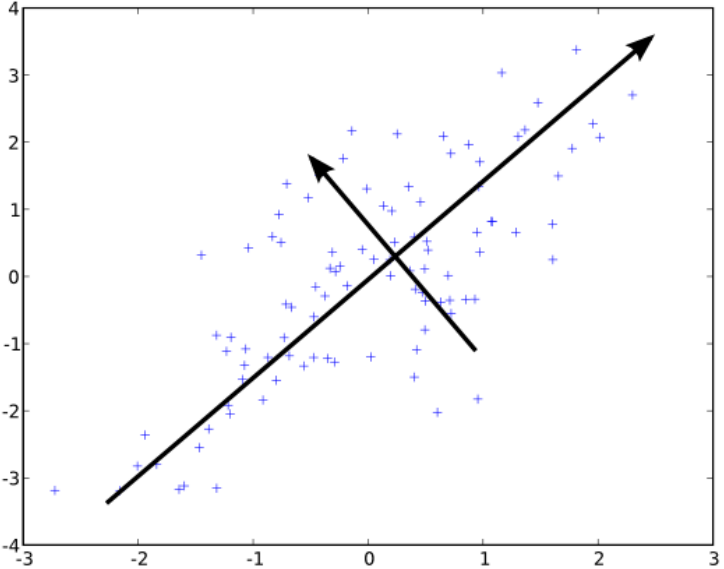
Let $A = \sum_{i=1}^{m} x_i x_i^\top$ and let $u_1, \ldots, u_n$ be n eigenvectors of A that correspond to the largest n eigenvalues of A. Then $U = \left(u_1 | u_2 | \cdots | u_n\right)$ and $W = U^\top$ are minimizers of the PCA problem.

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

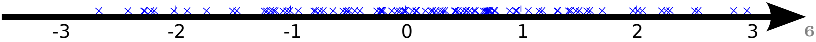
Principal Component Analysis – Visualization



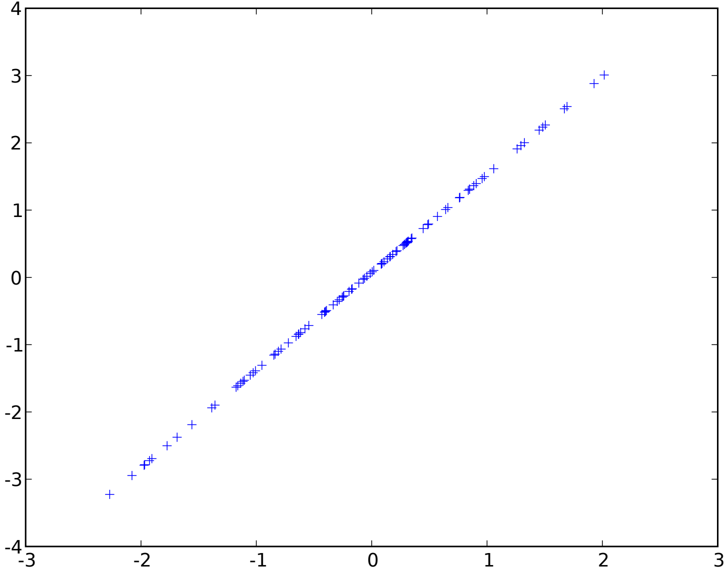
Principal Component Analysis – Visualization



Principal Component Analysis – Visualization



Principal Component Analysis – Visualization



Principal Component Analysis – Affine

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

$$U, W = \underset{U\in\mathbb{R}^{n\times d}, W\in\mathbb{R}^{d\times n}}{\mathbf{argmin}} \sum_{i=1}^{m} \|x_i - U(Wx_i + w) - u\|^2 \qquad \text{(AffinePCA)}$$

Theorem

Let $\mu = \frac{1}{m}\sum_{i=1}^{m} x_i$ the mean and $C = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)(x_i - \mu)^\top$ the covariance matrix of X. Let $u_1, \dots, u_n$ be n eigenvectors of C that correspond to the largest n eigenvalues. Then $U = \Big(u_1|u_2|\cdots|u_n\Big)$, $W = U^\top$, $w = W\mu$ and $u = \mu$ are minimizers of the affine PCA problem.

Simpler to remember: $\phi(x) = W(x - \mu), \quad \psi(y) = Uy + \mu$

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 – 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

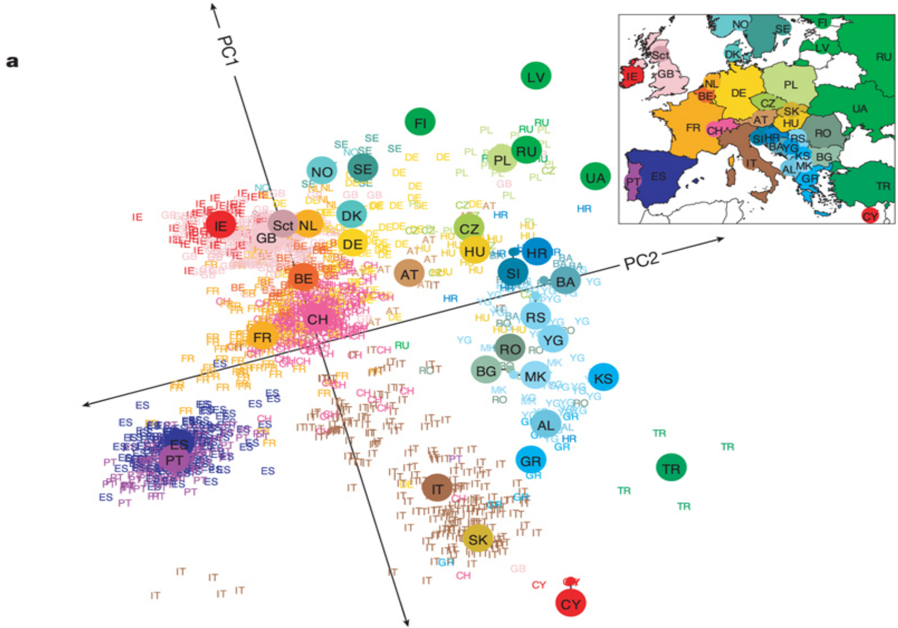
If the original data is high-dimensional, use PCA to obtain a lower-dimensional representation that requires less RAM/storage.

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.


a
PC1
PC2
LV
FI
RU
PL
UA
NO
SE
DK
IE
GB
Sct
NL
DE
CZ
HU
HR
SI
BA
AT
BE
CH
FR
RS
YG
RO
BG
MK
KS
AL
GR
TR
ES
PT
IT
SK
CY

Given: paired data

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

for example (after some preprocessing):

- *DNA expression* and *gene expression* (Monday's colloquium)
- *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 $U_1 \in \mathbb{R}^{d \times m}$ and $U_2 \in \mathbb{R}d' \times m$ 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

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

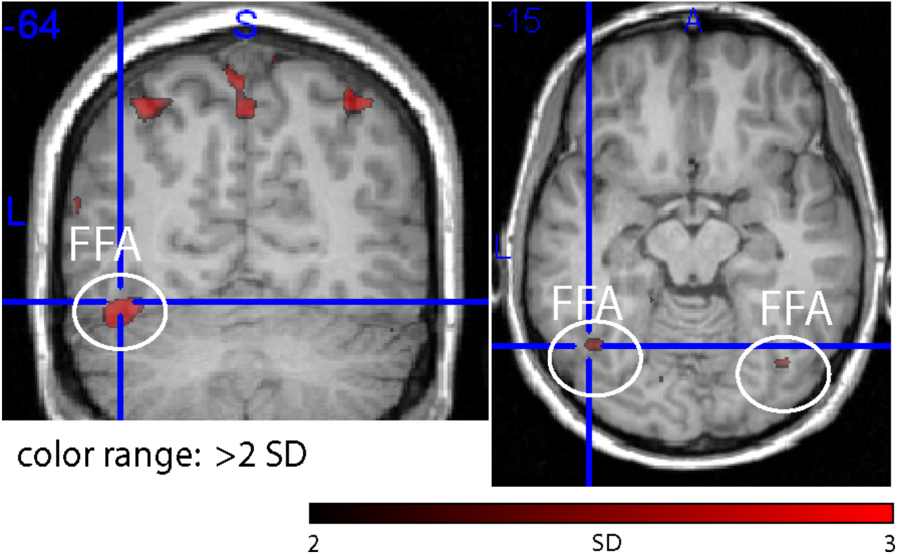
With $C_{11} = \text{cov}(X_1, X_1)$, $C_{22} = \text{cov}(X_2, X_2)$ and $C_{12} = \text{cov}(X_1, X_2)$,

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

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

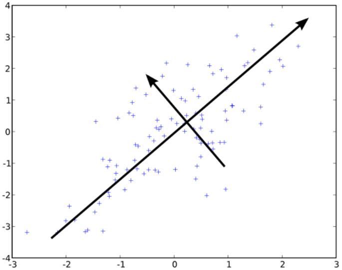
Kernel Principle Component Analysis (Kernel-PCA)

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: $x_i \mapsto \begin{pmatrix}\langle x_i, u_1\rangle \\ \langle x_i, u_2\rangle \\ \dots \\ \langle x_i, u_m\rangle\end{pmatrix} \in \mathbb{R}^n$

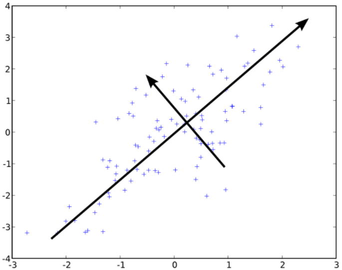
Kernel-PCA

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 $u_1, \ldots, u_n$ are the eigenvectors of the covariance operator

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

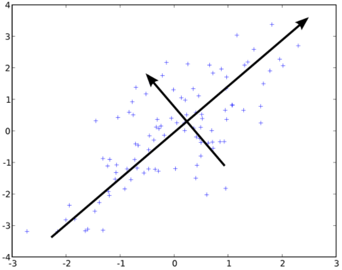
sorted by their eigenvalue.



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

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 $\mathcal{H}$.**

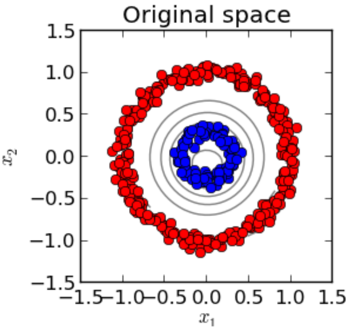
- Equivalently, we can use the eigenvectors u'_j and eigenvalues λ_j of $K \in \mathbb{R}^{m \times m}$, with $K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle = k(x_i, x_j)$

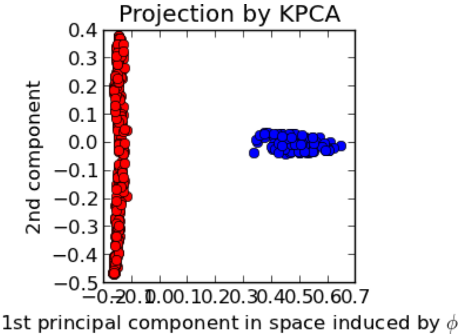


- Coordinate mapping: $x_i \mapsto (\ \sqrt{\lambda_1} u_1'^i, \ldots, \sqrt{\lambda_K} u_n'^i\ )$.

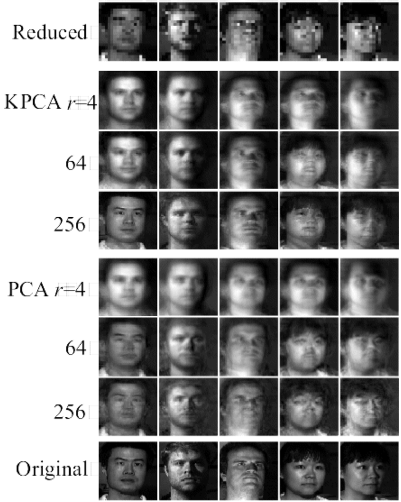
Kernel-PCA

Example: Canonical Correlation Analysis for fMRI Data





- 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.]

Recently, **random matrices** have been used for dimensionality reduction:

- Let $W \in \mathbb{R}^{d \times n}$ be a matrix with *random entries (i.i.d. Gaussian)*

Then one can show that $\phi : \mathbb{R}^d \to \mathbb{R}^n$ with $\phi(x) = Wx$ **does not distort Euclidean distances too much**.

Theorem

For fixed $x \in \mathbb{R}^d$ let $W \in \mathbb{R}^{n \times d}$ be a random matrix as above. Then, for every $\epsilon \in (0, 3)$,

$$\mathbb{P}\left[\left| \frac{\frac{1}{n}\|Wx\|^2}{\|x\|^2} - 1 \right| > \epsilon \right] \leq 2e^{-\epsilon^2 n/6}$$

Note: The dimension of the original data does not show up in the bound!

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** $\Delta_{ij} = \|x^i - x^j\|$.

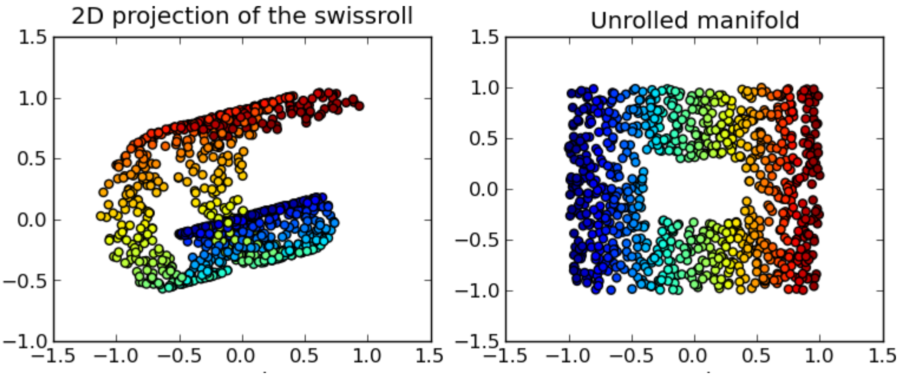
Solve, e.g., by gradient descent on

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

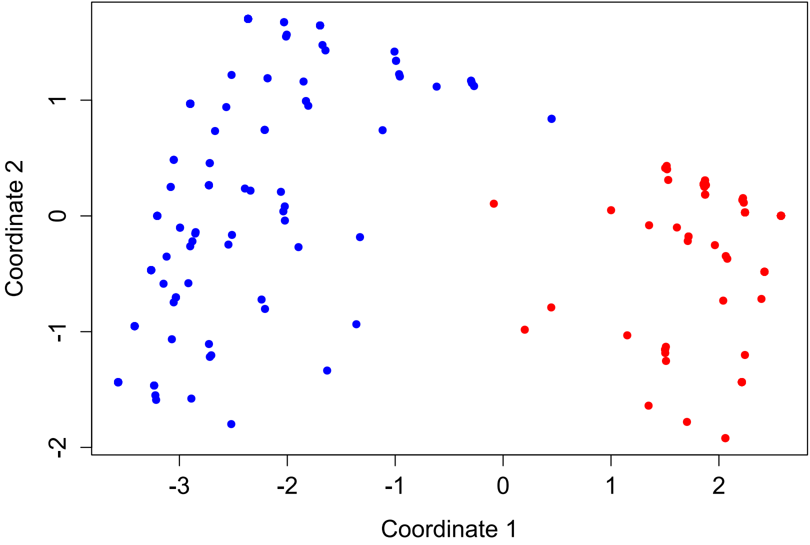
Multiple extensions:

- 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*

Unsupervised Learning
Clustering

Given: data

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

Clustering – Transductive

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

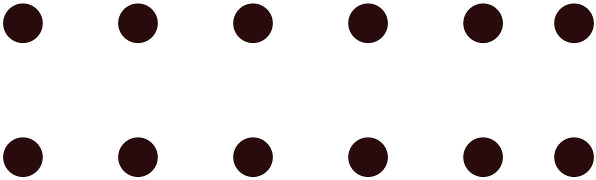
Idea: elements within a cluster are similar to each other, elements in different clusters are dissimilar

Clustering – Inductive

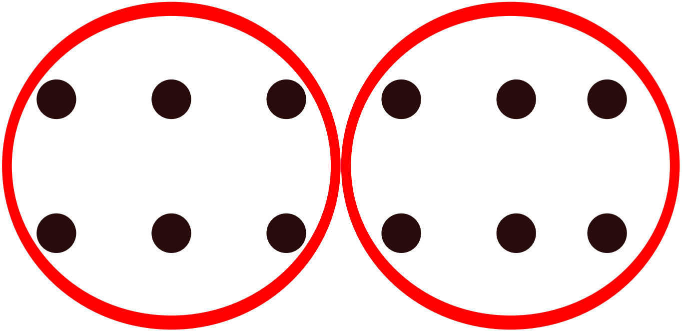
Task: define a partitioning function $f : \mathbb{R}^d \to \{1, \ldots, K\}$ and set $S_k = \{\ x \in X : f(x) = k\ \}$.

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

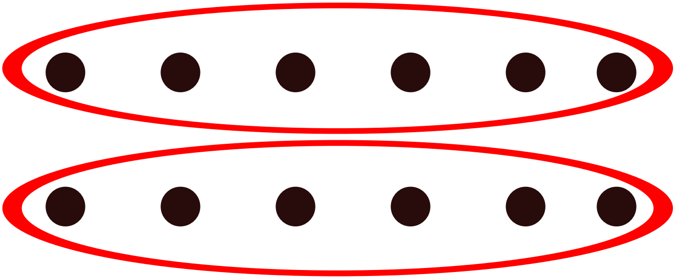
Clustering is fundamentally problematic and subjective



Clustering is fundamentally problematic and subjective



Clustering is fundamentally problematic and subjective



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?

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

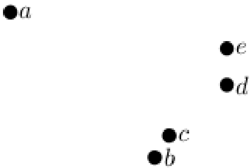
$$d(S, S') = \frac{1}{|S||S'|} \sum_{i \in S, j \in \mathbb{S}'} d(x_i, x_j)$$

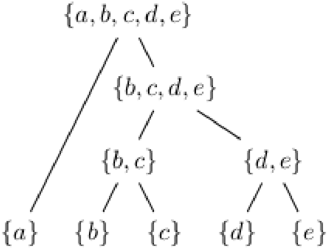
Max linkage clustering

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.

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

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

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

K-means objective

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

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

K-means objective

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

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

Lloyd's algorithm

- Initialize $c_1, \ldots, c_K$ (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`

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

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

Choice of edge set

ϵ-nearest neighbor graph

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

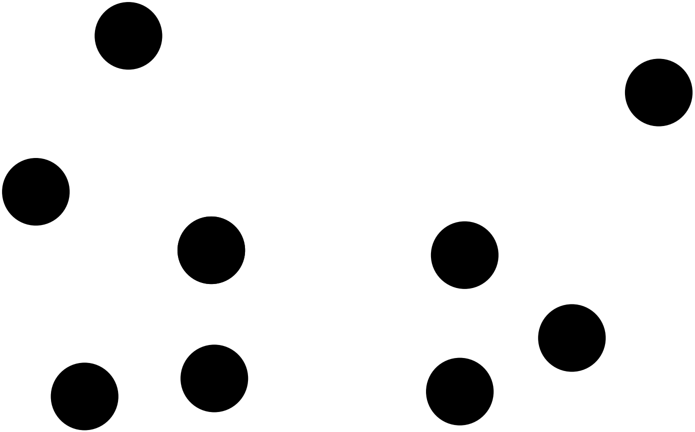
k-nearest neighbor graph

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

Weighted graph

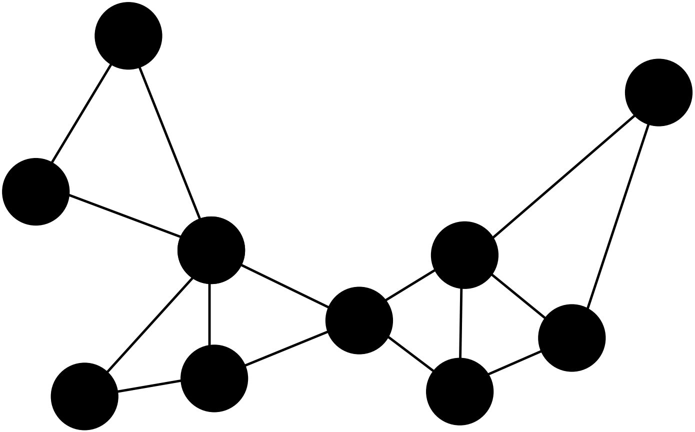
Fully connected, but define edge weights $w_{ij} = \exp(-\lambda \|x_i - x_j\|^2)$.

Example: Graph-based Clustering

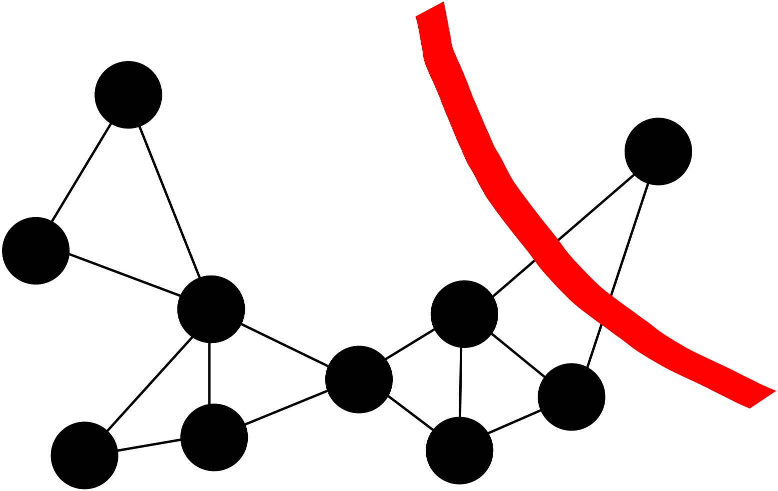


Data set

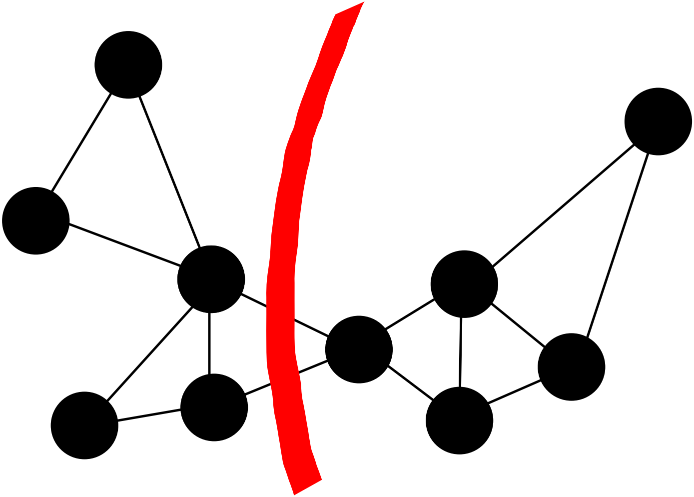
Example: Graph-based Clustering



Neighborhood Graph



Min Cut: biased towards small clusters



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

Approximate solution to *Normalized Cut*

Spectral Clustering

- Input: weight matrix $W \in \mathbb{R}^{m \times m}$
- compute graph Laplacian $L = W - D$,
  for $D = \text{diag}(d_1, \dots, d_m)$ 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 $\{1, \dots, m\}$

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 $\{1, \dots, m\}$

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...)

Final project

Part 1

- Go to `https://kaggle.com/join/ist_sml2016/` and participate in the challenge: *"Final project for Statistical Machine Learning Course 2016 at IST Austria"*

#	Δ3d	Team Name	Score	Entries	Last Submission UTC (Best – Last Submission)
1	—	AlexanderKolesnikov	0.97367	6	Tue, 01 Jul 2014 08:11:23 (-12.2h)
2	—	Jan Humplik	0.97263	6	Tue, 01 Jul 2014 13:56:24 (-2.7h)
3	new	Michal Rolínek	0.91640	2	Mon, 30 Jun 2014 10:45:30 (-1.3h)
4	↓1	Georg Nebehay	0.86330	9	Tue, 01 Jul 2014 15:07:58
5	new	michael.meidlinger	0.75163	3	Tue, 01 Jul 2014 12:05:58
6	↓2	**Christoph Lampert**	**0.48705**	**1**	**Wed, 18 Jun 2014 15:52:14**

passing criterion: beat the baselines (linear SVM and LogReg)

Part 2

- send Alex a short (one to two pages) report that explains what exactly you did to achieve these results, including data preprocessing, classifier, software used, model selection, etc.

Deadline: Thursday, 5th May midnight MEST