MAC 0460 / 5832
Introduction to Machine Learning

19 – Unsupervised Learning

• clustering • Self-organizing maps (SOM) •
• PCA • Auto-encoder •

IME/USP (23/06/2021)

Types of machine learning

So far we have assumed an input space $\mathcal{X}$ and an output space $\mathcal{Y}$ and that there is some relation between $\mathcal{X}$ and $\mathcal{Y}$

Two best known ML types: Supervised $\times$ unsupervised

Supervised: observations are of the form $(\mathbf{x}, y)$

$\Rightarrow$ Data labeling is an expensive task

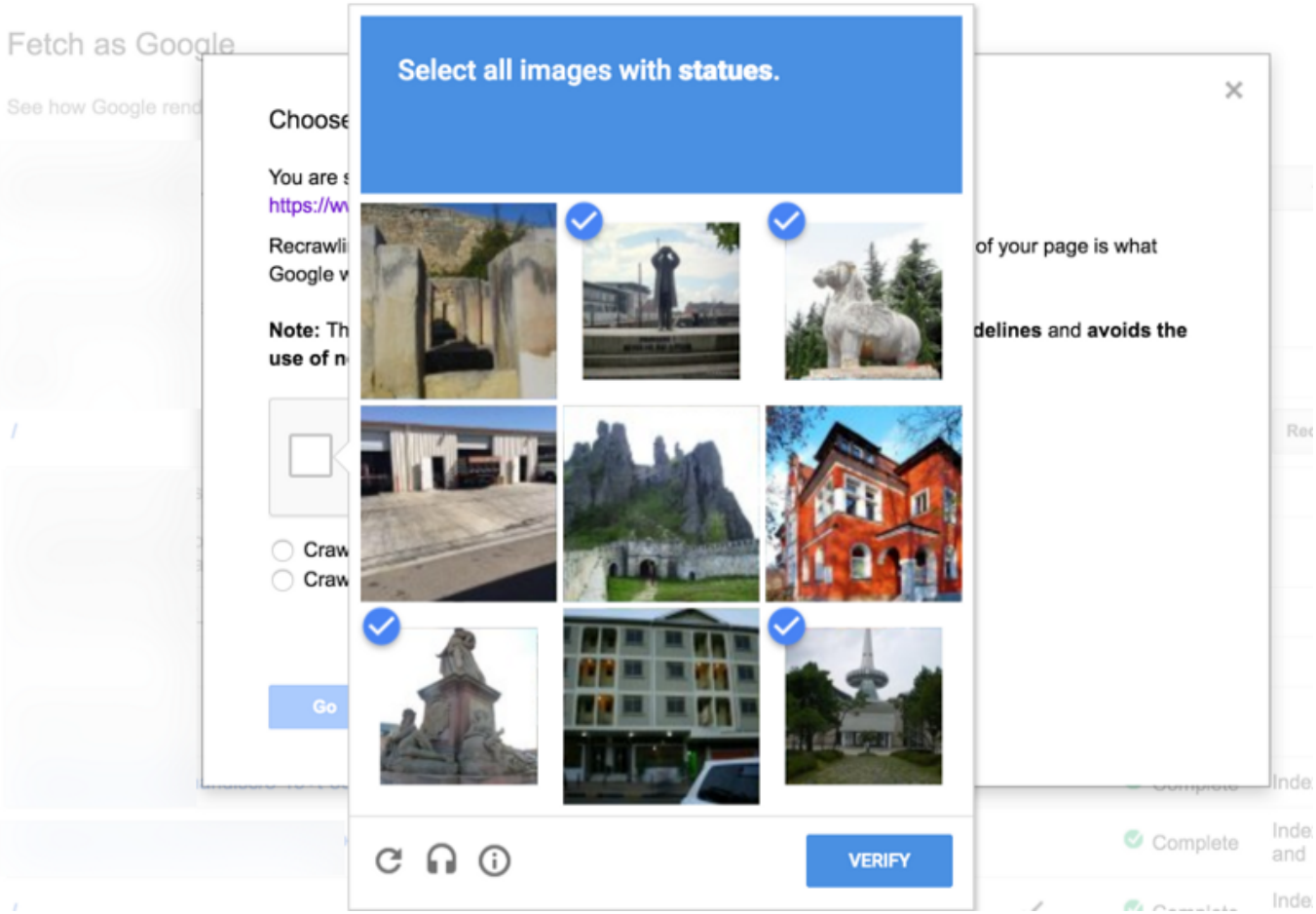
Unsupervised: observations are of the form $\mathbf{x}$

$\Rightarrow$ No target to guide us

$\Rightarrow$ (Data Mining)

Strategies for data labeling

Captcha



Strategies for data labeling

Mechanical Turks

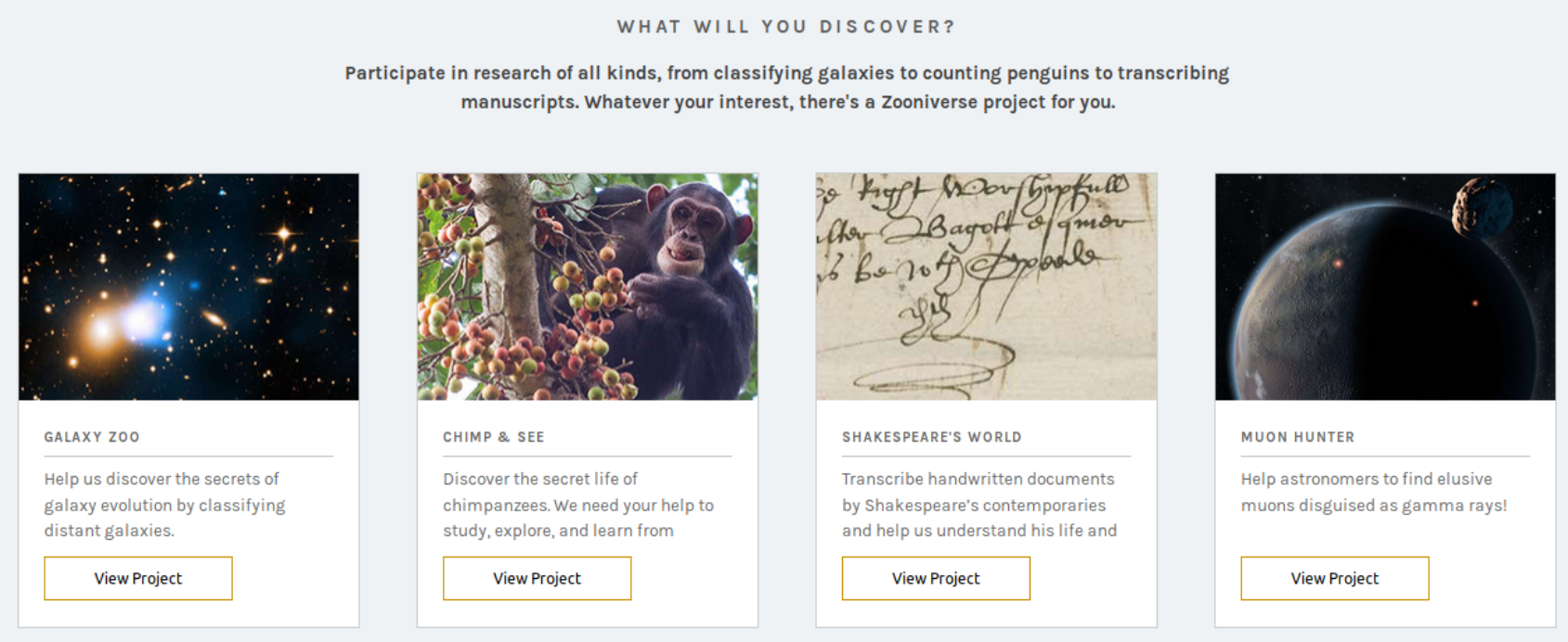
How it works

MTurk offers developers access to a diverse, on-demand workforce through a flexible user interface or direct integration with a simple API. Organizations can harness the power of crowdsourcing via MTurk for a range of use cases, such as microwork, human insights, and machine learning development.

https://www.mturk.com/

Strategies for data labeling

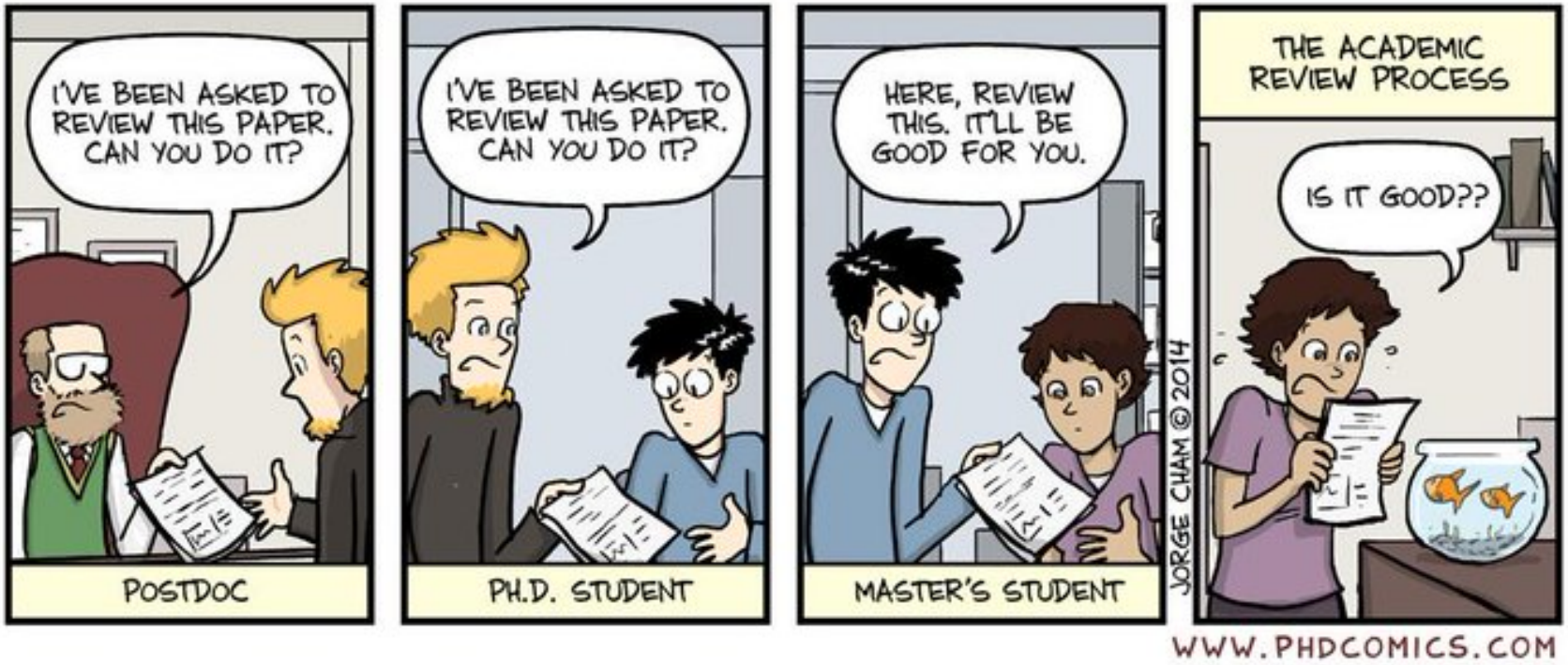
Crowdsourcing



https://www.zooniverse.org/

Strategies for data labeling

Homemade solution



This is a joke

Unsupervised learning

What can we do in the absence of target labels ?

Group data based on similarity $\Longrightarrow$ **clustering**

Simplify / reduce data dimension

Clustering

Grouping items into clusters
based on some similarity criteria

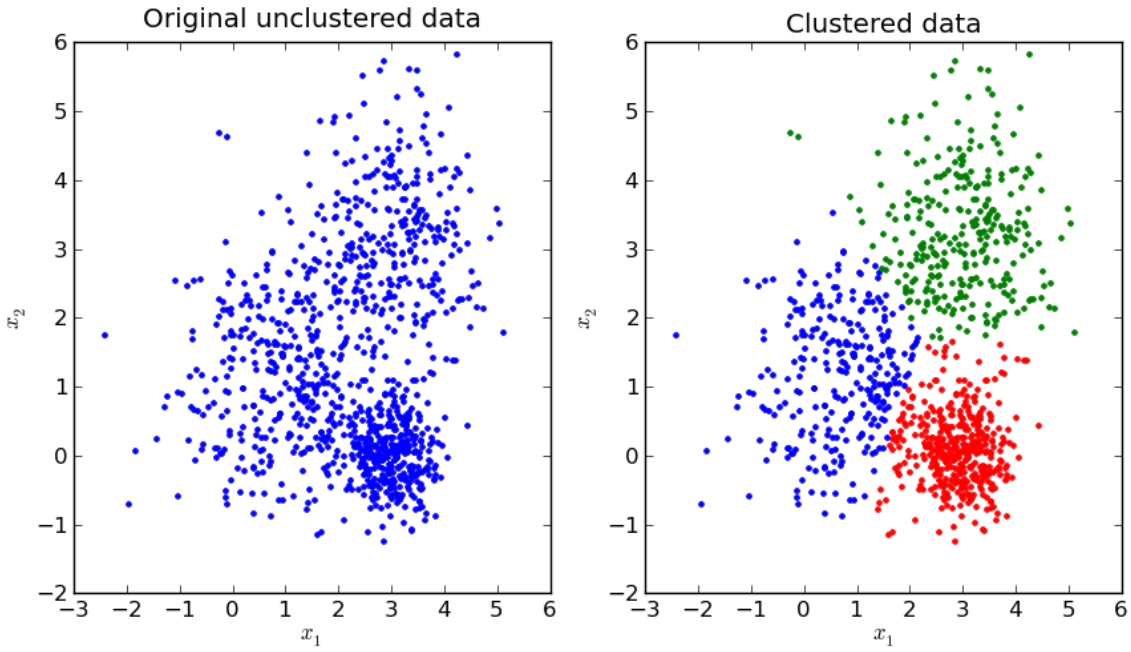
Standard clustering

Given: $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ (items without label)

Partition D into K disjoint clusters $\{C_1, C_2, \ldots, C_K\}$

K : number of clusters (usually unknown)

- $C_j \neq \emptyset, \qquad j = 1, \ldots, K$
- $\displaystyle\bigcup_{j=1}^{K} C_j = D$
- $C_i \cap C_j = \emptyset, \qquad i, j = 1, 2, \ldots, K,\ i \neq j$



Source: https://i.stack.imgur.com/cIDB3.png

What criterion we should use to build the clusters ?

Principle: An ideal clustering is one in which items within a group are highly similar each other, while items in distinct groups are highly dissimilar

Similarity must be adequately characterized (and it may depend on the application)

Clustering can be modeled as as problem of finding an optimal partition of D based on some cost function

Given N items

Number of bipartitions $= 2^{N-1} - 1$

Number of K-partitions (Stirling number):

$$\frac{1}{K!}\sum_{j=0}^{K}(-1)^{K-j}\binom{K}{j}j^N \sim \frac{K^N}{K!}$$

Impossible to build each one of them and compute their costs!

Approaches for clustering

Hierarchical: either agglomerative or divisive techniques

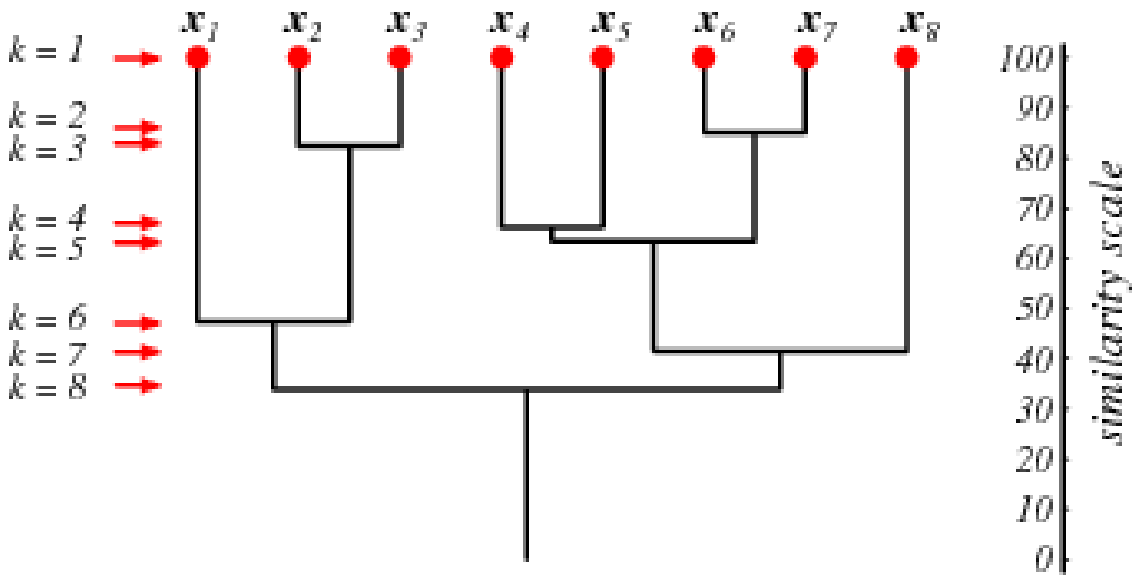
Iterative: clusters are iteratively changed, based on some cost function optimization

Hierarchical Clustering

(Agglomerative)

Agglomerative hierarchical clustering

Group items/clusters successively based on some similarity criterion (k: denotes the grouping step)



This tree-like structure is called *dendrogram*

Algorithm: Hierarchical agglomerative clustering

Initial partition: $\mathcal{C}_0 = \left\{ C_i = \{\mathbf{x}_i\}, \quad i = 1, 2, \ldots, N \right\}$

$k = 0$ (iteration)

While $|\mathcal{C}_k| > 1$

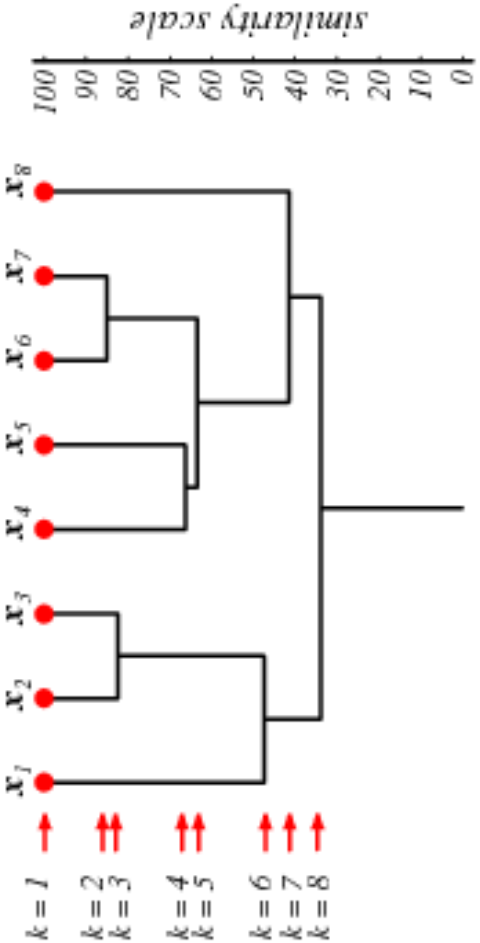
Among all pairs of clusters in $\mathcal{C}_k$, choose one that is most similar each other. Let (C_i, C_j) be such pair.

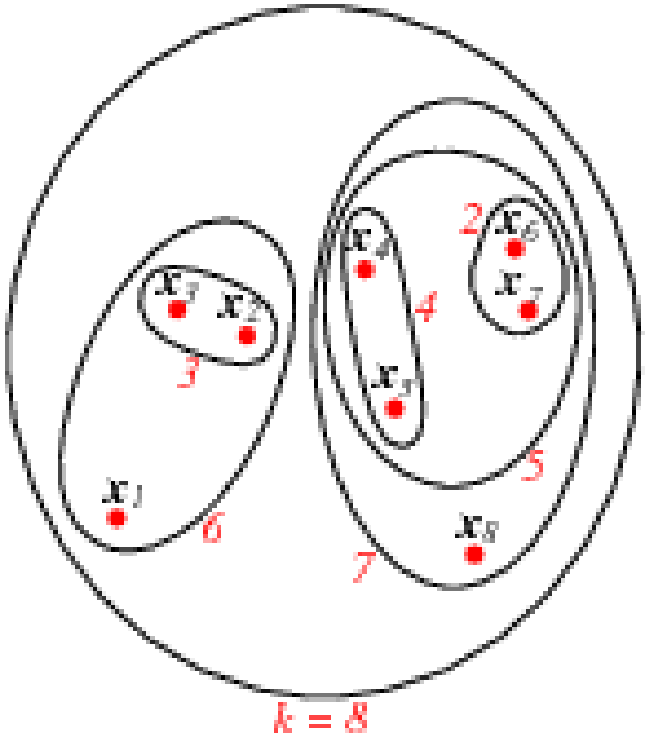
$C_q = C_i \cup C_j$

$\mathcal{C}_{k+1} = (\mathcal{C}_k \setminus \{C_i, C_j\}) \cup \{C_q\}$

$k = k + 1$

Cluster nesting





k : grouping iteration step

We need to **define similarity measures**:

- between two items: $s(\mathbf{x}_i, \mathbf{x}_j)$
- between an item and a group: $s(\mathbf{x}_i, C_k)$
- between two groups: $s(C_k, C_l)$

Variety of measures in the literature: Continuous data, categorical data, fuzzy measures, for documents, for images, for graphs, etc ...

similarity $\times$ **dissimilarity** (often related to distance measures)

Further reading: S. Santini and R. Jain, "Similarity measures," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 9, pp. 871-883, Sept. 1999, doi: 10.1109/34.790428

Distance between an item $\mathbf{x}$ and a cluster C:

- Smallest distance

$$d(\mathbf{x}, C) = \min_{\mathbf{y} \in C} d(\mathbf{x}, \mathbf{y})$$

- Largest distance

$$d(\mathbf{x}, C) = \max_{\mathbf{y} \in C} d(\mathbf{x}, \mathbf{y})$$

- Mean distance

$$d(\mathbf{x}, C) = \frac{1}{|C|} \sum_{\mathbf{y} \in C} d(\mathbf{x}, \mathbf{y})$$

Distance between an item $\mathbf{x}$ and a group C

Choose a representant $\mathbf{m}$ of group C and compute $d(\mathbf{x}, \mathbf{m})$

- **item** (spherical groups)
  - **mean item**: $\mathbf{m}_p = \dfrac{1}{|C|} \displaystyle\sum_{\mathbf{y} \in C} \mathbf{y}$
  - **central item**: $\mathbf{m}_c \in C$ such that
    $$\sum_{\mathbf{y} \in C} d(\mathbf{y}, \mathbf{m}_c) \leq \sum_{\mathbf{y} \in C} d(\mathbf{y}, \mathbf{m}), \quad \forall \mathbf{m} \in C$$
- **hyperplane, hypercircle**: distance between item $\mathbf{x}$ and the curve that represents the cluster C

Distance/similarity between groups

- **Single linkage** (smallest)

$$d(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y})$$

- **Complete linkage** (largest)

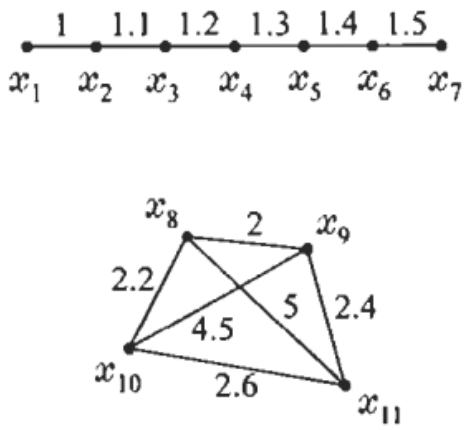
$$d(C_i, C_j) = \max_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y})$$

- **Average linkage** (mean)

$$d(C_i, C_j) = \frac{1}{|C_i|\,|C_j|} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y})$$

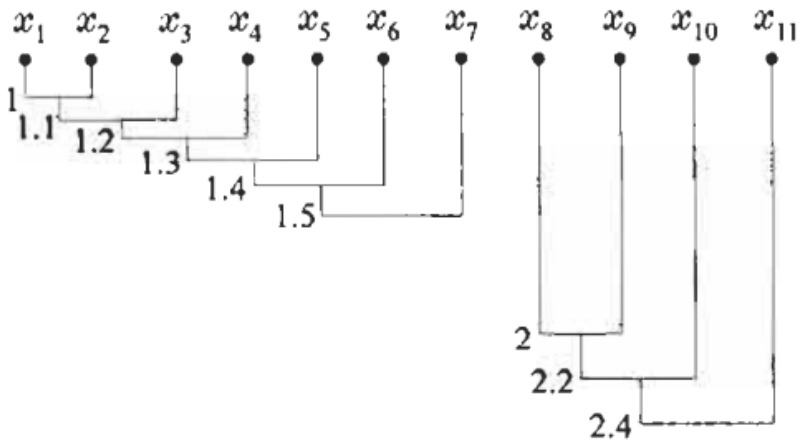
- distance between the representants of the groups

items and distances



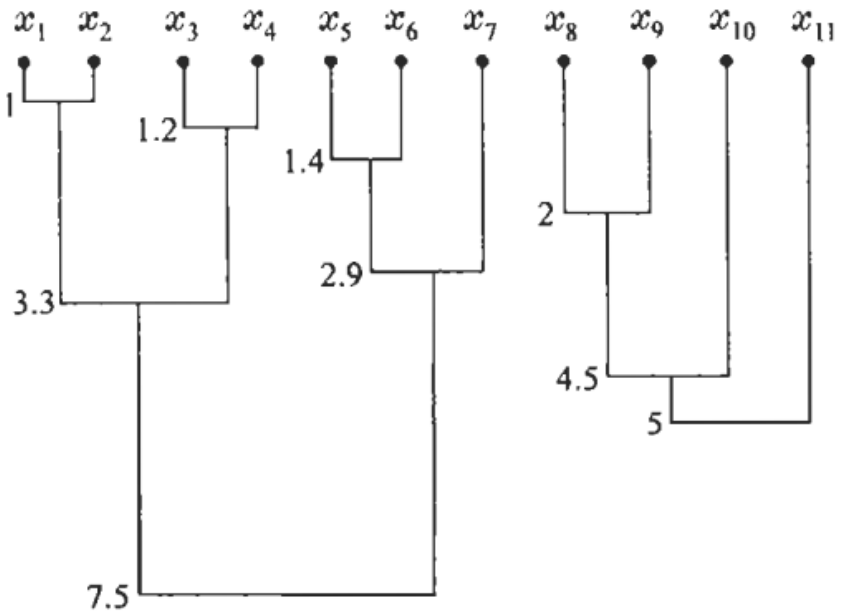
(a)

Single linkage



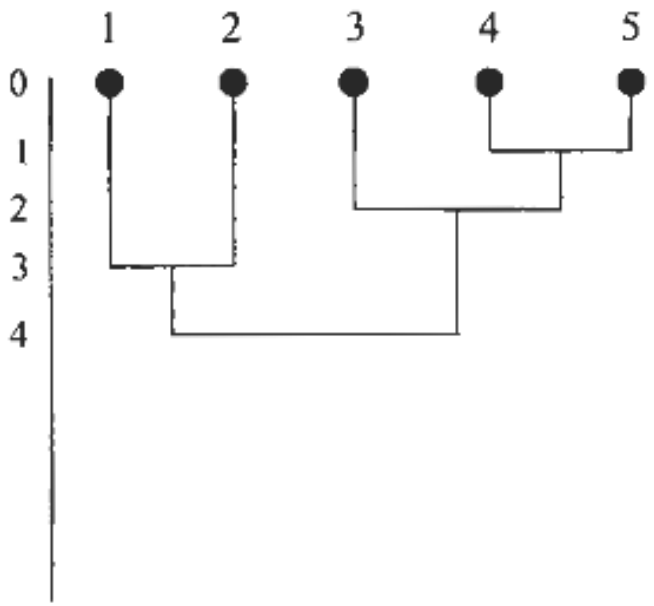
(b)

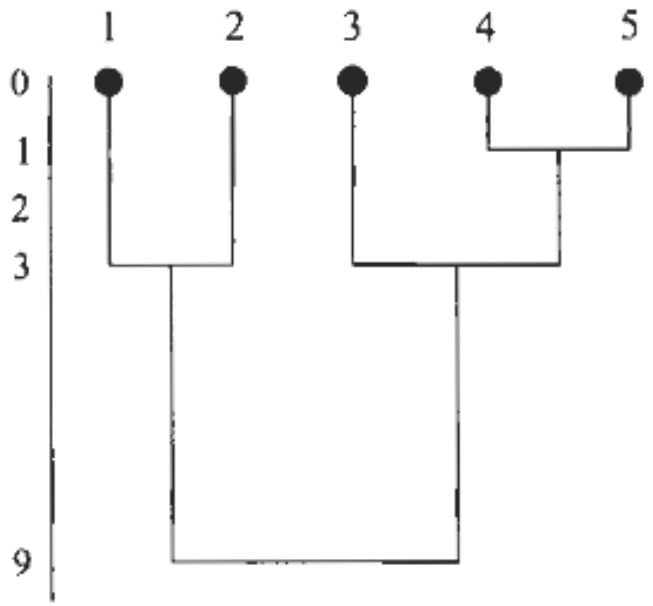
Complete linkage



How to define the final clusters ?

Lifetime of a cluster: difference between the similarity measure when it was created and when it was merged with another cluster





$\Longrightarrow$ Subjective

$\Longrightarrow$ Visualization becomes problematic if there are too many items

Some characteristics of agglomerative clustering:

- easy to understand
- any similarity metric can be used to group clusters
- requires large memory space
- not possible to undo previous grouping
- sensible to noise
- resulting dendrogram depends on the metric

Clustering

Iterative algorithms

(Best known: K-**means**)

Iterative algorithm mechanics

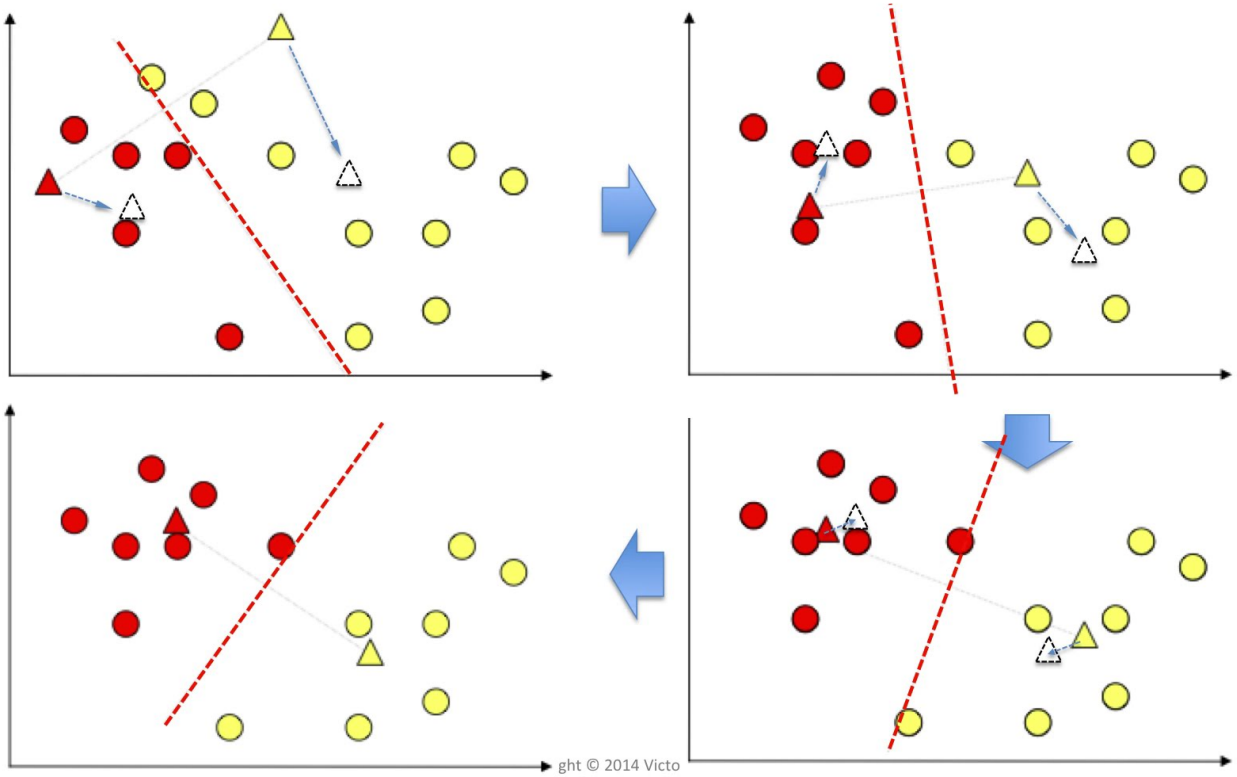
- Start the process with an arbitrary partition with K clusters
- Repeat until a stop criterion is met
  - update the partition
  - check if the updated partition has better cost and, if so, replace the previous partition with this one
- Return the current partition (the best obtained so far)

K-means algorithm

K, number of clusters, must be chosen a priori

1. Choose K items. They will be considered as the centroids (mean points) of the K initial clusters
2. Assign each item in D to the cluster of the closest centroid
3. Recompute the centroids, based on the assignment results
4. Repeat steps 2 and 3 until the centroids do not change or until some stop criterion is reached

K-means clustering example



Clustering

Additional comments

How to validade clusters

- Run the algorithm multiple times, varying its paramaters – if there is a pattern, the results should be similar
- Run distinct algorithms – if there is a pattern, multiple algorithms should find the same pattern
- Confront with expert knowledge – does the cluster make sense from the point of view of the application domain ?

Comparing clsusterings

Index (values in $[0, 1]$) that indicate how distinct/similar are two clusterings

- Purity
- Rand index
- Mutual information
- etc

References:

- Section 25.1, book by Kevin P. Murphy (Machine Learning: a Probabilistic Perspective)
- Davies, David L.; Bouldin, Donald W., "A Cluster Separation Measure," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.PAMI-1, no.2, pp.224,227, April 1979 (doi: 10.1109/TPAMI.1979.4766909)
- Marina Meila, Comparing clusterings – an information based distance, Journal of Multivariate Analysis, Volume 98, Issue 5, May 2007, Pages 873-895, http://dx.doi.org/10.1016/j.jmva.2006.11.013.

Clustering evaluation

An **ideal clustering** is one in which items within a group are highly similar each other, while items in disctinct groups are highly dissimilar

There are two common cluster similarity measures:

- within-class
- between-class

Number of items: N

K clusters: C_j, $j = 1, 2, \ldots, K$

Center point of cluster C_j with n_j items:

$$\overline{\mathbf{x}}_j = \frac{1}{n_j} \sum_{\mathbf{x} \in C_j} \mathbf{x}$$

Global center point of the total of N items:

$$\overline{\mathbf{x}} = \frac{1}{N} \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{x} = \frac{1}{N} \sum_{j=1}^{K} n_j \overline{\mathbf{x}}_j$$

Scatter matrices (related to covariance matrices)

Scatter matrix of cluster j:

$$S_j = \sum_{\mathbf{x} \in C_j} (\mathbf{x} - \overline{\mathbf{x}}_j)(\mathbf{x} - \overline{\mathbf{x}}_j)^t$$

Intra-class scatter matrix

$$S_W = \sum_{i=j}^{K} S_j$$

Between-class scatter matrix

$$S_B = \sum_{j=1}^{K} n_j (\overline{\mathbf{x}}_j - \overline{\mathbf{x}})(\overline{\mathbf{x}}_j - \overline{\mathbf{x}})^t$$

Total scatter matrix

$$S_T = \sum_{\mathbf{x} \in \mathbf{X}} (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^t$$

Some criteria based on scatter matrices

$$S_T = S_W + S_B$$

$$J_1 = \frac{\text{trace}\{S_T\}}{\text{trace}\{S_W\}}$$

$$J_2 = \frac{|S_T|}{|S_W|} = |S_W^{-1} S_T|$$

$$J_3 = \text{trace}\{S_W^{-1} S_T\}$$

Fuzzy clustering

Membership function:

$$u_j : D \to [0, 1], \quad j = 1, 2, \ldots, c$$

The sum must amount to 1:

$$\sum_{j=1}^{c} u_j(\mathbf{x}_i) = 1, \quad i = 1, 2, \ldots, N$$

At least one item with non-null membership score in each class j:

$$0 < \sum_{i=1}^{N} u_j(\mathbf{x}_i) < N, \quad j = 1, \ldots, c$$

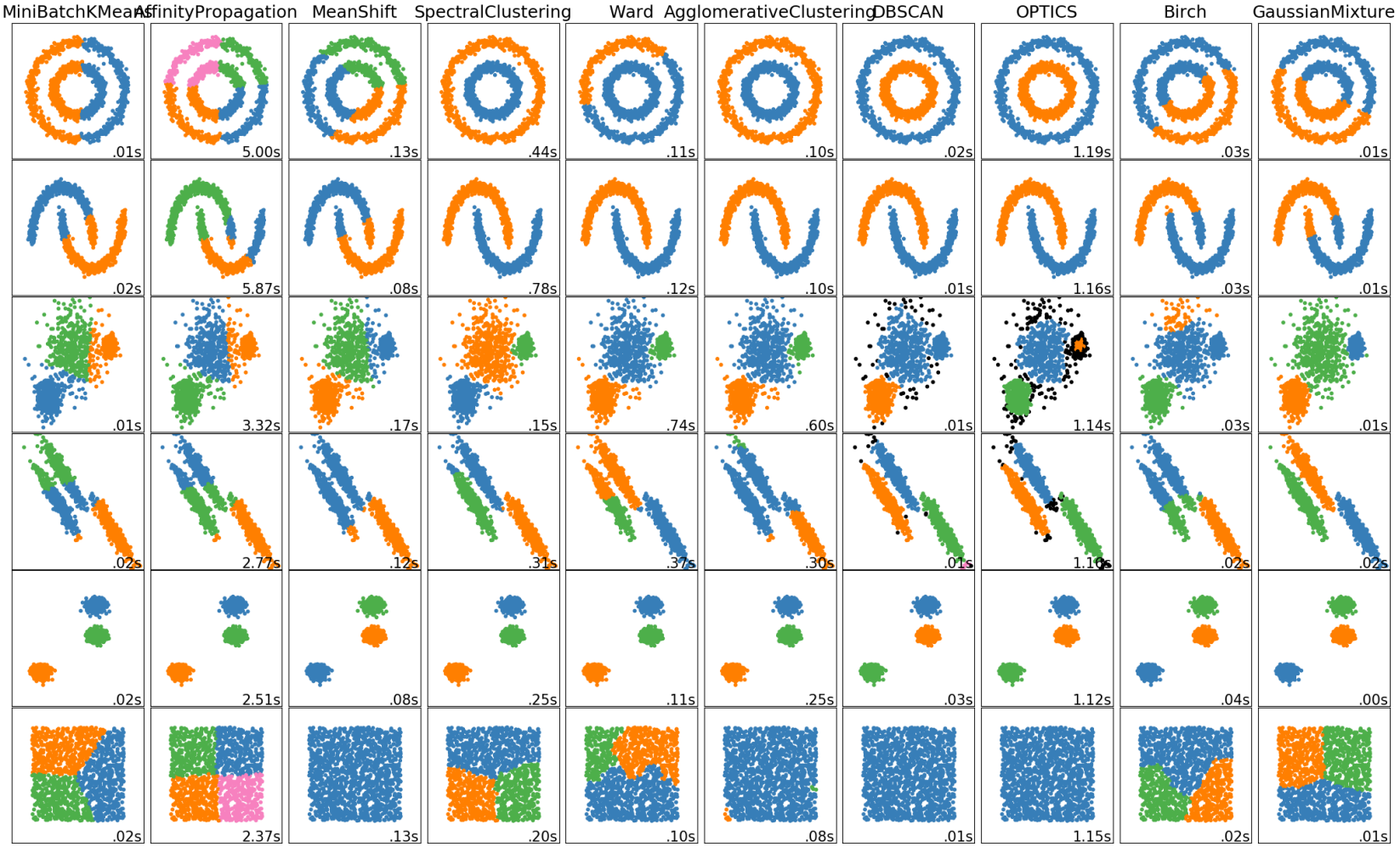
Standard clustering: for every $\mathbf{x} \in D$, there exists j such that $u_j(\mathbf{x}) = 1$ and $u_k(\mathbf{x}) = 0$ for all $k \neq j$

Big data and clustering

Proliferation of algorithms

Some clustering algorithms (circa 2013):

> Hierarchical, CURE, ROCK, Chameleon, k-means, k-medoids, Fuzzy, PAM, CLARA, CLARANS, SOM, DBSCAN, DENCLUE, CLIQUE, etc, etc



Source https:/https://scikit-learn.org/stable/modules/clustering.html

SOM - Self organizing maps

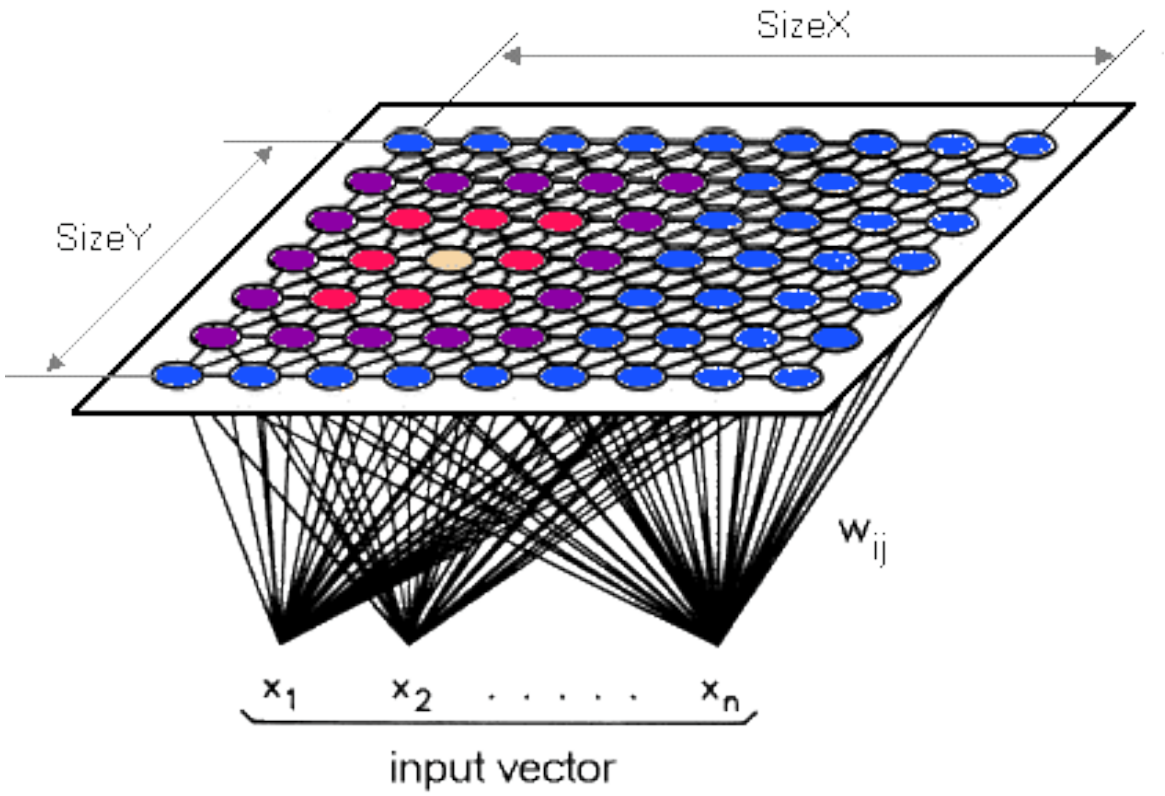
(Kohonen maps)

SOM (Self-organizing maps)

Idea: to map points from a high dimensional space to a low dimension space, keeping the proximity relation among items

- Original data in $\mathbb{R}^d$
- The new space is usually a 2D map with a discrete grid of nodes
- Each node $\mathbf{p}_k$ of the map has a coordinate and an updatable weight vector $\mathbf{w}_k \in \mathbb{R}^d$ assigned to it
- **OBS.**: it is frequently referred to as a type of neural network, but it is not similar to the networks we have seen previously

Architecture of a SOM



BMU (best matching unit) (yellow node): given $\mathbf{x}$, it is the node $\mathbf{p}^*$ in the map that has the closest weight vector to $\mathbf{x}$

Neighboors of BMU (red and purple nodes): $V(\mathbf{p}^*)$, defined by a kernel matrix (nothing to do with SVM kernel)

Examples of 1D kernel

- Rectangular

$$\phi(u) = \begin{cases} \frac{1}{2}, & \text{se } |u| < 1, \\ 0, & \text{c.c.} \end{cases}$$

- Triangular

$$\phi(u) = \begin{cases} 1 - |u|, & \text{se } |u| < 1, \\ 0, & \text{c.c.} \end{cases}$$

- Biweight

$$\phi(u) = \begin{cases} \frac{15}{16}(1 - u^2)^2, & \text{se } |u| < 1, \\ 0, & \text{c.c.} \end{cases}$$

- Gaussian (most frequently used)

$$\phi(u) = \frac{1}{\sqrt{2\pi}} e^{\frac{-u^2}{2}}$$

- Bartlett-Epanechnenko

$$\begin{cases} \frac{3}{4}\left(1 - \frac{u^2}{\sqrt{5}}\right), & \text{se } |u| < \sqrt{5}, \\ 0, & \text{c.c.} \end{cases}$$

SOM Algorithm

$t = 0$
Initialize $\mathbf{w}(0)$, $\forall \mathbf{p}$ on the map
Repeat
    For each $\mathbf{x} \in D$
        Let $\mathbf{p}^*$ be the BMU
        For each $\mathbf{p} \in \{\mathbf{p}^*\} \cup V(\mathbf{p}^*)$
            Let $\mathbf{w}$ be the weight vector of $\mathbf{p}$
            $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta(t)\phi(\mathbf{p}^* - \mathbf{p})(\mathbf{x} - \mathbf{w}(t))$
Until convergence

ϕ is a window function (or kernel function) — its value at the center is 1 and tends to diminish as we move away

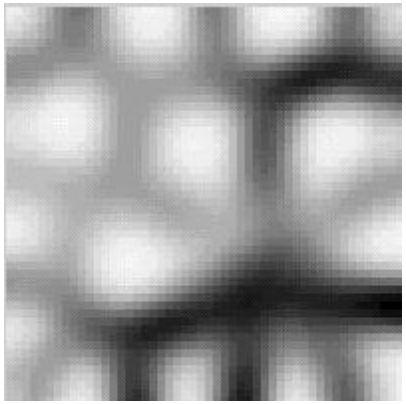
Thus, the weight $\mathbf{w}$ of the BMU and its closest neighboors become "more similar" to $\mathbf{x}$

The BMUs of two similar examples $\mathbf{x}_i$ and $\mathbf{x}_j$ will be close each other in the map

Thus, we may see a SOM as a projection from $\mathbb{R}^d$ to $\mathbb{R}^2$ that preserves neighboorhood relations at some extent

How do we find "clusters" in SOMs ?

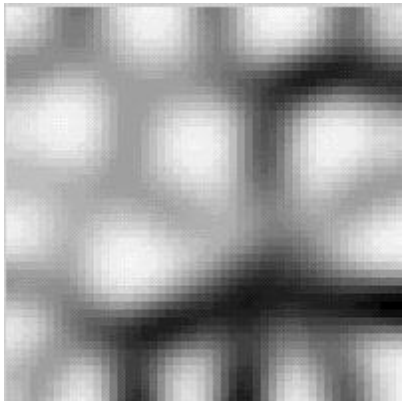
U-matrix (unified distance matrix): for each node of the map, compute distance between its weight to the weights of its neighboors (e.g., mean distance)



Color of nodes in the map: the darker the color, the larger the distance

White regions are the "clusters"

Dark regions are the frontiers



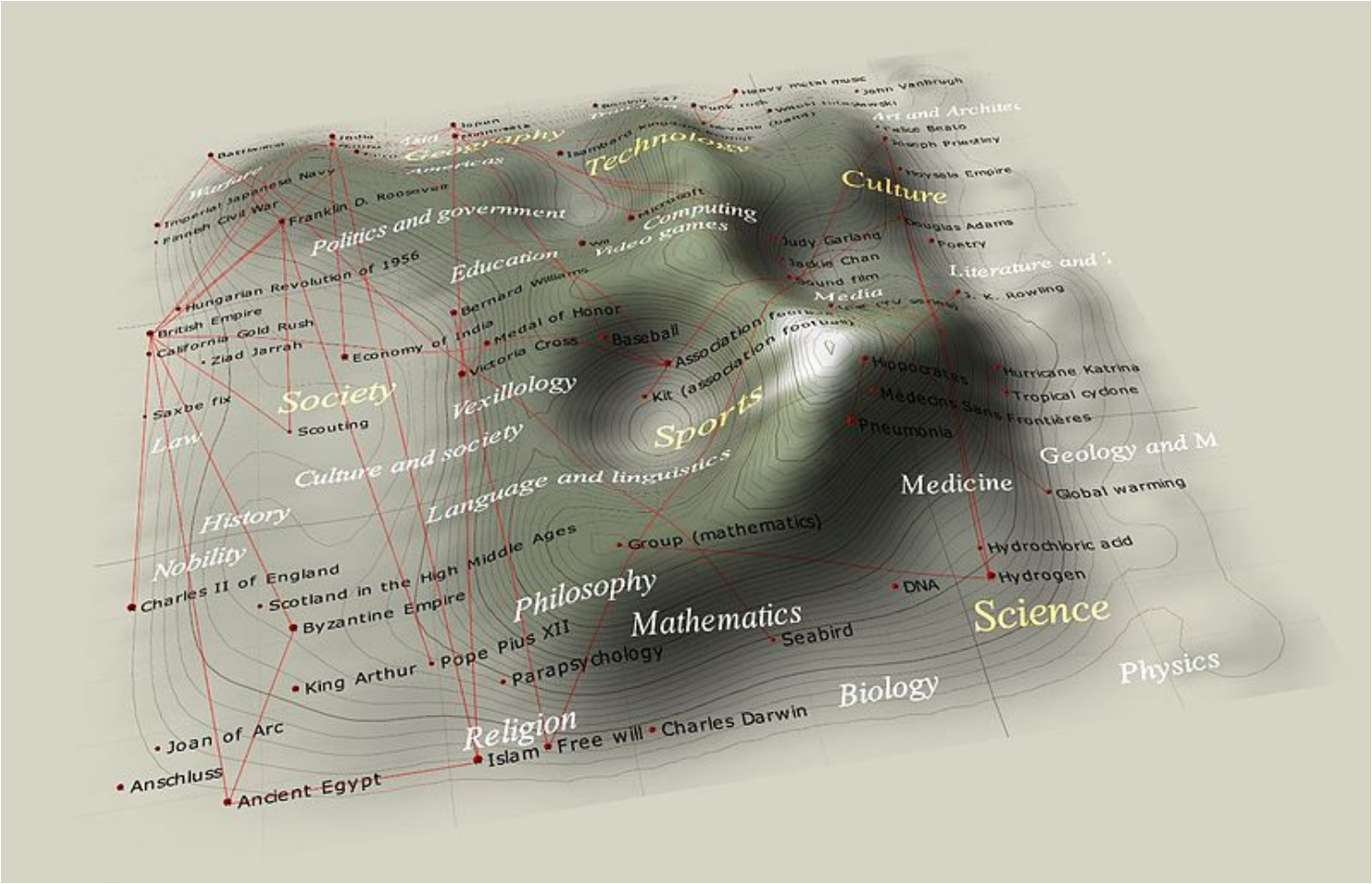
How many clusters?

What about the points that fall in the dark regions (frontiers) ?

No simple answer

We can employ image processing techniques for segmenting the "white" regions ...

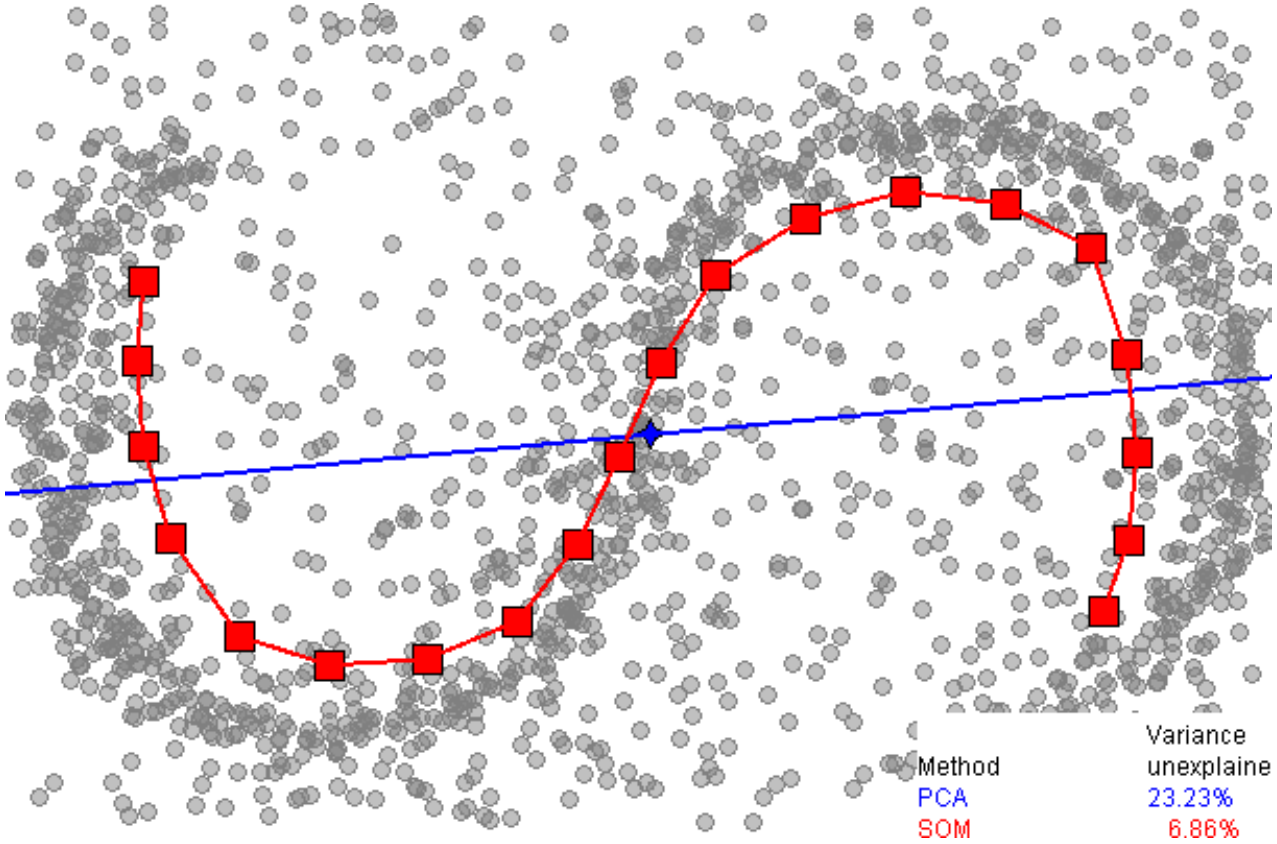
Example



Data: wikipedia articles, represented in terms of word frequencies (tf-idf perhaps?); red lines indicate links between articles

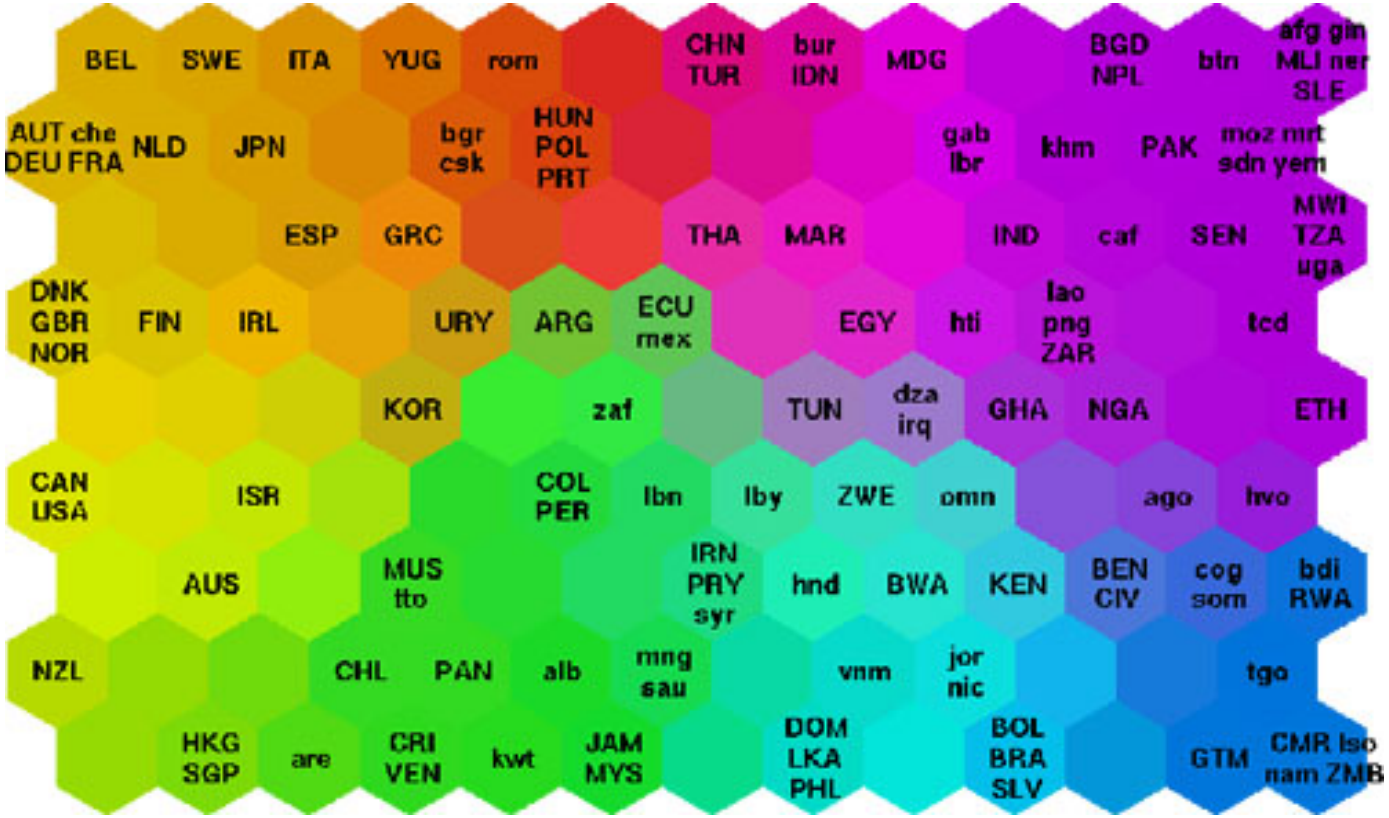
Example: Discrete approximation of the distribution in $\mathbb{R}^d$

Unidimensional SOM (20 nodes): each weight vector of the map "points" to a region of $\mathbb{R}^d$

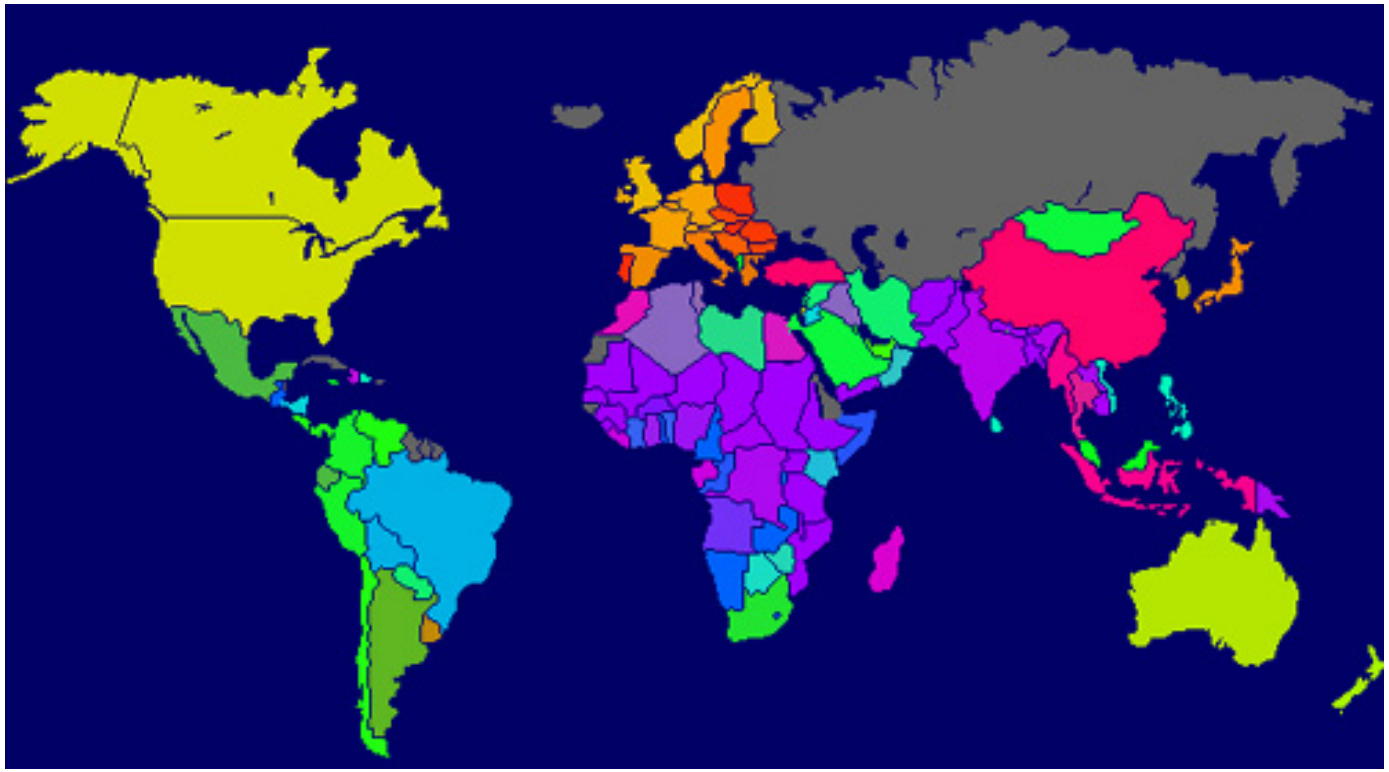


Blue: principal components Red: weight vectors of SOM

Example: the original space consists of diverse statistics related to country indicators (education, health, ...)



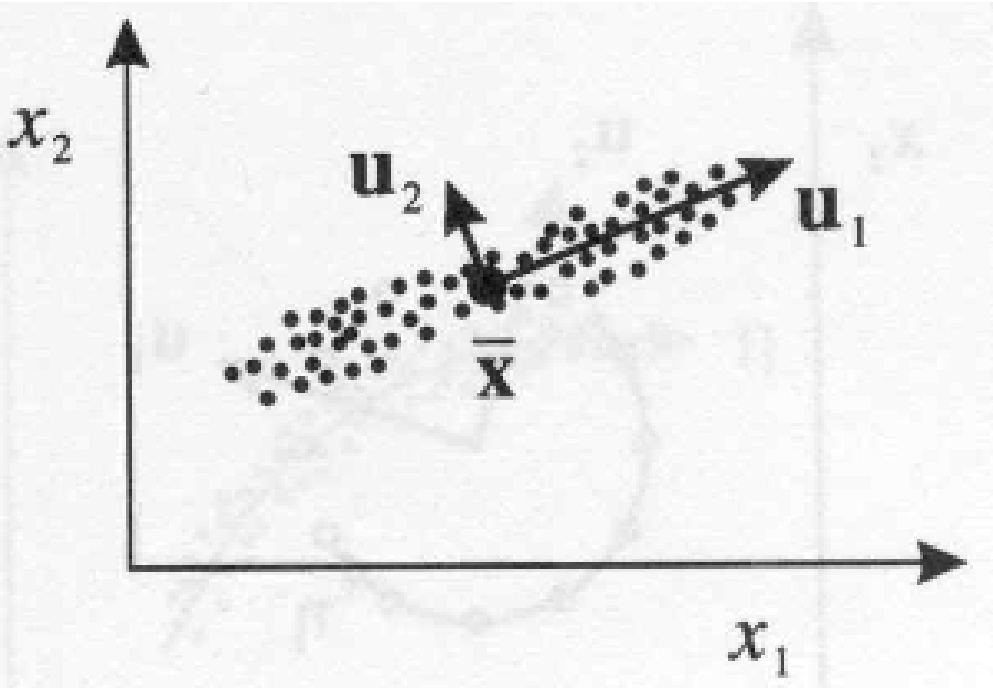
The colors of the nodes can be attributed by using three components of the weight vector as the RGB intensities or by means of pseudo-coloration



Data simplification / reduction

PCA – Principal component analysis

PCA – Intuition



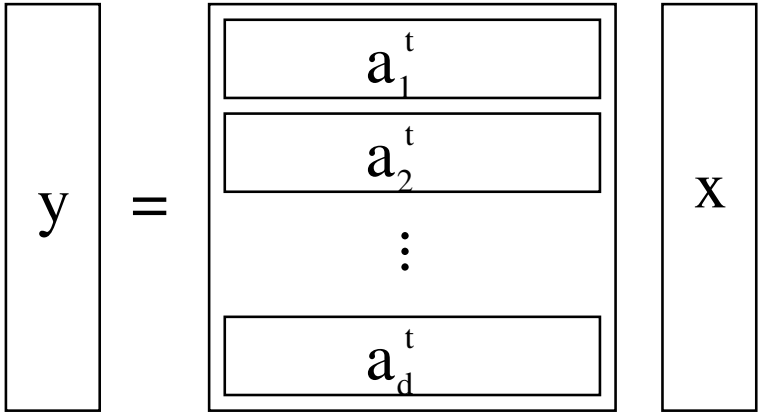
Change the basis, to align the axis to the directions with largest dispersion of the points.

PCA

The goal is to find a linear transformation $\mathbf{y} = \mathbf{A}\mathbf{x}$ that

(1) maximizes the variance of the transformed variables $y_i = \mathbf{a}_i^T \mathbf{x}$
(y_i can be seen as a projection of $\mathbf{x}$ on $\mathbf{a}_i$)

(2) any two vectors $\mathbf{a}_i$ and $\mathbf{a}_j$ are orthogonal each other



The desired linear transformation can be obtained from the eingenvectors of the covariance matrix of D

Covariance matrix: Example for $d = 3$

$$S = \begin{bmatrix} s_{11} & s_{12} & s_{13} \\ s_{12} & s_{22} & s_{23} \\ s_{13} & s_{23} & s_{33} \end{bmatrix}$$

Diagonals: variance of x_1, x_2, and x_3

Remaining elements: covariance

$$s_{ij} = s_{ji} = \frac{1}{N-1} \sum_{n=1}^{N} (x_i^{(n)} - \overline{x}_i)(x_j^{(n)} - \overline{x}_j)$$

S : covariance matrix of dataset D

The covariance matrix S is symmetric and positive semi-definite ($\mathbf{x}^t\, S\, \mathbf{x} > 0, \forall \mathbf{x} \neq \mathbf{0}$)

Let $\lambda_1 > \lambda_2 > \ldots > \lambda_d > 0$ be the eingenvalues of S (they exist and are all real, non-negative and distinct since S is a symmetric positive definite matrix)

Let $\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_d}$ be the unit eigenvectors of S corresponding to the eingenvalues $\lambda_1 > \lambda_2 > \ldots > \lambda_d$

Let M be the matrix whose columns are the eingenvectors $\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_d}$ of S

S is diagonalizable : $\Lambda = M^{-1}SM$

If we set $\mathbf{y} = M^{-1}\mathbf{x}$, we have:

- $y_i = \mathbf{e}_i^t\mathbf{x}$ is the i-th **principal component**
- $Var(y_i) = \lambda_i$
- $Cov(y_i, y_j) = 0, \forall j < i$

PCA keeps the total variance

$$\begin{aligned}\sum_{i=1}^{d} Var(x_i) &= s_{11} + s_{22} + \cdots + s_{dd} \overset{(1)}{=} tr(S) \\ &\overset{(2)}{=} tr(M\Lambda M^{-1}) \overset{(3)}{=} tr(M\Lambda M^T) \overset{(4)}{=} tr(\Lambda M^T M) \\ &\overset{(5)}{=} tr(\Lambda M^{-1} M) \overset{(6)}{=} \lambda_1 + \lambda_2 + \cdots + \lambda_d \overset{(7)}{=} \sum_{i=1}^{d} Var(y_i)\end{aligned}$$

(1) s_{ii} is the variance of x_i; variances are in the diagonal of S

(2) $\Lambda = M^{-1}SM \Longrightarrow S = M\Lambda M^{-1}$

(3,5) M is orthogonal; hence $M^{-1} = M^T$

(4) $tr(AB) = tr(BA)$

(6) the diagonal of Λ contains the eigenvalues of S

(7) property (previous page)

Interpretation

Total variance is equal to the sum of the eigenvalues

Thus, the **percentage of total variance explained by the k-th principal component** is

$$\frac{\lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_d}$$

Dimensionality reduction using PCA

How many components one should choose?

Choose the d' principal components such that

$$\frac{\sum_{i=1}^{d'} \lambda_i}{\sum_{i=1}^{d} \lambda_i} > T$$

In general, $T = 0.90$ or $T = 0.95$

In many cases, a few components is sufficient to explain most of the total variance.

What is the meaning of the selected components ?

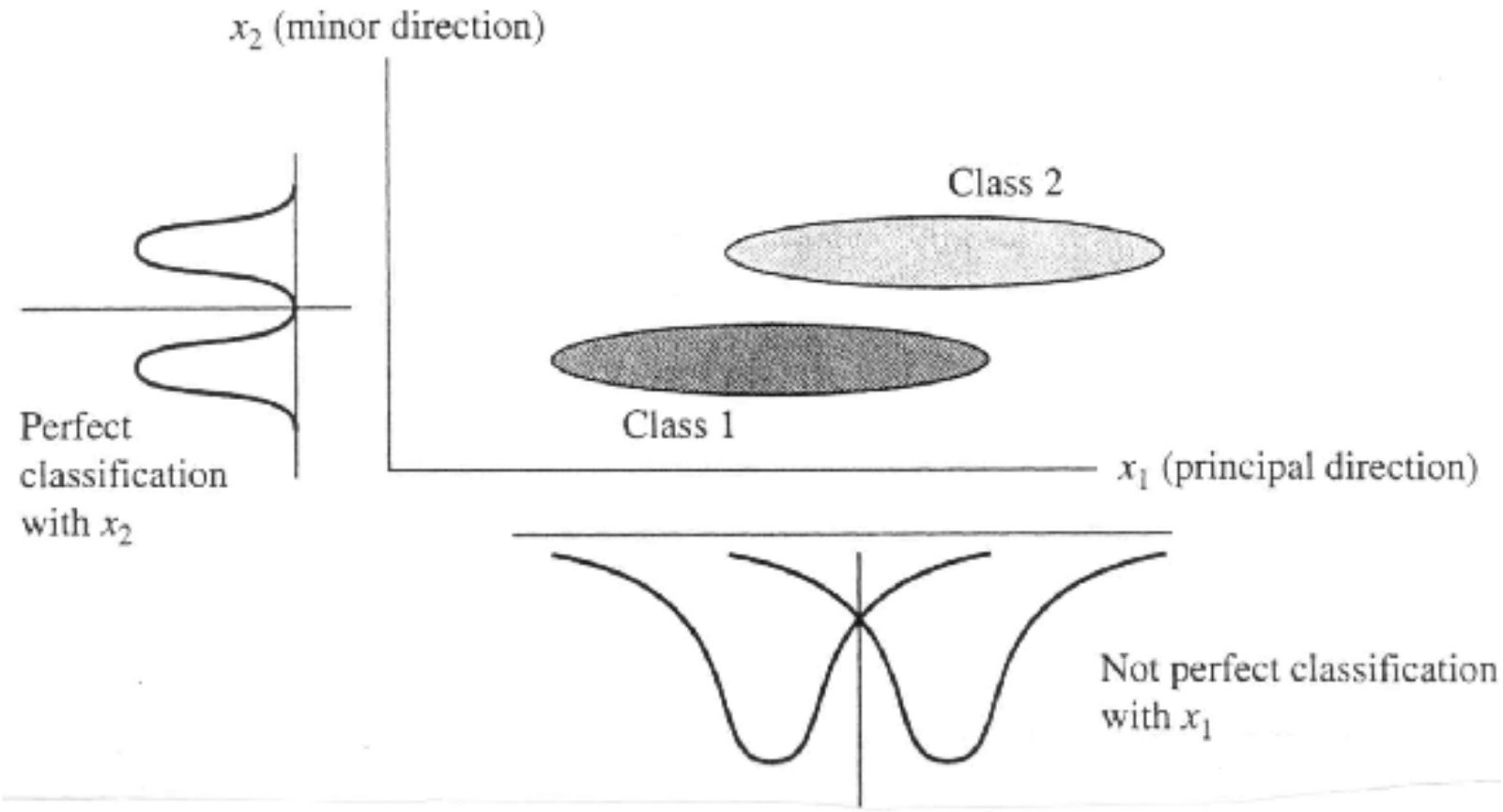
How do they relate to the original variables ?

We can examine the coefficients of the original variables in the projection $y_i = \mathbf{e}_i^t \mathbf{x}$

The magnitude of the Coefficients indicate the relevance or how each original variable contributes to the selected components.

Dimensionality reduction using PCA

In classification problems, not necessarily good



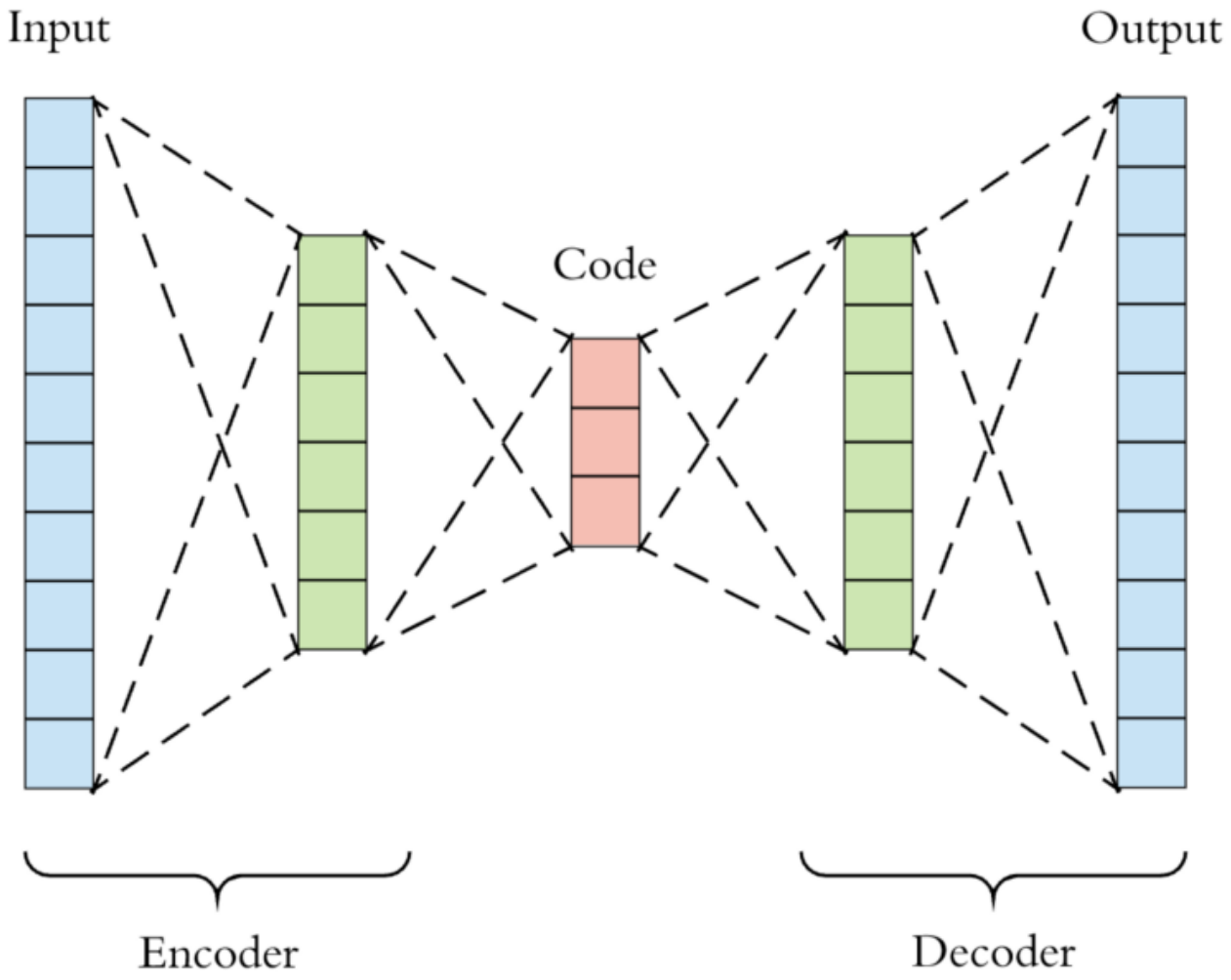
Some references for PCA

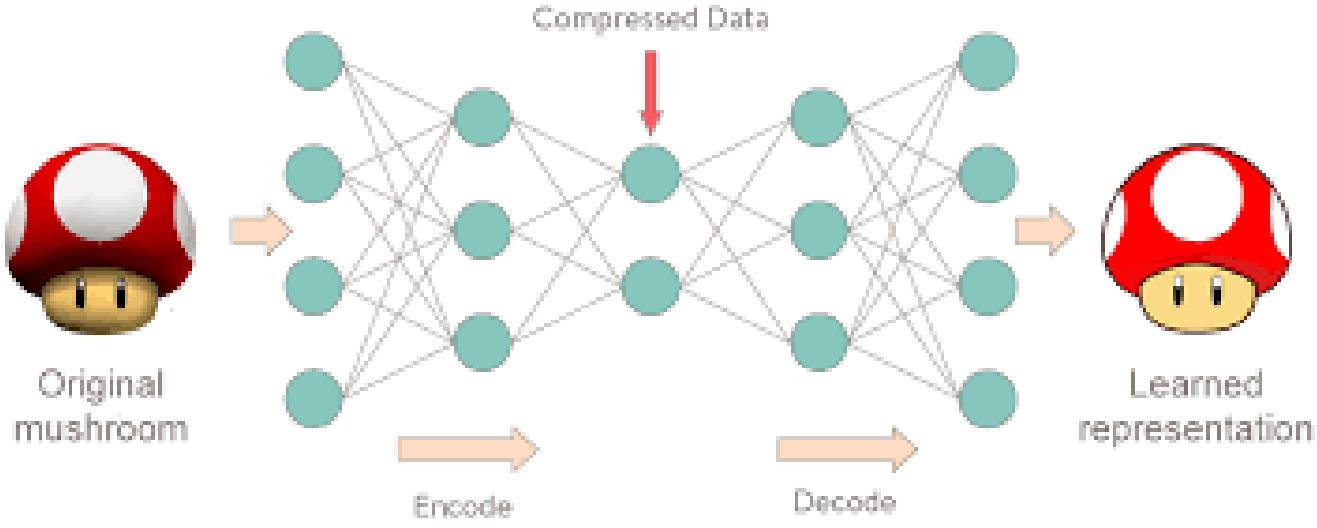
- Richard A. Johnson and Dean W. Wichern, *Applied Multivariate Statistical Analysis*, Prentice Hall
- Jon Shlens, *Tutorial on Principal Component Analysis*, December 2005.
- Lindsay I. Smith, *A tutorial on Principal Components Analysis*, February 2002 (some application examples in its final part)

Data simplification / reduction

Auto-encoders

Auto-encoders




Compressed Data
Original mushroom
Encode
Decode
Learned representation