Machine Learning

5 – Clustering

SS 2018

Gunther Heidemann

Overview

Motivation

Why clustering?

- Clusters are basic structures.
- Clusters in some feature space may indicate closeness of data on the level of semantics.
- Clusters imply rules (“if $x \in [3,6]$ then $y \in [-1,2]$”).
- Compression can be achieved by transmitting only cluster centers (instead of the data belonging to the clusters).

The definition of a cluster requires a distance measure.

Ideas for distance measures?

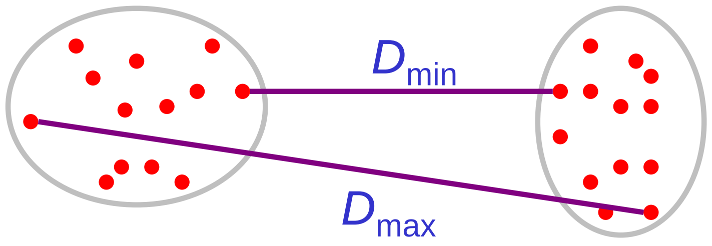
Cluster distance measures

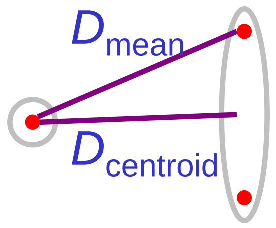
So far: Distance between *points*.

Now: Distance between *clusters*.

For clusters X and Y:

$D_{\mathrm{min}}(X,Y) = \min_{\vec{x}\in X,\vec{y}\in Y} d(\vec{x},\vec{y})$	minimum distance
$D_{\mathrm{max}}(X,Y) = \max_{\vec{x}\in X,\vec{y}\in Y} d(\vec{x},\vec{y})$	maximum distance
$D_{\mathrm{mean}}(X,Y) = 1/\lvert X\rvert\lvert Y\rvert \sum_{\vec{x}\in X,\vec{y}\in Y} d(\vec{x},\vec{y})$	mean of all distances
$D_{\mathrm{centroid}}(X,Y) = d\left(1/\lvert X\rvert \sum_{\vec{x}\in X} \vec{x},\ 1/\lvert Y\rvert \sum_{\vec{y}\in Y} \vec{y}\right)$	distance of centers



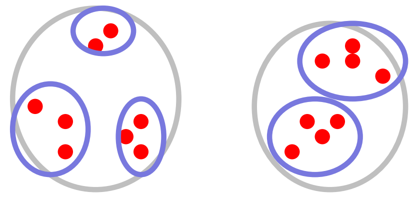


[H]

Note D_{min}, D_{max}, D_{mean} only need distances, but D_{centroid} requires numerical attribute values, so the „Luke-Leia-Han“ table wouldn´t work!

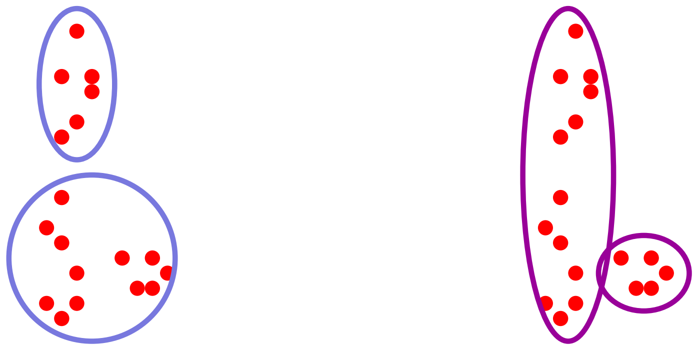
Definition of clusters

Given a data distribution, it is not clear what clusters an algorithm should find:



[H]

Both results make sense – definition of clusters depends on the scale.



[H]

Both results make sense, depending on preferred shape.

Bias in clustering

All clustering algorithms have a bias:

- Bias prefers a certain cluster model.
- The model comprises scale and shape of clusters.
- Optimally, the bias / model can be chosen explicitly.
- However, usually the bias / model is an implicit part of the algorithm.
- Adjustable parameters are usually processing parameters (of the algorithm), not model parameters.
- The connection between parameters and the implicit cluster model usually needs to be inferred from the way the algorithm is working.
- Hierarchical clustering solves the problem for the scale parameter insofar as all solutions on different scales are presented in an ordered way.

Hierarchical clustering

Two complementary methods:

- Agglomerative clustering:
  - Start with each data point as a cluster.
  - Merge clusters recursively bottom up.
- Divisive clustering:
  - Start with all data points as a single cluster.
  - Split clusters recursively top down.

The result is a dendrogram representing all data in a hierarchy of clusters.

Hierarchical clustering

Basic agglomeration algorithm:

1. Initialization: Assign each of n data elements to a cluster C_i, $i = 1...n$.
2. Find the pair of clusters C_i, C_j, $i < j$, that optimizes the linkage criterion.
3. Merge: $C_i \leftarrow C_i \cup C_j$.
4. If $j < n$: $C_j \leftarrow C_n$.
5. If $n > 2$: n-- ; goto 2.

Step 2 requires a distance measure.

The following algorithms differ by the choice of the distance measure.

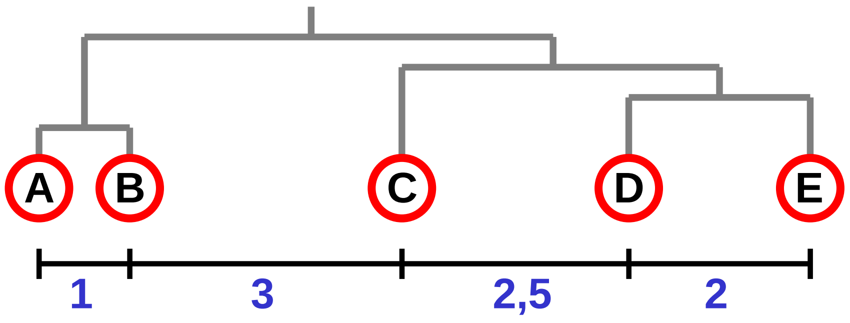
Single and complete linkage clustering

Single linkage clustering employs the *minimum* cluster distance

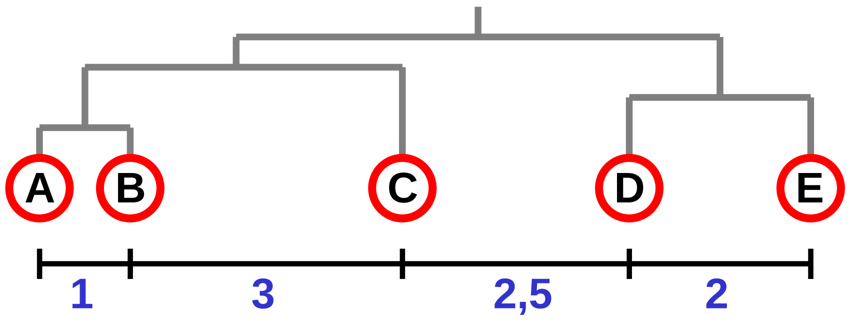
$$D_{\min} = \min_{\vec{x}\in X,\vec{y}\in Y} d(\vec{x},\vec{y}).$$

Complete linkage clustering employs the *maximum* cluster distance

$$D_{\max} = \max_{\vec{x}\in X,\vec{y}\in Y} d(\vec{x},\vec{y}).$$



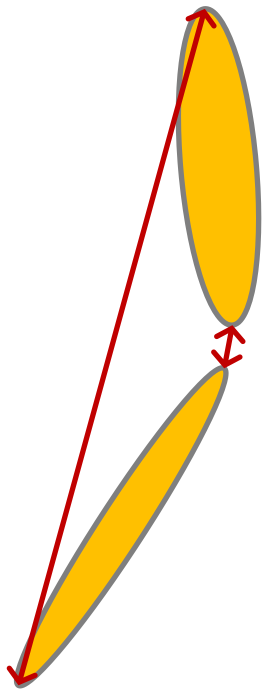
Single linkage

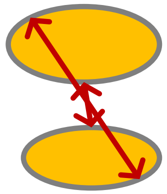


Complete linkage [H]

Single and complete linkage clustering

- Single linkage clustering tends to chaining
- Complete linkage clustering prefers compact clusters





Good single and good
complete linkage cluster

Good single linkage cluster
but bad complete linkage cluster

[H]

Average linkage and centroid clustering

Average linkage clustering or ***UPGMA***

(*Unweighted Pair Group Method with Arithmetic mean*):

$$D_{\text{mean}} \quad = \quad 1/|X||Y| \sum_{\vec{x}\in X,\vec{y}\in Y} d(\vec{x},\vec{y}).$$

Centroid clustering:

$$D_{\text{centroid}} \quad = \quad d\left(1/|X| \sum_{\vec{x}\in X} \vec{x},\ 1/|Y| \sum_{\vec{y}\in Y} \vec{y}\right).$$

- Clusters are represented by their centroids.
- Real valued attributes required for centroid computation.
- When joining two clusters, the resulting centroid is dominated by the cluster with more members.

Ward's minimum variance clustering

Idea:

Merge the pair of clusters for which the increase in total variance

$$E = \sum_i \sum_{\vec{x} \in C_i} (\vec{x} - \vec{\mu}_i)^2,$$

$$\vec{\mu}_i = 1/|C_i| \sum_{\vec{x} \in C_i} \vec{x}\ ,$$

is minimized.

In contrast to the previous approaches, this one is *optimization based*.

But it can be implemented by a distance measure:

$$D_{\text{Ward}} = D_{\text{centroid}}(X, Y) \ / \ \left(1/|X| + 1/|Y|\right).$$

Properties:

- Prefers spherical clusters and clusters of similar size (#members).
- Robust against noise but not against outliers (variance!)

Minimum energy clustering

Idea:

For merging, do not only evaluate the inter-cluster distance but take into account the size of clusters, preferring small ones:

$$
\begin{aligned}
D_{\text{energy}} \;=\; & 2/(|X||Y|) \textstyle\sum_{\vec{x}\in X,\vec{y}\in Y} |\vec{x}-\vec{y}\,|^2 \\
& -1/|X|^2 \textstyle\sum_{\vec{x},\,\vec{x}'\in X} |\vec{x}-\vec{x}'|^2 \\
& -1/|Y|^2 \textstyle\sum_{\vec{y},\,\vec{y}'\in Y} |\vec{y}-\vec{y}'|^2 \\
& 2\,D_{\text{mean}}(X,\,Y) - D_{\text{mean}}(X,\,X) - D_{\text{mean}}(Y,\,Y),
\end{aligned}
$$

where the euclidean distance is assumed for D_{mean}.

Properties of hierarchical clustering

- Any distance measure can be used.
- We need only the distance matrix (not the data).
- No parameters.
- Efficiency:
  - Agglomerative: $O(n^3)$ in naïve approach, $O(n^2)$ SLINK-algorithm.
  - Divisive: $O(2^n)$ in naïve approach, $O(n^2)$ CLINK-algorithm.
  - In general, efficiency can be increased by avoiding unnecessary re-computation of distances.
- Resulting dendrogram offers alternative clusterings.
- Dendrogram needs to be analyzed.
- Cut off at *different* levels of dendrogram may be necessary to get comparable clusters.
- Outliers are fully incorporated.

Optimization based clustering

Idea:

1. Think over what a good clustering should look like.
2. Put your ideas into a measure *E* which assigns a goodness value to any partitioning of the data.
3. Use an optimization scheme to partition a data set such that the goodness measure is maximized.

Note we are not bound to any specific way of processing the data (like, e.g., hierarchical clustering).

Basic maximization algorithm

1. Initialization: Partition data somehow into clusters $C_1 \ldots C_m$.
2. Choose an example $\vec{x}$ at random, denote its cluster as $C(\vec{x})$.
3. Select a random target cluster C_i.
4. Compute the change of the goodness function:

$$\Delta E = E(\text{“}\vec{x} \text{ in } C_i\text{”}) - E(\text{“}\vec{x} \text{ in } C(\vec{x})\text{”})$$

5. If $\Delta E > 0$ Put $\vec{x}$ from $C(\vec{x})$ to C_i.

   else Put $\vec{x}$ from $C(\vec{x})$ to C_i with probability $\exp(\beta \Delta E)$.
6. If(stop condition) STOP.
7. Increase β.
8. Goto 2.

$\beta > 0$.

Basic maximization algorithm

- May be caught on local maxima.
- Dependence on initial partitioning.
- To escape local maxima, downhill steps are accepted with probability $\exp(\beta \Delta E)$.
- Initially small β allows frequent downhill steps.
- Increasing β makes downhill steps less likely until the process “freezes” (simulated annealing).

Autocorrelation matrices for optimization

An important family of goodness measures is based on the following autocorrelation matrices for a dataset $D = \{\vec{x}_1, \vec{x}_2, \ldots\}, \vec{x}_i \in \mathbb{R}^d$, assigned to clusters $C_1 \ldots C_n$:

$\vec{\mu} = 1/\lvert D\rvert \sum_{\vec{x}\in D} \vec{x}$,	mean of all data.
$\vec{\mu}_i = 1/\lvert C_i\rvert \sum_{\vec{x}\in C_i} \vec{x}$,	mean of cluster C_i.
$A = 1/\lvert D\rvert \sum_{\vec{x}\in D} (\vec{x} - \vec{\mu}) (\vec{x} - \vec{\mu})^T$,	autocorrelation matrix of all data.
$A_i = 1/\lvert C_i\rvert \sum_{\vec{x}\in C_i} (\vec{x} - \vec{\mu}_i) (\vec{x} - \vec{\mu}_i)^T$,	autocorrelation matrix of cluster C_i.
$W = 1/\lvert D\rvert \sum_{i=1\ldots n} \lvert C_i\rvert A_i$,	weighted average cluster autocorrelation matrix.
$B = 1/n \sum_{i=1\ldots n} (\vec{\mu}_i - \vec{\mu}) (\vec{\mu}_i - \vec{\mu})^T$,	autocorrelation matrix of cluster centers.

$$\boxed{A = B + W}$$

Optimization functions

Minimize

$$\mathrm{tr}(W) = \sum_{i=1\ldots d} \lambda_i(W)$$

where λ_i are the eigenvalues.

- W is the average of the cluster autocorrelation matrices.
- $\lambda_i(W)$ is a measure for the variance of the average cluster along one coordinate.
- $\mathrm{tr}(W)$ is a measure for the complete variance of the average cluster.
- Minimizing $\mathrm{tr}(W)$ favors small, round clusters.

Optimization functions

Minimize

$$\det(W) = \prod_{i=1\ldots d} \lambda_i(W) .$$

- The product of all eigenvalues is a measure for the volume of the average cluster.
- Minimizing the product leads to clusters which have small volum on average.
- This does not necessarily lead to compactness.
- The result might be clusters which are small in a single dimension and large in $d–1$ dimensions.
- As W is the average of the clusters, minimizing volume favors clusters of similar shape.

Optimization functions

Minimize

$$\prod_{i=1\ldots n} \det(A_i)^{|C_i|} .$$

- Individual treatment of clusters avoids favoring a identical shape of all clusters.
- Problem: For d dimensions, at least d+1 data per cluster are necessary (otherwise det is 0).

Optimization functions

Maximize

$$\mathrm{tr}(BW^{-1}) = \mathrm{tr}((A-W)W^{-1}) = \mathrm{tr}(AW^{-1}-\mathbf{1}).$$

- Small $\mathrm{tr}(W)$ yields small clusters.
- Large $\mathrm{tr}(B)$ yields big variance of cluster centers.

Clustering and compression

Overview:

1. How can clustering be used for data compression?
2. K-means algorithm:
   - Most well known cluster algorithm.
   - Only one parameter (number of clusters), so effect can be easily observed.
3. Compression example: Image compression by reduction of color quantization.

Clustering and compression

- Given: A data set $D = \{\vec{x}_1, \vec{x}_2, \ldots\}$ of d-dimensional vectors $\vec{x}_i \in \mathbb{R}^d$.
- To transmit the data over some channel, the number of bits per data point depends on d and the required precision (and the statistics of the distribution).
- For a compressed transmission, a cluster algorithm yields a number $K < |D|$ of cluster centers $\vec{w}_j \in \mathbb{R}^d$ (also called *nodes*, *reference vectors* or *codewords*).
- A data vector $\vec{x}_i$ can now be approximated by its *best match* cluster center $\vec{w}_m$, where

$$m(\vec{x}_i) = \arg\min_j ||\vec{x}_i - \vec{w}_j||.$$

- So we have to transmit $\{\vec{w}_j\}$ once and after that only the number of the best match cluster center.
  - Small K : High compression ratio, bad approximation.
  - Large K : Low compression ratio, good approximation.

K-means clustering

K-means clustering (McQueen 1965):

- Divides D into clusters $C_1 \ldots C_K$, which are represented by their K centers of gravity (means) $\vec{w}_1 \ldots \vec{w}_K$.
- Minimizes the quadratic error measure

$$E(D, \{\vec{w}_i\}) = 1/|D| \sum_{i=1\ldots|D|} ||\vec{x}_i - \vec{w}_{m(\vec{x}_i)}||^2$$

- Iterative K-means clustering:
  - Start with randomly chosen reference vectors.
  - Assign all of the data to best match reference vectors.
  - Update reference vectors by shifting them to the mean of their cluster.
  - Stop if cluster centers have moved no more than ε, repeat otherwise.
- Usually only a local minimum of E can be found $\rightarrow$ clustering is not unique.

Iterative K-means clustering

```
1.  $t \leftarrow 0$
2.  Start with $K$ reference vectors $\vec{w}_1(0) \ldots \vec{w}_K(0)$ chosen randomly within a
    suitable bounding box in $\mathbb{R}^d$ .
3.  for $k \leftarrow 1 \ldots K$  begin $C_k \leftarrow \{\}$ endfor
4.  for $i \leftarrow 1 \ldots |D|$  begin
        $k^* \leftarrow \mathrm{argmin}_{k=1 \ldots K} ||\vec{x}_i - \vec{w}_k(t)||$
        $C_{k^*} \leftarrow C_{k^*} \cup \{\vec{x}_i\}$.
    endfor
5.  $t \leftarrow t + 1$
6.  for $k \leftarrow 1 \ldots K$ begin
        $\vec{w}_k(t) \leftarrow 1/|C_k| \sum_{\vec{x}_i \in C_k} \vec{x}_i$
    endfor
7.  if $(\exists\ k \in [1,K]:\ ||\vec{w}_k(t) - \vec{w}_k(t-1)|| > \varepsilon)$ then
        goto 3
    endif
```

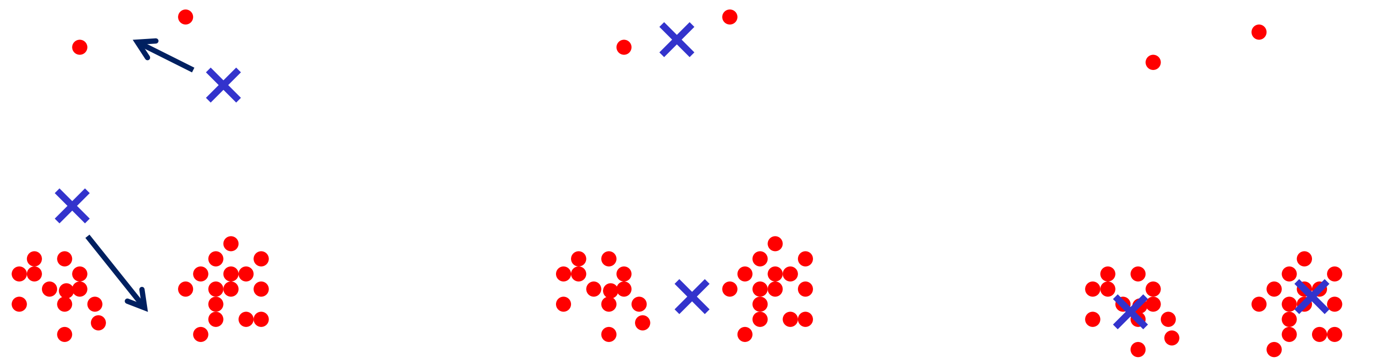
Properties of k-means

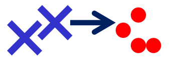
- Only one parameter: # clusters
  - implicitly defines scale and
  - resulting shape of clusters.
- Greedy optimization → local optima, depending on initial conditions:

How can different initial conditions result
in the selection of different optima?

Properties of k-means

- Only one parameter: # clusters
  - implicitly defines scale and
  - resulting shape of clusters.
- Greedy optimization → local optima, depending on initial conditions:


Initial conditions Resulting local optimum Global optimum [H]

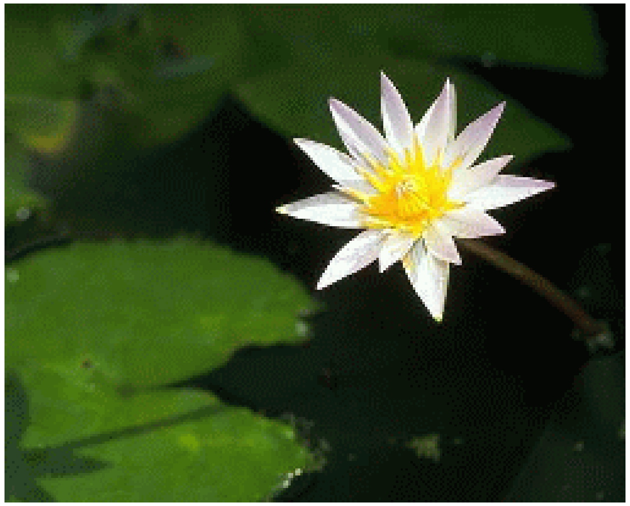
- Worst case is an empty cluster (idle node):  yields
- Fast algorithm

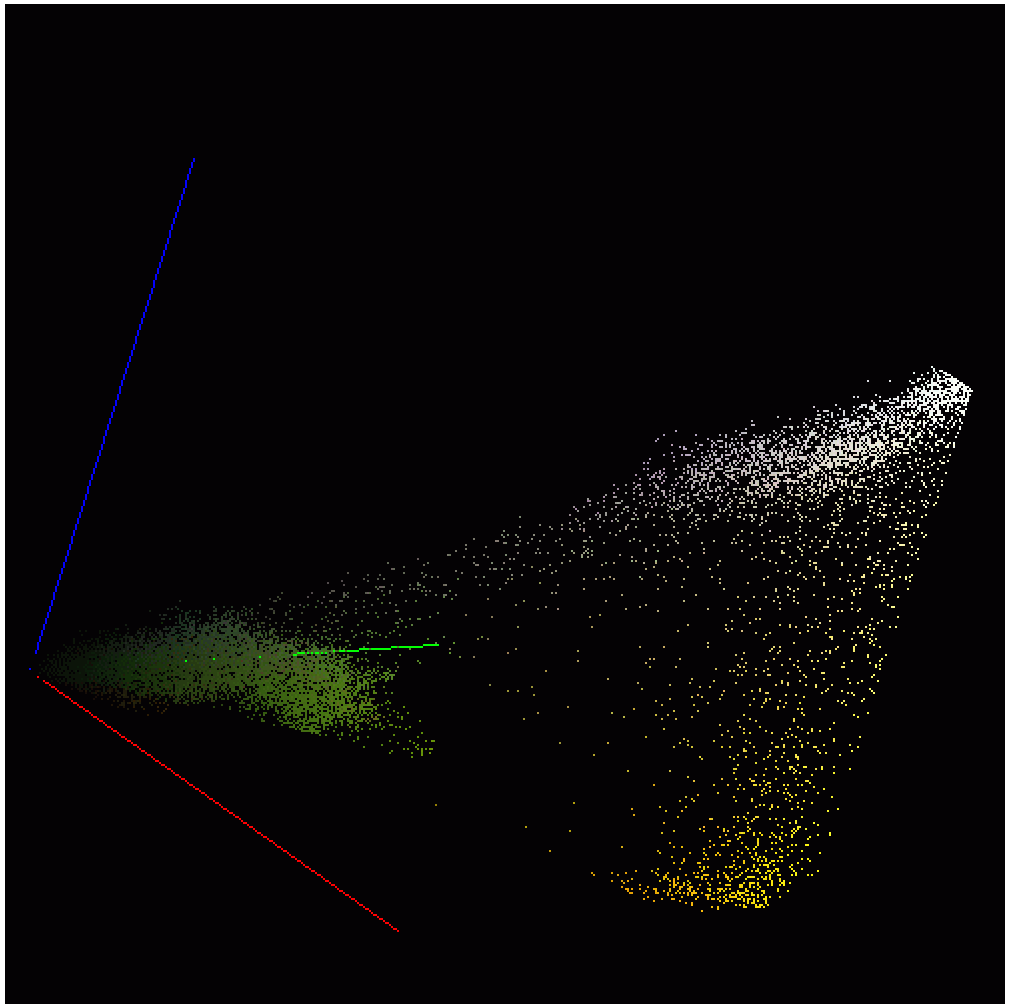
K-means clustering for color compression

Application to image compression:

- In a digital image, each pixel has three color values (red, green, blue), usually encoded by 3x8 bit (without compression).
- An easy way to achieve compression is reducing the $2^{3x8} \approx 16{,}7$ million colors to a smaller number of K prototypic colors. Each color of the original is replaced by its best match prototypic color.
- To obtain the prototypic colors, clustering can be used.
- A data vector $\vec{x}_i \in \mathbb{R}^3$ is the color triple of pixel number i.
- The cluster centers $\vec{w}_i \in \mathbb{R}^3$ are the prototypic colors.
- The result of K-means depends on the initial conditions, so different random initializations may lead to different compressed representations.

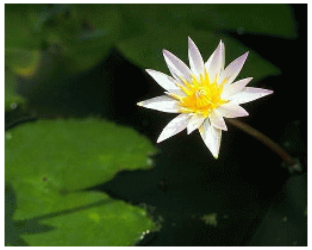
Image in color space


[A]

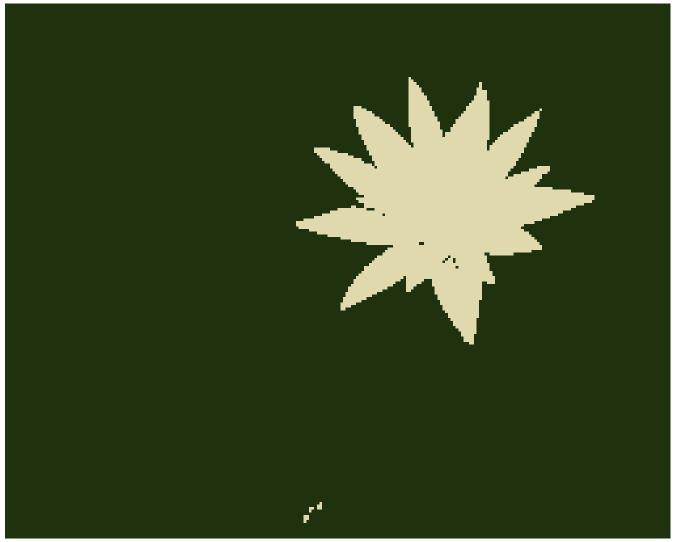

[H]

Each pixel of the image is visualized in RGB space in its color.

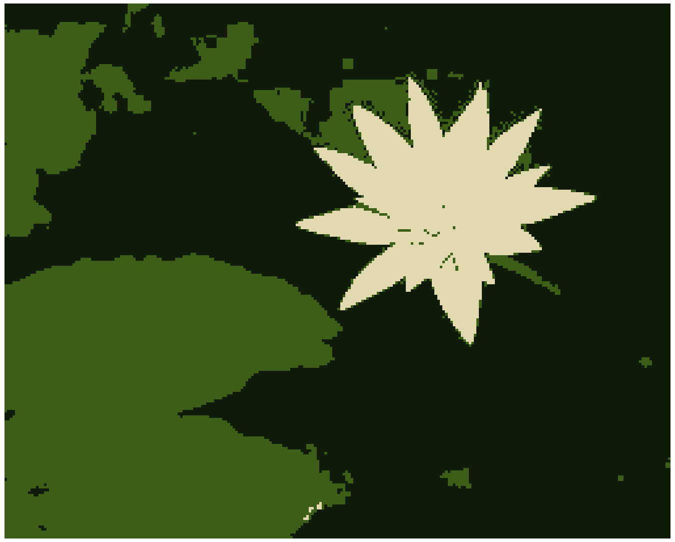
K-means clustering: Color compression



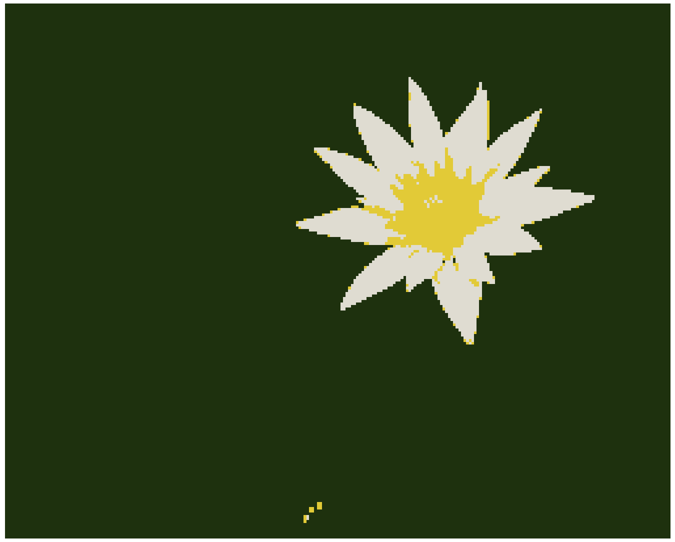
$K = 1$



$K = 2$



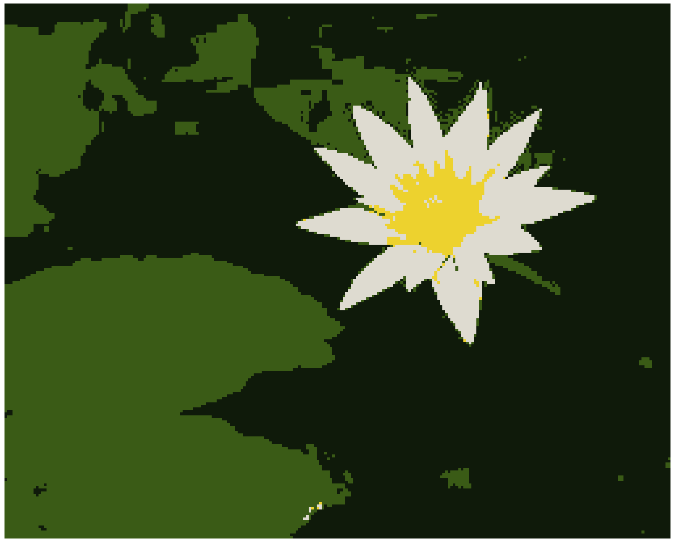
$K = 3$



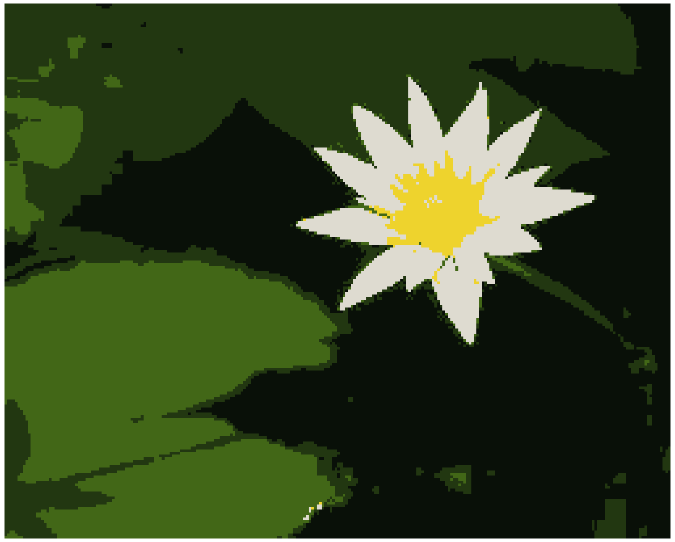
[H]

$K = 3$ (another local minimum)

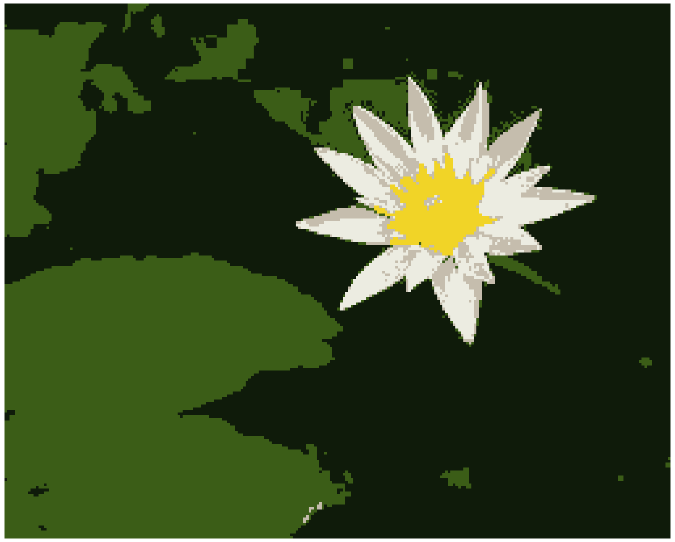
K-means clustering: Color compression



K = 4



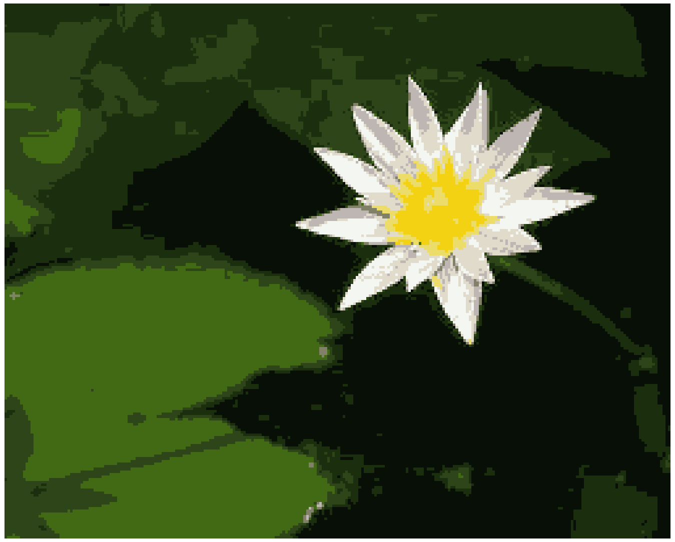
K = 5



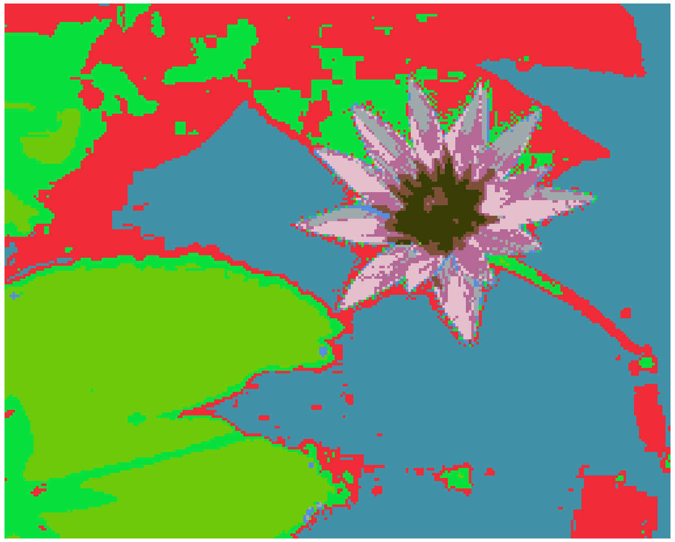
[H]

K = 5 (another local minimum)

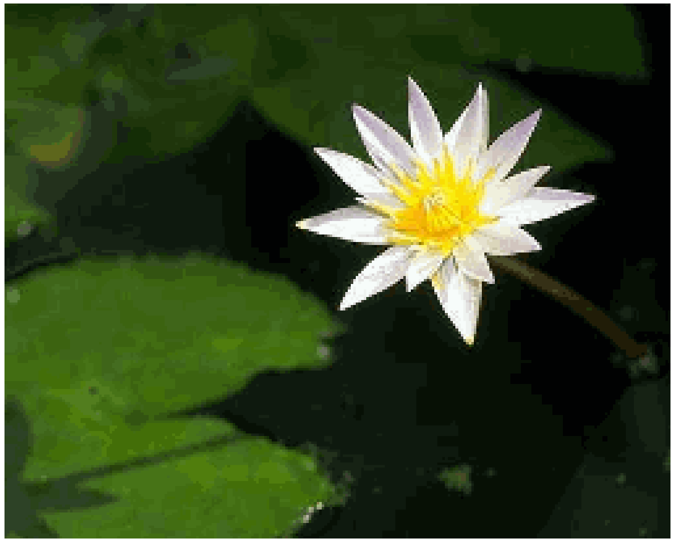
K-means clustering: Color compression



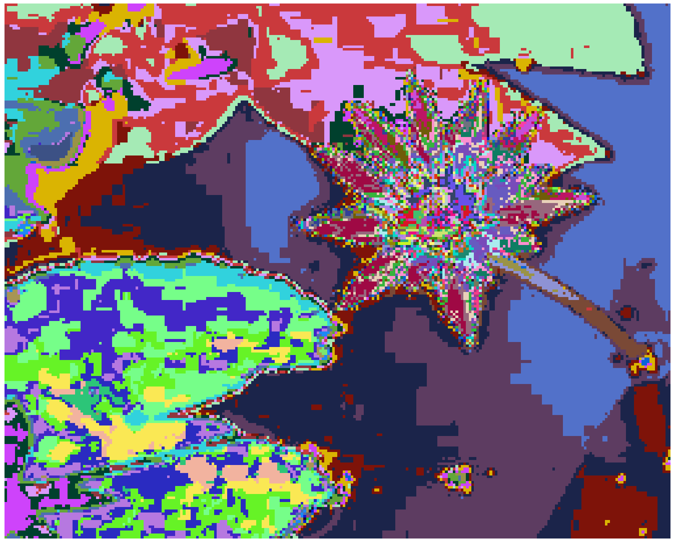
K = 10



Same as left, pseudocolors



K = 100



[H]

Same as left, pseudocolors

Soft clustering

So far:

- Clusters were described as sets of data points or by their centers → clusters are disjoint.
- This is called ***hard clustering***.
- Each data point is assigned to *a single* cluster.
- No way to express uncertainty about the assignment to a cluster.

Now:

- Describe data by probability distribution $P(\vec{x})$.
- ***Soft clustering***:
  - A data point is assigned to the clusters by probabilities (expressing uncertainty about the assignment or gradual assignment).
  - As a result, clusters do not have hard boundaries.

Mixture of Gaussians

Idea:

- Representing clusters by their centers was simple and efficient.
- Generalization to soft clustering: Assign a Gaussian to each cluster center.

So the ansatz for the probability density of the data distribution $D = \{\vec{x}_1, \vec{x}_2, \ldots\}$, $\vec{x}_i \in \mathbb{R}^d$, is a linear superposition of K Gaussians:

$$P(\vec{x}) = \sum_{k=1\ldots K} g_k \, N(\vec{x}, \vec{\mu}_k, C_k),$$

where $N(.,.,.)$ is a Gaussian with mean $\vec{\mu}$ and covariance matrix C.

The “amplitude” assigned to a Gaussian centered at $\vec{\mu}_k$ is g_k, which is also the a priori probability that a data point belongs to cluster k.

So $0 \le g_k \le 1$ and $\sum_{k=1\ldots K} g_k = 1$ must hold.

Mixture of Gaussians

Suppose you want to *generate* a data point according to

$$P(\vec{x}) = \sum_{k=1\ldots K} g_k\, N(\vec{x}, \vec{\mu}_k, C_k),$$

then there are two ways to do this:

- Regard $P(\vec{x})$ as a whole.
- First select one of the Gaussians with probability g_k, then generate a random $\vec{x}$ with probability $N(\vec{x}, \vec{\mu}_k, C_k)$.

Here we take the latter point of view:

Each Gaussian describes a separate cluster, but data assignment is represented by probabilities.

Mixture of Gaussians

The probability that an example drawn at random from D is in volume V in data space is

$$P(\vec{x} \in V) = \int_V P(\vec{x})\, d\vec{x}.$$

The prior (a priori probability) that an example drawn at random from D belongs to cluster k is g_k.

The a posteriori probability that a given data point $\vec{x}$ belongs to cluster k is

$$P^*_k(\vec{x}) = g_k\, N(\vec{x}, \vec{\mu}_k, C_k) \,/\, \sum_{i=1\ldots K} g_i\, N(\vec{x}, \vec{\mu}_i, C_i).$$

To find the best mixture of K Gaussians to fit a given data set D, the parameters $\{g_k, \vec{\mu}_k, C_k\}$ must be found by the EM-algorithm. The derivation of the procedure is left out here because the constraint $\sum_{k=1\ldots K} g_k = 1$ requires the Lagrange multiplier method.

EM for a mixture of Gaussians

1. Choose number of Gaussians K.
2. Step counter $t = 0$.
3. Choose initial values $\{g_k(0), \vec{\mu}_k(0), C_k(0)\}$.
4. E-step: A posteriori probabilities based on current parameters are

$$P^*_k(t+1,\vec{x}) = g_k(t)\, N(\vec{x}, \vec{\mu}_k(t), C_k(t)) \,/\, \sum_{i=1\ldots K} g_i(t)\, N(\vec{x}, \vec{\mu}_i(t), C_i(t)).$$

5. M-step: Improve parameter estimates using the new a posteriori probabilities:

$$N_k = \sum_{i=1\ldots|D|} P^*_k(t+1,\vec{x}_i),$$

$$g_k(t+1) = N_k/|D|,$$

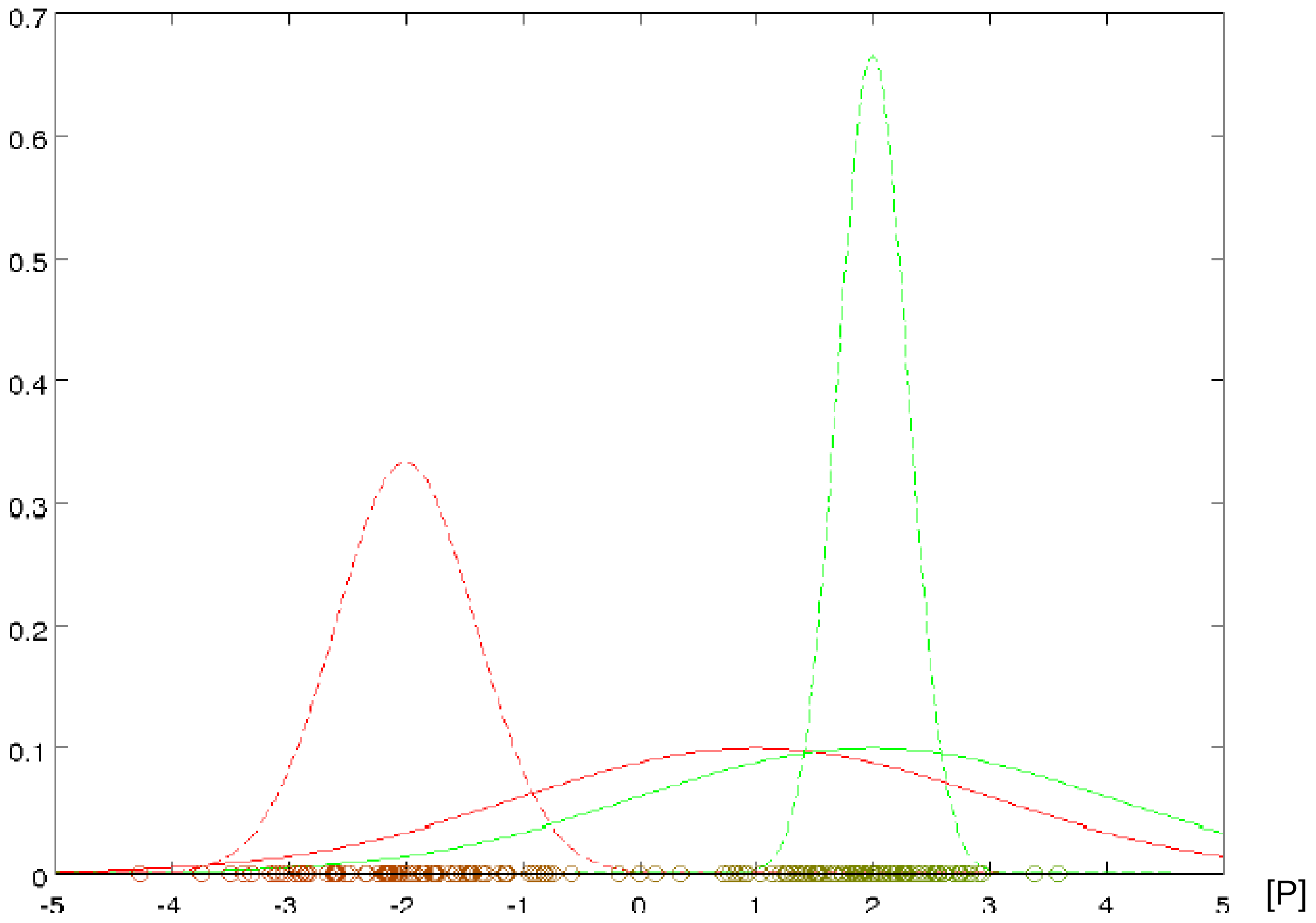
$$\vec{\mu}_k(t+1) = 1/N_k \sum_{i=1\ldots|D|} P^*_k(t+1,\vec{x}_i)\, \vec{x}_i,$$

$$C_k(t+1) = 1/N_k \sum_{i=1\ldots|D|} P^*_k(t+1,\vec{x}_i)\, (\vec{x}_i - \vec{\mu}_k(t+1))\, (\vec{x}_i - \vec{\mu}_k(t+1))^T$$

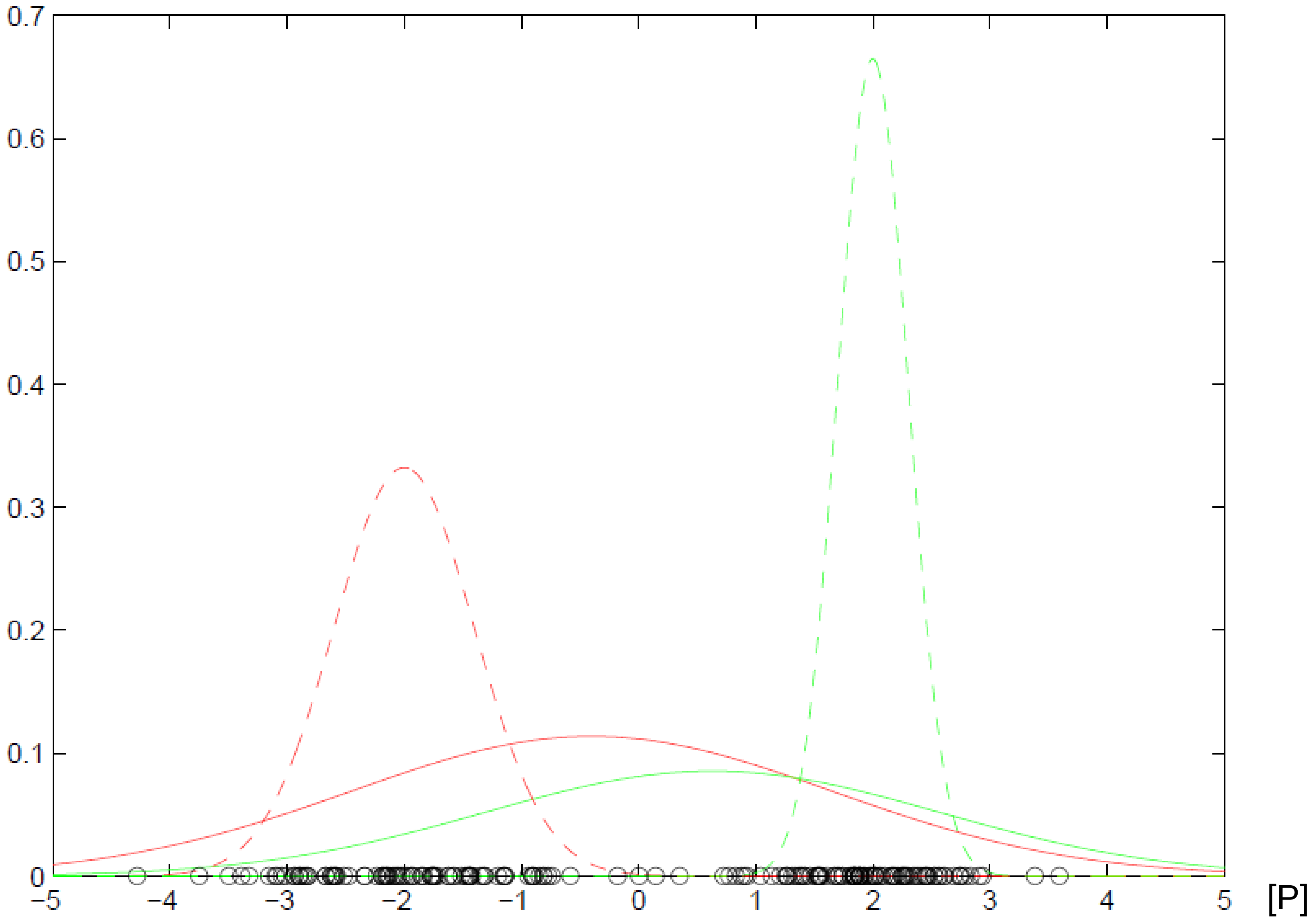
6. If (stop condition) STOP else t++; goto 4.

UNIVERSITÄT OSNABRÜCK

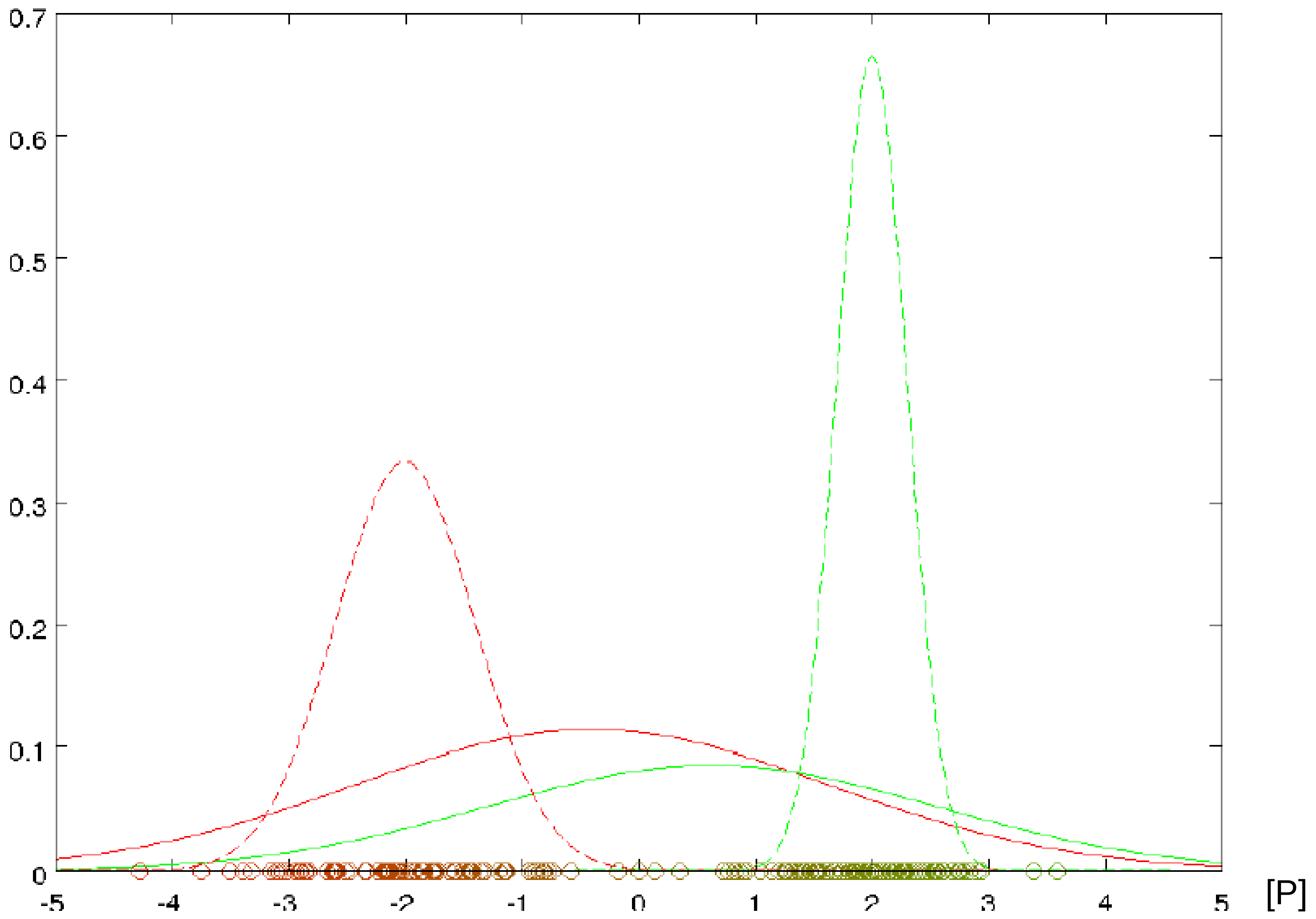
EM-Example: E-Step 1



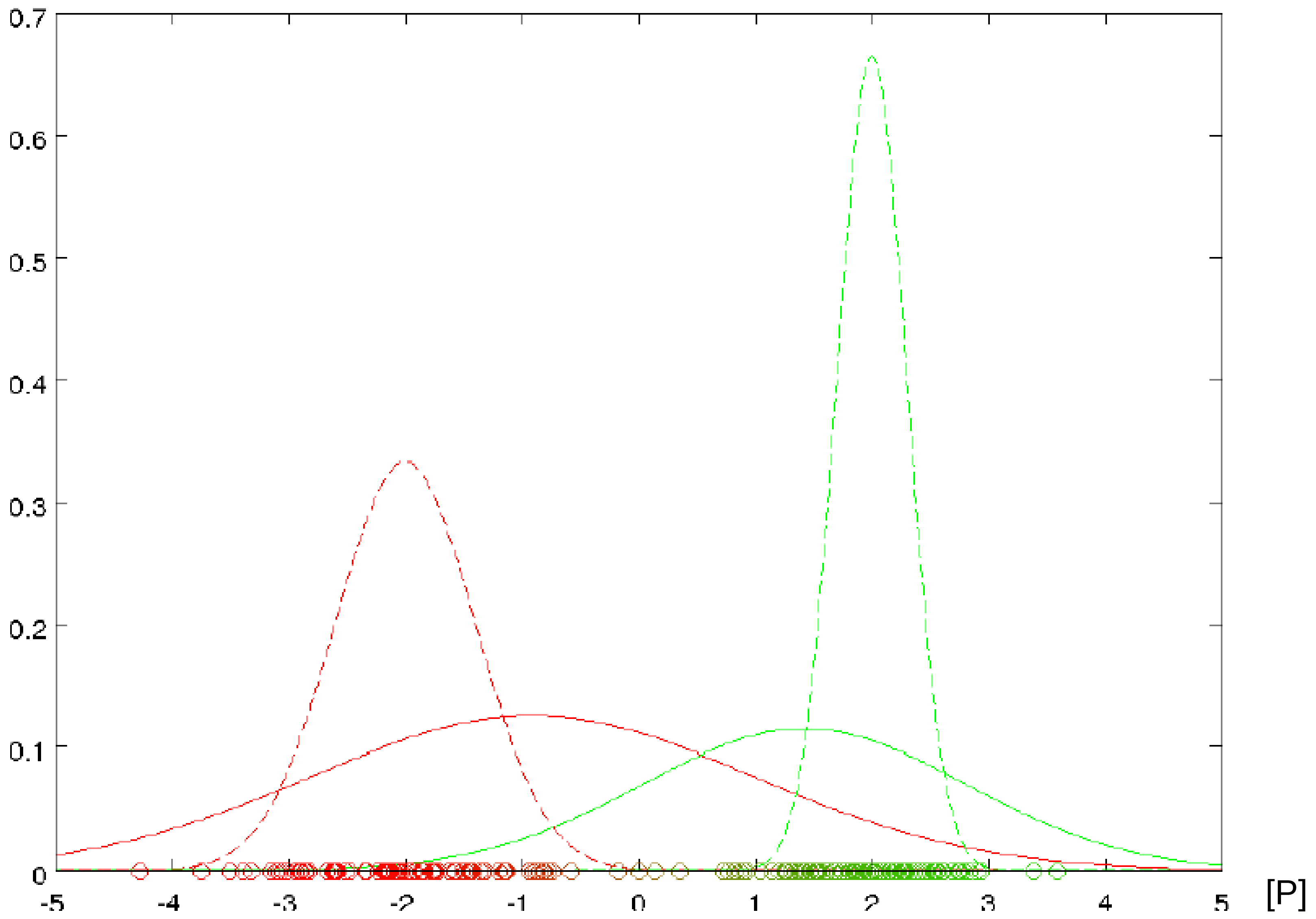
EM-Example: M-Step 1



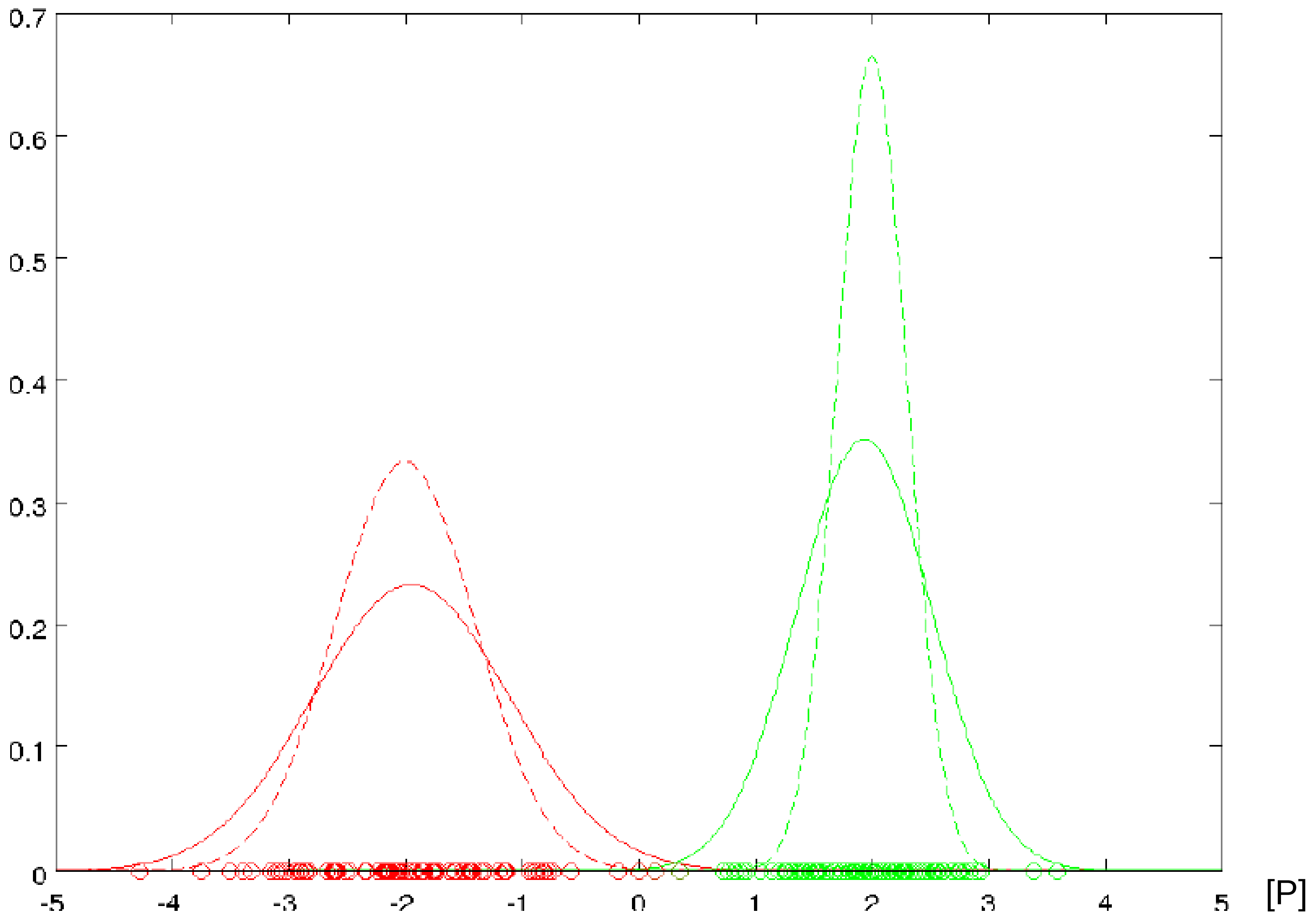
EM-Example: E-Step 2



EM-Example: E-Step 10



EM-Example: E-Step 20



Mixture of Gaussians

- EM algorithm yields only local optimum.
- Computationally much more expensive than K-means.
- Precautions against collapse of a Gaussian on a single data point necessary.
- K-means can provide useful initialization for $\vec{\mu}_k$, local PCA for C_k.
- There are $K!$ equivalent solutions. This is no problem for computation, but parameter identification may be a problem.

Evaluation of clusterings

There is no general method to decide whether an achieved clustering is good. These tests may help:

- Try on different data subsets.
- Check distribution of averaged distances in *k*-neighbor cluster.
- Compare *intra-cluster* distances (distances of data to cluster center) and *inter-cluster* distances (distances between cluster centers).
- Compare two clusterings:

(# pairs which are separated or together in both clusterings)

/

(# all pairs)

= probability that a randomly chosen pair is treated the same way in both clusterings.

Conceptual Clustering

Conceptual clustering

Supervised classification:

- Pre-defined classes
- Example set of pairs (*object, class*).

Unsupervised classification:

- Classes are not fixed a priori.
- Classes (also: *categories, concepts*) are formed from the examples.
- Examples are sorted into the formed categories.
- Bias of the system lies in the preferences of category formation.

Conceptual clustering:

Employs ideas of clustering and decision tree learning for unsupervised classification / categorization.

COBWEB: Motivation

COBWEB (Fisher 1987) is the most well known algorithm for conceptual clustering.

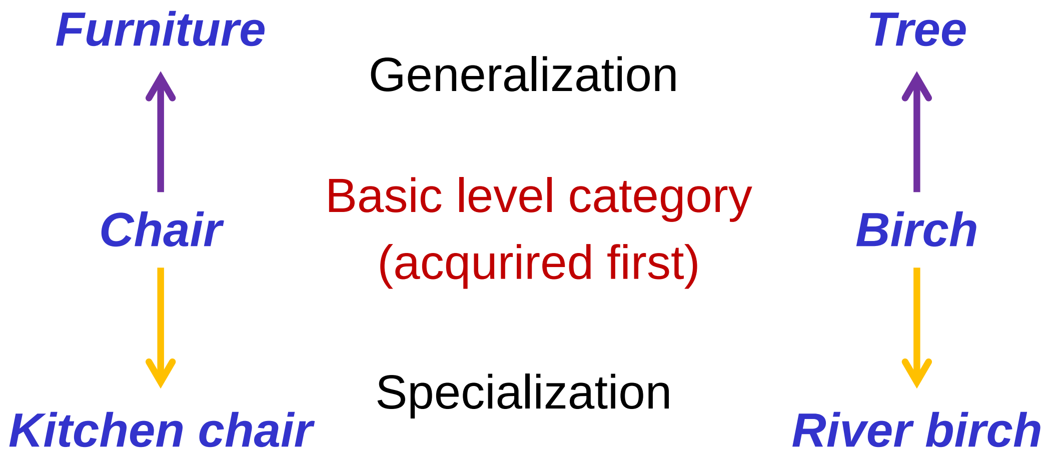
Motivation partly from drawbacks of ID3:

- Continuous attributes require thresholding.
- No flexibility in case of errors.
- Disjoint learning phase (building the tree) and application phase (classifying data) are unnatural.
- Each learning step divides data only along *one* dimension of the attribute space.
- Defines categories by propositional logic.

Aspects of human categorization

Eleanor Rosch, *Principles of Categorization*, 1978:

- Category formation is strongly connected to forming prototypical concepts :
  - *Robin* is a more typical bird than a *penguin*.
  - Depends on learner's context – consider "*Birch* is a more typical tree than *palm*."
- *Basic level categories* :



Aspects of human categorization

Family resemblance theory (Wittgenstein 1953):

- Categories definition is based on similarity in a complex way …
- … not by hard necessary / sufficient conditions.
- Example: *Game*
  - not all games have several players
  - not all games are for fun
  - not all games have rules
  - not all games are in competition

COBWEB

Ideas:

- Unsupervised learning.
- Incremental learning, no separation of training and test phase.
- Probabilistic representation: Gradual assignment of objects to categories.
- No a priori fixed number of categories.

Realization by global utility function which determines

- number of categories,
- number of hierarchy levels,
- assignment of objects to categories.

COBWEB

Global utility function for categories $C_1 \ldots C_N$, attributes A_i with values v_{ij}:

$$S = 1/N \sum_{n=1\ldots N} \sum_{i,j} P(A_i = v_{ij}) \cdot P(A_i = v_{ij} \mid C_n) \cdot P(C_n \mid A_i = v_{ij})$$

Interpretation:

$1/N$:	Prefers few categories.
$P(A_i = v_{ij} \mid C_n)$:	Predictability – probability that an object of category C_n has value v_{ij} for attribute A_i = average number of correctly predicted values v_{ij} for attribute A_i if you know it's category C_n. Alternative interpretation: *Intra-category-similarity*.
$P(C_n \mid A_i = v_{ij})$:	Predictiveness – probability, that an object with value v_{ij} for attribute A_i belongs to category C_n. Alternative interpretation: *Inter-category-dissimilarity*.
$P(A_i = v_{ij})$:	Stronger weighting of frequent attribute values.

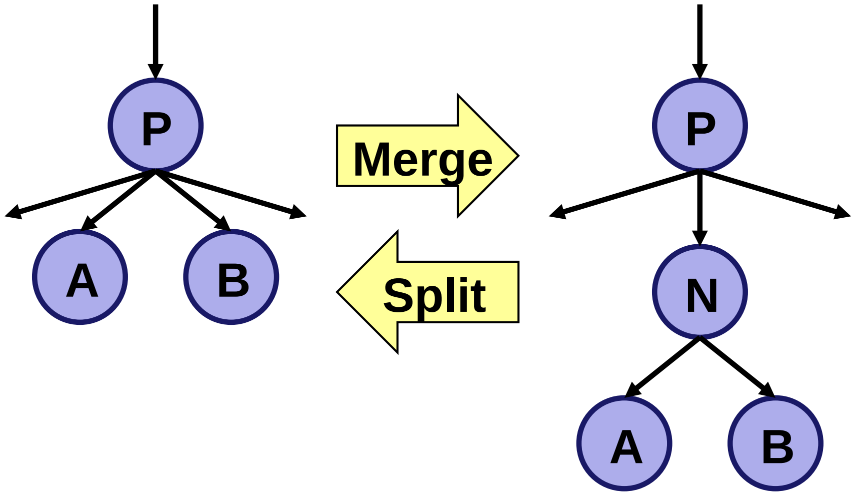
COBWEB

Learning:

$$\text{New example} \begin{cases} \nearrow \text{Insert into tree} \\ \searrow \text{Modify tree} \end{cases} \Bigg\} \text{ such that } S \text{ is maximized!}$$

Options for modification:

1. Create new terminal node.
2. Merge two nodes.
3. Split a node.



[H]

Summary

- Clustering is one of the most basic techniques in ML.
- Clustering is a simple means to achieve compression.
- Clustering requires definition of clusters (scale, shape).
- Most algorithms have an implicit definition of clusters, resulting from the way the algorithm works.
- Adjustable parameters are usually parameters of the algorithm, not of the clusters.
- Hierarchical clustering offers a choice among different clusterings, but the tree requires analysis.
- Optimization based clustering defines global measures for good clusters and good distribution of clusters.
- Optimization measures often aim at minimizing the intra-cluster variance and / or maximizing the inter-cluster variance.

Summary

- K-means is the most well known cluster algorithm derived from an optimization measure.
- Like other optimization procedures, clustering has the problem of local optima.
- Soft clustering avoids a strict assignment of data to clusters.
- Conceptual clustering allows concept formation by building a generalization of decision trees.

Image sources

[M] Online material available at www.cs.cmu.edu/~tom/mlbook.html for the textbook: Tom M. Mitchell: *Machine Learning*, McGraw-Hill

[A] *Artexplosion Explosion*® *Photo Gallery*, Nova Development Corporation, 23801 Calabasas Road, Suite 2005 Calabasas, California 91302-1547, USA.

[H] Gunther Heidemann, 2012.

[P] Michael Pardowitz, 2014.