Human-Oriented Robotics

Unsupervised Learning

Kai Arras

Social Robotics Lab, University of Freiburg

Winter term 2014/2015

Unsupervised Learning

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

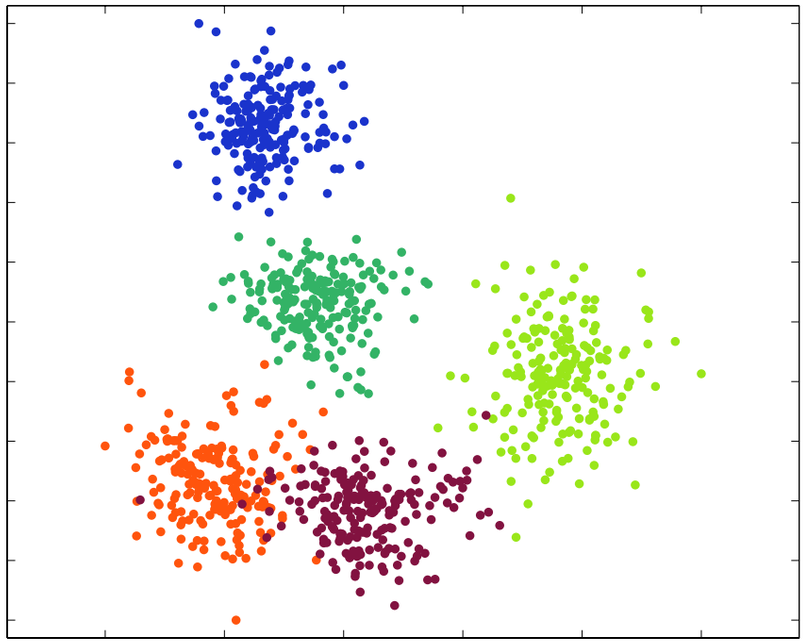
Contents

- Introduction
- Hierarchical Clustering
- K-Means
- Gaussian Mixture Models

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Introduction

- In **unsupervised** learning, data vectors $\mathbf{x}$ have **no class labels**

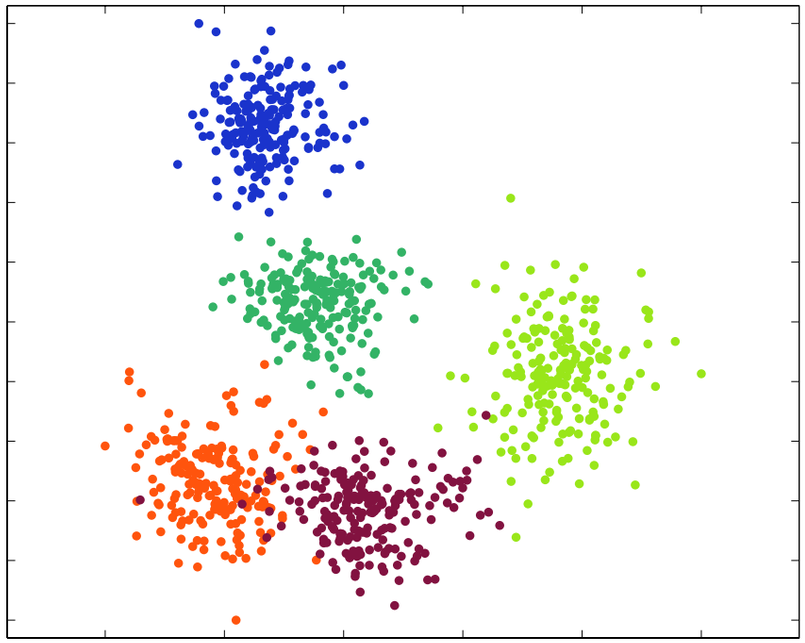


supervised learning

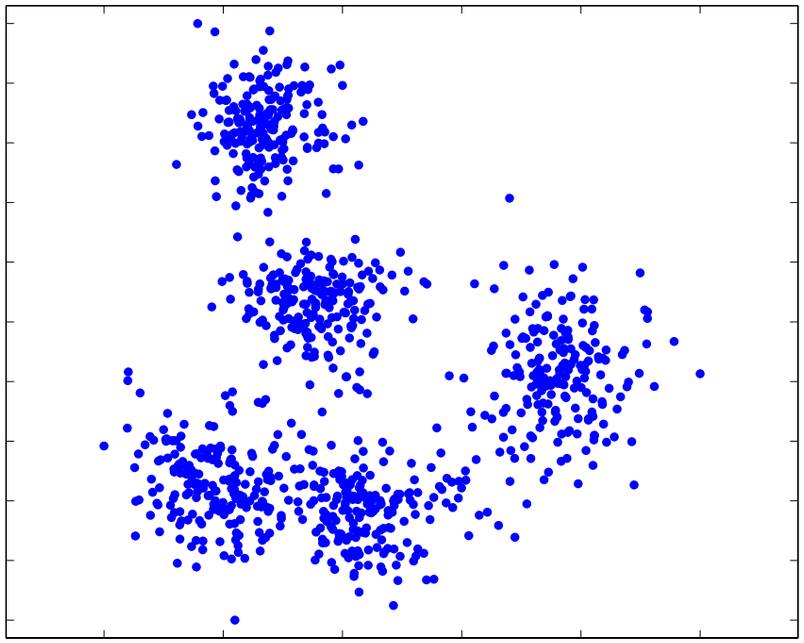
- The challenge is to find **hidden structures** in unlabeled data
- Approaches to unsupervised learning include clustering, outlier detection, density estimation, dimensionality reduction

Introduction

- In **unsupervised** learning, data vectors $\mathbf{x}$ have **no class labels**



supervised learning



unsupervised learning

- The challenge is to find **hidden structures** in unlabeled data
- Approaches to unsupervised learning include clustering, outlier detection, density estimation, dimensionality reduction

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Introduction

- **Clustering** is a set of techniques for organizing objects in such a way that objects in the **same group** are more similar to each other than to those in **other groups**
- This task is called **cluster analysis** and groups are called **clusters**
- Clustering requires the following components and steps
  1. Selection of features
  2. Similarity measure
  3. Clustering criterion
  4. Clustering algorithm
  5. Validation of the results
- **Applications:** data mining, big data, web science (e.g. social network analysis), computational biology, computer vision (e.g. image segmentation), robotics (e.g. finding modes in probability distributions)

Introduction

- Cluster analysis components and steps:

  1. **Selection of features**. As was the case with supervised learning, we assume that data are represented in terms of attributes or features, which form m-dimensional vectors $\mathbf{x}$. These features must be properly selected so as to encode as much information as possible concerning the task of interest. Preprocessing the features (e.g. scaling, whitening, PCA whitening etc.) may be necessary

  2. **Similarity measure**. The measure quantifies how similar or "close" two feature vectors are. It is assumed that all selected features contribute equally to the computation of the proximity measure and there are no features that dominate others

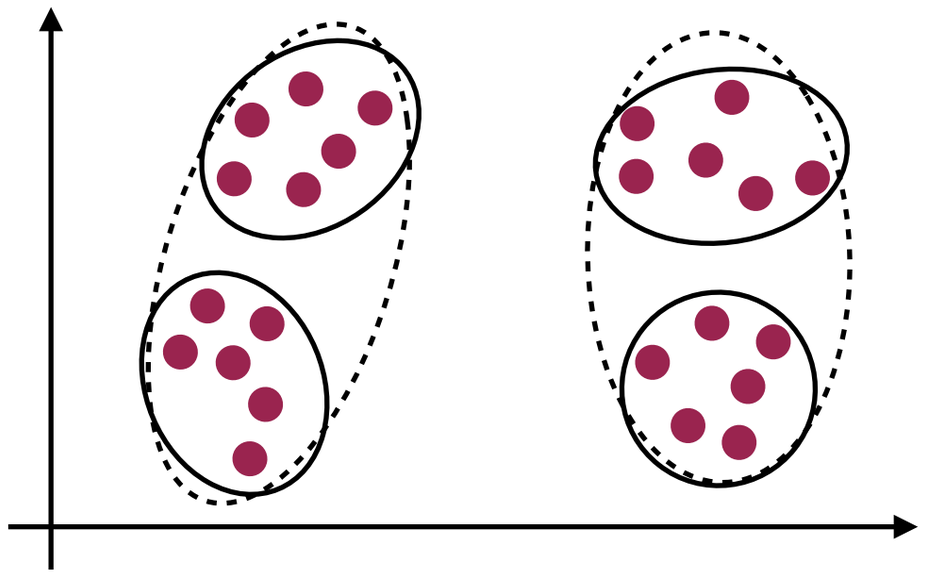
Introduction

- Cluster analysis components and steps (cont.):

3. **Clustering criterion.** The organization of data into clusters depends on task-relevant criteria. Animals, for example, are grouped differently if the criterion is the existence of lungs or the environment they live (water, air, land). People can be grouped into friends, family, colleagues, members of a theatre audience or combinations thereof. The criterion may be expressed via a cost function
4. **Clustering algorithm.** Based on a similarity measure and a criterion, the specific algorithm that unravels the hidden structures in the data
5. **Validation of the results.** Like in supervised learning, the validity of the obtained result is verified using appropriate tests

Introduction

- Different choices of similarity measures, clustering criteria or clustering algorithms may lead to **totally different** clustering results
- Which clustering is "correct"? To a certain extent, **subjectivity** plays a role



- We now consider the three most popular clustering methods: **hierarchical clustering**, **k-means**, and **Gaussian mixture models**
- Let us introduce some **notation** common to those methods:
  Let $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ be a data set consisting of N observations, each of dimension m. Our goal is to partition the data into K clusters

Hierarchical Clustering

- Hierarchical clustering is a method of cluster analysis which seeks to build a **hierarchy of clusters**. Algorithms generally fall into two categories:
  - **Agglomerative**: a "bottom up" approach in which each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy
  - **Divisive**: a "top down" approach in which all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy
- We will consider the agglomerative approach. Divisive methods are more expensive and rarely used in practice
- Let $\mathcal{R} = \{\mathcal{C}_i\}_{i=1}^{K}$ be a clustering, that is, the partition of $\mathcal{D}$ into K non-empty sets $\mathcal{C}_i$ (clusters) such that $\cup_{i=1}^{K} \mathcal{C}_i = \mathcal{R}$ (exhaustive) and $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset \;\; \forall k, l \in \{1, \ldots, N\}$ (mutually exclusive)

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Agglomerative Hierarchical Clustering (AHC)

- Set $\mathcal{R}_0 = \{\{\mathbf{x}_1\}, \{\mathbf{x}_2\}, \ldots, \{\mathbf{x}_N\}\}$ as the initial clustering and let $t = 0$

Repeat

1. Find the **closest pair of clusters**

$$d(\mathcal{C}_i, \mathcal{C}_j) = \min_{k,l} d(\mathcal{C}_k, \mathcal{C}_l) \quad \forall k, l \in \{1, \ldots, N\}$$

2. **Merge** $\mathcal{C}_m = \mathcal{C}_i \cup \mathcal{C}_i$
3. Produce **new clustering** $\mathcal{R}_{t+1} = (\mathcal{R}_t / \{\mathcal{C}_i, \mathcal{C}_j\}) \cup \mathcal{C}_m$
4. $t \leftarrow t + 1$

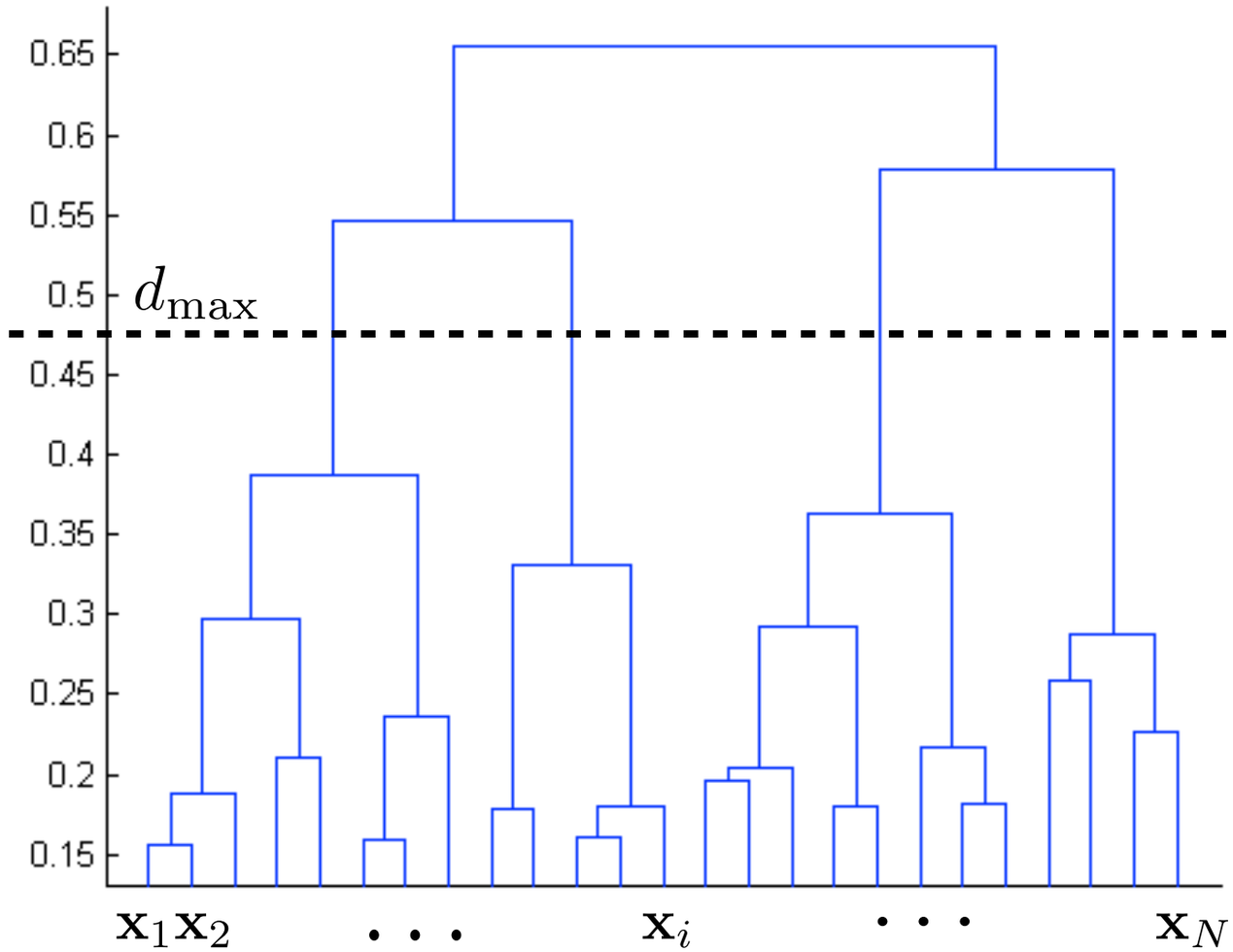
Until $|\mathcal{R}_{t+1}| = K$

- Alternative termination conditions: $d(\mathcal{C}_i, \mathcal{C}_j) \geq d_{\max}$ or $|\mathcal{R}_{t+1}| = 1$

Hierarchical Clustering

Dendrogram

- The result of hierarchical clustering can be drawn as a hierarchical structure known as **dendrogram**
- **Leaves** correspond to single data points
- The grouping of points is given by **the order they are merged**
- Can be **intersected at any level** to get the wanted number of clusters K or minimal similarity $d_{\max}$
- Merge decisions are **hard** and cannot be revised



Hierarchical Clustering

Similarity Measures

- In order to decide which clusters should be merged, we require both, a **similarity** (or dissimilarity/distance) **metric** between pairs of data points and a **linkage criterion** which specifies the **similarity** (or dissimilarity) **of clusters**
- For the former, distances are typically measured with a **Minkowski distance** or L^p-norm

$$d(\mathbf{x}, \mathbf{x}') = (\sum_{i=1}^{m} |x_i - x'_i|^p)^{1/p}$$

which is, for example, the **Euclidian distance** for $p = 2$, the **Manhattan** (taxicab) **distance** for $p = 1$, the maximum or **Chebyshev distance** for the case of p reaching infinity

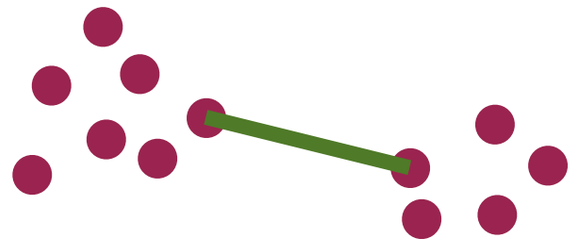
- Many distance metrics exist also for **discrete** or **non-numeric** data

Hierarchical Clustering

Linkage Criterion

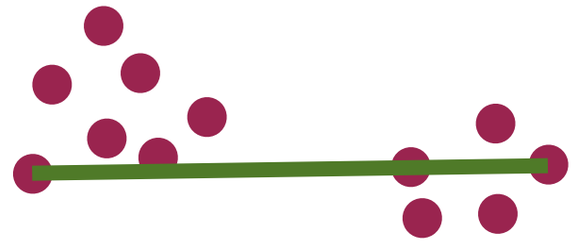
- The linkage criterion is a **similarity measure** $d(\mathcal{C}_i, \mathcal{C}_j)$ **between clusters** which, in turn, relies on the similarity measure between pairs of data points in the clusters. Among a **large variety** of criteria, the most common are:
- **Single-linkage**

$$d(\mathcal{C}_i, \mathcal{C}_j) = \min_{\mathbf{x}_k \in \mathcal{C}_i, \mathbf{x}_l \in \mathcal{C}_j} d(\mathbf{x}_k, \mathbf{x}_l)$$



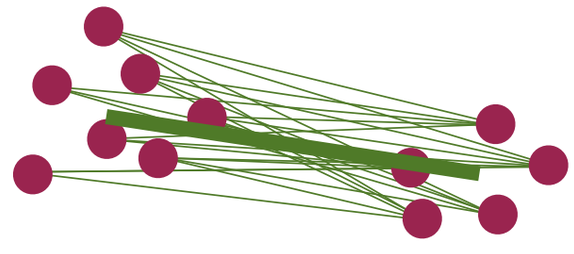
- **Complete-linkage**

$$d(\mathcal{C}_i, \mathcal{C}_j) = \max_{\mathbf{x}_k \in \mathcal{C}_i, \mathbf{x}_l \in \mathcal{C}_j} d(\mathbf{x}_k, \mathbf{x}_l)$$



- **Average-linkage**

$$d(\mathcal{C}_i, \mathcal{C}_j) = \frac{1}{|\mathcal{C}_i||\mathcal{C}_j|} \sum_{\mathbf{x}_k \in \mathcal{C}_i} \sum_{\mathbf{x}_l \in \mathcal{C}_j} d(\mathbf{x}_k, \mathbf{x}_l)$$



Hierarchical Clustering

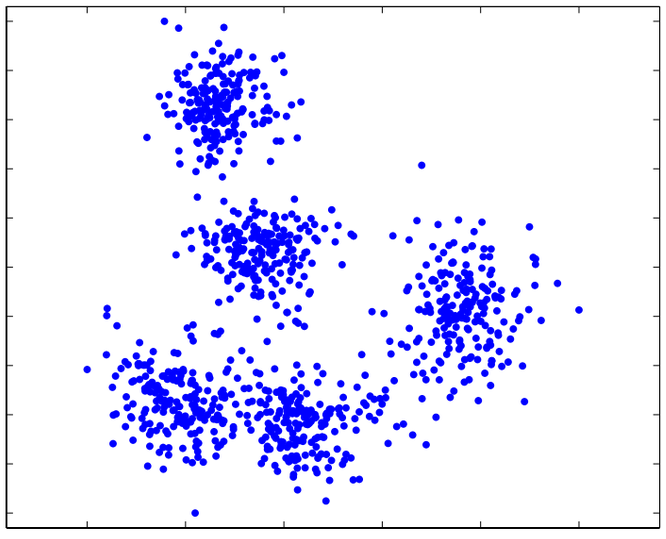
Properties

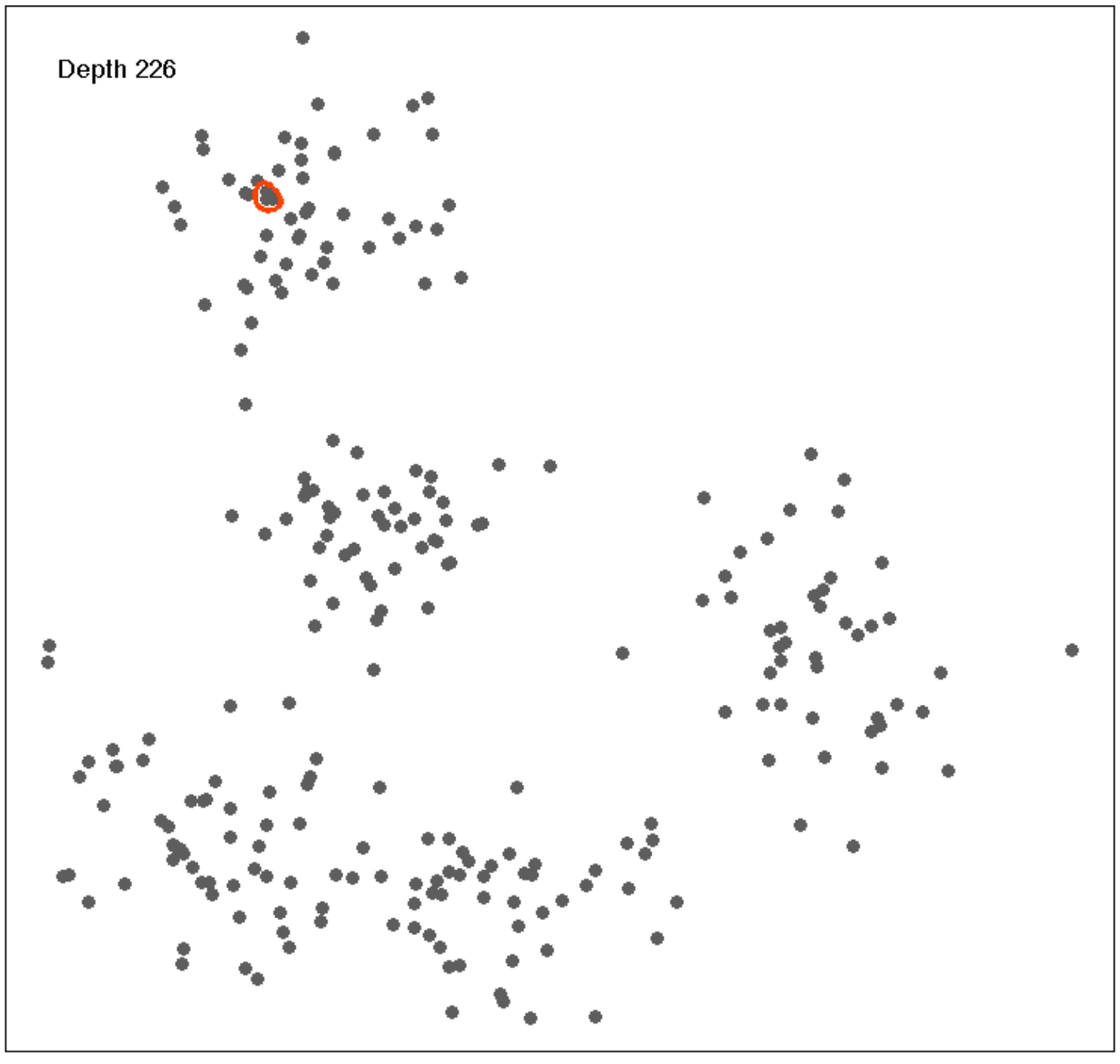
- Different **choices of similarity measures** for both pairs of points or pairs of clusters may lead to **totally different** clustering results
- Hierarchical clustering can use **any valid distance measure**: data points are never required on their own, they only **enter the algorithm in pairwise distances**. Thus, the methods can be readily applied to various data types (discrete, non-numeric, etc.)
- In some clustering tasks, it may be more natural to define a minimal similarity $d_{\max}$, in other tasks K is easy to define. Hierarchical clustering allows to terminate with **both criteria**
- For an **implementation**, it is typical to maintain a **distance matrix**, where the number in the i-th row j-th column is the distance between the i-th and j-th data points. Then, as clustering progresses, rows and columns are merged as the clusters are merged and the distances updated

Hierarchical Clustering

Examples

- Shows the bottom-up progression of AHC
- Only clusters with $|\mathcal{C}| > 2$ are highlighted

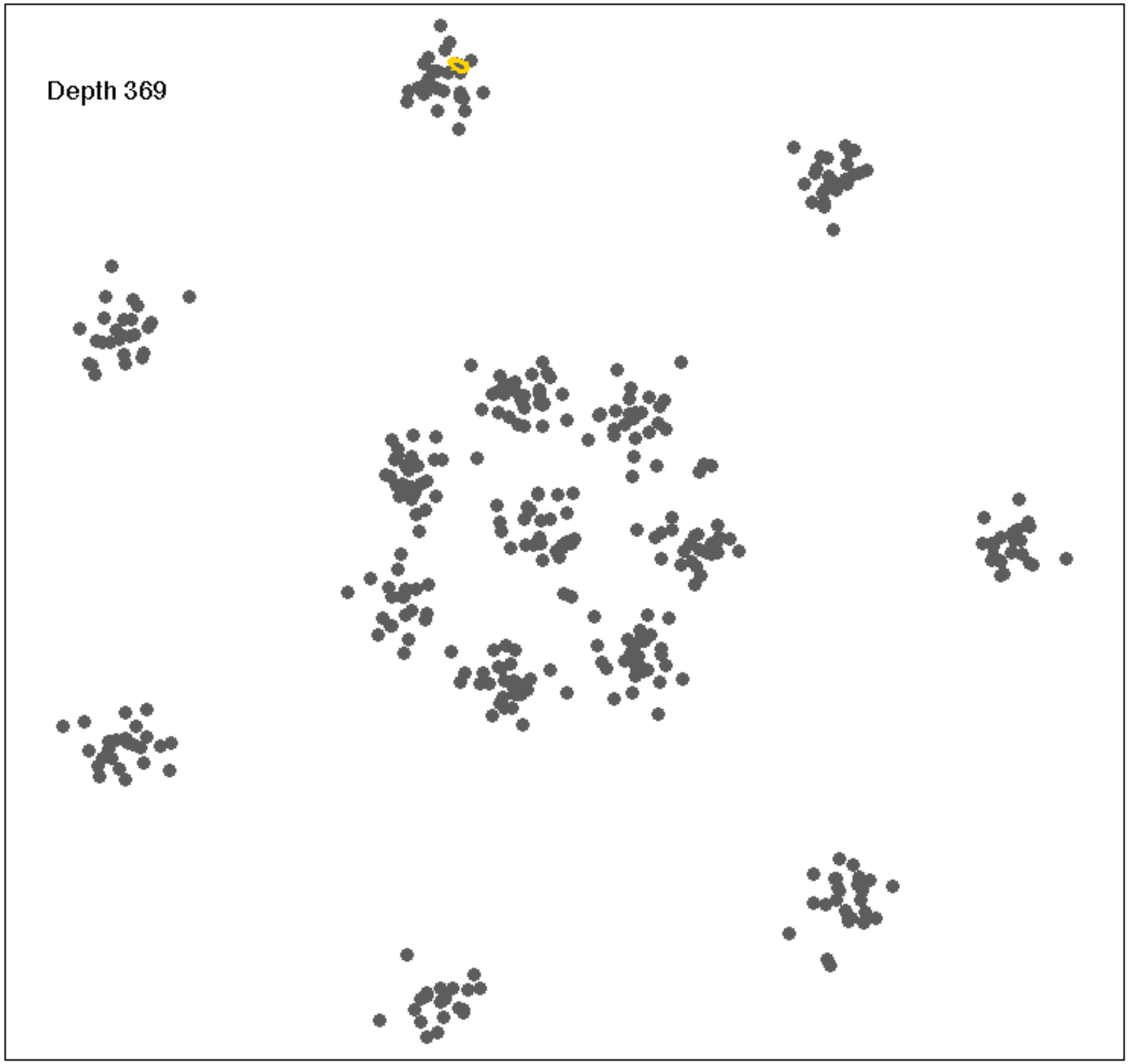




Hierarchical Clustering

Examples

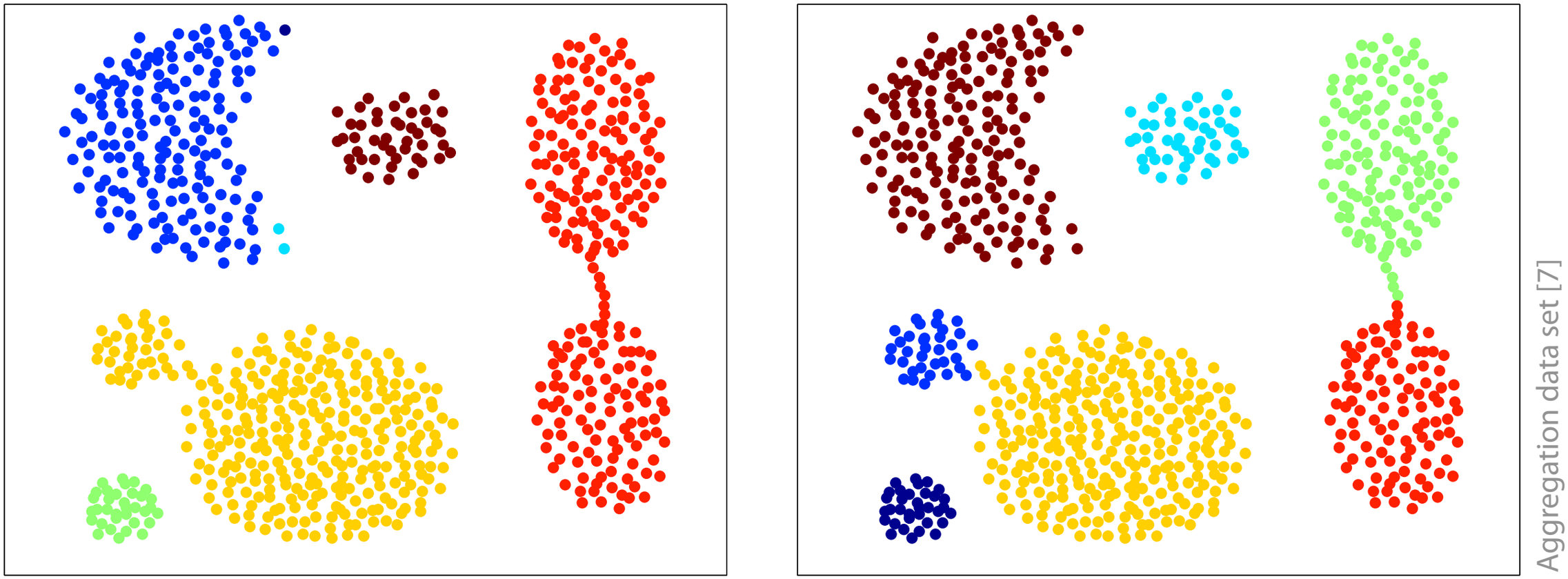
- Shows the bottom-up progression of AHC
- Termination at $K = 15$
- Only clusters with $|\mathcal{C}| > 2$ are high-lighted



R15 data set [7]

Examples

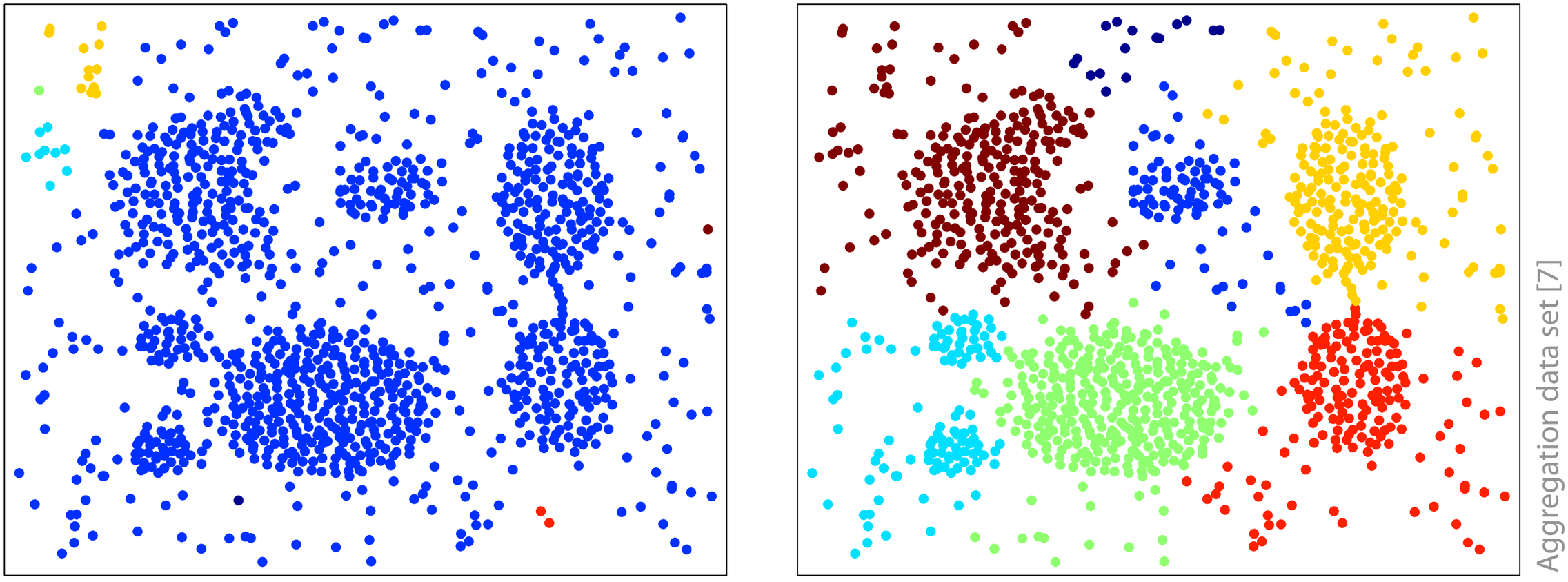
- **Single** linkage (left) vs. **average** linkage (right), $K = 7$



- Single linkage is able to recover **elongated clusters** but undersegments
- **Complete** linkage (not shown) tends to oversegment data, cannot handle non-globular clusters very well

Examples

- **Single** linkage (left) vs. **average** linkage (right), $K = 7$

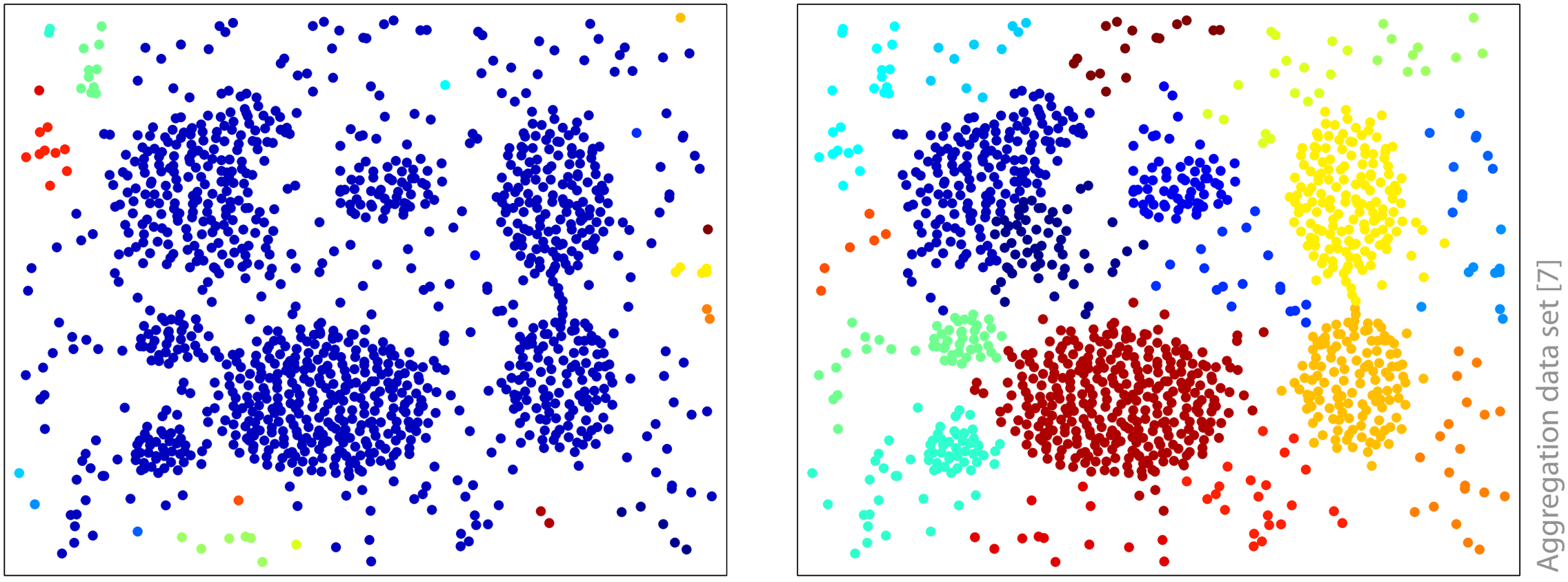


- Single linkage fails quickly in the presence of **noise**

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Examples

- **Single** linkage (left) vs. **average** linkage (right), $K = 20$



Aggregation data set [7]

- Increasing the number of clusters K to "account for" noisy data points does not help

Hierarchical Clustering

Discussion

- Hierarchical clustering methods are easy to understand and simple
- However, they are **not very robust** towards **outliers** or **noise** as such points will either show up as additional clusters or cause other clusters to merge (chaining phenomenon), in particular with single-linkage criterion
- Can **never undo** what was done previously
- Assignments from points to clusters are **hard**
- **They are slow**. Time complexity of $O(N^3)$ from scanning a N x N distance matrix for the largest similarity in each of N–1 iterations. Smarter implementations reach $O(N^2 \log N)$ but this is still too high for large N

- This brings us to consider a more efficient and very popular clustering method: **k-means**

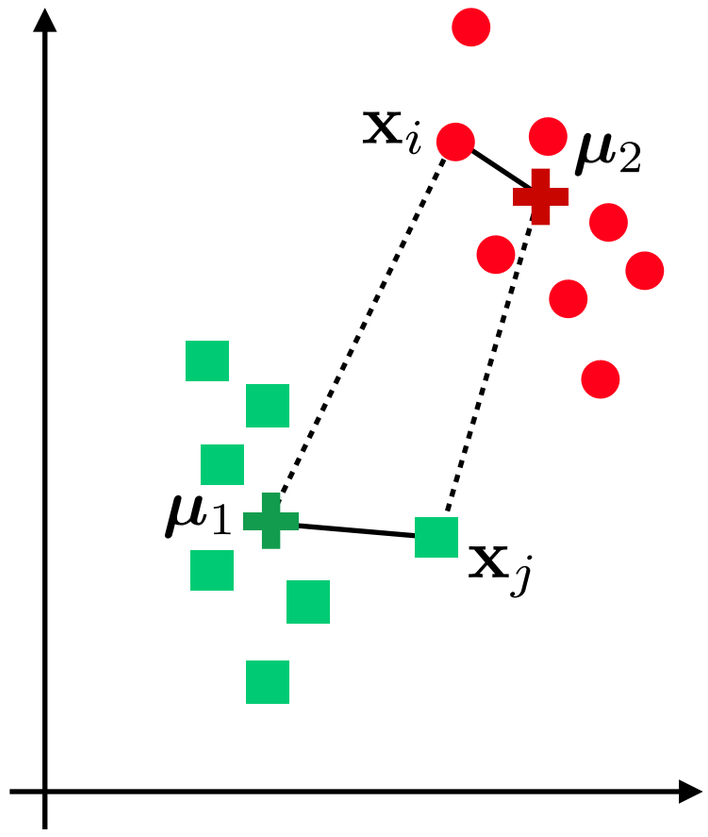
Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Contents

K-Means

Prototypes

- K-means clustering aims to partition the data set $\mathcal{D}$ into clusters in which each point belongs to the cluster with the **nearest mean**, serving as a **prototype** or centroid $\boldsymbol{\mu}_k$ of the cluster
- The **goal** of k-means is then to find an assignment of data points to clusters, as well as to find the set of vectors $\{\boldsymbol{\mu}_k\}_{k=1}^{K}$, such that the **sum of the squares of the distances** of each data point to its closest vector is minimal



- Let $r_{nk} \in \{0, 1\}$ be a **binary indicator variable** for each data point $\mathbf{x}_n$ describing which of the K clusters the data point $\mathbf{x}_n$ is assigned to. If point $\mathbf{x}_n$ is assigned to cluster k then $r_{nk} = 1$ otherwise $r_{nk} = 0$

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Objective Function

- We can then define an **objective function** (sometimes called **distortion measure**) given by

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

  which represents the sum of the squared distances of each data point to its assigned prototype $\boldsymbol{\mu}_k$. Our goal is find values of all r_{nk} and $\boldsymbol{\mu}_k$ so as to minimize J

- How can we minimize J? Let us consider the variables of interest separately and how they can minimize the objective function
- Because J is a **linear** function of r_{nk}, this optimization can be done in closed form and for each summand independently. We see that we should simply choose r_{nk} to be 1 for whichever n and k the distance $\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$ is minimal

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Objective Function

- J is quadratic in $\boldsymbol{\mu}_k$. Thus, J is minimized by setting its derivative w.r.t. $\boldsymbol{\mu}_k$ to zero giving

$$2\sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \boldsymbol{\mu}_k) = 0$$

  which we can solve for $\boldsymbol{\mu}_k$ to give

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} r_{nk}\mathbf{x}_n}{\sum_{n=1}^{N} r_{nk}}$$

  The denominator is equal to the number of points assigned to cluster k, and so this expression computes the mean of all data points in cluster k

- The k-means algorithm (in the variant of Lloyd) uses a **two-step iterative refinement** technique to minimize J, alternating between an optimization step w.r.t. r_{nk} and an optimization step w.r.t. $\boldsymbol{\mu}_k$

K-Means

Algorithm

- Given an initial set $\{\boldsymbol{\mu}_k\}_{k=1}^K$, k-means alternates between two steps

1. **Assignment step**: minimize J w.r.t. the r_{nk} keeping the $\boldsymbol{\mu}_k$ fixed

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

2. **Update step**: minimize J w.r.t. the $\boldsymbol{\mu}_k$ keeping the r_{nk} fixed

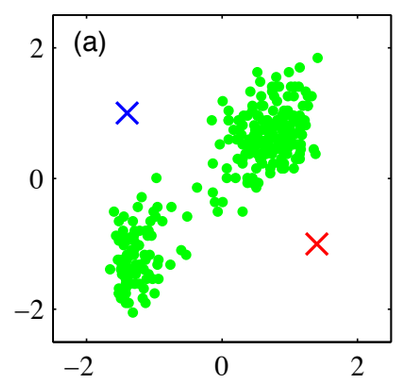
$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^N r_{nk}\mathbf{x}_n}{\sum_{n=1}^N r_{nk}}$$

- Because each phase reduces the value of the objective function J, **convergence** is assured. However, it may converge to a local rather than a global minimum

K-Means

Algorithm

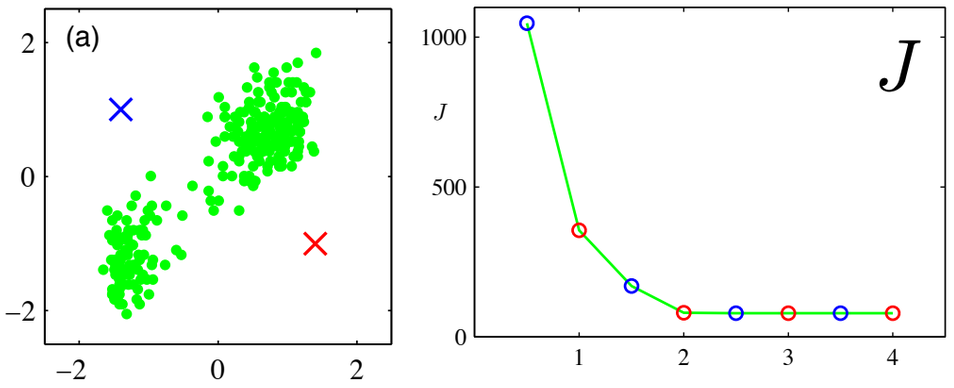
- The two phases of re-assigning data points to clusters and re-computing the cluster means are repeated until there is **no further change** in the assignments or until some **maximum number of iterations** is exceeded
- There are K^N possible clusterings
- The algorithm due to Lloyd finds an approximate solution to the problem, the exact solution, that is, the optimal partitioning of the data into clusters under the objective function is **NP-hard**
- Notice the connection between the similarity measure (e.g. Euclidian distance) and the update step expression for the cluster centers.
  We expect **different similarity measures** to lead to **different update rules**
- Let us illustrate the algorithm using a simple data set in 2D with poor initial values for the cluster centers

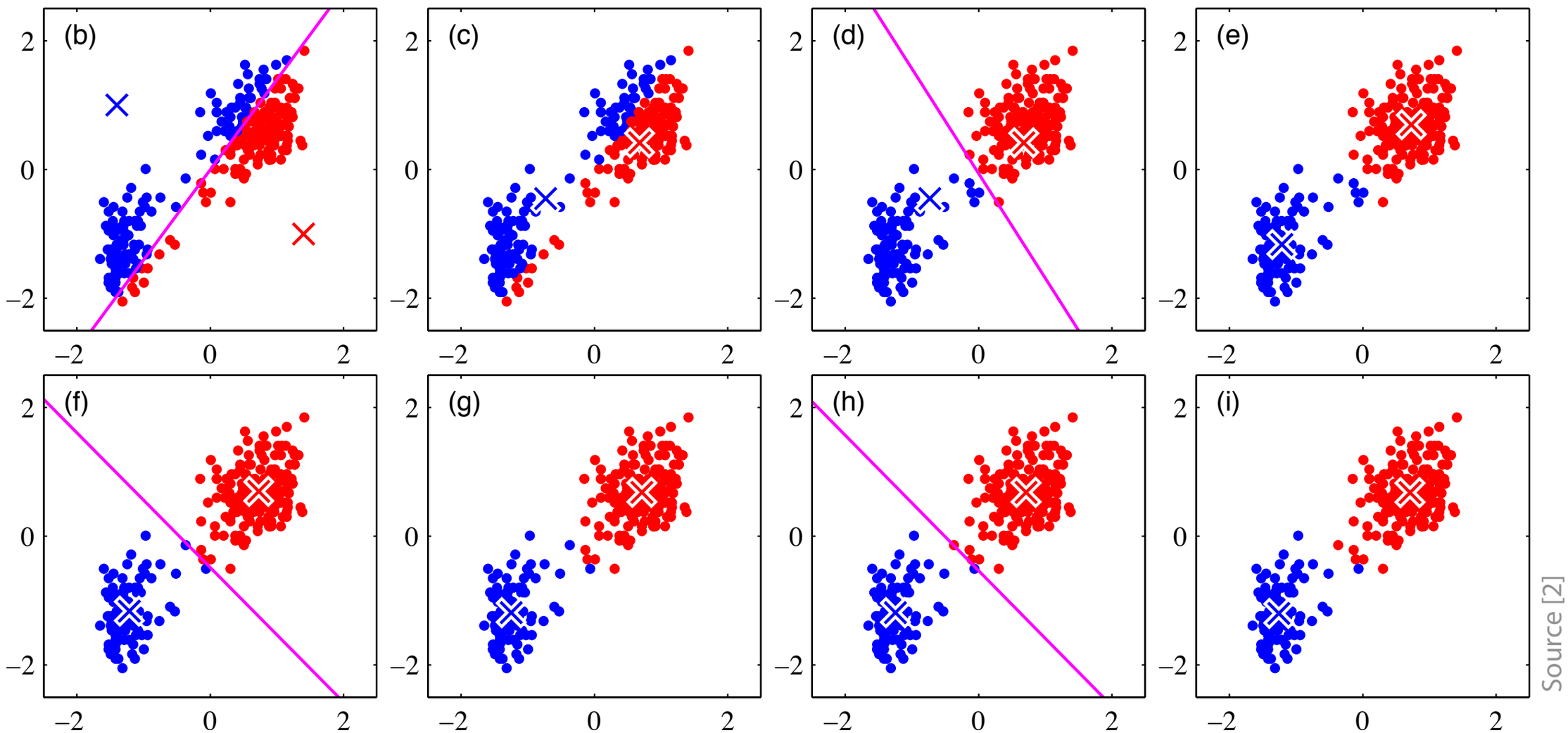


K-Means

Examples

- The algorithm alternates between re-assigning and updating, minimizing objective function J



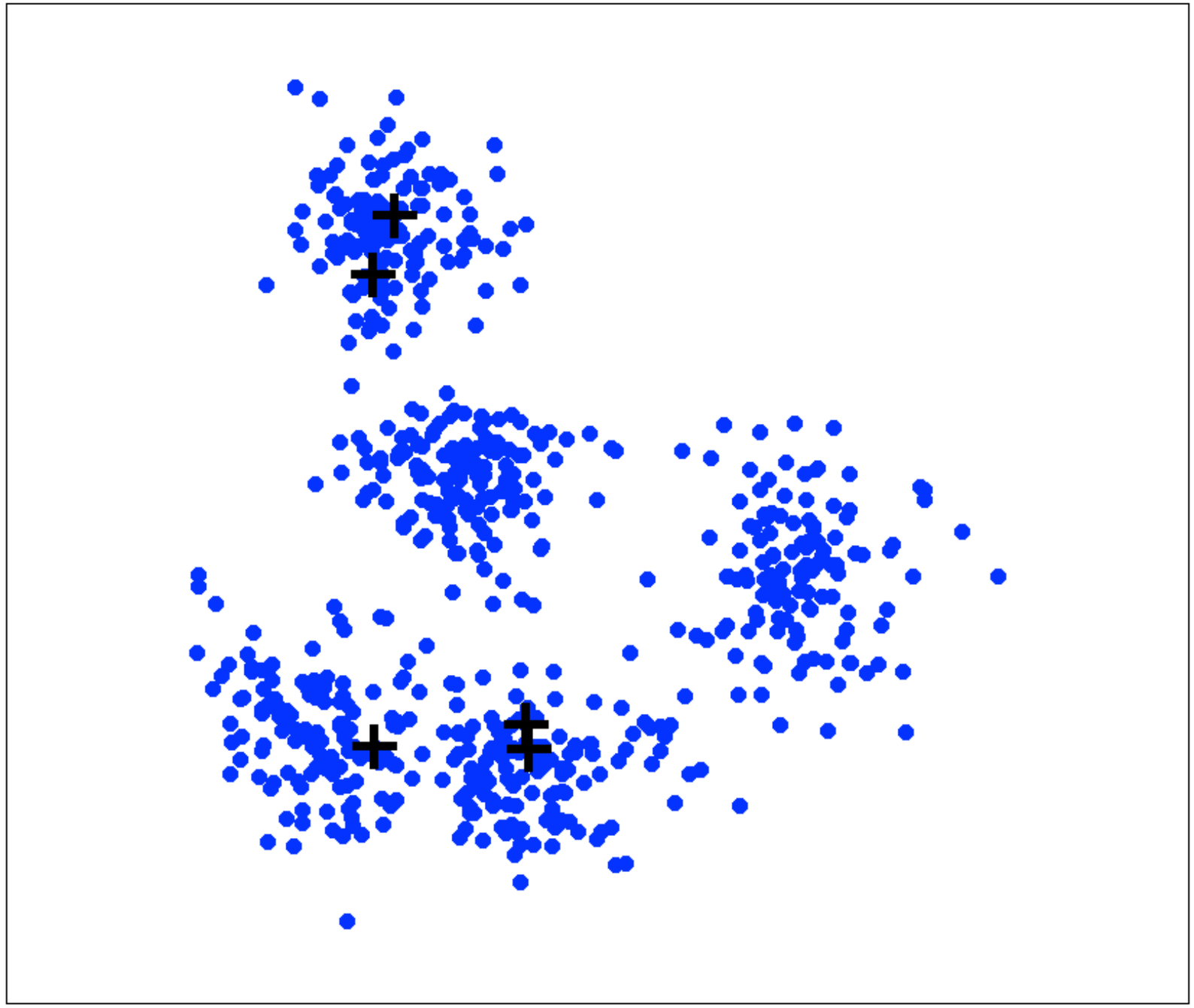


Source [2]

K-Means

Examples

- $K = 5$
- K-means partitions the data space into a **Voronoi diagram**

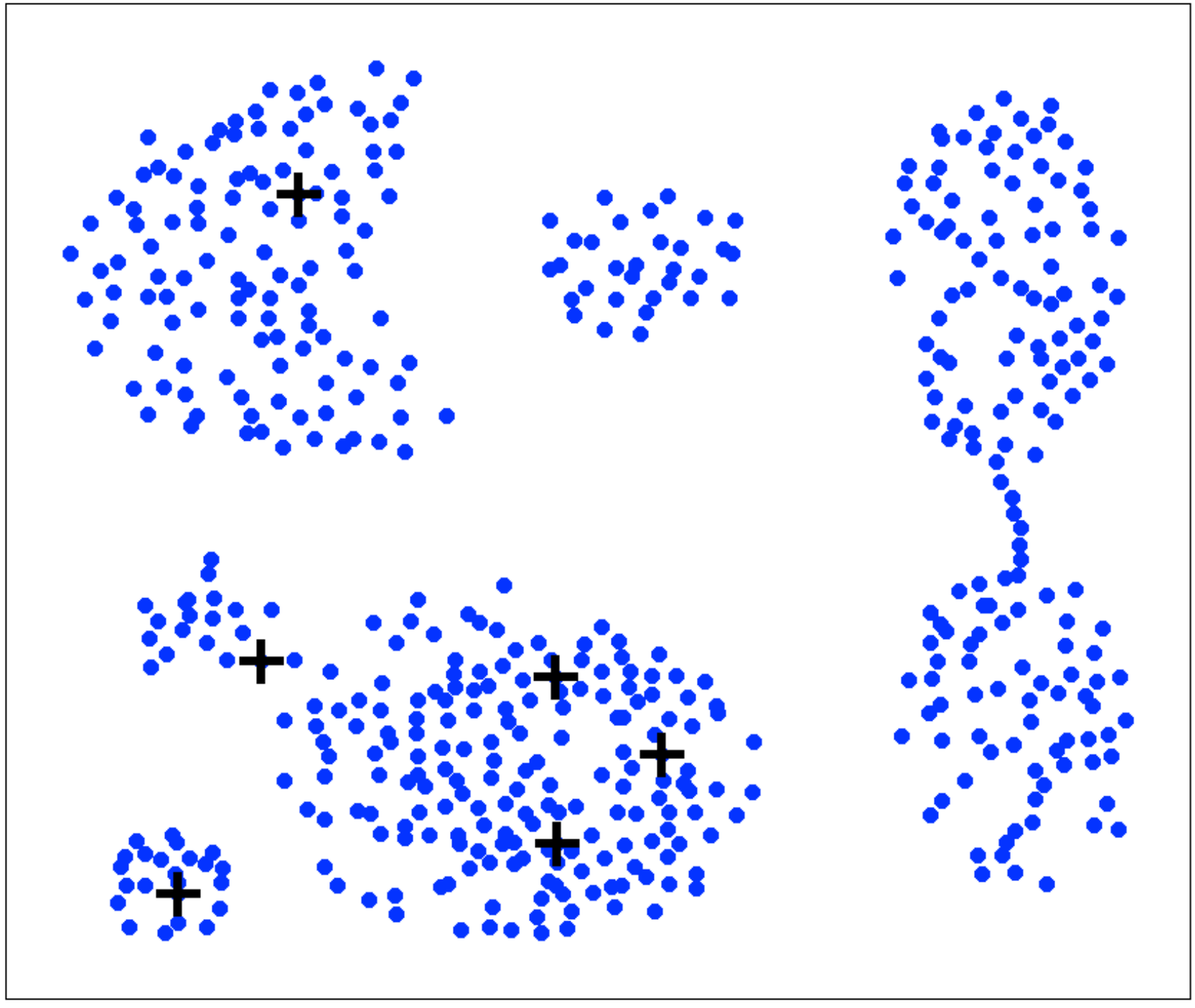


Movie

K-Means

Examples

- $K = 6$

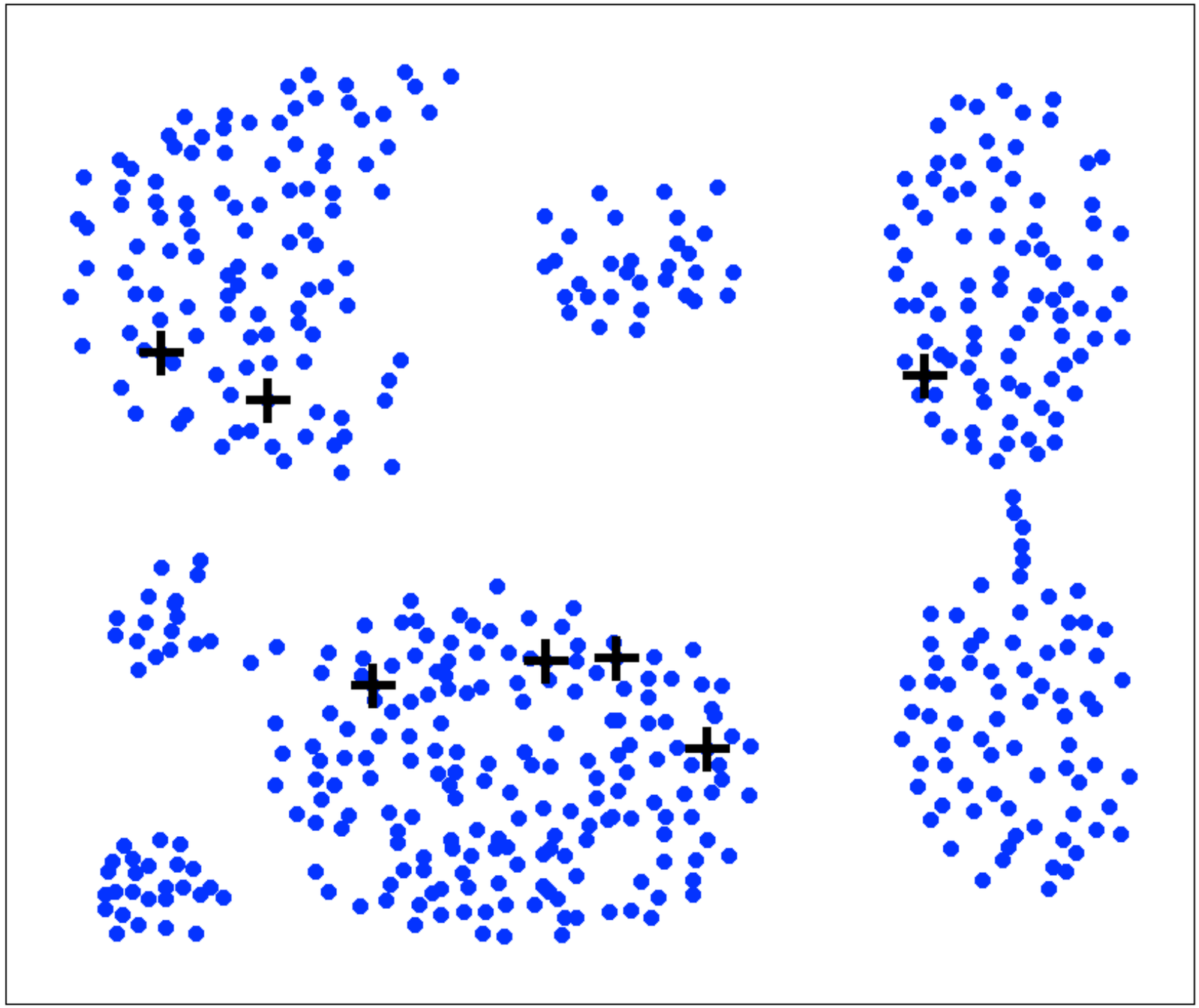


Aggregation data set [7]

K-Means

Examples

- $K = 7$

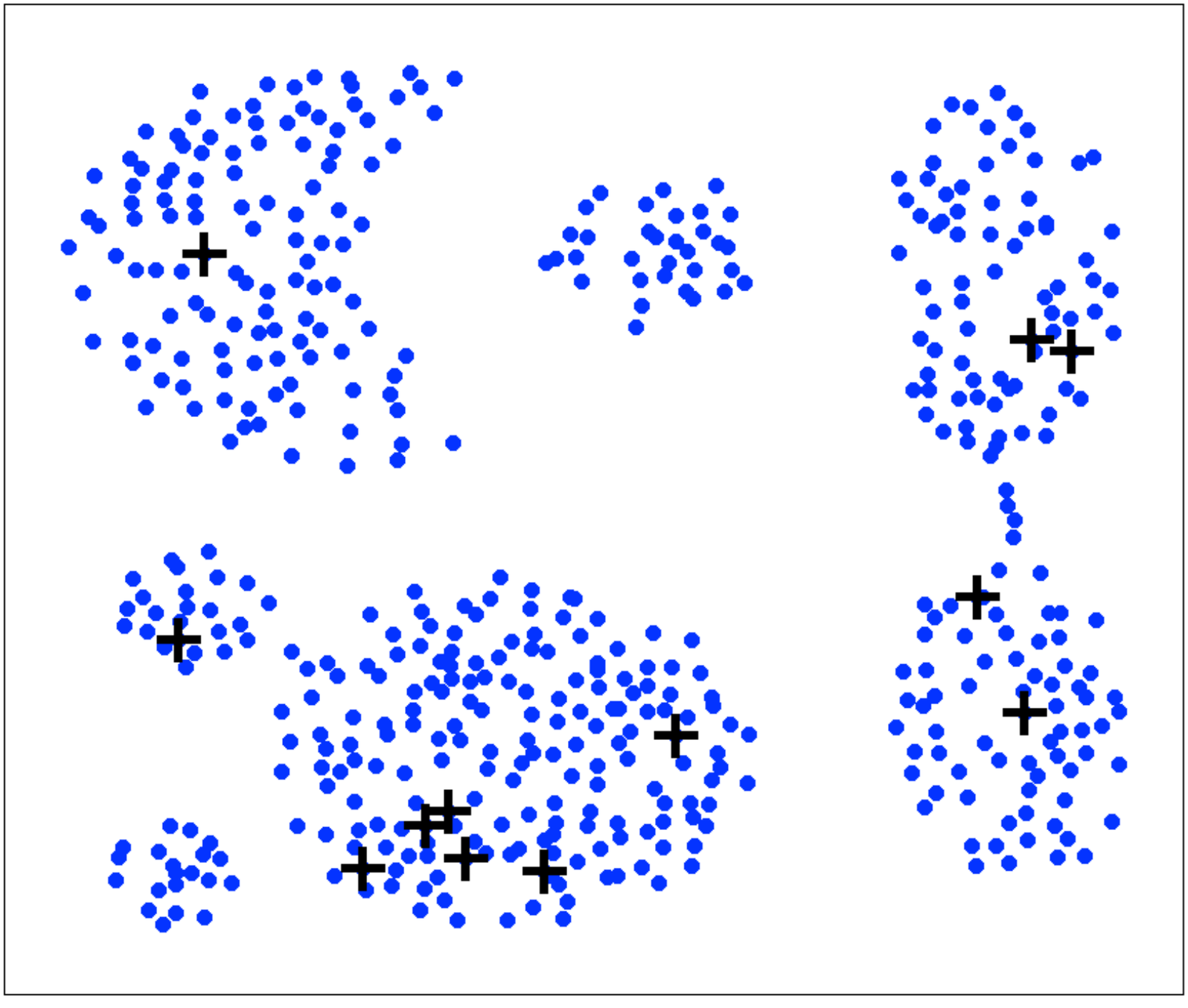


Aggregation data set [7]

K-Means

Examples

- $K = 12$

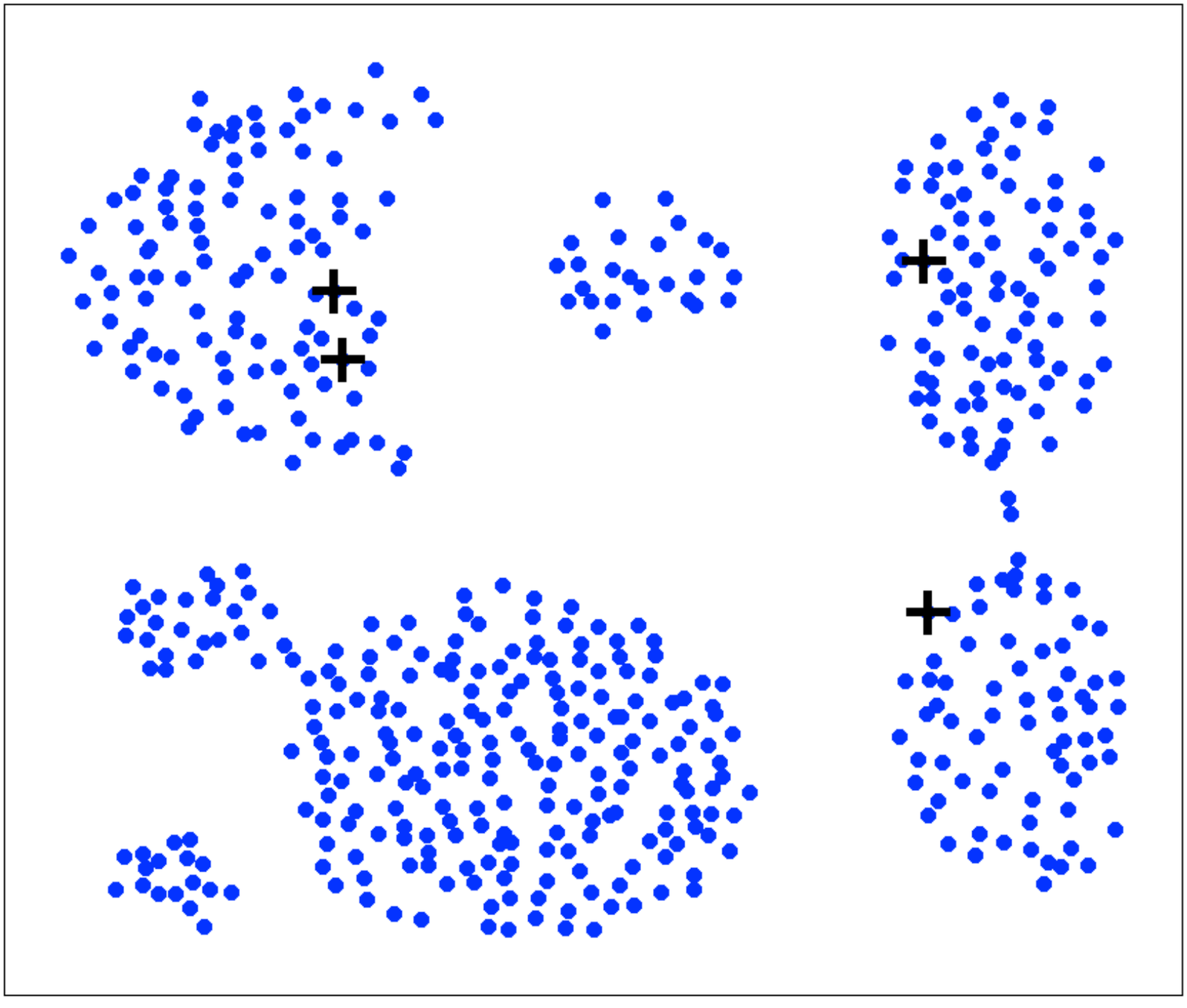


Aggregation data set [7]

K-Means

Examples

- $K = 4$



Aggregation data set [7]

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Initialization of the Centroids

- The performance of k-means **strongly** depends and the initialization of the cluster centers
- The simplest strategy is to **randomly draw** the initial K prototypes within the data range. A better strategy is to choose the initial centroids **uniformly at random from** $\mathcal{D}$
- A popular initialization technique is **k-means++**
  - It chooses centers at random from $\mathcal{D}$ with a probability proportional to the squared distance from the closest already chosen center
  - While the approximation found by the regular algorithm can be arbitrarily bad with respect to the objective function compared to the optimal clustering, k-means++ is guaranteed to find a solution that is O(log K) competitive to the optimal k-means solution
  - Can lead to considerable improvement of k-means both in accuracy and speed

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Initialization of *K*

- The performance of k-means depends **heavily** also on K
- How to choose K? Most methods for automatically determining the number of clusters cast it into a **model selection problem**
- Generally, the clustering algorithms is ran with **different values** of K and **the best** K is chosen based on a predefined criterion
- Being a model selection problem, typical criteria include the minimum description length (MDL), the Bayes Information Criterion (BIC) or the Akiake Information Criterion (AIC). They all trade off **data likelihood** (how well the model explains the data) and **model complexity** (K in this case)
- The **X-means algorithm**, another k-means extension, uses the BIC

$$\mathrm{BIC}(M|\mathcal{D}) = \log p(\mathcal{D}|M) - \frac{p}{2}\log N$$

with M being the model and $p = K(m + 1)$ its number of parameters

K-Means

Extensions

- **K-Medians.** Instead of calculating the mean for each cluster to determine its centroid, the **median** is calculated. This corresponds to minimizing the error over all clusters with respect to the **1-norm distance metric**
- **K-Medoids.** Uses a more **general similarity measures** between two points than the Euclidian distance and chooses **data points as centers** (medoids). The latter is a consequence of the former because optimization in the update step is potentially more complex
- **On-line.** Unlike the batch version of the algorithm, there is an **on-line version** of k-means using a sequential update rule for prototype vectors
- **Speed.** A naive implementation can be **slow** because each assignment computes the Euclidean distance between every prototype and every data point. Extensions to speed up k-means use, for example, **tree data structures** (e.g. kd-trees) that accelerate access to nearby points

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Extensions

- K-means is sensitive to **outliers** and **noise** because such points are necessarily assigned to one of the clusters and influence the respective means. Based on the assumption that small clusters are likely formed by outliers, a simple approach is have a **size filter** discard such clusters
- K-means works with continuous valued features. Variants that can deal with **discrete** (categorial/nominal) data have been proposed, too
- K-means can only separate clusters that are **linearly separable**. **Kernel k-means** maps the points to a higher-dimensional feature space using a nonlinear function, and then partitions the points by linear separators in the new space
- An alternative approach to this issue is **spectral clustering**

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Discussion

- K-means is simple and converges quickly to a local optimum. It has **linear time complexity** $O(I\ K\ N)$ where I is the number of iterations
- However, K-means prefers clusters of approximately similar size, as it will always assign a data point to the nearest centroid. This often leads to **incorrectly cut borders** in between of clusters (which is not surprising, as the algorithm optimized cluster centers, not cluster borders)
- K-Means is restricted to data which has the **notion of a center**. It is not well suited for elongated, convex or non-globular clusters

- While k-means relaxes the irreversibility of decisions in hierarchical clustering, **assignments** of data points to clusters are still **hard**. This is a poor model for points near the boundary, for outliers or noisy data
- Thus, let us consider techniques with **soft assignments**

Unsupervised Learning

Contents

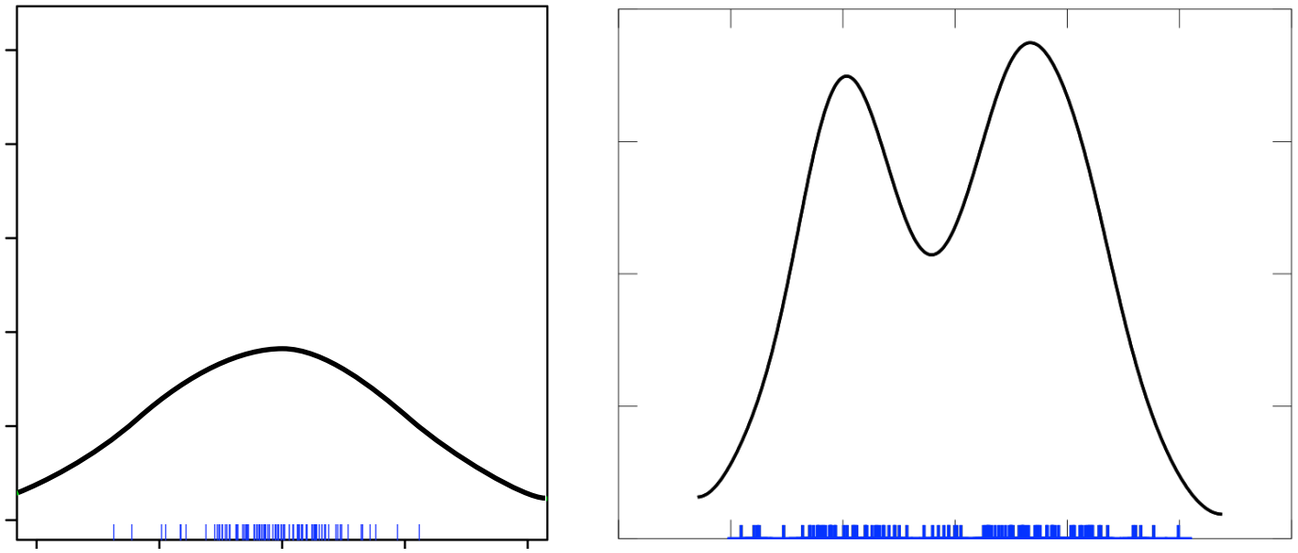
- Introduction
- Hierarchical Clustering
- K-Means
- Gaussian Mixture Models

Unsupervised Learning

Density Estimation

- Let us take a **probabilistic view** and frame the clustering problem as a **parametric density estimation** problem

- The idea is to estimate a **parametric probability distribution** $p(\mathbf{x})$ over data $\mathbf{x}$ and then **recover the clusters** from $p(\mathbf{x})$



- A parametric density estimation example: fitting a Gaussians to individual attributes/features in the Naive Bayes classifier
- However, for clustering, data densities are very complex, a single distributions **will not do the job**

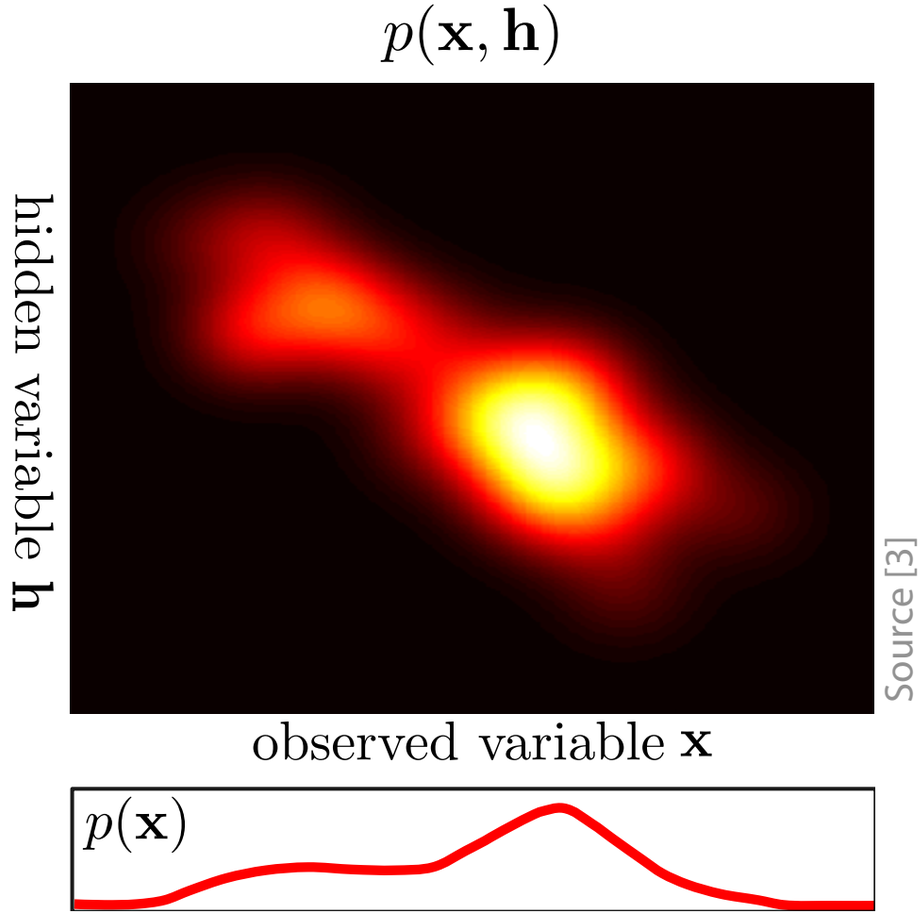
Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Hidden Variables

- To model complex probability densities, let us consider a flexible family of distributions that emerges from the introduction of **hidden variables**
- Hidden variables, also known as **latent variables**, can be discrete or **continuous**
- To exploit hidden variables, we describe the wanted density $p(\mathbf{x})$ as the **marginal of the joint** $p(\mathbf{x}, \mathbf{h})$

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{h})\, d\mathbf{h}$$

- We then **concentrate on working with the joint density** $p(\mathbf{x}, \mathbf{h})$
- Proper choices for $p(\mathbf{x}, \mathbf{h})$ will produce powerful yet simple models



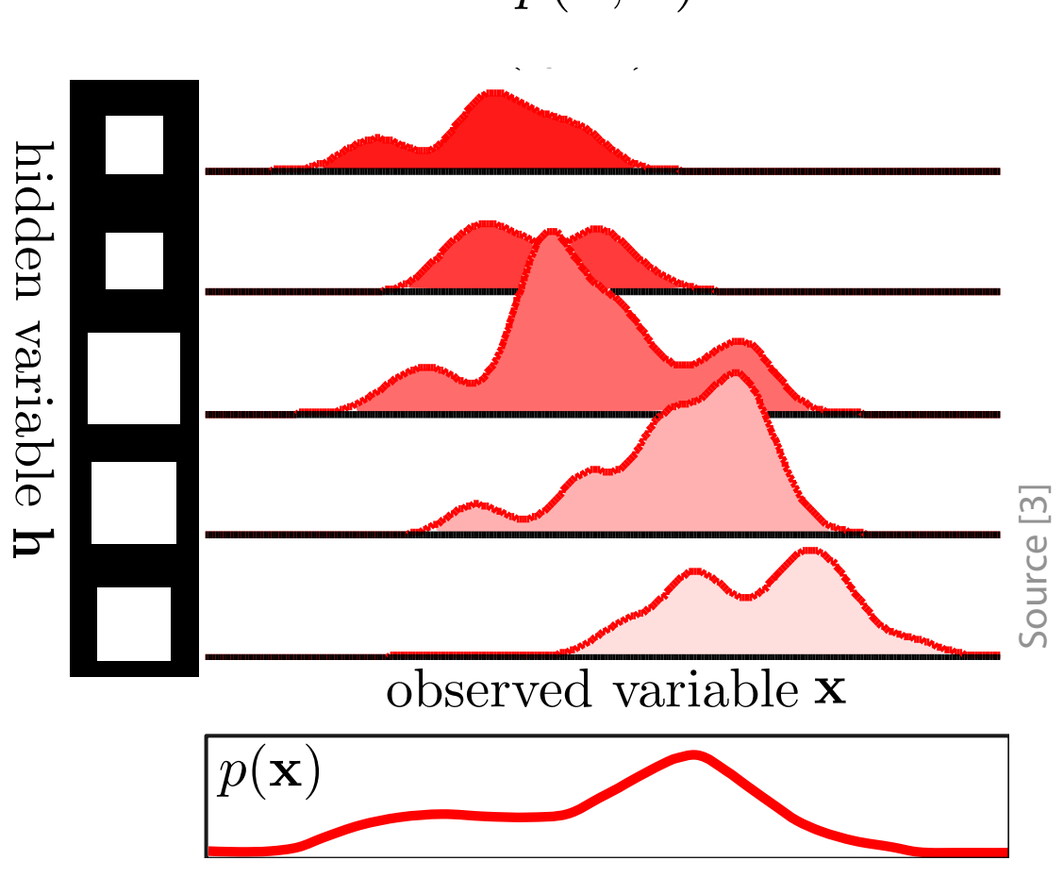
Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Hidden Variables

- To model complex probability densities, let us consider a flexible family of distributions that emerges from the introduction of **hidden variables**
- Hidden variables, also known as **latent variables**, can be **discrete** or continuous
- To exploit hidden variables, we describe the wanted density $p(\mathbf{x})$ as the **marginal of the joint** $p(\mathbf{x}, \mathbf{h})$

$$p(\mathbf{x}) = \sum_{\mathbf{h}} p(\mathbf{x}, \mathbf{h})$$

- We then **concentrate on working with the joint density** $p(\mathbf{x}, \mathbf{h})$
- Proper choices for $p(\mathbf{x}, \mathbf{h})$ will produce powerful yet simple models



Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Mixture Models

- For discrete $\mathbf{h}$, $p(\mathbf{x}, \mathbf{h})$ is a **mixture model**, a flexible family of distributions for describing complex data densities via a **linear combination of density functions**

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{h} = k) p(\mathbf{x}|\mathbf{h} = k) \qquad \sum_{k=1}^{K} p(\mathbf{h} = k) = 1$$

- The model assumes that K **distributions** contribute to $p(\mathbf{x})$
- The hidden variable $\mathbf{h}$ has values $k = 1...K$ and denotes the respective **mixture component**. $\mathbf{h}$ follows a categorial distribution
- It can be shown that this modeling can approximate **arbitrarily closely any continuous density function** for a sufficient number of components
- It is a **generative model**: points $\mathbf{x}$ can be generated by first choosing a component with probability p_k, and then generating a sample from it

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

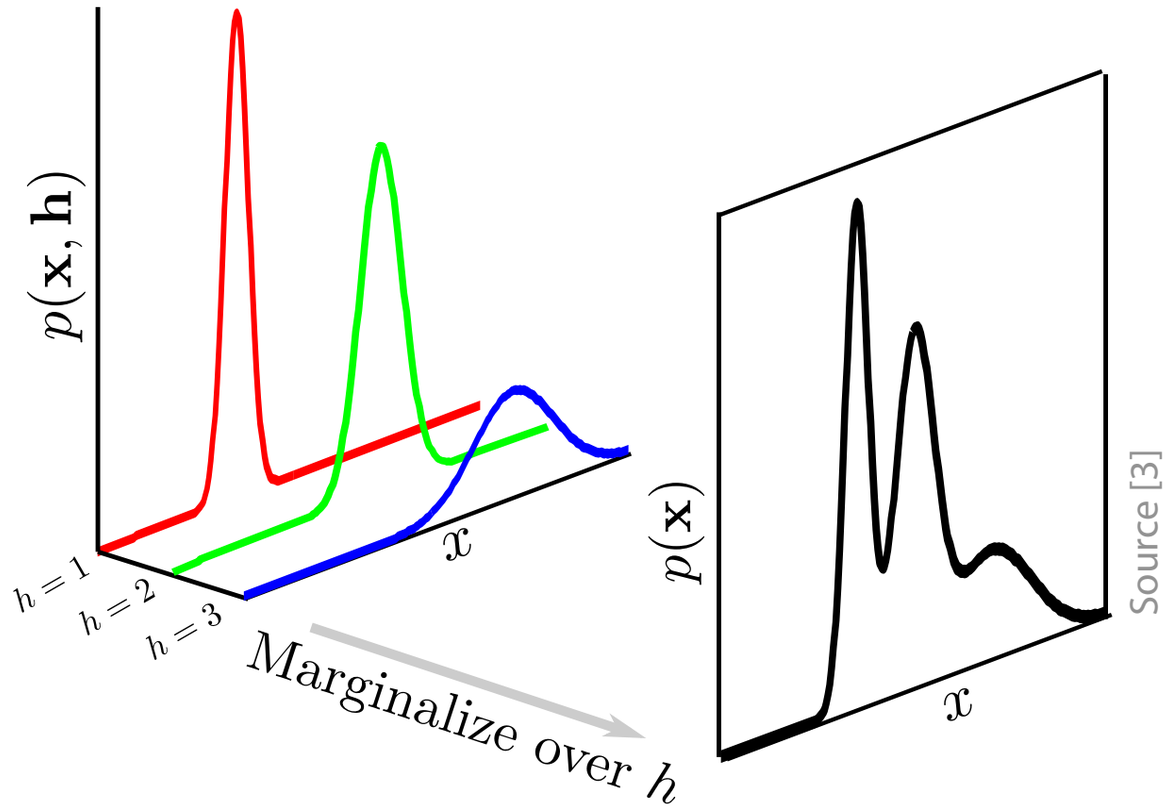
Gaussian Mixture Models

- Gaussian mixture models (GMM) have **Gaussian** mixture components

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{h}=k)\, p(\mathbf{x}|\mathbf{h}=k)$$

$$= \sum_{k=1}^{K} p_k \underbrace{\mathcal{N}_{\mathbf{x}}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{\text{Component}}$$

Mixture coefficient



Source [3]

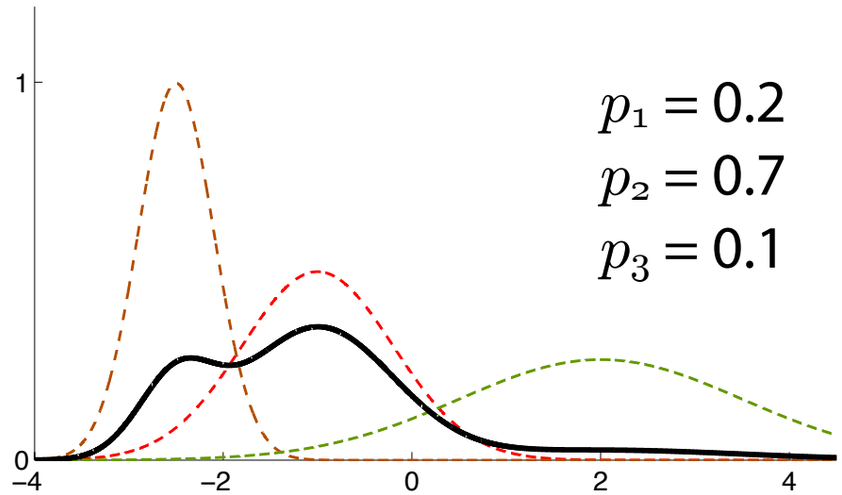
Gaussian Mixture Models

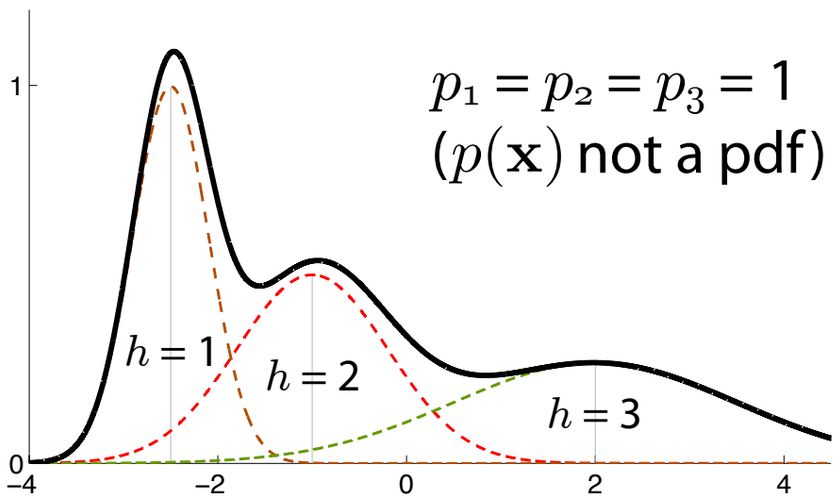
- Gaussian mixture models (GMM) have **Gaussian** mixture components

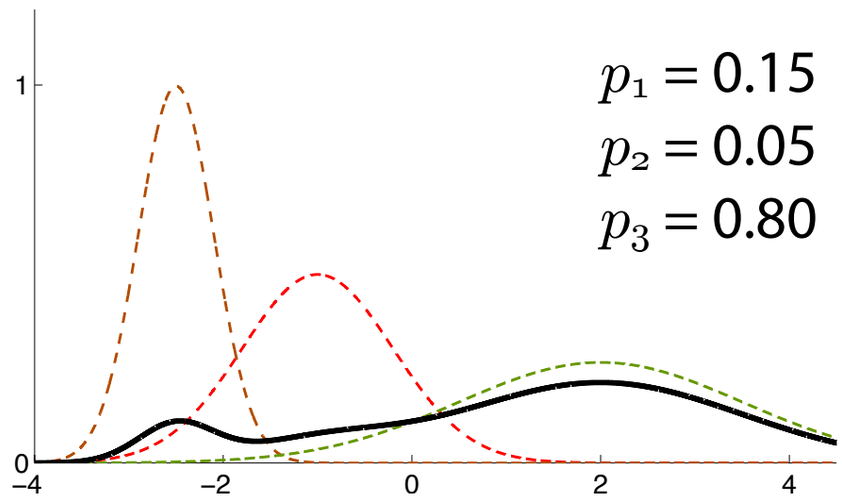
$$p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{h}=k)\, p(\mathbf{x}|\mathbf{h}=k)$$

$$= \sum_{k=1}^{K} p_k \underbrace{\mathcal{N}_{\mathbf{x}}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{\text{Component}}$$

Mixture coefficient



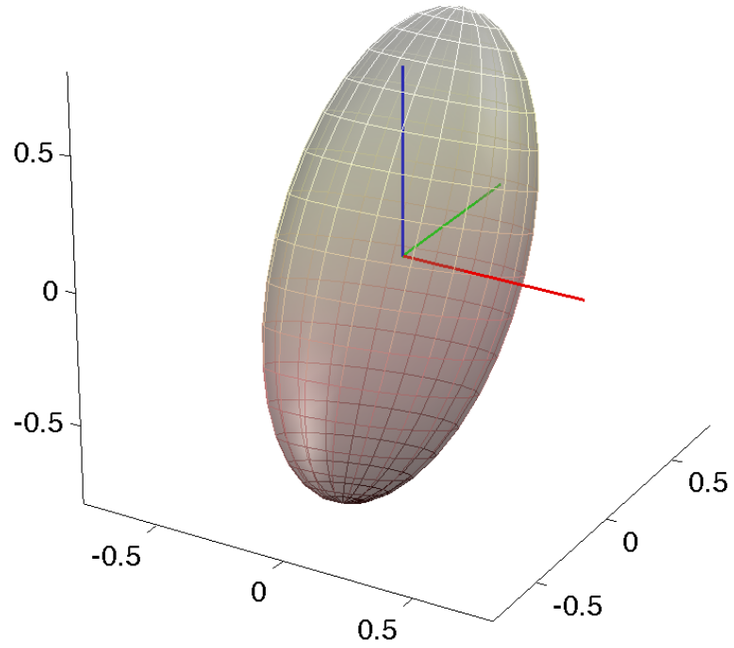




Common Probability Distributions

Multivariate Gaussian Distribution

- For d-dimensional random vectors, the **multivariate Gaussian distribution** is governed by a d-dimensional **mean vector** $\boldsymbol{\mu}$ and a D x D **covariance matrix** Σ that must be symmetric and positive semi-definite



- **Probability density function**

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})\right\}$$

- Notation

$$\mathcal{N}_{\mathbf{x}}(\boldsymbol{\mu}, \Sigma) = p(\mathbf{x})$$

Parameters

- $\boldsymbol{\mu}$: mean vector
- Σ: covariance matrix

Expectation

- $\mathrm{E}[\mathbf{x}] = \boldsymbol{\mu}$

Variance

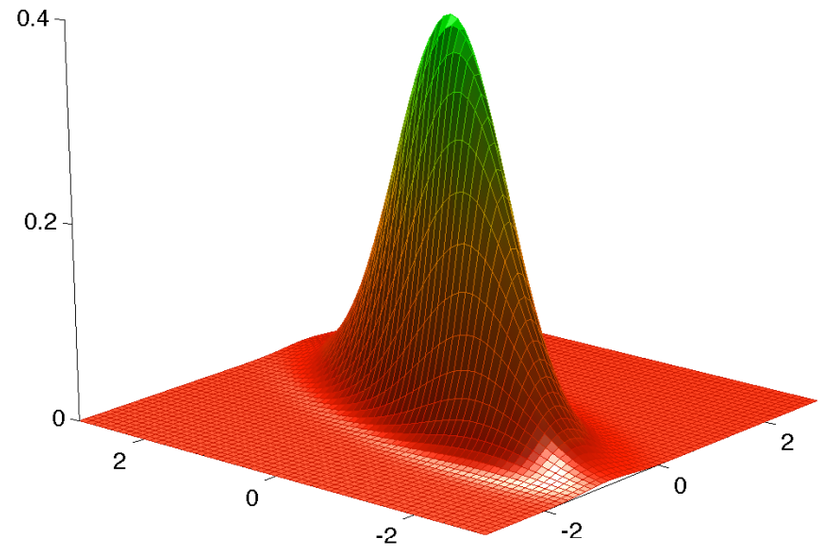
- $\mathrm{Var}[\mathbf{x}] = \Sigma$

Common Probability Distributions

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Multivariate Gaussian Distribution

- For $d = 2$, we have the **bivariate** Gaussian distribution
- The covariance matrix Σ (often C) determines the **shape of the distribution** (video)

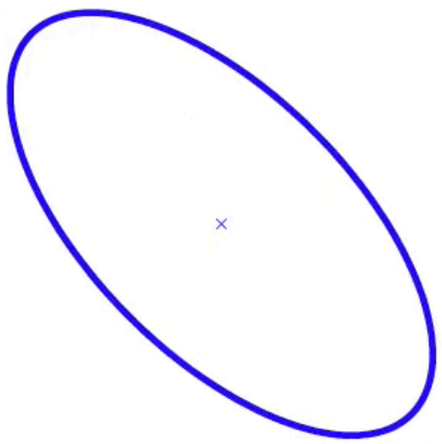


$$C = \begin{bmatrix} 0.020 & -0.012 \\ -0.012 & 0.020 \end{bmatrix}$$

$\lambda_1 = 0.008$

$\lambda_2 = 0.032$

$\rho = \sigma_{XY} / \sigma_X \sigma_Y = -0.618$



Parameters

- $\boldsymbol{\mu}$: mean vector
- Σ: covariance matrix

Expectation

- $\mathrm{E}[\mathbf{x}] = \boldsymbol{\mu}$

Variance

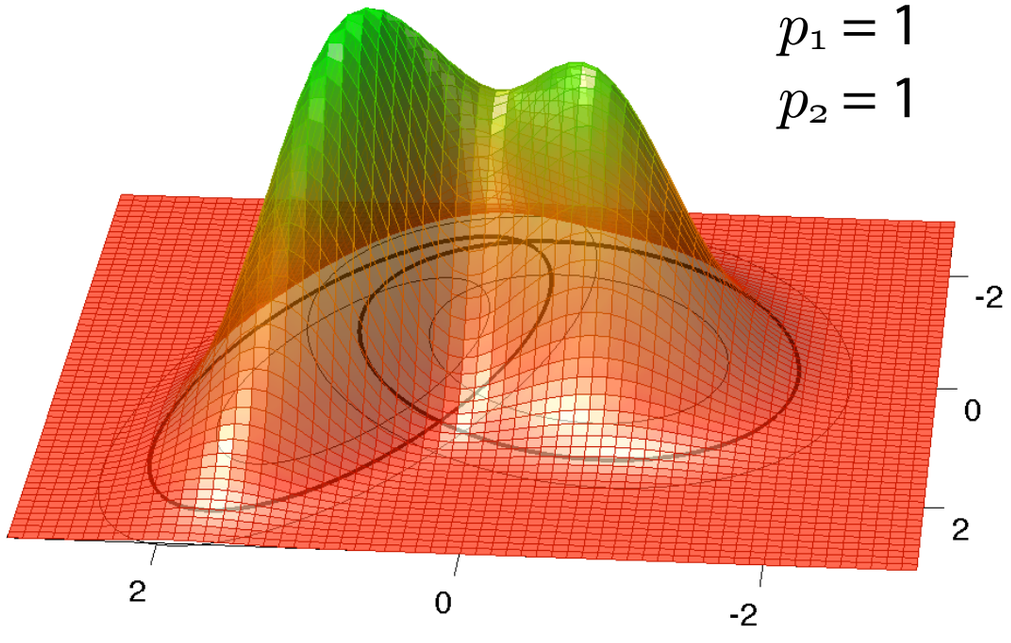
- $\mathrm{Var}[\mathbf{x}] = \Sigma$

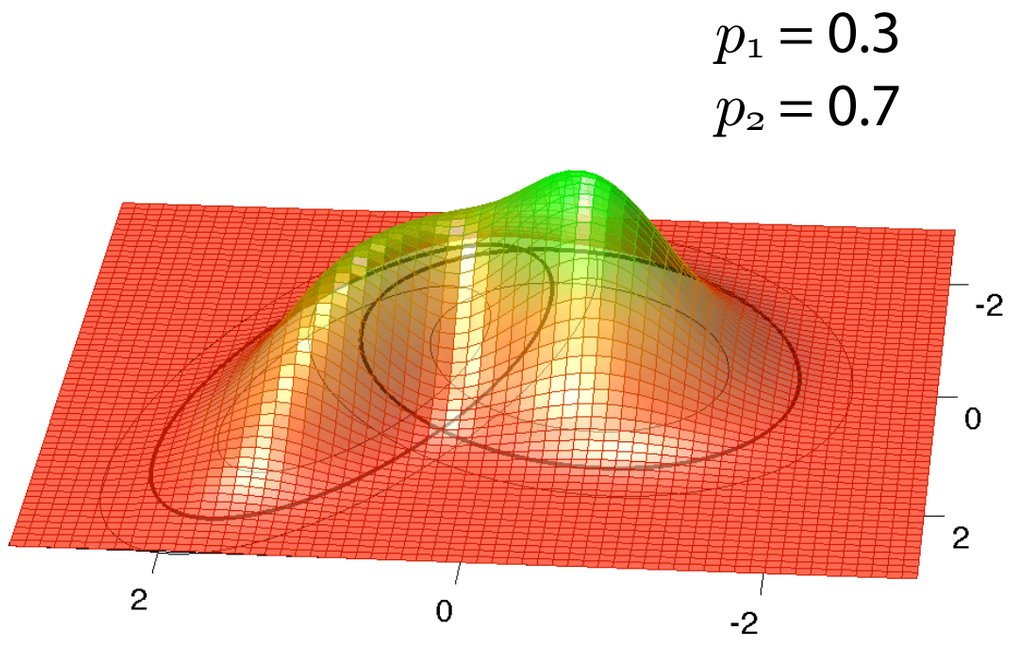
Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Gaussian Mixture Models

- Bivariate example

$$p(\mathbf{x}) = \sum_{k=1}^{K} p_k \, \mathcal{N}_{\mathbf{x}}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$





- Parameters of a Gaussian mixture model are: p_k , the mixture coefficient or **weight** of each component, $\boldsymbol{\mu}_k$, the **mean** of each component, and Σ_k , the **covariance** of each component

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Learning Gaussian Mixture Models

- **Learning** a Gaussian mixture model consists in (the usual) fitting of the model parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, p_k\}_{k=1}^{K}$ to data
- The standard approach would be to maximize the data log-likelihood

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \left[\sum_{i=1}^{N} \log(p(\mathbf{x}_i|\boldsymbol{\theta})) \right] = \arg\max_{\boldsymbol{\theta}} \left[\sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} p_k \, \mathcal{N}_{\mathbf{x}_i}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right) \right]$$

  where we have written $p(\mathbf{x}) = p(\mathbf{x}|\boldsymbol{\theta})$ to make the dependence of the parametric model from its parameters explicit (usual notation skips this)
- Unfortunately, if we take derivates w.r.t. $\boldsymbol{\theta}$ and equate to zero, we will **not** obtain a **closed-form** equation system (due to the sum in the log)
- Non-linear **optimization** would be **very complex** as we would have to account for many **constraints** on the parameters: the weights p_k have to sum up to 1 and the covariances $\boldsymbol{\Sigma}_k$ need to be positive definite

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Learning Gaussian Mixture Models

- We have to look out for another approach...
- Learning would be easy if we **knew the parameters** of each component: then, we could assign each data point to the component that maximizes the likelihood $p(\mathbf{x}|\mathbf{h} = k) = \mathcal{N}_{\mathbf{x}}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ and derive the weights
- It would also be easy if we **knew which component** generated each data point: we could simply select all points from a given component and fit the parameters of the Gaussian to those data
- The problem is that we know **neither** the assignments **nor** the component parameters
- Hence the name "hidden" variables. They are **not observable** in the data available for learning
- This is where the **expectation-maximization** algorithm comes into play

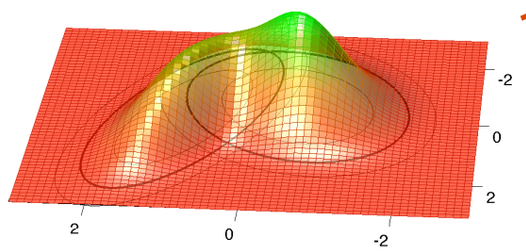
Expectation-Maximization

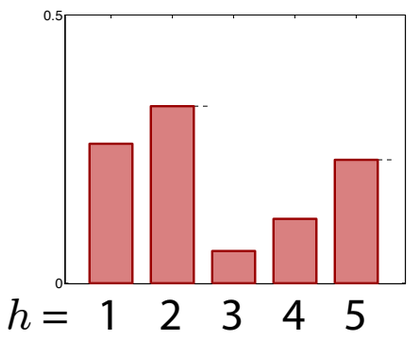
- The **expectation-maximization** algorithm (EM) is an algorithm for fitting parameters θ in models with hidden variables
- The basic idea of EM (in this context) is to:
  1. **Pretend** that we know the parameters of the components and then to infer the **probability that each data point belongs** to each component
  2. **Refit the components** to the data using those probabilities. Each component is fitted to the entire data set with each point weighted by the probability that it belongs to that component
- EM **alternates** between these two steps until convergence
- Notice the similarity of this procedure and the k-means algorithm where we have an "**assignment**" **step** followed by an "**update**" **step**

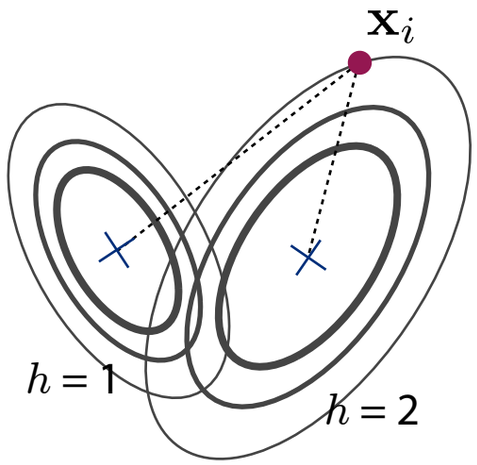
Expectation-Maximization

- The first step is called **expectation step**, or E-step
- In the E-step we compute the **probability that a data point $\mathbf{x}_i$ belongs** to a given mixture component $\mathbf{h}_i = k$
- Doing so for all components yields the discrete distribution $p(\mathbf{h}_i = k|\mathbf{x}_i)$ over the hidden variable which we can compute via Bayes' rule

$$p(\mathbf{h}_i = k|\mathbf{x}_i) = \frac{p(\mathbf{x}_i|\mathbf{h}_i = k)\, p(\mathbf{h}_i = k)}{p(\mathbf{x}_i)}$$







Expectation-Maximization

- The first step is called **expectation step**, or E-step
- In the E-step we compute the **probability that a data point $\mathbf{x}_i$ belongs** to a given mixture component $\mathbf{h}_i = k$
- Doing so for all components yields the discrete distribution $p(\mathbf{h}_i = k|\mathbf{x}_i)$ over the hidden variable which we can compute via Bayes' rule

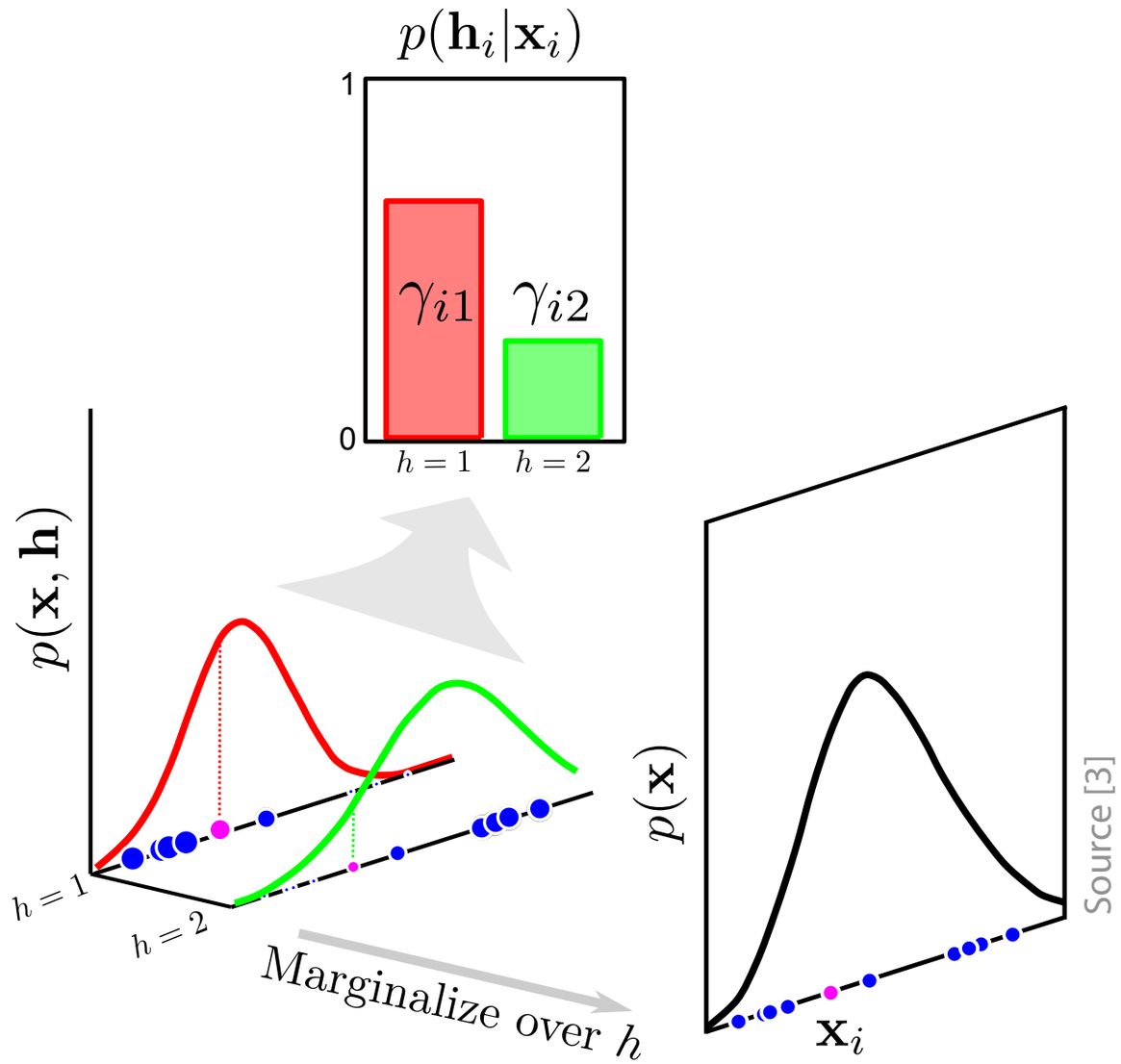
$$
\begin{aligned}
p(\mathbf{h}_i = k|\mathbf{x}_i) &= \frac{p(\mathbf{x}_i|\mathbf{h}_i = k)\, p(\mathbf{h}_i = k)}{p(\mathbf{x}_i)} \\
&= \frac{p_k\, \mathcal{N}_{\mathbf{x}_i}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} p_j\, \mathcal{N}_{\mathbf{x}_i}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \\
&= \gamma_{ik}
\end{aligned}
$$

Soft assignment

Gaussian Mixture Models

Expectation-Maximization

- The first step is called **expectation step**, or E-step
- p_k is the prior probability of $\mathbf{h}_i = k$, and the quantity γ_{ik} the **posterior probability** once we have observed $\mathbf{x}_i$
- γ_{ik} is called **responsibility** because it is the probability that the k-th Gaussian was responsible for the i-th data point $\mathbf{x}_i$



Source [3]

Expectation-Maximization

- The second step is called **maximization step**, or M-step
- In the M-step, we **update the component parameters** based on the updated responsibilities. Concretely, we fit the component to the **entire** data set with each point weighted by γ_{ik}

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{ik}\, \mathbf{x}_i$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{ik}\, (\mathbf{x}_i - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_i - \boldsymbol{\mu}_k^{\text{new}})^T$$

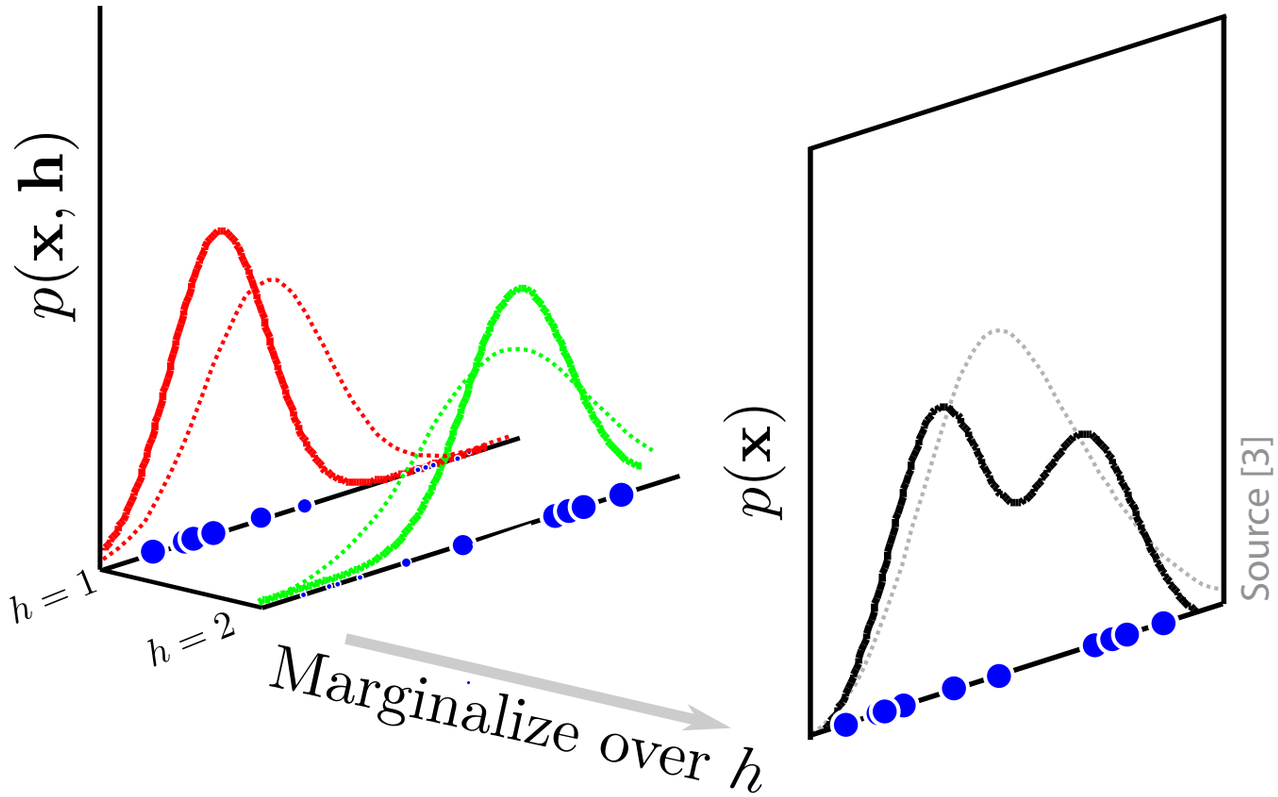
$$p_k^{\text{new}} = \frac{N_k}{N}$$

where $N_k = \sum_{i=1}^{N} \gamma_{ik}$ is the **relative total responsibility** of component k

Gaussian Mixture Models

Expectation-Maximization

- The second step is called **maximization step**, or M-step
- Data points that are **more associated** with the k-th component (high probability γ_{ik}) have **more effect** on its parameter updates
- Dashed and solid lines represent fit before and after the update, respectively. Size of data points indicate responsibility



Gaussian Mixture Models

Expectation-Maximization

- Given initial parameters $\boldsymbol{\theta}_0$, alternate until convergence

1. **E-step**: compute responsibilities keeping $\boldsymbol{\theta} = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, p_k\}_{k=1}^K$ fixed

$$\gamma_{ik} = \frac{p_k \, \mathcal{N}_{\mathbf{x}_i}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K p_j \, \mathcal{N}_{\mathbf{x}_i}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \qquad N_k = \sum_{i=1}^N \gamma_{ik}$$

2. **M-step**: update component parameters keeping the γ_{ik} fixed

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{i=1}^N \gamma_{ik} \, \mathbf{x}_i \qquad p_k^{\text{new}} = \frac{N_k}{N}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{i=1}^N \gamma_{ik} \, (\mathbf{x}_i - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_i - \boldsymbol{\mu}_k^{\text{new}})^T$$

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Expectation-Maximization

- Why does EM work? EM is a **general algorithm** for fitting parameters $\boldsymbol{\theta}$ in models with latent variables. It maximizes the data log-likelihood

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \left[\sum_{i=1}^{N} \log \left(\int p(\mathbf{x}_i, \mathbf{h}_i | \boldsymbol{\theta}) \, d\mathbf{h}_i \right) \right]$$

  by defining a (cleverly chosen) **lower bound** and **iteratively increasing this bound**
- The bound is a function of the parameters $\boldsymbol{\theta}$ and N probability distributions $\boldsymbol{q}_i(\mathbf{h}_i)$ over the hidden variables (here $p(\mathbf{h}_i = k | \mathbf{x}_i)$)
- The $\boldsymbol{q}_i(\mathbf{h}_i)$'s are manipulated in the E-step and the $\boldsymbol{\theta}$'s are manipulated in the M-step, both in a way that in each step the bound is **guaranteed to be improved**
- Thus, EM is guaranteed to converge at least to a **local maximum**

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

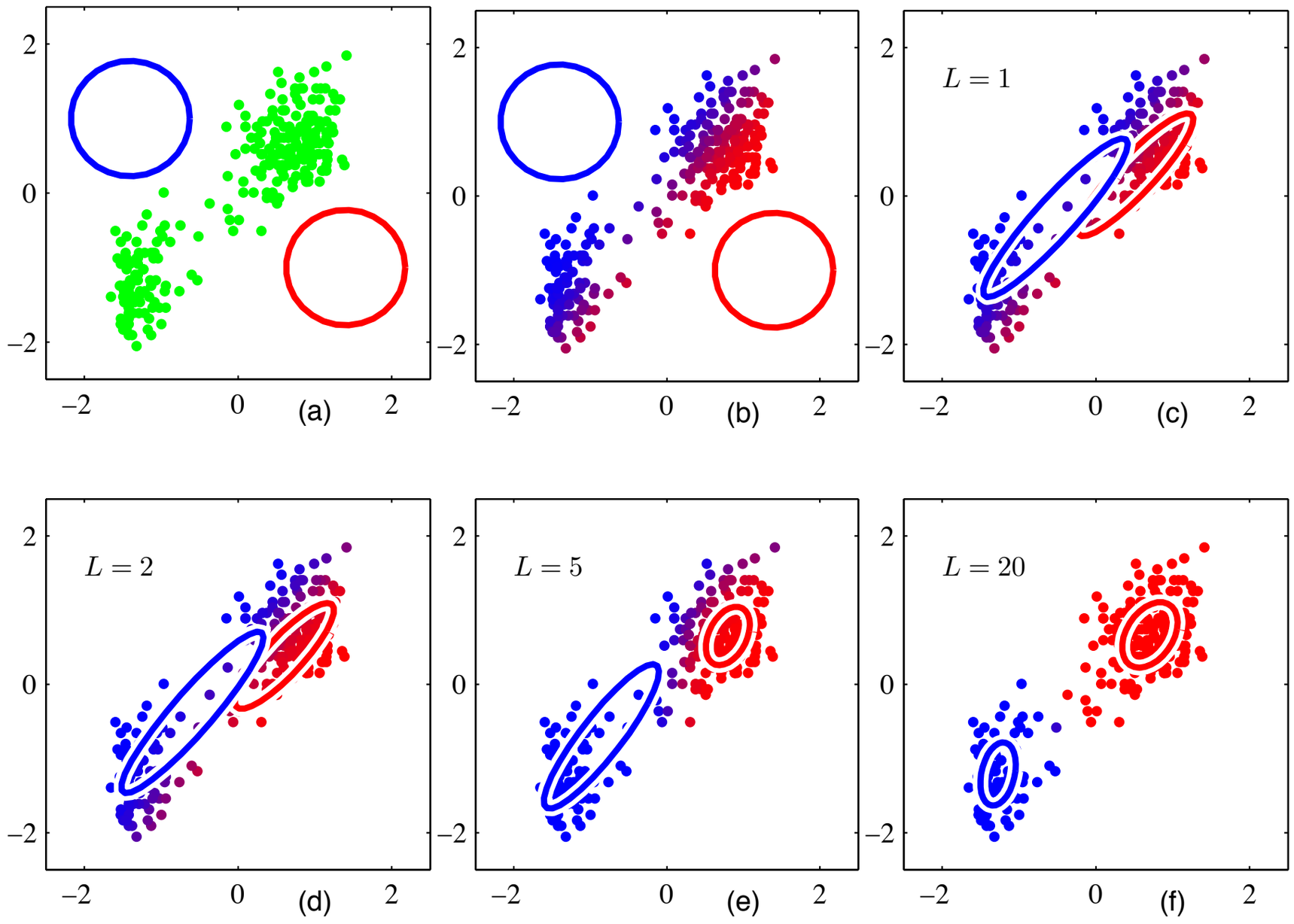
Initialization

- Clearly, the performance of EM depends strongly on the initialization of parameters $\theta = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, p_k\}_{k=1}^{K}$ and number of components K
- For θ, it is common to **run k-means** to **initialize EM**: covariances can be initialized to the sample covariance of the clusters found by k-means, the mixing coefficients can be set to the fractions of cluster points
- This makes sense because EM is typically much slower to converge and more expensive to compute
- As with k-means, K can be determined by running EM with **different values** of K minimizing a proper **model selection criterion** (e.g. BIC)
- **K-means can be derived from EM** for the case of spherical covariances of equal constant size ε for all components. Then, if we consider the limit $\varepsilon \to 0$, the responsibilities γ_{ik} become the hard assignments r_{nk}, the data log-likelihood becomes the distortion measure

Gaussian Mixture Models

Examples

- Point colors indicate γ_{ik}

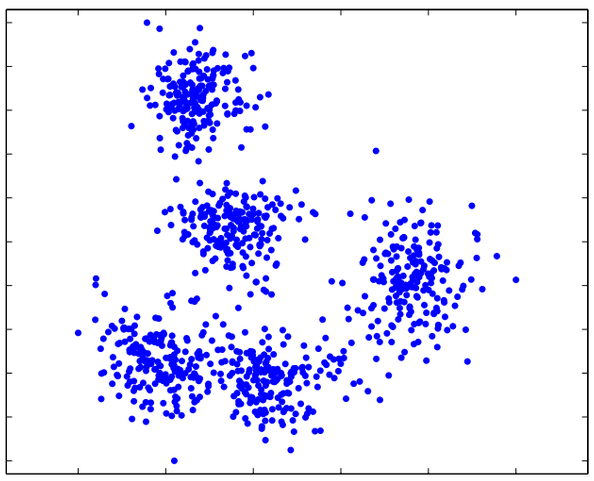


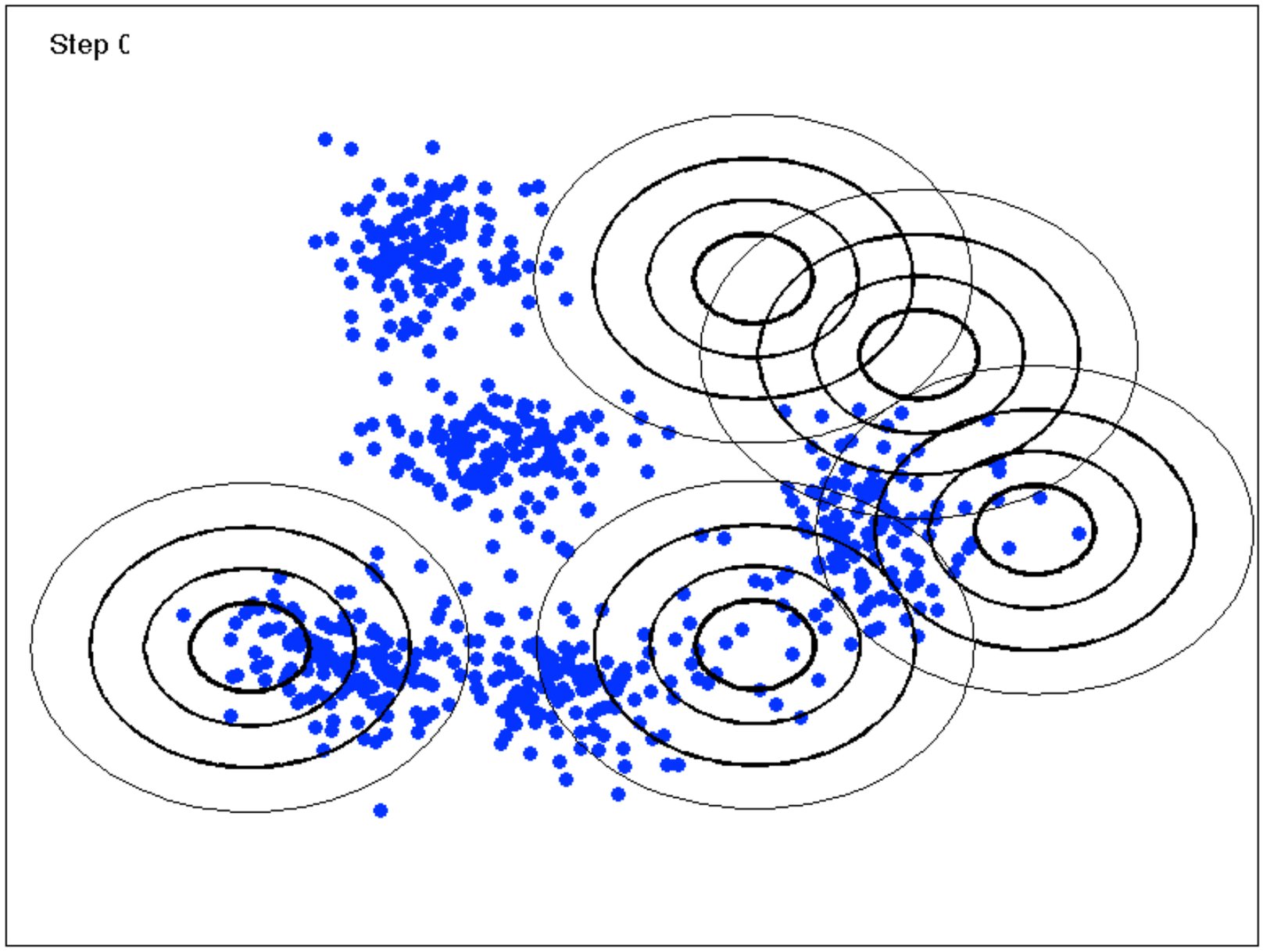
Source [2]

Gaussian Mixture Models

Examples

- $K = 5$
- Randomly initialized components with spherical covariances

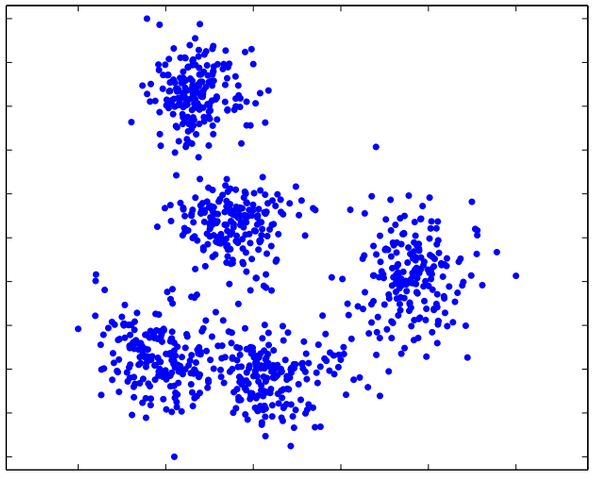


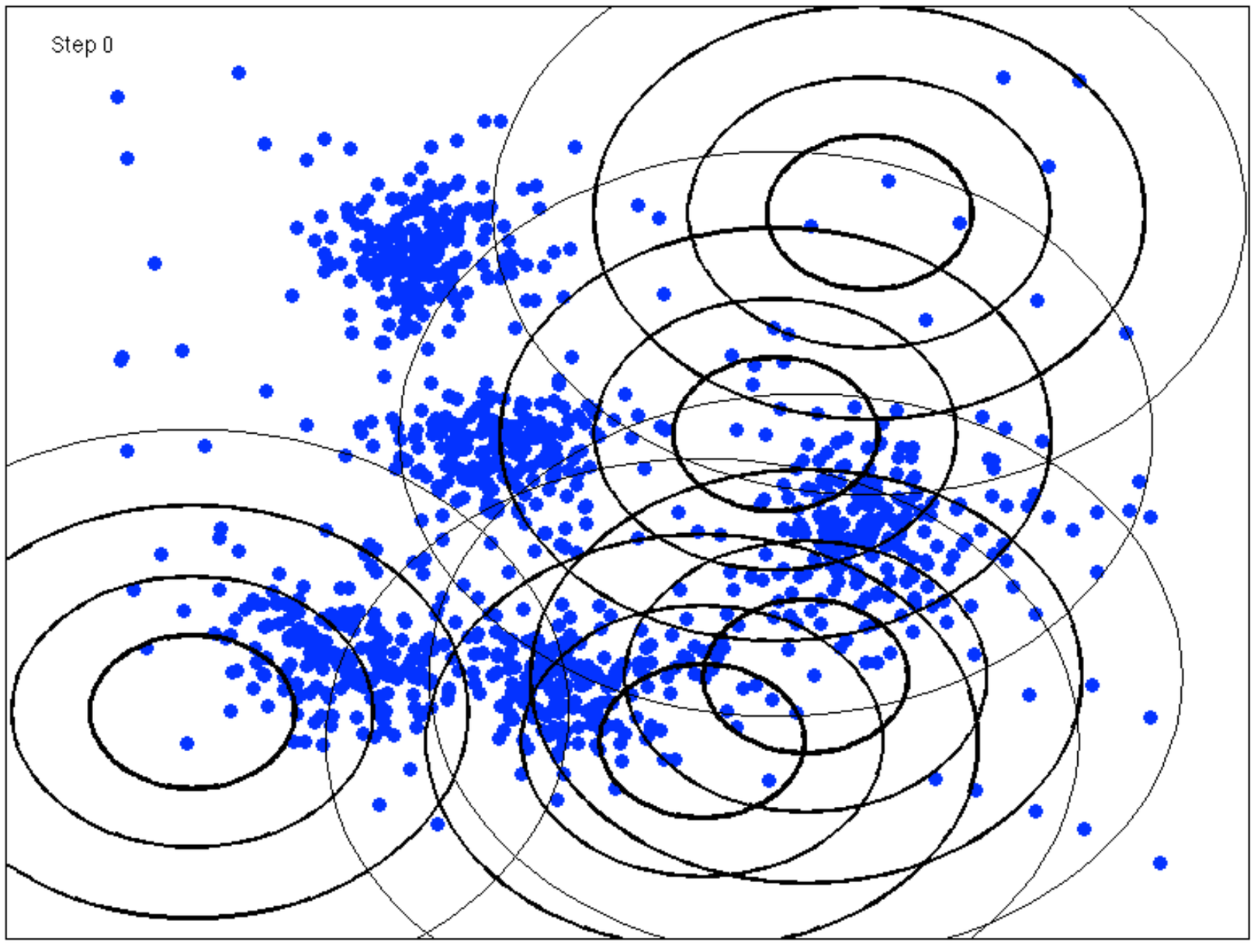


Gaussian Mixture Models

Examples

- $K = 5$
- Randomly initialized components with spherical covariances



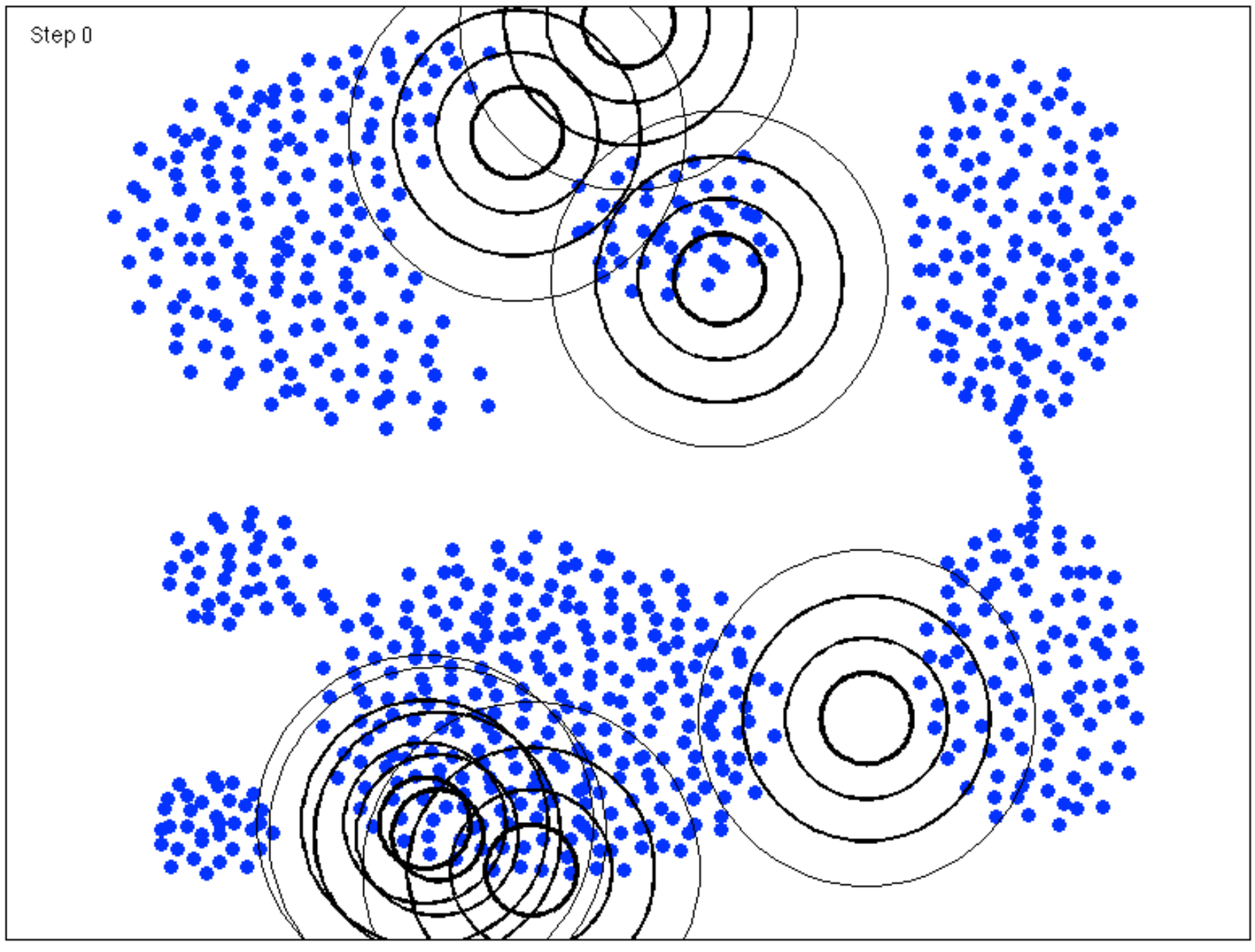


Gaussian Mixture Models

Human-Oriented Robotics
Prof. Kai Arras
Social Robotics Lab

Examples

- $K = 7$
- Randomly initialized components with spherical covariances

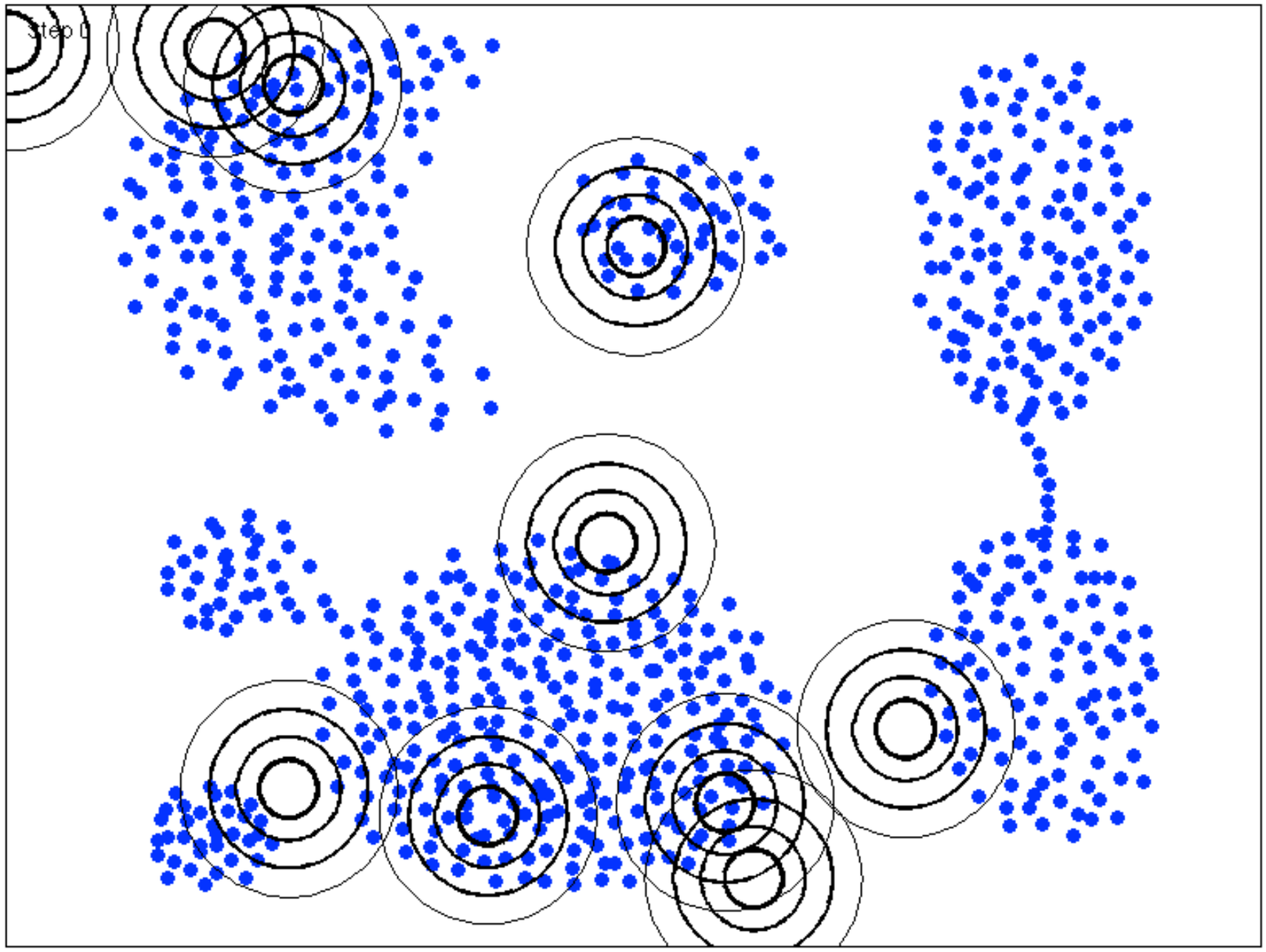


Aggregation data set [7]

Gaussian Mixture Models

Examples

- $K = 10$
- Randomly initialized components with spherical covariances



Aggregation data set [7]

Cluster Validity

- Once a clustering result has been obtained, how can we **evaluate it**?
- This is the task of **cluster validity**: evaluating the results in a quantitative and objective fashion. There are **internal**, **relative**, and **external** criteria:

1. **Internal criteria** assess the fit between the structure imposed by the clustering algorithm and the data using the data alone. E.g. test on low intra-cluster distances and high inter-cluster distances. Notice, certain criteria may favor certain types of structures
2. Indices based on **relative criteria** compare multiple structures (generated by different algorithms, for example) and decide which of them is better in some sense
3. **External indices** measure the performance by matching the clustering result to ground truth information (labels!). Uses performance measures from supervised learning

Unsupervised Learning

Summary

- Unsupervised learning is finding **hidden structures** in unlabeled data
- Clustering, the most prominent unsupervised learning problem, is trying to group data in a way that **intra-group distances** are **small** and **inter-group distances** are **large**

- **Hierarchical clustering**
  - Builds a hierarchy of clusters
  - Forms clusters by connecting points based on their distance ("connectivity-based clustering")
  - Does not optimize a global objective function, decisions are made local when merging clusters
  - Easy to understand and implement
  - Merges are final, hard assignments, not robust to noise, costly

Summary

- **K-Means**
  - Clusters are represented by centroids (“centroid-based clustering”)
  - Two-step linear complexity iterative algorithm, converges quickly to a local optimum
  - Speed and simplicity of k-means make it appealing, not its accuracy.
    Cannot deal with non-globular clusters, problem of cutting borders
  - Very sensitive to initial conditions, hard assignments, not robust to noise
  - Finding K automatically is typically framed as a model selection problem
  - Many extensions (k-means++, X-means, kernel k-means, etc.)

- **Gaussian Mixture Models**
  - Probabilistic view of clustering, posed as a parametric density estimation problem (“distribution-based clustering”)
  - GMM use EM for learning, a two-step iterative algorithm, converges to local optimum

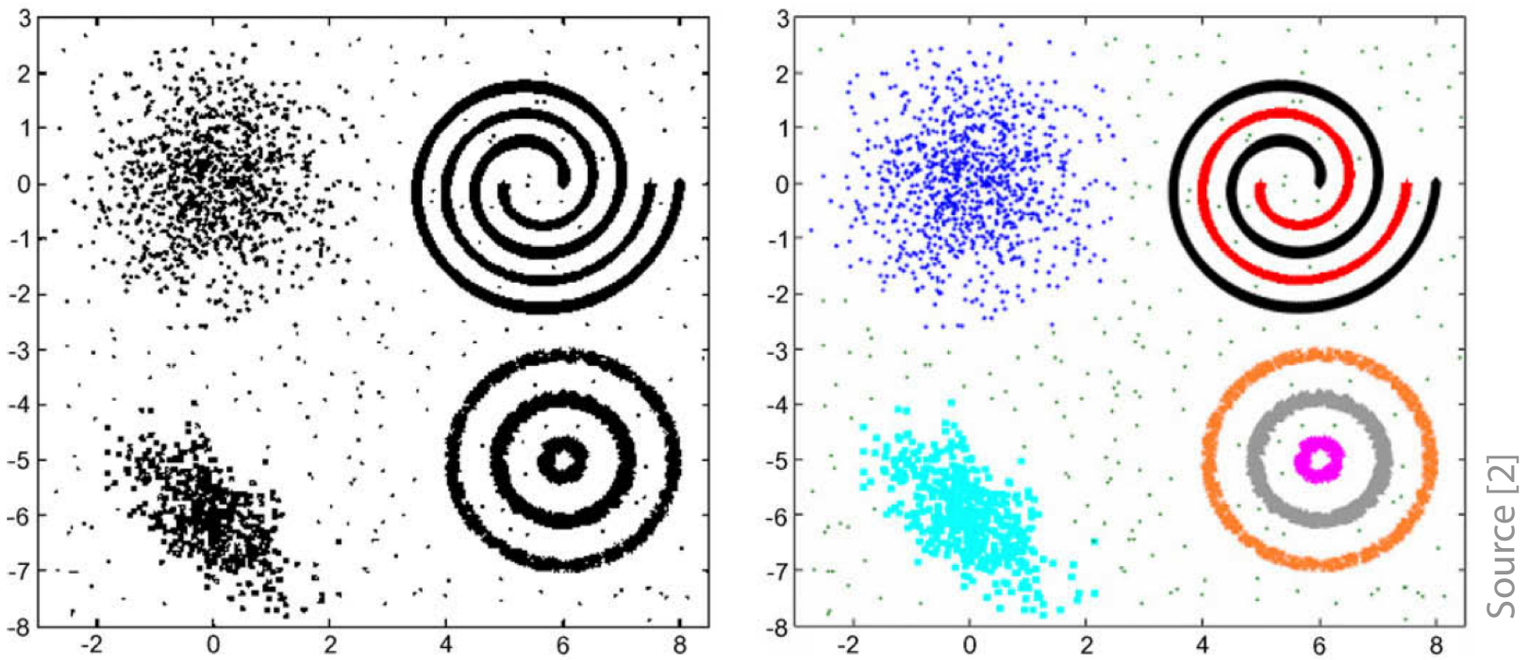
Summary

- **Gaussian Mixture Models** (cont.)
  - Soft assignments, some robustness to noise
  - Very sensitive to initial conditions, use k-means to initialize EM
  - K-means is a special case of EM

- **Which is the best clustering algorithm?** Cannot be answered, depends on the task/data
- Each clustering algorithm imposes a structure on the data either explicitly or implicitly. When there is a **good match** between that model and the data, good partitions are obtained
- Since the structure of the data is not known a priori: **trial and error,** use cluster validation

Summary

- **Current trends** in clustering: very large N (millions) at high dimensions (thousands), cluster ensembles, semi-supervised clustering, etc.



Source [2]

- "None of the available clustering algorithms can detect all these clusters" (A.K. Jain, "Data clustering: 50 years beyond K-means," Pattern Recognition Letters, 31(8), 2010). Excellent article!

Sources and Further Reading

These slides follow and contain material from the books by Theodoridis and Koutroumbas [1] (chapters 11-15), Bischop [2] (chapter 9), Prince [3] (chapter 7), Alpaydin [4] (chapter 7) as well as the Wikipedia article on cluster analysis [5]. Regarding feature preprocessing, see also the recent paper by Coates et al. [6]. An excellent article to read in this context is Jain [8]. The on-line Java applets on k-means [9] and GMM [10] are very instructive.

[1] S. Theodoridis, K. Koutroumbas, "Pattern Recognition", 4th ed., Elsevier, 2009. Online: http://cgi.di.uoa.gr/~stpatrec/Welcome.html (Dec 2013)

[2] C.M. Bischop, "Pattern Recognition and Machine Learning", Springer, 2nd ed., 2007. See http://research.microsoft.com/en-us/um/people/cmbishop/prml

[3] S.J.D. Prince, "Computer vision: models, learning and inference", Cambridge University Press, 2012. See www.computervisionmodels.com

[4] E. Alpaydin, "Introduction to Machine Learning", The MIT Press, 2009. See http://www.cmpe.boun.edu.tr/~ethem/i2ml2e

[5] Wikipedia, "Cluster analysis" article. Online: http://en.wikipedia.org/wiki/Cluster_analysis

Sources and Further Readings

[6] A. Coates, A. Y. Ng, H. Lee, "An analysis of single-layer networks in unsupervised feature learning," AISTATS 2011

[7] Clustering datasets, Speech and Image Processing Unit, University of Eastern Finland. Online: http://cs.joensuu.fi/sipu/datasets/ (Dec 2013)

[8] A. K. Jain, "Data clustering: 50 years beyond K-means," Pattern Recognition Letters, vol. 31, no. 8, 2010

[9] E.M. Mirkes, K-means and K-medoids Applet, University of Leicester, 2011. Online: http://www.math.le.ac.uk/people/ag153/homepage/KmeansKmedoids/Kmeans_Kmedoids.html (Dec 2013)

[10] I. Dinov, "EM for Mixture Models Applet", online: http://www.socr.ucla.edu/Applets.dir/MixtureEM.html (Dec 2013)