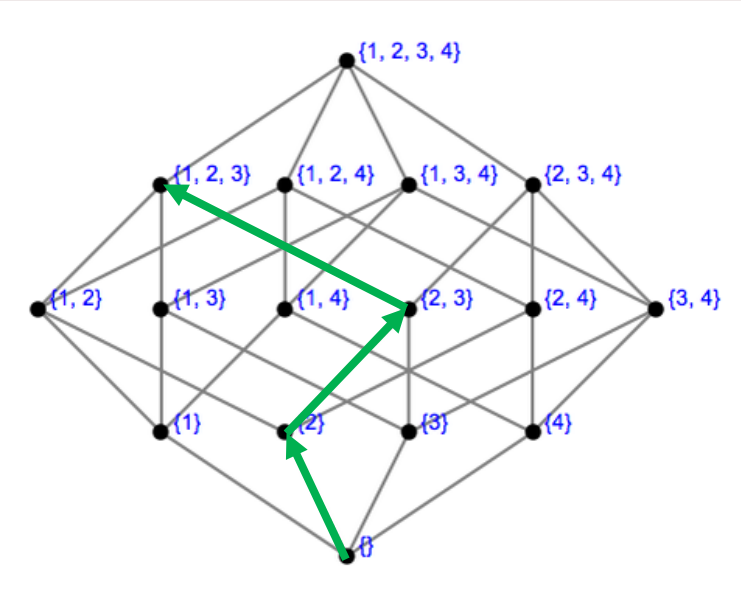
# Analysis II: Clustering & unsupervised learning

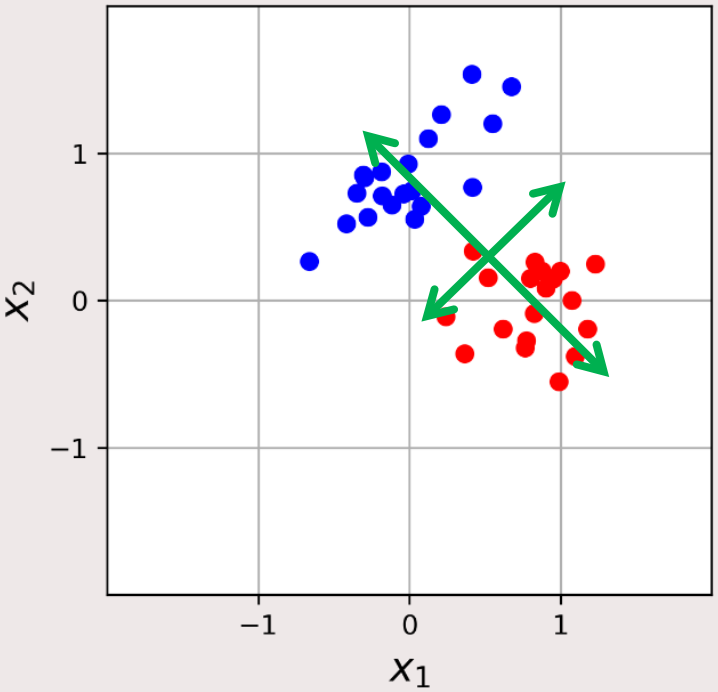
5ARB0: DATA ACQUISITION & ANALYSIS

Wouter Kouw

Dept EE – Signal Processing Systems group

# Recap: feature selection & extraction



# Outline

## Clustering

- When to cluster?
- Hierarchical clustering
- K-means clustering

## Unsupervised learning

- Expectation-maximization
- Gaussian mixture model

# Clustering

**Clustering refers to a family of techniques that groups data points based on similarity.**

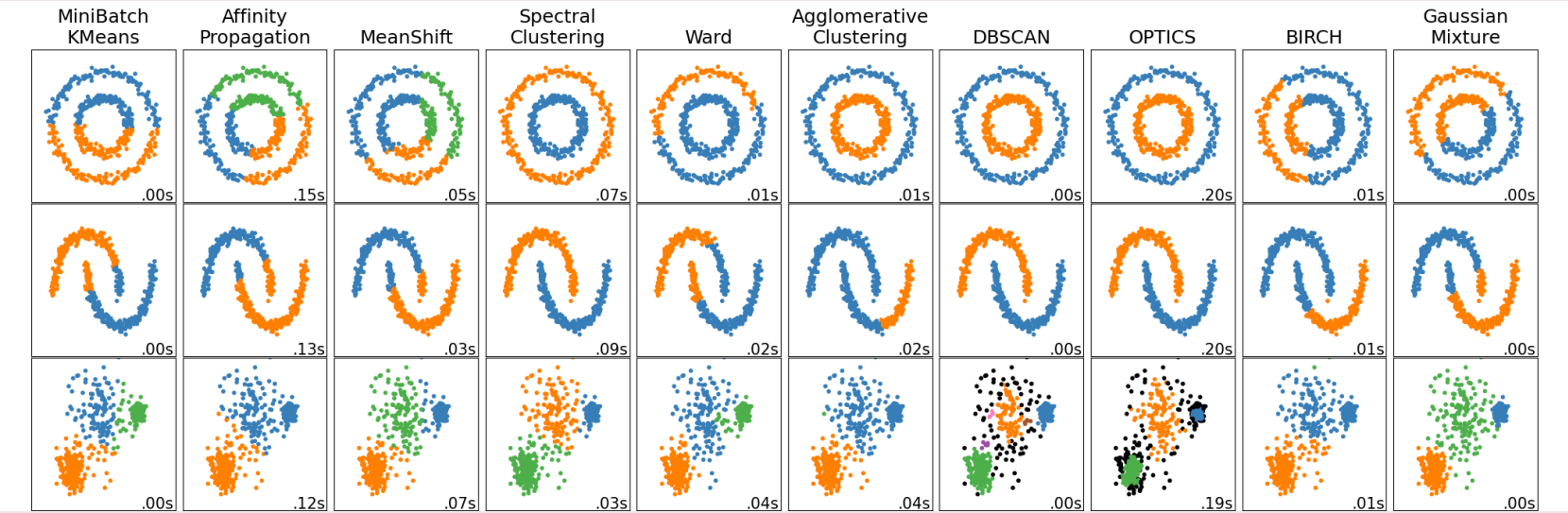
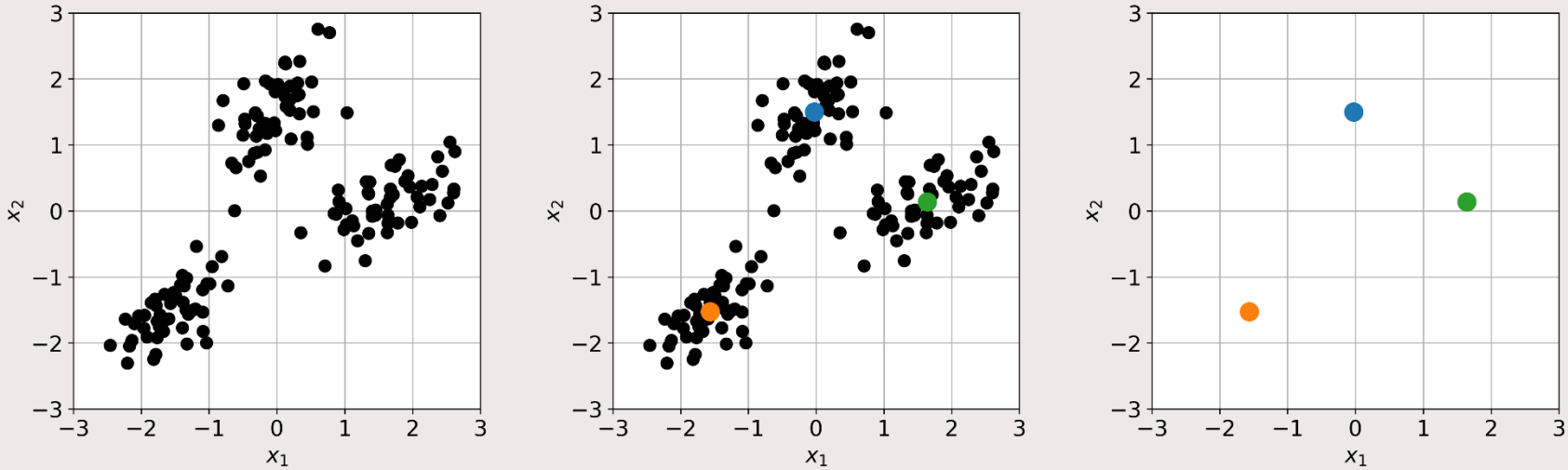



Figure taken from *Scikit-learn: Comparing clustering algorithms (link)*.

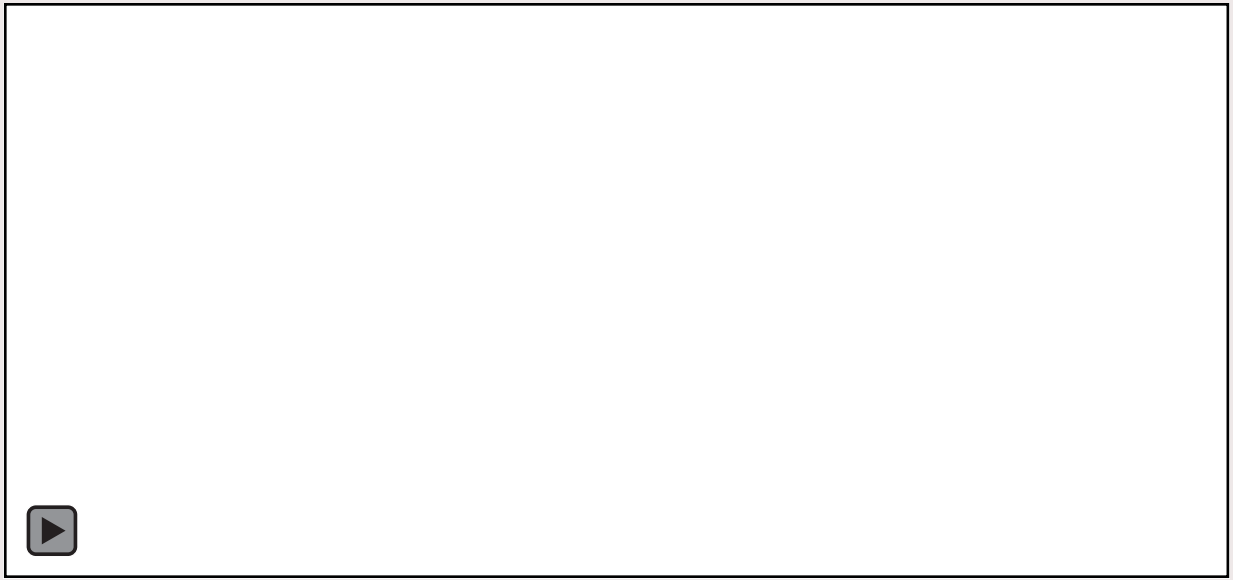
# When to cluster?

**Data compression: Groups of points can be replaced by representative samples.**



# When to cluster?

**Visualization: High-level overviews can be constructed by merging points in clusters.**



# When to cluster?

**Subgraph detection: communities in networks can be found by clustering nodes.**

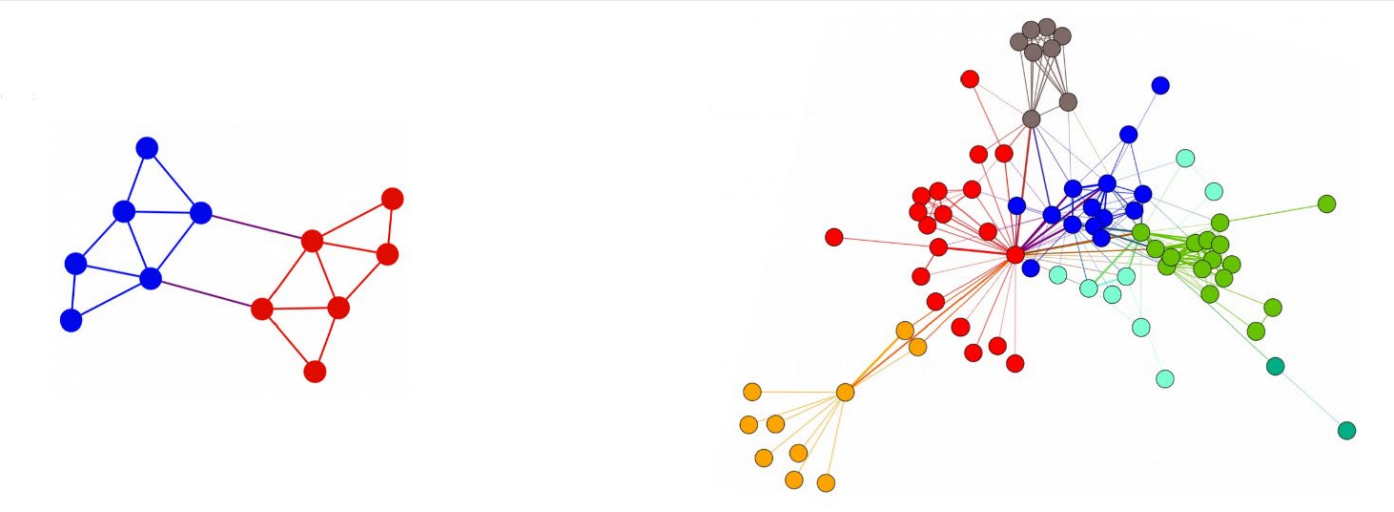
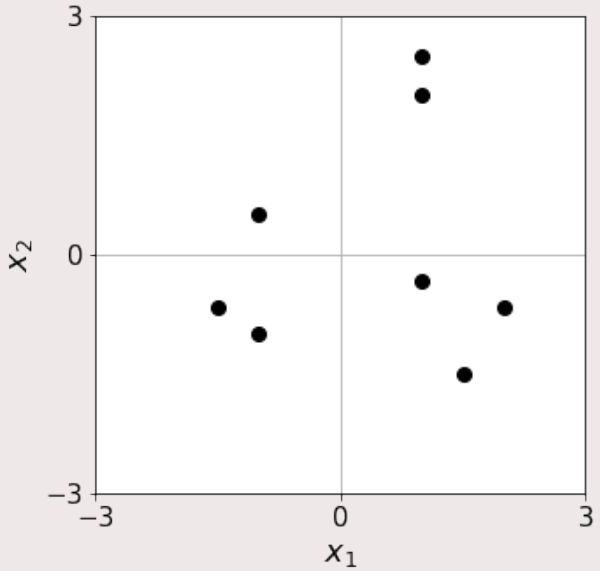


Figure taken from *Negre, Ushijima-Mwesigwa, Mniszewski (2020), Detecting multiple communities using quantum annealing on the D-Wave system.*

# Clustering

**Q: How would you cluster the following data set?**




- **How many clusters to use?**
- **How big should every cluster be?**
- **What similarity metric?**
- **Select or create representative samples?**

# Hierarchical clustering

**"Hierarchical clustering" algorithms generate a *hierarchy* of clusterings of points.**

- The hierarchy is typically encoded in a tree graph.
- *Agglomerative clustering* groups points from the bottom up.
- *Divisive clustering* groups points from the top down.
- Similarity is typically expressed in terms of distances between points.

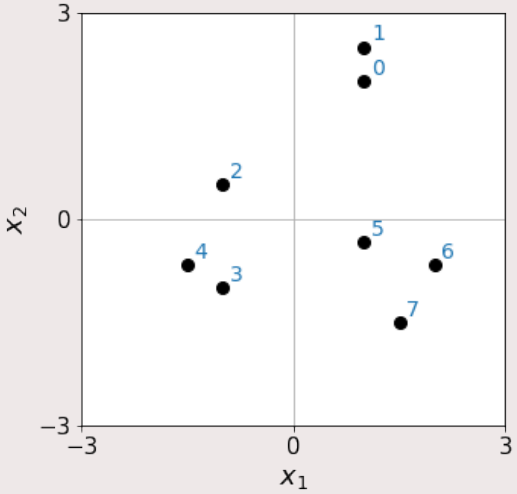
**Q: What depth will the hierarchy have?**

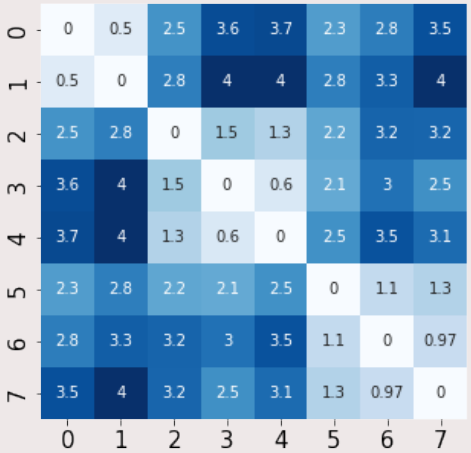
At the top is **1** cluster (all samples) and at the bottom are $\boldsymbol{N}$ clusters (1 for each sample).

# Agglomerative clustering: single-linkage

**Single-linkage is a similarity metric based on the distance between the closest two points.**

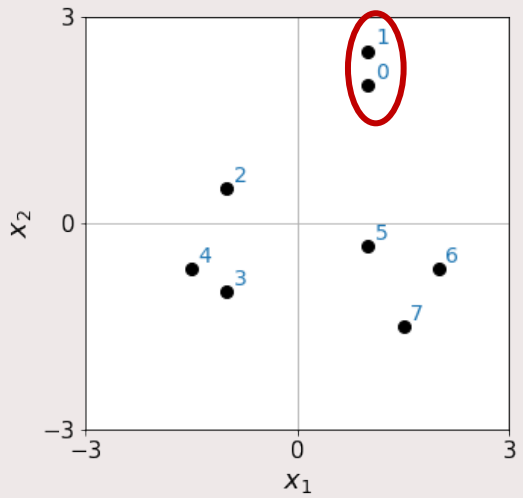
**At L(0), each sample is a cluster and cluster similarities are pure pointwise distances:**

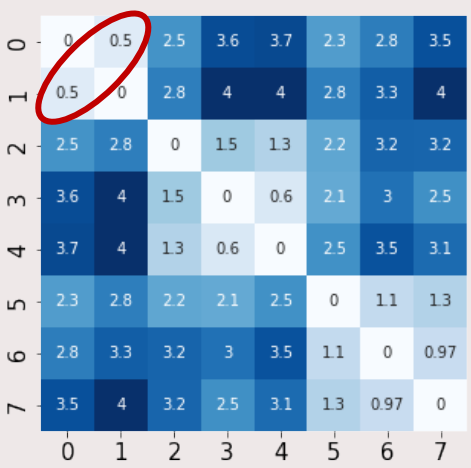



# Agglomerative clustering: single-linkage

**At L(1), we first find the closest pair of clusters based on our similarity matrix.**

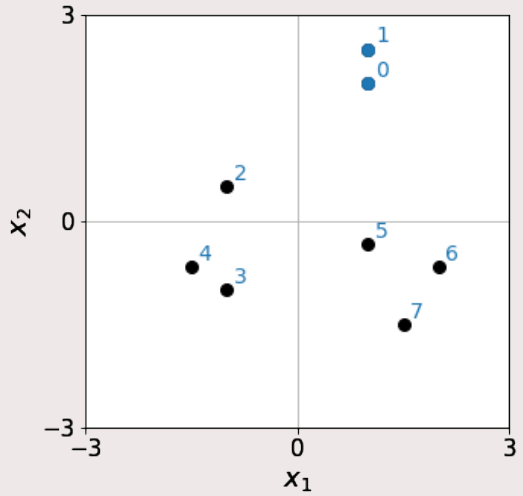


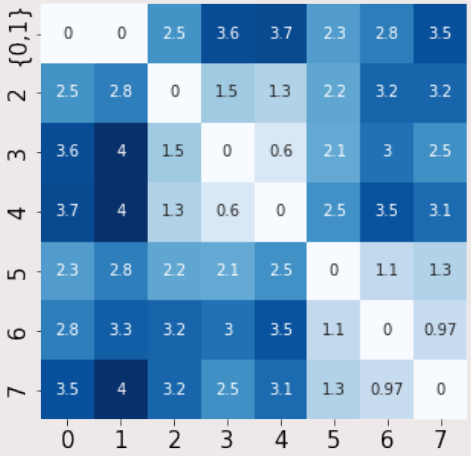



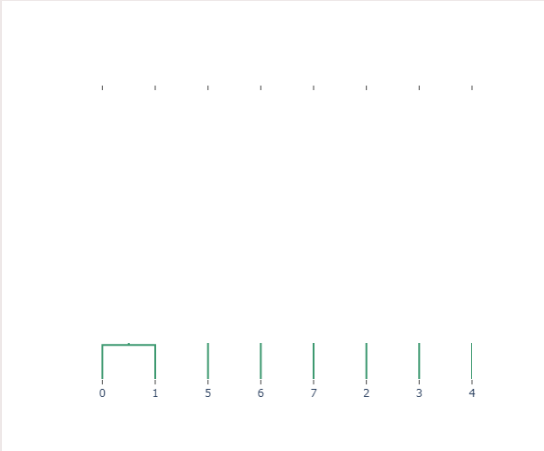

# Agglomerative clustering: single-linkage

**At L(1), we first find the closest pair of clusters based on our similarity matrix.**

**Then, we merge them based on minimal distances; $d\big(k,(i,j)\big) = \min\{d(k,i), d(k,j)\}$.**

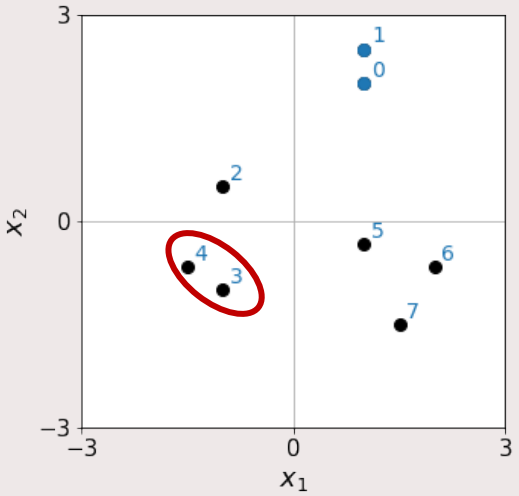


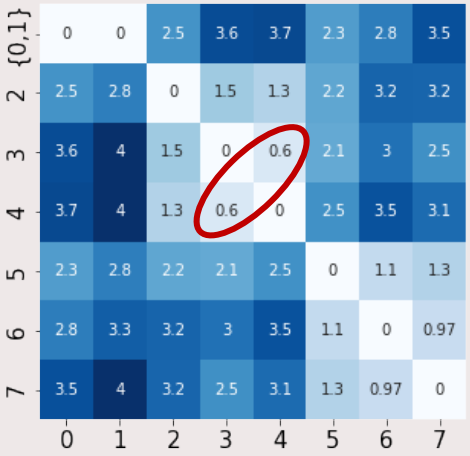



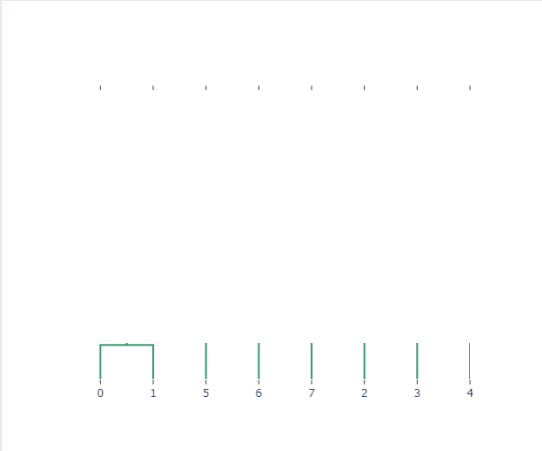



# Agglomerative clustering: single-linkage

**At L(2), we find the closest pair of clusters based on our updated similarity matrix.**



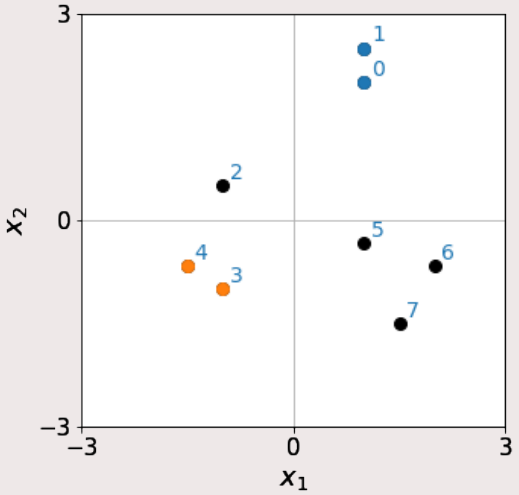



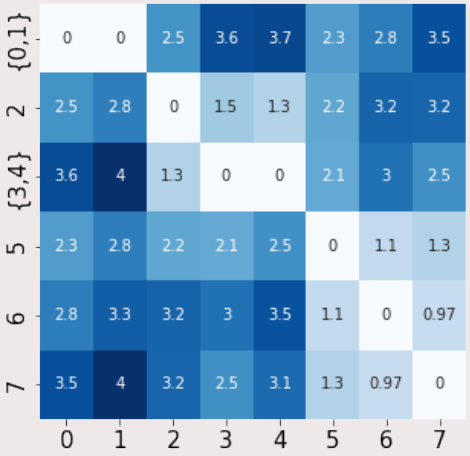



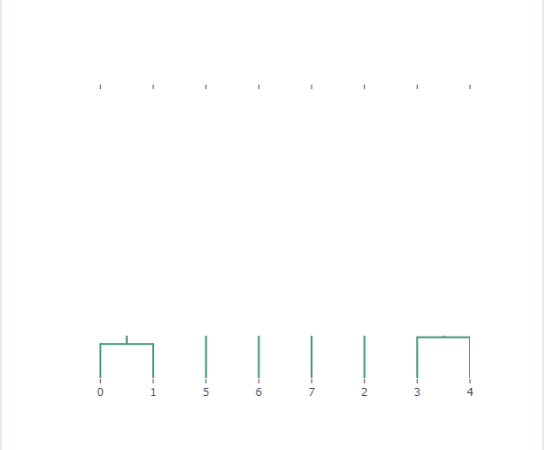
# Agglomerative clustering: single-linkage

**At L(2), we find the closest pair of clusters based on our updated similarity matrix.**

**Again, we merge the closest clusters into one, using the minima of distances.**

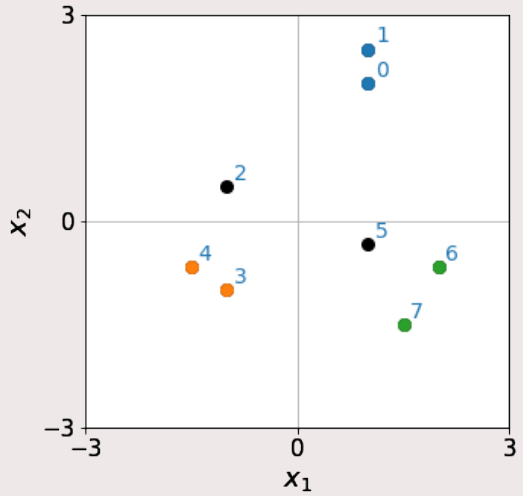


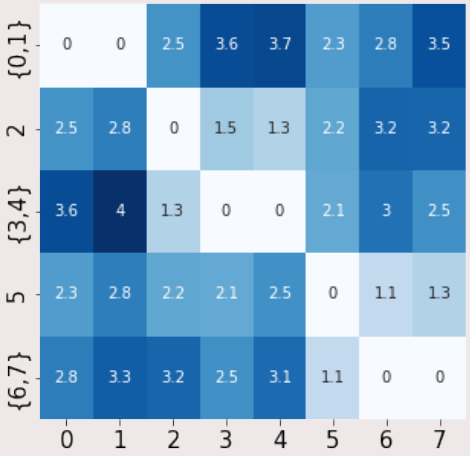



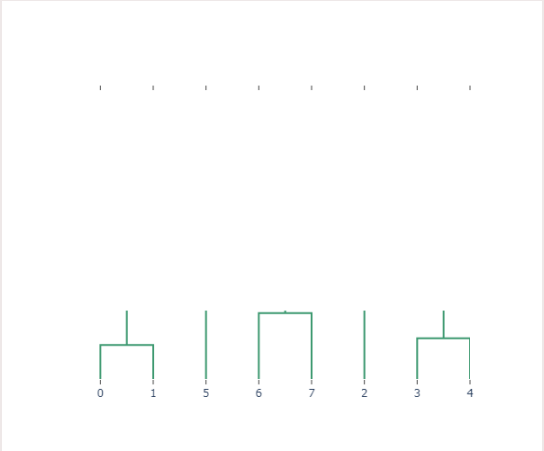



# Agglomerative clustering: single-linkage

**We repeat the same procedure to get L(3).**

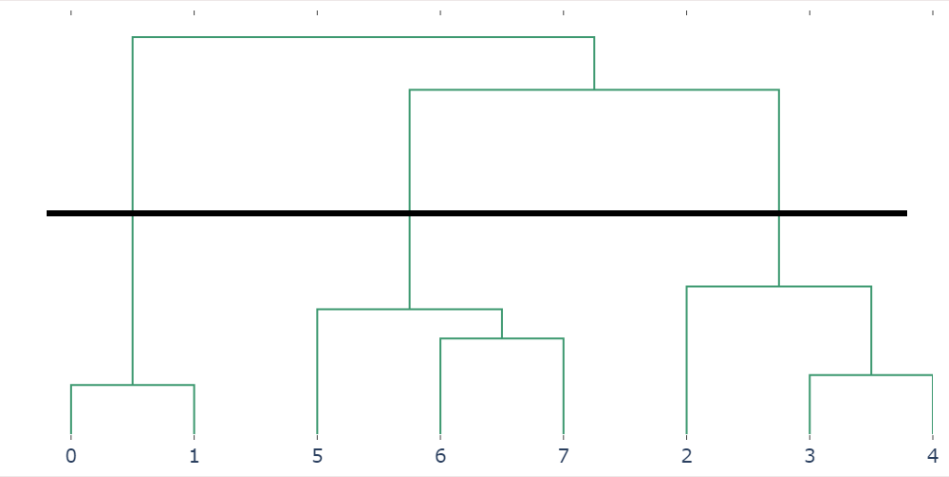


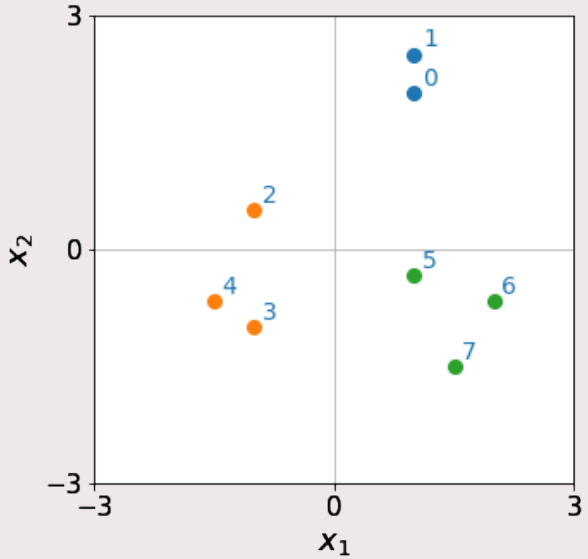



# Agglomerative clustering: single-linkage

**If we continue this procedure until L($N$), then we obtain the following tree:**




**Cutting this tree at L($m$), gives you $m$ clusters.**

# Agglomerative clustering: linkage functions

**Single-linkage is just one of many different functions used to merge clusters.**

- **"Complete linkage":** $d(k, C) = \max_{c \epsilon C} d(k, c)$.
- **"Weighted average linkage":** $d(k, \{i, j\}) = \frac{1}{2}\left(d(k, i) + d(i, j)\right)$
- **"Unweighted average linkage":** $d(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{k \in C_i} \sum_{l \in C_j} d(k, l)$
- **"Centroid linkage":** $d(C_i, C_j) = d(c_i, c_j)$ **where** $c_i, c_j$ **are cluster centroids.**
- **"Ward's": this function describes the increase in variance with each merger.**

# Limitations of hierarchical clustering

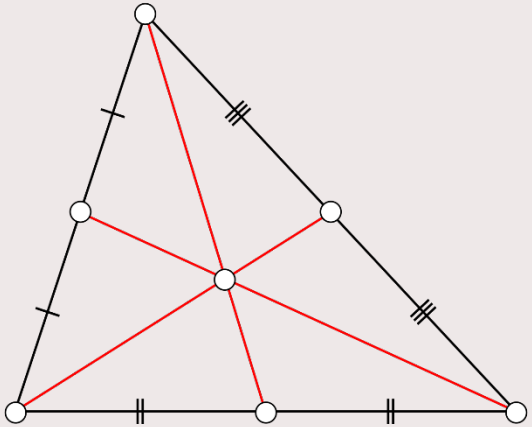
**Q: What are the limitations of hierachical clustering algorithms?**

1. **All pairwise distances are required before the algorithm can even start.**
   - Computing pairwise distances for all points is $O(\boldsymbol{N^2})$.
   - All those distances need to be kept in memory.
2. **You cannot skip steps by pre-specifying the number of clusters.**
3. **Similarity metrics are less intuitive for non-numeric data.**
   - For example, how do you compute similarity between genes?

# K-means clustering

***K*-means clustering is an algorithm that groups points based on distances to centroids.**

- A *centroid* is the average of a set of points in high-dimensional space.
- Centroids are not part of the data set.

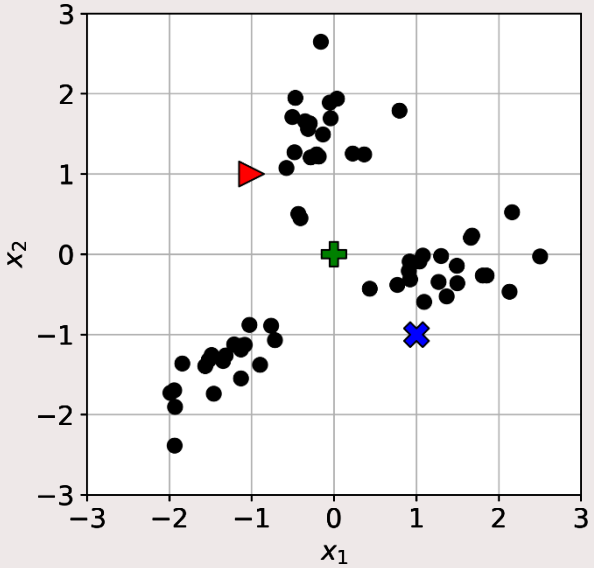


***K*-means is a simple iterative procedure consisting of two steps:**

1. Assign points to clusters based on distance to centroids.
2. Update clusters centroids based on assigned points.

# K-means clustering

**K-means is best explained through a step-by-step demonstration.**




**First, we initialize a set of $K$ centroids:**

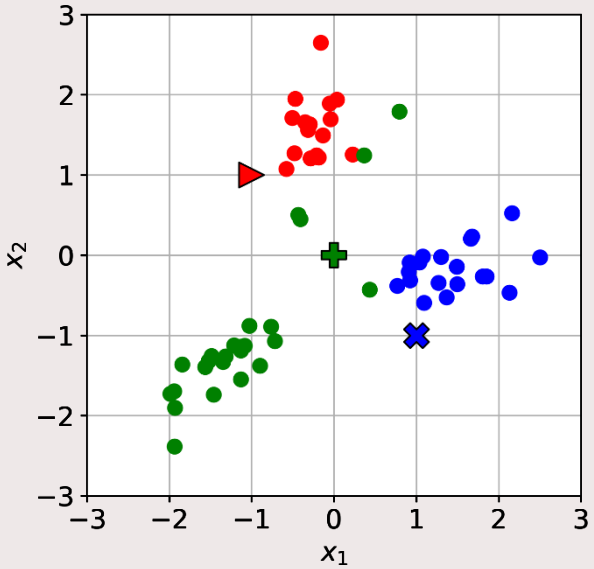
$$C = \{c_k : k = 1, \dots K\}.$$

**These points should not start at the same coordinates.**

**If so, the distances to each centroid will be the same and all points will be tied for assignments.**

# K-means clustering

**We then iterate the 2-step procedure:**




1. **Compute the squared Euclidean distance between the data and the centroids,**

$$d(x_i, c_k) = \left|\left|c_k - x_i\right|\right|^2,$$

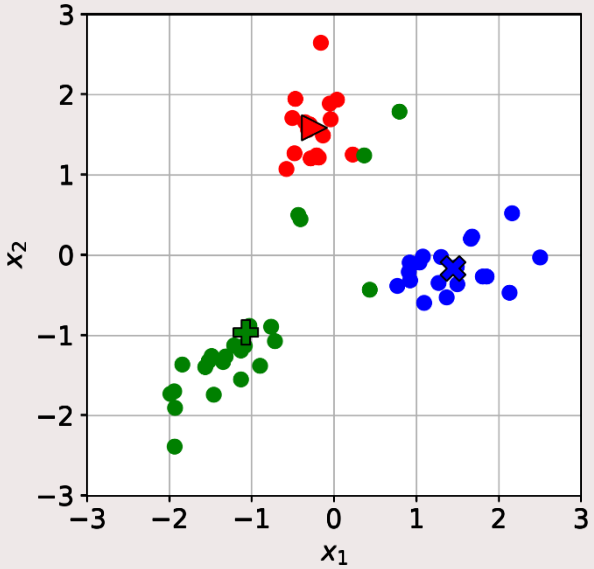
**and assign each point to the closest centroid**

$$z_i = \arg\min_k d(x_i, c_k)$$

**where $z_i$ is the assignment variable.**

# K-means clustering

**We then iterate the 2-step procedure:**




2. **Update the centroids according to:**

$$c_k = \frac{1}{|S_k|} \sum_{x_j \in S_k} x_j$$

**where $S_k$ is the set of points assigned to cluster $k$. This can alternatively be computed with:**

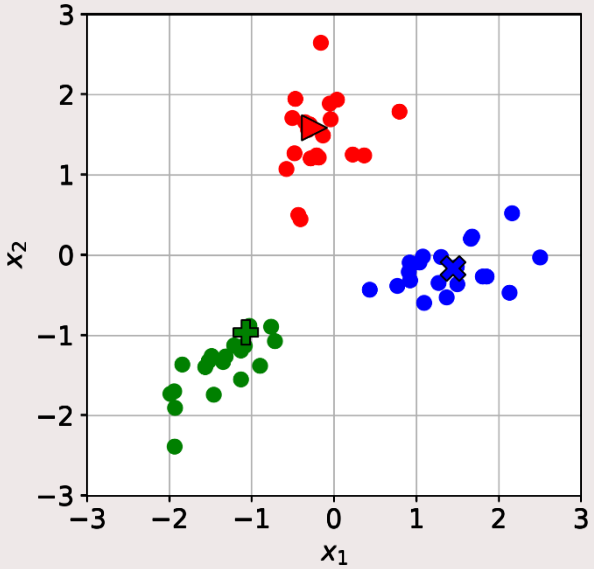
$$c_k = \sum_{i=1}^{N} \frac{[z_i = k] \cdot x_i}{|z_i = k|}.$$



TU/e

# K-means clustering

**We then iterate the 2-step procedure until convergence:**

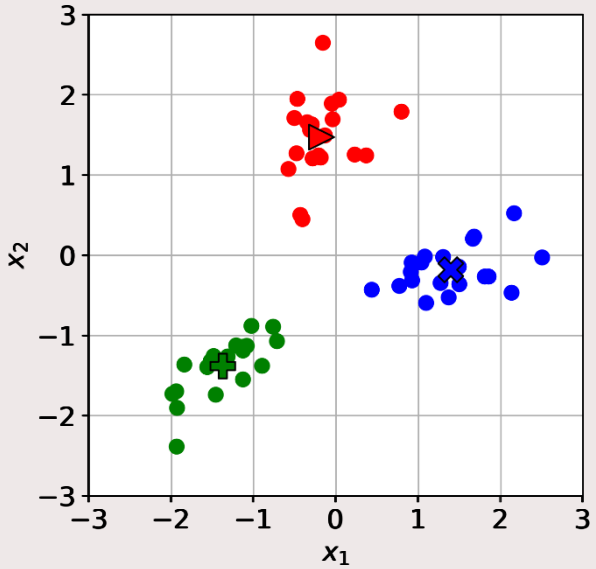



1. **Assign each data point to a cluster based on minimal Euclidean distance.**

2. Update centroids according to the arithmetic mean of all points assigned to clusters.

# K-means clustering

**We then iterate the 2-step procedure until convergence:**

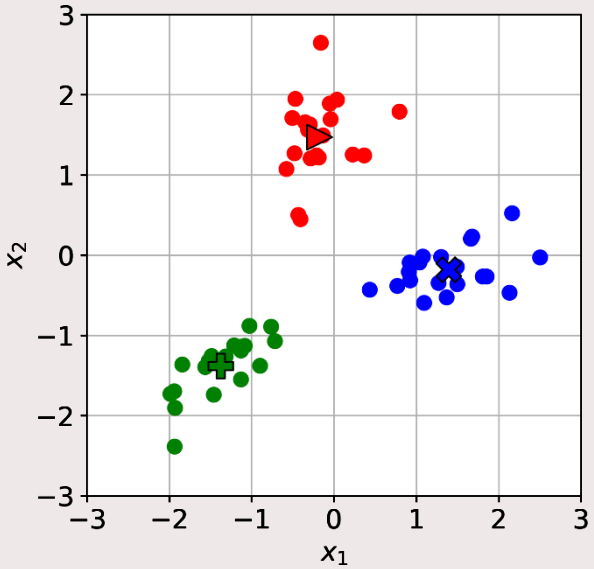



1. Assign each data point to a cluster based on minimal Euclidean distance.

2. **Update centroids according to the arithmetic mean of all points assigned to clusters.**

# K-means clustering

**We then iterate the 2-step procedure until convergence:**

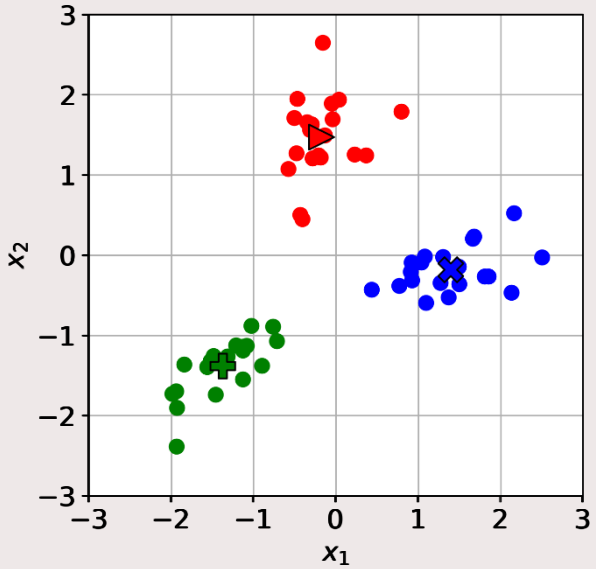



1. **Assign each data point to a cluster based on minimal Euclidean distance.**

2. Update centroids according to the arithmetic mean of all points assigned to clusters.

# K-means clustering

**We then iterate the 2-step procedure until convergence:**

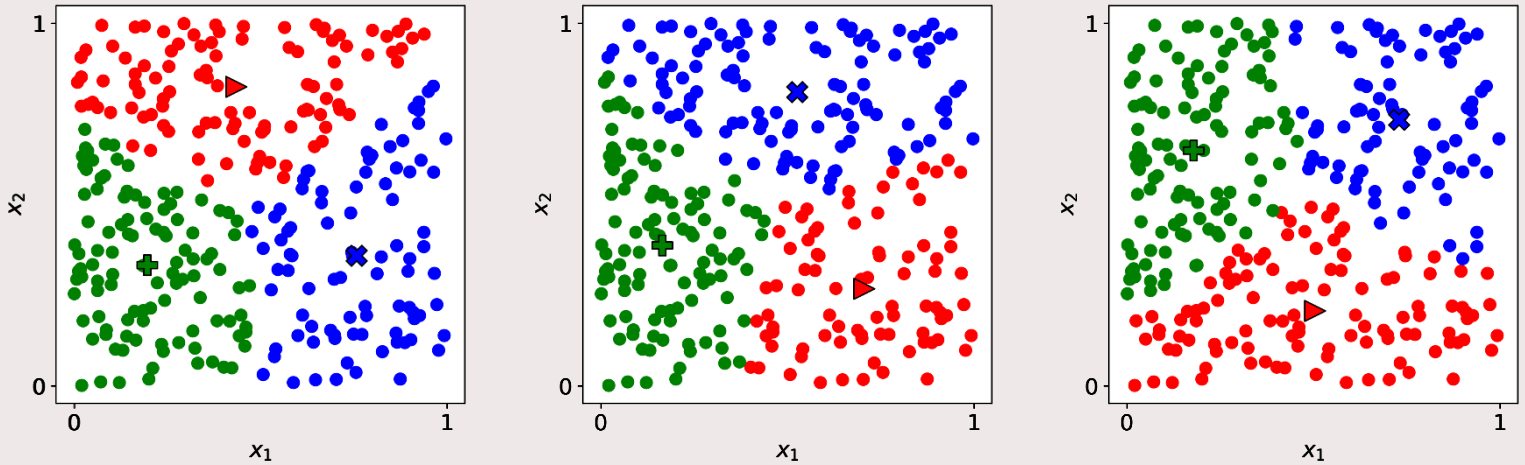



1. Assign each data point to a cluster based on minimal Euclidean distance.

2. **Update centroids according to the arithmetic mean of all points assigned to clusters.**

# Limitations of K-means

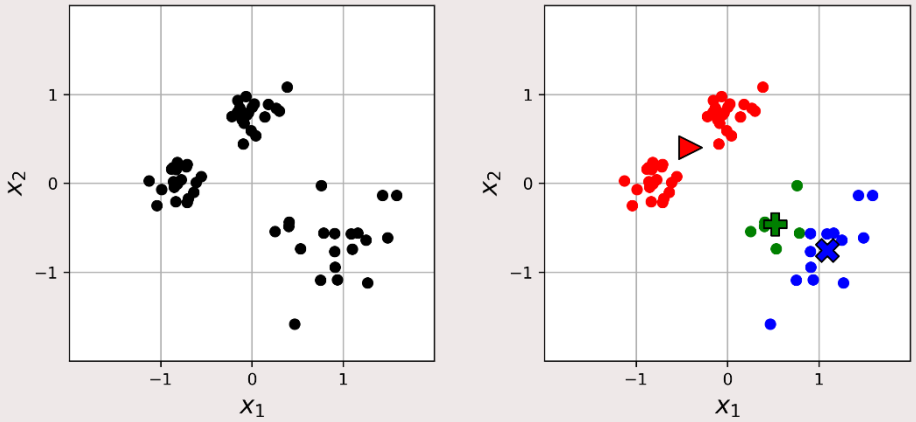
**If the data is not clustered, then the centroids are random and meaningless.**



# Limitations of K-Means

**Q: What more could go wrong with K-Means?**

**K-Means is blind to cluster size and can get stuck in unintuitive solutions.**




- **It merges the two smaller clusters on the top-left.**
- **It splits the larger cluster on the right-bottom.**

# Break

# Unsupervised learning

***Unsupervised learning* refers to improving the performance of mathematical models with respect to some task using data, but without labels or without responses.**

- Unsupervised learners can also do clustering.

**Unsupervised learning is a general framework where we often want to:**

- Find hierarchical factors, such as clusters of clusters.
- Estimate uncertainties / point spreads in latent spaces.
- Detect anomalies / outliers / surprising events.
- Generate new samples from a learned representation.

# Unsupervised learning

**Typically, the kinds of models where unsupervised learning is applied to assume that the data was generated according to a unknown or "latent" variables.**

- There is often a complicated functional relationship, characterized by parameters.

**Examples include:**

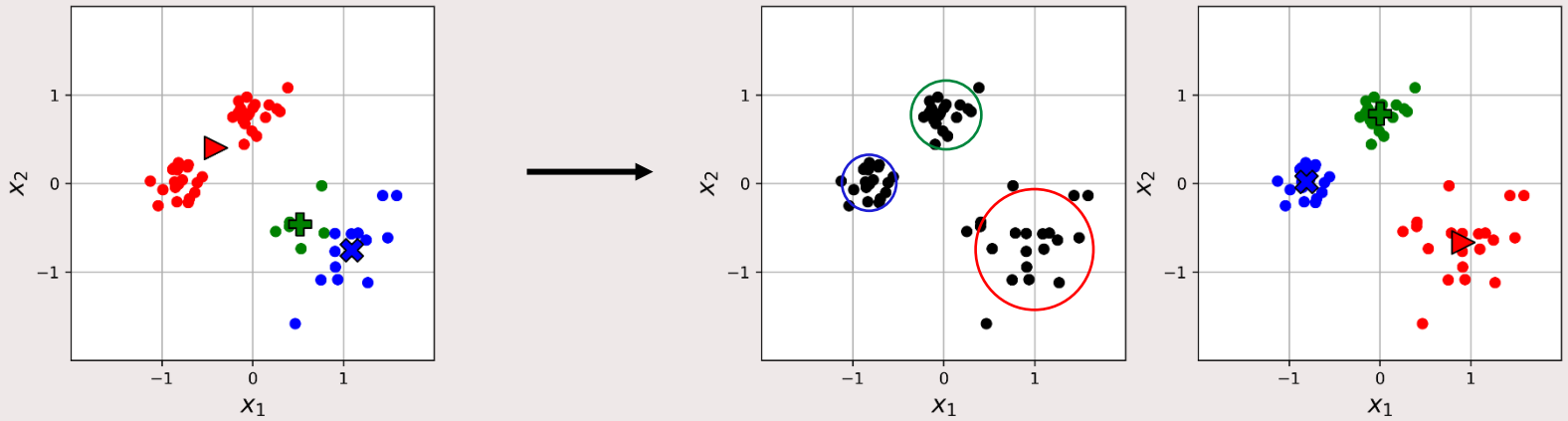
- Mixture models: the observed data is a mixture of a latent set of components.
- Autoencoders: the observed data was transformed non-linearly from latent factors.

**We "fit" models to data by optimizing the parameters with respect to the probability of the observed data or the prediction error of the model on unseen data.**

# Unsupervised learning

**We saw that K-Means fails to account for cluster size.**

**With unsupervised learning, we can take the cluster density into account as well.**



# Mixture models

**A mixture model is a probability distribution that consists of a weighted combination of $K$ "component" probability distribution functions:**

$$p(x) = \sum_{k=1}^{K} \pi_k \, p_k(x)$$

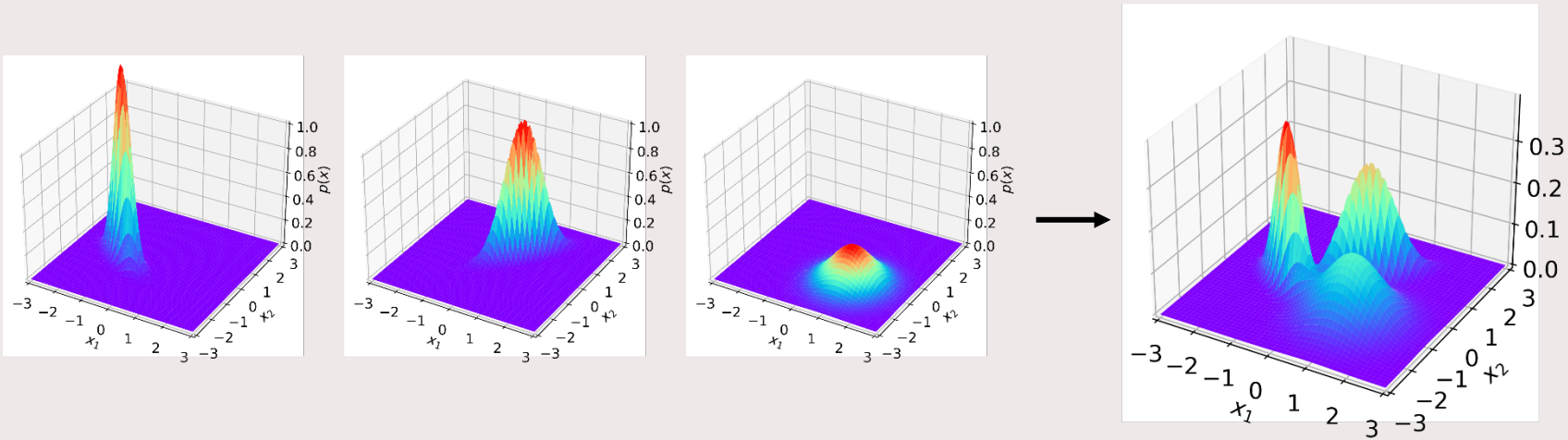
**where $p_k(x)$ is the k-th component and the $\pi_k$ are the weights.**

**The weights $\pi_k$ are fixed to be numbers between 0 and 1, that sum to 1.**

# Mixture models

**The most common variant is the *Gaussian mixture model* (GMM), which uses Gaussian probability distribution functions as components:**

$$p_k(x) = \mathcal{N}(x \mid \mu_k, \Sigma_k)$$



# Mixture models

**Q: How do we “fit” this mixture model to a data set?**

**Expectation-maximization.**

**EM is an elegant estimation procedure with roots in statistical inference.**

**For mixture models, the procedure boils down to the same steps as in K-means:**

1. “Assign” data points to component distributions.
2. Update the parameters of each component distribution.

# Mixture models

1. “Assign” data points to component distributions.

We define a latent variable called the *responsibility* of component $k$ for data point $i$:

$$r_{ik} = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}$$

The responsibilities represent the probabilities under each component normalized by the total probability, i.e., the sum of probabilities under components.

For example, for probabilities $p_1(x_i)$ = 0.2 and $p_2(x_i)$ = 0.4 with weights $\pi_1 = \pi_2 = \frac{1}{2}$,

$$r_{i1} = \frac{\frac{1}{2}0.2}{\frac{1}{2}0.2+\frac{1}{2}0.4} = \frac{1}{3} \text{ and } r_{i2} = \frac{\frac{1}{2}0.4}{\frac{1}{2}0.2+\frac{1}{2}0.4} = \frac{2}{3}.$$

# Mixture models

**In k-means, we had an assignment variable indicating which cluster the data point $x_i$ belonged to, $z_i \in \{1 \dots K\}$.**

- That can be re-written as a one-hot vector, e.g., $\tilde{z}_i = [0\ 0\ 1]$.
- That resembles the responsibilities, e.g., $r_i = [0.2\ \ 0.3\ \ 0.5]$.

**The responsibilities are often considered "soft assignments"; the data points do not belong to only one component, but rather to all components simultaneously.**

- This ensures robustness to outlying data points.

# Mixture models

2. Update parameters of each component distribution.

The likelihood of a GMM with $K$ components on $N$ samples is:

$$p(X|\theta) = \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)$$

where $\theta = (\pi_1, \dots \pi_K, \mu_1, \dots \mu_K, \Sigma_1, \dots \Sigma_K)$.

We will use maximum-likelihood to estimate the parameters:

$$\theta^* = \arg\max_{\theta} \ln p(X|\theta)$$

where $\ln$ represents the natural logarithm.

# Mixture models

**Q: Why do we maximize the log-likelihood and not just the likelihood?**

**Because products become sums and exponents become multiplications:**

$$\ln p(X|\theta_k) = \sum_{i=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)$$

**Note that the maximizer is invariant to the logarithm,**

$$\arg\max_{\theta} \ln p(X|\theta) = \arg\max_{\theta} p(X|\theta)\,,$$

**because the logarithm is a monotonically increasing function.**

# Mixture models

**First we take the derivative of the likelihood with respect to the means:**

$$\frac{\partial}{\partial \mu_k} \ln p(X|\theta) = \sum_{i=1}^{N} \frac{1}{p(x_i|\theta)} \frac{\partial p(x_i|\theta)}{\partial \mu_k}$$

$$= \sum_{i=1}^{N} \frac{1}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)} \pi_k (x_i - \mu_k)^T \Sigma_k^{-1} \mathcal{N}(x_i|\mu_k, \Sigma_k)$$

**Using the definition of the responsibilities, we get:**

$$\frac{\partial}{\partial \mu_k} \ln p(X|\theta) = \sum_{i=1}^{N} r_{ik} (x_i - \mu_k)^T \Sigma_k^{-1}$$

# Mixture models

**If we set that derivative to 0, we get:**

$$\sum_{i=1}^{N} r_{ik}(x_i - \mu_k)^T \Sigma_k^{-1} = 0$$

$$\sum_{i=1}^{N} r_{ik} x_i = \sum_{i=1}^{N} r_{ik} \mu_k$$

$$\mu_k = \sum_{i=1}^{N} \frac{r_{ik} x_i}{\sum_{i=1}^{N} r_{ik}}$$

**which resembles the arithmetic mean operation in k-means.**

# Mixture models

**Next, we have take the derivative of the likelihood with respect to the covariances:**

$$\frac{\partial}{\partial \boldsymbol{\Sigma}_k} \ln p(\boldsymbol{X}|\boldsymbol{\theta}) = \sum_{i=1}^{N} \frac{1}{p(\boldsymbol{x}_i|\boldsymbol{\theta})} \frac{\partial p(\boldsymbol{x}_i|\boldsymbol{\theta})}{\partial \boldsymbol{\Sigma}_k}$$

**This follows the same line of reasoning as with the update for the means but with much more terms and steps.**

**I will skip these and give the final result directly:**

$$\boldsymbol{\Sigma}_k = \sum_{i=1}^{N} \frac{\boldsymbol{r}_{ik}}{\sum_{i=1}^{N} \boldsymbol{r}_{ik}} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T (\boldsymbol{x}_i - \boldsymbol{\mu}_k)$$

# Mixture models

**Lastly, for the mixture weights, we also need to consider the constraint that they should sum to 1, which we can do by forming a Lagrangian:**

$$\mathcal{L} = \ln p(X|\theta) + \lambda\left(\sum_{k=1}^{K} \pi_k - 1\right)$$

**The optimal solution can be obtained by setting the partial derivatives with respect to $\pi_k$ and $\lambda$ to 0 and solving simultaneously:**

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = 0 \text{ and } \frac{\partial \mathcal{L}}{\partial \lambda} = 0$$

# Mixture models

**The first partial derivative is:**

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = \sum_{i=1}^{N} \frac{1}{p(x_i|\theta)} \frac{\partial p(x_i|\theta)}{\partial \pi_k} + \frac{\partial \mathcal{L}}{\partial \pi_k} \lambda \left( \sum_{k=1}^{K} \pi_k - 1 \right)$$

$$= \sum_{i=1}^{N} \frac{\mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)} + \lambda$$

**We can again use our definition of the responsibilities to get:**

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = \sum_{i=1}^{N} \frac{r_{ik}}{\pi_k} + \lambda$$

# Mixture models

**The second partial derivative is:**

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \sum_{k=1}^{K} \pi_k - 1$$

**Setting both to 0 yields the system of equations:**

$$\sum_{i=1}^{N} \frac{r_{ik}}{\pi_k} = -\lambda$$

$$\sum_{k=1}^{K} \pi_k = 1$$

# Mixture models

**Re-arranging the first equation gives:**

$$-\sum_{i=1}^{N}\frac{r_{ik}}{\lambda}=\pi_k$$

**We may plug this result in the second equation and solve for $\lambda$:**

$$-\sum_{k=1}^{K}\sum_{i=1}^{N}\frac{r_{ik}}{\lambda}=1$$

$$-\frac{N}{\lambda}=1$$

$$\lambda=-N$$

# Mixture models

**Plugging the optimal Lagrange multiplier into the partial derivative gives:**

$$\sum_{i=1}^{N} \frac{r_{ik}}{\pi_k} - N = 0$$

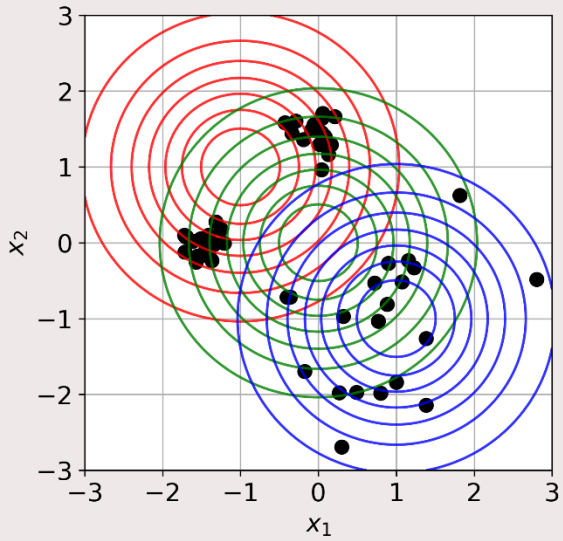
**Which leads to a final result of:**

$$\pi_k = \frac{1}{N} \sum_{i=1}^{N} r_{ik}$$

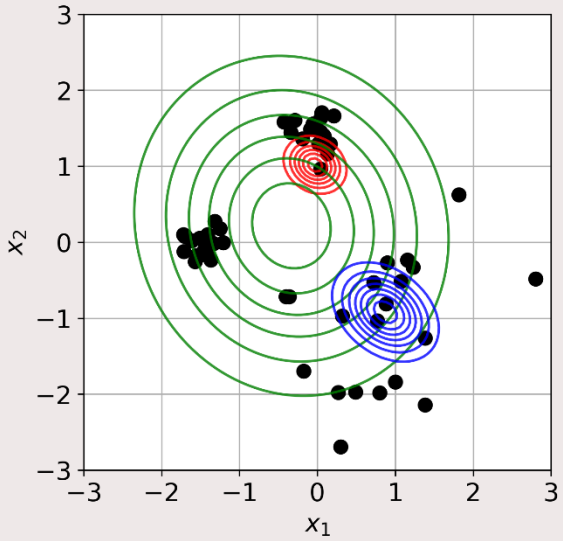
**So, the mixture weights can be interpreted as the "average responsibility" of each component.**

# Mixture models

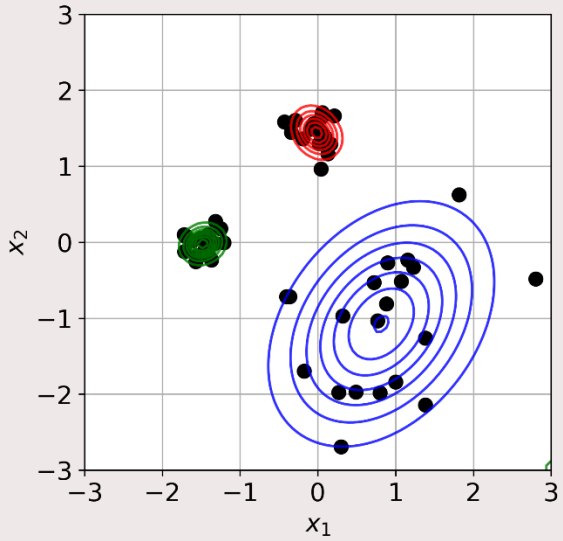
**EM for a GMM may evolve as follows:**




**Initial configuration**




**After 1 iteration**




**At convergence**

# Limitations of Gaussian mixture models

1. **The total optimization procedure is not convex.**
   - Different starting positions may lead to different local optima.
   - Hence, the initial parameters affect the final estimates.
2. **Numerical stability can be a problem.**
   - During optimization, a GMM component might focus on a single data point, which will cause the covariance matrix to shrink to having eigenvalues of 0.
   - This covariance matrix with eigenvalues 0 has to be inverted which will typically result in numerical under- and overflow problems.

## Questions

# Next time: classification and regression