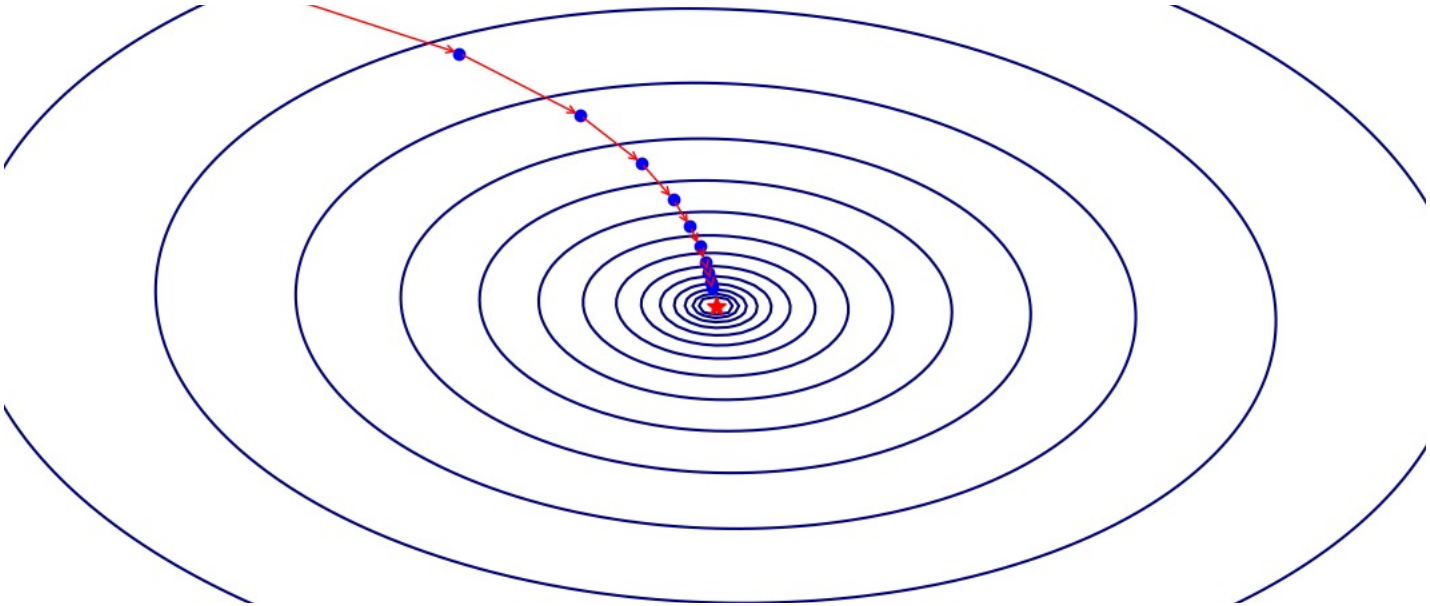


Machine Learning in Signal Processing

Winter Semester 2023/24

10. Clustering

16.01.2023

Prof. Dr. Vasileios Belagiannis

Chair of Multimedia Communications and Signal Processing

Friedrich-Alexander-Universität
Technische Fakultät

Course Topics

1. Introduction.
2. Basics and terminology.
3. Linear regression.
4. Linear classification.
5. Performance evaluation.
6. Neural networks.
7. Deep neural networks.
8. Decision trees.
9. Ensemble Learning.
10. **Clustering.**
11. Dimensionality reduction.
12. Support vector machines.
13. Recap and Q&A.

Acknowledgements

Ideas and inspiration from:

- CSC311 Introduction to Machine Learning, University of Toronto.
- Introduction to Machine Learning: LMU Munich.
- Introduction to Machine Learning, CSAIL, MIT.
- CSE 574 Introduction to Machine Learning, University of Buffalo.
- Special thanks Arij Bouazizi, Julia Hornauer, Julian Wiederer, Adrian Holzbock and Youssef Dawoud for contributing to the lecture preparation.

- The exam will be written.
- Test (ungraded) around the middle of the lectures.
- We will have an exam preparation test before the end of the year.

Last Lecture Recap

- Ensemble learning.
- Bagging.
- Boosting.
- Stacking.

Today's Agenda and Objectives

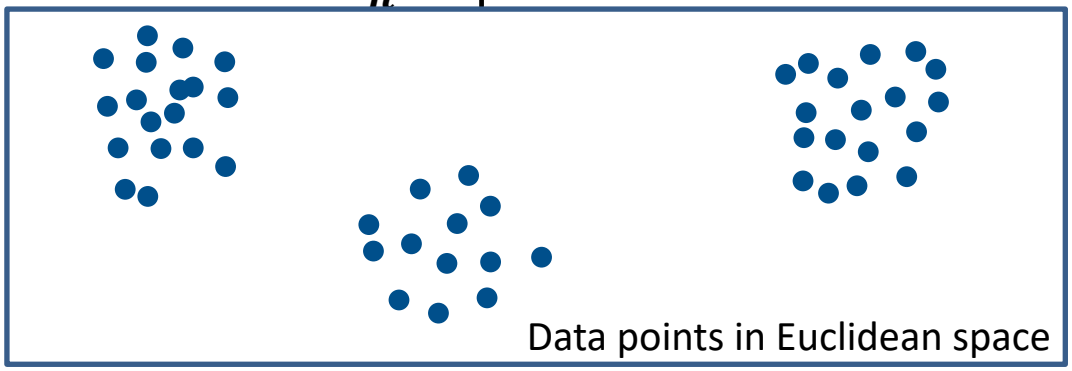
- Unsupervised learning
- Clustering
- Types of clustering
- K-Means algorithm
- Generative models
- Mixture of Gaussians
- Expectation Maximization
- Agglomerative Clustering
- Spectral Clustering

Unsupervised Learning

- Unsupervised learning mainly refers to the lack of labels during the learning (training) phase of our machine learning algorithm.
  - Supervised learning → $D = \{\{\boldsymbol{x_1}, \boldsymbol{y_1}\}, \dots, \{\boldsymbol{x_n}, \boldsymbol{y_N}\}\}$, where we have access to pairs of input features $\boldsymbol{x_n} \in \mathbb{R}$ and labels/targets $\boldsymbol{y_n}$.
  - Unsupervised learning → $D = \{\{\boldsymbol{x_1}\}, \dots, \{\boldsymbol{x_N}\}\}$, where we learn the data probability distribution $p(\boldsymbol{x})$ on implicit or explicit way.
- Clustering is considered is considered as an unsupervised learning family of algorithms.
- *What is data clustering (or cluster analysis)?*

Clustering

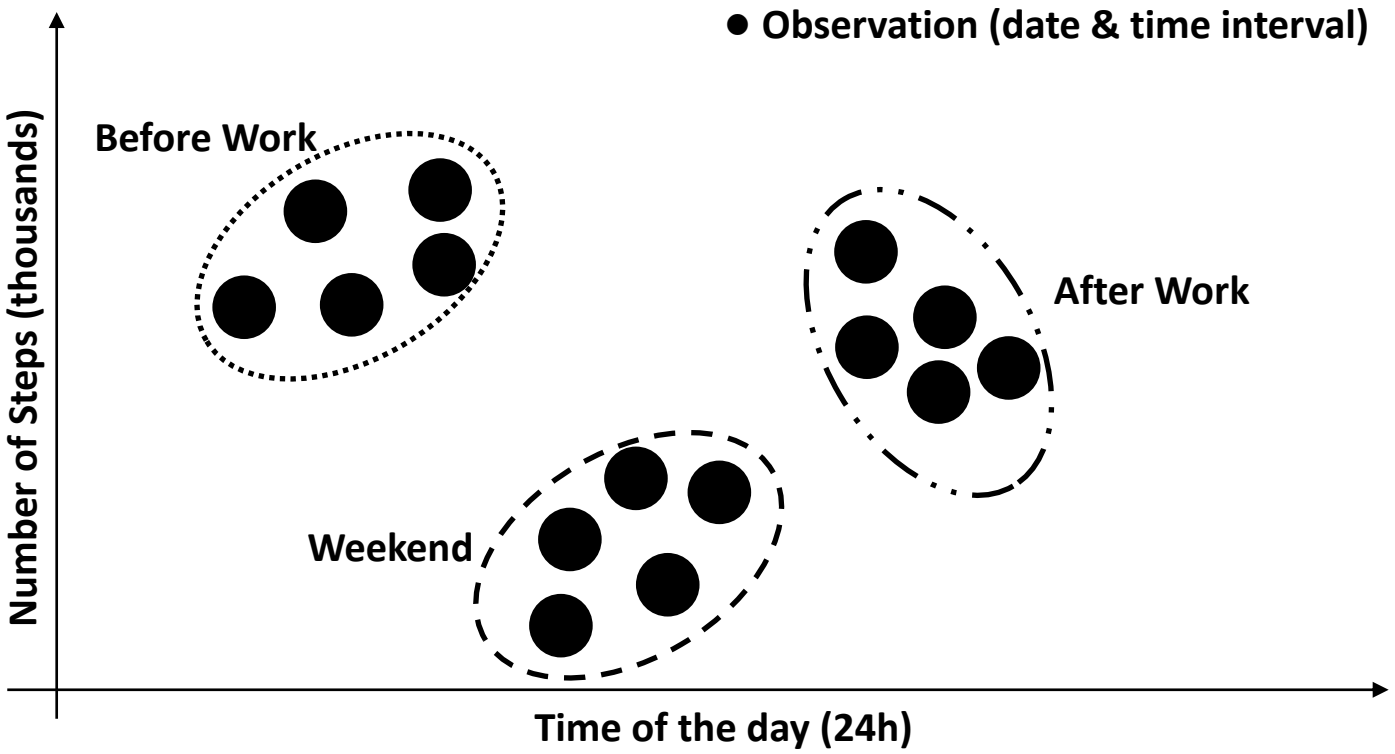
- Consider a set of input features $D = \{\{\boldsymbol{x_1}\}, \ldots, \{\boldsymbol{x_N}\}\}$ with $\boldsymbol{x}_n \in \mathbb{R}^m$ for m-dimensional Euclidean space. For instance, we could consider 2D features for $\boldsymbol{x}_n$ as presented below.



- The data may pose meaningful patterns. Given the input features, groups of similar samples can be formed. We refer to these groups as clusters.
- Clustering (or cluster analysis / data segmentation) is the process of constructing groups of samples. Each groups shares common properties.
- Different clusters / groups are dissimilar to each other.

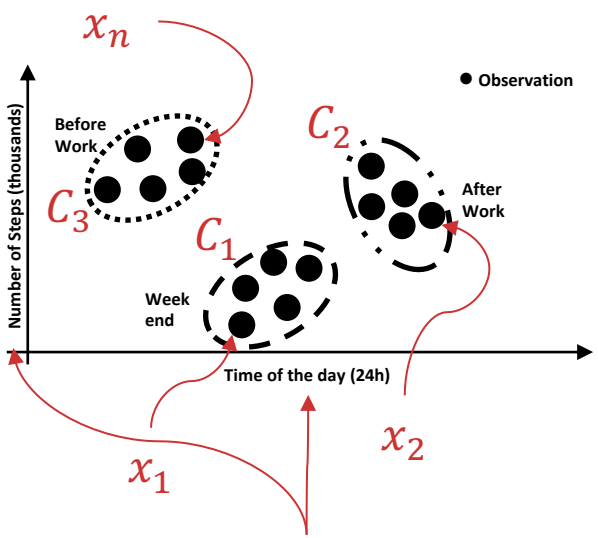
Clustering (Cont.)

- Our goal is to recognise meaningful groups.



Clustering Algorithms

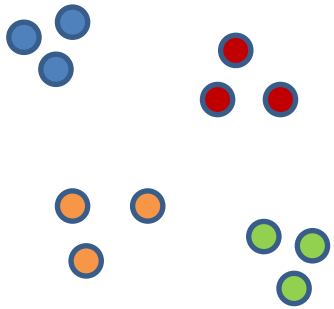
- Designing a clustering approach requires the following:
  - Cluster formation algorithm.
  - Proximity measure to indicate how similar / dissimilar are two samples, e.g. distance function $d: \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}_+$.
  - Criterion to evaluate the clusters.
  - Number of required clusters k (optional).
- Clustering Algorithm:
  - It is going to partition $D = \{\{\boldsymbol{x_1}\}, \ldots, \{\boldsymbol{x_N}\}\}$ into k disjoint subsets.
  - Subset $\rightarrow$ cluster, $C = (C_1, \ldots C_K)$.
  - It is possible also to perform a soft clustering or dendrogram as we will later discuss.



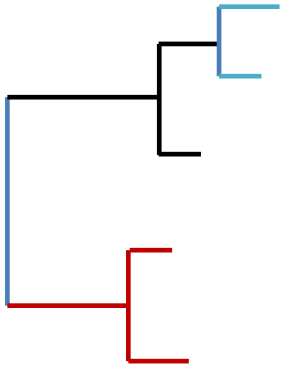
Element $m = 2$.
Clusters $K = 3$.
Euclidean distance d.

Clustering Algorithms (Cont.)

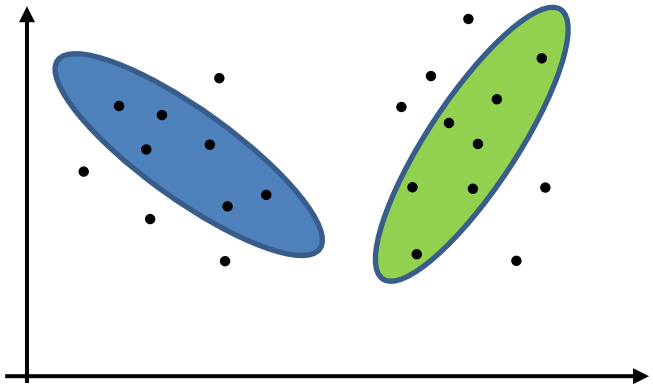
- Common algorithm categories:
  - Centroid-based.
  - Hierarchical.
  - Distribution-based.



K-means.



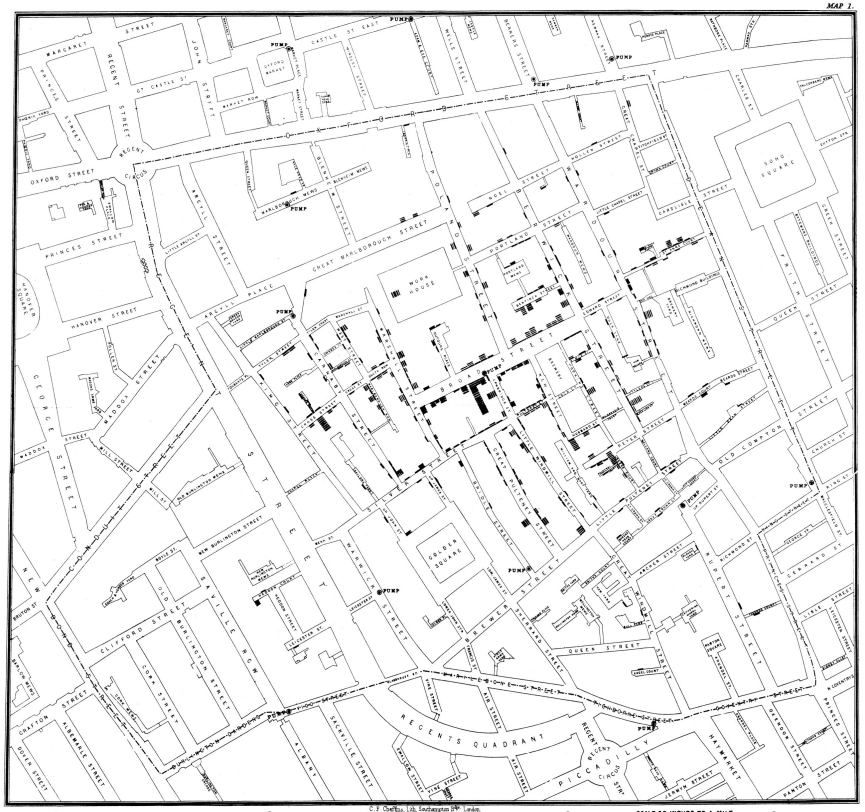
Hierarchical.



Gaussian Mixture Models (GMMs).

History of Cluster Analysis

- Data cluster analysis is older than we might think.
- The work of the English physician John Snow* helped to identify the high mortality rates around the Broad Street (London, 1854) pump.
- To reach this outcome, he clustered all the cholera cases on the map of Soho.
  - Each case is a bar on the map.
  - At specific locations there were (clustered) several cases.
- Based on his cluster analysis, he concluded that the source of transmission was the water.

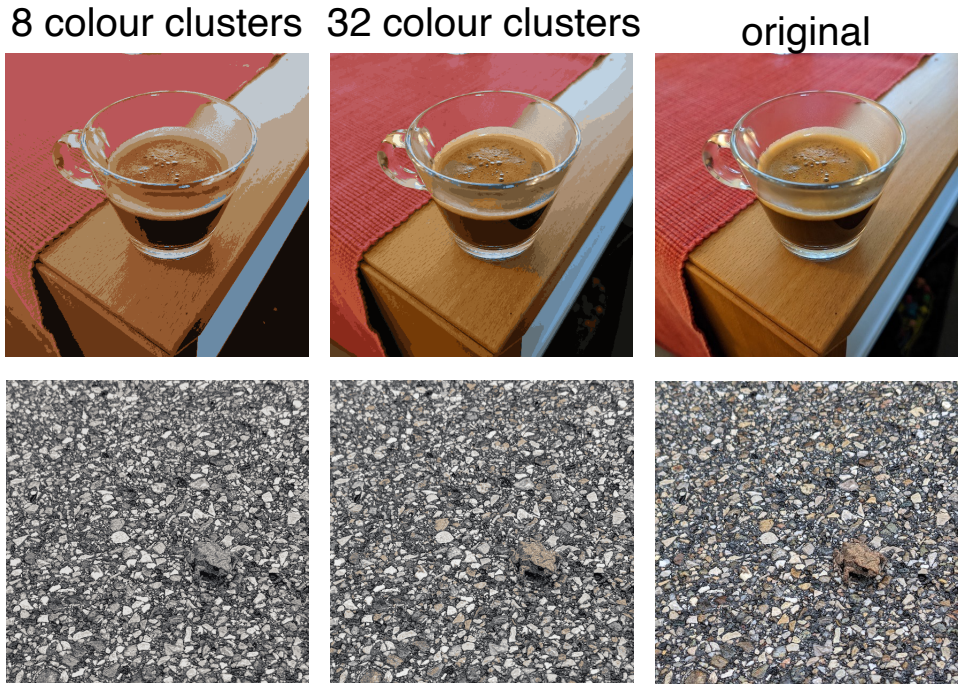


By John Snow - Map of the book; "On the Mode of Communication of Cholera"; by John Snow, originally published in 1854 by C.F. Cheffins, Lith, Southampton Buildings, London, England. The uploaded images is a digitally enhanced version found on the UCLA Department of Epidemiology website., Public Domain, https://commons.wikimedia.org/w/index.php?curid=2278605

Shiode, Narushige, et al. "The mortality rates and the space-time patterns of John Snow's cholera epidemic map." *International journal of health geographics* 14.1 (2015): 1-15.

Few Clustering Applications

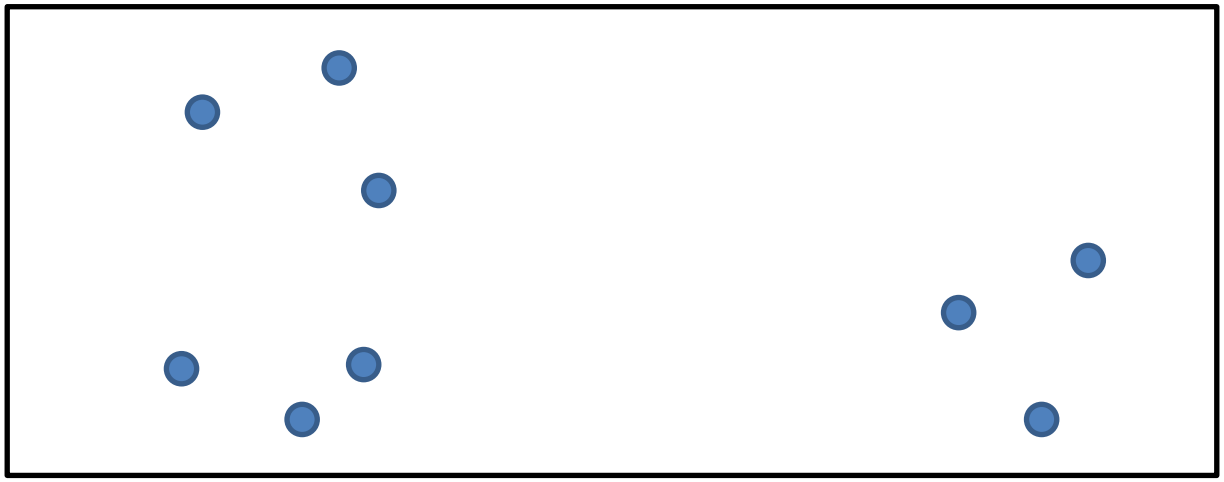
- Image Segmentation & compression.
- Social network analysis.
- Animal Species clustering.
- Search queries grouping, e.g. products, news or articles.



Source Code: https://github.com/preetmishra/image-compression-kmeaxns

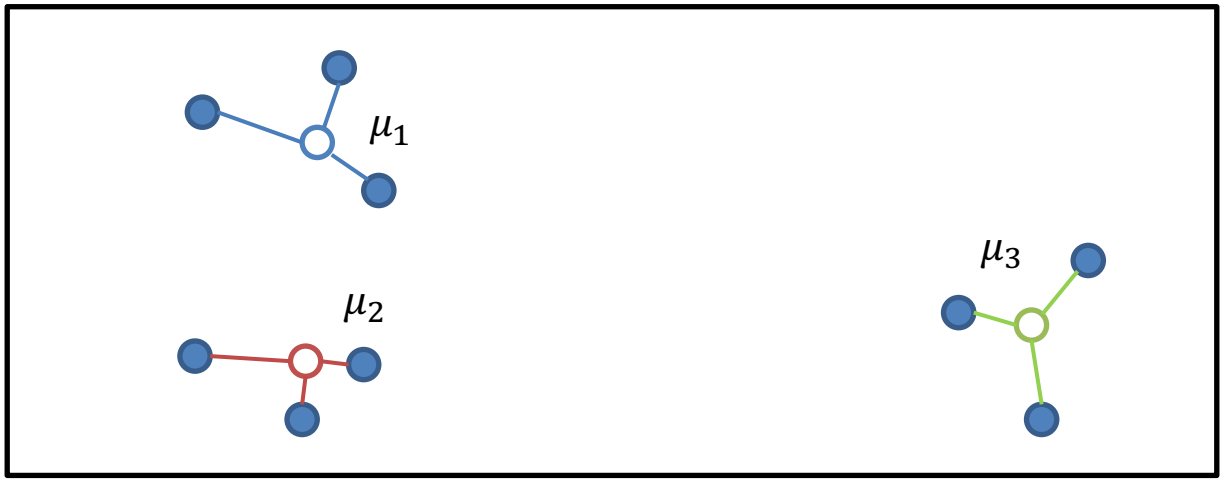
K-means Clustering Problem

- Assume a set of unlabeled data points $D = \{\{\boldsymbol{x_1}\}, \ldots, \{\boldsymbol{x_N}\}\}$ positioned in the Euclidean space, with $\boldsymbol{x}_n \in \mathbb{R}^M$ for M-dimensional Euclidean space and N is the number of observations.
- We also assume that each data point $\boldsymbol{x}_n$ belongs to one of K clusters and that data points of the same cluster are semantically similar, where the Euclidean distance between the points is small.
- Our objective is to find those clusters such that we have a meaningful grouping of data points.



K-means Clustering Objective

- We look for the cluster centres $\boldsymbol{\mu} = \{\boldsymbol{\mu_1}, \ldots, \boldsymbol{\mu_K}\}$ and the cluster assignments $\boldsymbol{l} = \{\boldsymbol{l_1}, \ldots, \boldsymbol{l_N}\}$ for each sample.
- The cluster assignments can be encoded as one-hot vector of K elements. For example, the assignment $l_1 = [1\ 0\ 0]$ means that the sample $\boldsymbol{x}_1$ belongs to the cluster $\boldsymbol{\mu_1}$ and we have in total 3 clusters. The figure below provides this information too.



K-means Clustering Objective

- K-means is an iterative clustering algorithm. We formally define the minimization objective function as follows:
  - $\underset{\boldsymbol{\mu},\boldsymbol{l}}{arg\min} \sum_{n=1}^{N} \sum_{k=1}^{K} l_{n,k} ||\boldsymbol{\mu}_k - \boldsymbol{x}_n||^2$
  - Where $l_{n,k}$ is an element of the one-hot encoding vector $\boldsymbol{l_n}$ that indicates the assignment of $\boldsymbol{x}_n$ to cluster k.
  - For example, if $K = 3$ and $\mathbf{x}_n$ belongs to cluster 1 then $l^{(n)} = [1,0,0]$ and the above equation simplifies to $\sum_{k=1}^{K} l_{n,k} \|\mu_k - \mathbf{x}_n\|^2 = \|\mu_1 - \mathbf{x}_n\|^2$.
- The minimization objective requires joint optimization of $\boldsymbol{\mu}$ and $\boldsymbol{l}$. Thus, finding an optimal solution is an NP-hard problem*.
- To simplify the problem, we consider the following optimization:
  - Minimize $\boldsymbol{l}$ and fix $\boldsymbol{\mu}$ (cluster assignment step).
  - Minimize $\boldsymbol{\mu}$ and fix $\boldsymbol{l}$ (centroid update step).
  - Keep alternating (iterate) between $\boldsymbol{\mu}$ and $\boldsymbol{l}$ until convergence.
- This is a coordinate descent optimization.
- When is the convergence reached for the above problem?

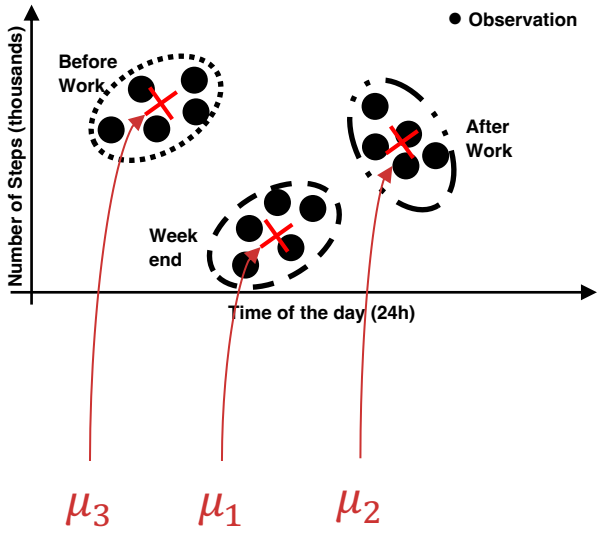
Aloise, Daniel, et al. "NP-hardness of Euclidean sum-of-squares clustering." Machine learning 75.2 (2009): 245-248.

K-means Clustering Alternating Minimisation

- Our objective function is $\underset{\boldsymbol{\mu},\boldsymbol{l}}{arg\min} \sum_{n=1}^{N} \sum_{k=1}^{K} l_{n,k} ||\boldsymbol{\mu}_k - \boldsymbol{x}_n||^2$.
  - Unknowns sets: $\boldsymbol{l}_1, \dots \boldsymbol{l}_N$ and $\boldsymbol{\mu_1}, \dots \boldsymbol{\mu_K}$
- To find our unknown we must:
  - Initialise $\boldsymbol{\mu_1}, \dots \boldsymbol{\mu_K}$. We can randomly pick a point for each cluster.
  - First, optimize for $\boldsymbol{l}_1, \dots \boldsymbol{l}_N$ and fix $\boldsymbol{\mu_1}, \dots \boldsymbol{\mu_K}$. For each data point n, we find the nearest cluster center with:
    - $k^* = \underset{k}{arg\min} \sum_{k=1}^{K} l_{n,k} \|\mu_k - \mathbf{x}_n\|^2$
    - Set $l_{n,k^*} = 1$ and set the rest entries of $\boldsymbol{l}_n$ to zero.
    - No gradient required in this optimization (it's brute force).
  - Second, optimize for $\boldsymbol{\mu_1}, \dots \boldsymbol{\mu_K}$ and fix $\boldsymbol{l}_1, \dots \boldsymbol{l}_N$.
    - We compute gradient of the object function w.r.t $\boldsymbol{\mu_k}$ and then set it to zero. For $j = 1, \dots, K$:
      - $\frac{\partial}{\partial \mu_j} \sum_{n=1}^{N} \sum_{k=1}^{K} l_{n,k} \|\boldsymbol{\mu_k} - \boldsymbol{x}_n\|^2 = 0 \implies 2 \sum_{n=1}^{N} l_{n,j} (\mu_j - \mathbf{x}_n) = 0 \implies$
      - $\mu_j = \frac{\sum_{n=1}^{N} l_{n,j} \mathbf{x}_n}{\sum_{n=1}^{N} l_{n,j}}$.

K-means Clustering Alternating Minimisation (Cont.)

- The two optimisation steps are performed until the cluster assignments are not changing anymore or the cluster centres are stable.
- Centroid update: $\mu_j = \frac{\sum_{n=1}^{N} l_{n,j} \mathbf{x}_n}{\sum_{n=1}^{N} l_{n,j}}$. It is the mean of all x_n that are assigned to the cluster k.
- In summary the K-means algorithm consists of three simple steps:
  - Initialization: randomly initialize cluster centers, e.g., a data point is chosen at random to be the initial cluster center.
  - Iterate over the following two steps until convergence:
    - Cluster assignment.
    - Centroid update.



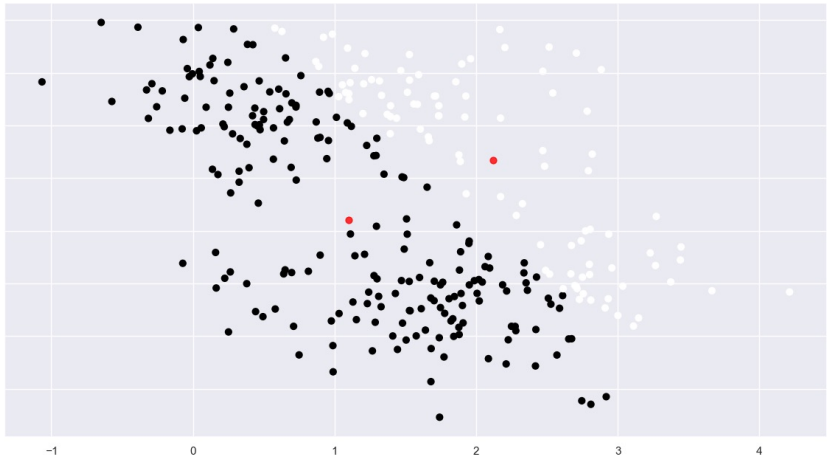
K-means Python Algorithm

- Python Example for 2 clusters

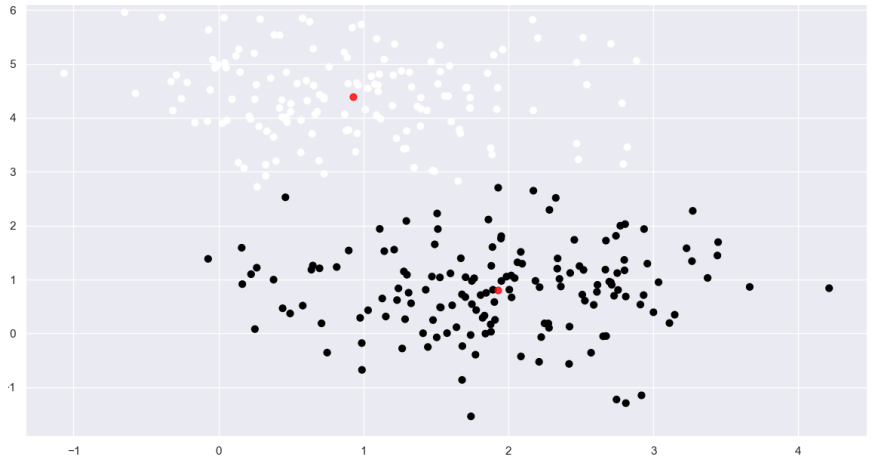
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.metrics import pairwise_distances_argmin
from sklearn.datasets import make_blobs
sns.set()

k_clusters = 2
X, y = make_blobs(n_samples=300, centers=2, cluster_std=0.80, random_state=0)

centroids = np.stack((np.random.uniform(low=np.min(X,0)[0],
high=np.max(X,0)[0],size=2), np.random.uniform(low=np.min(X,1)[0], high=np.max(X,1)[0],size=2)), axis=0)

cnt=0
while True:
      cnt+=1

      clusters = pairwise_distances_argmin(X, centroids)
      centroids_update = np.array([X[clusters == i].mean(0) for i in range(k_clusters)])

      if cnt % 2:
      plt.scatter(X[:, 0], X[:, 1], c=clusters, s=40, cmap='gray');
      plt.scatter(centroids_update[:, 0], centroids_update[:, 1], c='red', s=40, alpha=0.8);
      plt.show()

      if np.all(centroids == centroids_update):
            break

      centroids = centroids_update
```

K-means Python Algorithm (Cont.)

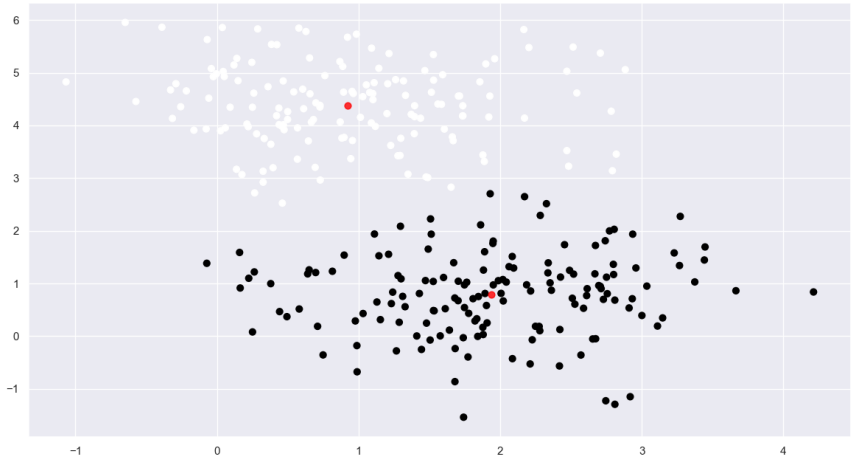
Iteration 2



Iteration 4

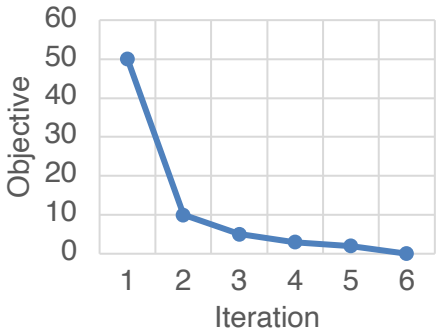


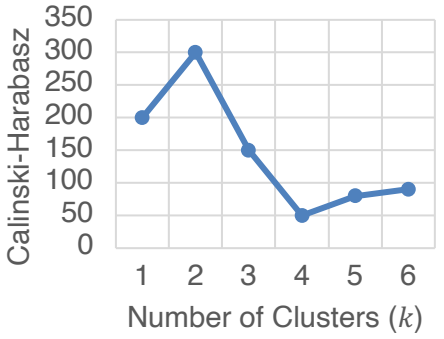
Iteration 6



K-means Clustering Observations

- Convergence: We have a monotonically decreasing objective function*. We always have a local minima solution.
- There is no guarantee on the number of required iterations.
- We have to run the algorithm multiple times with different initialisations and keep the run with the lowest objective**.
- Selecting the number of clusters K: We can rely on the elbow method (Calinski-Harabasz index, F-Ratio, Bayesian information criterion).
- There are also the following algorithm variants:
  - k-medoids objective function.
  - k-median objective function.





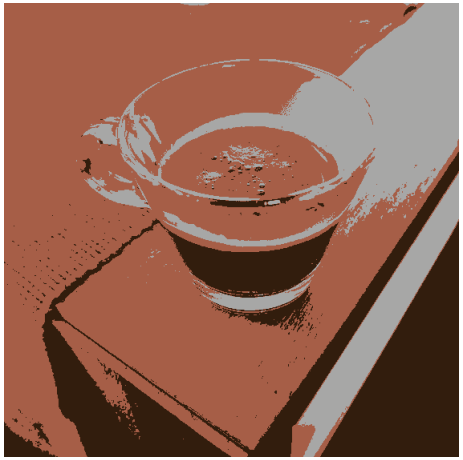
*Bottou, Leon, and Yoshua Bengio. "Convergence properties of the k-means algorithms." Advances in neural information processing systems. 1995.
**Likas, Aristidis, Nikos Vlassis, and Jakob J. Verbeek. "The global k-means clustering algorithm." Pattern recognition 36.2 (2003): 451-461.

Soft K-means Algorithm

- We have discussed until now the standard version of K-means algorithm, that is hard K-means algorithm.
  - Each sample belongs to one cluster and does not belong to the rest ones.
- K-means can work also with soft assignments. For instance, for K=2 a data point may belong to cluster 1 with probability 0.7 and cluster 2 with probability 0.3.
  - Recall the multiclass classification where we had the SoftMax activation function to show the probability of a data sample belonging to a certain class.
- In soft K-means algorithm we have the cluster assignment step defined as:
  - $l_{n,k} = \frac{\exp[-\beta\|\boldsymbol{\mu}_k - \boldsymbol{x}_n\|^2]}{\sum_j \exp[-\beta\|\boldsymbol{\mu}_j - \boldsymbol{x}_n\|^2]}$
  - where β is a weighting hyperparameter.
  - Note that as $\beta \to \infty$, soft K-means is reduced to standard K-means.

K-Means Applications: Image Segmentation

3 clusters



original

Source Code: https://github.com/x4nth055/pythoncode-tutorials/tree/master/machine-learning/kmeans-image-segmentation

Clustering with a Generative Model

- Clustering can be formulated as generative modeling. We can treat the observed samples as being generated from some source with an underlying probability distribution. In turn, our objective would be to model this distribution such that it maximizes the probability of generating the observed samples.
- We consider the mixture of Gaussian distributions as our generative model.
  - It is a model that is comprised of K different Gaussian distributions.
  - For each Gaussian, we have mean μ_k, covariance Σ_k and the mixing weight π_k with $k = 1, \dots, K$.
  - For the clustering problem, each Gaussian describes a different cluster.
  - Also, the mixing weights sum up to one: $\sum_{k=1}^{K} \pi_k = 1$.
  - Our algorithms should function as follows: For a given sample $\boldsymbol{x}$ we look for the probability that it came from the Gaussian k. We set this probability equal to the mixing weight, given as: $p(= k) = \pi_k$ where c is the cluster of $\boldsymbol{x}$. Also, we have $p(\mathbf{x}|c = k) = \mathcal{N}\left(\mathrm{x}\middle|\mu_c, \Sigma_c\right)$ the probability density function.
- Our goal is to estimate the parameters of the Gaussians μ_k, Σ_k and π_k that best explain our training set $D = \left\{\{\boldsymbol{x_1}\}, \dots, \{\boldsymbol{x_N}\}\right\}$.

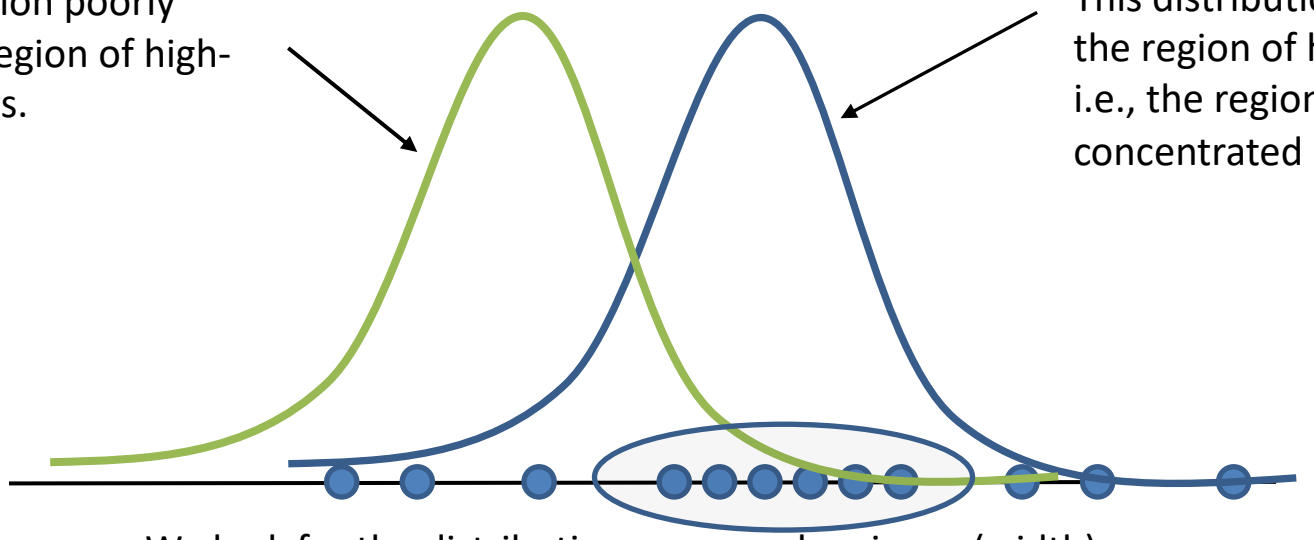
Clustering with a Generative Model (Cont.)

- Based on the assumption $p(\boldsymbol{x}|c=k)=\mathcal{N}\left(\boldsymbol{x}\middle|\mu_c,\Sigma_c\right)$, we would like now to define the joint probability $p(c,\boldsymbol{x})=p(c)p(\boldsymbol{x}|c)$ with parameters $\{\mu_k,\pi_k\}_{k=1}^{K}$.
  - We assume $\Sigma_k=I$ for simplicity.
- Then we can marginalize over c to obtain:
  - $p(\boldsymbol{x})=\sum_c p(c,\boldsymbol{x})$.
- To determine the probability of $\mathbf{x}$ coming from cluster c we use the Bayes' rule:
  - $p(c=k|\boldsymbol{x})=\frac{p(c=k)p(\boldsymbol{x}|c=k)}{p(\boldsymbol{x})}$.

Clustering with a Generative Model (Cont.)

- Our objective is to maximize the likelihood of the observed data $D = \{\{\boldsymbol{x_1}\}, \dots, \{\boldsymbol{x_N}\}\}$.
  - Maximizing the likelihood is equivalent to finding the parameters of the data distribution. In our case the parameters are $\{\mu_k, \pi_k\}_{k=1}^{K}$.
  - Also consider that we do not observe the cluster assignment c. This is a latent variable.

This distribution poorly models the region of high-density points.



This distribution fits well to the region of high-density i.e., the region with the most concentrated data points.

We look for the distribution mean and variance (width).

Clustering with a Generative Model (Cont.)

- We define our objective function as:
  - $\max_D \log p(D) = \sum_{n=1}^{N} \log p(\boldsymbol{x_n})$.
  - We use the log-scale for numerical stability.
- Then, we obtain $p(\boldsymbol{x})$ by marginalizing over c:
  - $p(\boldsymbol{x}) = \sum_{k=1}^{K} p(c = k, \boldsymbol{x}) = \sum_{k=1}^{K} p(c = k) p(\boldsymbol{x} | c = k)$.
- By assuming $p(\boldsymbol{x}|c = k) = \mathcal{N}\left(\boldsymbol{x}\middle|\mu_c, \Sigma_c\right)$ and $p(c = k) = \pi_k$, we have:
  - $p(\boldsymbol{x}) = \sum_{k=1}^{K} p(c = k) p(\mathbf{x}|c = k) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathrm{x}|\mu_k, \mathbf{I})$.
  - This is the Gaussian mixture model with the mixing weight (called also coefficient) π_k.
  - For a non-identity covariance, we would have: $p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\mathrm{x}\middle|\mu_k, \Sigma_k\right)$.

Clustering with Gaussian Mixture Models

- To reach our objective we need to fit a mixture of Gaussian distributions with parameters $\{\mu_k, \pi_k\}_{k=1}^{K}$ to the observed data $D = \{\{\boldsymbol{x_1}\}, \dots, \{\boldsymbol{x_N}\}\}$.
- We further expand the maximum likelihood objective to:
  - $\max_{D} \log p(D) = \sum_{n=1}^{N} \log p(\mathbf{x}_n) = \sum_{n=1}^{N} \log(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{I}))$
  - Because of the logarithm, taking the gradient of this equation is not easy.
- To find a solution, let us first assume knowledge of the cluster assignments for every $\mathbf{x}_n$. We now have $\widehat{D} = \{c_n, \mathbf{x}_n\}_{n=1}^{N}$. The maximum likelihood objective becomes:
  - $\max_{D} \log p(\widehat{D}) = \sum_{n=1}^{N} \log p(c_n, \mathbf{x}_n) = \sum_{n=1}^{N} \log p(\mathbf{x}_n|c_n) + \log p(c_n) = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[c_n = k] (\log \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{I}) + \log \pi_k.$
  - where $\mathbb{I}[c_n = k]$ is one-hot encoding vector with 1 at k-th entry if $\mathbf{x}_n$ belongs to cluster k.

Clustering with Gaussian Mixture Models (Cont.)

- The maximum likelihood objective now is:
  - $\max_D \log p(\widehat{D}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[c_n = k] \left(\log \mathcal{N}(\mathbf{x}_n | \mu_k, \mathbf{I}) + \log \pi_k\right)$
  - However, we do not know the cluster assignment that is need at $\mathbb{I}[c_n = k]$.
- Even without knowledge about the cluster assignment, we can still compute $p(c = k|\mathrm{x})$ using the Bayes' rule:
  - $p(c = k|\mathrm{x}) = \frac{p(c=k)p(\mathbf{x}|c=k)}{p(\mathbf{x})} = \frac{p(c=k)p(\mathbf{x}|c=k)}{\sum_{j=1}^{K} p(c=j)p(\mathbf{x}|c=j)} = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k,\mathbf{I})}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j,\mathbf{I})}$
- Also we can replace $\mathbb{I}[c_n = k]$ with its expectation:
  - $\mathbb{E}[\mathbb{I}[c_n = k]|\mathbf{x}_n] = p(c_n = k|\mathbf{x}_n)$.
- Our maximum likelihood objective now becomes:
  - $\sum_{n=1}^{N} \sum_{k=1}^{K} l_{n,k} (\log \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{I}) + \log \pi_k$
  - Where $l_{n,k} = p(c_n = k|\mathbf{x}_n)$.
  - The solution to this objective is given by:
    - $\mu_k = \frac{\sum_{n=1}^{N} l_{n,k}\mathbf{x}_n}{\sum_{n=1}^{N} l_{n,k}}$.
    - $\pi_k = \frac{1}{N}\sum_{n=1}^{N} l_{n,k}$.
  - If we fix $l_{n,k}$ then we can computer the parameters μ_k and π_k.
  - This motivates us for the Expectation-Maximization algorithm.

Expectation Maximization Algorithm

- The idea of the algorithm is to iterate between two steps (like what we learned with K-means).
- First, we need to randomly initialise μ_k and π_k.
- Then, we iterate over the following two steps until convergence:
  - E-step: We compute the posterior probabilities given the current Gaussian parameters. For each sample $\boldsymbol{x}_n$, we compute the assignment:
    - $l_{n,k} = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k,\mathbf{I})}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j,\mathbf{I})} = \frac{\pi_k \exp\{-0.5 \|x_n-\mu_k\|^2\}}{\sum_{j=1}^{K} \pi_j \exp\{-0.5 \|x_n-\mu_j\|^2\}}.$
    - This steps tell us how likely is each sample $\boldsymbol{x}_n$ to stem out from the the cluster k.
  - M-step: We update μ_k and π_k of each Gaussian component using the calculated (and fixed) $l_k^{(n)}$. We use the equations which we already seen in the previous slide.
    - $\mu_k = \frac{\sum_{n=1}^{N} l_{n,k}\mathbf{x}_n}{\sum_{n=1}^{N} l_{n,k}}$
    - $\pi_k = \frac{1}{N}\sum_{n=1}^{N} l_{n,k}$
    - This step is responsible for adapting the parameters to maximize the probability that each Gaussian generated the samples which lie within it.

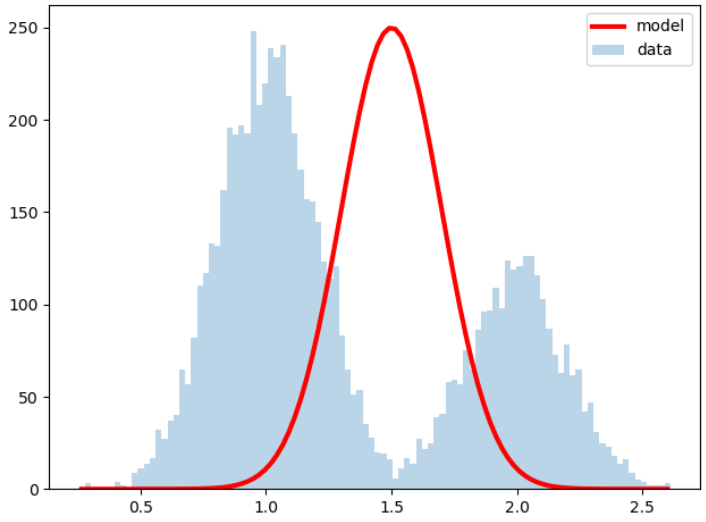
Expectation Maximization Algorithm (Cont.)

- Each cluster is treated as a different Gaussian distribution.
- In general, we can replace a Gaussian distribution with any other distribution.
  - However, a Gaussian distribution models many real life statistics.
- We assumed the covariance term is an identity matrix, nevertheless, we can ignore this assumption to model clusters with different spatial extents.
  - The algorithm remains the same, but we would have to optimize for Σ_k.
- The algorithm is related to K-means clustering. Instead of the cluster assignment and centroid update steps, we now have the expectation and the maximization steps.
  - Both algorithms are based on alternating optimization.
  - Both algorithms reach local minima.

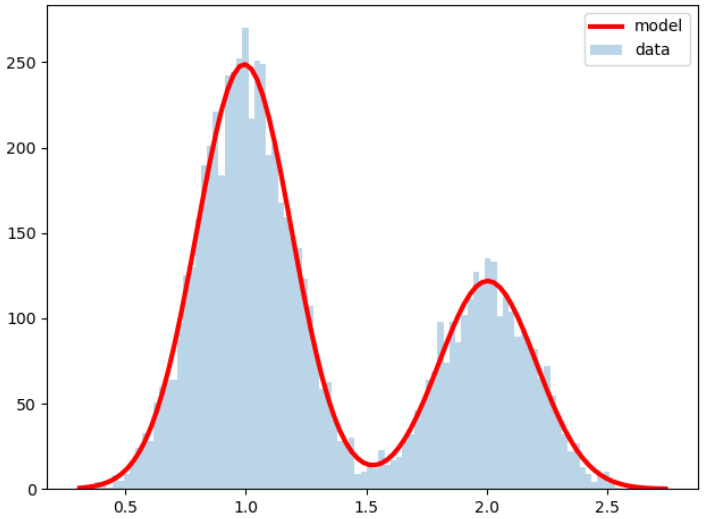
GMM 1D

- Fitting Gaussian distribution to a 1-dimensional data would look as follows.

K=1



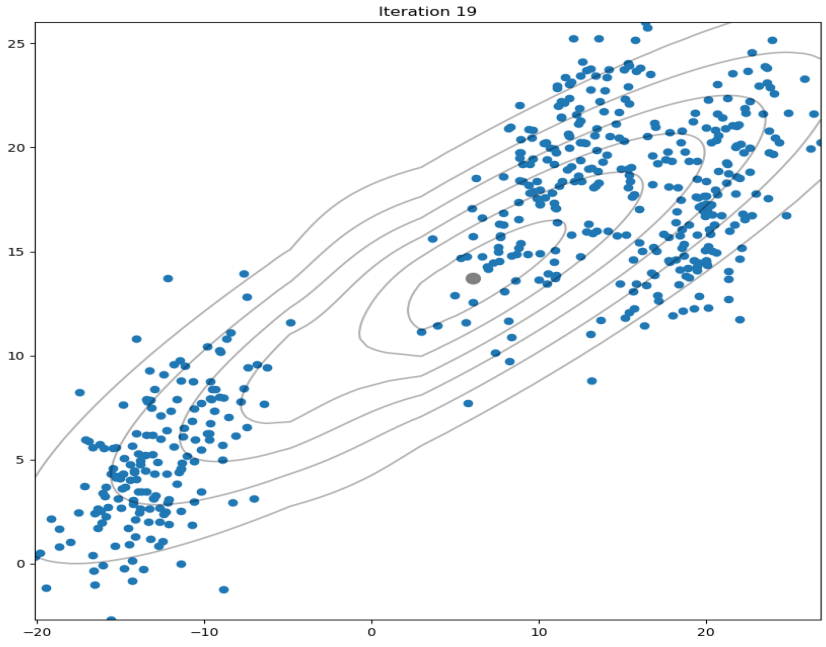
K=2, fits better



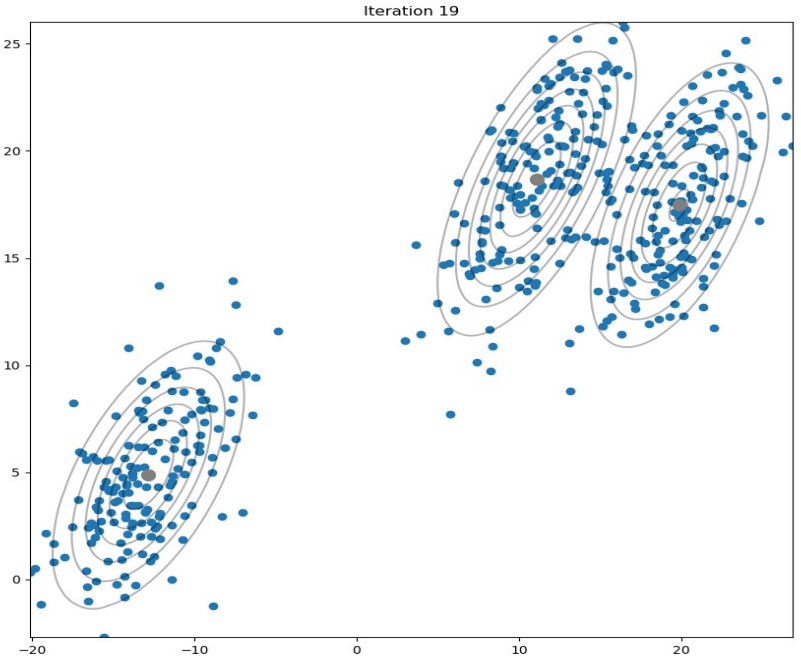
GMM 2D

- Fitting Gaussian distribution to a 2-dimensional data would look as follows.
- How do we choose the number of clusters / mixtures?

K=1

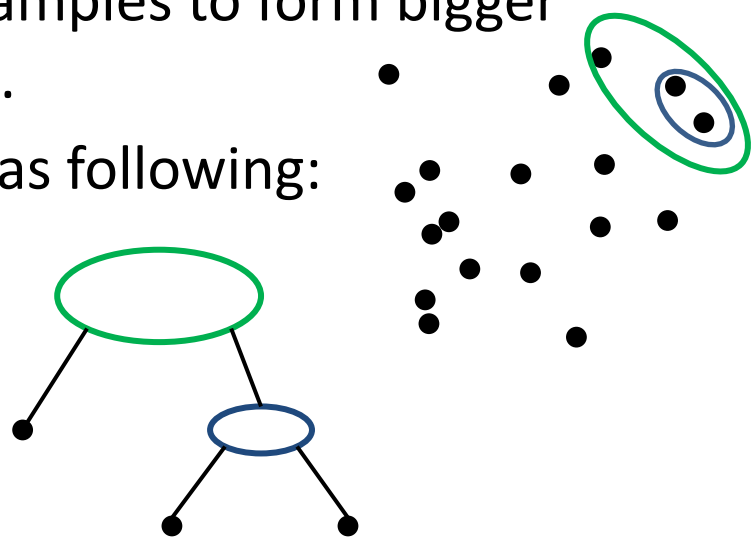


K=3, fits better



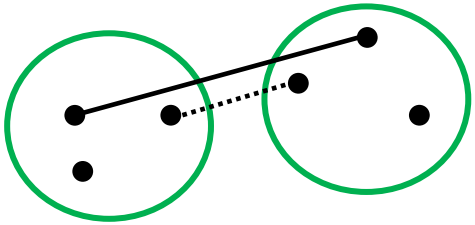
Hierarchical Clustering

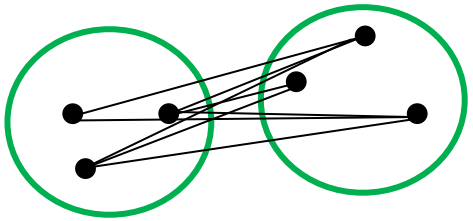
- In hierarchical clustering we build different levels of clusters based on a hierarchy.
- There two categories of hierarchical clustering: agglomerative and spectral.
- Agglomerative: We start with each sample being its own cluster. Then incrementally merge samples to form bigger clusters. This a bottom-up approach.
- We can think the general algorithm as following:
  - Pick the two closest clusters.
  - Merge them into a new cluster.
  - Iterate until a single cluster is left.
- It is is produce a dendrogram.



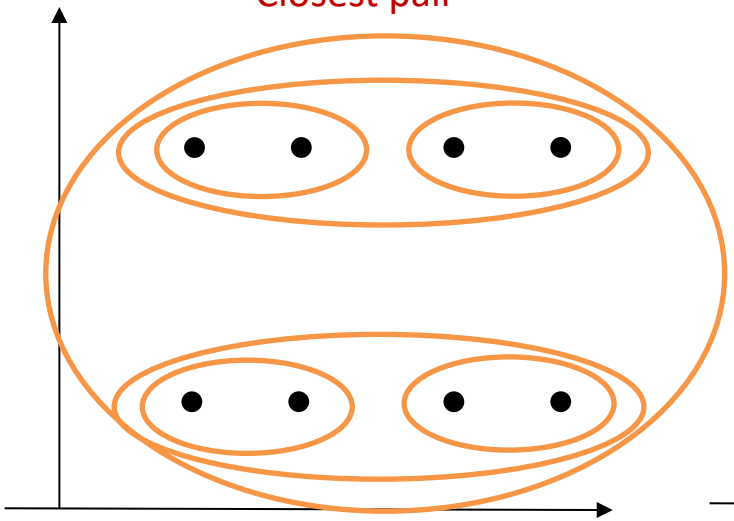
Agglomerative Clustering

- *How do we define the closest cluster?*
- To define the closest clusters, we have the following the options:
  1. Measure the distance between closest pair.
  2. Measure the distance between farthest pair.
  3. Average distance of all pairs.
- Based on the chosen option we obtain different clustering output.
- The distance metric could be the squared Euclidean distance for all cases.

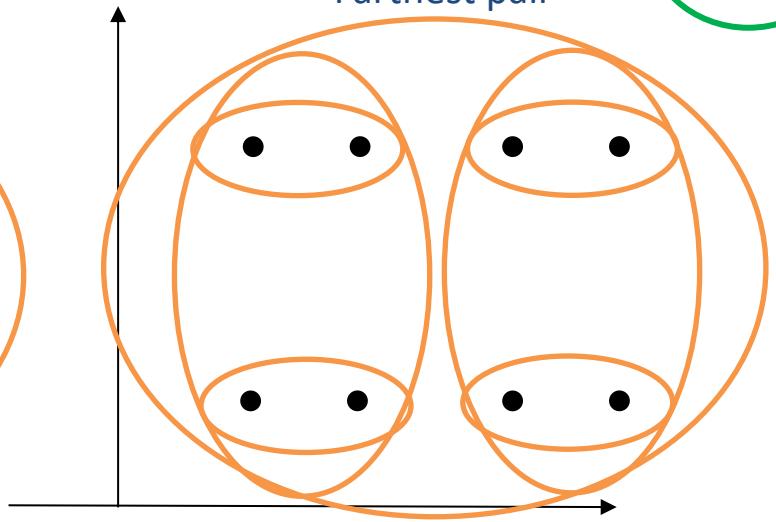




Closest pair



Farthest pair



Spectral Clustering

- All samples belong to a single cluster at the beginning of spectral clustering. Iteratively we create additional clusters. This a top-down approach.
- We rely on a graph to represent our samples here. Also an adjacency matrix A represents the similarity between the samples.
  - The objective is find to build meaning clusters based on the sample similarity / difference.
  - We can run K-means on the eigenvectors of the Laplacian matrix of A.
  - We do not rely on all eigenvectors but only these ones with small eigenvalues.

Study Material

Not for sharing (LMS, Friedrich-Alexander-Universität Erlangen-Nürnberg)

- *The Elements of Statistical Learning, Trevor Hastie et. al, Chapter 9.*
- *Pattern Recognition and Machine Learning, Christopher Bishop, Chapter 10.*

Next Lecture

Dimensionality Reduction