CS-EJ3211 Machine Learning with Python

Session 5 - Clustering

Alexander Pavlyuk

Aalto University
FITech

28.06.23

Supervised vs. Unsupervised learning

Supervised learning (week 2-4):

- **Labeled** training set
- Supervised methods rely on the true labels

Unsupervised learning (week 5-6):

- **Unlabeled** training set
- Unsupervised methods rely on the intrinsic structure of data points

Unsupervised learning

Application examples:

- News sections
- Computer vision (object recognition)
- Anomaly detection
- Recommendation engines
- Customer personas

Unsupervised learning

Pros:

- Identifies patterns in large volumes of data more quickly
- Does not require human intervention to label the data

Cons:

- Risks of inaccurate results
- Computational complexity due to a high volume of training data
- Lack of transparency into the basis on which data was clustered

Unsupervised learning

Methods:

- Clustering (K-means, Gaussian Mixture Models (GMM), DBSCAN)
- Feature learning (PCA, MDS, Isomap, etc.)

Clustering

Definition: decomposing a set of data points into few subsets (clusters) that consist of similar data points is called clustering.

Clustering methods are roughly divided into two groups:

- **Hard clustering** methods - assign each data point to exactly one cluster
- **Soft clustering** methods - assign each data point to several different clusters with varying degrees of belonging

Hard Clustering: K-means

- Given: number of clusters k (hyperparameter)
- Similarity measure: Euclidean norm (distance)
- Idea: iteratively assign each data point to one of the k clusters by minimizing the sum of squared distances of the data points and their respective cluster means until the stopping criterion is met
- Stopping criterion examples:
  - Cluster means change less than defied tolerance (algorithm converged)
  - Maximum number of iterations is achieved
- Scikit-learn implementation

Hard Clustering: K-means

Algorithm:

1. select k samples as initial centroids (cluster means)
2. create k clusters by assigning each sample to the closest centroid

$$\hat{y}^{(i)} = \underset{c \in \{1,\ldots,k\}}{\operatorname{argmin}} \|\mathbf{x}^{(i)} - \mu^{(c)}\|^2$$

3. create k new centroids by averaging samples in each cluster
4. if centroids do not change (algorithm converged) or other stopping criterion is met - break, else - go to the step 2

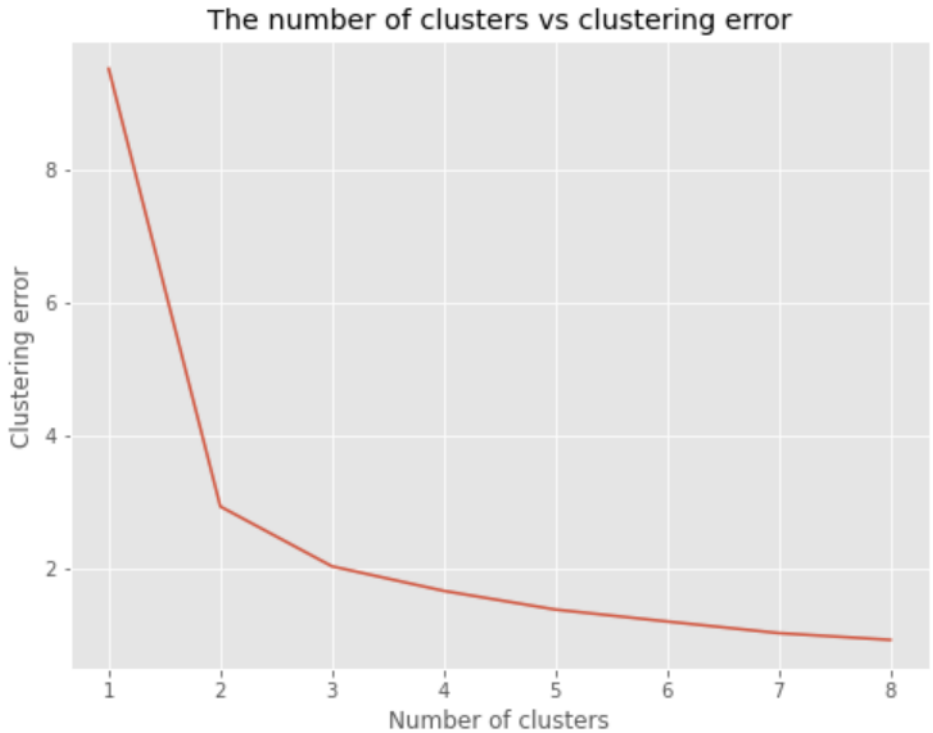
Click for animation

K-means: Handling local minima

- Given enough time, K-means will always converge. However, this may be to a local minimum (dependent on the initialization of the centroids)
- $\rightarrow$ Do computation several times, with different initializations of the centroids
- sklearn.cluster.KMeans has default param `init='k-means++'`. This initializes the centroids to be (generally) distant from each other. It also makes several trials at each sampling step and chooses the best centroid among them

K-means: How many clusters?

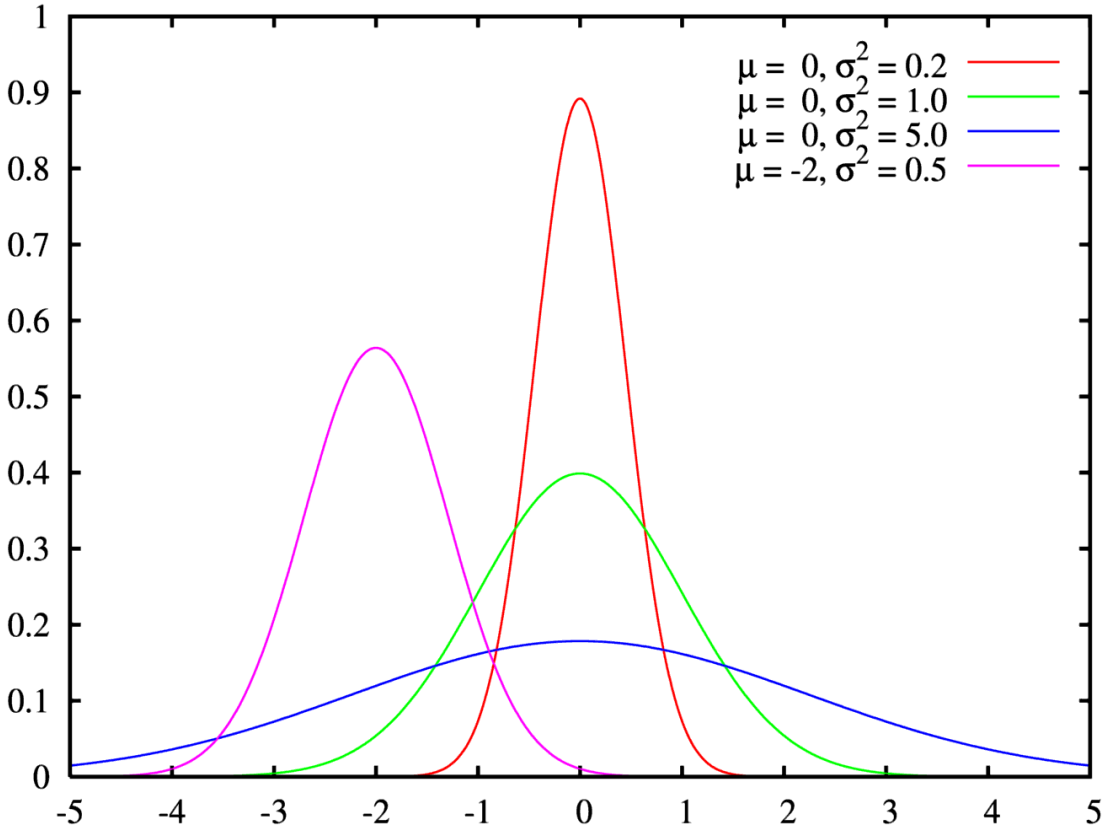
- Visualization - few clusters
- Pre-processing before supervised methods - use validation set to choose parameter k
- "Elbow" method - try different k values



Soft clustering: Gaussian Mixture Models

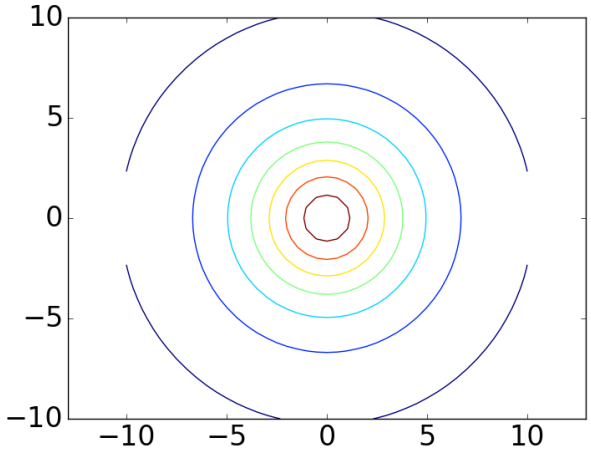
Gaussian probability distribution (1D):

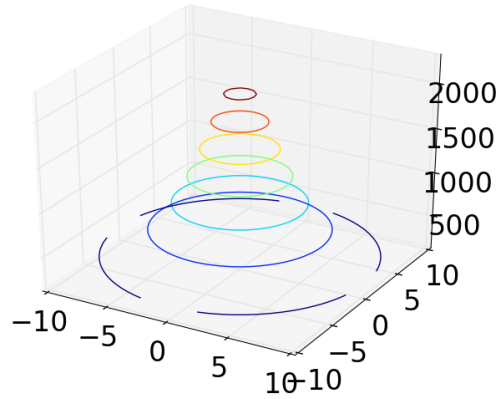
$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

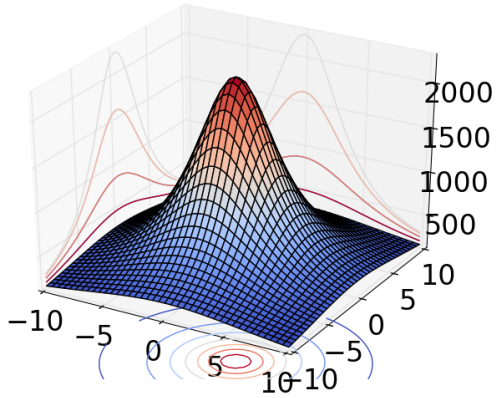


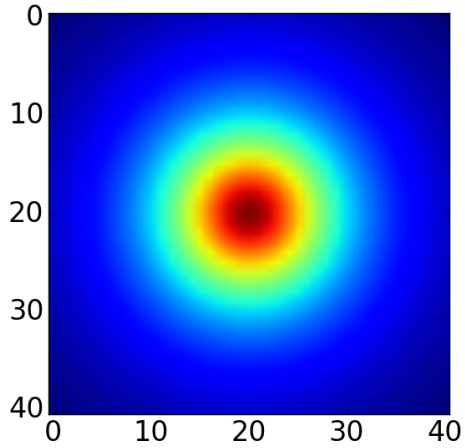
Soft clustering - Gaussian Mixture Models

Gaussian probability distribution (2D, 3D):









Soft clustering - Gaussian Mixture Models

- Data is assumed to be drawn from k different multivariate Gaussian distributions
- Each Gaussian distributions is parametrized by a mean vector $\mu^{(c)}$ and a covariance matrix $\mathbf{C}^{(c)}$
- The model has the parameters p_c representing the probability of drawing a data point from the distribution c
- The model is fitted by finding the parameters $\mu_c, \mathbf{C}_c, p_c$, for each $c = 1, \ldots, k$ (where k is the number of clusters), that maximize the likelihood of the observed data

Soft clustering Gaussian Mixture Models

Algorithm:

1. randomly select Gaussian parameters $\mu^{(c)}$, $\mathbf{C}^{(c)}$
2. compute probabilities of a datapoint coming from each Gaussian

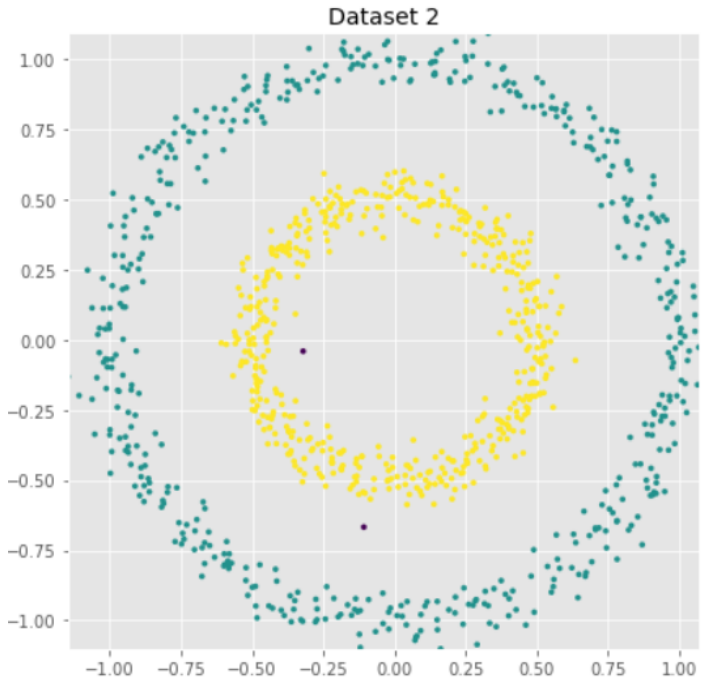
$$\mathbf{y}_c^{(i)} = \frac{p_c \mathcal{N}(\mathbf{x}^{(i)}; \mu^{(c)}, \mathbf{C}^{(c)})}{\sum_{c'=1}^{k} p_{c'} \mathcal{N}(\mathbf{x}^{(i)}; \mu^{(c')}, \mathbf{C}^{(c')})}$$

3. update parameters $\mu^{(c)}$, $\mathbf{C}^{(c)}$ to maximize likelihood
4. if log-likelihood do not change significantly (algorithm converged) - break, else - go to step 2

Click for animation

DBSCAN algorithm

- DBSCAN stands for density-based spatial clustering of applications with noise
- Connectivity-based similarity measure



DBSCAN algorithm

Pros:

- No need no specify the number of clusters k
- DBSCAN can find arbitrarily-shaped clusters
- DBSCAN algorithm is robust to outliers

Cons:

- DBSCAN is not deterministic (use DBSCAN* instead)
- DBSCAN cannot cluster data sets with large differences in densities
- Choosing hyperparameters is a difficult task for data that is not well understood

Additional material

- Clustering with sklearn - Scikit-learn docs
- Determining the number of clusters in a data set - Wikipedia
- EM, GMM lecture - Youtube
- DBSCAN - Wikipedia