Non-parametric Density Estimation and the Expectation-Maximization (EM) Algorithm

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

Summer term 2025

Find all slides on GitHub (DaWe1992/Applied_ML_Fundamentals)

Lecture Overview

Agenda for this Unit

1. Non-parametric Density Estimation
2. *k*-nearest Neighbors for Classification
3. EM Algorithm for GAUSSian Mixture Models
4. Wrap-Up

Section:

Non-parametric Density Estimation

Disadvantages of parametric Density Estimation

- So far we have focused on the use of probability distributions having **specific functional forms**
- Those were **governed by a small number of parameters** whose values we estimated on the dataset
- This is called the **parametric approach**

Disadvantage: An important limitation of this approach is that the chosen density might be a **poor model of the distribution** that generates the data, which can result in **poor predictive performance**. For instance, if the process that generates the data is multimodal, then this aspect of the distribution can never be captured by a GAUSSian, which is necessarily unimodal.

Histograms

- Let us start our discussion of **non-parametric methods** with **histograms**
- Consider a dataset of N observations of a single continuous variable $x \in \mathbb{R}$
- Histograms partition x into distinct bins of width Δ_i and then count the number N_i of observations of x falling into bin i
- In order to turn this into a normalized probability, we compute

$$p_i = \frac{N_i}{N\Delta_i} \tag{1}$$

- Often we set $\Delta_i := \Delta$ (all bins have equal width)

Advantages and Disadvantages of Histograms

Advantage: The **training data can be discarded** once we have the histograms

Disadvantages:

1. **Curse of dimensionality:** In high-dimensional space we need an **exponential amount of data** to obtain meaningful results
2. **Artificial discontinuities:** The probability density function suffers from artificial discontinuities due to the bin edges – we prefer a smooth function

Non-parametric Density Estimation

- Suppose we have a training set comprising N examples

$$\boldsymbol{X} := \left\{ \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^N \right\}$$

  and we wish to estimate the value $p(\boldsymbol{x}')$ for an unknown data point $\boldsymbol{x}'$
- Consider a small region $\mathcal{R}$ containing $\boldsymbol{x}'$
- The **probability mass** P associated with region $\mathcal{R}$ is given by:

$$P = \int_{\mathcal{R}} p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \tag{2}$$

Non-parametric Density Estimation (Ctd.)

- Since each data point $\boldsymbol{x}^n$ falls into $\mathcal{R}$ with probability P, the total number of points k in $\mathcal{R}$ will be distributed according to the **binomial distribution**

$$b(k; N, P) = \frac{N!}{k!(N-k)!} P^k (1-P)^{N-k} \tag{3}$$

- The mean fraction of points falling inside region $\mathcal{R}$ and the respective variance are:

$$\mathbb{E}\{k/N\} = \frac{1}{N}\mathbb{E}\{k\} = \frac{1}{N}NP = P \tag{4}$$

$$\mathbb{V}\{k/N\} = \frac{1}{N^2}\mathbb{V}\{k\} = \frac{1}{N^2}NP(1-P) = \frac{P(1-P)}{N} \tag{5}$$

Non-parametric Density Estimation (Ctd.)

- For large N the distribution will be **sharply peaked** around the mean, i. e. $k \approx NP$
- If $\mathcal{R}$ is sufficiently small, then the probability density $p(\boldsymbol{x})$ is roughly constant over the region, i. e. $P \approx p(\boldsymbol{x'})V$ (where V is the volume of $\mathcal{R}$)
- Combining both equations, we obtain

$$p(\boldsymbol{x'}) = \frac{k}{NV} \tag{6}$$

Note: Please note that the validity of equation (6) depends on two **contradictory assumptions**: $\mathcal{R}$ must be sufficiently small that the probability is roughly constant and yet sufficiently large so that the distribution is sharply peaked.

Kernel Density Estimation vs. k-nearest Neighbors

Equation (6) gives rise to two different approaches to non-parametric density estimation:

Kernel Density Estimation (KDE):
We can fix the volume V and determine the value of k from the data.

k-nearest Neighbors (kNN):
We can fix k and determine the value of V from the data.

We will discuss both approaches below. It can be shown that both approaches **converge to the true probability density** in the limit $N \to \infty$, provided V shrinks suitably with N, and k grows with N.

Kernel Density Estimation (KDE)

- We begin by discussing the **kernel density method** in more detail
- Suppose $\mathcal{R}$ is a small **hypercube** centered on the point $\boldsymbol{x}'$ at which we wish to determine the probability density
- We have to count the number of data points which fall within this hypercube
- To do so, we define the following **kernel function** / **PARZEN window**

$$K(\boldsymbol{u}) := \begin{cases} 1 & \text{if } |u_d| \leqslant 1/2 \qquad d = 1, 2, \ldots, D \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

- Equation (7) represents a hypercube centered on the origin

Kernel Density Estimation (Ctd.)

- The quantity $K\left(\left(\boldsymbol{x}' - \boldsymbol{x}^n\right)/h\right)$ will be 1, if the data point $\boldsymbol{x}^n$ lies inside a cube of size h centered on $\boldsymbol{x}'$, and 0 otherwise
- The total number of data points falling into the hypercube is then given by

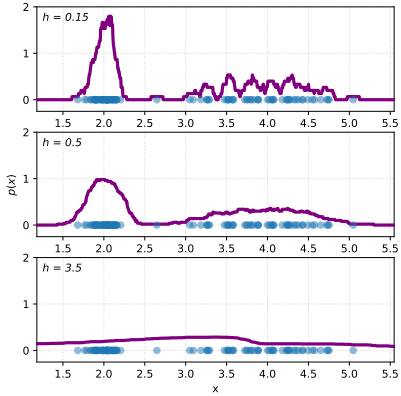
$$k = \sum_{n=1}^{N} K\left(\frac{\boldsymbol{x}' - \boldsymbol{x}^n}{h}\right) \tag{8}$$

- We substitute this equation into (6) to obtain ($V := h^D$):

$$p(\boldsymbol{x}') = \frac{k}{NV} = \frac{1}{Nh^D} \sum_{n=1}^{N} K\left(\frac{\boldsymbol{x}' - \boldsymbol{x}^n}{h}\right) \tag{9}$$

KDE Example – Vanilla Kernel

Note: We see that the kernel density estimator – as it stands – suffers from **artificial discontinuities** (similarly to histograms).



Gaussian Kernel

- We tackle this issue by introducing another kernel function
- The **Gaussian kernel**:

$$K(\boldsymbol{u}) := \frac{1}{\left(\sqrt{2\pi}\right)^D} \exp\left(-\frac{\|\boldsymbol{u}\|^2}{2}\right) \tag{10}$$

- Thus, our density model is obtained by **placing a Gaussian over each data point** and adding up the contributions over the whole dataset, and then dividing by N so that the density is correctly normalized

GAUSSian Kernel

We substitute the GAUSSian kernel (10) into (6) to obtain ($V := h^D$):

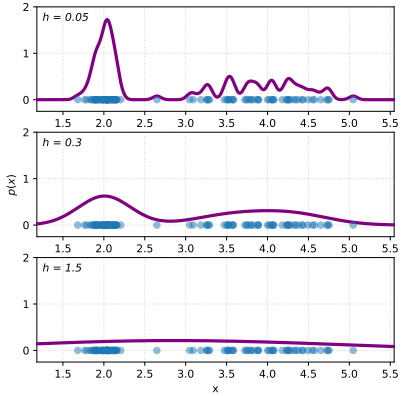
$$p(\boldsymbol{x}') = \frac{1}{Nh^D} \sum_{n=1}^{N} K\left(\frac{\boldsymbol{x}' - \boldsymbol{x}^n}{h}\right)$$

$$= \frac{1}{Nh^D} \sum_{n=1}^{N} \frac{1}{\left(\sqrt{2\pi}\right)^D} \exp\left(-\frac{\left\|\boldsymbol{x}' - \boldsymbol{x}^n\right\|^2}{2h^2}\right) \tag{11}$$

$$= \frac{1}{N\left(\sqrt{2\pi}h\right)^D} \sum_{n=1}^{N} \exp\left(-\frac{\left\|\boldsymbol{x}' - \boldsymbol{x}^n\right\|^2}{2h^2}\right) \tag{12}$$

KDE Example – GAUSSian Kernel

We see that h acts as a smoothing parameter (standard deviation of the GAUSSians) and that if it is set too small (top panel), the result is a **very noisy density model**, whereas if it is set too large (bottom panel), then the **bimodal nature** of the underlying distribution from which the data is generated is **washed out**. **The best density model is obtained for some intermediate value** of h (middle panel).



Conditions for Kernels

- We can choose any other kernel function $K(\boldsymbol{u})$ which satisfies the conditions:

$$K(\boldsymbol{u}) \geqslant 0 \tag{13}$$

$$\int K(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u} = 1 \tag{14}$$

- These conditions ensure that the probability is non-negative and integrates to one
- Another popular kernel function is the **EPANECHNIKOV kernel**

k-nearest Neighbors

- One problem with the kernel approach is that the bandwidth parameter h is **fixed for all kernels**
- This may lead to...
  - ...**over-smoothing** in regions of high density if h is large
  - ...**noisy estimates** in regions with less density if h is too small
- The optimal choice for h may depend on the location within the data space
- This issue is addressed by **nearest-neighbor** methods for density estimation
- We now consider a fixed value of k in equation (6) and aim to find an appropriate value for the volume V

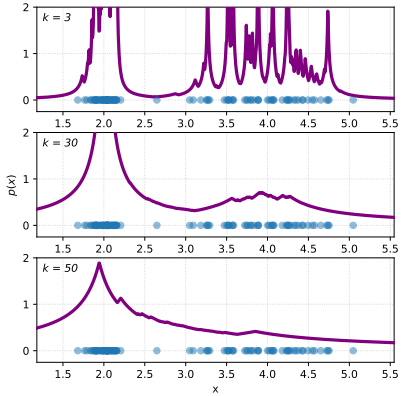
k-nearest Neighbors (Ctd.)

- To find the value of V, we consider a small sphere centered on the point $\boldsymbol{x}'$ at which we wish to estimate the density $p(\boldsymbol{x})$
- We allow the sphere to grow until it precisely contains k data points
- The probability $p(\boldsymbol{x})$ is then given by equation (6) with V set to the volume of the resulting sphere
- The hyperparameter k governs the **degree of smoothing**

Note: The model produced by k-nearest neighbors is **not** a true density model **because the integral over the entire space diverges**.

Example *k*-nearest Neighbors

Note: The larger we choose k, the smoother the resulting density estimate will become!

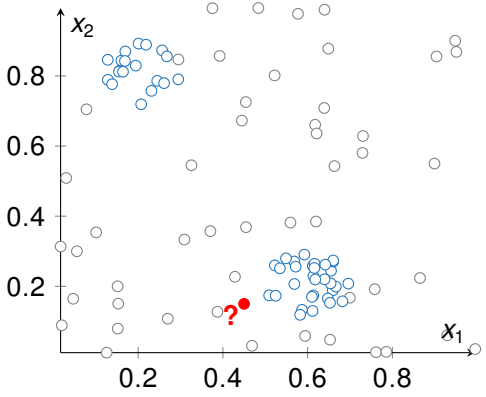


Section:

k-nearest Neighbors for Classification

Derivation of the Algorithm
Distance Metrics
An alternative View: Similarity Metrics
k-nearest Neighbors Algorithm
Choice of k

Introduction



- **Basic idea:** Predict the class label based on nearby known examples
- We do not learn any model parameters
- **The data speaks for itself**
- We refer to this as **instance-based learning** or **lazy learning**

Derivation of the Algorithm

- We can apply the k-nearest neighbor technique for density estimation to **classification problems**
- To do this, we estimate the density for each class separately
- Then we make use of **BAYES' theorem** to get the class posterior probabilities
- **Notation:**
  - Suppose we have N_j data points in class $\mathcal{C}_j$
  - Then $N = \sum_j N_j$ is the total number of data points in the training dataset
  - To classify $\boldsymbol{x}'$, we draw a sphere of volume V centered on $\boldsymbol{x}'$ containing exactly k points irrespective of their class
  - Let k_j be the number of data points in the sphere belonging to class $\mathcal{C}_j$

Derivation of the Algorithm (Ctd.)

- Equation (6) gives us the class conditional and the unconditional density:

$$p(\boldsymbol{x}'|\mathcal{C}_j) = \frac{k_j}{N_j V} \qquad p(\boldsymbol{x}') = \frac{k}{NV} \tag{15}$$

- The prior probability is given by $p(\mathcal{C}_j) = \frac{N_j}{N}$

kNN decision rule / BAYES' theorem:

$$p(\mathcal{C}_j|\boldsymbol{x}') = \frac{p(\boldsymbol{x}'|\mathcal{C}_j) \cdot p(\mathcal{C}_j)}{p(\boldsymbol{x}')} = \frac{k_j}{N_j V} \cdot \frac{N_j}{N} \cdot \frac{NV}{k} = \boxed{\frac{k_j}{k}} \tag{16}$$

BAYES Optimality of *k*-Nearest Neighbors

- We **minimize the misclassification rate** by assigning the test point $\boldsymbol{x}'$ to the class with the **highest posterior probability**
- This is the class which appears most often in the neighborhood of $\boldsymbol{x}'$

Advantages:

- The *k*-nearest neighbors model is **BAYES optimal**
- The model can **capture complex structures** in the data

Disadvantage: Inference is **super expensive**!

Distance Metrics

- **Central question:** How to find the neighborhood of a data point?
- Or put differently: How to measure the distance between two data points $\boldsymbol{x}$ and $\boldsymbol{x}'$?

Let $d : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}_+$ be a function that maps a pair of input vectors onto a positive real number (including zero). d is called **distance metric** if it satisfies the following three properties:

1. $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$ **(commutativity, symmetry)**
2. $d(\boldsymbol{x}, \boldsymbol{y}) = 0 \implies \boldsymbol{x} = \boldsymbol{y}$
3. $d(\boldsymbol{x}, \boldsymbol{y}) \leqslant d(\boldsymbol{x}, \boldsymbol{z}) + d(\boldsymbol{z}, \boldsymbol{y})$ **(triangle inequality)**

Distance Metrics (Ctd.)

Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^M$ be M-dimensional vectors.

Manhattan distance:

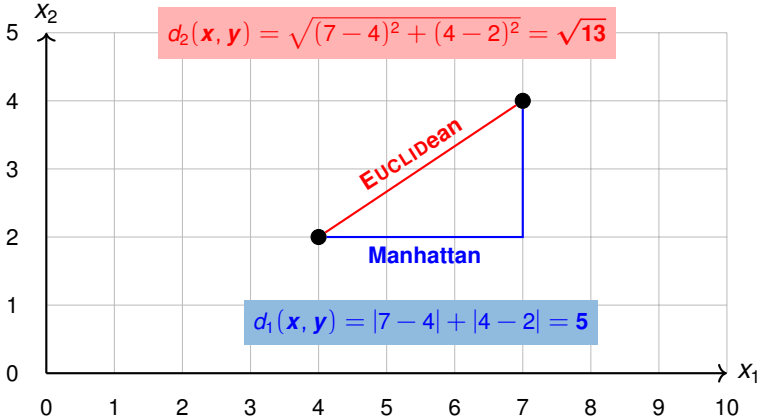
$$d_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{m=1}^{M} |x_m - y_m|$$

EUCLIDean distance:

$$d_2(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{m=1}^{M} |x_m - y_m|^2 \right)^{1/2}$$

Both are instances of the more general **MINKOWSKI distance:** $d_p(\boldsymbol{x}, \boldsymbol{y}) = \sqrt[p]{\sum_{m=1}^{M} |x_m - y_m|^p}$:
$p = 1$ recovers the Manhattan distance, $p = 2$ the EUCLIDean distance.

Distance Metrics (Ctd.)



Cosine Similarity

- **Similarity metrics** are an alternative to distance metrics
- The **cosine similarity** of two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^M$ is the cosine of the angle between the two vectors:

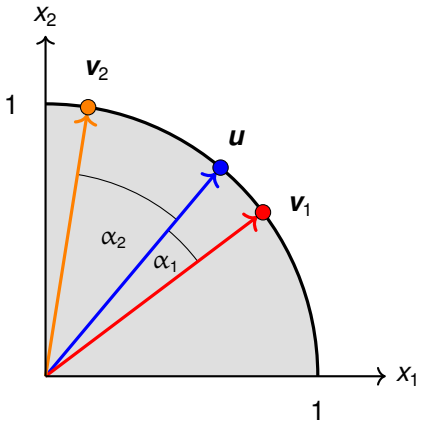
$$\cos \measuredangle(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a}^\top \boldsymbol{b}}{\|\boldsymbol{a}\| \cdot \|\boldsymbol{b}\|} = \frac{\sum_{m=1}^{M} a_m \cdot b_m}{\sqrt{\sum_{m=1}^{M} (a_m)^2} \cdot \sqrt{\sum_{m=1}^{M} (b_m)^2}} \tag{17}$$

- The dot product is defined as (geometric interpretation):

$$\boldsymbol{a}^\top \boldsymbol{b} = \|\boldsymbol{a}\| \cdot \|\boldsymbol{b}\| \cdot \cos \measuredangle(\boldsymbol{a}, \boldsymbol{b}) \tag{18}$$

Cosine Similarity (Ctd.)

- $\boldsymbol{v}_1$ is closer to $\boldsymbol{u}$ than $\boldsymbol{v}_2$ because $\cos(\alpha_1) < \cos(\alpha_2)$
- Remember:
  - $\cos(0°) = 1$ and
  - $\cos(90°) = 0$
- **Do you see any issues?**



Predictions with *k*-Nearest Neighbors

k-Nearest Neighbors Algorithm:

1. Calculate the distances between the test data point $\boldsymbol{x}'$ and **all data points in the training dataset**
2. Sort the data points by distances **in ascending order**
   (sort in descending order if similarity metrics are used)
3. Consider the first k examples and **count how often each class occurs**
4. Predict the class with **the maximum score**

❶ Calculation of Distances

n	x_1	x_2	$\mathcal{C}$	$d_2(\boldsymbol{x}', \boldsymbol{x}^n)$
1	0.66	0.24	1	0.23
2	0.25	0.79	1	0.67
3	0.16	0.81	1	0.73
4	0.57	0.21	1	0.13
5	0.21	0.72	1	0.62
6	0.66	0.27	1	0.24
7	0.27	0.11	0	0.19
8	0.39	0.13	0	0.07
9	0.39	0.86	0	0.71
10	0.44	0.67	0	0.52
11	0.31	0.33	0	0.23
12	0.03	0.51	0	0.55
⋮	⋮	⋮	⋮	⋮

- $\boldsymbol{x}' := \big(0.45, 0.15\big)^\top$
- Calculate the **EUCLIDean distance** between $\boldsymbol{x}'$ and all data points $\boldsymbol{x}^n$ in the training dataset

Caution: Depending on the size of the dataset, predictions might be expensive!

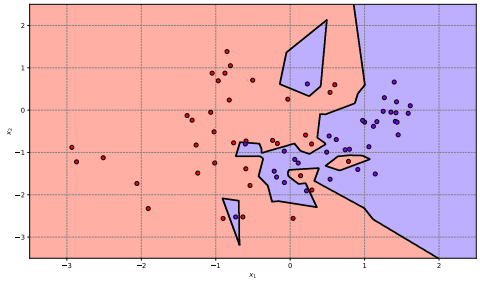
❷/❸/❹ Prediction of the Class Label

- Let k be set to 10
- Step ❷: Sort dataset by distances
  (cf. table on the right)
- Step ❸: Count class occurrences
  - **Class 0:** 3
  - **Class 1:** 7
- Step ❹: **Predict class 1!**

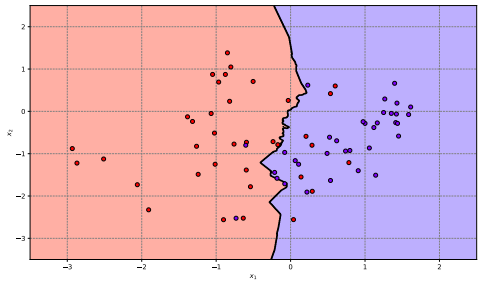
x_1	x_2	$\mathcal{C}$	$d_2(\boldsymbol{x}', \boldsymbol{x}^n)$
0.51	0.17	1	0.06
0.39	0.13	0	0.07
0.52	0.17	1	0.08
0.43	0.23	0	0.08
0.47	0.03	0	0.12
0.52	0.26	1	0.13
0.57	0.21	1	0.13
0.53	0.25	1	0.13
0.58	0.12	1	0.14
0.59	0.13	1	0.14
⋮	⋮	⋮	⋮

How to choose *k*?

k is a hyperparameter of the model. **The choice of *k* is important:**



$k = 1$ (☠ **overfitting** ☠)



$k = 30$ (about right)

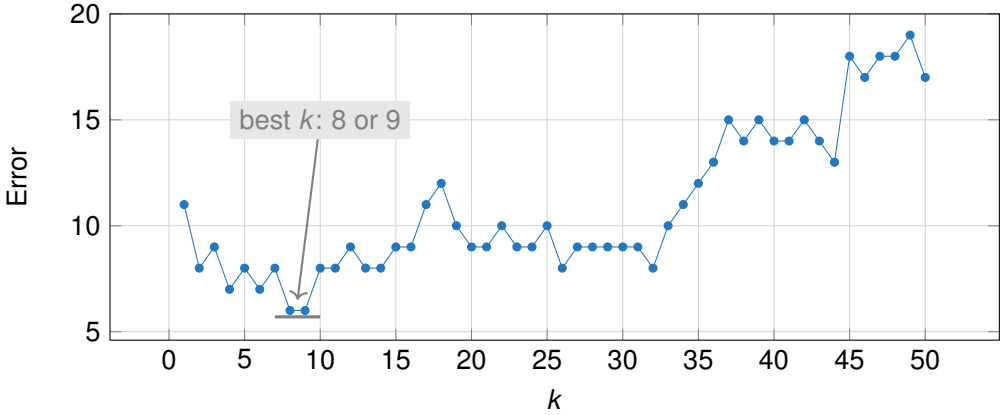
How to choose k? (Ctd.)

- In a **binary classification problem**, it is recommended to use **odd values** for k *(no tie-breaking necessary; at least in binary classification problems)*
- Compute the value of k depending on the size of the dataset $\mathcal{D}$:

$$k := \sqrt{\frac{N}{2}} \qquad \text{or} \qquad k := \sqrt{N} \tag{19}$$

- **Usually better strategy:** Evaluate different values of k on a separate **(!)** development set and choose the best one *(see next slide)*

How to choose *k*? (Ctd.)



Section:

EM Algorithm for GAUSSian Mixture Models

Definition of GAUSSian Mixture Models (GMMs)
Concept of Responsibilities
Parameter Learning via Maximum Likelihood Estimation
Expectation-Maximization Algorithm
Number of Mixture Components

Parametric Density Estimation revisited

- As mentioned earlier, parametric models have **limited modeling capabilities**
- E. g. a GAUSSian distribution **fails to capture multimodal data**
- In this section we will consider a more expressive family of distributions:

Mixture models

- Mixture models describe a distribution $p(\boldsymbol{x})$ by a **convex combination** of K simple (base) distributions $p_k(\boldsymbol{x})$, e. g. GAUSSians, BERNOULLIs, Gammas, etc.:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k p_k(\boldsymbol{x}) \qquad \text{with} \qquad 0 \leqslant \pi_k \leqslant 1 \text{ and } \sum_{k=1}^{K} \pi_k = 1 \tag{20}$$

GAUSSian Mixture Models (GMMs)

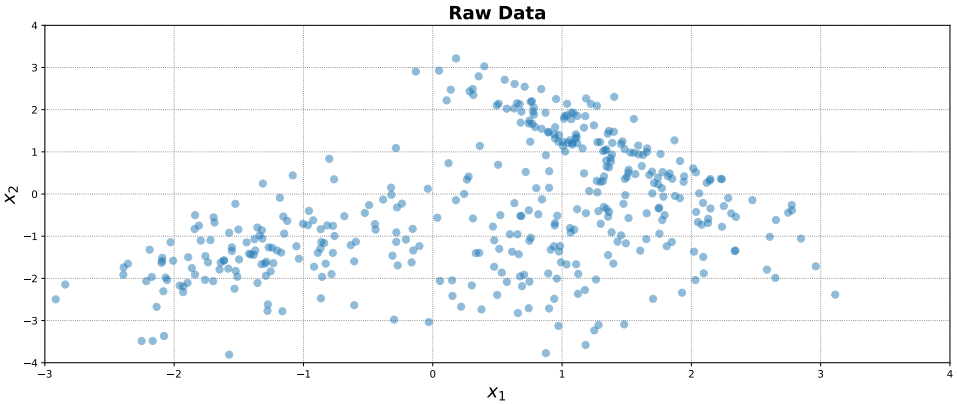
A **GAUSSian mixture model (GMM)** is a density model where we combine a finite number of K GAUSSian distributions $\mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}^k, \boldsymbol{\Sigma}_k\right)$ so that

$$p(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}^k, \boldsymbol{\Sigma}_k\right) \tag{21}$$

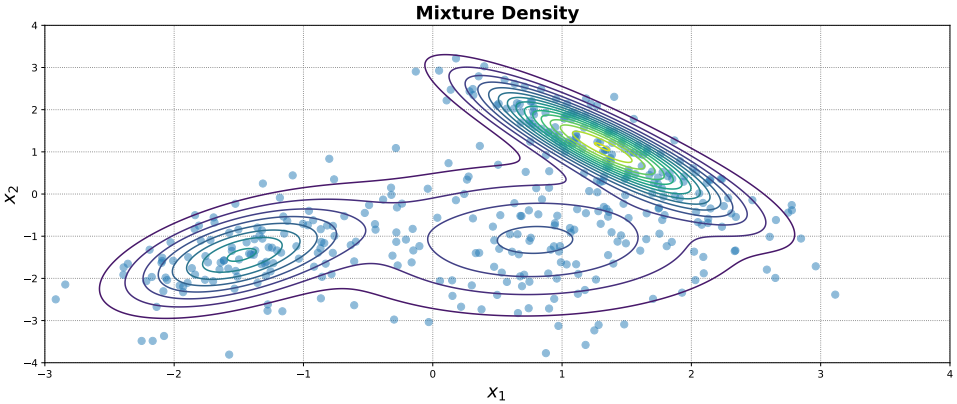
$$0 \leqslant \pi_k \leqslant 1, \qquad \sum_{k=1}^{K} \pi_k = 1, \tag{22}$$

where $\boldsymbol{\theta} := \left\{\pi_k, \boldsymbol{\mu}^k, \boldsymbol{\Sigma}_k : k = 1, \ldots, K\right\}$ are the parameters of the model.

Example: GMM Raw Data



Example: GMM Density



Responsibilities

We define the quantity

$$r_{nk} := \frac{\pi_k \cdot \mathcal{N}\left(\boldsymbol{x}^n; \boldsymbol{\mu}^k, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{K} \pi_j \cdot \mathcal{N}\left(\boldsymbol{x}^n; \boldsymbol{\mu}^j, \boldsymbol{\Sigma}_j\right)} \tag{23}$$

as the **responsibility** of the k-th mixture component for the data point $\boldsymbol{x}^n$.

Remark: Note that $\boldsymbol{r}^n := \left(r_{n1}, \ldots, r_{NK}\right)^\top$ is a **normalized probability vector**, i. e. $\sum_k r_{nk} = 1$ with $r_{nk} \geqslant 0$. We can think of $\boldsymbol{r}^n$ as a *'soft assignment'* of $\boldsymbol{x}^n$ to the K mixture components.

Responsibilities (Ctd.)

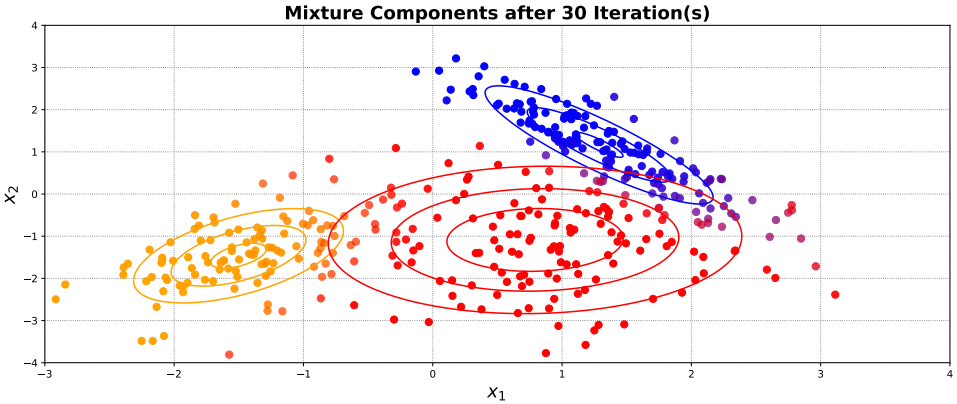
- r_{nk} is **proportional to the likelihood** of $\boldsymbol{x}^n$ with respect to the k-th mixture component
- This likelihood is given by:

$$p\left(\boldsymbol{x}^n; \pi_k, \boldsymbol{\mu}^k, \boldsymbol{\Sigma}_k\right) = \pi_k \mathcal{N}\left(\boldsymbol{x}^n; \boldsymbol{\mu}^k, \boldsymbol{\Sigma}_k\right) \tag{24}$$

- Mixture components have a high responsibility for a data point when it could be a **plausible sample from that mixture component**

r_{nk} is the probability that $\boldsymbol{x}^n$ has been generated by the k-th mixture component!

Visualization of Responsibilities



Training Setup

- Suppose we have a set of N training data points

$$\boldsymbol{X} := \left\{ \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^N \right\}$$

- Our objective is to find a good approximation of the unknown distribution $p(\boldsymbol{x})$ which generated $\boldsymbol{X}$ by means of a GMM with K components
- From the data we have to estimate the parameters $\boldsymbol{\mu}^k$, $\boldsymbol{\Sigma}_k$, and the mixture weights π_k which we summarize in $\boldsymbol{\theta}$
- We will use the **maximum likelihood approach**

Unfortunately, there will be no closed-form solution!

(Log-)Likelihood of a GMM

- We assume the data $\boldsymbol{X}$ to be **i. i. d.** *(independent and identically distributed)*
- This leads to the factorized **likelihood**:

$$p(\boldsymbol{X}; \boldsymbol{\theta}) = \prod_{n=1}^{N} p\left(\boldsymbol{x}^n; \boldsymbol{\theta}\right) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{x}^n; \boldsymbol{\mu}^k, \boldsymbol{\Sigma}_k\right) \tag{25}$$

- The **log-likelihood** is then given by taking the logarithm of the likelihood (25):

$$\mathcal{L}(\boldsymbol{\theta}) := \log\left(p(\boldsymbol{X}; \boldsymbol{\theta})\right) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{x}^n; \boldsymbol{\mu}^k, \boldsymbol{\Sigma}_k\right) \tag{26}$$

Maximization of the Log-Likelihood

- We aim to find parameters $\boldsymbol{\theta}^{\text{ML}}$ which maximize the log-likelihood (26)
- The usual approach would be to compute the gradient $\frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$ of the log-likelihood function, set it to $\mathbf{0}$, and solve for $\boldsymbol{\theta}$
- However, we **cannot obtain a closed-form solution** *(this is because we do not know which component generated which data points; If we did we could use the approach discussed in the last lecture)*
- We are forced to resort to an iterative scheme, called the

Expectation-Maximization algorithm (EM)

Gradient of the Log-Likelihood

We note that any local optimum of a function has the property that its gradient with respect to the parameters **must vanish**. Therefore, we obtain the following necessary conditions:

$$\frac{\partial}{\partial \boldsymbol{\mu}^k}\mathcal{L}(\boldsymbol{\theta}) = \mathbf{0} \iff \sum_{n=1}^{N} \frac{\partial}{\partial \boldsymbol{\mu}^k} \log p\left(\boldsymbol{x}^n; \boldsymbol{\theta}\right) = \mathbf{0} \tag{27}$$

$$\frac{\partial}{\partial \boldsymbol{\Sigma}_k}\mathcal{L}(\boldsymbol{\theta}) = \mathbf{0} \iff \sum_{n=1}^{N} \frac{\partial}{\partial \boldsymbol{\Sigma}_k} \log p\left(\boldsymbol{x}^n; \boldsymbol{\theta}\right) = \mathbf{0} \tag{28}$$

$$\frac{\partial}{\partial \pi_k}\mathcal{L}(\boldsymbol{\theta}) = 0 \iff \sum_{n=1}^{N} \frac{\partial}{\partial \pi_k} \log p\left(\boldsymbol{x}^n; \boldsymbol{\theta}\right) = 0 \tag{29}$$

Updating the Means

Lemma (Update of the GMM means): The update of the mean parameters $\boldsymbol{\mu}^k$, $k = 1, \ldots, K$, of the GMM is given by

$$\left(\boldsymbol{\mu}^k\right)^{\text{new}} := \frac{\sum_{n=1}^{N} r_{nk} \boldsymbol{x}^n}{\sum_{n=1}^{N} r_{nk}}, \tag{30}$$

where the responsibilities r_{nk} are defined by (23).

Proof: See [Deisenroth.2019], chapter 11. ■

Updating the Covariances

Lemma (Update of the GMM covariances): The update of the covariance parameters $\boldsymbol{\Sigma}_k$, $k = 1, \ldots, K$, of the GMM is given by

$$\boldsymbol{\Sigma}_k^{\text{new}} := \frac{\sum_{n=1}^{N} r_{nk} \left(\boldsymbol{x}^n - \boldsymbol{\mu}^k\right) \left(\boldsymbol{x}^n - \boldsymbol{\mu}^k\right)^\top}{\sum_{n=1}^{N} r_{nk}}, \tag{31}$$

where the responsibilities r_{nk} are defined by (23).

Proof: See [Deisenroth.2019], chapter 11. ■

Updating the Mixture Weights

Lemma (Update of the GMM mixture weights): The mixture weights of the GMM are updated according to

$$\pi_k^{\text{new}} := \frac{1}{N} \sum_{n=1}^{N} r_{nk}, \qquad k = 1, \ldots, K, \tag{32}$$

where the responsibilities r_{nk} are defined by (23).

Proof: See [Deisenroth.2019], chapter 11. ■

Expectation-Maximization (EM) Algorithm for GMMs

WHILE **not converged** DO

1. Initialize π_k, $\boldsymbol{\mu}^k$, $\boldsymbol{\Sigma}_k$ for $k = 1, \ldots, K$
2. **E-step:** Evaluate the responsibilities r_{nk} for every data point $\boldsymbol{x}^n$ and every mixture component using the current parameters π_k, $\boldsymbol{\mu}^k$, $\boldsymbol{\Sigma}_k$:

$$r_{nk} := \frac{\pi_k \cdot \mathcal{N}\left(\boldsymbol{x}^n; \boldsymbol{\mu}^k, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{K} \pi_j \cdot \mathcal{N}\left(\boldsymbol{x}^n; \boldsymbol{\mu}^j, \boldsymbol{\Sigma}_j\right)}$$

$$\vdots$$

Expectation-Maximization (EM) Algorithm for GMMs (Ctd.)

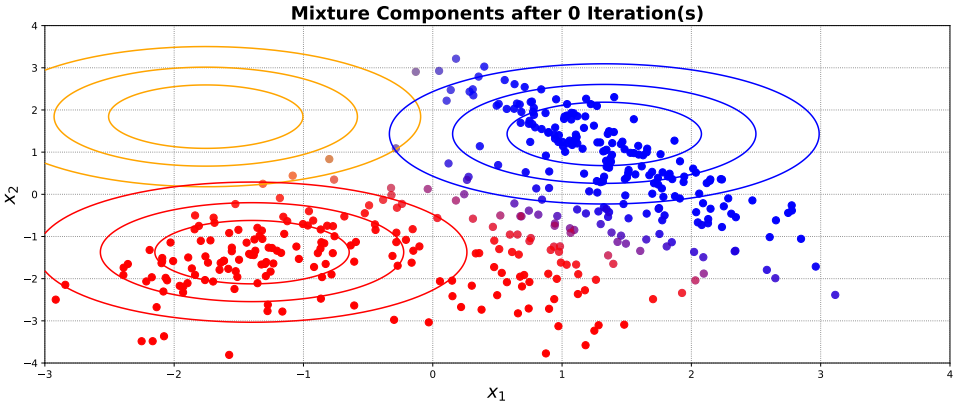
❸ **M-step:** Re-estimate the parameters π_k, $\boldsymbol{\mu}^k$, $\boldsymbol{\Sigma}_k$ for $k = 1, \ldots, K$ using the responsibilities obtained in the E-step above:

$$\pi_k^{\text{new}} := \frac{1}{N} \sum_{n=1}^{N} r_{nk} \qquad \text{and} \qquad \left(\boldsymbol{\mu}^k\right)^{\text{new}} := \frac{\sum_{n=1}^{N} r_{nk} \boldsymbol{x}^n}{\sum_{n=1}^{N} r_{nk}} \qquad \text{and}$$

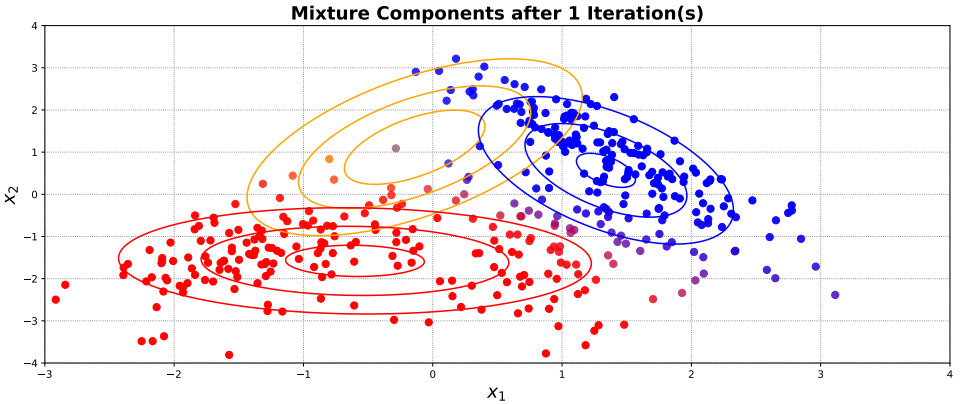
$$\boldsymbol{\Sigma}_k^{\text{new}} := \frac{\sum_{n=1}^{N} r_{nk} \left(\boldsymbol{x}^n - \boldsymbol{\mu}^k\right) \left(\boldsymbol{x}^n - \boldsymbol{\mu}^k\right)^\top}{\sum_{n=1}^{N} r_{nk}}$$

END

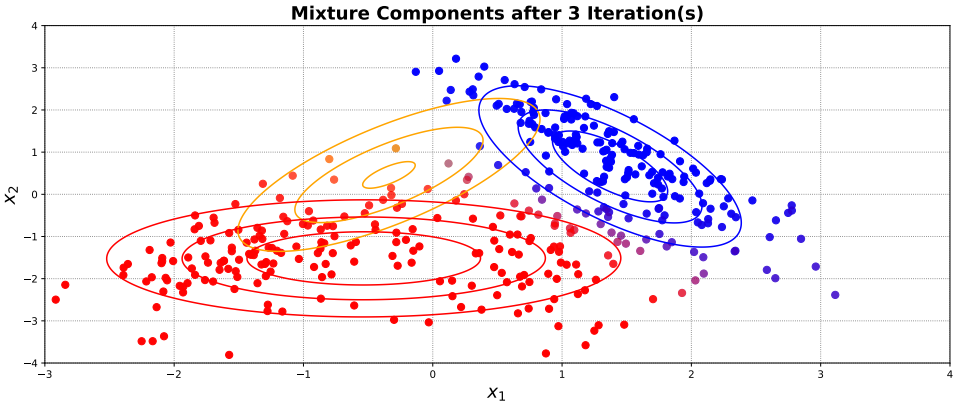
Visualization of the EM Algorithm



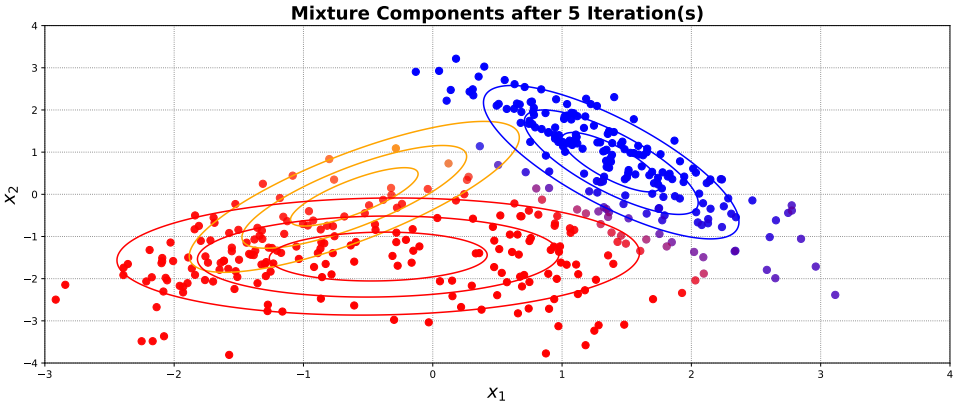
Visualization of the EM Algorithm (Ctd.)



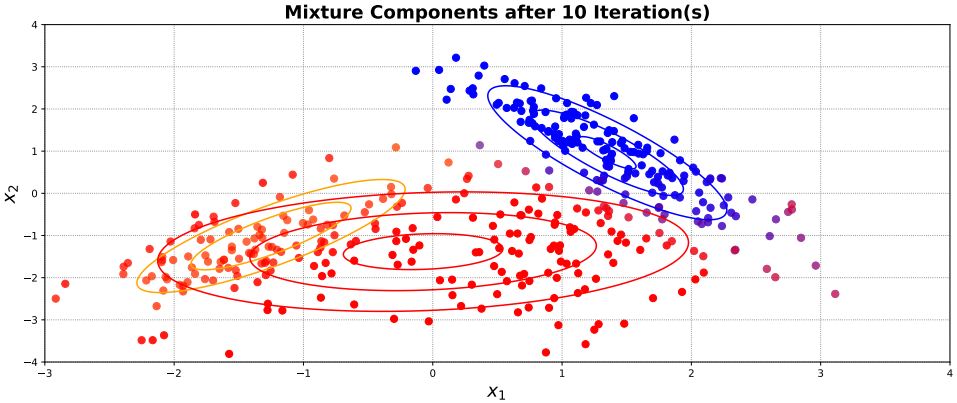
Visualization of the EM Algorithm (Ctd.)



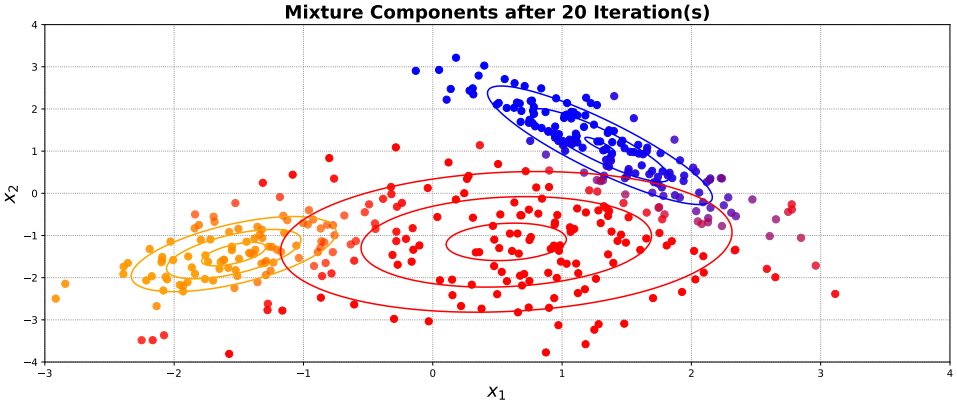
Visualization of the EM Algorithm (Ctd.)



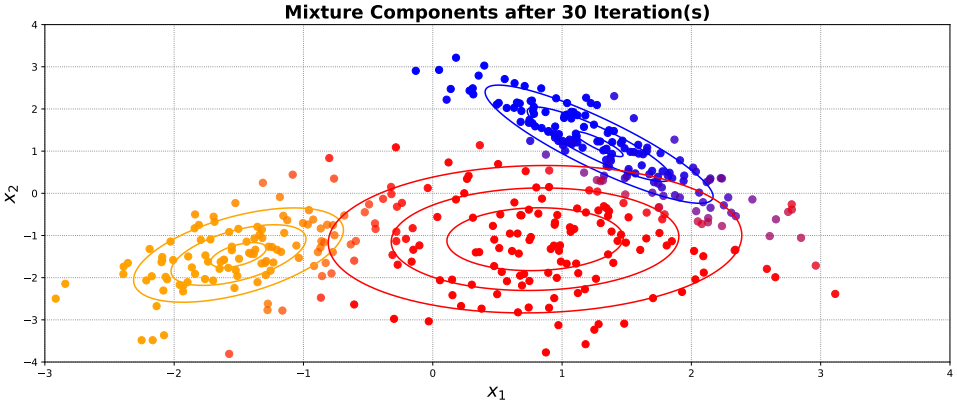
Visualization of the EM Algorithm (Ctd.)



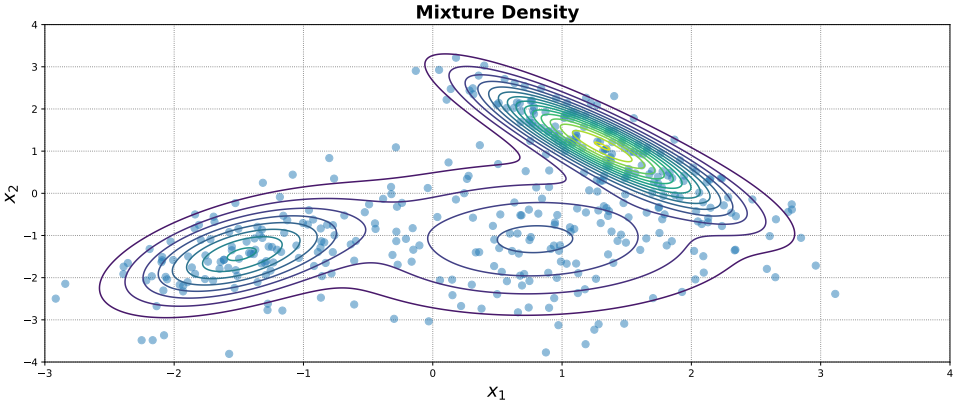
Visualization of the EM Algorithm (Ctd.)



Visualization of the EM Algorithm (Ctd.)



Visualization of the EM Algorithm (Ctd.)



Choice of the Number of Mixture Components

BAYESian Information Criterion (BIC):

$$\text{BIC} := \log p\left(\boldsymbol{X}; \boldsymbol{\theta}^{\text{ML}}\right) - \frac{1}{2} K \log N \tag{33}$$

K is the number of mixture components and N is the number of data points

AKAIKE Information Criterion (AIC):

$$\text{AIC} := \log p\left(\boldsymbol{X}; \boldsymbol{\theta}^{\text{ML}}\right) - K \tag{34}$$

Computation of the BIC in the above Example

- `sklearn` implements the BIC in the `GaussianMixture` class
- Let us compute the BIC scores for several values for K *(the smaller the better!)*:

K	BIC Score
1	2,785.43
2	2,514.44
3	2,463.05
4	2,485.60
5	2,519.81

Section:

Wrap-Up

Summary
Recommended Literature
Self-Test Questions
Lecture Outlook

Summary

1. **Non-parametric density estimation:**
   - Parametric models **might not capture the structure** of the data
   - Non-parametric methods **do not assume a fixed parametric form**
   - Examples:
     - Histograms *(very basic)*
     - Kernel density estimation
     - *k*-nearest neighbors
   - Kernel density estimators usually provide good results when using appropriate **kernels**

Summary (Ctd.)

2. ***k*-nearest neighbors:**
   - The basic idea is to classify unknown instances **based on nearby examples**
   - The algorithm is an example of **instance-based learning**
   - **Distance metrics** allow to calculate the distance between data points:
     - Manhattan distance
     - EUCLIDean distance
     - Cosine similarity (as an alternative to distance metrics)
   - Choose the value of k wisely:
     - Too small: **Overfitting**
     - Too large: **Underfitting**

Summary (Ctd.)

3. **GAUSSian mixture models:**
   - An alternative to non-parametric models is to use **multiple parametric models**
   - The base distributions can be chosen freely
   - The most common model is to use a **mixture of GAUSSian distributions**
   - There is **no closed-form solution**, we have to resort to numerical methods
   - This leads to the **expectation-maximization (EM)** algorithm
   - Choose the number of mixture components using the
     - **BAYESian information criterion**
     - **AKAIKE information criterion**

Recommended Literature

1. **Non-parametric density estimation:**
   - [BISHOP.2006], chapter 2.5, pages 120 – 125
2. ***k*-nearest neighbors:**
   - [BISHOP.2006], chapter 2.5, pages 125 – 127
3. **GAUSSian mixture models:**
   - [DEISENROTH.2019], chapter 11

(For free PDF versions, see list in GitHub readme!)

Self-Test Questions

1. Outline the k-nearest neighbors algorithm.
2. What is instance-based learning (in contrast to model-based learning)?
3. How can you compute distances? What properties do distance metrics have?
4. What is the intuition behind the triangle inequality?
5. How can you choose k?
6. Suppose you have a dataset comprising $N = 50$ examples.
   If you set $k := N$, what class does the algorithm predict?
7. What are advantages and disadvantages of the algorithm?

What's next...?

	I	Machine Learning Introduction
	II	Optimization Techniques
	III	Bayesian Decision Theory
	IV	Non-parametric Density Estimation
•	**V**	Probabilistic Graphical Models
	VI	Linear Regression
	VII	Logistic Regression
	VIII	Deep Learning
	IX	Evaluation
	X	Decision Trees
	XI	Support Vector Machines
	XII	Clustering
	XIII	Principal Component Analysis
	XIV	Reinforcement Learning
	XV	Advanced Regression

Thank you very much for the attention!

*** * * Artificial Intelligence and Machine Learning * * ***

Topic: Non-parametric Density Estimation and the Expectation-Maximization (EM) Algorithm

Term: Summer term 2025

Contact:
Daniel Wehner, M.Sc.
SAP SE / DHBW Mannheim
daniel.wehner@sap.com

Do you have any questions?