GMMs

June 22, 2025

Perfect, let's begin the presentation.

Slide 1: Gaussian Mixture Models (GMMs)

"Good [morning/afternoon], everyone.

Welcome to this session on **Gaussian Mixture Models**, or simply **GMMs**, which are a fundamental tool in probabilistic modeling and unsupervised learning.

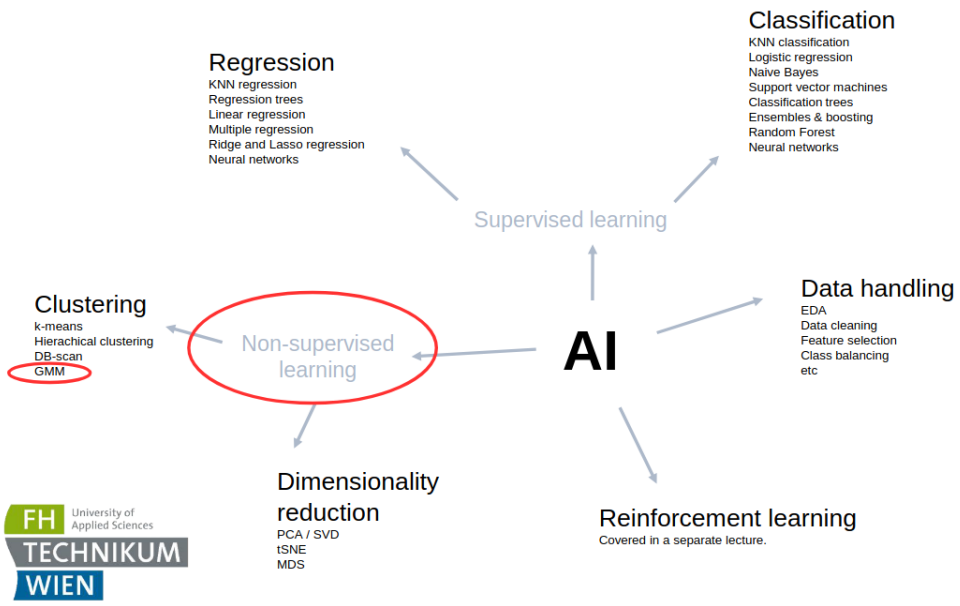
This presentation is part of the **ML2: AI Concepts and Algorithms** course for the Summer Semester 2025, here at the **Faculty of Computer Science and Applied Mathematics**, at the **University of Applied Sciences Technikum Wien**.

The lecture today is delivered by **Rosana Gomes**, and the material was prepared by **S. Rezagholi** and **R.O. Gomes**.

In this talk, we'll explore how GMMs allow us to move beyond the limitations of traditional clustering methods like k-means, and how they leverage the power of probabilistic modeling to produce soft assignments of data points to clusters.

Let's move on to understand the motivation behind using GMMs."

Please send the next slide when ready.



Great, let's continue with the second slide.

Slide 2: GMM in the AI Landscape

"As we dive into Gaussian Mixture Models, it's important to understand where they fit in the broader landscape of Artificial Intelligence.

AI methods are commonly divided into **supervised**, **unsupervised**, and **reinforcement learning** categories.

GMMs are part of the **unsupervised learning** family — specifically under the umbrella of **clustering algorithms**. While supervised methods rely on labeled data, clustering aims to uncover hidden patterns or groupings within **unlabeled data**.

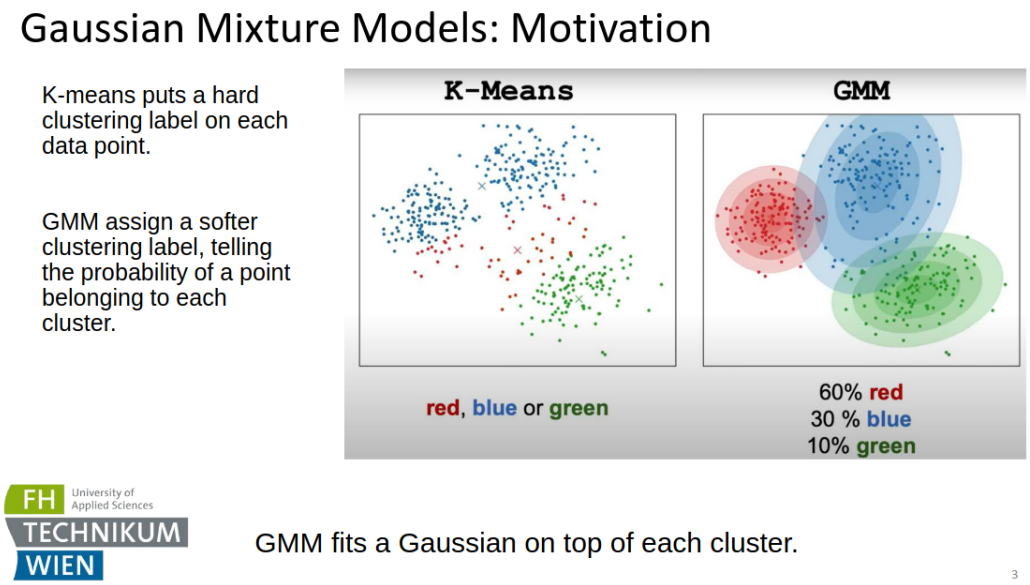
As you can see here, GMMs are listed alongside other clustering techniques like **k-means**, **hierarchical clustering**, and **DB-scan**.

However, unlike k-means — which assigns data points to a single cluster — GMMs offer a **probabilistic approach**, allowing for **soft assignments**. This makes them much more flexible in modeling real-world data where boundaries between clusters are often fuzzy.

Throughout this presentation, we'll see how this flexibility is a key strength of Gaussian Mixture Models.

Let's now explore what motivates the use of GMMs in more detail."

Send the next slide when ready.



Excellent — here's how we present this third slide.

Slide 3: Gaussian Mixture Models — Motivation

"This slide helps us understand **why** Gaussian Mixture Models are valuable, especially in comparison to k-means.

On the **left**, we see the traditional **k-means** clustering result. Each data point is assigned a **hard label** — it's either red, blue, or green. There's no room for uncertainty, even for those points that lie near the boundaries.

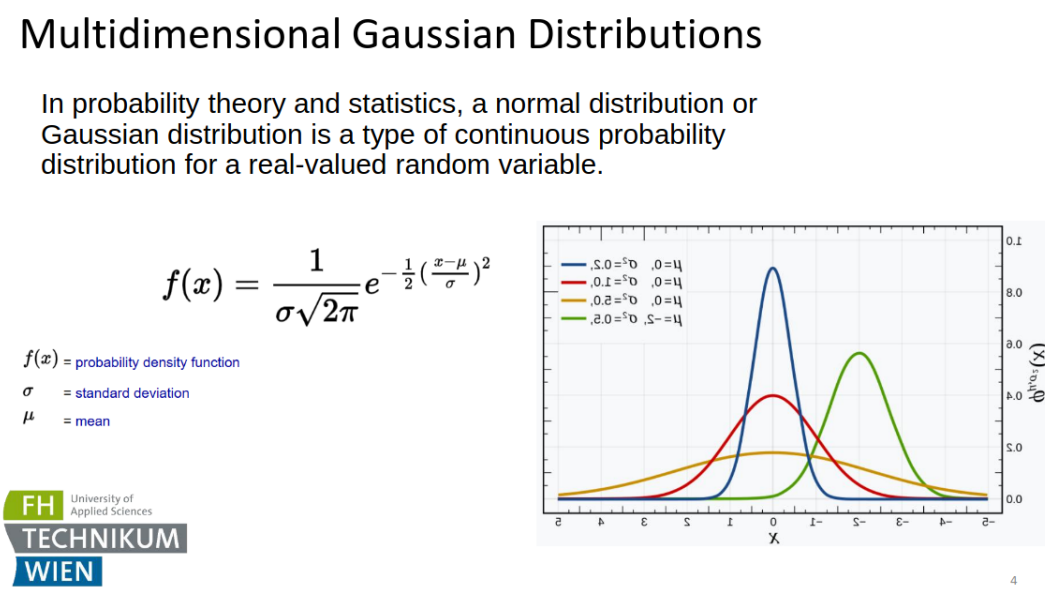
On the **right**, we see the result of a **Gaussian Mixture Model**. Unlike k-means, GMMs offer **soft clustering**, meaning each data point is assigned a **probability** of belonging to each cluster.

So for example, one point might be 60% likely to be red, 30% blue, and 10% green. This is much more informative when clusters overlap or when data points lie in ambiguous regions.

What makes this possible is that GMM fits a **Gaussian distribution** — a bell-shaped probability curve — **on top of each cluster**. These Gaussians can overlap and vary in size, shape, and orientation, making GMMs much more flexible and powerful.

Let's now take a closer look at how Gaussian distributions work, especially in multiple dimensions."

Ready for the next slide when you are.



Excellent — here is the script for slide 4.

Slide 4: Multidimensional Gaussian Distributions

"Before we get into the mechanics of Gaussian Mixture Models, let's quickly review the **Gaussian distribution** — also known as the **normal distribution** — which forms the core building block of GMMs.

As shown here, the probability density function of a **1-dimensional Gaussian** is given by this formula:

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

Here,

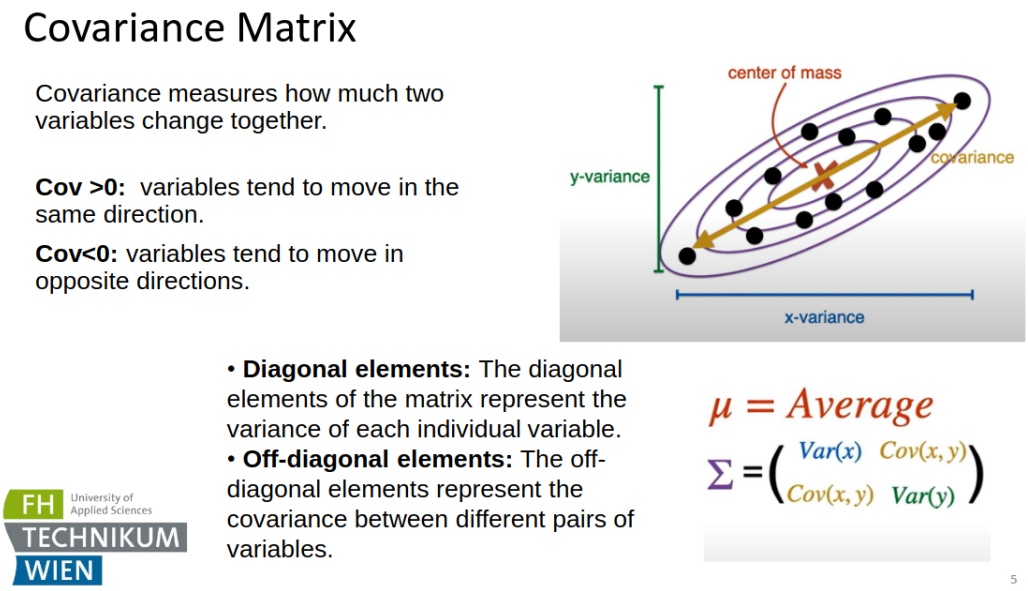
- μ is the **mean** — the center of the distribution,
- σ is the **standard deviation**, which controls the spread, and
- $f(x)$ is the **probability density function**, showing how likely we are to observe a value x.

On the right, you can see several Gaussian curves with different combinations of μ and σ. Notice how shifting the mean moves the curve horizontally, while changing the standard deviation compresses or spreads it vertically.

In real applications, especially in clustering, we deal with **multidimensional data**, so we'll later extend this to **multivariate Gaussian distributions** that describe ellipses instead of simple curves.

Next, we'll explore the concept of covariance — which plays a key role in defining the shape of these multivariate Gaussians."

Please send the next slide when ready.



Great, here's the presentation script for **slide 5**.

Slide 5: Covariance Matrix

"To work with **multidimensional Gaussians**, we need to understand how variables interact with each other. This is where the **covariance matrix** comes into play.

Covariance measures how much two variables **change together**:

- If the covariance is **greater than zero**, the variables tend to move in the **same direction**.
- If it's **less than zero**, they move in **opposite directions**.

In two dimensions, this interaction is captured in a **covariance matrix**, often denoted by the Greek letter **Σ (Sigma)**.

This matrix has:

- **Diagonal elements**: which represent the **variance** of each individual variable — for example, Var(x) and Var(y).
- **Off-diagonal elements**: which capture the **covariance between pairs of variables** — like Cov(x, y) and Cov(y, x).

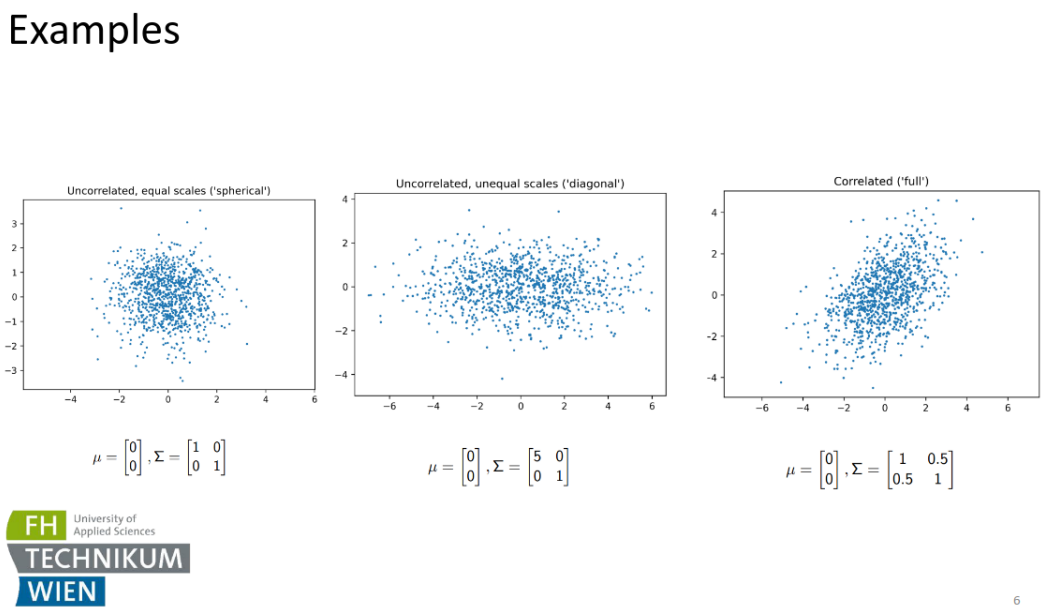
On the right, we can see a **visual example**. Each black dot is a data point. The ellipses represent different levels of probability density. The orientation and shape of these ellipses are determined by the covariance matrix.

So, when fitting a Gaussian to a cluster, GMM doesn't just estimate the center and spread — it also captures the **directional relationships** between variables.

This ability to represent elliptical and oriented clusters gives GMMs a major edge over simple spherical models like k-means.

Let's now look at some concrete examples of GMMs in practice."

Let me know when you're ready with the next slide.



Great — here's the presentation script for **slide 6**.

Slide 6: Examples of Gaussian Distributions

"This slide shows three concrete examples of how Gaussian distributions can differ depending on the values in the **covariance matrix**.

Let's go through them one by one:

- On the **left**, we have a spherical Gaussian. The covariance matrix here is:

$$\Sigma = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$$

  This means there's **equal variance** in all directions and **no correlation** between the x and y variables. The result is a **perfect circle** around the mean.

- In the **middle**, we still have **no correlation**, but the variances differ. The covariance matrix is:

$$\Sigma = \begin{bmatrix}5 & 0\\0 & 1\end{bmatrix}$$

This stretches the distribution along the x-axis, creating an **elliptical shape** aligned with the axes.

- On the **right**, we see the most general case: A **correlated Gaussian** with a covariance matrix:

$$\Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

The off-diagonal values (0.5) indicate a **positive correlation** — as one variable increases, so does the other. The result is a **tilted ellipse**.

These examples illustrate how GMMs can model a wide variety of cluster shapes by adjusting the covariance matrix. This flexibility is what makes them so powerful compared to rigid models like k-means.

Now let's dive into the actual structure and mathematics behind Gaussian Mixture Models."

Ready when you are for the next slide.

Gaussian Mixture Models (GMMs)

The Gaussian mixture distribution can be written as a linear superposition of Gaussians in the form

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

parameters

where the parameters $\{\pi_k\}$ must satisfy

$$0 \leqslant \pi_k \leqslant 1$$

together with

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

Mixture model follow two-stage process for data generation:

1. Choose the Gaussian to sample (define distribution parameters)
2. Sample from the respective distribution: x ~ fi

FH University of Applied Sciences TECHNIKUM WIEN

7

Perfect — here's the script for **slide 7**.

Slide 7: Gaussian Mixture Models (GMMs)

"Now we formalize what a **Gaussian Mixture Model** actually is.

GMMs model a probability distribution as a **weighted sum of multiple Gaussian components**. Mathematically, this is expressed as:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}(\mathbf{x} \mid \mu_k, \Sigma_k)$$

Let's break this down:

- π_k is the **mixing coefficient** for the k-th Gaussian — it defines the **weight** or relative importance of that component. These weights must satisfy two conditions:
  1. Each π_k must be between 0 and 1,
  2. All the weights must sum up to 1.
- μ_k is the **mean vector** of the k-th Gaussian — indicating its center.
- Σ_k is the **covariance matrix** — controlling the shape and orientation of the k-th Gaussian.

This model reflects a **two-stage generative process**:

1. First, you choose one of the K Gaussians based on the probabilities π_k,
2. Then, you sample a point $\mathbf{x}$ from that chosen Gaussian.

In essence, GMMs allow us to represent **complex data distributions** by combining simpler Gaussian components. This is incredibly useful in modeling data with **overlapping clusters**.

Next, we'll see how GMM uses these components to perform probabilistic clustering — starting with the concept of 'responsibility'."

Feel free to send the next slide when you're ready.

Gaussian Mixture Models (GMMs)

Responsibility of k for x:
The probability that observation x comes from component k

$$\frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

Clustering: assign the datapoint x to the component with the highest responsibility for the point.

FH University of Applied Sciences TECHNIKUM WIEN

8

Excellent — here's the script for **slide 8**.

Slide 8: Responsibility — Soft Assignment in GMMs

"Once we have a mixture of Gaussians defined, the next step is to compute the **responsibility** of each component for each data point.

This is a core concept in GMMs — it tells us:

> *What is the probability that a given point* $\mathbf{x}$ *was generated by Gaussian component* k*?*

Mathematically, the **responsibility** $\gamma(z_k)$ is defined as:

$$\gamma(z_k) = \frac{\pi_k \, \mathcal{N}(\mathbf{x} \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \, \mathcal{N}(\mathbf{x} \mid \mu_j, \Sigma_j)}$$

This is a normalized probability:

- The numerator gives the likelihood of point $\mathbf{x}$ under Gaussian k, weighted by π_k.
- The denominator sums up the likelihoods across **all** components, ensuring the result is a true probability between 0 and 1.

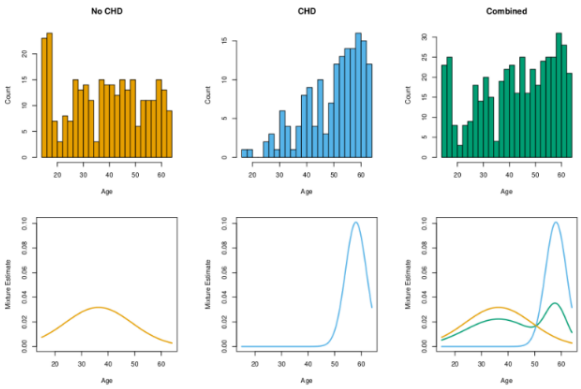
So for each data point, we don't just assign it to a single cluster like in k-means — instead, we compute a **soft assignment** across all clusters.

In **clustering applications**, we typically assign a point to the cluster with the **highest responsibility**, but we can also use the full distribution for **probabilistic reasoning** or **anomaly detection**.

This mechanism is a key part of the **Expectation-Maximization algorithm**, which we'll look at next."

Please go ahead with the next slide.

Example: Binary Problem



[Source: Hastie et al. (2017): The Elements of Statistical Learning.]

FH University of Applied Sciences TECHNIKUM WIEN

Great — here's the presentation script for **slide 9**.

Slide 9: Example — Binary Problem

"This slide presents a real-world **binary classification** example, adapted from *The Elements of Statistical Learning* by Hastie et al.

The dataset distinguishes between two groups:

- People **with CHD** (coronary heart disease)
- And those **without CHD**

We're looking at the **age distribution** across these two classes:

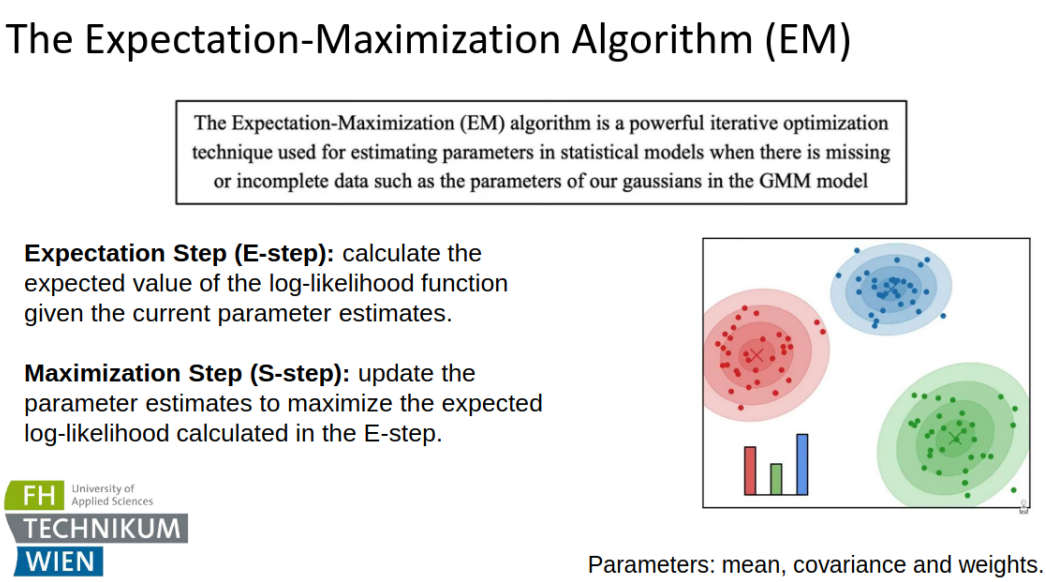
- The **top row** shows histograms of age:
  - On the **left**, we see the distribution for people **without CHD**. It's fairly spread out.
  - In the **middle**, those **with CHD** tend to be older.
  - On the **right**, we have the **combined** population.
- The **bottom row** shows how Gaussian Mixture Models can fit these distributions:
  - Each curve is a **mixture estimate** — in other words, a combination of Gaussians that approximates the histogram above it.
  - These curves show that GMMs are capable of flexibly modeling different distributions for each class.

By fitting a GMM to each class, we can later compute **class-conditional probabilities**, which can be very useful in **probabilistic classification** and even in **anomaly detection** tasks.

This example gives a glimpse of how GMMs go beyond simple clustering — they can help us **model and reason about distributions** in a structured and interpretable way.

Let's now move on to the algorithm that powers GMM training: the **Expectation-Maximization algorithm**, or EM."

Ready when you are for the next slide.



Perfect — here's the presentation script for **slide 10**.

Slide 10: The Expectation-Maximization Algorithm (EM)

"To estimate the parameters of a Gaussian Mixture Model, we use an iterative optimization method called the **Expectation-Maximization algorithm**, or **EM**.

This algorithm is especially useful when the data is **incomplete or latent** — which is exactly the case with GMMs, where we don't know which component generated which point.

EM consists of two repeating steps:

Expectation Step (E-step) In this step, we **compute the responsibilities** — the expected value of the hidden variables — using the current estimates of the parameters.

Put simply: We estimate how likely it is that each point belongs to each Gaussian component.

Maximization Step (M-step) Here, we **update the parameters** — that is, the means μ_k, covariances Σ_k, and weights π_k — to **maximize** the expected log-likelihood we calculated in the E-step.

Visually, as shown on the right:

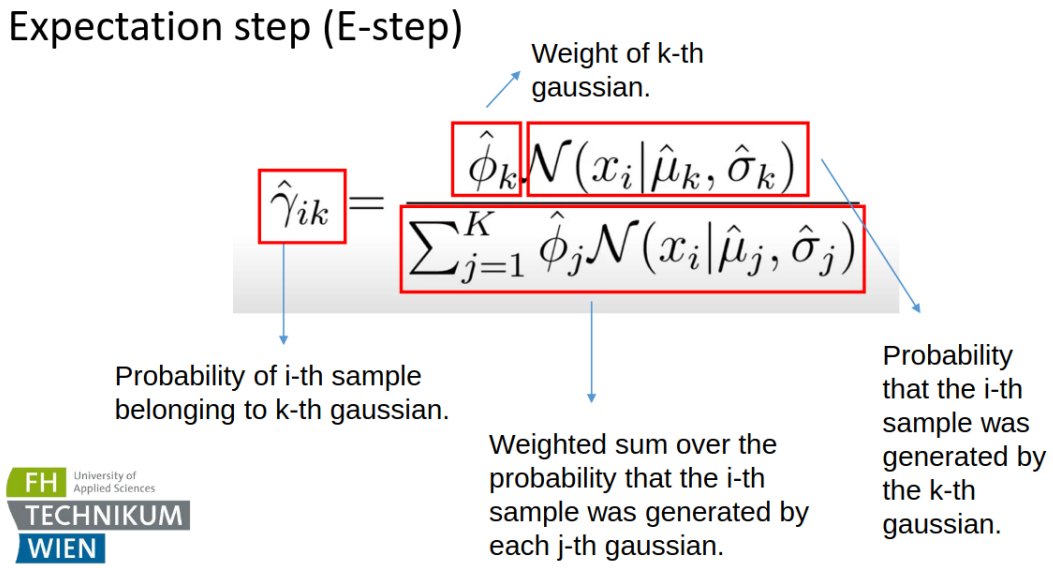
- We begin with an initial guess of where the Gaussians are.
- Then, we alternate between reassigning points to clusters (softly), and refining the shape and location of those Gaussians.

- This continues until the process **converges** — usually when the log-likelihood no longer improves significantly.

This elegant cycle allows GMMs to learn **both the structure of the data and the clustering responsibilities** without needing any labels.

Next, we'll look at the **details of the E-step** in action."

Send the next slide whenever you're ready.



Great — here's the script for **slide 11**.

Slide 11: Expectation Step (E-step)

"Now let's zoom into the **Expectation Step** of the EM algorithm — the part where we compute **responsibilities** for each data point.

The formula here defines $\hat{\gamma}_{ik}$, which is:

> the probability that the **i-th sample** belongs to the **k-th Gaussian**.

This is a soft assignment, meaning it gives us a value between 0 and 1 rather than a hard cluster label.

Let's break it down:

- In the **numerator**, we multiply:
  - $\hat{\phi}_k$, the **weight** (or mixing coefficient) of the k-th Gaussian,
  - by the value of the **Gaussian density function** evaluated at x_i, using the current estimates of the mean and variance: $\hat{\mu}_k$ and $\hat{\sigma}_k$.

- The **denominator** is a **normalization term** — it sums over all components j from 1 to K, applying the same Gaussian formula but for each component.

The result is a **normalized responsibility**:

> *How confident are we that point x_i was generated by component k, based on the current model?*

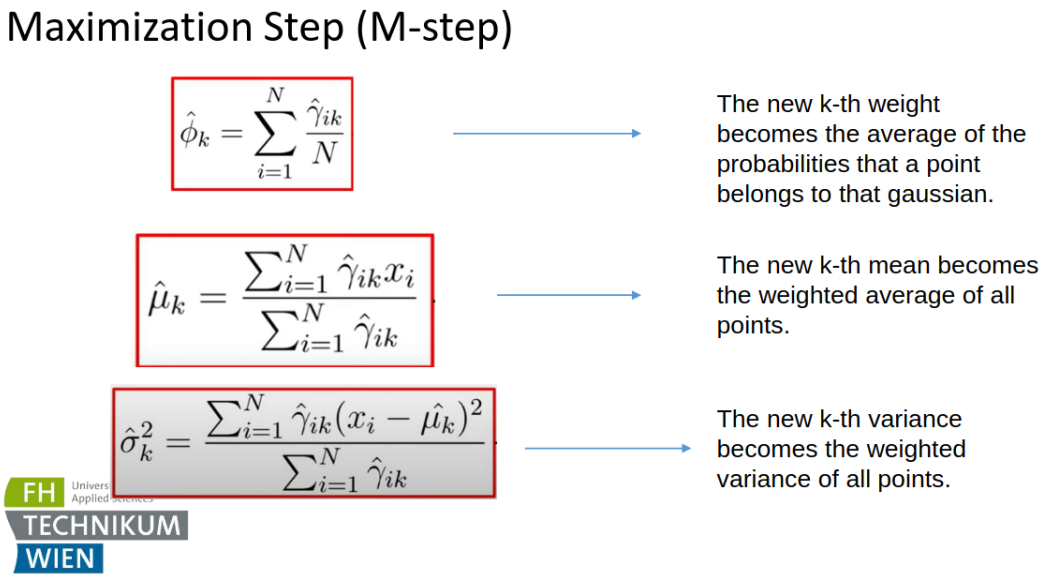
These values are then used in the **next M-step** to refine the parameters.

So this step doesn't move or change the model yet — it just asks:

> "Given what I currently know, how likely is each Gaussian to have generated this point?"

We'll now see how we use this information in the **Maximization Step**."

Please continue with the next slide.



Perfect — here's the script for **slide 12**.

Slide 12: Maximization Step (M-step)

"Now that we've computed the responsibilities in the **E-step**, we move to the **Maximization Step**, where we use those values to **update the parameters** of our Gaussians.

Let's break it down:

Updating the weights $\hat{\phi}_k$:

$$\hat{\phi}_k = \frac{1}{N} \sum_{i=1}^{N} \hat{\gamma}_{ik}$$

This is the **average responsibility** that Gaussian k has for all data points. It updates the **mixing coefficient** — the probability of choosing this Gaussian in the generative process.

Updating the mean $\hat{\mu}_k$:

$$\hat{\mu}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik} x_i}{\sum_{i=1}^{N} \hat{\gamma}_{ik}}$$

This is the **weighted average** of all data points, where each point's influence depends on how strongly it is associated with component k.

Updating the variance $\hat{\sigma}_k^2$:

$$\hat{\sigma}_k^2 = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik} (x_i - \hat{\mu}_k)^2}{\sum_{i=1}^{N} \hat{\gamma}_{ik}}$$

This captures how spread out the points are around the new mean — again, weighted by their responsibilities.

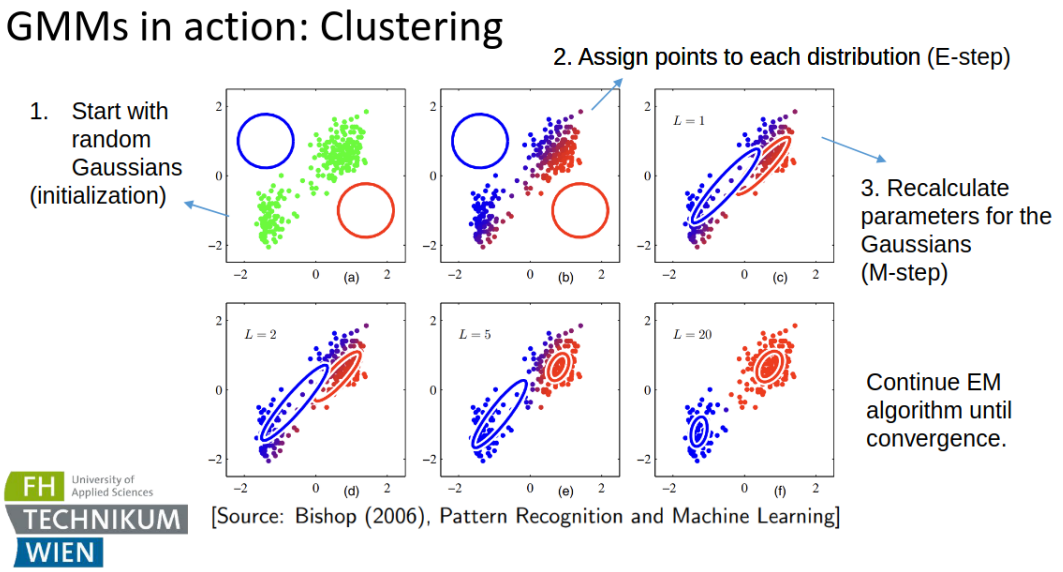
So, to summarize:

- In the **E-step**, we compute how likely each point is to belong to each Gaussian.
- In the **M-step**, we use those probabilities to update the Gaussians' parameters.

This process **iterates until convergence**, forming the full EM algorithm loop.

Next, we'll see this process in action through a visual example of GMM-based clustering."

You may proceed with the next slide.



Excellent — here is the script for **slide 13**.

Slide 13: GMMs in Action — Clustering with EM

"This slide shows a step-by-step visual example of how the **EM algorithm trains a Gaussian Mixture Model** for clustering.

Let's walk through the process:

---

1 **Initialization** In image (a), we start with a random guess: the Gaussian distributions are initialized with arbitrary means and covariances. The data points — shown in green — haven't been assigned yet.

---

2 **E-step: Assign Responsibilities** In (b), each point is softly assigned to a Gaussian based on its current probability — illustrated by the **color gradients** between blue and red.

---

3 **M-step: Update Parameters** In (c), after one iteration, the model updates the **means and covariances** of the Gaussians based on those soft assignments.

---

4 **Repeat EM Iterations** Images (d), (e), and (f) show how the Gaussians are refined with more iterations:

- At $L = 2$: we see initial improvement,
- At $L = 5$: the contours are aligning with the true structure of the data,

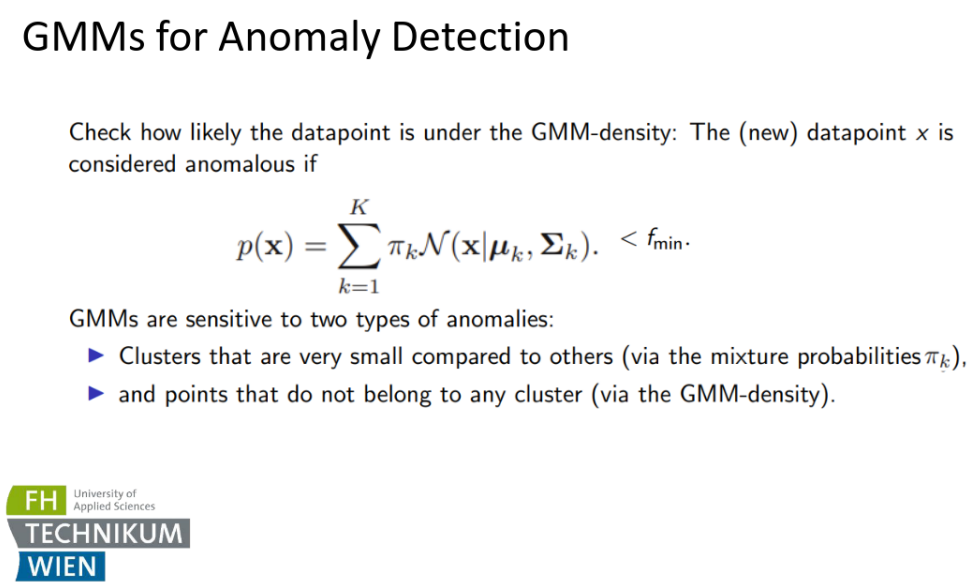
- At $L = 20$: the model has almost fully converged, with ellipses tightly matching the data clusters.

As you can see, the EM algorithm **iteratively alternates** between assigning points to clusters and improving the cluster definitions, until the model stabilizes.

This elegant dynamic — driven by **soft assignments** and **flexible covariance structures** — is what gives GMMs their power in modeling real-world data.

Next, we'll look at a practical use case: **anomaly detection with GMMs**."

Send the next slide whenever you're ready.



Excellent — here's the script for **slide 14**.

Slide 14: GMMs for Anomaly Detection

"In addition to clustering, **Gaussian Mixture Models** are also very effective for **anomaly detection**.

The idea is simple:

> Check how likely a data point is under the model's probability distribution.

For a new point $\mathbf{x}$, we compute its **total probability under the GMM**:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}(\mathbf{x} \mid \mu_k, \Sigma_k)$$

If this probability is **less than a threshold** $f_{\min}$, then the point is considered an **anomaly**.

GMMs detect two types of anomalies:

1. **Points that don't fit any cluster well** These lie far away from all Gaussians — resulting in low density under the model.
2. **Clusters that are too small** If a component has a very small weight π_k, even if a point fits its shape, it might be flagged because the cluster itself is rare.

This makes GMMs a **probabilistic and interpretable** approach to detecting unusual data points — far more nuanced than just using distance thresholds like in k-means.

Let's now see how GMMs can combine clustering **and** anomaly detection in one framework."

Send the next slide when you're ready.

GMMs for Anomaly Detection + Clustering

GMMs can easily be used as clustering models that are able to say: 'I do not know what this is!' This 'out-of-training-set usability' of GMMs is very useful, especially for sensitive applications.

- Step 1: If the GMM-density of the (new) datapoint is too low, do not assign it to a component.
- Step 2: Otherwise assign the new datapoint to a component using the responsibility. (One may also introduce an additional responsibility condition.)

FH University of Applied Sciences TECHNIKUM WIEN

Excellent — here is the script for **slide 15**.

Slide 15: GMMs for Anomaly Detection + Clustering

"One of the greatest strengths of Gaussian Mixture Models is their ability to **combine clustering and anomaly detection** in a unified framework.

Unlike rigid clustering methods, GMMs have what we can call an **'out-of-training-set awareness'**. That is, they can say:

> *'This point doesn't look like anything I've seen before.'*

This behavior is crucial for sensitive applications — for example, fraud detection or medical diagnostics.

The idea works in two steps:

Step 1: If the **GMM density** for a new point **x** is **too low**, then the model says:

> "This point is so unlikely under any of my components, I won't assign it to any known cluster." This flags the point as **anomalous**.

Step 2: If the density is acceptable, then we proceed as usual:

> Assign the point to the most likely component, based on its **responsibility** values.

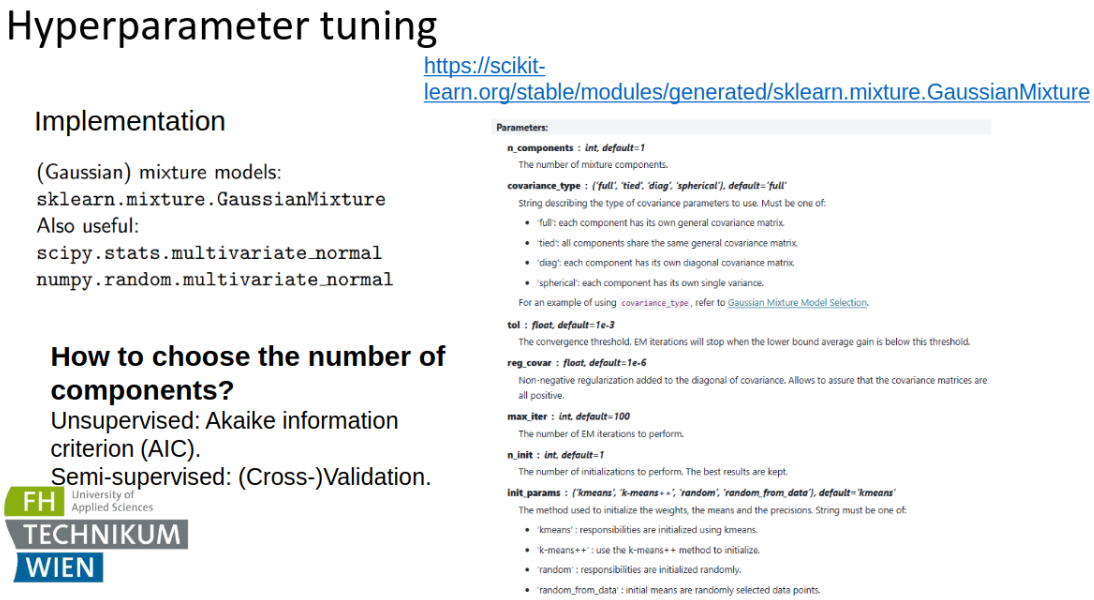
Optionally, we can add extra checks — for example, only assigning the point if its responsibility for one cluster **dominates**.

This makes GMMs ideal in cases where you want to:

- Cluster your data, but also…
- Recognize when something **doesn't fit any existing pattern**.

Let's now talk about how to **choose the number of Gaussians** — a key hyperparameter in GMMs."

Send the next slide whenever ready.



Perfect — here's the presentation script for **slide 16**.

Slide 16: Hyperparameter Tuning in GMMs

"Like many machine learning models, Gaussian Mixture Models have important **hyperparameters** that must be tuned for optimal performance.

Implementation

In practice, GMMs are implemented using:

- `sklearn.mixture.GaussianMixture` — the most widely used implementation in Python.
- You may also find utilities in:
  - `scipy.stats.multivariate_normal`
  - `numpy.random.multivariate_normal`

On the right, we see some key parameters from the `sklearn` documentation:

- `n_components`: the number of Gaussians.
- `covariance_type`: controls the flexibility of the shape (e.g., full vs. spherical).
- `tol`, `max_iter`: control the EM convergence behavior.
- `init_params`: allows different initialization methods like `kmeans` or `random`.

How do we choose the number of components?

This is one of the most critical choices. There are two typical strategies:

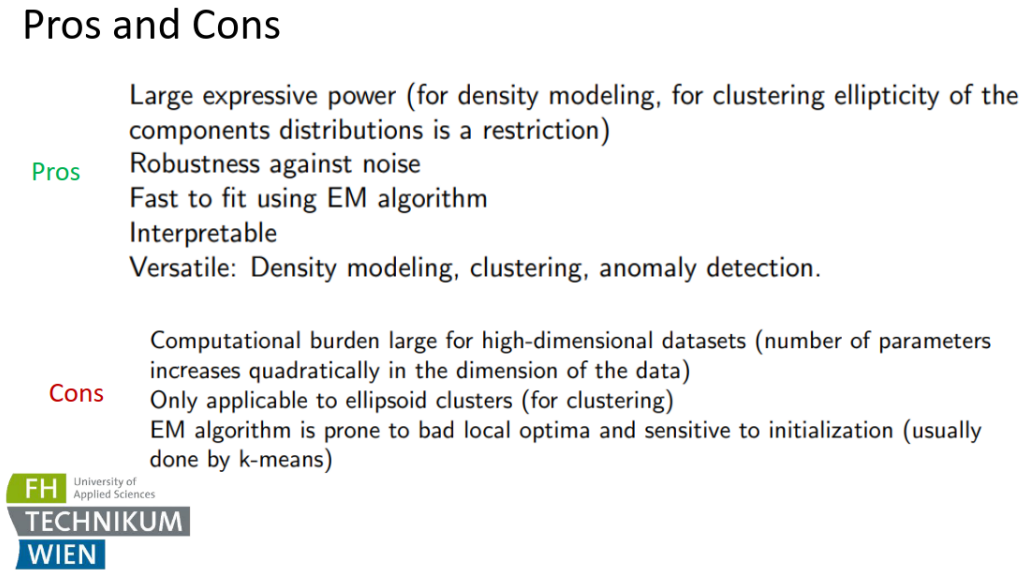
- **Unsupervised**: Use **AIC** (Akaike Information Criterion) or **BIC** (Bayesian Information Criterion) to balance model fit and complexity.
- **Semi-supervised**: Use **cross-validation** on a labeled subset to choose the value that yields the best generalization.

You try different values for `n_components`, compute the corresponding AIC or validation error, and pick the one that minimizes it.

This helps avoid both underfitting (too few components) and overfitting (too many components).

Let's now wrap up with a quick overview of the pros and cons of using GMMs."

Send the final slide whenever you're ready.



Perfect — here is the closing script for **slide 17**.

Slide 17: Pros and Cons of Gaussian Mixture Models

"As we wrap up, let's reflect on the main **strengths and limitations** of GMMs.

Pros:

- **Large expressive power**: GMMs can model complex, overlapping distributions — especially useful in density estimation.
- **Robustness to noise**: Thanks to soft probabilistic assignments, outliers don't drastically affect the model.
- **Efficient training**: The **EM algorithm** is relatively fast and well-studied.
- **Interpretability**: Each Gaussian has a clear meaning — with parameters for center, shape, and weight.
- **Versatility**: GMMs aren't just for clustering. They're also excellent for **density modeling** and **anomaly detection**.

Cons:

- **Scalability**: In high-dimensional data, the number of parameters in each covariance matrix grows **quadratically**, which can be computationally expensive.
- **Shape assumption**: GMMs work best when clusters are **ellipsoidal**. They may struggle with arbitrary shapes like concentric rings or spirals.
- **Local optima**: The **EM algorithm** can get stuck in poor solutions if initialization is bad — typically mitigated by initializing with **k-means**.

In conclusion, GMMs are a **powerful, flexible tool** in the machine learning toolbox. When used thoughtfully — with good initialization and dimensionality control — they provide a deep probabilistic insight into the structure of your data.

Thank you for your attention!"

Let me know if you'd like a compiled script of the entire presentation or help with visuals, speaker notes, or timing.