Probabilistic generative models

Introduction & motivation

Introduction and Motivation - What is a Generative Model?

A generative model for images

Generative models are class of machine learning models that are capable of generating new data points that resemble a given dataset. These models learn the underlying patterns and structures in the input data and use this knowledge to produce novel examples.

A generative model for text

Introduction and Motivation - What is a Generative Model?

$X =$  $Y = \text{"}cat\text{"}$

Formal definitions

- **Discriminative model**: learns $p(y|x)$

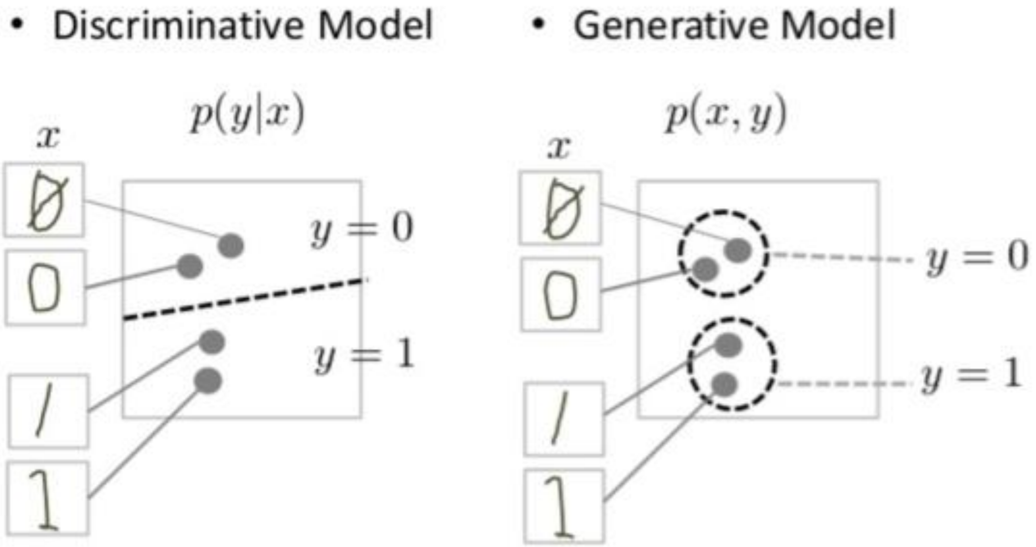
→ e.g., logistic regression, SVM.
Optimizes the decision boundary between classes.

- **Generative model**: learns $p(x, y) = p(y)\,p(x|y)$

or

$$p(x) = \int p(x, z)\, dz$$

→ allows **sampling** new data points.

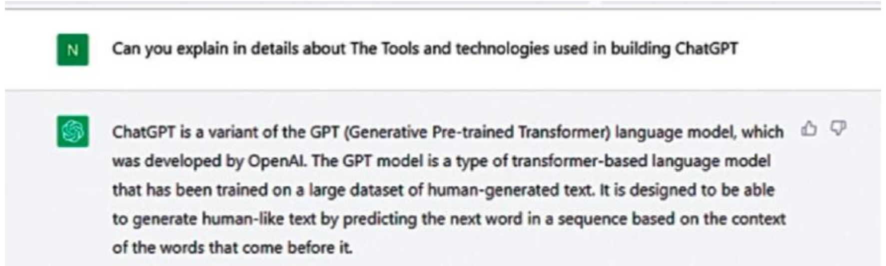


Why discriminative models cannot generate data

- A **discriminative models** can classify inputs but cannot **generate** new inputs.
- It does not model the distribution of x, only the boundary between classes.
- No way to sample a plausible x given a class or a latent vector.

Introduction and Motivation - Link with Domains

Domain	Typical Generative Use	Example
Machine Learning	Data augmentation, regularization	VAE, GAN, diffusion
Computer Vision	Image synthesis, inpainting, super-resolution	LDM, DiT, ControlNet
Natural Language	Text generation, translation, summarization	GPT, T5, LLaMA
Signal Processing	Denoising, source separation, synthesis	Generative filters, waveform models
Biology / Med / Sci	Protein folding, molecular generation, simulation	AlphaFold, diffusion over graphs



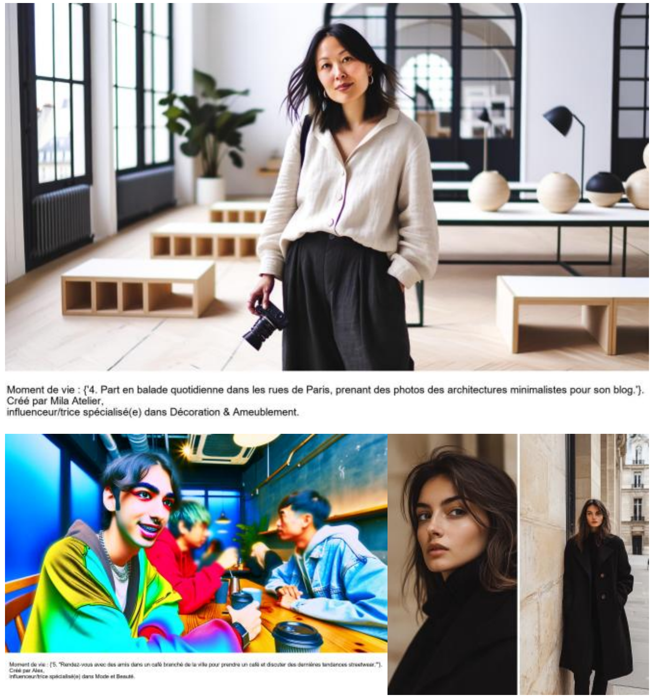
These models operate over structured or unstructured data, often conditioned on text, class, or latent variables.

Introduction and Motivation – 10 years of evolution

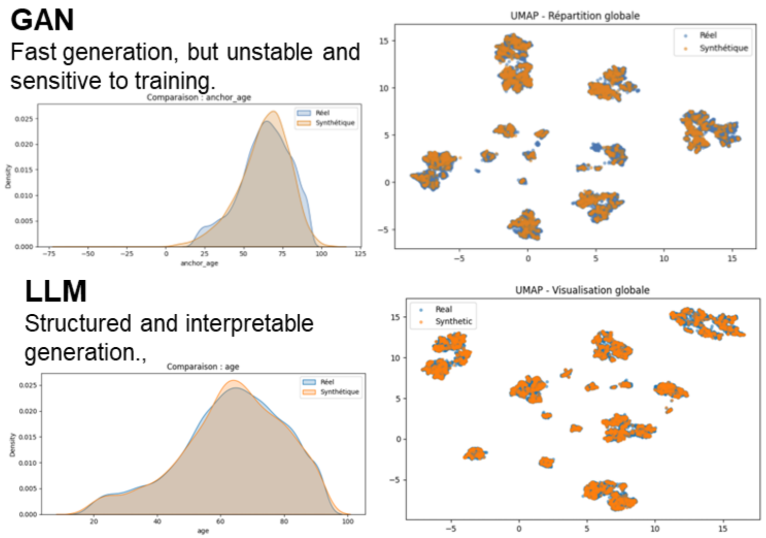
Introduction and Motivation – Three use-cases

Health news synthesis
Automatic generation of health-related news briefs from structured medical summaries.
Example models: GPT

Synthetic influencer images
Creation of realistic avatars placed in diverse lifestyle or commercial contexts (e.g. social media, fashion).
Example models: GPT, Stable Diffusion



Synthetic patient data
Simulation of tabular data mimicking cancer patients (e.g. head and neck cancer), for research and privacy-preserving analytics.
Example models: Adversarial LLM.



Introduction and Motivation – Conclusion of the introduction

- Generative models are powerful tools in the machine learning arsenal, capable of modeling complex distributions and generating new and realistic data.
- Their understanding and application are essential to tackle advanced problems in various domains ranging from image processing to text processing, to medical applications and beyond.

Probabilistic generative models

Probability and Statistics Review

Probability and Statistics Review – Random Variables

- Definition:
  - A random variable is a function that assigns a real number to each possible outcome of a random experiment.
  - Notation: X can represent a random variable.
- Examples of Discrete and Continuous Variables:
  - Discrete Variables:
    - Take a finite or countable number of values.
    - Examples: number of faces of a die (1, 2, 3, 4, 5, 6), success/failure (0, 1).
  - Continuous Variables:
    - Take an infinity of values in an interval.
    - Examples: blood pressure, temperature, height.
- Concepts of Mean, Variance, and Moments:
  - Mean (Expectation) μ:
    - Discrete
    - Continuous

$$E(X) = \sum_i x_i P(X = x_i) \qquad E(X) = \int_{-\infty}^{\infty} x f_X(x) dx$$

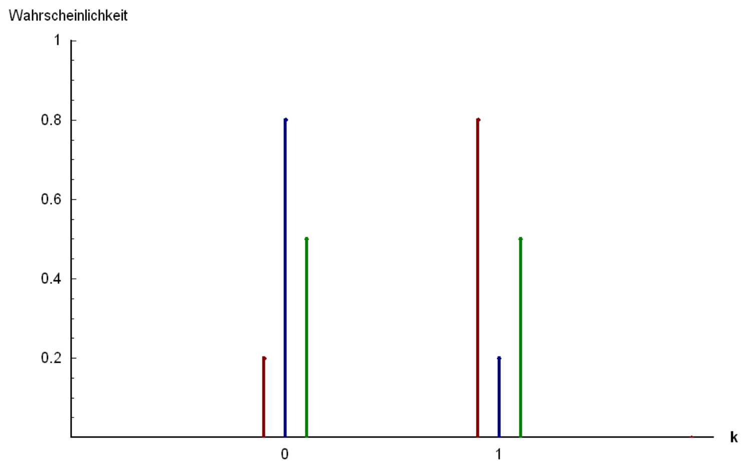
  - Variance σ^2 :
    - Measure of the dispersion of the values of the variable around the mean. $\mathrm{Var}(X) = E[(X - E(X))^2]$
  - Moments:
    - Centered and non-centered moments.
    - Examples: Moments of order 2 (variance), moments of order 3 (skewness), moments of order 4 (kurtosis).

Probability and Statistics Review – Probability Distributions

Discrete Probability Distributions:

- Bernoulli:
  - Models a single trial with two possible outcomes (success or failure).
  - Parameter *p* (probability of success).
  - Examples:

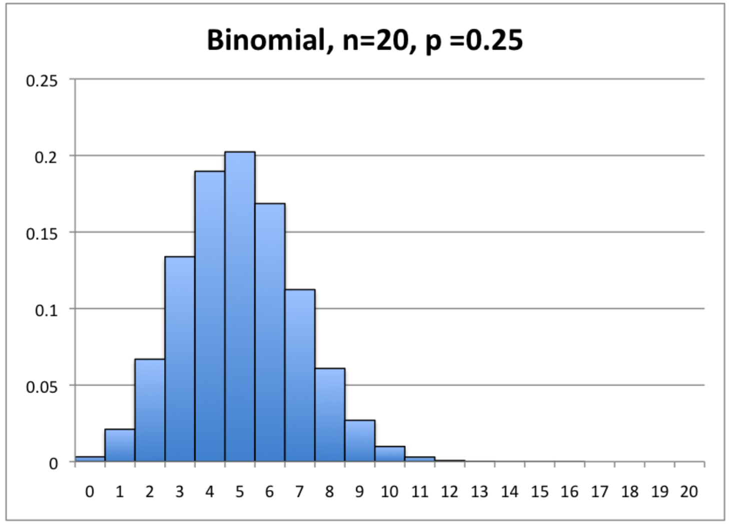
$$P(X=1)=p,, P(X=0)=1-p$$



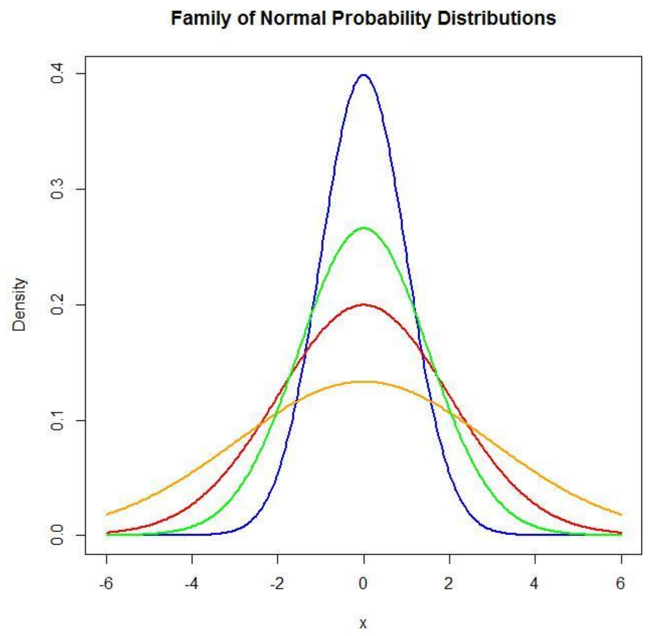
- Binomial:
  - Models the number of successes in *n* independent Bernoulli trials.

$$P(X=k)=\binom{n}{k}p^k(1-p)^{n-k}$$

  - Applications: number of successes in a series of draws.



Probability and Statistics Review – Probability Distributions

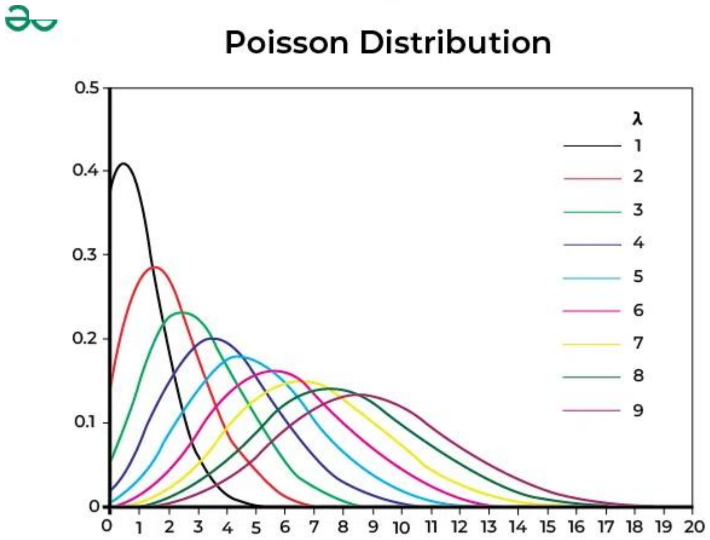


Continuous Probability Distributions:

- Normal (Gaussian):
  - Symmetric distribution, characterized by its mean μ and its variance σ^2.
  - Density function: $$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

.

  - Used in many statistical models because of the Central Limit Theorem.



- Exponential:
  - Models the time between events in a Poisson process.
  - Parameter λ (rate parameter).
  - Density function: $$f(x) = \lambda e^{-\lambda x} \text{ for } x \geq 0$$

Probability and Statistics Review – Conditional Distributions

- The conditional distribution of *X* given *Y* is the distribution of *X* when a specific value of *Y* is given.

$$P(X = x|Y = y)$$

- Discrete Variable:
  - If *X* is the grade of a student on an exam and *Y* is the number of hours studied, $P(X = x|Y = y)$
  can model the probability of different grades depending on the time spent studying.
- Continuous Variable:
  - In meteorology, one can model the distribution of the temperature *T* given the pressure *P*, $P(T = t|P = p)$
- Use in Probabilistic Graphical Models:
  - Conditional distributions facilitate the factorization of distributions in Bayesian networks and Markov graphical models.
  - Example: In a Bayesian network, each variable is conditioned on its parents, which simplifies the complexity of the joint distribution.

Types of Generative Models – Explicit vs Implicit Models

Explicit Models :

- Allow an explicit calculation of the probability of the data.
- Consist of precise specifications of probability distributions.
- Gaussian Mixtures (GMMs) :
  - Model the data as a combination of several Gaussian distributions.
  - Used for data clustering and density estimation.
  - EM algorithm (Expectation-Maximization) is commonly used to estimate the parameters.
- Probabilistic Graphical Models :
  - Use graphs to represent the dependencies between variables.
- Bayesian Networks :
  - Acyclic directed graphs where nodes represent variables and edges represent conditional dependence relations.
  - Example : Modeling complex systems such as medical diagnostics.
- Hidden Markov Fields (HMMs) :
  - Models for temporal sequences where a hidden state follows a Markov process.
  - Example : Activity recognition, speech transcription.

Types of Generative Models – Explicit vs Implicit Models

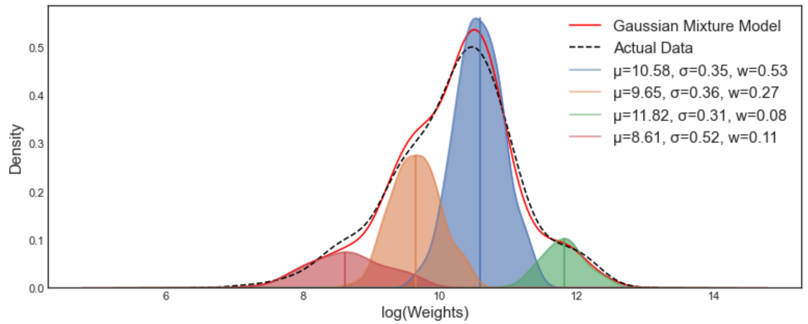
Implicit models :

- Generate data without explicitly specifying a probability function.
- Often use neural networks to implicitly learn the probability distribution.
- Generative Adversarial Networks (GANs) :
  - Consist of two neural networks (Generator and Discriminator) that train adversarially.
  - The generator produces synthetic data, and the discriminator evaluates their authenticity.
  - Applications : Generation of realistic images, video game design, artistic content creation.
- Models Based on Neural Networks :
  - Use neural architectures such as Autoencoders.
  - Variational Autoencoders (VAEs) :
    - Combination of neural networks and probabilistic methods to generate data.
    - Explicit probabilistic models, but the data generation is complex and indirect.

Types of Generative Models – Parametric vs Non-parametric Models

Parametric Models:

- The number of parameters is fixed, regardless of the size of the data.
- The structure of the model is defined in advance and does not change with the amount of available data.

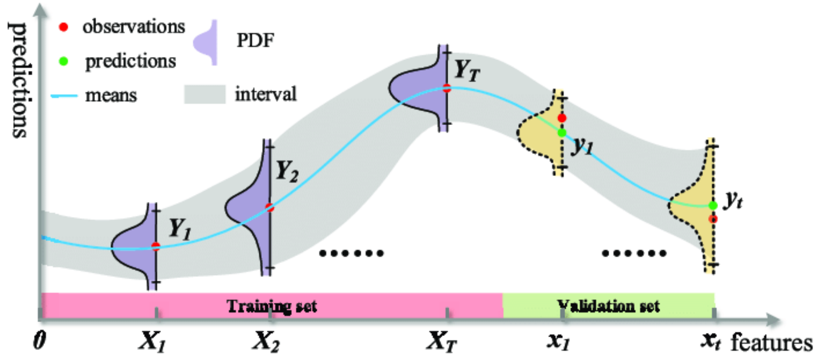


- **Gaussian Mixture Models (GMMs):**
    - Use a fixed number of Gaussian distributions (each distribution has a mean and a covariance that define their parameters).
    - Used in various fields for clustering and time series analysis.
- **Neural Networks:**
    - Parametric; the number of weights and biases is determined by the network's architecture (number of layers, number of neurons per layer).
    - Applicable to a wide variety of tasks such as image classification and voice recognition.

Types of Generative Models – Parametric vs Non-parametric Models

Non-Parametric Models:

- The number of parameters can increase with the amount of data.
- More flexible to adapt to the increasing complexity of data.



- **Gaussian Process Models:**
  - Use a distribution over functions to perform non-linear regression.
  - Capable of capturing complex relationships without explicitly specifying a functional form.
  - Applications: Time series prediction, survival analysis.
- **Kernel Methods:**
  - Kernels allow mapping data into higher-dimensional spaces to capture complex structures.
  - **Support Vector Machines (SVMs) with Kernels:** Builds hyperplanes in high-dimensional spaces to separate different classes.
  - **Kernel Ridge Regression:** A non-linear regression approach allowing flexible model fitting to the data.

Types of Generative Models – Conclusion

- Understanding the different types of generative models is crucial for choosing the appropriate methodology based on the data and target applications.

- Explicit models allow for direct probabilistic processing, whereas implicit models leverage the capacity of neural networks to model complex distributions.

- The choice between parametric and non-parametric depends on the needs in terms of flexibility and the amount of data available.

Probabilistic generative models

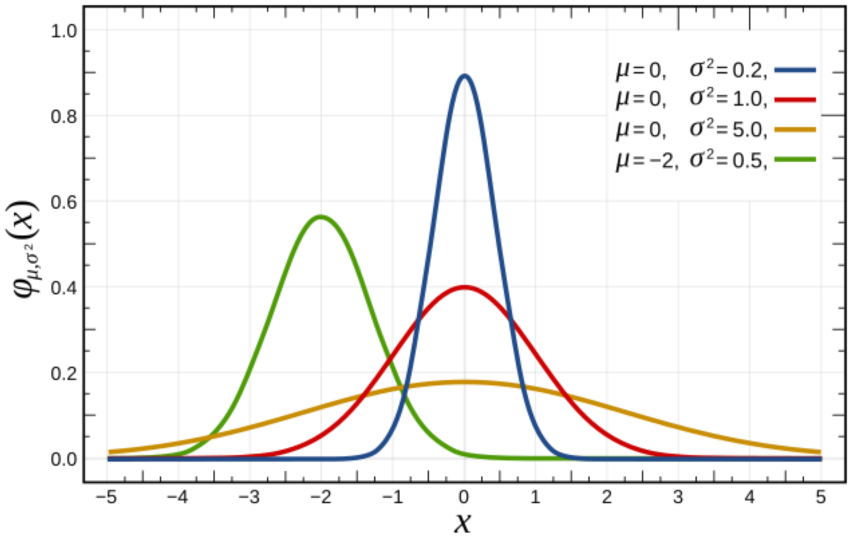
Introduction to Gaussian Mixture Models (GMMs)

Introduction to Gaussian Mixture Models (GMMs) – Gaussian Distribution

- **Form of the Density Function:**
- The ***Gaussian distribution*** or ***normal distribution*** is defined by its probability density function:

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

  - Where:
    - μ is the ***mean***.
    - σ^2 is the ***variance***.
    - σ is the ***standard deviation***.

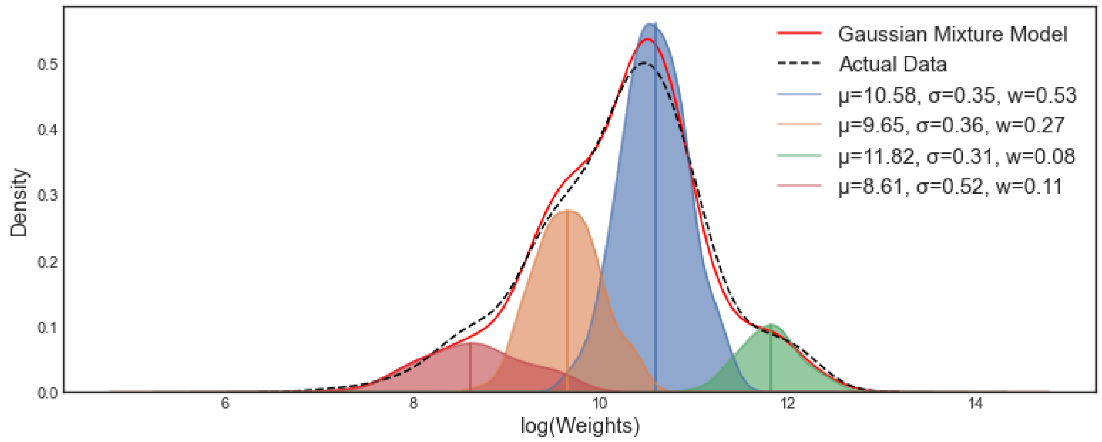


- **Symmetry Properties:**
  - The distribution is ***symmetric*** about its mean μ.
  - The values deviate symmetrically from the mean.
- **Mean and Variance:**
  - **Mean μ :**
    - Indicates the central point of the distribution. $\mu = E(X)$
  - **Variance σ^2 :**
    - Measures the dispersion of values around the mean. $\sigma^2 = E[(X-\mu)^2]$

Introduction to Gaussian Mixture Models (GMMs) – GMMs

- A ***Gaussian Mixture Model (GMM)*** models data as a combination of K multiple Gaussian distributions.

$$P(x) = \sum_{i=1}^{K} \pi_i \mathcal{N}(x|\mu_i, \sigma_i)$$



  - Where:
    - K is the ***number of components (or clusters)***.
    - π_i is the ***weight of the i-th component***.
    - $\mathcal{N}(x|\mu_i, \sigma_i)$ is the i-th Gaussian distribution with mean μ_i and covariance σ_i

- **Usage for Clustering and Density Estimation:**
  - **Clustering:** GMMs can identify subgroups or clusters in the data.
  - **Density Estimation:** Models the density of the data, useful in various applications (anomaly detection, interpolation).

Introduction to Gaussian Mixture Models (GMMs) – EM Algorithm (Expectation-Maximization)

E-Step (Expectation):

- Calculate the expectations for the latent variables using the current parameters of the model.
- Responsibility formula :

$$\gamma_{ik} = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i|\mu_j, \sigma_j)}$$

Where γ_{ik} is the responsibility of the k-th component for the i-th data point.

M-Step (Maximization):

- Maximize the likelihood to adjust the model parameters using the expected values of the latent variables.
- Parameter updates:
  - Means:
  - Covariances:
  - Weights:

$$\mu_k = \frac{\sum_{i=1}^{N} \gamma_{ik} x_i}{\sum_{i=1}^{N} \gamma_{ik}} \qquad \sigma_k = \frac{\sum_{i=1}^{N} \gamma_{ik}(x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^{N} \gamma_{ik}}$$

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

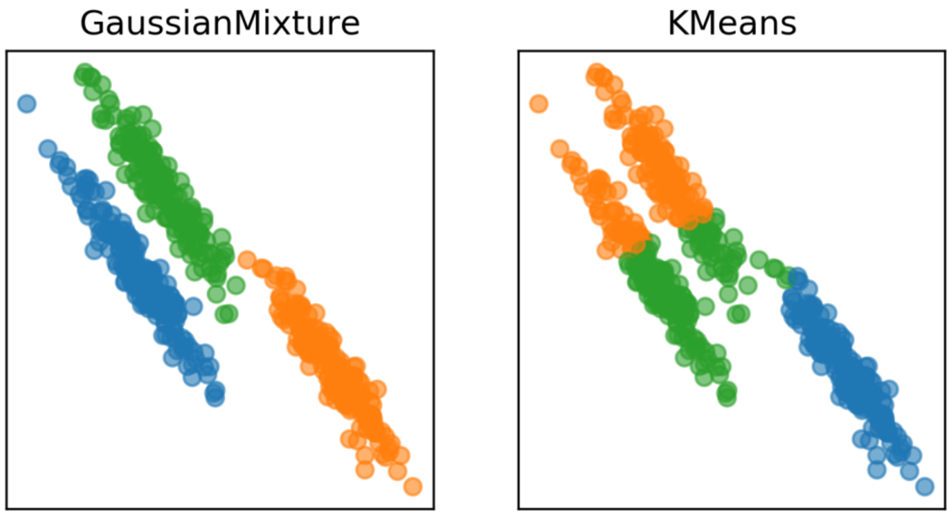
Iteration:

- Repeat the E and M steps until convergence, i.e., until the parameters cease to change significantly.

Introduction to Gaussian Mixture Models (GMMs) – EM Algorithm (Expectation-Maximization)

Practical Example, Clustering with GMMs:

- Apply GMMs to multivariate data, segmenting the data into groups or clusters based on Gaussian distributions.
- **Comparison with Other Clustering Techniques (like k-means):**
  - **K-means:** Assumes spherical clusters with equal dimensions and uses Euclidean distances.
  - **GMMs:** Models ellipsoidal clusters of varied sizes and orientations using Gaussian distributions.



Implementing the EM Algorithm for GMMs in Python– Installation and Configuration

Install Necessary Libraries:

```
pip install numpy scipy scikit-learn matplotlib
```

Implementation Steps:

- **Initialization of Parameters:** Initialize the weights, means, and covariances of the Gaussians.

```python
import numpy as np

def initialize_params(data, K):
        n, d = data.shape
        weights = np.ones(K) / K
        means = data[np.random.choice(n, K, False)]
        covariances = np.array([np.eye(d)] * K)
        return weights, means, covariances
```

Implementing the EM Algorithm for GMMs in Python– Installation and Configuration

- **E-Step (Expectation):** Calculate the responsibilities of the clusters for each data point.

```
def expectation_step(data, weights, means, covariances):
    n, d = data.shape
    K = len(weights)
    responsibilities = np.zeros((n, K))
    for k in range(K):
        resp = weights[k] * multivariate_normal.pdf(data, mean=means[k], cov=covariances[k])
        responsibilities[:, k] = resp
    responsibilities /= responsibilities.sum(1)[:, np.newaxis]
    return responsibilities
```

Implementing the EM Algorithm for GMMs in Python– Installation and Configuration

- **M-Step (Maximization):** Update the parameters (weights, means, covariances) by maximizing the likelihood.

```
def maximization_step(data, responsibilities):
    n, d = data.shape
    K = responsibilities.shape[1]
    Nk = responsibilities.sum(axis=0)
    weights = Nk / n
    means = np.dot(responsibilities.T, data) / Nk[:, np.newaxis]
    covariances = np.zeros((K, d, d))
    for k in range(K):
        centered_data = data - means[k]
        covariances[k] = np.dot(centered_data.T, centered_data * responsibilities[:, k][:, np.newaxis]) / Nk[k]
    return weights, means, covariances
```

Implementing the EM Algorithm for GMMs in Python– Installation and Configuration

- **Convergence:** Iterate the E and M steps until parameter convergence.

```
def compute_log_likelihood(data, weights, means, covariances):
    n, d = data.shape
    K = len(weights)
    log_likelihood = 0
    for i in range(n):
        prob = 0
        for k in range(K):
            prob += weights[k] * multivariate_normal.pdf(data[i], mean=means[k], cov=covariances[k])
        log_likelihood += np.log(prob)
    return log_likelihood

def em_algorithm(data, K, max_iter=100, tol=1e-4):
    weights, means, covariances = initialize_params(data, K)
    log_likelihoods = []
    for iteration in range(max_iter):
        responsibilities = expectation_step(data, weights, means, covariances)
        weights, means, covariances = maximization_step(data, responsibilities)
        log_likelihood = compute_log_likelihood(data, weights, means, covariances)
        log_likelihoods.append(log_likelihood)
        if len(log_likelihoods) > 1 and abs(log_likelihoods[-1] - log_likelihoods[-2]) < tol:
            break
    return weights, means, covariances, log_likelihoods
```

Implementing the EM Algorithm for GMMs in Python– Coding and Testing

```
from scipy.stats import multivariate_normal
import matplotlib.pyplot as plt

# Generate synthetic data
np.random.seed(0)
c = np.array([[0.0, -0.1], [1.7, 0.4]])
X = np.r_[np.dot(np.random.randn(200, 2), c), np.dot(np.random.randn(200, 2), c) + np.array([3, 3])]

# Apply the EM algorithm for GMMs
K = 2 # Number of clusters
weights, means, covariances, log_likelihoods = em_algorithm(X, K)

# Visualiser les résultats
plt.scatter(X[:, 0], X[:, 1], s=4, label='Data’)
for m, c in zip(means, covariances):
        plt.scatter(m[0], m[1], s=100, marker='X', label='Mean’)
        eigvals, eigvecs = np.linalg.eigh(c)
        order = eigvals.argsort()[::-1]
        eigvals, eigvecs = eigvals[order], eigvecs[:, order]
        angle = np.degrees(np.arctan2(*eigvecs[:, 0][::-1]))
        for j in range(1, 4):
                ell_radius_x = np.sqrt(eigvals[0]) * j
                ell_radius_y = np.sqrt(eigvals[1]) * j
                ell = Ellipse(xy=m, width=ell_radius_x * 2, height=ell_radius_y * 2, angle=angle, edgecolor='r', facecolor='none', lw=2, alpha=0.5/j)
                plt.gca().add_patch(ell)
plt.title('GMM Clustering’)
plt.legend()
plt.show()
```

Implementing the EM Algorithm for GMMs in Python– Analysis of Results

- **Visualizing Obtained Clusters:**
  - Examine the formed clusters and the fit of the Gaussian distributions.

- **Comparison with Other Clustering Methods:**
  - Compare the efficiency and accuracy of GMMs to techniques like k-means.
  - Advantages of GMMs in modeling ellipsoidal clusters of varied sizes and orientations.

Probabilistic generative models

disclosure & fairness

Disclosure & fairness – Why Ethics in Generative Models?

Risks

- Bias amplification (gender, race, socio-economic)
- Misinformation & deepfakes
- Hallucinations presented as facts

Data issues

- Copyright & licensing conflicts
- Privacy leaks (training on sensitive data)

Scientific & medical risks

- Over-trust in synthetic data
- Use in high-stakes decisions (health, law, finance)

Takeaway
Generative models are **powerful but not neutral** → they require **critical use**.

Disclosure & fairness – Regulation & GDPR + AI Act

GDPR (since 2018)
- Protects **personal data** in training datasets.
- Requires lawful basis for data processing (consent, legitimate interest, research).
- Right to **erasure** (the “right to be forgotten”) vs. difficulty with large-scale ML training.
- Generative models must avoid **re-identification** of individuals.

EU AI Act (in force Aug 2024)
- Applies to **General Purpose AI (GPAI)** models.
- **Transparency**: disclose training data categories.
- **Synthetic content**: must be **labeled** (watermark, disclaimer).
- **Risk classification**: medical/biometric = **high risk**.
- Timeline: full application expected **2025–2026**.

Takeaway

In projects: always **cite datasets**, **label generated content**, and **respect data protection principles**.

Compliance = technical + legal responsibility.

Disclosure & fairness – Ethics in Practice (for your project)

Good practices

- Always document datasets, prompts, seeds, checkpoints
- Respect licensing of datasets & models

Risks to avoid

- Presenting synthetic data as real without disclosure
- Using copyrighted material without attribution

Project requirement

Each project must include a short **‘Ethics & Compliance note”** covering:

1. Data provenance (real vs synthetic).
2. Rights & licenses
3. Known biases and risks
4. Disclosure of generated outputs