© JHU WSE Spring 2025

Statistical Algorithms

This document presents an in-depth exploration of statistical algorithms, with a focus on their application in data science and machine learning. It covers foundational methods such as the Expectation-Maximization (EM) algorithm, Bayesian classifiers, Naïve Bayes, and Mixture Models, including Gaussian, Poisson, and Exponential Mixture Models. The Expectation-Maximization algorithm is detailed with its mathematical formulation, step-by-step iterative process, and convergence criteria. Bayesian classifiers and Naïve Bayes are discussed as probabilistic classification techniques with applications in supervised learning. The study of Mixture Models highlights their role in clustering and density estimation, emphasizing parameter estimation via the EM algorithm. The document also explores variations of Gaussian Mixture Models (GMMs) and introduces modular implementations that extend to Poisson and Exponential Mixture Models. Through mathematical derivations, algorithmic implementations, and practical examples, this document serves as a comprehensive guide to statistical algorithms for data science applications.

This document is an extension of the research and lecture notes completed at Johns Hopkins University, Whiting School of Engineering, Engineering for Professionals, Artificial Intelligence Master's Program, Computer Science Master's Program, and Data Science Master's Program.

Contents

1 Introduction to Statistical Algorithms **1**

2 Expectation Maximization **2**
2.1 Mathematical Description of Expectation Maximization 2
2.2 Expectation-Maximization (EM) Algorithm 3
2.3 Example 5
2.4 Summary - Expectation Maximization 10

3 Bayes Classifier **12**
3.1 Mathematical Description of Bayes Classifier 12
3.2 Training Algorithm 13
3.3 Testing Algorithm for Classification 13
3.4 Summary - Bayes Classifier 15

4 Naive Bayes **16**
4.1 Naive Bayes Model as a Classifier 16
4.2 Naive Bayes Algorithm 17
4.3 Summary - Naive Bayes 19

5 Mixture Models **20**
5.1 Defining a Gaussian Mixture Models (GMMs) 20
5.2 Types of GMMs 20
5.3 Poisson Mixture Model (PMM) 23
5.4 Parameter Estimation using the Expectation-Maximization (EM) Algorithm 23
5.5 Exponential Mixture Model (EMM) 25
5.6 Expectation-Maximization (EM) Algorithm for Mixture Models 27
5.7 Summary - Mixture Models 30

6 Summary **31**

1 Introduction to Statistical Algorithms

Statistical algorithms form the foundation of many machine learning and data science applications, providing robust methods for probabilistic modeling, classification, clustering, and data estimation. These algorithms leverage principles from probability theory and statistical inference to derive meaningful insights from data and enable decision-making under uncertainty.

This document explores key statistical algorithms, including the Expectation-Maximization (EM) algorithm, Bayesian classifiers, Naïve Bayes, and Mixture Models such as Gaussian, Poisson, and Exponential Mixture Models. The EM algorithm serves as a fundamental optimization technique for estimating model parameters in probabilistic frameworks, particularly in cases where data contain latent variables. Bayesian classifiers, including Naïve Bayes, apply Bayes' theorem to perform classification tasks, making assumptions about feature independence to enhance computational efficiency. Mixture models extend probabilistic modeling by representing data distributions as combinations of multiple probability distributions, each capturing different components of the data.

The document systematically presents the mathematical formulations, algorithmic implementations, and practical applications of these statistical techniques. The modular structure of the algorithms allows for their extension to various domains, such as natural language processing, image recognition, and predictive analytics. By integrating theoretical foundations with computational strategies, this document aims to provide a comprehensive guide to statistical algorithms, emphasizing their significance in modern data-driven applications.

2 Expectation Maximization

The Expectation-Maximization (EM) algorithm is an iterative method for finding maximum likelihood estimates of parameters in probabilistic models, particularly when the data has missing or unobserved components [2]. The algorithm is widely used in machine learning, statistics, and data science, with applications in clustering (e.g., Gaussian Mixture Models), hidden variable modeling (e.g., Hidden Markov Models), and missing data problems.

The core idea behind the EM algorithm is to iteratively estimate missing data (or latent variables) and use these estimates to refine model parameters. It consists of two main steps:

- **Expectation Step (E-Step)**: Computes the expected value of the missing (latent) data using the current estimates of the model parameters.
- **Maximization Step (M-Step)**: Updates the model parameters by maximizing the expected log-likelihood function computed in the E-Step.

The algorithm alternates between these two steps until a convergence criterion is met, typically when the change in parameters falls below a predefined threshold ϵ. The EM algorithm ensures that the likelihood function does not decrease in each iteration, making it a reliable approach for parameter estimation. However, it converges to a local rather than a global optimum, making parameter initialization crucial for its success.

2.1 Mathematical Description of Expectation Maximization

The EM algorithm consists of choosing initial parameters for the means, $\mu_k^{(j)}$, standard deviations, $\sigma_k^{(j)}$, and mixing probabilities, $p_k^{(j)}$, for a user defined number of clusters, K, then performing the E-Step and M-Step successively until convergence, where j is the current iteration and n is the number of samples/observations. The convergence criteria is determined by examining when the parameters quit changing, i.e., when $\left|\mu_k^{(j)} - \mu_k^{(j+1)}\right| < \epsilon$ & $\left|\sigma_k^{(j)} - \sigma_k^{(j+1)}\right| < \epsilon$ & $\left|p^{(j)}(k|n) - p^{(j+1)}(k|n)\right| < \epsilon$ for some epsilon (ϵ) and distance calculation (Euclidian distance). The maximum likelihood estimation is a method of estimating the parameters of the distributions based upon the observed data.

The expectation step (E-Step) calculates the membership probabilities, $p(k|n)$ [4]. The mixing probabilities p_k are viewed as the sample mean of the membership probabilities $p(k|n)$ assuming a uniform distribution over all the data points. The Gaussian function, $g(\mathbf{x}_n; \mu_k^{(j)}, \sigma_k^{(j)})$, is used to compute mixture of Gaussian functions as shown in the denominator of $p(k|n)$.

$$p^{(j)}(k|n) = \frac{p_k^{(j)} g(\mathbf{x}_n; \mu_k^{(j)}, \sigma_k^{(j)})}{\sum_{k=1}^{K} p_k^{(j)} g(\mathbf{x}_n; \mu_k^{(j)}, \sigma_k^{(j)})} \tag{1}$$

$$g(\mathbf{x}_n; \mu_k^{(j)}, \sigma_k^{(j)}) = \frac{1}{\left(\sqrt{2\pi}\sigma_k\right)^D} \exp\left\{ -\frac{1}{2}\left(\frac{\|\mathbf{x}_n - \mu_k\|}{\sigma_k}\right)^2 \right\} \tag{2}$$

Note that $\|\mathbf{x}_n - \mu_k\|$ is the vector norm in which the distance between the observation vector and the cluster mean vector is calculated. To account for the individual values of the vectors $\mathbf{x}_n$ and μ_k the vectors are temporarily written as $\mathbf{x}_{n_d}$ and μ_{k_d} in the following equation to account for the dimension of the vectors:

$$\|\mathbf{x}_{n_d} - \mu_{k_d}\| = \sqrt{\sum_{d=1}^{D} (\mathbf{x}_{n_d} - \mu_{k_d})^2} = \sqrt{(\mathbf{x}_{n_1} - \mu_{k_1})^2 + (\mathbf{x}_{n_2} - \mu_{k_2})^2 + \cdots + (\mathbf{x}_{n_D} - \mu_{k_D})^2} \tag{3}$$

The maximization step (M-Step) uses the data from the expectation step as if it were measured data to determine the maximum likelihood estimate of the parameter [4]. This estimated data is often referred to as the imputed data. This step is dependent upon the membership probabilities $p(k|n)$ which are computed in the E-Step. The EM algorithm consists of iterating the mean, standard deviation, and mixing probabilities until convergence. The mixing probabilities are the sample mean of the conditional probabilities $p(k|n)$ assuming a uniform distribution over all the data points.

$$\mu_k^{(j+1)} = \frac{\sum_{n=1}^{N} p^j(k|n)\mathbf{x}_n}{\sum_{n=1}^{N} p^j(k|n)} \tag{4}$$

$$\sigma_k^{(j+1)} = \sqrt{\frac{1}{D} \frac{\sum_{n=1}^{N} p^j(k|n) \left\| \mathbf{x}_n - \mu_k^{(j+1)} \right\|^2}{\sum_{n=1}^{N} p^j(k|n)}} \tag{5}$$

$$p_k^{(j+1)} = \frac{1}{N} \sum_{n=1}^{N} p^j(k|n) \tag{6}$$

2.2 Expectation-Maximization (EM) Algorithm

The EM algorithm consists of the following steps:

- **Step 1:** Initialization
  Initialize the parameters $\boldsymbol{\theta}^{(0)}$, which may include class means μ_k, covariances Σ_k, and mixing coefficients p_k in the case of a Gaussian Mixture Model (GMM).
- **Step 2:** Expectation Step (E-Step)
  Given the current estimates of the parameters $\boldsymbol{\theta}^{(j)}$, compute the expected value of the complete-data log-likelihood function. This involves computing the posterior probabilities (responsibilities) of the latent variables:

Algorithm 1 Expectation-Maximization (EM) Algorithm for Gaussian Mixture Model

Input: Observed data $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T$, number of clusters K, convergence threshold ϵ
Output: Estimated parameters μ_k, σ_k, p_k for each cluster k
function ExpectationMaximization($\mathbf{X}, K, \epsilon$)
Initialize: Means $\mu_k^{(0)}$, standard deviations $\sigma_k^{(0)}$, and mixing probabilities $p_k^{(0)}$ randomly
Set iteration counter $j \leftarrow 0$
repeat
E-Step: Compute membership probabilities $p^{(j)}(k|n)$ for each cluster k and each sample $\mathbf{x}_n$:

$$p^{(j)}(k|n) = \frac{p_k^{(j)} g(\mathbf{x}_n; \mu_k^{(j)}, \sigma_k^{(j)})}{\sum_{k=1}^{K} p_k^{(j)} g(\mathbf{x}_n; \mu_k^{(j)}, \sigma_k^{(j)})}$$

where the Gaussian function is given by:

$$g(\mathbf{x}_n; \mu_k^{(j)}, \sigma_k^{(j)}) = \frac{1}{\left(\sqrt{2\pi}\sigma_k\right)^D} \exp\left\{ -\frac{1}{2}\left(\frac{\|\mathbf{x}_n - \mu_k^{(j)}\|}{\sigma_k^{(j)}} \right)^2 \right\}$$

M-Step: Update parameters using membership probabilities:
Compute updated means:

$$\mu_k^{(j+1)} = \frac{\sum_{n=1}^{N} p^j(k|n)\mathbf{x}_n}{\sum_{n=1}^{N} p^j(k|n)}$$

Compute updated standard deviations:

$$\sigma_k^{(j+1)} = \sqrt{\frac{1}{D} \frac{\sum_{n=1}^{N} p^j(k|n) \left\| \mathbf{x}_n - \mu_k^{(j+1)} \right\|^2}{\sum_{n=1}^{N} p^j(k|n)}}$$

Compute updated mixing probabilities:

$$p_k^{(j+1)} = \frac{1}{N} \sum_{n=1}^{N} p^j(k|n)$$

Check for Convergence:
if $\left|\mu_k^{(j+1)} - \mu_k^{(j)}\right| < \epsilon$ **and** $\left|\sigma_k^{(j+1)} - \sigma_k^{(j)}\right| < \epsilon$ **and** $\left|p_k^{(j+1)} - p_k^{(j)}\right| < \epsilon$ for all k **then**
Break
end if
Set $j \leftarrow j + 1$
until convergence criteria are met
return μ_k, σ_k, p_k for all k
end function

$$p^{(j)}(k|n) = \frac{p_k^{(j)} g(\mathbf{x}_n; \mu_k^{(j)}, \sigma_k^{(j)})}{\sum_{k=1}^{K} p_k^{(j)} g(\mathbf{x}_n; \mu_k^{(j)}, \sigma_k^{(j)})} \tag{7}$$

where $g(\mathbf{x}_n; \mu_k^{(j)}, \Sigma_k^{(j)})$ represents the Gaussian probability density function:

$$g(\mathbf{x}_n; \mu_k^{(j)}, \Sigma_k^{(j)}) = \frac{1}{\left(\sqrt{2\pi}\sigma_k\right)^D} \exp\left\{ -\frac{1}{2}\left(\frac{\|\mathbf{x}_n - \mu_k^{(j)}\|}{\sigma_k^{(j)}}\right)^2 \right\} \tag{8}$$

- **Step 3:** Maximization Step (M-Step)
  Using the responsibilities computed in the E-step, update the parameters by maximizing the expected complete-data log-likelihood:

$$\mu_k^{(j+1)} = \frac{\sum_{n=1}^{N} p^j(k|n)\mathbf{x}_n}{\sum_{n=1}^{N} p^j(k|n)} \tag{9}$$

$$\sigma_k^{(j+1)} = \sqrt{\frac{1}{D}\frac{\sum_{n=1}^{N} p^j(k|n)\left\|\mathbf{x}_n - \mu_k^{(j+1)}\right\|^2}{\sum_{n=1}^{N} p^j(k|n)}} \tag{10}$$

$$p_k^{(j+1)} = \frac{1}{N}\sum_{n=1}^{N} p^j(k|n) \tag{11}$$

- **Step 4:** Convergence Check
  The algorithm iterates between the E-step and the M-step until convergence is achieved. Convergence is typically defined by changes in parameters below a certain threshold ϵ:

$$\left|\mu_k^{(j+1)} - \mu_k^{(j)}\right| < \epsilon, \quad \left|\sigma_k^{(j+1)} - \sigma_k^{(j)}\right| < \epsilon, \quad \left|p_k^{(j+1)} - p_k^{(j)}\right| < \epsilon \tag{12}$$

2.3 Example

Given a matrix and generated mean and standard deviation the following variables are assigned:

$$\mathbf{x} = \begin{bmatrix} 1 & 2 \\ 4 & 2 \\ 1 & 3 \\ 4 & 3 \end{bmatrix}, \bar{x}_{col} = \begin{bmatrix} 2.5 & 2.5 \end{bmatrix}^T, \sigma_{col} = \begin{bmatrix} 1.7321 & 0.57735 \end{bmatrix}^T$$

The variable $K \equiv$ number of sub populations. This is a priori knowledge the use will need to know or guess. Now a set of initial parameters are generated to ensure convergence is reached for each

subpopulation. Notice that the column standard deviation is multiplied by random numbers from -1 to 1 which is used to calculate the initial mean for the subpopulations. For the standard deviation the average is used for simplicity. The membership probabilities are derived by the number of desired sub populations.

$$\mu = \bar{x}_{col}\begin{bmatrix}1 & 1\end{bmatrix} + \sigma_{col}\begin{bmatrix}randn(1,K)\end{bmatrix} = \begin{bmatrix}2.5\\2.5\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix} + \begin{bmatrix}1.7321\\0.57735\end{bmatrix}\begin{bmatrix}-0.1867 & 0.7257\end{bmatrix} = \begin{bmatrix}2.1766 & 3.7571\\2.3922 & 2.9190\end{bmatrix}$$

$$\sigma = \sigma_{col}\begin{bmatrix}0.5 & 0.5\\0.5 & 0.5\end{bmatrix} = \begin{bmatrix}1.7321 & 0.57735\end{bmatrix}\begin{bmatrix}0.5 & 0.5\\0.5 & 0.5\end{bmatrix} = \begin{bmatrix}1.1547 & 1.1547\end{bmatrix}$$

$$p_k = \frac{\begin{bmatrix}1 & 1\end{bmatrix}}{K} = \begin{bmatrix}\frac{1}{2} & \frac{1}{2}\end{bmatrix}$$

The Expectation Step (E-Step) $p^{(j)}(k|n)$ is the conditioning argument (n) to the expectation (k) and is regarded as fixed. Computes the expected value of the x_n data using the current estimation of the parameter and the observed data. This step uses the prior probabilities of the sub population centroids, $p_k^{(j)}$. The Gaussian function, $g()$, is used to compute mixture of Gaussian functions as shown in the denominator of $p^{(j)}(k|l)$.

$$p^{(j)}(k|n) = \frac{p_k^{(j)} g(\mathbf{x}_n; \mu_k^{(j)}, \sigma_k^{(j)})}{\sum_{k=1}^{K} p_k^{(j)} g(\mathbf{x}_n; \mu_k^{(j)}, \sigma_k^{(j)})} \tag{13}$$

Now we rewrite Equation 14 for the individual samples/observations as follows:

$$g(\mathbf{x}_n; \mu_k^{(j)}, \sigma_k^{(j)}) = \frac{1}{\left(\sqrt{2\pi}\sigma_k\right)^D} \exp\left\{ -\frac{1}{2}\left(\frac{\|\mathbf{x}_n - \mu_k\|}{\sigma_k}\right)^2\right\} \tag{14}$$

Now lets add some numbers to the equations based on the give matrix $\mathbf{x}$ and initial conditions set. For $n = k = j = 1$,

$g(x_1; \mu_1^{(1)}, \sigma_1^{(1)}) = \frac{1}{(\sqrt{2\pi}1.1547)^2} \exp\left\{ -\frac{1}{2}\left(\frac{\|\begin{bmatrix}1 & 2\end{bmatrix} - \begin{bmatrix}2.1766 & 2.3922\end{bmatrix}\|}{1.1547}\right)^2\right\} = \frac{1}{8.3776} \exp\left\{ -\frac{1}{2}\left(\frac{1.5382}{1.3333}\right)\right\} = 0.0670$

$p_i^{(1)} g(x_1; \mu_1^{(1)}, \sigma_1^{(1)}) = (0.5)(0.0670) = 0.0335$

For $n = 2, k = j = 1$,

$g(x_2; \mu_1^{(1)}, \sigma_1^{(1)}) = \frac{1}{(\sqrt{2\pi}1.1547)^2} \exp\left\{ -\frac{1}{2}\left(\frac{\|\begin{bmatrix}4 & 2\end{bmatrix} - \begin{bmatrix}2.1766 & 2.3922\end{bmatrix}\|}{1.1547}\right)^2\right\} = \frac{1}{8.3776} \exp\left\{ -\frac{1}{2}\left(\frac{3.4786}{1.3333}\right)\right\} = 0.0324$

$p_i^{(1)} g(x_2; \mu_1^{(1)}, \sigma_1^{(1)}) = (0.5)(0.0324) = 0.0162$

For $n = 3, k = j = 1$,

$g(x_3; \mu_1^{(1)}, \sigma_1^{(1)}) = \frac{1}{(\sqrt{2\pi}1.1547)^2} \exp\left\{-\frac{1}{2}\left(\frac{\|\begin{bmatrix}1 & 3\end{bmatrix} - \begin{bmatrix}2.1766 & 2.3922\end{bmatrix}\|}{1.1547}\right)^2\right\} = \frac{1}{8.3776} \exp\left\{-\frac{1}{2}\left(\frac{1.7538}{1.3333}\right)\right\} = 0.0618$

$p_i^{(1)} g(x_3; \mu_1^{(1)}, \sigma_1^{(1)}) = (0.5)(0.0618) = 0.0309$

For $n = 4, k = j = 1$,

$g(x_4; \mu_1^{(1)}, \sigma_1^{(1)}) = \frac{1}{(\sqrt{2\pi}1.1547)^2} \exp\left\{-\frac{1}{2}\left(\frac{\|\begin{bmatrix}4 & 3\end{bmatrix} - \begin{bmatrix}2.1766 & 2.3922\end{bmatrix}\|}{1.1547}\right)^2\right\} = \frac{1}{8.3776} \exp\left\{-\frac{1}{2}\left(\frac{3.6942}{1.3333}\right)\right\} = 0.0299$

$p_i^{(1)} g(x_4; \mu_1^{(1)}, \sigma_1^{(1)}) = (0.5)(0.0299) = 0.0149$

For $n = 1, k = 2, j = 1$,

$g(x_1; \mu_2^{(1)}, \sigma_2^{(1)}) = \frac{1}{(\sqrt{2\pi}1.1547)^2} \exp\left\{-\frac{1}{2}\left(\frac{\|\begin{bmatrix}1 & 2\end{bmatrix} - \begin{bmatrix}3.7571 & 2.9190\end{bmatrix}\|}{1.1547}\right)^2\right\} = \frac{1}{8.3776} \exp\left\{-\frac{1}{2}\left(\frac{8.4463}{1.3333}\right)\right\} = 0.0050$

$p_i^{(1)} g(x_1; \mu_2^{(1)}, \sigma_2^{(1)}) = (0.5)(0.0050) = 0.0025$

For $n = 2, k = 2, j = 1$,

$g(x_2; \mu_2^{(1)}, \sigma_2^{(1)}) = \frac{1}{(\sqrt{2\pi}1.1547)^2} \exp\left\{-\frac{1}{2}\left(\frac{\|\begin{bmatrix}4 & 2\end{bmatrix} - \begin{bmatrix}3.7571 & 2.9190\end{bmatrix}\|}{1.1547}\right)^2\right\} = \frac{1}{8.3776} \exp\left\{-\frac{1}{2}\left(\frac{0.9036}{1.3333}\right)\right\} = 0.0851$

$p_i^{(1)} g(x_2; \mu_2^{(1)}, \sigma_2^{(1)}) = (0.5)(0.0851) = 0.0425$

For $n = 3, k = 2, j = 1$,

$g(x_3; \mu_2^{(1)}, \sigma_2^{(1)}) = \frac{1}{(\sqrt{2\pi}1.1547)^2} \exp\left\{-\frac{1}{2}\left(\frac{\|\begin{bmatrix}1 & 3\end{bmatrix} - \begin{bmatrix}3.7571 & 2.9190\end{bmatrix}\|}{1.1547}\right)^2\right\} = \frac{1}{8.3776} \exp\left\{-\frac{1}{2}\left(\frac{7.6082}{1.3333}\right)\right\} = 0.0069$

$p_i^{(1)} g(x_3; \mu_2^{(1)}, \sigma_2^{(1)}) = (0.5)(0.0069) = 0.0034$

For $n = 4, k = 2, j = 1,$

$g(x_4; \mu_2^{(1)}, \sigma_2^{(1)}) = \frac{1}{(\sqrt{2\pi}1.1547)^2} \exp\left\{-\frac{1}{2}\left(\frac{\|\begin{bmatrix}4 & 3\end{bmatrix} - \begin{bmatrix}3.7571 & 2.9190\end{bmatrix}\|}{1.1547}\right)^2\right\} = \frac{1}{8.3776} \exp\left\{-\frac{1}{2}\left(\frac{0.0656}{1.3333}\right)\right\} = 0.1165$

$p_i^{(1)} g(x_4; \mu_2^{(1)}, \sigma_2^{(1)}) = (0.5)(0.1165) = 0.0582$

Now we calculate the membership probabilities of $p^{(1)}(k|n)$ as follows:

$$p^{(1)}(k|n) = \frac{p_k^{(1)} g(\mathbf{x}; \mu_k^{(1)}, \sigma_k^{(1)})}{\sum_{k=1}^{K} p_k^{(1)} g(\mathbf{x}; \mu_k^{(1)}, \sigma_k^{(1)})}$$

For $n = 1, k = 1, j = 1,$

$$p^{(1)}(1|1) = \frac{p_1^{(1)} g(x_1; \mu_1^{(1)}, \sigma_1^{(1)})}{\sum_{k=1}^{K} p_k^{(1)} g(x_1; \mu_k^{(1)}, \sigma_k^{(1)})} = \frac{0.0335}{0.0360} = 0.9302$$

For $n = 2, k = 1, j = 1,$

$$p^{(1)}(1|2) = \frac{p_1^{(1)} g(x_2; \mu_1^{(1)}, \sigma_1^{(1)})}{\sum_{k=1}^{K} p_k^{(1)} g(x_2; \mu_k^{(1)}, \sigma_k^{(1)})} = \frac{0.0162}{0.0587} = 0.2758$$

For $n = 3, k = 1, j = 1,$

$$p^{(1)}(1|3) = \frac{p_1^{(1)} g(x_3; \mu_1^{(1)}, \sigma_1^{(1)})}{\sum_{k=1}^{K} p_k^{(1)} g(x_3; \mu_k^{(1)}, \sigma_k^{(1)})} = \frac{0.0309}{0.0344} = 0.8998$$

For $n = 4, k = 1, j = 1,$

$$p^{(1)}(1|4) = \frac{p_1^{(1)} g(x_4; \mu_1^{(1)}, \sigma_1^{(1)})}{\sum_{k=1}^{K} p_k^{(1)} g(x_4; \mu_k^{(1)}, \sigma_k^{(1)})} = \frac{0.0149}{0.0732} = 0.2041$$

For $n = 1, k = 2, j = 1,$

$$p^{(1)}(2|1) = \frac{p_2^{(1)} g(x_1; \mu_2^{(1)}, \sigma_2^{(1)})}{\sum_{k=1}^{K} p_k^{(1)} g(x_1; \mu_k^{(1)}, \sigma_k^{(1)})} = \frac{0.0025}{0.0360} = 0.0693$$

For $n = 2, k = 2, j = 1$,

$$p^{(1)}(2|2) = \frac{p_2^{(1)} g(x_2; \mu_2^{(1)}, \sigma_2^{(1)})}{\sum_{k=1}^{K} p_k^{(1)} g(x_2; \mu_k^{(1)}, \sigma_k^{(1)})} = \frac{0.0425}{0.0587} = 0.7242$$

For $n = 3, k = 2, j = 1$,

$$p^{(1)}(2|3) = \frac{p_2^{(1)} g(x_3; \mu_2^{(1)}, \sigma_2^{(1)})}{\sum_{k=1}^{K} p_k^{(1)} g(x_3; \mu_k^{(1)}, \sigma_k^{(1)})} = \frac{0.0034}{0.0344} = 0.1002$$

For $n = 4, k = 2, j = 1$,

$$p^{(1)}(2|4) = \frac{p_2^{(1)} g(x_4; \mu_2^{(1)}, \sigma_2^{(1)})}{\sum_{k=1}^{K} p_k^{(1)} g(x_4; \mu_k^{(1)}, \sigma_k^{(1)})} = \frac{0.0582}{0.0732} = 0.7959$$

Now we can calculate the following sum of mixture probabilities for each k as follows:

For $N = 4, k = 2$ and $j = 1$,

$$\sum_{n=1}^{4} p^1(1|n) = 0.9302 + 0.2758 + 0.8998 + 0.2041 = 2.3100 \tag{15}$$

For $N = 4, k = 2$ and $j = 1$,

$$\sum_{n=1}^{4} p^1(2|n) = 0.0698 + 0.7242 + 0.1002 + 0.7959 = 1.6900 \tag{16}$$

Now we can calculate the M Step values as follows:

For $N = 4, k = 1$ and $j = 1$

$$\mu_1^{(2)} = \frac{\sum_{n=1}^{4} p^1(1|n)\mathbf{x}_n}{\sum_{n=1}^{4} p^1(1|n)} = \frac{0.9302 \begin{bmatrix} 1 & 2 \end{bmatrix} + 0.2758 \begin{bmatrix} 4 & 2 \end{bmatrix} + 0.8998 \begin{bmatrix} 1 & 3 \end{bmatrix} + 0.2041 \begin{bmatrix} 4 & 3 \end{bmatrix}}{0.9302 + 0.2758 + 0.8998 + 0.2041} = \begin{bmatrix} 1.6232 & 2.4779 \end{bmatrix} \tag{17}$$

$$\sigma_1^{(2)} = \sqrt{\frac{\frac{1}{2}\sum_{n=1}^{4} p^1(1|n)\left\|\mathbf{x}_n - \mu_1^{(2)}\right\|^2}{\sum_{n=1}^{4} p^1(1|n)}} = \sqrt{\frac{1}{2}\frac{0.9302(0.6168) + 0.2758(5.8774) + 0.8998(0.6610) + 0.2041(5.9216)}{0.9302 + 0.2758 + 0.8998 + 0.2041}} \tag{18}$$

$$p_1^{(2)} = \frac{1}{4}\sum_{n=1}^{4} p^1(1|n) = \frac{0.9302 + 0.2758 + 0.8998 + 0.2041}{4} = 0.5775 \tag{19}$$

For $N = 4, k = 2$ and $j = 1$

$$\mu_2^{(2)} = \frac{\sum_{n=1}^{4} p^1(2|n)\mathbf{x}_n}{\sum_{n=1}^{4} p^1(2|n)} = \frac{0.0698\begin{bmatrix}1 & 2\end{bmatrix} + 0.7242\begin{bmatrix}4 & 2\end{bmatrix} + 0.1002\begin{bmatrix}1 & 3\end{bmatrix} + 0.7959\begin{bmatrix}4 & 3\end{bmatrix}}{0.0698 + 0.7242 + 0.1002 + 0.7959} = \begin{bmatrix}3.6984 & 2.5302\end{bmatrix} \tag{20}$$

$$\sigma_2^{(2)} = \sqrt{\frac{\frac{1}{2}\sum_{n=1}^{4} p^1(2|n)\left\|\mathbf{x}_n - \mu_2^{(2)}\right\|^2}{\sum_{n=1}^{4} p^1(2|n)}} = \sqrt{\frac{1}{2}\frac{0.0698(7.5623) + 0.7242(0.3721) + 0.1002(7.5020) + 0.7959(0.3117)}{0.0698 + 0.7242 + 0.1002 + 0.7959}} \tag{21}$$

$$p_1^{(2)} = \frac{1}{4}\sum_{n=1}^{4} p^1(2|n) = \frac{0.0698 + 0.7242 + 0.1002 + 0.7959}{4} = 0.4225 \tag{22}$$

Now the convergence criteria can be determined using $\left|\mu_k^{(j)} - \mu_k^{(j+1)}\right| < \epsilon$ & $\left|\sigma_k^{(j)} - \sigma_k^{(j+1)}\right| < \epsilon$ & $\left|p^{(j)}(k|n) - p^{(j+1)}(k|n)\right| < \epsilon$ for some epsilon (ϵ) and distance calculation (Euclidian distance).

2.4 Summary - Expectation Maximization

The Expectation-Maximization (EM) algorithm provides a powerful and general-purpose framework for estimating parameters in models with latent variables. By iteratively alternating between estimating missing data (E-Step) and optimizing parameters (M-Step), the EM algorithm refines its estimates to maximize the likelihood function.

Key features of the EM algorithm include:

- It is particularly effective when dealing with incomplete data or probabilistic models with hidden variables.

- The likelihood function is guaranteed to increase (or remain unchanged) at each iteration.
- The algorithm provides a robust approach to estimating parameters for Gaussian Mixture Models (GMMs), Hidden Markov Models (HMMs), and other probabilistic models.

Despite its advantages, the EM algorithm has some limitations:

- It may converge to a local maximum, making initialization crucial for obtaining good results.
- It can be computationally expensive, especially for large datasets, due to iterative updates.
- The convergence speed can be slow, particularly when parameter updates result in small likelihood improvements.

In practice, the EM algorithm is widely used in clustering, density estimation, and latent variable modeling, making it a fundamental tool in statistical machine learning and artificial intelligence applications. Proper initialization strategies and convergence monitoring are key to ensuring the effectiveness of the algorithm.

3 Bayes Classifier

The Bayes Classifier is a fundamental statistical classification method based on Bayes' theorem. It provides a probabilistic approach to classification by computing the posterior probability of a sample belonging to each class and assigning it to the class with the highest probability. The classifier assumes that the data follows a multivariate normal distribution, where each class is characterized by a distinct mean vector and covariance matrix.

The Bayes Classifier is particularly effective in scenarios where class distributions can be well approximated by Gaussian densities. The classifier's decision rule is derived from the maximum a posteriori (MAP) estimation, making it optimal under the assumption of correct prior and likelihood distributions. When the class covariance matrices are equal, the Bayes Classifier results in a linear decision boundary; otherwise, it produces quadratic decision boundaries.

This section introduces the mathematical formulation of the Bayes Classifier, describes its training process, and presents an algorithmic implementation for classification. The goal is to provide a structured understanding of the classifier's mechanics, its assumptions, and its practical application in supervised learning tasks.

3.1 Mathematical Description of Bayes Classifier

The Bayes classifier is a probabilistic model used for classification tasks, where an input sample is assigned to the most probable class based on Bayes' theorem. Given an input sample represented as a feature vector $\mathbf{x}_0 \in \mathbb{R}^n$, the classifier determines the class $\mathbf{C} = [C_1, C_2, \ldots, C_c]$, where each class C_j is indexed by $j = 1, 2, \ldots, c$.

The Bayes classifier extends the general multivariate normal case, where each class has its own covariance matrix Σ_j. Each class C_j is assumed to be normally distributed, centered around a mean vector μ_j with a covariance matrix Σ_j. These parameters, μ_j and Σ_j, are estimated from the training data.

Since $\mathbf{x}_0$ is an n-dimensional observed data vector, the probability of $\mathbf{x}_0$ belonging to class C_j is computed using Bayes' theorem:

$$P(C_j|\mathbf{x}_0) = \frac{P(C_j)g(\mathbf{x}_0; \mu_j, \Sigma_j)}{\sum_{i=1}^{c} P(C_i)g(\mathbf{x}_0; \mu_i, \Sigma_i)} \tag{23}$$

where:

- $P(C_j|\mathbf{x}_0)$ is the posterior probability of class C_j given the observed sample $\mathbf{x}_0$.
- $P(C_j)$ is the prior probability of class C_j.
- $g(\mathbf{x}_0; \mu_j, \Sigma_j)$ is the class-conditional probability density function (PDF) modeled as a multivariate normal distribution:

$$g(\mathbf{x}_0; \mu_j, \Sigma_j) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_j|}} \exp\left(-\frac{1}{2}(\mathbf{x}_0 - \mu_j)^T \Sigma_j^{-1} (\mathbf{x}_0 - \mu_j)\right) \tag{24}$$

where $|\Sigma_j|$ and Σ_j^{-1} are the determinant and inverse of the covariance matrix Σ_j, respectively.

Using the density function $g(\mathbf{x}_0; \mu_j, \Sigma_j)$, the Bayes classifier can be expressed in terms of the prior probabilities, $P(C_i)$, and posterior probability of class membership as follows:

$$P(C_j|\mathbf{x}_0) = \frac{P(C_j)\frac{1}{\sqrt{(2\pi)^n|\Sigma_j|}}\exp\Big[-\frac{1}{2}\big(\mathbf{x}_0-\mu_j\big)^T\Sigma_j^{-1}\big(\mathbf{x}_0-\mu_j\big)\Big]}{\sum_{i=1}^{c}P(C_i)\frac{1}{\sqrt{(2\pi)^n|\Sigma_i|}}\exp\Big[-\frac{1}{2}\big(\mathbf{x}_0-\mu_i\big)^T\Sigma_i^{-1}\big(\mathbf{x}_0-\mu_i\big)\Big]} \tag{25}$$

The final classification decision is made by selecting the class C_j that maximizes the posterior probability of $\mathbf{x}_0$ as the class $C_{\hat{y}}$:

$$\hat{y} = \arg\max_j P(C_j|\mathbf{x}_0) \tag{26}$$

where $\hat{y}$ represents the predicted class label for the input sample $\mathbf{x}_0$.

Under the assumption that $\Sigma_j = \Sigma$ for all j, meaning that all classes share the same covariance matrix, the classifier simplifies to a linear decision boundary. Otherwise, when each class has its own covariance matrix, the classifier forms quadratic decision boundaries.

The Bayes classifier provides a robust mathematical framework for classification, leveraging both prior knowledge and observed data likelihoods to make optimal predictions [1, 3, 4].

3.2 Training Algorithm

The training process of the Bayes Classifier involves estimating key statistical parameters from the given training dataset. The input consists of training samples represented by feature vectors $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T$ and their corresponding class labels $\mathbf{y} = [y_1, y_2, \ldots, y_N]^T$. The objective of the training phase is to estimate the prior probabilities $P(C_j)$, mean vectors μ_j, and covariance matrices Σ_j for each class C_j.

- The prior probability $P(C_j)$ is estimated as the proportion of training samples belonging to class C_j.
- The mean vector μ_j for each class is computed as the average of the feature vectors associated with that class.
- The covariance matrix Σ_j is calculated using the statistical variance and covariance of features for each class.

These parameters define the multivariate normal distribution used for classification. The computed class statistics are then used in the testing phase to evaluate new samples.

3.3 Testing Algorithm for Classification

The testing phase of the Bayes Classifier involves predicting the class label for a new input sample $\mathbf{x}_0$ based on the parameters estimated during training. The input to the testing algorithm includes:

- The test sample $\mathbf{x}_0$.

Algorithm 2 Train Bayes Classifier

Input: Training data $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T$, labels $\mathbf{y} = [y_1, y_2, \ldots, y_N]^T$, number of classes c
Output: Class prior probabilities $P(C_j)$, class mean vectors μ_j, and covariance matrices Σ_j for each class C_j
function TrainBayesClassifier($\mathbf{X}, \mathbf{y}, c$)
  for each class $C_j, j = 1$ to c **do**
    Extract subset $\mathbf{X}_j$ where $\mathbf{x}_i \in \mathbf{X}$ and $y_i = C_j$
    Compute class prior probability:

$$P(C_j) = \frac{\text{Number of samples in } C_j}{N}$$

    Compute mean vector:

$$\mu_j = \frac{1}{|\mathbf{X}_j|} \sum_{\mathbf{x}_i \in \mathbf{X}_j} \mathbf{x}_i$$

    Compute covariance matrix:

$$\Sigma_j = \frac{1}{|\mathbf{X}_j|} \sum_{\mathbf{x}_i \in \mathbf{X}_j} (\mathbf{x}_i - \mu_j)(\mathbf{x}_i - \mu_j)^T$$

  end for
  return $P(C_j)$, μ_j, Σ_j for all j
end function

- The learned parameters: prior probabilities $P(C_j)$, mean vectors μ_j, and covariance matrices Σ_j for each class.

Algorithm 3 Test Bayes Classifier

Input: Test sample $\mathbf{x}_0$, trained parameters $P(C_j)$, μ_j, Σ_j for each class C_j, number of classes c
Output: Predicted class label $\hat{y}$
function TestBayesClassifier($\mathbf{x}_0, P(C_j), \mu_j, \Sigma_j, c$)
  for each class $C_j, j = 1$ to c **do**
    Compute posterior probability:

$$P(C_j|\mathbf{x}_0) = \frac{P(C_j) \frac{1}{\sqrt{(2\pi)^n |\Sigma_j|}} \exp\left[-\frac{1}{2}(\mathbf{x}_0 - \mu_j)^T \Sigma_j^{-1} (\mathbf{x}_0 - \mu_j) \right]}{\sum_{i=1}^{c} P(C_i) \frac{1}{\sqrt{(2\pi)^n |\Sigma_i|}} \exp\left[-\frac{1}{2}(\mathbf{x}_0 - \mu_i)^T \Sigma_i^{-1} (\mathbf{x}_0 - \mu_i) \right]}$$

  end for
  Classify $\mathbf{x}_0$ as the class $C_{\hat{y}}$ where:

$$\hat{y} = \arg\max_j P(C_j|\mathbf{x}_0)$$

  return $\hat{y}$
end function

For each class C_j, the classifier computes the posterior probability $P(C_j|\mathbf{x}_0)$ using Bayes' theorem. The likelihood function is modeled using a multivariate normal distribution as shown in Eq. 24. The posterior probability is computed as shown in Eq. 23.

The classifier assigns $\mathbf{x}_0$ to the class with the highest posterior probability:

$$\hat{y} = \arg\max_j P(C_j|\mathbf{x}_0) \tag{27}$$

This decision rule ensures that the sample is assigned to the most probable class given the observed data and prior class distributions.

The Bayes Classifier is particularly effective when the assumption of normally distributed class densities holds. However, in cases where class distributions deviate significantly from normality, alternative approaches such as non-parametric classifiers may be required.

3.4 Summary - Bayes Classifier

The Bayes Classifier is a probabilistic model that assigns a class label to an input sample based on Bayes' theorem. It leverages prior knowledge of class distributions and computes posterior probabilities to make classification decisions. The classifier assumes that the data is generated from a multivariate normal distribution, where each class is represented by a mean vector and a covariance matrix.

The training phase of the classifier involves estimating the class priors, mean vectors, and covariance matrices from the training data. These parameters define the likelihood function, which is used to compute the posterior probability during classification. The decision rule is based on selecting the class with the highest posterior probability.

During the testing phase, the classifier computes the posterior probability for each class using the multivariate Gaussian likelihood function. The class with the highest probability is assigned as the predicted label for the test sample. When class covariance matrices are identical, the decision boundaries are linear; otherwise, they become quadratic.

The Bayes Classifier is a powerful and mathematically grounded approach to classification. Its effectiveness is dependent on the assumption of normally distributed features, which may not always hold in real-world data. However, in cases where this assumption is reasonable, the Bayes Classifier provides optimal decision boundaries and serves as a fundamental building block for more advanced classification models.

4 Naive Bayes

Naive Bayes is a fundamental statistical classification algorithm based on Bayes' theorem, which provides a probabilistic framework for decision-making under uncertainty [1, 3]. It assumes that the features in a dataset are conditionally independent given the class label, a simplification that allows for efficient computation while maintaining competitive accuracy in many practical applications.

This classifier is particularly effective in text classification tasks such as spam filtering, sentiment analysis, and document categorization due to its ability to handle high-dimensional data. Despite its "naive" assumption of independence among features, Naive Bayes often performs well in real-world scenarios, demonstrating its robustness and reliability in various domains.

This section introduces the mathematical foundation of the Naive Bayes classifier, describes different variants such as Gaussian, Multinomial, and Bernoulli Naive Bayes, and presents its strengths and limitations. Additionally, the algorithmic implementation of the classifier is outlined to provide a structured approach to its application in practical problems.

4.1 Naive Bayes Model as a Classifier

Naive Bayes classifiers are a family of probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Bayes' theorem describes the probability of an event, based on prior knowledge of conditions that might be related to the event. The formula is:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \tag{28}$$

In the context of classification:

- $P(A|B)$ is the posterior probability of class A given predictor(s) B.
- $P(B|A)$ is the likelihood which is the probability of predictor(s) B given class A.
- $P(A)$ is the prior probability of class A.
- $P(B)$ is the prior probability of predictor(s) B.

The types of Naive Bayes are Gaussian Naive Bayes, where assumptions are made that the features follow a normal distribution; Multinomial Naive Bayes, typically used for document classification, where the features are the frequencies of the words or tokens; and Bernoulli Naive Bayes, which is useful when features are binary (0s and 1s).

The independence assumption of Naive Bayes is that all features are independent of each other given the class label. While this assumption is rarely true in real-world data, the classifier often performs well in practice.

There are a few advantages, which include being simple and easy to implement, requiring a small amount of training data to estimate the parameters, performing well in multi-class prediction, and being effective in scenarios where the independence assumption holds.

Limitations also exist with assumptions that independent features are rarely true, which can be a major limitation, not the best choice for tasks with high feature interdependency, and may perform

poorly if the categorical variable has a category in the test data set that was not present in the training data set.

The Naive Bayes classifier is valuable in the data scientist's toolkit, especially for applications where computational efficiency and simplicity are critical. Despite its 'naive' assumption, it can yield surprisingly accurate results, particularly in NLP tasks. Understanding its underlying principles, such as Bayes' theorem and the independence assumption, is crucial for effectively applying this model and interpreting its results. Integration with preprocessing steps like tokenization and feature extraction is often necessary for practical applications.

4.2 Naive Bayes Algorithm

Naive Bayes is a probabilistic classifier based on Bayes' Theorem, assuming independence among features. It is particularly useful for text classification tasks, such as spam detection and sentiment analysis. The algorithm consists of two main steps: training and testing. During training, the prior probabilities and conditional probabilities (likelihoods) for each class are estimated. In the testing phase, these probabilities are used to compute the posterior probability for each class given a new sample, and the sample is assigned to the class with the highest posterior probability.

Training Algorithm for Naive Bayes Classifier

Training Phase:

- The algorithm calculates the prior probability $P(C_j)$ for each class.
- For each feature, the conditional probability $P(x_d|C_j)$ is estimated.
- If the feature is continuous, it is modeled using a Gaussian distribution with mean μ_{jd} and variance σ^2_{jd}.
- If the feature is categorical, probability estimates are computed using frequency counts with optional Laplace smoothing.

Testing Algorithm for Naive Bayes Classification

Testing Phase:

- For a given test sample $\mathbf{x}_0$, the posterior probability $P(C_j|\mathbf{x}_0)$ is computed using Bayes Theorem.
- Logarithms are used to prevent numerical underflow in probability calculations.
- The class with the highest posterior probability is chosen as the predicted class.

In this code, TrainNaiveBayes function implements the Naive Bayes classification. It involves calculating the prior probabilities of each class, determining the likelihood of each feature within each class, and then using these to calculate the posterior probability of each class given a test data point. The class with the highest posterior probability is chosen as the prediction. The independence assumption of Naive Bayes simplifies the computation of likelihoods but can be a limitation if features are dependent. This high-level pseudocode provides a general structure for Naive Bayes and can be adapted to specific classification tasks or expanded with details on probability calculations and feature handling.

Algorithm 4 Train Naive Bayes Classifier

Input: Training data $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T$, labels $\mathbf{y} = [y_1, y_2, \ldots, y_N]^T$, number of classes c
Output: Class prior probabilities $P(C_j)$, conditional feature probabilities $P(x_d|C_j)$
function TRAINNAIVEBAYES($\mathbf{X}, \mathbf{y}, c$)
for each class $C_j, j = 1$ to c **do**
Extract subset $\mathbf{X}_j$ where $\mathbf{x}_i \in \mathbf{X}$ and $y_i = C_j$
Compute class prior probability:

$$P(C_j) = \frac{\text{Number of samples in } C_j}{N}$$

for each feature dimension $d = 1$ to D **do**
Estimate conditional probability $P(x_d|C_j)$ based on feature type:
if feature is continuous **then**
Compute mean and variance for Gaussian distribution:

$$\mu_{jd} = \frac{1}{|\mathbf{X}_j|} \sum_{\mathbf{x}_i \in \mathbf{X}_j} x_{id}$$

$$\sigma_{jd}^2 = \frac{1}{|\mathbf{X}_j|} \sum_{\mathbf{x}_i \in \mathbf{X}_j} (x_{id} - \mu_{jd})^2$$

else
Compute frequency-based probabilities for categorical features:

$$P(x_d|C_j) = \frac{\text{count of } x_d \text{ in } C_j + \alpha}{\text{total count in } C_j + |V|}$$

where α is a smoothing parameter and $|V|$ is the vocabulary size.
end if
end for
end for
return $P(C_j)$, μ_{jd}, σ_{jd}^2 (for continuous features), $P(x_d|C_j)$ (for categorical features)
end function

Algorithm 5 Test Naive Bayes Classifier

Input: Test sample $\mathbf{x}_0$, trained parameters $P(C_j)$, μ_{jd}, σ_{jd}^2, $P(x_d|C_j)$, number of classes c
Output: Predicted class label $\hat{y}$
function TestNaiveBayes($\mathbf{x}_0, P(C_j), \mu_{jd}, \sigma_{jd}^2, P(x_d|C_j), c$)
  for each class $C_j, j = 1$ to c **do**
    Compute log posterior probability:

$$\log P(C_j|\mathbf{x}_0) = \log P(C_j) + \sum_{d=1}^{D} \log P(x_{0d}|C_j)$$

where:

- For continuous features (Gaussian assumption):

$$P(x_{0d}|C_j) = \frac{1}{\sqrt{2\pi\sigma_{jd}^2}} \exp\left(-\frac{(x_{0d} - \mu_{jd})^2}{2\sigma_{jd}^2}\right)$$

- For categorical features (Multinomial/Bernoulli model):

$$P(x_{0d}|C_j) \text{ is retrieved from frequency-based estimates}$$

  end for
  Classify $\mathbf{x}_0$ as the class $C_{\hat{y}}$ where:

$$\hat{y} = \arg\max_j P(C_j|\mathbf{x}_0)$$

  return $\hat{y}$
end function

4.3 Summary - Naive Bayes

Naive Bayes classifiers represent a family of probabilistic models that apply Bayes' theorem to classify data points into predefined categories. The classifier assumes that the features are independent given the class label, an assumption that simplifies computation and enables fast classification.

The Naive Bayes model is a probabilistic classifier that leverages Bayes' theorem to compute posterior probabilities for classification. It is highly efficient computationally and performs well in high-dimensional datasets, making it particularly useful for large-scale applications. The model has several variants, including Gaussian Naive Bayes for continuous data, Multinomial Naive Bayes for frequency-based data such as text classification, and Bernoulli Naive Bayes for binary feature data. Despite its strong independence assumption, Naive Bayes often yields effective results in practice, especially in text classification and natural language processing tasks. However, its limitations include reduced effectiveness when features are highly correlated and potential challenges in handling unseen categorical variables in test data.

Naive Bayes is an essential tool in machine learning and data science, offering a simple yet powerful approach to classification problems. Its ability to handle large-scale datasets with minimal training requirements makes it a valuable algorithm in various real-world applications.

5 Mixture Models

Gaussian Mixture Models (GMMs) and other mixture models are fundamental techniques in classical generative AI used for modeling complex, multidimensional data distributions. By combining multiple probability distributions, such as Gaussian, Poisson, or Exponential distributions, these models can capture the underlying statistical properties of data and generate new, statistically determined data points. This section provides a detailed exploration of GMMs, including their mathematical definitions, parameter estimation using the Expectation-Maximization (EM) algorithm, and practical applications. Various types of mixture models, such as Diagonal, Spherical, Tied, and Full GMMs, are discussed, with examples illustrating how these models can be used to represent and analyze multidimensional datasets.

The EM algorithm is introduced as the primary approach for estimating the parameters of mixture models. By iteratively applying the Expectation step (E-Step) and the Maximization step (M-Step), the algorithm refines the parameter estimates to achieve a maximum likelihood solution. We explore different types of mixture models beyond GMMs, including Poisson Mixture Models (PMMs) and Exponential Mixture Models (EMMs), to highlight the flexibility and adaptability of the EM algorithm for a wide range of data types.

Figure 1 captures the data distribution as a mixture of two Gaussian distributions, each with its own mean and covariance.

5.1 Defining a Gaussian Mixture Models (GMMs)

A Gaussian Mixture Model is a probabilistic model that assumes that the data is generated from a mixture of several Gaussian distributions with unknown parameters. Each Gaussian distribution in the mixture is defined by its mean vector and covariance matrix. The probability density function of a GMM for a d-dimensional data point x is given by:

$$p(x) = \sum_{k=1}^{K} p_k g(x \mid \mu_k, \Sigma_k)$$

where, K is the number of Gaussian components, p_k are the mixing coefficients, with $\sum_{k=1}^{K} p_k = 1$ and $0 \leq p_k \leq 1$, and $g(x \mid \mu_k, \Sigma_k)$ is the multivariate Gaussian distribution with mean vector μ_k and covariance matrix Σ_k:

$$g(x \mid \mu_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)\right)$$

5.2 Types of GMMs

Different types of GMMs include Diagonal GMM, Spherical GMM, Tied GMM, and Full (Unrestricted) GMM. This subsection covers the fundamental concepts, definitions, and formulas related to these GMMs, with a focus on their application to modeling data distributions. We will illustrate these concepts with practical examples to demonstrate how these GMMs can be used to model data.

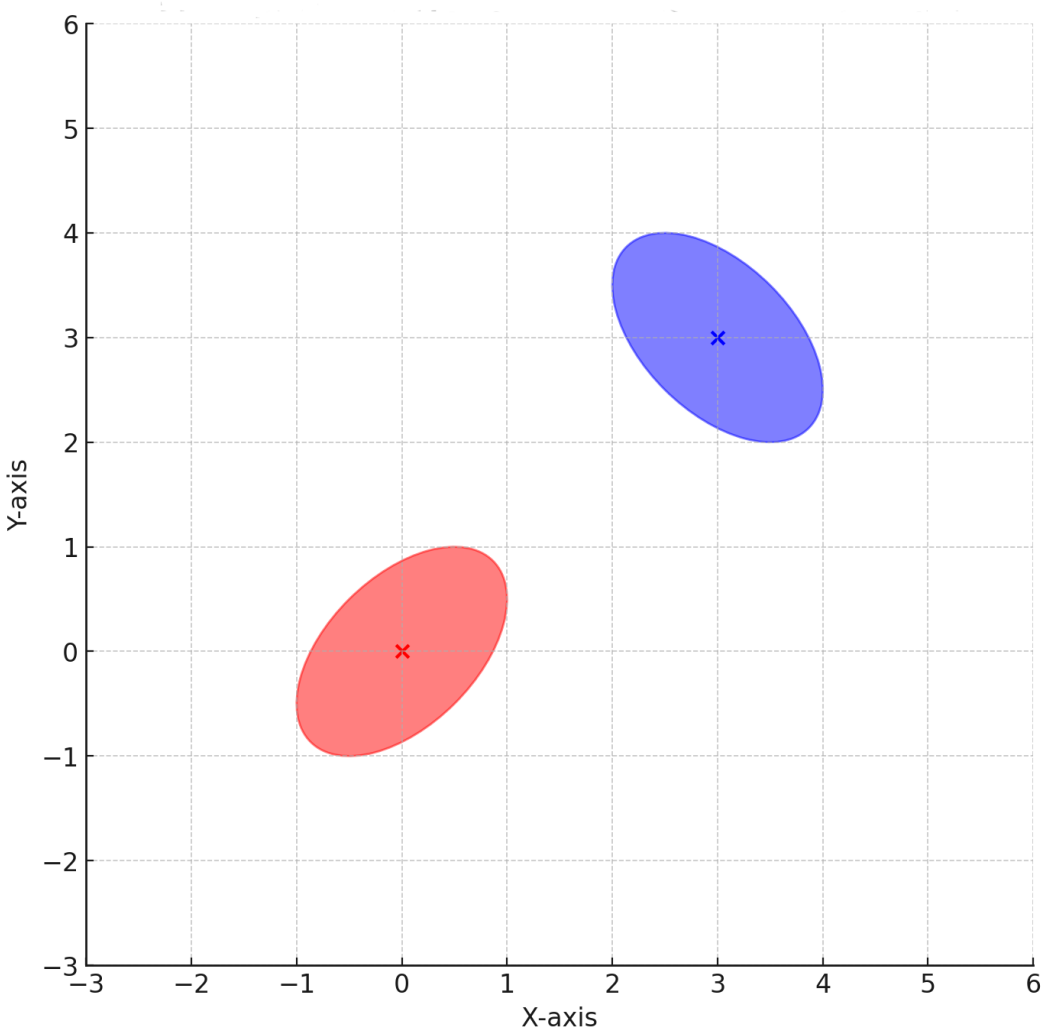

Figure 1: Example of Gaussian Mixture Model

Diagonal GMM

In a Diagonal GMM, the covariance matrices Σ_k are diagonal, meaning that the off-diagonal elements are zero. This assumes that the features are uncorrelated, but each feature can have a different variance.

$$\Sigma_k = \text{diag}(\sigma_{k1}^2, \sigma_{k2}^2, \ldots, \sigma_{kd}^2)$$

Spherical GMM

In a Spherical GMM, the covariance matrices Σ_k are proportional to the identity matrix, meaning that all features have the same variance and are uncorrelated.

$$\Sigma_k \overline{\overline{21}} \sigma_k^2 I$$

Tied GMM

In a Tied GMM, all Gaussian components share the same covariance matrix Σ.

$$\Sigma_k = \Sigma$$

Full (Unrestricted) GMM

In a Full GMM, the covariance matrices Σ_k are full matrices, meaning that they can have any value and capture correlations between the features.

$$\Sigma_k = \begin{pmatrix} \sigma_{k11} & \sigma_{k12} & \cdots & \sigma_{k1d} \\ \sigma_{k21} & \sigma_{k22} & \cdots & \sigma_{k2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{kd1} & \sigma_{kd2} & \cdots & \sigma_{kdd} \end{pmatrix}$$

Example: Modeling a Bivariate Distribution with a Full GMM

Problem Statement

Consider a two-dimensional dataset generated from a mixture of two Gaussian distributions:

$$p(x) = \pi_1 \mathcal{N}(x \mid \mu_1, \Sigma_1) + \pi_2 \mathcal{N}(x \mid \mu_2, \Sigma_2)$$

with parameters:

$$\pi_1 = 0.4, \quad \mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \Sigma_1 = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$$

$$\pi_2 = 0.6, \quad \mu_2 = \begin{pmatrix} 3 \\ 3 \end{pmatrix}, \quad \Sigma_2 = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

The probability density function is:

$$p(x) = 0.4 \cdot \frac{1}{2\pi\sqrt{|\Sigma_1|}} \exp\left(-\frac{1}{2}(x-\mu_1)^T \Sigma_1^{-1}(x-\mu_1)\right) + 0.6 \cdot \frac{1}{2\pi\sqrt{|\Sigma_2|}} \exp\left(-\frac{1}{2}(x-\mu_2)^T \Sigma_2^{-1}(x-\mu_2)\right)$$

where:

$$|\Sigma_1| = 1 \cdot 1 - 0.5 \cdot 0.5 = 0.75, \quad |\Sigma_2| = 2 \cdot 2 - 0 \cdot 0 = 4$$

Substituting the values, we get:

$$p(x) = 0.4 \cdot \frac{1}{2\pi\sqrt{0.75}} \exp\left(-\frac{1}{2}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^T \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}^{-1} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) + 0.6 \cdot \frac{1}{2\pi\sqrt{4}} \exp\left(-\frac{1}{2}\begin{pmatrix} x_1 - 3 \\ x_2 - 3 \end{pmatrix}^T \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x_1 - 3 \\ x_2 - 3 \end{pmatrix}\right)$$

This example demonstrates how different types of Gaussian Mixture Models can be used to model multidimensional data by combining multiple Gaussian components.

5.3 Poisson Mixture Model (PMM)

The Poisson Mixture Model (PMM) is a type of mixture model used to represent count data, where the observations follow a Poisson distribution. The Poisson distribution is commonly used to model the number of occurrences of an event in a fixed interval of time or space when the occurrences are independent. The PMM is particularly useful when dealing with heterogeneous count data that exhibits multiple latent subpopulations, each of which follows a Poisson distribution with a different rate parameter.

Definition of the Poisson Distribution

A Poisson-distributed random variable X with rate parameter λ follows the probability mass function (PMF):

$$P(X = x \mid \lambda) = \frac{\lambda^x e^{-\lambda}}{x!}, \quad x \in \mathbb{N}_0, \quad \lambda > 0 \tag{29}$$

where:

- λ is the mean and variance of the Poisson distribution, which represents the expected number of occurrences.
- $x!$ is the factorial of x, ensuring that the probability values are properly normalized.
- The Poisson distribution assumes that events occur independently within a fixed period or space.

Poisson Mixture Model (PMM) Formulation

A Poisson Mixture Model assumes that the observed count data $\mathbf{X} = [x_1, x_2, \ldots, x_N]$ arises from a mixture of K underlying Poisson-distributed subpopulations. Each component k in the mixture has an associated rate parameter λ_k, which governs the Poisson distribution for that component.

The probability of an observed count x_n given the mixture components is given by:

$$P(x_n \mid \boldsymbol{\lambda}, \boldsymbol{p}) = \sum_{k=1}^{K} p_k \frac{\lambda_k^{x_n} e^{-\lambda_k}}{x_n!} \tag{30}$$

where:

- p_k is the mixing coefficient for component k, satisfying $\sum_{k=1}^{K} p_k = 1$ and $0 \leq p_k \leq 1$.
- λ_k is the Poisson rate parameter for component k.
- x_n represents an observed count.

5.4 Parameter Estimation using the Expectation-Maximization (EM) Algorithm

To estimate the parameters of the Poisson Mixture Model, we use the Expectation-Maximization (EM) algorithm, which iteratively updates the parameter estimates to maximize the likelihood function.

E-Step (Expectation)

In the E-Step, we compute the posterior probabilities (responsibilities) γ_{nk}, which represent the probability that an observation x_n belongs to mixture component k:

$$\gamma_{nk} = P(k \mid x_n) = \frac{p_k P(x_n \mid \lambda_k)}{\sum_{j=1}^{K} p_j P(x_n \mid \lambda_j)} \tag{31}$$

Expanding the likelihood term using the Poisson PMF:

$$\gamma_{nk} = \frac{p_k \frac{\lambda_k^{x_n} e^{-\lambda_k}}{x_n!}}{\sum_{j=1}^{K} p_j \frac{\lambda_j^{x_n} e^{-\lambda_j}}{x_n!}} \tag{32}$$

M-Step (Maximization)

In the M-Step, we update the Poisson rate parameter λ_k using the expected counts assigned to each component. The update formula is:

$$\lambda_k^{(j+1)} = \frac{\sum_{n=1}^{N} \gamma_{nk} x_n}{\sum_{n=1}^{N} \gamma_{nk}} \tag{33}$$

where:

- The numerator computes the total weighted sum of counts assigned to component k.
- The denominator normalizes by the total weight assigned to component k, ensuring a valid Poisson mean estimate.

The updated mixing coefficients π_k are computed as:

$$p_k^{(j+1)} = \frac{1}{N} \sum_{n=1}^{N} \gamma_{nk} \tag{34}$$

Convergence Criterion

The EM algorithm iterates until the parameter changes fall below a predefined threshold ϵ:

$$\|\lambda_k^{(j+1)} - \lambda_k^{(j)}\| < \epsilon, \quad \forall k \tag{35}$$

Example: Poisson Mixture Model with Two Components

Suppose we have a dataset of observed event counts $\mathbf{X} = [2, 3, 8, 4, 5, 6, 9, 1, 0, 7]$, and we assume that the data is generated from a mixture of two Poisson distributions.

Initialization:

$$\lambda_1^{(0)} = 3, \quad \lambda_2^{(0)} = 7, \quad \pi_1^{(0)} = 0.5, \quad \pi_2^{(0)} = 0.5$$

E-Step (First Iteration): Compute responsibilities using Eq. 31.

M-Step (First Iteration): Update parameters using Eq. 33 and Eq. 34.

The EM algorithm iterates until the Poisson rate parameters λ_k converge. The resulting λ_k values provide the estimated rates for each Poisson-distributed component in the mixture.

Summary - Poisson Mixture Model (PMM)

The Poisson Mixture Model (PMM) is a probabilistic framework for modeling heterogeneous count data by assuming that observations arise from a mixture of Poisson-distributed subpopulations. The Expectation-Maximization (EM) algorithm is used to estimate the parameters iteratively, where the E-Step computes the posterior probabilities of component membership and the M-Step updates the Poisson rate parameters based on expected counts.

By allowing multiple Poisson distributions to model different subpopulations within count data, the PMM provides a flexible and interpretable approach to clustering and density estimation in domains where count data is prevalent.

5.5 Exponential Mixture Model (EMM)

The Exponential Mixture Model (EMM) is a probabilistic model used to represent data that follows an exponential distribution. The exponential distribution is commonly used to model the time between independent events occurring at a constant rate, such as waiting times, failure rates, or arrival times in queuing systems. The EMM extends this by assuming that the observed data arises from a mixture of multiple exponential distributions, each corresponding to a different latent subpopulation.

Definition of the Exponential Distribution

The exponential distribution is defined by a single rate parameter λ, which determines the expected time until the next event. The probability density function (PDF) of an exponentially distributed random variable X is given by:

$$f(x \mid \lambda) = \lambda e^{-\lambda x}, \quad x \geq 0, \quad \lambda > 0 \tag{36}$$

where:

- λ is the rate parameter, representing the expected number of events per unit time.
- The mean of the distribution is given by $E[X] = \frac{1}{\lambda}$.
- The variance is given by $\text{Var}[X] = \frac{1}{\lambda^2}$.

Exponential Mixture Model (EMM) Formulation

An Exponential Mixture Model assumes that an observed dataset $\mathbf{X} = [x_1, x_2, \ldots, x_N]$ is generated from a mixture of K exponential distributions, each with its own rate parameter λ_k. The probability density function of the mixture model is:

$$P(x_n \mid \boldsymbol{\lambda}, \boldsymbol{p}) = \sum_{k=1}^{K} p_k \lambda_k e^{-\lambda_k x_n} \tag{37}$$

where:

- p_k is the mixing coefficient for component k, satisfying $\sum_{k=1}^{K} p_k = 1$ and $0 \leq p_k \leq 1$.
- λ_k is the rate parameter for component k.
- x_n represents an observed sample.

Parameter Estimation using the Expectation-Maximization (EM) Algorithm

To estimate the parameters of the Exponential Mixture Model, we use the Expectation-Maximization (EM) algorithm, which iteratively updates the parameters to maximize the likelihood function.

E-Step (Expectation)

In the E-Step, we compute the posterior probabilities (responsibilities) γ_{nk}, which represent the probability that an observation x_n belongs to mixture component k:

$$\gamma_{nk} = P(k \mid x_n) = \frac{p_k P(x_n \mid \lambda_k)}{\sum_{j=1}^{K} p_j P(x_n \mid \lambda_j)} \tag{38}$$

Expanding the likelihood term using the exponential PDF:

$$\gamma_{nk} = \frac{p_k \lambda_k e^{-\lambda_k x_n}}{\sum_{j=1}^{K} p_j \lambda_j e^{-\lambda_j x_n}} \tag{39}$$

M-Step (Maximization)

In the M-Step, we update the rate parameter λ_k using the expected values of x_n assigned to each component. The update formula is:

$$\lambda_k^{(j+1)} = \frac{\sum_{n=1}^{N} \gamma_{nk}}{\sum_{n=1}^{N} \gamma_{nk} x_n} \tag{40}$$

where:

- The numerator represents the total responsibility (posterior probability) assigned to component k.
- The denominator is the sum of the weighted observations assigned to component k.

The updated mixing coefficients π_k are computed as:

$$\pi_k^{(j+1)} = \frac{1}{N} \sum_{n=1}^{N} \gamma_{nk} \tag{41}$$

Convergence Criterion

The EM algorithm iterates until the parameter changes fall below a predefined threshold ϵ:

$$\|\lambda_k^{(j+1)} - \lambda_k^{(j)}\| < \epsilon, \quad \forall k \tag{42}$$

Example: Exponential Mixture Model with Two Components

Suppose we have a dataset of observed waiting times $\mathbf{X} = [0.5, 1.2, 2.5, 0.8, 1.1, 3.0, 4.2, 0.3, 2.7, 1.5]$, and we assume that the data is generated from a mixture of two exponential distributions.

Initialization:

$$\lambda_1^{(0)} = 1.5, \quad \lambda_2^{(0)} = 0.5, \quad \pi_1^{(0)} = 0.5, \quad \pi_2^{(0)} = 0.5$$

E-Step (First Iteration): Compute responsibilities using Eq. 38.

M-Step (First Iteration): Update parameters using Eq. 40 and Eq. 41.

The EM algorithm iterates until the exponential rate parameters λ_k converge. The resulting λ_k values provide the estimated rates for each exponentially distributed component in the mixture.

Summary - Exponential Mixture Model (EMM)

The Exponential Mixture Model (EMM) is a probabilistic framework for modeling heterogeneous waiting time or survival time data by assuming that observations arise from a mixture of exponential distributions. The Expectation-Maximization (EM) algorithm is used to estimate the parameters iteratively, where the E-Step computes the posterior probabilities of component membership and the M-Step updates the exponential rate parameters based on expected waiting times.

By allowing multiple exponential distributions to model different subpopulations within waiting time data, the EMM provides a flexible and interpretable approach to clustering and density estimation in domains where exponential waiting times are observed.

5.6 Expectation-Maximization (EM) Algorithm for Mixture Models

The EM algorithm is designed to work with various types of mixture models. It is modularized by introducing a function, SelectModel, that allows different probability distributions to be used.

The ExpectationMaximization function performs the following steps:

1. **Initialize Parameters**: Randomly initialize the parameters θ_k (which could include means, variances, and mixing probabilities).
2. **E-Step**: Compute the responsibility of each component for each data point using the selected probability density function (PDF).
3. **M-Step**: Update the parameters of the mixture model by maximizing the expected complete-data log-likelihood.
4. **Check for Convergence**: The process repeats until the difference between consecutive parameter estimates is below a threshold ϵ.

Algorithm 6 Expectation-Maximization (EM) Algorithm for Mixture Models

Input: Observed data $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T$, number of clusters K, probability model ModelType, convergence threshold ϵ
Output: Estimated parameters θ_k for each component k
function ExpectationMaximization($\mathbf{X}, K$, ModelType, ϵ)
Initialize: Component parameters $\theta_k^{(0)}$ (e.g., means, variances, mixing weights) randomly
Set iteration counter $j \leftarrow 0$
repeat
E-Step: Compute membership probabilities (responsibilities) $p^{(j)}(k|n)$ for each component k and each sample $\mathbf{x}_n$:

$$p^{(j)}(k|n) = \frac{p_k^{(j)} f(\mathbf{x}_n; \theta_k^{(j)})}{\sum_{k=1}^{K} p_k^{(j)} f(\mathbf{x}_n; \theta_k^{(j)})}$$

where $f(\mathbf{x}_n; \theta_k^{(j)})$ is defined by the selected probability model.
M-Step: Update parameters using membership probabilities:
for each component $k = 1$ to K **do**
Call SelectModel to update parameters:
$\theta_k^{(j+1)} \leftarrow$ SelectModel($\mathbf{X}, p^{(j)}(k|n), \theta_k^{(j)}$, ModelType)
Compute updated mixing probabilities:

$$p_k^{(j+1)} = \frac{1}{N} \sum_{n=1}^{N} p^j(k|n)$$

end for
Check for Convergence:
if $\|\theta_k^{(j+1)} - \theta_k^{(j)}\| < \epsilon$ for all k **then**
Break
end if
Set $j \leftarrow j + 1$
until convergence criteria are met
return θ_k, p_k for all k
end function

Mixture Model Selection

The SelectModel function is responsible for selecting the appropriate mixture model (e.g., Gaussian, Poisson, or Exponential) and updating its parameters accordingly:

- **Gaussian Mixture Model (GMM)**:

Algorithm 7 Select and Update Mixture Model Parameters

function SELECTMODEL($\mathbf{X}, p(k|n), \theta_k,$ MODELTYPE)

Compute updated mixing probabilities:

$$p_k^{(j+1)} = \frac{1}{N} \sum_{n=1}^{N} p^j(k|n)$$

if MODELTYPE = GMM (Gaussian Mixture Model) **then**

Compute updated mean and updated covariance:

$$\mu_k = \frac{\sum_{n=1}^{N} p(k|n)\mathbf{x}_n}{\sum_{n=1}^{N} p(k|n)}$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} p(k|n)(\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^T}{\sum_{n=1}^{N} p(k|n)}$$

return (μ_k, Σ_k, p_k)

else if MODELTYPE = PMM (Poisson Mixture Model) **then**

Compute updated Poisson rate parameter:

$$\lambda_k = \frac{\sum_{n=1}^{N} p(k|n)x_n}{\sum_{n=1}^{N} p(k|n)}$$

return (λ_k, p_k)

else if MODELTYPE = EMM (Exponential Mixture Model) **then**

Compute updated rate parameter for exponential distribution λ_k:

$$\lambda_k = \frac{\sum_{n=1}^{N} p(k|n)}{\sum_{n=1}^{N} p(k|n)x_n}$$

return (λ_k, p_k)

else

Default case: Use Maximum Likelihood Estimation (MLE) for general models. Solve θ_k:

$$\theta_k = \arg\max_{\theta} \sum_{n=1}^{N} p(k|n) \log f(\mathbf{x}_n; \theta)$$

return (θ_k, p_k)

end if

end function

- ◇ Updates the **mean vector** μ_k and **covariance matrix** Σ_k.
- ◇ The Gaussian probability density function (PDF) is used.

- **Poisson Mixture Model (PMM)**:
  - ◇ Updates the **Poisson rate parameter** λ_k.
  - ◇ The Poisson distribution is used to model count-based data.
- **Exponential Mixture Model (EMM)**:
  - ◇ Updates the **rate parameter** λ_k.
  - ◇ The Exponential distribution is used to model time-between-events data.
- **General Case**:
  - ◇ If none of the predefined models are selected, the function falls back on Maximum Likelihood Estimation (MLE), solving for the most likely parameters θ_k.

5.7 Summary - Mixture Models

In this section, we explored Gaussian Mixture Models (GMMs) and other mixture models, including their fundamental definitions, mathematical formulations, and practical applications. We examined how GMMs combine multiple Gaussian distributions to model complex data distributions, and we highlighted the use of the Expectation-Maximization (EM) algorithm for parameter estimation.

The modular SelectModel function allows for the integration of different types of probability distributions, such as GMMs, PMMs, and EMMs, making the EM algorithm highly adaptable for various data analysis tasks. Additionally, we discussed different types of GMMs, including Diagonal, Spherical, Tied, and Full GMMs, and demonstrated their applications through practical examples.

Overall, GMMs and other mixture models provide powerful tools for clustering, density estimation, and generative modeling, making them essential techniques in the field of machine learning and data analysis.

6 Summary

Statistical algorithms play a critical role in probabilistic modeling, classification, and clustering, forming the backbone of many modern machine learning and data science applications. This document provides a comprehensive overview of fundamental statistical algorithms, including the Expectation-Maximization (EM) algorithm, Bayesian classifiers, Naïve Bayes, and various Mixture Models such as Gaussian, Poisson, and Exponential Mixture Models.

The Expectation-Maximization (EM) algorithm is a widely used iterative approach for estimating parameters in probabilistic models, particularly when dealing with missing or latent variables. It alternates between an Expectation Step (E-Step), which estimates membership probabilities, and a Maximization Step (M-Step), which updates model parameters until convergence.

Bayesian classifiers, including the Naïve Bayes classifier, rely on Bayes theorem to compute posterior probabilities for classification. Despite the simplifying assumption of conditional independence between features, Naïve Bayes classifiers often perform remarkably well in applications such as text classification and spam detection.

Mixture models extend traditional probability distributions by representing complex data distributions as weighted combinations of simpler distributions. Gaussian Mixture Models (GMMs) are widely used for clustering and density estimation, while Poisson Mixture Models (PMMs) are suited for modeling count-based data, and Exponential Mixture Models (EMMs) are effective in modeling time-to-event data.

The document presents the theoretical foundations, algorithmic implementations, and mathematical derivations of these statistical techniques, highlighting their applications in various domains, including natural language processing, image analysis, and predictive modeling. By integrating probabilistic principles with computational methodologies, these statistical algorithms offer robust solutions for data-driven decision-making and pattern recognition.

Through the structured exploration of these methods, this document serves as a valuable resource for understanding the principles and practical implementations of statistical algorithms, demonstrating their significance in solving real-world data analysis challenges.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. URL: https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf.

[2] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22. DOI: 10.1111/j.2517-6161.1977.tb01600.x.

[3] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. 2nd. Wiley-Interscience, 2000.

[4] Carlo Tomasi. *Estimating Gaussian Mixture Densities with EM – A Tutorial*. Retrieved September 2006. Duke University Course Notes. 2006. URL: http://www.cs.duke.edu/courses/spring04/cps196.1/handouts/EM/tomasiEM.pdf.