Model-based Learning: Clustering with Gaussian Mixture Models

EL ADLOUNI Mohammed ali
Université Lyon 2 - Master MALIA

2 décembre 2024

Résumé

This report explores the implementation of a Gaussian mixture model for clustering quantitative data, utilizing the Expectation-Maximization (EM) algorithm. Two variants of the model are developed : a full model with unrestricted covariance matrices and a homoscedastic model with shared covariance matrices across clusters. The implementation is encapsulated within an R package, which includes functionality for parameter estimation, cluster assignment, and model selection via the Bayesian Information Criterion (BIC). The methodology is applied to the Iris and MNIST datasets, and the results are evaluated for clustering performance.

1 Introduction

Clustering is a key technique in machine learning, used to group data based on inherent characteristics. The Expectation-Maximization (EM) algorithm is particularly effective for modeling data as mixtures of probability distributions, often Gaussian, and uncovering hidden structures.

This report examines the EM algorithm's performance in clustering, focusing on its application to two datasets : the Iris dataset, with its simple structure, and the MNIST dataset, used for high-dimensional data analysis.

A custom implementation of the EM algorithm was developed in R, incorporating features like initialization methods (random sampling, k-means), iterative optimization (E-step, M-step), model selection (log-likelihood, BIC), and convergence visualization.

The algorithm was tested on both datasets to analyze its performance with low- and high-dimensional data. This report presents the methods, results, and insights from these experiments.

2 Theoretical Background

2.1 The Expectation-Maximization (EM) Algorithm

The EM algorithm is an iterative method used to estimate the parameters of probabilistic models, particularly when the data has latent (hidden) variables. In the context of GMMs, the EM algorithm estimates the parameters of the Gaussian components, namely the means μ_k, covariances Σ_k, and mixing coefficients π_k.

2.1.1 Initialization

The EM algorithm begins by initializing the parameters of the GMM. This step can be performed using several methods, such as random initialization or using clustering algorithms like k-means to estimate the initial parameters. Common initializations include :

— Randomly selecting K data points as the initial means μ_k.
— Using k-means clustering to determine initial cluster centers and covariance estimates.
— Initializing the mixing coefficients π_k to be equal for all components.
— Setting initial values for the covariance matrices Σ_k (e.g., as the empirical covariance of the data or a diagonal matrix).

The goal of initialization is to provide a starting point for the iterative steps of the EM algorithm.

2.1.2 E-step (Expectation Step)

In the E-step, the algorithm computes the *posterior probabilities* for each data point. posterior probabilities represent the probability that a given data point belongs to a particular Gaussian component. These posterior probabilities are computed using Bayes' theorem as follows :

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

where : - γ_{ik} is the posterior probability that the k-th Gaussian component has for the i-th data point. - x_i is the i-th data point. - π_k is the mixing coefficient of the k-th Gaussian. - $\mathcal{N}(x_i|\mu_k, \Sigma_k)$ is the Gaussian probability density function evaluated at x_i.

The posterior probabilities are used to estimate the contribution of each Gaussian component to each data point.

2.1.3 M-step (Maximization Step)

In the M-step, the parameters of the GMM (means μ_k, covariances Σ_k, and mixing coefficients π_k) are updated using the posterior probabilities computed in the E-step. The updates are computed as follows :

— Update the means :

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

where N is the number of data points.

— Update the covariances :

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

— Update the mixing coefficients :

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

These updates are performed until convergence, which typically occurs when the change in the log-likelihood between successive iterations is below a predefined threshold.

2.2 Loss Computation

The loss function for GMMs is the *log-likelihood* of the data under the current model parameters. The log-likelihood is computed as :

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i | \mu_k, \Sigma_k) \right)$$

The goal of the EM algorithm is to maximize the log-likelihood by iterating through the E-step and M-step. As the algorithm progresses, the log-likelihood generally increases, indicating that the model parameters are better fitting the data.

3 Implementation

The `em_algorithm` function is the heart of the package, providing an easy-to-use interface for fitting Gaussian Mixture Models via the EM algorithm. The modularity of the implementation, with separate functions for initialization, the E-step, M-step, loss computation, and evaluation, makes it flexible and extensible. This structure allows the package to be adapted to various clustering problems, including those with different covariance models and initialization strategies.

This section describes the implementation of the EM (Expectation-Maximization) algorithm for Gaussian Mixture Models (GMM), which has been packaged into an R function. The `em_algorithm` function serves as the main function, doing the execution of the entire algorithm by calling several sub-functions. The individual functions that comprise the full EM algorithm are stored in separate files, promoting modularity and ease of maintenance.

3.1 Main Function : `em_algorithm`

The core of the package is the `em_algorithm` function, which runs the EM algorithm for clustering data points into a specified number of Gaussian components. It uses a variety of helper functions to initialize parameters, perform the E-step (t_i_k calculation), and M-step (parameter update), and monitor convergence. The function also computes the loss (log-likelihood) and BIC (Bayesian Information Criterion), and optionally plots the loss at each iteration.

Key Parameters :

— `X` : The data matrix (observations $\times$ features).
— `K` : The number of clusters.
— `model` : Specifies the type of covariance model for the GMM. Options include "basic", "homoscedastic", "CEM", and "SEM".
— `nb_initializations` : The number of random initializations to perform to avoid poor local optima.
— `max_iterations` : The maximum number of iterations for the EM algorithm.
— `loss_threshold` : Convergence threshold for the log-likelihood change between iterations.
— `plot_losses` : A boolean flag to plot the evolution of the loss during the iterations.
— `nb_iter_init_step` : The number of iterations for the initialization phase.
— `init_method` : Method to initialize the parameters ("random" or "kmeans").

Function Flow :

1. **Parameter Initialization :** The function first calculates the number of model parameters using the `compute_nb_model_params` function. Then, it initializes the model parameters using the `initialize_parameters` function.
2. **EM Iterations :**
   — **E-step :** posterior probabilities are computed using the `E_step` function, which calculates the posterior probability of each point belonging to each cluster.
   — **M-step :** The parameters are updated based on the posterior probabilities computed in the E-step using the `M_step` function.
3. The log-likelihood is calculated at each iteration using the `compute_loss` function, and convergence is checked by comparing the change in log-likelihood.
4. **Final Output :** After convergence or reaching the maximum number of iterations, the algorithm returns the final parameters (means, covariances, and proportions), the log-likelihood values for each iteration, the final BIC value, and the predicted cluster assignments for each observation.

3.2 Helper Functions

3.2.1 initialize_parameters

This function initializes the parameters (means, covariances, and proportions) of the GMM model. It supports two initialization methods :

— **K-means initialization :** Uses the K-means algorithm to determine initial cluster centers, which are then used as the means of the Gaussian components.
— **Random initialization :** Randomly selects data points as initial means and assigns identity covariance matrices to each cluster.

The function runs the EM algorithm for a specified number of iterations (`nb_iter_init_step`) for each initialization to refine the parameter estimates, selecting the initialization that minimizes the loss.

3.2.2 E_step

The E-step computes the posterior probabilities, which are the posterior probabilities that each data point belongs to each of the K Gaussian components. This is done by computing the likelihood of each observation under each Gaussian component using the multivariate normal density function. The likelihoods are then normalized to obtain the posterior probabilities.

3.2.3 M_step

The M-step updates the parameters of the model based on the current posterior probabilities. It computes :

— The proportions (weights) of each Gaussian component by summing the posterior probabilities for each cluster and dividing by the total number of observations.
— The means (μ) of the Gaussian components as the weighted averages of the data points, where the weights are the posterior probabilities.
— The covariances (σ) for each cluster. Depending on the model type :
   — In the "homoscedastic" model, a common covariance matrix is shared across all clusters.

3.2.4 compute_loss

This function calculates the log-likelihood (loss) for the current model parameters (means, covariances, and proportions) given the data. It computes the likelihood of each data point under each Gaussian component and sums the log of the total likelihood for all data points.

3.2.5 compute_nb_model_params

This function computes the number of model parameters based on the number of clusters (K), the covariance model type (`model`), and the number of features (`nb_features`). The number of parameters depends on the model type :

- **Basic :** Each cluster has a mean vector (dimension equal to the number of features) and a covariance matrix.
- **Homoscedastic :** A shared covariance matrix is used for all clusters.

3.2.6 bic

This function computes the Bayesian Information Criterion (BIC) based on the log-likelihood and the number of model parameters. The BIC is used for model selection, where a lower BIC indicates a better-fitting model.

3.2.7 predict_gmm

This function applies the trained GMM to new data points (X_{test}). It uses the final model parameters to calculate the posterior probabilities for each data point and assigns each observation to the cluster with the highest posterior probability.

3.2.8 evaluate_gmm

This function evaluates the performance of the GMM by comparing the predicted cluster assignments to the true labels. It calculates metrics like the accuracy or adjusted Rand index to assess the quality of the clustering.

3.3 Model Types

The package supports several covariance models, which influence how the covariance matrices of the Gaussian components are handled :

- **"basic"** : Each cluster has its own covariance matrix.
- **"homoscedastic"** : All clusters share a common covariance matrix.

Each model type affects the number of parameters in the model and the computations in the M-step, providing flexibility for different types of data and clustering scenarios.

4 Application to Iris Dataset

This section presents the application of the EM algorithm for Gaussian Mixture Models (GMM) on the Iris dataset. We describe the preprocessing steps, the model training process, and the clustering results. Additionally, we evaluate the model's performance on a test dataset and compare the clustering quality with the true labels of the Iris dataset using a confusion matrix and performance scores.

4.1 Iris Data and Preprocessing

The Iris dataset consists of 150 observations, each with four features : sepal length, sepal width, petal length, and petal width. The dataset includes three classes of Iris flowers : Setosa, Versicolor, and Virginica, with 50 observations per class. The goal is to use the EM algorithm to cluster the dataset into three Gaussian components corresponding to these classes. The total number of model parameters is 45.

Before applying the EM algorithm, we standardize the features using z-score normalization. This step ensures that all features contribute equally to the clustering process, as they have different units and ranges. The dataset is split into training and test sets, with 80

4.2 Model Training

The EM algorithm was applied to the Iris dataset using a Gaussian Mixture Model (GMM) with three components. We initialized the parameters randomly and selected a basic covariance model for simplicity. The model was trained with a convergence threshold on the log-likelihood and a maximum of 50 iterations.

The key hyperparameters for the EM algorithm were as follows :

— Number of clusters (K) : 3
— Covariance model : "basic"
— Number of initializations : 5
— Maximum iterations : 50
— Loss threshold : 1e-6
— Initialization method : random (means are randomly initialized)

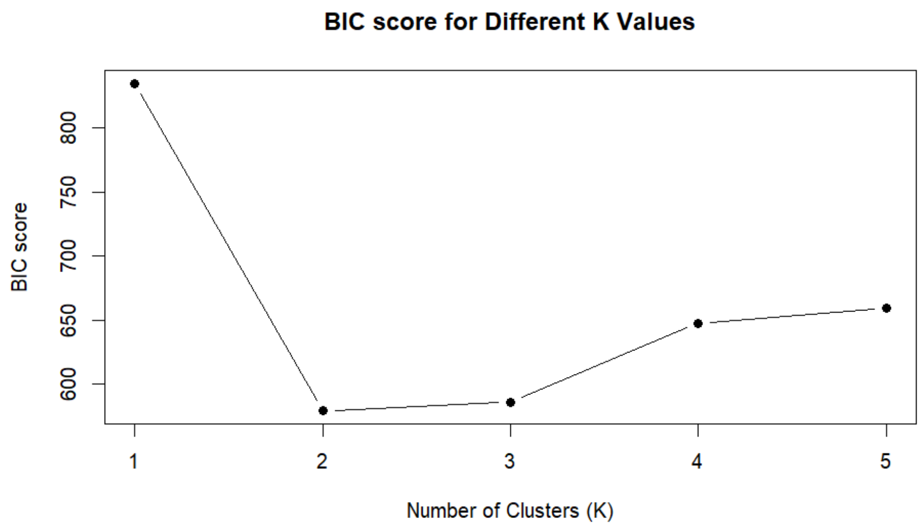

FIGURE 1 – BIC score for different K values

During the training process, the log-likelihood was calculated at each iteration to monitor convergence. The BIC score was also computed to evaluate the model fit for different cluster counts. We can see in figure 1 that BIC score is lowest for K=2. This happens because there are two overlapping clusters in the data.

4.3 Clustering Results

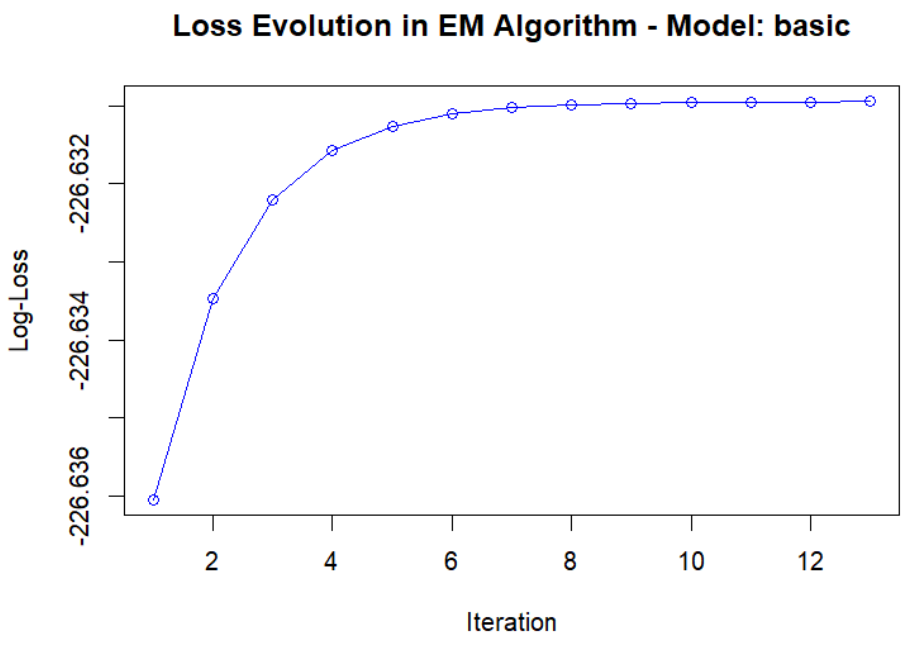

FIGURE 2 – Log-Likelihood Over Iterations for Iris dataset

The plot above shows the log-likelihood values over the iterations, indicating how well the EM algorithm converged to a stable solution. As the number of iterations increased, the log-likelihood value improved, demonstrating that the model was fitting the data better with each step.

With the training dataset, we got a Training Accuracy of 98.33%

4.4 Model Evaluation on Test Dataset

To evaluate the quality of the clustering, we applied the trained GMM model to a test dataset and compared the predicted cluster labels with the true species labels from the Iris dataset. We used a confusion matrix to visualize how well the clusters matched the actual classes.

	True Label 1	**True Label 2**	**True Label 3**
Aligned Cluster 1	40	0	0
Aligned Cluster 2	0	38	0
Aligned Cluster 3	0	2	40

TABLE 1 – Confusion Matrix for Iris dataset

The confusion matrix above shows the comparison between the predicted cluster assignments and the true species labels. Each cell represents the number of data points assigned to a particular cluster (rows) and their true class (columns). The diagonal elements represent the correctly clustered instances. We get for testing data an accuracy of 100 %

4.5 Conclusion

The application of the EM algorithm to the Iris dataset demonstrated the effectiveness of Gaussian Mixture Models for clustering. However, the dataset is simple. Now, we will try our models on MNIST data which is more complex to see if it works as well on this type of data.

5 Application to MNIST Dataset

5.1 MNIST Data and Preprocessing

The MNIST dataset contains grayscale images of handwritten digits ranging from 0 to 9, with each image represented as a 28x28 grid of pixel intensity values. The training dataset was subsampled to 5000 samples for computational efficiency. Labels corresponding to the digits were also loaded and processed.

The data preprocessing involved standardization to ensure that all features had a mean of zero and unit variance. Dimensionality reduction techniques such as PCA (Principal Component Analysis) and t-SNE (t-distributed Stochastic Neighbor Embedding) were applied to further reduce the dimensionality of the dataset for clustering purposes.

5.2 Choose number of clusters

We see in figure that after cluster equal to 10 BIC score does not improve by a lot. So we choose K = 10.

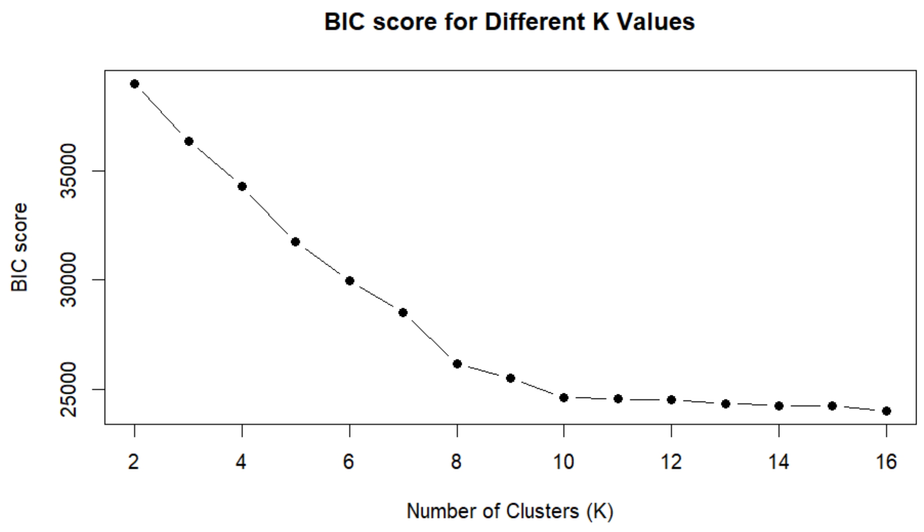

FIGURE 3 – Total variance captured by number of PCA eigenvectors.

5.3 Model Training

The Gaussian Mixture Model (GMM) was trained on the high-dimensional MNIST data. However, the EM algorithm encountered difficulties when inverting the variance-covariance matrix. To address this issue, we performed dimensionality reduction and used different feature sets for clustering :

— PCA-reduced features (50 components).
— t-SNE-reduced features (3 components).

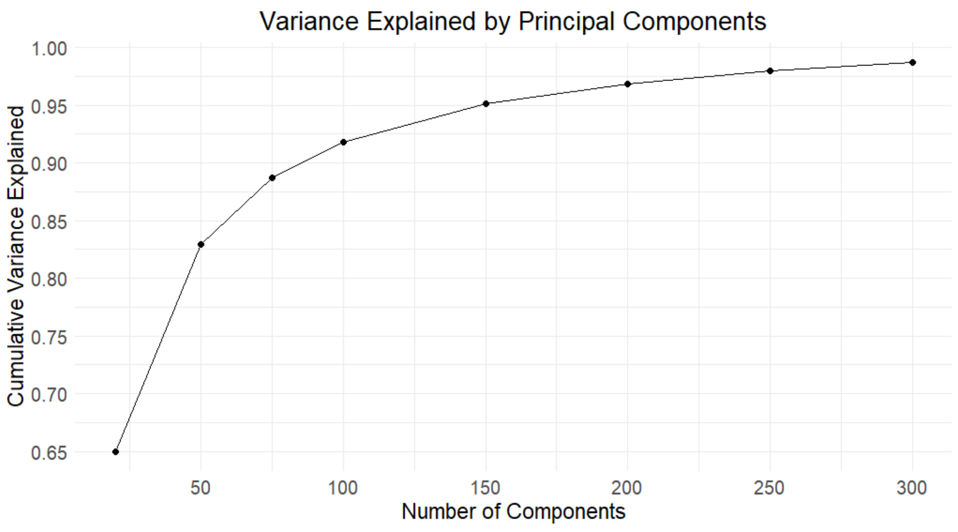

FIGURE 4 – Variance explained by each principal component in PCA.

The Bayesian Information Criterion (BIC) scores were calculated for the EM algorithm applied to the reduced feature sets : - BIC for EM with PCA-reduced data : 570134.2 - BIC for EM with t-SNE-reduced data : 24632.4

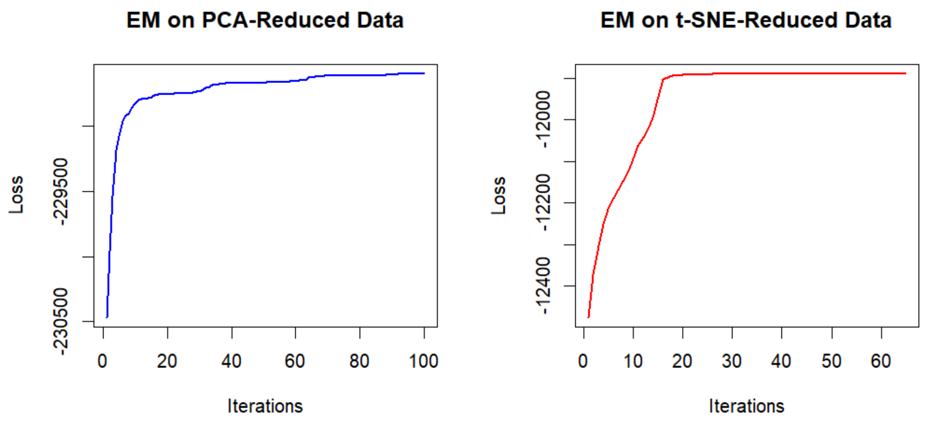

FIGURE 5 – Loss comparison : EM on PCA-reduced data vs EM on t-SNE-reduced data.

Given the significantly lower BIC and much faster convergence with t-SNE-reduced data (3 features), we will focus exclusively on t-SNE-reduced features for the remainder of the analysis. Here, the EM algorithm was initialized using the k-means clustering method and configured to run for a maximum of 100 iterations, with early stopping based on a predefined loss threshold. The number of clusters, K, was set to 10, corresponding to the 10 classes in the MNIST dataset.

5.4 Clustering Results

The training accuracy for the clustering results is as follows :

- Training Accuracy for t-SNE-reduced data : 89.76% - Training Accuracy for PCA-reduced data : 52.78%

	Label 1	**Label 2**	**Label 3**	**Label 4**	**Label 5**	**Label 6**	**Label 7**	**Label 8**	**Label 9**	**Label 10**
Cluster 1	469	0	2	0	0	0	4	0	3	2
Cluster 2	1	554	5	3	7	3	2	10	8	3
Cluster 3	0	3	456	4	1	1	0	1	1	0
Cluster 4	1	1	4	460	0	31	0	0	18	6
Cluster 5	0	0	0	0	307	0	0	5	0	1
Cluster 6	4	1	5	9	5	379	3	2	24	2
Cluster 7	4	0	2	2	6	8	492	0	2	2
Cluster 8	0	1	9	5	1	0	0	499	0	11
Cluster 9	0	2	4	7	0	2	0	0	404	0
Cluster 10	0	1	1	3	208	10	0	33	2	468

TABLE 2 – Confusion Matrix for t-SNE reduced data

The confusion matrix reveals that the majority of the images have been accurately classified, demonstrating that the EM algorithm performs effectively with the t-SNE-reduced data. The results further validate the algorithm's robustness and performance.

5.5 Testing with Different Parameters

Next, we explore the impact of varying the number of initializations on the model's convergence behavior.

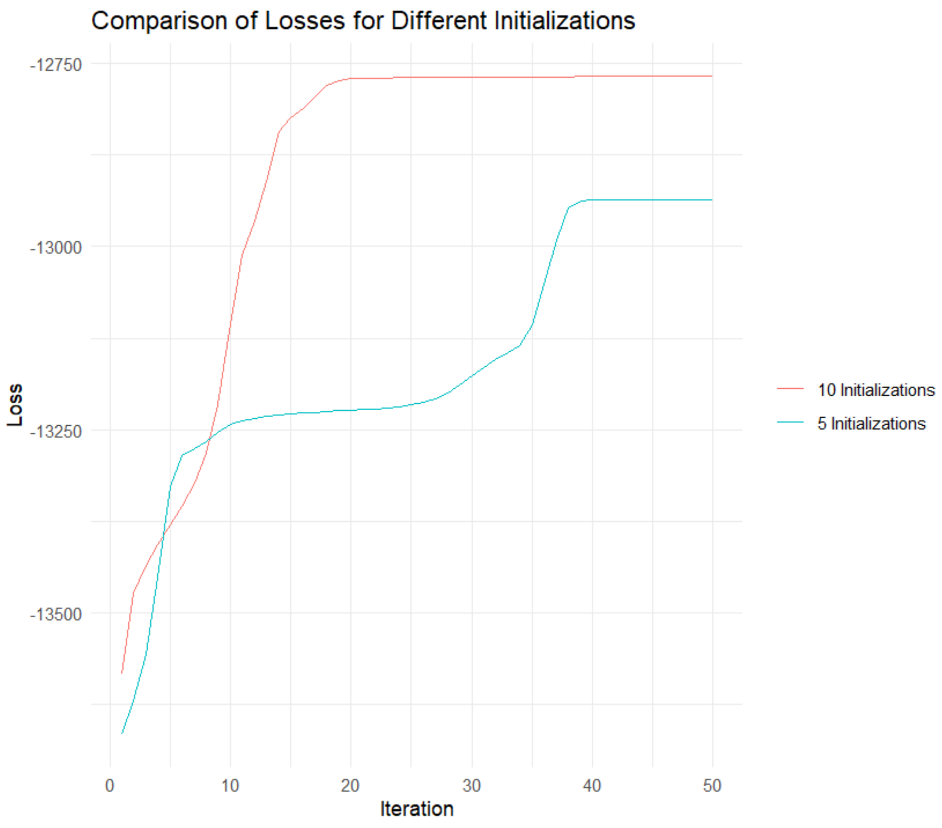

FIGURE 6 – Effect of different numbers of initializations on convergence.

As shown in the figure, the number of initializations significantly affects the convergence speed and performance. A larger number of initializations can lead to faster convergence and improved performance for the same model, highlighting the importance of carefully selecting this parameter to optimize the algorithm's efficiency.

6 Conclusion

This work applied the Expectation-Maximization (EM) algorithm to cluster the MNIST dataset using dimensionality reduction techniques such as PCA and t-SNE. The results demonstrated that t-SNE-reduced data led to faster convergence and higher clustering accuracy compared to PCA, with a training accuracy of 89.76%.

The EM algorithm, when combined with appropriate dimensionality reduction, proved to be effective in clustering high-dimensional data. Future work could involve experimenting with different clustering algorithms and dimensionality reduction methods to further enhance model performance.