• Cover page with the project title and the names of the group members with their student IDs

anyone

Image Segmentation Using K-means Clustering, Gaussian Mixture model and Expectation Maximization

ENCS 6161

Group Members:

40232982 Zunao Hu

40258305 Qian Sun

• Abstract

Sun:

K-Means clustering and Gaussian Mixture Models (GMM) are two popular methods for data clustering. K-Means divides data into K clusters based on distance between data points, while GMM models the data as a combination of Gaussian distributions. Both methods use the Expectation-Maximization (EM) algorithm for parameter estimation. The EM algorithm alternates between an estimation step, which assigns data to clusters or distributions, and a maximization step, which updates the cluster/distribution parameters to better fit the data. Convergence is checked by evaluating the log likelihood, and the process is repeated until convergence is achieved. This paper explores the K-means algorithm and Gaussian Mixture Model (GMM) with Expectation Maximization (EM). We first implement the K-means algorithm for greyscale images, using intensity histograms in one scenario and without in another. Then we implement the K-means algorithm for color image without using intensity histograms. Additionally, we develop a custom function for fitting GMMs and use it to segment color images, comparing the results to those in a provided PDF. We also create our own function for fitting GMMs to data and compare the convergence plot to the PDF. Overall, this study provides a thorough examination of both K-means and GMM-EM, and offers new insights into their capabilities and limitations.

• Table of contents

Anyone:

I. Introduction A. Background information B. Statement of the problem C. Objectives of the study

II. Literature Review A. Previous studies on the topic B. Gaps in the literature

III. Methodology A. Research design B. Data collection methods C. Sampling procedure

IV. Results A. Presentation of findings B. Analysis of data V. Discussion A. Interpretation of results B. Implications of the findings

VI. Conclusion A. Summary of the study B. Recommendations for future research

• List of abbreviations in alphabetical order

Anyone:

Example 1: List of Abbreviations for a Medical Research Paper

AIDS - Acquired Immune Deficiency Syndrome BMI - Body Mass Index CVD - Cardiovascular Disease DM - Diabetes Mellitus HIV - Human Immunodeficien

• List of symbols

Anyone:

List of Symbols for a Physics Thesis

a - Acceleration c - Speed of Light E - Energy F - Force G - Gravitational Constant h - Planck's Constant m - Mass p - Momentum r - Radius v - Velocity

• List of Figures

anyone

a - Acceleration c - Speed of Light E - Energy F - Force G - Gravitational Constant h - Planck's Constant m - Mass p - Momentum r - Radius v - Velocity

• List of Tables

anyone

Table 1: Market Analysis by Product Type Table 2: Sales Forecasts by Region Table 3: Price Comparison of Competitors' Products Table 4: Operating Expenses by Quarter Table 5: Break-Even Analysis Table 6: Cash Flow Projections Table 7: Return on Investment Analysis Table 8: Key Performance Indicators

**1. Introduction**

**Sun**

K-means: The k-means algorithm is a commonly employed traditional clustering method that involves a division approach. Its similarity calculation entails determining the distance between a data object and the cluster center, and then dividing the distance from the cluster center into a cluster. This process is repeated until the criterion function converges. The algorithm's time complexity is O(nkt), where n is the total number of objects, k is the number of clusters, and t is the number of iterations. It is highly efficient; however, it has some limitations. Specifically, it can only handle numeric data, cannot process categorical data, is highly sensitive to exception data, and cannot handle clusters with non-convex shapes.

GMM: The Expectation-Maximization (EM) algorithm is a commonly used tool for estimating the parameters of a Gaussian Mixture Model (GMM). It is an iterative procedure that serves as a maximum-likelihood estimator. Despite its appealing properties, the EM algorithm has some well-documented drawbacks, such as the need for good initial values and the possibility of being trapped in local optima. However, despite these limitations, EM plays an important role in parameter estimation for mixture models.

**2. Scope and objectives of the project**

Zunao

The scope of this project is to perform image segmentation using concepts, ideas, and techniques covered in the course. The project objectives are as follows:

1) Learning the K-means algorithm: The K-means algorithm is a clustering technique used to partition a set of data points into K clusters based on similarity. The project aims to implement the K-means algorithm and its variations to gain proficiency in unsupervised learning techniques.

2) Learning Gaussian Mixture Model (GMM) and Expectation Maximization (EM): Gaussian Mixture Model (GMM) is a statistical model that uses a mixture of Gaussian distributions to represent a given set of data points. Expectation Maximization (EM) is an iterative algorithm used to estimate the parameters of a GMM. The project objective is to learn how to implement GMM and EM algorithms.

3) Implementing Image Segmentation: Image segmentation is the process of dividing an image into multiple segments or regions based on similar characteristics such as color, texture, or intensity. The project aims to apply the K-means algorithm and GMM/EM algorithms learned in objectives 1 and 2 for image segmentation.

4) Fitting 2-D Gaussian Mixture Data: The objective of this project is to learn how to fit 2-D Gaussian mixture data using the GMM and EM algorithms learned in objective 2.

By achieving these objectives, this project aims to provide a comprehensive understanding of the unsupervised learning techniques used in image segmentation and their practical applications, as taught in the course. It also aims to provide hands-on experience in implementing these techniques using Python and relevant libraries such as scikit-learn and OpenCV.

**3. Detailed methodology and implementation**

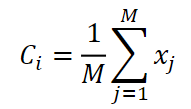
3.1 K-Means methodology: Sun

The k-means algorithm belongs to the partition-based clustering algorithms. It defines the initial centroid value to determine the number of groups [1]. Determining the number of clusters (k) precisely is crucial for the k-means algorithm as the initial cluster center may change, leading to unstable data grouping [2]. The output of the k-means algorithm depends on the selected center values for clustering. The algorithm determines the clusters based on the initial value of the cluster's center point. Randomly assigning the initial cluster centroid can have an impact on the performance of the cluster [3]. k-means is a distance-based clustering algorithm that partitions set data into K clusters. It works well for numerical attributes.

The K-Means algorithm involves the following steps:

1. Determine the number of clusters (K) and the maximum number of iterations.

2. Initialize the K midpoint clusters, followed by counting the feature centroids using the appropriate equation:

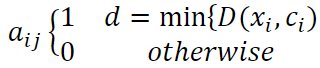


Equation 1 is done as much as p dimensions from i = 1 to i = p

3. Assign each observation data to the nearest cluster using the Euclidean distance metric, which can be calculated using equation 2.

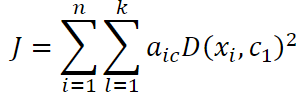


1. The algorithm reallocates the data to each group based on the comparison of the distance between the data and each group's centroid. [4].



5. Recalculate the cluster midpoint position.

The value of the membership of point xi to the centers of group c1 is denoted as aij, d represents the shortest distance from data point xi to group K after comparison, and c1 is the center of group 1. The method uses an objective function that combines the distance and the value of the data membership in the group. The objective function can be determined using the following equation.



In the equation, n represents the amount of data, k represents the number of groups, ai1 denotes the membership value of data point xi to group c1, and a can have a value of either 0 or 1. If the data point belongs to a group, the value of ai1 is 1; otherwise, it is 0.

6. If there is a change in the position of the cluster centroid or if the number of iterations is less than the maximum number of iterations, return to step 3. Otherwise, return the clustering result.

---------------------------------------------------------------------------------------------------

3.2 k-means implementation:

Zunao

The code implementation employs the Python programming language and relevant libraries such as scikit-learn and OpenCV.

1. Grayscale image segmentation with histogram

Necessary libraries such as OpenCV, numpy and matplotlib were imported for image processing, numerical operations and visualization.



Figure 1 Libraries to import for K-Means

The first step is to load a grayscale image and compute its intensity histogram. The histogram is then reshaped into a 2D array of pixel intensities, and a user-defined number of clusters (K) is chosen.



Figure 2 Loading the image and generating histogram

Cluster centroids are initialized using the intensity histogram, and pixels are assigned to clusters based on their intensity values. The centroids are then updated by computing the mean of each cluster.



Figure 3 Assigning pixels and Updating centroids

K-Means clustering was performed on the grayscale image using numpy and OpenCV libraries in Python. The algorithm assigned each pixel to its nearest centroid based on the pixel intensity and updated the centroids until convergence. Finally, the image was segmented and visualized using matplotlib.

Figure 4 Looping till convergence and visualization

1. Grayscale image segmentation without histogram

This implementation performs grayscale image segmentation without using histograms. Instead of using histograms, the pixel values are reshaped into a 2D array and assigned to clusters using the K-Means algorithm.



Figure 5 Loading the image without using histogram

The algorithm iteratively updates the centroids until convergence is achieved.

The segmented image is then reconstructed from the assigned cluster labels and their respective centroids. This approach differs from histogram-based segmentation in that it does not rely on the distribution of pixel intensities, but rather on the clustering of individual pixel values.

1. Color image segmentation without histogram

The RGB image was reshaped into a 2D array of pixels, where each pixel was represented by a vector of its RGB intensities.



The number of clusters was chosen, and cluster centroids were initialized randomly. Pixels were then assigned to clusters based on the Euclidean distance between the pixel vector and each cluster centroid.



The centroids were updated as the mean of the pixels in each cluster, and the process was repeated until convergence or a maximum number of iterations was reached, which was similar to the previous implementations.

Finally, the segmented image was reconstructed by assigning the mean value of each cluster to its corresponding pixels, and the original and segmented images were displayed using Matplotlib.

Compared to the grayscale image segmentation without histogram, the main difference in this code was that the pixel values were represented as 3D vectors instead of 1D vectors, and the distance between a pixel vector and a cluster centroid was calculated as the Euclidean distance instead of the absolute difference between the pixel intensity and the centroid intensity. The update of centroids and segmentation of the image was similar to the grayscale implementation.

3.3 GMM-EM methodology: Sun

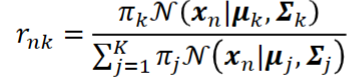
The Gaussian Mixture Model (GMM) is a probabilistic model that uses a combination of Gaussian (Normal) probability distributions to estimate the density of the data. In GMM, the parameters of the Gaussian distributions, such as mean and standard deviation, are estimated using the Expectation-Maximization (EM) algorithm. The EM algorithm consists of two steps: the estimation step and the maximization step. In the estimation step, the algorithm assigns a probability value to each data point to indicate the likelihood of it belonging to a particular cluster. In the maximization step, the algorithm optimizes the parameters of the probability distributions to best capture the density of the data. This iterative process continues until a good set of latent values and a maximum likelihood that fit the data are achieved.

1. Initializing the clusters

Foremost, we must select how many segments we'd like to partition our data into. After selecting k segments to partition the data into, we initialize random Gaussian models.

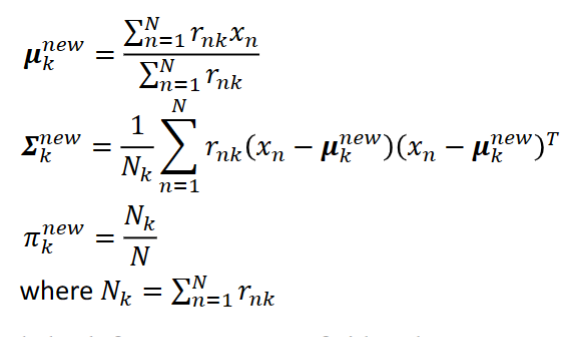
2. Probabalistic assignment to clusters (Expectation)

After initializing k random Gaussian models we can calculate our expectation of responsibilities rnk, a vector of probabilities that xi belongs to the kth clusters for k=1 to k=k.

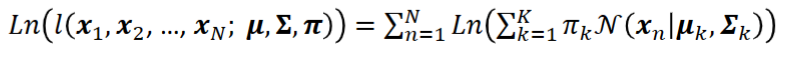


3. Reformulating the Gaussian models (Maximization)

We'll then recalculate our Gaussian models leveraging the weights we found in the expectation step. The expectation, rnk, represents the likelihood that the ith observation belongs to cluster kth.



4.Evaluate the log likelihood and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.



--------------------------------------------------------------------------------------------------------------

3.4 GMM-EM implementation: Zunao

(1) Fit function

The implemented code of the fit function using GMM and EM incorporates the "gaussian\_pdf()" function, which is responsible for computing the probability density function of a multivariate Gaussian distribution. To ensure invertibility, a small positive constant has been added to the diagonal of the covariance matrix. Furthermore, the "np.nan\_to\_num()" function has been employed to replace any nan or inf values in the output array with 0.

In addition, a 'GMM' class was created to perform Gaussian mixture modeling. It includes methods to compute the likelihood and posterior probability of each component given a set of data points, as well as to predict the most likely component for each data point.



The function “fit\_gmm()” was defined to fit a Gaussian Mixture Model (GMM) to data using the Expectation-Maximization (EM) algorithm.



The steps involved in this algorithm are:

1) Initialization: The means and covariances of the Gaussian components and mixing coefficients are initialized. If no initial values are provided, the means are randomly selected from the data, and the covariances are initialized as diagonal matrices with random variances.



2) E-step: The responsibilities of each component are computed for each data point based on the current parameter estimates. The likelihood of each data point given each component is computed, and then, the posterior probability of each component given each data point is calculated. Finally, the log-likelihood of the data is computed and compared to the previous iteration to check for convergence.



3) M-step: The parameters of the model are updated using the responsibilities computed in the E-step. The mixing coefficients, means, and covariances of each component are updated based on the total responsibility of each component.



4) Repeat steps 2 and 3 until convergence or maximum iterations are reached.

5) A Gaussian mixture model object is created with the final parameter estimates, and it is returned as the output of the function.



(2) Grayscale image segmentation using GMM and EM.

This code segment implements grayscale image segmentation using Gaussian Mixture Model (GMM) and Expectation Maximization (EM) algorithm. The input grayscale image is first reshaped into a large vector and then the GMM is fitted to this vector data using the “fit\_gmm()” function. The GMM is initialized with 4 clusters and initial means and covariances.



Once the GMM is fitted, cluster labels are assigned to each pixel in the image, and the pixel intensities are mapped to the mean values of the clusters. Finally, the segmented image is displayed and saved with a filename that includes the number of clusters used.



(3) Color image segmentation using GMM and EM

Color image segmentation using GMM and EM started with loading a color image using OpenCV's “imread()” function. The image was then reshaped into a large vector, with each pixel being represented by three values (RGB). The means and covariances for GMM were initialized using numpy arrays.



The GMM was then fit to the image vector data using the “fit\_gmm()” function. The cluster labels were assigned to pixels in the image, and the mean values of clusters were mapped to pixel intensities. The resulting segmented image was displayed using OpenCV's “imshow()” function, and saved using “imwrite()” function. The code can be easily modified to experiment with different values of K and initialization parameters.

(4) 2D dataset segmentation using GMM and EM

First, a 2D dataset was generated using numpy's “random.multivariate\_normal()” function. The dataset consisted of 4 clusters, each with a different mean and covariance matrix. The means and covariances were defined as follows:



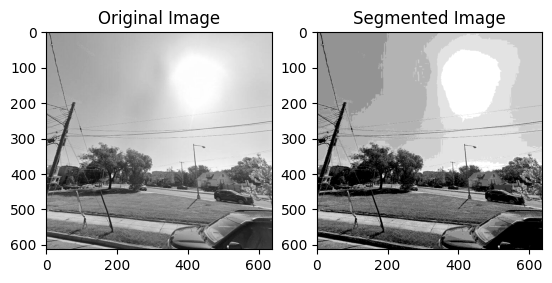
The dataset was generated by stacking 200 samples from each cluster, except for the fourth cluster which only had 80 samples. The resulting dataset x was then used to fit a GMM using the “fit\_gmm()” function. The GMM was specified to have 4 components using the “n\_components” argument.

After fitting the GMM, the “predict()” method was used to assign cluster labels to each data point in the dataset x. These labels were then used to plot the original signal distribution and the segmented signal distribution. In the segmented signal distribution plot, each cluster was assigned a different color for visualization purposes.

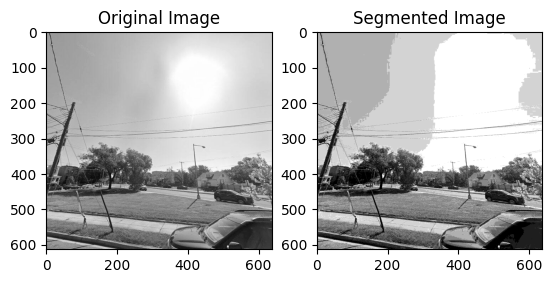


**4. Experimental results**

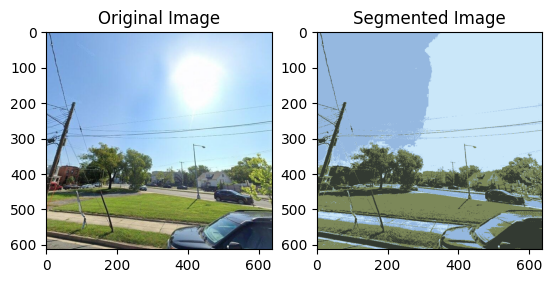
(1) Grayscale image segmentation using K-Means with histogram



(2) Grayscale image segmentation using K-Means without histogram



(3) Color image segmentation using K-Means without histogram



(4) Grayscale image segmentation using GMM and EM

Our segmentation results:

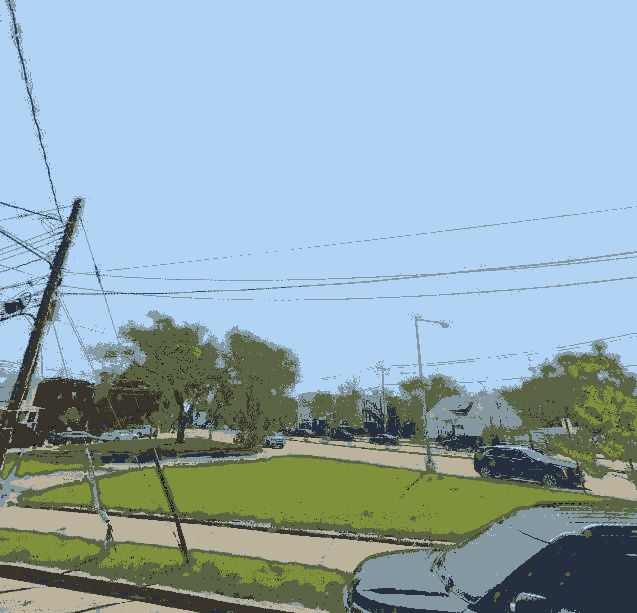


The segmentation result using gmm.fit():

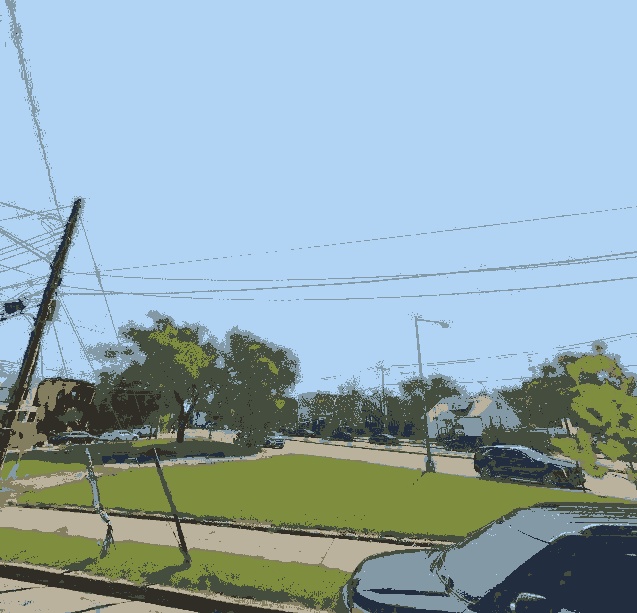


(5) Color image segmentation using GMM and EM

Our segmentation result:

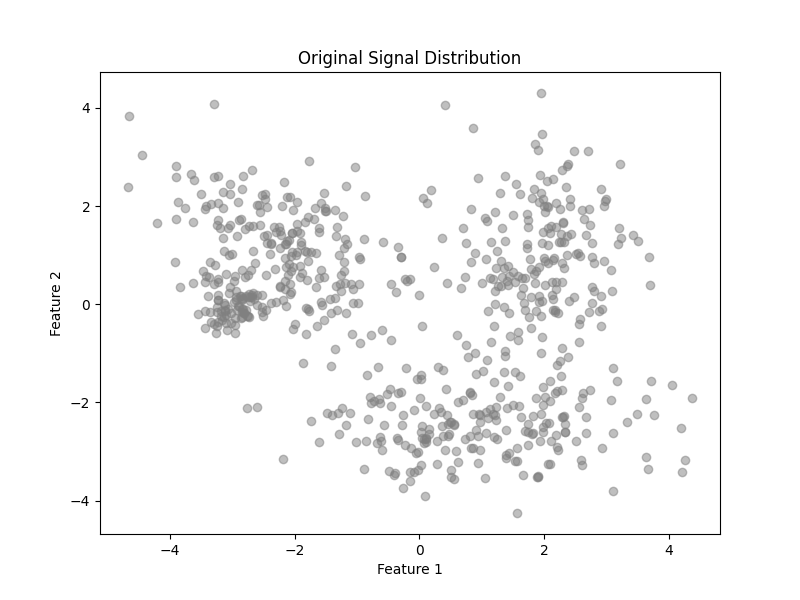


The segmentation result using gmm.fit():

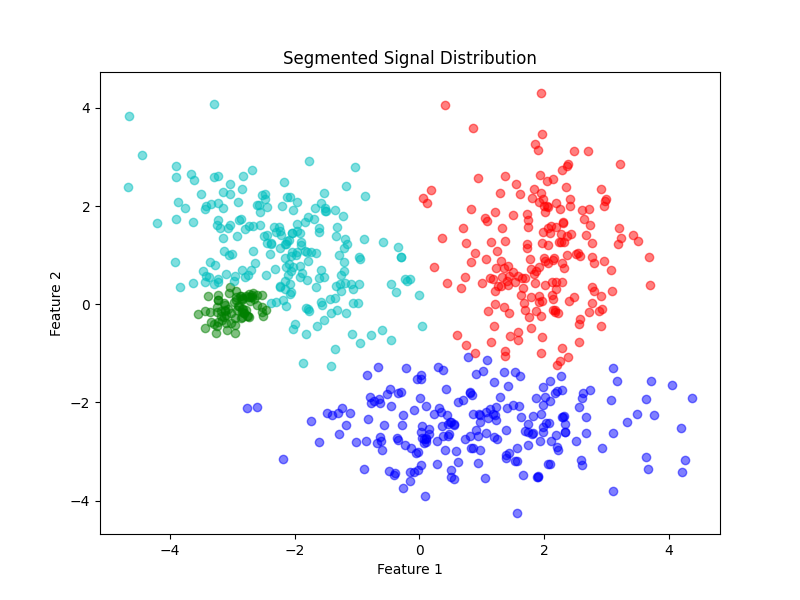


(6) 2D dataset segmentation using GMM and EM

The original dataset distribution:



The segmented distribution:



**5. Conclusion**

Zunao

In conclusion, this report has demonstrated the implementation of k-means and GMM/EM algorithms for image segmentation on grayscale and color images, as well as 2D datasets. The obtained results suggest the effectiveness of these techniques in clustering pixels, which can contribute to the development of computer vision applications.

The report's results are promising, but there are potential limitations to consider. The experiments were conducted only on a single image, and the performance of these techniques could vary with more complex images. In addition, the algorithms require the selection of the appropriate number of clusters, which can be challenging in some cases. These limitations suggest the need for further research to explore the use of these techniques in a wider range of scenarios and to investigate possible modifications or extensions to overcome these challenges.

**References**

Sun

Zunao

[1] Eltibi M F and Ashour W M 2011 Initializing K-Means Clustering Algorithm using Statistical Information. K-means clustering algorithm is one of the best known, XXIX 7 p51

[2] Aristidis L, Nikos V, Jacob J V 2011 The global k-means Clustering algorithm IAS technical report series, IAS-UVA-01-02

[3] Cosmin M P, Marian C M, Mihai M An Optimized Version of the K-Means Clustering Algorithm, Proceedings of the 2014 Federated Conference on Computer Science and Information Systems (ACSIS) 2 p695

[4] Joshi K D and Nalwade P S 2013 Modified K-Means for Better Initial Cluster Centres. International Journal of Computer Science and Mobile Computing II7 p2

**Appendices**

Zunao