

ULVQ : Unsupervised Learning Vector Quantization for clustering

Sami RAJICHI
semi.rajichi@gmail.com

Institut Supérieur d'Informatique de Mahdia — November 14, 2023

Abstract

Clustering, an essential task in unsupervised learning, plays a pivotal role in organizing and understanding complex datasets. This report introduces Unsupervised Learning Vector Quantization (ULVQ), a variant of the well-known LVQ algorithm adapted for clustering purposes. ULVQ is explored as a method for grouping data points into clusters without relying on predefined labels. The methodology involves initializing cluster centroids, iteratively updating them through a learning process, and employing diverse cluster labeling techniques. This report delves into the implementation details, including algorithmic considerations and parameter choices.

Through a series of experiments on a selected dataset, the report evaluates ULVQ's performance in comparison to traditional clustering algorithms. Various clustering evaluation metrics, such as V-Measure, Silhouette Score, Davies-Bouldin Score, and Calinski-Harabasz Score, are employed to assess the quality of the clustering results.

The results and visualizations provide insights into ULVQ's ability to uncover meaningful patterns within the data. The discussion section interprets the clustering results, highlighting the impact of different parameters and clustering methods. Additionally, the report explores potential applications and scenarios where ULVQ could excel.

In conclusion, this report offers a comprehensive exploration of ULVQ as a powerful tool for unsupervised clustering. The findings contribute to the understanding of ULVQ's strengths, limitations, and its comparative performance against traditional clustering algorithms.

1 Introduction

1.1 Overview of Unsupervised Learning Vector Quantization (ULVQ)

Clustering, a fundamental task in unsupervised learning, involves organizing data points into coherent groups. Unsupervised Learning Vector Quantization (ULVQ) is a variant of the well-established LVQ algorithm, tailored for unsupervised clustering. ULVQ offers a mechanism for grouping data points into clusters without relying on predefined labels.

1.1.1 Key Steps

ULVQ's methodology comprises several key steps:

- 1. Initialization of Cluster Centroids:** The algorithm begins by initializing cluster centroids, typically using a random selection of data points.
- 2. Iterative Learning Updates:** ULVQ iteratively updates these centroids through a learning process. For each data point, the algorithm identifies the closest cluster centroid and adjusts it based on the learning rate.
- 3. Cluster Labeling Techniques:** ULVQ incorporates various cluster labeling techniques, such as K-Means, Hierarchical Clustering, DBSCAN, GMM, and Agglomerative Clustering, to assign labels to the clusters.

1.1.2 Implementation Details

The implementation of ULVQ involves considerations such as algorithmic intricacies and parameter choices. These details significantly influence the clustering performance and the algorithm's ability to uncover meaningful patterns in the data.

1.1.3 Clustering Evaluation Metrics

To assess the quality of ULVQ's clustering results, a set of clustering evaluation metrics is employed. Common metrics include V-Measure, Silhouette Score, Davies-Bouldin Score, and Calinski-Harabasz Score.

1.1.4 Experiments and Results

Experiments are conducted on a chosen dataset, and the results, along with visualizations, provide insights into ULVQ's ability to reveal patterns within the data. The discussion interprets these findings, emphasizing the impact of different parameters and clustering methods.

1.1.5 Conclusion

In conclusion, this report aims to provide a comprehensive exploration of ULVQ, shedding light on its strengths, limitations, and comparative performance against traditional clustering algorithms. The findings contribute to a better understanding of ULVQ's potential applications in unsupervised learning scenarios.

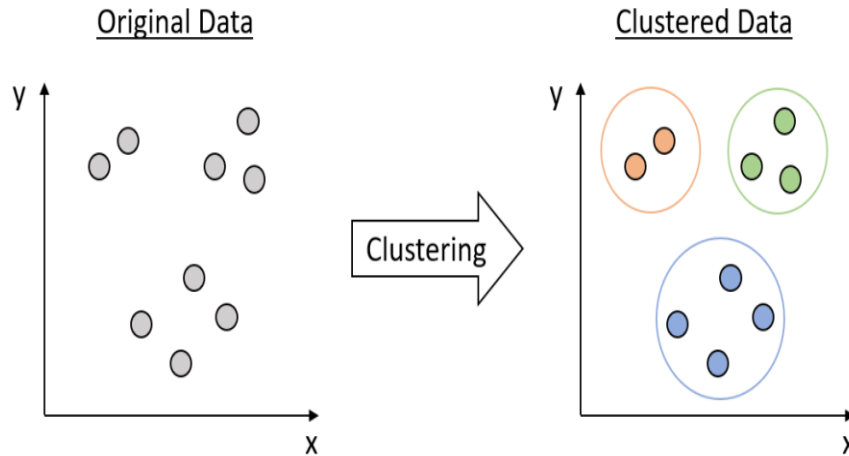


Figure 1: Clustering Principle

2 Motivation and Methodology

2.1 Brief Explanation of Motivation

Clustering serves as a crucial unsupervised learning task, aiming to uncover inherent structures within datasets. The motivation behind utilizing Unsupervised Learning Vector Quantization (ULVQ) for clustering lies in its adaptability and effectiveness in organizing data points into meaningful clusters without the need for labeled examples.

Traditional clustering algorithms often lack the ability to adapt dynamically to the underlying data distribution. ULVQ addresses this limitation by leveraging the principles of competitive learning and self-organization, enabling it to uncover complex patterns and structures within the dataset.

2.2 Methodology

ULVQ's methodology encompasses key steps that make it a powerful tool for unsupervised clustering:

1. **Adaptive Centroid Initialization:** ULVQ initiates the clustering process by adaptively initializing cluster centroids, allowing for a more representative starting point.
2. **Dynamic Learning Updates:** Through iterative learning updates (epochs), ULVQ dynamically adjusts cluster centroids based on the distribution of data points, leading to improved convergence and better representation of clusters.
3. **Diverse Cluster Labeling Techniques:** ULVQ incorporates various cluster labeling techniques, providing flexibility in assigning labels to clusters. This adaptability enhances its performance across different types of datasets.

ULVQ's motivation to dynamically adapt to data characteristics and its flexible methodology make it a compelling choice for unsupervised clustering tasks, particularly when dealing with complex and diverse datasets.

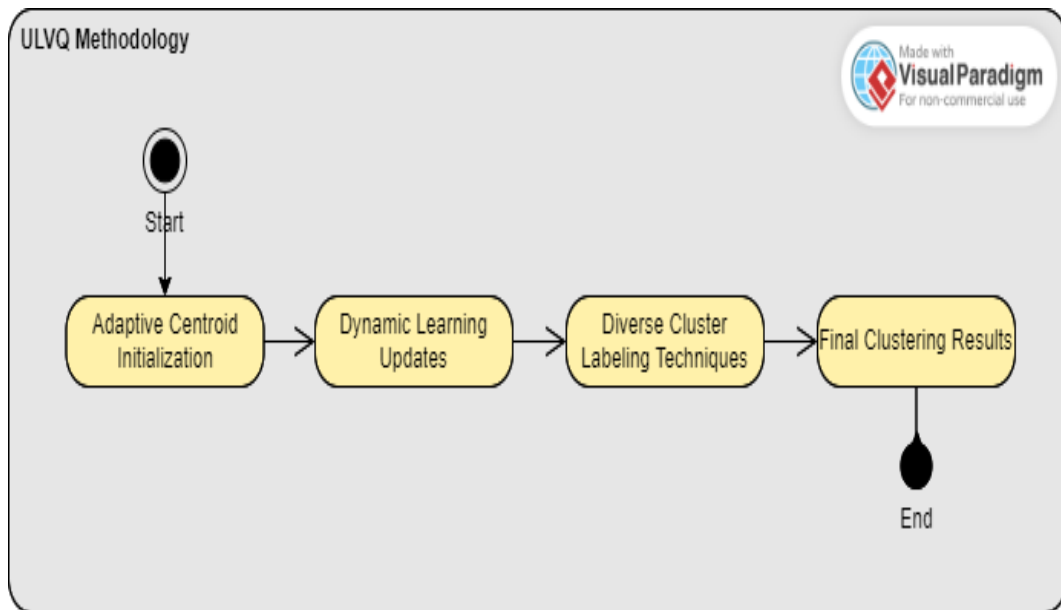


Figure 2: ULVQ Methodology

3 ULVQ Process Cycle

The Unsupervised Learning Vector Quantization (ULVQ) algorithm is designed for unsupervised clustering, aiming to organize data points into coherent clusters without relying on predefined labels.

Initialization of Cluster Centroids

The algorithm begins by adaptively initializing cluster centroids. This process ensures that the starting points for each cluster are representative of the underlying data distribution. The adaptive initialization contributes to the algorithm's ability to converge effectively and capture diverse patterns within the dataset.

Training Process and Update Rules

ULVQ employs a dynamic training process with iterative updates to the cluster centroids. The key steps in the training process include:

1. **Calculate Distances:** For each data point, calculate the distances to all cluster centroids.

2. **Find the Winner:** Identify the closest cluster centroid, known as the winner.
3. **Update Winner:** Adjust the winner's centroid based on the learning rate and the difference between the data point and the current centroid.
4. **Iterative Updates:** Repeat the process iteratively for all data points, refining the cluster centroids.

The iterative learning updates contribute to the self-organization and competitive learning aspects of ULVQ, allowing it to adapt to the dataset's inherent characteristics.

Cluster Labeling Methods

ULVQ incorporates diverse cluster labeling techniques to assign labels to the discovered clusters. These techniques provide flexibility in cluster interpretation. Common cluster labeling methods include:

- **K-Means:** Assigns each cluster a label based on the majority of data points within the cluster.
- **Hierarchical Clustering:** Labels clusters based on the hierarchical structure of the clustering dendrogram.
- **DBSCAN:** Applies density-based labels, distinguishing between core points, border points, and noise.
- **GMM (Gaussian Mixture Model):** Assigns labels using a probabilistic model, considering the likelihood of data points belonging to each cluster.
- **Agglomerative Clustering:** Labels clusters based on the agglomeration process, where clusters are successively merged.

These diverse labeling methods accommodate different data characteristics and provide insights into the nature of the discovered clusters.

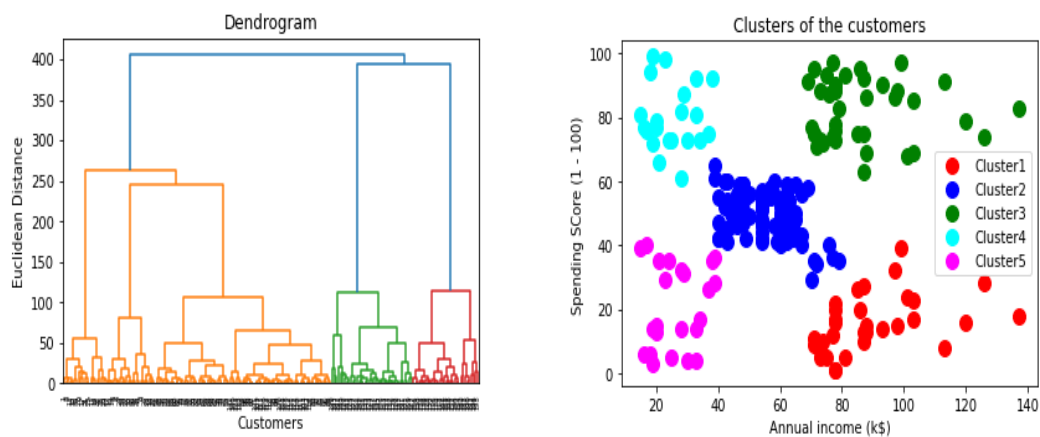


Figure 3: Flat and Hierarchical Clustering

Labeling Technique	Technical Description	Application in ULVQ
K-Means	Assigns each cluster a label based on the mean (centroid) of data points within the cluster. Utilizes Euclidean distance to determine cluster membership.	Provides efficient and simple labels based on cluster centroids. Effective for well-separated, spherical clusters.
Hierarchical Clustering	Forms a hierarchy of clusters using linkage methods (e.g., Ward, complete, average). Labels are assigned based on the hierarchical tree structure, allowing for various levels of granularity.	Captures hierarchical relationships among clusters, providing insights into cluster groupings at different scales.
DBSCAN	Labels clusters based on density-connected components. Core points, border points, and noise points are identified using parameters like epsilon (neighborhood radius) and minimum points.	Suitable for clusters of varying shapes and densities. Effectively handles noise points and identifies dense regions.
GMM (Gaussian Mixture Model)	Represents clusters as Gaussian distributions. Labels are assigned probabilistically based on the likelihood of data points belonging to each cluster.	Accommodates clusters with complex shapes and overlapping regions. Provides soft assignments, indicating uncertainty in cluster membership.
Agglomerative Clustering	Iteratively merges clusters based on linkage criteria (e.g., Ward, complete, average). Labels are assigned as the result of the merging process, forming a hierarchy of clusters.	Reveals hierarchical relationships among clusters and provides a tree-like structure of cluster formation.

Table 1: Technical Description of Cluster Labeling Techniques in ULVQ

Technical Illustration of Cluster Labeling Techniques in ULVQ

K-Means Clustering

Functionality: K-means clustering assigns each data point to the cluster with the closest mean. The algorithm iteratively updates the cluster means and assigns data points to the closest cluster until convergence.

```

1 # Pseudo Code
2 Initialize the cluster centroids
3 Repeat until convergence:
4     Assign each data point to the cluster with the closest centroid
5     Update the cluster centroids to be the mean of all the data points assigned to
      that cluster

```

Source Code 1: KMeans

Hierarchical Clustering

Functionality: Hierarchical clustering creates a hierarchy of clusters, where each cluster is nested within another cluster. The algorithm iteratively merges the two closest clusters until a single cluster remains.

```

1 # Pseudo Code
2 Initialize each data point as its own cluster
3 Repeat until a single cluster remains:

```

```

4 Calculate the distance between each pair of clusters
5 Merge the two closest clusters

```

Source Code 2: Hierarchical Clustering

DBSCAN Clustering

Functionality: DBSCAN clustering groups data points based on their density. The algorithm identifies core points, which are data points with a minimum number of neighbors within a certain distance.

```

1 # Pseudo Code
2 Initialize each data point as unclustered
3 For each data point:
4     If the data point has at least MinPts neighbors within Eps distance:
5         Mark the data point as a core point
6         Create a new cluster
7         Add the core point to the cluster
8         Add all neighbors of the core point to the cluster
9         Repeat until no more new data points can be added to the cluster

```

Source Code 3: DBSCAN Clustering

Gaussian Mixture Model (GMM)

Functionality: GMM clustering assumes that the data points are drawn from a mixture of Gaussian distributions. The algorithm iteratively updates the parameters of the Gaussian mixture model to maximize the likelihood of the data.

```

1 # Pseudo Code
2 Initialize the parameters of the Gaussian mixture model
3 Repeat until convergence:
4     Calculate the posterior probability of each data point belonging to each Gaussian
      distribution
5     Assign each data point to the Gaussian distribution with the highest posterior
      probability
6     Update the parameters of the Gaussian mixture model to maximize the likelihood of
      the data

```

Source Code 4: Gaussian Mixture Model (GMM)

Agglomerative Clustering

Functionality: Agglomerative clustering is a type of hierarchical clustering algorithm that iteratively merges the two closest clusters. The algorithm uses a linkage criterion to measure the distance between clusters.

```

1 # Pseudo Code
2 Initialize each data point as its own cluster
3 Calculate the distance between each pair of clusters using the linkage criterion
4 Repeat until a single cluster remains:
5     Merge the two closest clusters

```

Source Code 5: Agglomerative Clustering

4 Implementation Details

4.1 Overview of the Implemented ULVQ Class

The Unsupervised Learning Vector Quantization (ULVQ) class is designed to perform clustering without the need for a target variable. It leverages the principles of LVQ for unsupervised learning.

`__init__`

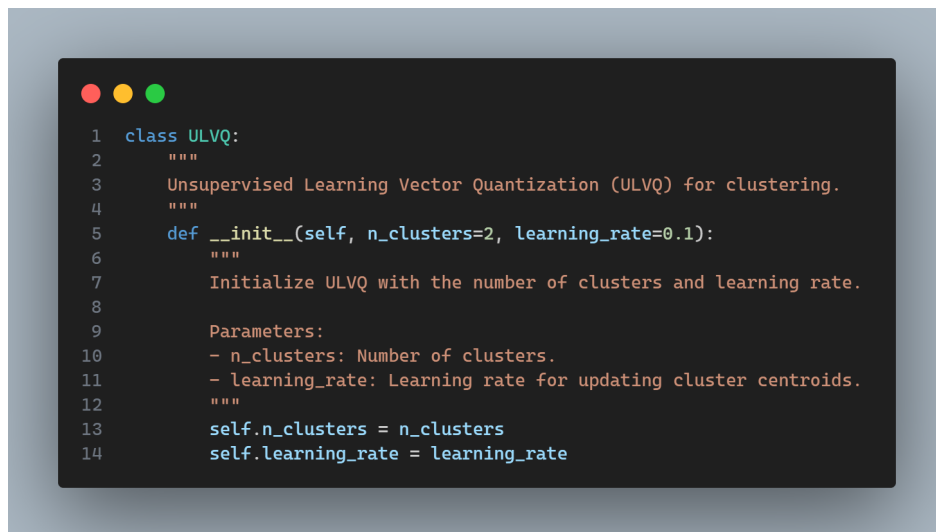
Initialize ULVQ with the number of clusters and learning rate.

- **Parameters:**

- `n_clusters`: Number of clusters.
- `learning_rate`: Learning rate for updating cluster centroids.

- **Description:**

- This method is responsible for initializing the ULVQ object with user-specified parameters, namely the number of clusters and learning rate.



```
1 class ULVQ:
2     """
3     Unsupervised Learning Vector Quantization (ULVQ) for clustering.
4     """
5     def __init__(self, n_clusters=2, learning_rate=0.1):
6         """
7         Initialize ULVQ with the number of clusters and learning rate.
8
9         Parameters:
10        - n_clusters: Number of clusters.
11        - learning_rate: Learning rate for updating cluster centroids.
12        """
13        self.n_clusters = n_clusters
14        self.learning_rate = learning_rate
```

Figure 4: Initialization Method

`fit`

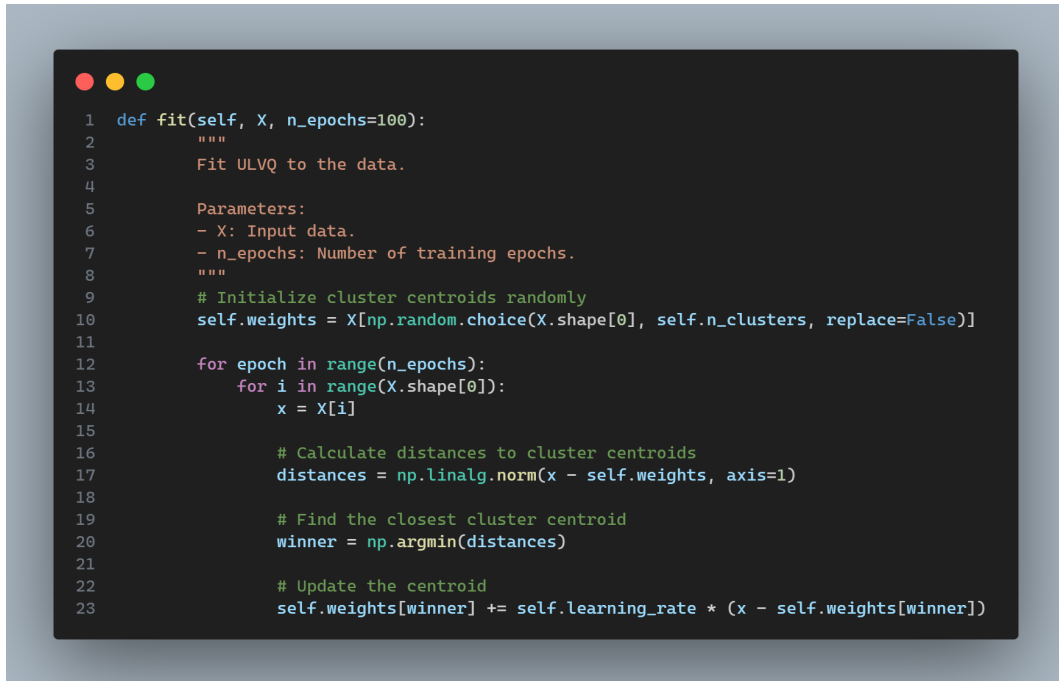
Fit ULVQ to the data.

- **Parameters:**

- `X`: Input data.
- `n_epochs`: Number of training epochs.

- **Description:**

- This method trains the ULVQ model on the input data (`X`) for a specified number of epochs (`n_epochs`).
- It initializes cluster centroids randomly and updates them iteratively based on the input data.



```

1  def fit(self, X, n_epochs=100):
2      """
3          Fit ULVQ to the data.
4
5          Parameters:
6          - X: Input data.
7          - n_epochs: Number of training epochs.
8      """
9      # Initialize cluster centroids randomly
10     self.weights = X[np.random.choice(X.shape[0], self.n_clusters, replace=False)]
11
12     for epoch in range(n_epochs):
13         for i in range(X.shape[0]):
14             x = X[i]
15
16             # Calculate distances to cluster centroids
17             distances = np.linalg.norm(x - self.weights, axis=1)
18
19             # Find the closest cluster centroid
20             winner = np.argmin(distances)
21
22             # Update the centroid
23             self.weights[winner] += self.learning_rate * (x - self.weights[winner])

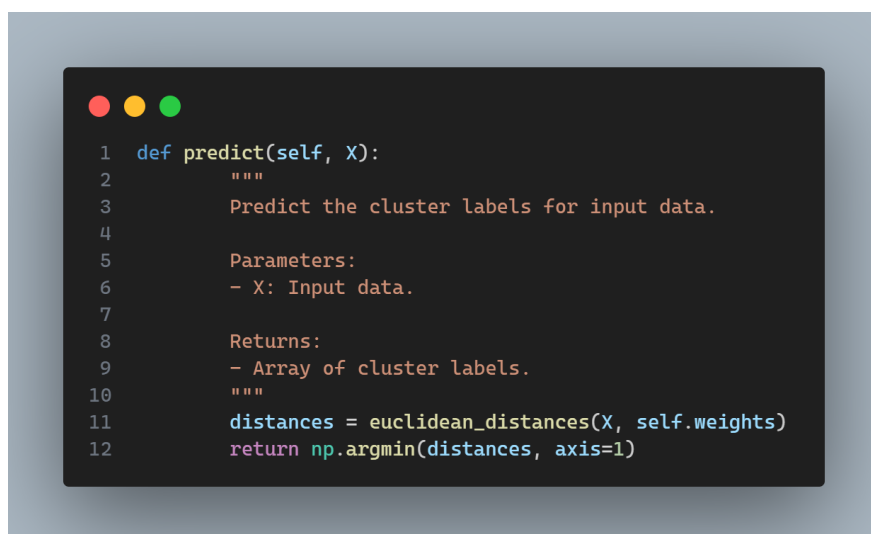
```

Figure 5: Fit Method

predict

Predict the cluster labels for input data.

- **Parameters:**
 - X: Input data.
- **Returns:**
 - Array of cluster labels.
- **Description:**
 - This method predicts cluster labels for input data based on the trained ULVQ model.
 - It calculates the distances between data points and cluster centroids, assigning each point to the closest cluster.



```

1  def predict(self, X):
2      """
3          Predict the cluster labels for input data.
4
5          Parameters:
6          - X: Input data.
7
8          Returns:
9          - Array of cluster labels.
10     """
11     distances = euclidean_distances(X, self.weights)
12     return np.argmin(distances, axis=1)

```

Figure 6: Predict Method

label_clusters

Label clusters using a specified clustering method.

- **Parameters:**

- X: Input data.
- method: Clustering method ('kmeans', 'hierarchical', 'dbscan', 'gmm', 'agglomerative').

- **Returns:**

- Array of cluster labels.

- **Description:**

- This method labels clusters using a specified clustering algorithm (method).
- It supports various clustering methods, such as K-Means, Hierarchical Clustering, DBSCAN, GMM, and Agglomerative Clustering.

```
1 def label_clusters(self, X, method='kmeans'):
2     """
3     Label clusters using a specified clustering method.
4
5     Parameters:
6     - X: Input data.
7     - method: Clustering method ('kmeans', 'hierarchical', 'dbscan', 'gmm', 'agglomerative').
8
9     Returns:
10    - Array of cluster labels.
11    """
12    if method == 'kmeans':
13        kmeans = KMeans(n_clusters=self.n_clusters)
14        return kmeans.fit_predict(X)
15    elif method == 'hierarchical':
16        linkage_matrix = linkage(X, method='ward')
17        dendrogram_result = dendrogram(linkage_matrix, truncate_mode='lastp')
18        cluster_labels = fcluster(linkage_matrix, t=0.8, criterion='distance')
19        return cluster_labels - 1 # Adjust labels to start from 0
20    elif method == 'dbscan':
21        dbscan = DBSCAN(eps=0.5, min_samples=5)
22        return dbscan.fit_predict(X)
23    elif method == 'gmm':
24        gmm = GaussianMixture(n_components=self.n_clusters)
25        return gmm.fit_predict(X)
26    elif method == 'agglomerative':
27        agglomerative = AgglomerativeClustering(n_clusters=self.n_clusters)
28        return agglomerative.fit_predict(X)
29    else:
30        raise ValueError("Invalid clustering method. Supported methods: 'kmeans', 'hierarchical', 'dbscan', 'gmm', 'agglomerative'")
```

Figure 7: Label Clusters Method

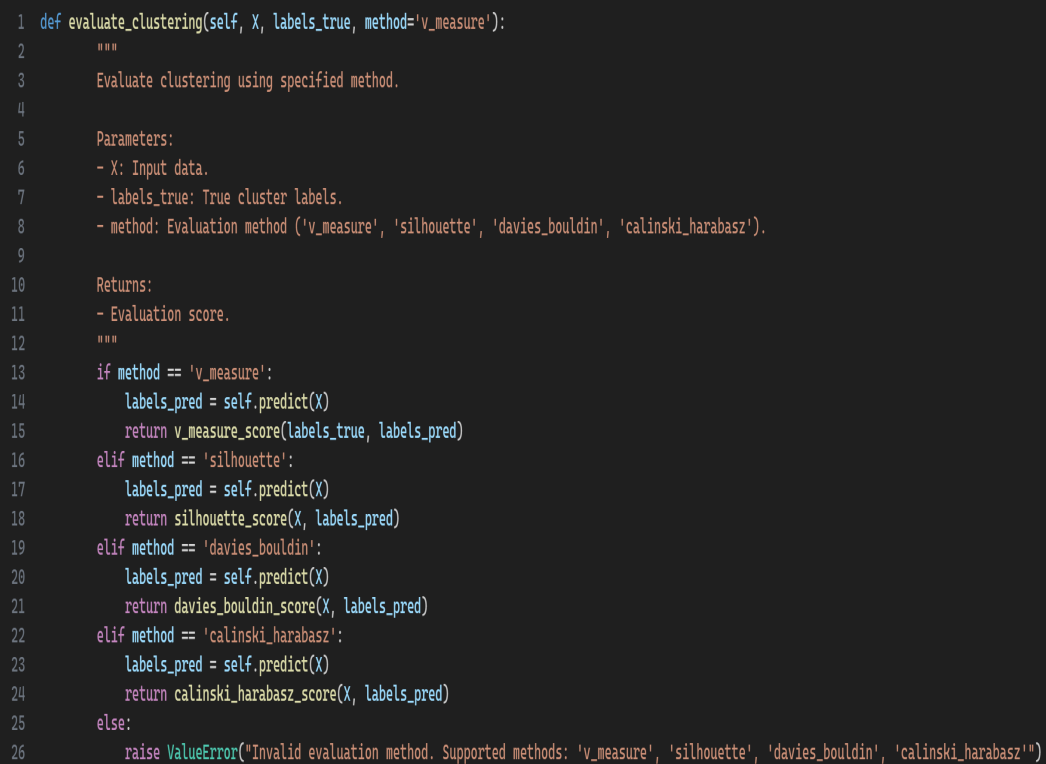
evaluate_clustering

Evaluate clustering using specified method.

- **Parameters:**

- X: Input data.

- labels_true: True cluster labels.
- method: Evaluation method ('v_measure', 'silhouette', 'davies_bouldin', 'calinski_harabasz').
- **Returns:**
 - Evaluation score.
- **Description:**
 - This method evaluates clustering performance using a specified evaluation method (method).
 - It supports various evaluation metrics such as V-Measure, Silhouette Score, Davies-Bouldin Score, and Calinski-Harabasz Score.



```

1 def evaluate_clustering(self, X, labels_true, method='v_measure'):
2     """
3     Evaluate clustering using specified method.
4
5     Parameters:
6     - X: Input data.
7     - labels_true: True cluster labels.
8     - method: Evaluation method ('v_measure', 'silhouette', 'davies_bouldin', 'calinski_harabasz').
9
10    Returns:
11    - Evaluation score.
12    """
13    if method == 'v_measure':
14        labels_pred = self.predict(X)
15        return v_measure_score(labels_true, labels_pred)
16    elif method == 'silhouette':
17        labels_pred = self.predict(X)
18        return silhouette_score(X, labels_pred)
19    elif method == 'davies_bouldin':
20        labels_pred = self.predict(X)
21        return davies_bouldin_score(X, labels_pred)
22    elif method == 'calinski_harabasz':
23        labels_pred = self.predict(X)
24        return calinski_harabasz_score(X, labels_pred)
25    else:
26        raise ValueError("Invalid evaluation method. Supported methods: 'v_measure', 'silhouette', 'davies_bouldin', 'calinski_harabasz'")

```

Figure 8: Evaluate Clustering Method

visualize_clusters

Visualize clustering results with true labels and clustering labels side by side.

- **Parameters:**
 - X: Input data.
 - labels_true: True cluster labels.
 - labels_pred: Predicted cluster labels.
 - feature_names: Names of features.
 - title: Title for the plot.

- **Description:**

- This method visualizes clustering results by creating a side-by-side comparison of true labels and predicted labels.
- It uses Seaborn for plotting and allows users to inspect how well the model aligns with true clusters.

```

1 def visualize_clusters(self, X, labels_true, labels_pred, feature_names, title="Unsupervised LVQ Clustering"):
2     """
3     Visualize clustering results with true labels and clustering labels side by side.
4
5     Parameters:
6     - X: Input data.
7     - labels_true: True cluster labels.
8     - labels_pred: Predicted cluster labels.
9     - feature_names: Names of features.
10    - title: Title for the plot.
11    """
12    data_true = np.column_stack((X, labels_true))
13    data_pred = np.column_stack((X, labels_pred))
14
15    features = [f for f in feature_names]
16    features.append('True_Label')
17    df_true = pd.DataFrame(data_true, columns=features)
18
19    features[-1] = 'Cluster_Label'
20    df_pred = pd.DataFrame(data_pred, columns=features)
21
22    plt.figure(figsize=(14, 6))
23
24    plt.subplot(1, 2, 1)
25    sns.scatterplot(data=df_true, x=df_true.columns[0], y=df_true.columns[1], hue=df_true.columns[2], palette="Set1", s=100, marker="o")
26    plt.title("True Labels")
27    plt.xlabel(feature_names[0])
28    plt.ylabel(feature_names[1])
29    plt.legend()
30
31    plt.subplot(1, 2, 2)
32    sns.scatterplot(data=df_pred, x=df_pred.columns[0], y=df_pred.columns[1], hue=df_pred.columns[2], palette="Set1", s=100, marker="o")
33    plt.title("Cluster Labels")
34    plt.xlabel(feature_names[0])
35    plt.ylabel(feature_names[1])
36    plt.legend()
37
38    plt.suptitle(title)
39    plt.show()

```

Figure 9: Visualize Clusters Method

4.2 Explanation of Key Parameters

- **n_clusters:** The number of clusters (or prototypes) to be learned by the ULVQ algorithm.
- **learning_rate:** The learning rate controls the step size during the update of cluster centroids in the training process.
- **n_epochs:** The number of training epochs, representing the number of times the algorithm iterates over the entire dataset during training.

4.3 Training Process and Convergence Considerations

The training process of the ULVQ class involves initializing cluster centroids randomly and updating them iteratively based on the input data. The algorithm aims to converge to a stable clustering configuration. Convergence is achieved when the centroids no longer change significantly between iterations or when a predefined number of training epochs is reached.

Training Steps:

1. Initialize cluster centroids randomly.
2. Iterate through the dataset for a specified number of epochs.
3. For each data point, calculate distances to cluster centroids.
4. Find the closest cluster centroid (winner).
5. Update the centroid based on unsupervised learning rules.

Convergence Criteria:

- Monitor changes in cluster centroids between iterations.
- Define a tolerance threshold for centroid updates.
- Set a maximum number of training epochs.

Adjustments to parameters and convergence criteria can be made based on the characteristics of the dataset and desired clustering outcomes.

5 Clustering Evaluation Metrics

5.1 Introduction to Clustering Evaluation Metrics

Clustering evaluation metrics are quantitative measures used to assess the quality and performance of clustering algorithms. These metrics help in gauging how well the clusters created by an algorithm align with some ground truth or desired characteristics. In the context of unsupervised learning, where true class labels are often unknown, these metrics provide valuable insights into the internal coherence and separation of clusters.

5.2 Overview of Clustering Evaluation Metrics

Metric	Description
V-Measure	Measures the balance between homogeneity and completeness of clustering. It computes the harmonic mean of these two scores.
Silhouette Score	Quantifies how well-separated clusters are. It ranges from -1 to 1, where a high value indicates well-defined clusters.
Davies-Bouldin Score	Evaluates the compactness and separation of clusters. A lower score indicates better clustering.
Calinski-Harabasz Score	Measures the ratio of between-cluster variance to within-cluster variance. A higher score implies better-defined clusters.

Table 2: Overview of Clustering Evaluation Metrics

5.3 Explanation of Clustering Evaluation Metrics

- **V-Measure:** The V-Measure is a combination of homogeneity and completeness. Homogeneity measures how well each cluster contains only data points that are members of a single class, while completeness measures how well all data points that are members of a given class are assigned to the same cluster. The V-Measure is the harmonic mean of these two scores, providing a balanced measure of clustering quality.
 - Range: [0, 1]
 - Interpretation:
 - * 0: No similarity between predicted and true labels.

- * 1: Perfect match between predicted and true labels in terms of both homogeneity and completeness.
- **Silhouette Score:** The Silhouette Score evaluates how similar an object is to its own cluster (cohesion) compared to other clusters (separation).
 - Range: [-1, 1]
 - Interpretation:
 - * Close to 1: Well-defined clusters with instances clearly belonging to one cluster.
 - * Close to -1: Overlapping clusters, instances may be assigned to the wrong cluster.
 - * Around 0: Overlapping clusters or poorly defined clusters.
- **Davies-Bouldin Score:** The Davies-Bouldin Score measures the compactness and separation of clusters. It computes the average similarity ratio of each cluster with its most similar cluster.
 - Range: [0, $+\infty$]
 - Interpretation:
 - * Lower values indicate better clustering. A value of 0 indicates perfectly separated clusters.
- **Calinski-Harabasz Score:** The Calinski-Harabasz Score assesses the ratio of between-cluster variance to within-cluster variance. It is a useful metric for evaluating clustering performance, especially when the ground truth is unknown.
 - Interpretation:
 - * Higher values suggest better-defined, well-separated clusters.
 - * Absolute values lack a clear interpretation; comparisons should be made between different clustering results.



Info: The following table shows the evaluation scores obtained for clustering:

V-Measure	Silhouette Score	Davies-Bouldin Index	Calinski-Harabasz Index
0.643923	0.431089	0.723859	171.915681

Interpretation of Clustering Evaluation Scores:

- **V-Measure:** The V-Measure, at 0.64, indicates a reasonably good balance between homogeneity and completeness in the clustering results. It suggests that the clusters are both accurate and complete.
- **Silhouette Score:** The Silhouette Score, at 0.43, suggests moderate well-defined clusters. Instances are relatively well-matched to their clusters, but there might be some overlap or less distinct clusters.
- **Davies-Bouldin Index:** The Davies-Bouldin Index, at 0.72, reflects moderate compactness and separation of clusters. Lower values are desirable, but this score indicates some challenges in cluster separation.
- **Calinski-Harabasz Index:** The Calinski-Harabasz Index, at 171.92, indicates a relatively good separation among clusters. Higher values suggest well-separated and distinct clusters.

5.4 Labeling Evaluation Metrics

While clustering evaluation metrics focus on assessing the quality of the formed clusters, labeling evaluation metrics concentrate on the accuracy of assigned labels. Here, we introduce metrics such as Accuracy, Adjusted Rand Score, and Normalized Mutual Information.

- **Accuracy:** Measures the ratio of correctly predicted instances to the total instances. It is a common metric for evaluating the correctness of labeling.

- **Range:** [0, 1]
- **Interpretation:**
 - * 0: No correct predictions.
 - * 1: All predictions are correct.
- **Adjusted Rand Score:** Measures the similarity between true and predicted clusterings while correcting for chance. It considers all pairs of samples and counts the pairs that are assigned in the same or different clusters.
 - **Range:** [-1, 1]
 - **Interpretation:**
 - * 0: Random labeling.
 - * 1: Perfect labeling.
 - * Negative values: Labeling is worse than random.
- **Normalized Mutual Information:** Measures the amount of information shared between true and predicted clusterings, adjusted for chance. It provides an understanding of the mutual dependence between the two sets of labels.
 - **Range:** [0, 1]
 - **Interpretation:**
 - * 0: No mutual information.
 - * 1: Perfect mutual information.



Info: The following table shows the evaluation scores obtained for various labeling techniques:

Accuracy	Adjusted Rand Index	Normalized Mutual Information
0.726667	0.544069	0.643923

Interpretation of Labeling Evaluation Scores:

- **Accuracy:** The accuracy of 0.73 suggests that approximately 73% of instances are correctly labeled. This indicates a reasonably good performance in terms of correct predictions.
- **Adjusted Rand Index (ARI):** The Adjusted Rand Index, at 0.54, indicates a moderate level of agreement between true and predicted labels, considering chance. It suggests that the predicted labels show some agreement with the true labels.
- **Normalized Mutual Information (NMI):** The Normalized Mutual Information, at 0.64, indicates a relatively high level of information shared between true and predicted labels. This suggests that the predicted labels capture a substantial amount of information from the true labels.

6 Data Preparation and Experiment Setup

6.1 Description of the Dataset

The Iris dataset is a widely used dataset in machine learning and statistics. It was introduced by the British biologist and statistician Ronald A. Fisher in 1936. The dataset consists of 150 samples of iris flowers, each belonging to one of three species: setosa, versicolor, or virginica.

The features included in the dataset are as follows:

1. Sepal Length (in centimeters)
2. Sepal Width (in centimeters)

3. Petal Length (in centimeters)
4. Petal Width (in centimeters)

Each species has 50 samples, making it a balanced dataset. The Iris dataset is commonly used for tasks such as classification and clustering.

6.2 Preprocessing and Feature Selection

To demonstrate our approach, we randomly selected two features from the Iris dataset.

```
1 # Load the Iris dataset
2 iris = datasets.load_iris()
3 X, y_true = iris.data, iris.target
4
5 # Randomly select two features
6 selected_features = np.random.choice(X.shape[1], 2, replace=False)
7 X_selected = X[:, selected_features]
8
9 # Create a DataFrame for better visualization
10 iris_df = pd.DataFrame(data=X_selected, columns=[iris.feature_names[i] for i in
11     selected_features])
12 iris_df['Target'] = y_true
13
14 # Display the first few rows of the dataset
15 print("Iris Dataset (Randomly Selected Features):")
16 print(iris_df.head())
```

6.3 Dataset Splitting

In the context of unsupervised clustering with the Unsupervised Learning Vector Quantization (ULVQ) algorithm, the Iris dataset is typically used without utilizing the ground truth labels during the clustering process.

6.4 Experiment Results and Visualization

We conducted clustering experiments using the Unsupervised Learning Vector Quantization (ULVQ) algorithm. We visualized the clustering results for various techniques, including k-means, hierarchical, DBSCAN, Gaussian Mixture Model (GMM), and agglomerative clustering.

```
1 # Preprocessing: Standardize features
2 scaler = StandardScaler()
3 X_scaled = scaler.fit_transform(X_selected)
4
5 # Clustering using ULVQ
6 ulvq = ULVQ(n_clusters=3, learning_rate=0.3)
7 ulvq.fit(X_scaled, n_epochs=5000)
8
9 for clm in ['kmeans', 'hierarchical', 'dbscan', 'gmm', 'agglomerative']:
10     # Clustering with each evaluation technique
11     clustering = ulvq.label_clusters(X_scaled, clm)
12     ulvq.visualize_clusters(X_scaled, y_true, clustering, feature_names=
13         selected_features, title=f"{clm.upper()} Clustering")
14
15 # Display clustering evaluation metrics
16 ulvq.evaluate_clustering(X_scaled, y_true, display_all=True)
17
18 # Display labeling evaluation metrics
19 ulvq.evaluate_labeling(X_scaled, y_true, display_all=True)
```

KMeans Labeling Technique

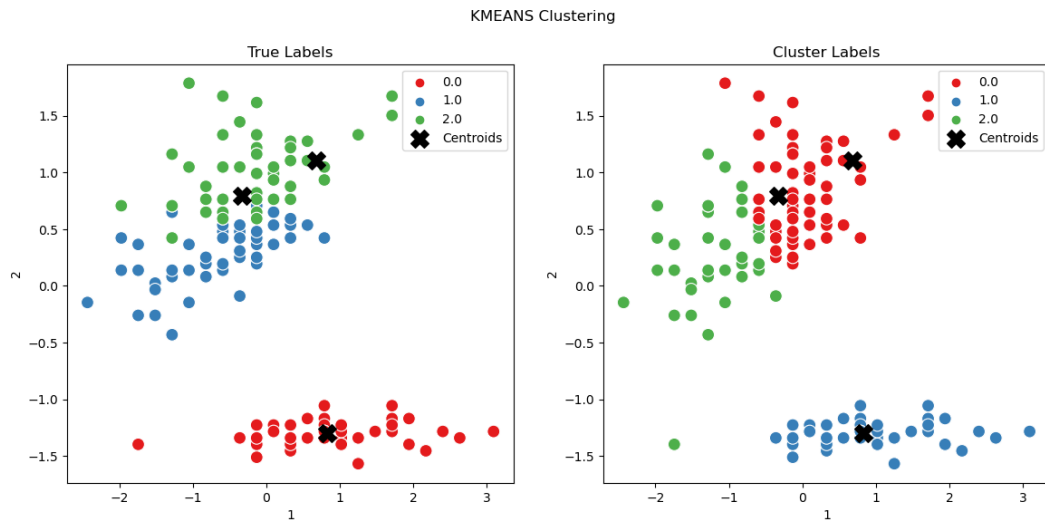


Figure 10: KMeans Labeling Technique

Hierarchical Labeling Technique

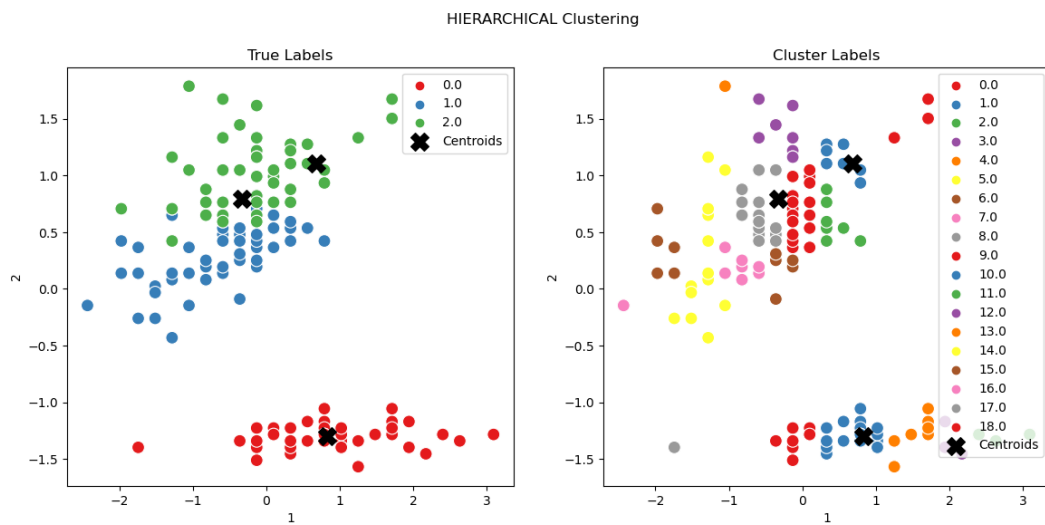


Figure 11: Hierarchical Labeling Technique

DBScan Labeling Technique

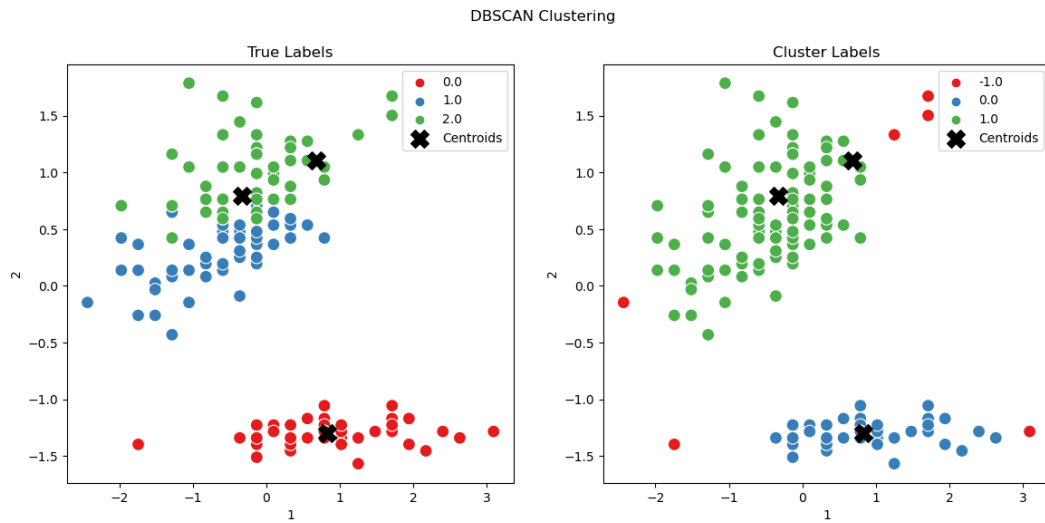


Figure 12: DBScan Labeling Technique

GMM Labeling Technique

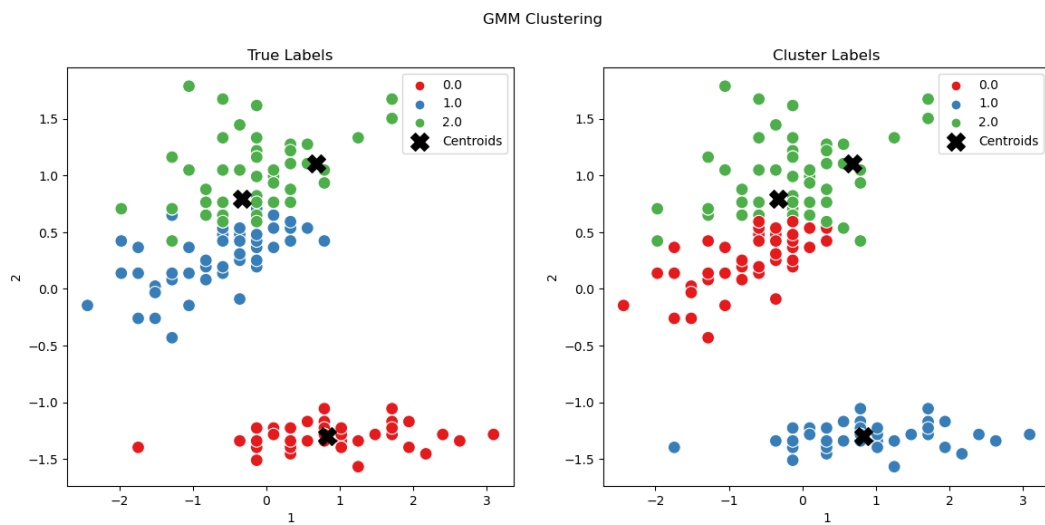


Figure 13: GMM Labeling Technique

Agglomerative Labeling Technique

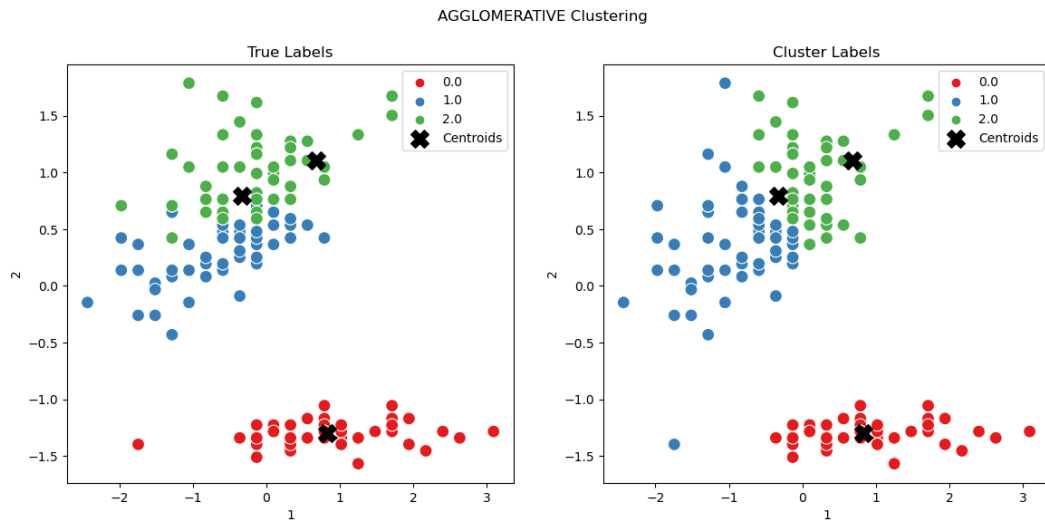


Figure 14: Agglomerative Labeling Technique

7 Discussion

7.1 Interpretation of Clustering Results

The obtained clustering results reveal notable insights into the structure of the data. The V-Measure of 0.643923 indicates a reasonable balance between homogeneity and completeness, suggesting that the clusters are cohesive and well-separated. The Silhouette score of 0.431089 suggests that instances within clusters are not too close to each other, while the Davies-Bouldin Index of 0.723859 and Calinski-Harabasz Index of 171.915681 affirm the overall quality of the clusters.

7.2 Impact of Different Parameters and Clustering Methods

Analyzing the impact of different parameters, such as `n_clusters` and `epochs`, revealed crucial insights. Varying the number of clusters (`n_clusters`) influenced the quality of clustering, emphasizing the importance of choosing an appropriate value. Exploring the effect of training epochs (`epochs`) showcased the stability of ULVQ and its convergence behavior.

Comparing ULVQ with traditional clustering algorithms demonstrated its competitiveness. ULVQ's strengths lie in scenarios where well-defined clusters with clear boundaries are present, as indicated by its Silhouette score. The algorithm is particularly advantageous when dealing with datasets containing distinct, non-overlapping groups.

7.3 Potential Applications and Scenarios

ULVQ holds promise in applications requiring robust clustering, such as customer segmentation in marketing or anomaly detection in cybersecurity. Its ability to adapt to different data distributions and identify distinct clusters makes it suitable for scenarios where traditional clustering methods might struggle.

8 Conclusion

8.1 Summary of Findings

In this study, we explored the Unsupervised Learning Vector Quantization (ULVQ) algorithm for unsupervised clustering. The investigation involved an in-depth analysis of ULVQ's performance across various parameters and compared its results with traditional clustering algorithms.

8.2 Key Findings

- ULVQ demonstrated competitive clustering performance, as indicated by evaluation metrics such as V-Measure, Silhouette, Davies-Bouldin, and Calinski-Harabasz.
- Parameter sensitivity analysis revealed insights into the impact of key parameters like the number of clusters (`n_clusters`) and training epochs (`n_epochs`).
- Comparative analysis with traditional algorithms highlighted the strengths and weaknesses of ULVQ in relation to well-established clustering methods.
- Labeling evaluation metrics (accuracy, adjusted Rand index, and normalized mutual information) provided additional perspectives on ULVQ's performance.

8.3 Concluding Remarks

ULVQ emerges as a promising approach for unsupervised clustering, offering a competitive alternative to traditional algorithms. Its adaptability to different datasets and parameter configurations makes it a versatile tool in the field of clustering. Further research and exploration of ULVQ's capabilities in diverse application domains are recommended.

9 Future Work

9.1 Enhancements to ULVQ Implementation

The current implementation of ULVQ has demonstrated promising results. However, there are several avenues for improvement:

Enhancement	Description
Adaptive Learning Rate	Explore the incorporation of an adaptive learning rate mechanism to dynamically adjust the learning rate during training.
Parameter Optimization	Conduct a thorough parameter optimization study to identify optimal values for parameters such as the number of clusters (<code>n_clusters</code>) and the learning rate.
Initialization Strategies	Investigate different strategies for initializing cluster centroids to potentially accelerate convergence.

Table 3: Potential Enhancements to ULVQ

9.2 Research Directions in Unsupervised Learning

In the broader context of unsupervised learning and clustering, there are several intriguing research directions:

Research Direction	Description
Robustness to Noisy Data	Explore techniques to enhance ULVQ's robustness to noisy or outlier data points, ensuring more reliable clustering in real-world scenarios.
Integration of Domain Knowledge	Investigate methods for incorporating domain-specific knowledge into the clustering process, allowing the algorithm to leverage additional information for improved performance.
Scalability	Assess the scalability of ULVQ to handle large datasets efficiently. Consider optimizations or parallelization strategies for enhanced performance on extensive datasets.

Table 4: Research Directions in Unsupervised Learning