Course: DVA262 — Machine Learning Concepts

Objective: The purpose of the Lab is to apply your understanding of cluster algorithms by implementing the k-Means and Fuzzy c-Means algorithms from scratch in the C++ programming using the Lab skeleton.

Lab 5: Clustering Algorithms Implementation

Execution and demonstration

This lab is executed a group of two students or individually. The completed lab should be demonstrated to a lab assistant.

Useful aids

- The course literature
- Eigen https://eigen.tuxfamily.org/dox/GettingStarted.html
- Lecture slides and notes
- Google & Internet

Part 1: k-Means for Clustering

You will implement the k-Means algorithm for the clustering problem in this task.

Dataset: The classification problem will be based on the **Iris dataset,** which is already available in the project folder. However, when you load the data set you will see that the labels are discarded from the data.

The Visual Interface: The main form (i.e., MainForm.h) calls all the class objects and functions that are already implemented in the skeleton. However, you may need to edit some parts of the code to make it consistent with your code to display the results correctly.

- *Loading and splitting the dataset*
  The code for loading and preparing the dataset is already implemented, so no additional code is required for this part. Use the following function in your code:

- You can use the function loadAndPreprocessDataset() to prepare the dataset, which is located in the class name DataLoader.cpp under DataUtils folder.
- Use the splitDataset() function in the class name DataPreprocessor.cpp under the DataUtils folder to split the dataset into training and test sets.

Your Task: You will mainly work with the KMeans.cpp class located under the Clustering folder.

- *Creating class member functions*

The KMeans.cpp class will have the following member functions:

- Constructor: initialises the parameters, i.e., number of the clusters and max iterations.
- fit() function: this function trains the model for making predictions later.
- predict() function: calculates the closest centroid for each point in the given data set and returns the labels of the closest centroids.

Task 1: In this Lab work you must complete the k-Means clustering implementation by completing the fit(), and predict() functions.

You should use one of the similarity functions from the SimilarityFunctions.cpp class.

- *Evaluating*

Evaluating the clustering's performance in the dataset using the Davies-Bouldin index and Silhouette Score. The code for these metrics is already done, so no additional code is required.

- You can use the calculateDaviesBouldinIndex() and calculateSilhouetteScore() functions, which are located in the class name Metrics.cpp under the Evaluation folder for the clustering evaluation.

Task 2: What value of *numClusters*, *maxIteration* have you used? Try different values for these parameters. What is the largest value of *numClusters* you can choose? Discuss your findings during the demonstration.

Note: The values of these parameters *are* set when you create an object of the KMeans.cpp class in the MainForm.cpp class

- *Visualisation:*

A visual representation (for training and test sets results) is needed. However, the code for this task is already done, so use the following functions for that:

- Use the performPCA() function to perform Principal Component Analysis (PCA) and project the data onto a lower-dimensional space, which can be found in the class name PCADimensionalityReduction.cpp under the Utils folder.
- Use the PlotReducedDimensionalData() function to visualise the K-Means clustering result located in the MainForm.h.

An example of the result from the k-means cluster is shown in the Figure 1. The visual interface is the same for all clustering algorithms.

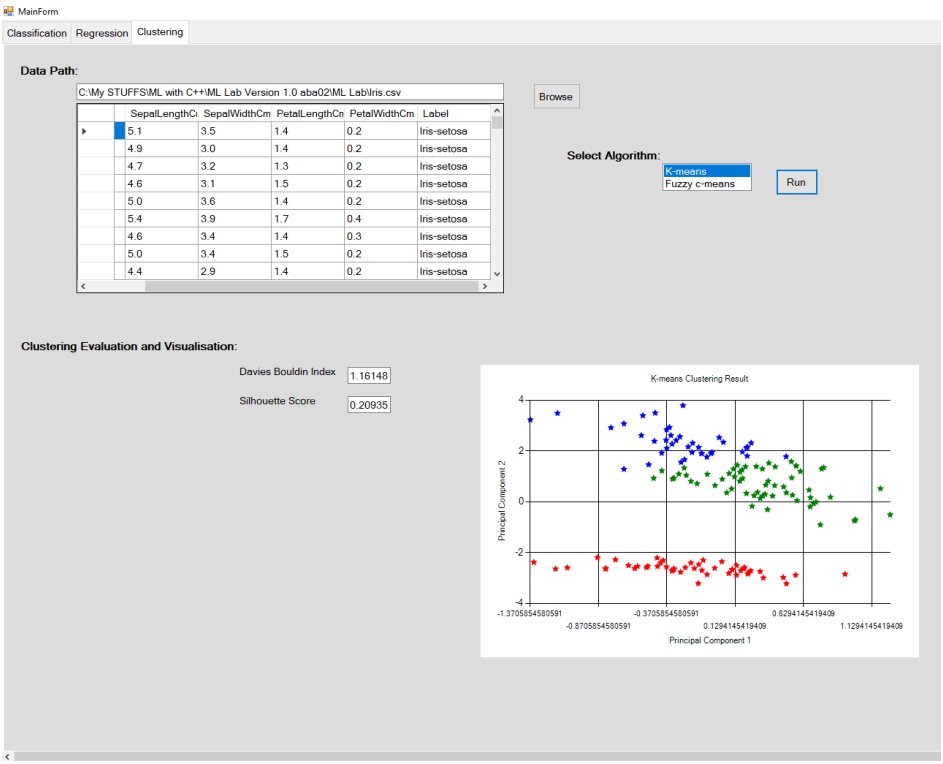

Figure 1: Example result of k-Means Clustering in the visual interface.

Part 2: Fuzzy c-Means for Clustering

In this task, you will implement Fuzzy c-Means algorithm for the clustering problem.

Dataset: The classification problem will be based on the **Iris dataset,** which is already available in the project folder. However, when you load the data set you will see that the labels are discarded from the data.

The Visual Interface: The main form (i.e., MainForm.h) calls all the class objects and functions that are already implemented in the skeleton. However, you may need to edit some parts of the code to make it consistent with your code to display the results correctly.

- *Loading and splitting the dataset*
  The code for loading and preparing the dataset is already implemented, so no additional code is required for this part. Use the following function in your code:

  - You can use the function loadAndPreprocessDataset() to prepare the dataset, which is located in the class name DataLoader.cpp under DataUtils folder.
  - Use the splitDataset() function in the class name DataPreprocessor.cpp under the DataUtils folder to split the dataset into training and test sets.

Your Task: You will mainly work with the FuzzyCMeans.cpp class located under the Clustering folder.

- *Creating class member functions*

The FuzzyCMeans.cpp class will have the following member functions:

- Constructor: initialises the parameters, i.e., number of the clusters and max iterations and fuzziness.
- fit() function: this function trains the model for making predictions later
- predict() function: calculates membership values for each cluster and assigns the data point to the cluster with the highest membership.

Task 3: In this part of the work you must complete the Fuzzy c-Means clustering implementation by completing the fit(), and predict() functions.

You should use one of the similarity functions from the SimilarityFunctions.cpp class.

- *Evaluating (Same as K-Means)*

Evaluating the clustering's performance in the dataset using the Davies-Bouldin index and Silhouette Score. The code for these metrics is already done, so no additional code is required.

- You can use the calculateDaviesBouldinIndex() and calculateSilhouetteScore() functions, which are located in the class name Metrics.cpp under the Evaluation folder for the clustering evaluation.

Task 4: What value of *numClusters*, *maxIteration* and *fuzziness* have you used? Try different values for these parameters. Discuss your findings during the demonstration.

Note: The values of these parameters *are* set when you create an object of the FuzzyCMeans.cpp class in the MainForm.cpp class

- *Visualisation*

A visual representation (for training and test sets results) is needed. However, the code for this task is already done, so use the following functions for that:

- Use the performPCA() function to perform Principal Component Analysis (PCA) and project the data onto a lower-dimensional space, which can be found in the class name PCADimensionalityReduction.cpp under the Utils folder.
- Use the PlotReducedDimensionalData() function to visualise the K-Means clustering result located in the MainForm.h.

The expected results from the Fuzzy C-Means should be visible in the visual interface, as an example shown in Figure 1.

Some Hints:

- Include the necessary header files at the beginning of your code.
- It is better to initialize centroids randomly in k-Means algorithm.
- If you already implemented several similarity functions in the SimilarityFunctions.cpp class, check which one performs best.
- In Fuzzy c-Means, randomly initialize the membership matrix.