Kennesaw State University
CS8267 – Advanced Machine Learning
Fall 2023

Assignment 1
Unsupervised Learning: K-Means Clustering

Use Case: Identifying topics from articles on the same topic

Purpose

Clustering is an unsupervised method for grouping patterns based on similarity such that the patterns in the same group are similar to one another, and those in different groups are less similar to one another. Clustering has many uses, including outlier detection and unsupervised image segmentation. The K-Means clustering algorithm is one of the most widely used clustering algorithms. In this assignment, you are going to implement basic k-means.

Goals

- Learn machine learning platforms, libraries and tools
- Learn how distance based clustering algorithms work
- Understand how to apply clustering
- Understand which data sets k-means clustering is appropriate for

Description

Implement a program to analyze textual data collection using large language models (LLMs). Use your favorite web search engine to find articles on a machine learning topic (e.g., transformers, text summarization, ChatGPT, etc.) and select 5-10 articles. Your program will analyze the content of these articles by applying k-means clustering to identify the topics that are covered.

Input

It is important that you implement k-means algorithm and verify the correctness of the algorithm before deploying in a larger context. The kmtest.arff (or kmtest.csv) file is very simple data set with two attributes that you can use to debug your k-means code. The ARFF files are formatted as text files with one sample / pattern vector per line of input. Each line will contain a series of attribute values, separated by commas or spaces. The files have an ARFF header consisting of a set of tags beginning with the '@' character. These tags

provide descriptions of the attributes (name and type). The data begins after the @data tag, which denotes the end of the header. Do not use *class* attributes in these files for clustering.

Use **langchain** for processing textual data and use available resources. For example, the link https://python.langchain.com/docs/use_cases/question_answering.html provides a simple use case for a question answering system from a set of documents. You need to split the data into smaller overlapping fixed sized segments and generate *embeddings* for each segment.

Embeddings are numeric vectors used as a feature representation for the data. Embedding libraries are available, and your program should just indicate the embedding libraries to be used when generating the embeddings. OpenAI supports embeddings; however, you need to subscribe with a credit card. Another library comes from Hugging Face (HuggingFaceEmbeddings). If you use hugging face, you should register hugging face to receive a token, but no credit card is required. You may use any available embeddings.

These embeddings could be stored in a vector store such as ChromaDB. Use embeddings to apply k-means algorithm. For each cluster, find the closest embedding to the center. Then use a chat LLM to rephrase the selected data for each cluster using a prompt. You may use any chat LLM. You may consider using text2text generation pipeline or check How to Paraphrase Text using Transformers in Python - Python Code (thepythoncode.com). You may consider using different k values (3, 5, 10).

Principal Components Analysis (PCA): Use PCA to determine if it would help to cluster with fewer principal components. Use PoV as 0.9 to determine the number of principal components. In addition, try 3, 7, and 19 principal components for analysis.

Deliverables

- Submit your Jupyter notebook or similar with the outputs of all cells.
- Make sure that cluster centers are computed and presented for the kmtest file.
- **Note about filename notation:** Your notebook file must start with your login name. Your file name will look like raygunClustering.ipnyb
- Submission by Email is not acceptable.
- 50% penalty for one day late submission (except for medical and other emergency reasons). Submissions one day late will not be accepted.
- All submissions are due class time unless stated otherwise.

Notes

The following additional comments apply:

1. Good programming style must be observed. This includes using meaningful variable names, descriptive comments, and readable formatting

2. The program source code file must contain a header comment at the beginning of the file. This comment must include the following items:
    - Source code file name
    - Student Name
    - Date
3. Students should work independently. Each student is responsible for handing in an original program.