Statistical Methods in Artificial Intelligence

Assignment 2

Deadline : 17 September 2024 11:55 P.M

Instructor: Prof Ravi Kiran Sarvadevabhatla

September 9, 2024

1 General Instructions

- Your assignment must be implemented in Python.
- While you're allowed to use LLM services for assistance, you must explicitly declare in comments the prompts you used and indicate which parts of the code were generated with the help of LLM services.
- Plagiarism will only be taken into consideration for code that is not generated by LLM services. Any code generated with the assistance of LLM services should be considered as a resource, similar to using a textbook or online tutorial.
- The difficulty of your viva or assessment will be determined by the percentage of code in your assignment that is not attributed to LLM services. If during the viva you are unable to explain any part of the code, that code will be considered as plagiarized.
- Clearly label and organize your code, including comments that explain the purpose of each section and key steps in your implementation.
- Properly document your code and include explanations for any non-trivial algorithms or techniques you employ.
- Ensure that your files are well-structured, with headings, subheadings, and explanations as necessary.
- Your assignment will be evaluated not only based on correctness but also on the quality of code, the clarity of explanations, and the extent to which you've understood and applied the concepts covered in the course.
- Make sure to test your code thoroughly before submission to avoid any runtime errors or unexpected behavior.

- Submit your assignment sticking to the format mentioned below. Not doing so will result in penalization.

```
├── assignments
│   ├── 1
│   │   └── a1.py
│   ├── 2
│   │   ├── a2.py
│   │   ├── figures
│   │   │   └── README.md
│   │   └── README.md
│   ├── 3
│   │   └── a3.py
│   ├── 4
│   │   └── a4.py
│   └── 5
│       └── a5.py
├── data
│   ├── external
│   │   ├── spotify.csv
│   │   └── word-embeddings.feather
│   ├── interim
│   │   ├── 1
│   │   ├── 2
│   │   ├── 3
│   │   ├── 4
│   │   ├── 5
│   │   └── README.md
│   └── README.md
├── models
│   ├── gmm
│   │   └── gmm.py
│   ├── k-means
│   │   └── k-means.py
│   ├── knn
│   │   └── knn.py
│   ├── pca
│   │   └── pca.py
│   └── README.md
├── performance-measures
│   └── README.md
└── README.md
```

  - The `data/external` folder contains the datasets being provided to you. You should use the `data/interim` folder to store data that has been transformed and that is in use.

  - There are folders for each assignment inside the `data/interim` directory, and make sure to store data specific to each assignment accordingly.
  - Store implementations of the models in the `models` folder.
    * Contrary to what was mentioned in the tutorial, you are supposed to make `fit` and `predict` routines inside the specific model class instead of making separate train, test and eval files. These routines **should** be present in the specific model class and have the naming convention mentioned.
  - For particular tasks related to the assignments, do them inside the folder for that specific assignment and import classes and data from the other files.
  - `performance-measures/` should contain the generalized implementations of all evaluation metrics, in a way such that they can be used for any model as and when needed.
  - Each assignment folder has a `README.md` which you will have to modify to make the final report with all your observations, analyses, and graphs.
  - The figures folder has been moved into the assignment-specific directory. You should save all the generated plots, animations, etc inside that folder and then include them in the report.
  - Mention the references used for each assignment in the report.

- The deadline will not be extended.
- MOSS will be run on all submissions along with checking against online resources.
- We are aware of how easy it is to write code now in the presence of LLM services, but we strongly encourage you to write the code yourself.
- We are aware of the possibility of submitting the assignment late in GitHub classrooms using various hacks. Note that we will have measures in place accordingly and anyone caught attempting to do the same will be given a straight zero in the assignment.
- You are not allowed to use external libraries except for standard ones such as numpy and matplotlib unless specified in the question or consulted by a TA.
- Vectorize the code as much as possible as that will also be evaluated along with the results and code.

2 Structure of The Dataset

The dataset for this assignment is available in the GitHub repository named as `word-embeddings.feather`. This file contains the data where each row represents a single data point, corresponding to an example in the dataset. The first column in each row contains a word, while the second column contains the corresponding 512-dimensional embedding generated by VIT. Throughout the assignment, references to the dataset pertain to this 512-dimensional word embedding, which serves as your feature space.

3 K-Means Clustering [2 Days]

3.1 Implement a K-Means Class

- Implement the K-means Clustering Class that replicates the core functionalities of the built-in K-means library, including methods such as `fit()` and `predict()`.
- The `fit()` function trains the K-means model by finding the k optimal cluster centroids based on the input data.
- The `predict()` method assigns a cluster number to each data point based on the optimal centroids determined by the fit() function.
- The `getCost()` method returns the cost of K-Means which is the Within-Cluster Sum of Squares (WCSS).
- Specify the number of clusters (`k`) while instantiating the class.

3.2 Determine the Optimal Number of Clusters for 512 dimensions

- Use the Elbow Method to determine the optimal number of clusters for the 512-dimensional dataset. Vary the value of k and plot the Within-Cluster Sum of Squares (WCSS) against k to identify the "elbow" point, which indicates the optimal number of clusters. We'll refer to this as $k_{kmeans1}$.

  ***Note**: Do not directly use the code provided in the linked article, as this will be considered plagiarism. Instead, use it as a reference to understand the concept and write your own implementation.*

- Perform K-means clustering on the dataset using the number of clusters as $k_{kmeans1}$.

4 Gaussian Mixture Models [3 Days]

4.1 Implement the GMM Class

- Write your own GMM class. Your GMM class should include methods like `fit()`, `getParams()`, `getMembership()` and `getLikelihood()`.
- The `fit()` method implements the Expectation-Maximization (EM) algorithm on the dataset to determine the optimal parameters for the model.
- The `getParams()` method returns the parameters of the Gaussian components in the mixture model.
- The `getMembership()` method returns the membership values r_{ic} for each sample in the dataset. The membership value r_{ic} for a sample x_i is the probability that the sample belongs to cluster c in the given GMM.
- The `getLikelihood()` method returns the overall likelihood of the entire dataset under the current model parameters. The likelihood values measure how likely the observed data is given the model.

4.2 Determine the Optimal Number of Clusters for 512 dimensions

- Perform GMM clustering on the give dataset for any number of clusters. **Does the class work? Why?** Now perform GMM clustering using the sklearn GMM class. **Does this class work? Why?**
- Use BIC (Bayesian Information Criterion) and AIC (Akaike Information Criterion) to determine the optimal number of clusters for the 512-dimensional dataset. Use whichever GMM class worked in the above sub part. We'll refer to this as k_{gmm1}.
- Perform GMM on the dataset using the number of clusters as k_{gmm1} using the class which worked in the previous subparts.

5 Dimensionality Reduction and Visualization [3 Days]

You want to observe the given data which consists of words and their corresponding word embeddings. The words are encoded into 512-dimensional which cannot be visualized directly.

We will be looking into PCA (principal component analysis), for this task. Principal Component Analysis (PCA) is the general name for a technique which uses sophisticated underlying mathematical principles to transform several possibly correlated variables into a smaller number of variables called principal components.

5.1 Implement a PCA Class

- Write your own PCA Class. Your PCA class should include methods like `fit()`, `transform()` and `checkPCA()`.
- The `fit()` method obtains the principal components of our dataset
- The `transform()` method transforms the data using the principal components after we fit the data.
- The `checkPCA()` method should help you check if your Class reduces the dimensions appropriately. It should return **True** if the class works and **False** otherwise.
- Specify the number of components (`n_components`) when instantiating the class to define how many dimensions the data should be reduced to.

5.2 Perform Dimensionality Reduction

- Use the PCA class created above to `fit` and `transform` the data into **2** and **3** dimensions.
- Verify the functionality of the class using the `checkPCA` method.
- Visualize the results by plotting the data in both 2D and 3D.

5.3 Data Analysis

- Can you identify what each of the new axes that are obtained from PCA represent?
- Examine the 2D and 3D plots to identify any visible patterns or clusters. Without performing any clustering, estimate the approximate number of clusters based on the plots. We'll refer to this estimated number as k_2, which will be used in subsequent problems.

6 PCA + Clustering [2 Days]

6.1 K-means Clustering Based on 2D Visualization

- Perform K-means clustering on the dataset using the number of clusters estimated from the 2D visualization of the dataset (k_2).

6.2 PCA + K-Means Clustering

- Generate a scree plot to identify the optimal number of dimensions for reduction. Apply dimensionality reduction based on this optimal number of dimensions. We'll refer to this as the *reduced dataset*
- Determine the optimal number of clusters for the *reduced dataset* using the Elbow Method. We'll refer to this as $k_{kmeans3}$.
- Perform K-means clustering on the *reduced dataset* using number of clusters as $k_{kmeans3}$.

6.3 GMM Clustering Based on 2D Visualization

- Perform GMMs on the dataset using the number of clusters estimated from the 2D visualization (k_2), using the **GMM class written by you.**

6.4 PCA + GMMs

- Determine the optimal number of clusters for the reduced dataset as obtained in 6.2 using AIC or BIC. Let us call this the new k_{gmm3}
- Apply GMMs with k_{gmm3} to the dimensionally reduced dataset, using the **GMM class written by you.**

7 Cluster Analysis [2 Days]

7.1 K- Means Cluster Analysis

- Compare the clustering results obtained using $k_{kmeans1}$, k_2, and $k_{kmeans3}$.
- Assess which clustering approach yields better results by evaluating the coherence and meaningfulness of the clusters. In this context, coherence refers to how similar and well-grouped the target words are within each cluster, indicating that the words within the same cluster are closely related or similar in meaning. Based on your analysis, identify the best k for which the clustering results are most coherent and interpretable. Label this as k_{kmeans}

7.2 GMM Cluster Analysis

- Compare the clustering results obtained using k_{gmm1}, k_2, and k_{gmm3}.
- Assess which clustering approach yields better results by evaluating the coherence and meaningfulness of the clusters. In this context, coherence refers to how similar and well-grouped the target words are within each cluster, indicating that the words within the same cluster are closely related or similar in meaning. Based on your analysis, identify the best k for which the clustering results are most coherent and interpretable. Label this as k_{gmm}

7.3 Compare K-Means and GMMs

- Compare the clusters obtained from K-means with k_{kmeans} and from GMM with k_{gmm}.
- Assess the effectiveness of each clustering method by evaluating how well the target words are grouped. Comment on which clustering approach results in more meaningful groupings. Consider factors such as the similarity within clusters and the separation between different clusters.

8 Hierarchical Clustering [2 Days]

Hierarchical clustering is a popular method for grouping objects. It creates groups so that objects within a group are similar to each other and different from objects in other groups. Clusters are visually represented in a hierarchical tree called a dendrogram. Reference for hierarchical clustering and linkages

- Apply hierarchical clustering to the dataset to uncover natural groupings. Use in-built functions such as `hc.linkages(X, linkage type)` to compute the linkage matrix and `hc.dendogram(Z)` to plot the dendogram.

- Experiment with different linkage methods and distance metrics to obtain and analyze the linkage matrix. Plot the dendrograms and document your observations.
- Using the Euclidean distance metric and the best linkage method identified from your analysis, create clusters by cutting the dendrogram at the points corresponding to k_{best1} (the best k from K-means clustering) and k_{best2} (the best k from GMM clustering). Compare these clusters to see if they align or differ from the K-Means and GMM clustering respectively.

  ***Hint**: To cut the dendrogram and form clusters, look into using the* `fcluster()` *function from the* `scipy.cluster.hierarchy module`.

9 Nearest Neighbor Search [2 Days]

We have seen the advantages of PCA. Let us see how it affects our Machine Learning Model.

9.1 PCA + KNN

- Use the Spotify dataset (can be found in the GitHub repository or here) from Assignment 1 to generate a scree plot and determine the optimal number of dimensions for reduction. Apply dimensionality reduction based on this optimal number of dimensions.
- Use the KNN model implemented in Assignment 1 on the reduced dataset using the best {`k`, distance metric} pair obtained.

9.2 Evaluation

- After splitting the dataset into training and validation subsets, calculate and return the validation F1 score, accuracy, precision, and recall.
- Compare these metrics with those obtained in Assignment 1 and provide an analysis of any differences or improvements.
- Plot the inference time for the KNN model on both the complete dataset and the reduced dataset. Comment on the differences and implications of the inference times.

Good luck with the assignment!