Assignment 2 Report

Subject: Parallel Algorithms I
Faculty: Faculty of Electrical Engineering and Computer Science
ID: EUN0006
Name: Eunji Jeon

Index

I. Introduction

1.1 Overview of Affinity Propagation Clustering

Affinity Propagation (AP) is a representative-based clustering algorithm that belongs to the category of unsupervised machine learning techniques. Unlike traditional clustering methods like k-means or hierarchical clustering, AP does not require the number of clusters to be specified in advance. Instead, it automatically identifies the representative data points, called "cluster centers" or "exemplars", and assigns the remaining data points to these centers.

1.2 Objectives of the Project

The primary objective of this project is to implement the Affinity Propagation clustering algorithm and apply it to the MNIST dataset of handwritten digits. The goal is not to cluster the data based on the actual digit labels, but rather to analyze the intrinsic clusters that the AP algorithm discovers within the unlabeled data. By studying the characteristics of the resulting clusters, we aim to gain insights into the underlying structure of the MNIST dataset.

II. Theoretical Background

2.1 Affinity Propagation Algorithm: A Comprehensive Overview

Affinity Propagation (AP) represents a sophisticated approach to clustering that fundamentally differs from traditional clustering methodologies. At its core, the algorithm operates through a message-passing mechanism between data points, dynamically discovering cluster structures without requiring predefined cluster numbers.

2.1.1 Similarity Matrix Computation

The algorithmic journey begins with constructing a similarity matrix S, which quantifies the intrinsic relationships between data points. Unlike distance-based metrics that simply measure spatial proximity, AP calculates similarities using the negative squared Euclidean distance. This approach transforms raw spatial distances into a nuanced representation of point interactions, where smaller distances yield higher similarity values.

The mathematical formulation S(i,j) = -||xi - xj||^2 ensures that closely situated points receive higher similarity scores, creating a foundation for identifying potential cluster structures. Critically, the diagonal elements of this matrix are strategically initialized, typically using the mean of existing similarity values, which helps stabilize the initial clustering process.

2.1.2 Message Passing Dynamics

The algorithm's distinctive feature lies in its two-matrix message-passing mechanism: the responsibility matrix R and the availability matrix A. These matrices engage in an intricate dialogue, iteratively refining cluster assignments through sophisticated update rules.

The responsibility matrix R(i,k) captures how strongly a data point i considers another point k as a potential cluster representative. It does this by comparing k's suitability against all other potential representatives, creating a competitive selection process. Simultaneously, the availability matrix A(i,k) represents the accumulated evidence supporting a point k becoming a cluster exemplar, essentially measuring how "available" k is to represent other points.

This bidirectional communication between responsibilities and availabilities allows the algorithm to converge towards an optimal clustering configuration, where representative points emerge naturally from the data's inherent structure.

2.1.3 Cluster Identification and Convergence

After numerous iterations, the algorithm determines final cluster assignments by combining the responsibility and availability matrices into a criterion matrix C. Points with the highest values in this matrix are designated as cluster representatives, with each data point assigned to the cluster of its most suitable representative.

2.2 Algorithmic Characteristics

Affinity Propagation offers a unique set of capabilities that distinguish it from conventional clustering techniques. Its ability to automatically determine cluster count provides significant flexibility, eliminating the need for manual cluster number specification. By identifying representative data points (exemplars), the algorithm not only clusters data but also provides interpretable insights into each cluster's characteristics.

The algorithm's design allows it to transcend limitations of traditional clustering methods. Unlike algorithms assuming uniform cluster shapes, AP can discover clusters with more complex geometries. However, this flexibility comes with computational trade-offs. The method's iterative nature and the requirement to maintain and update large matrices make it computationally intensive, particularly for extensive datasets.

The algorithm's performance is sensitive to the damping factor, a crucial hyperparameter that controls the rate of matrix updates. An inappropriately chosen damping factor can lead to unstable convergence or prevent the algorithm from discovering meaningful clusters.

III. Implementation Details

3.1 Python Implementation

3.1.1 Data Preprocessing

In the data preprocessing stage, a careful approach was taken to effectively cluster the MNIST dataset. The `load_mnist_data()` function facilitated data loading and processing through a series of critical steps. The data was loaded from a CSV file using Pandas, with a random selection of 500 samples to efficiently test the algorithm's performance while managing computational complexity. Standardization was achieved using scikit-learn's StandardScaler, which adjusted all features to have a mean of 0 and variance of 1. This process equalized the scale of pixel features, enhancing the accuracy of distance-based similarity calculations.

3.1.2 Affinity Propagation Clustering

The `AffinityPropagationClusterer` class was responsible for the core implementation of the Affinity Propagation algorithm. The similarity matrix calculation method utilized the negative squared Euclidean distance to define point similarities, with diagonal elements initialized to the matrix's average value. The iterative matrix update process in the `fit()` method incorporated a damping factor to ensure update stability, with carefully defined minimum and maximum iteration limits of 15 and 200 respectively. The method compared matrix states between consecutive iterations to determine convergence, creating a robust mechanism for cluster assignment.

The implementation employed a sophisticated approach to updating responsibility and availability matrices. By using a damping coefficient, the algorithm maintained numerical stability while allowing for dynamic cluster representation discovery. The final cluster representatives and data point assignments were determined based on the criterion matrix, which combined responsibility and availability matrices. This approach enabled the algorithm to automatically identify cluster centers without predefined cluster numbers.

3.1.3 Visualization

The visualization stage leveraged UMAP (Uniform Manifold Approximation and Projection) to reduce the high-dimensional MNIST data to a two-dimensional space. Carefully tuned hyperparameters, including `n_neighbors=8` and `min_dist=0.1`, were selected to preserve the local data structure during dimensionality reduction. The visualization process employed matplotlib to create high-quality (300 dpi) graphics that effectively represented the clustering results.

The visualization strategy included distinct color representation for each cluster, with cluster representatives highlighted using black 'x' markers. This approach provided an intuitive and informative view of the discovered cluster structures, allowing for easy interpretation of the Affinity Propagation algorithm's results. The method effectively transformed complex, high-dimensional data into a comprehensible visual representation that captured the intrinsic grouping of MNIST

dataset samples.

This implementation approach successfully captured the complexity of the Affinity Propagation algorithm while focusing on exploring the inherent structure of the MNIST dataset through an innovative clustering methodology.

IV. Results

4.1 Cluster Analysis

The application of the Affinity Propagation (AP) algorithm to the MNIST dataset resulted in the discovery of 73 distinct clusters. An in-depth analysis of these clusters revealed several key characteristics:

- Cluster Sizes: The cluster sizes ranged from a minimum of 1 data point to a maximum of 39 data points, showcasing a diverse range of groupings within the dataset.

- Cluster Composition: While some clusters were composed entirely of a single digit label, a significant number of clusters contained a mixture of 2 to 5 different digit labels. This heterogeneous nature of the clusters suggests that the AP algorithm was able to identify intrinsic relationships between data points that may not strictly align with the pre-defined digit classes.

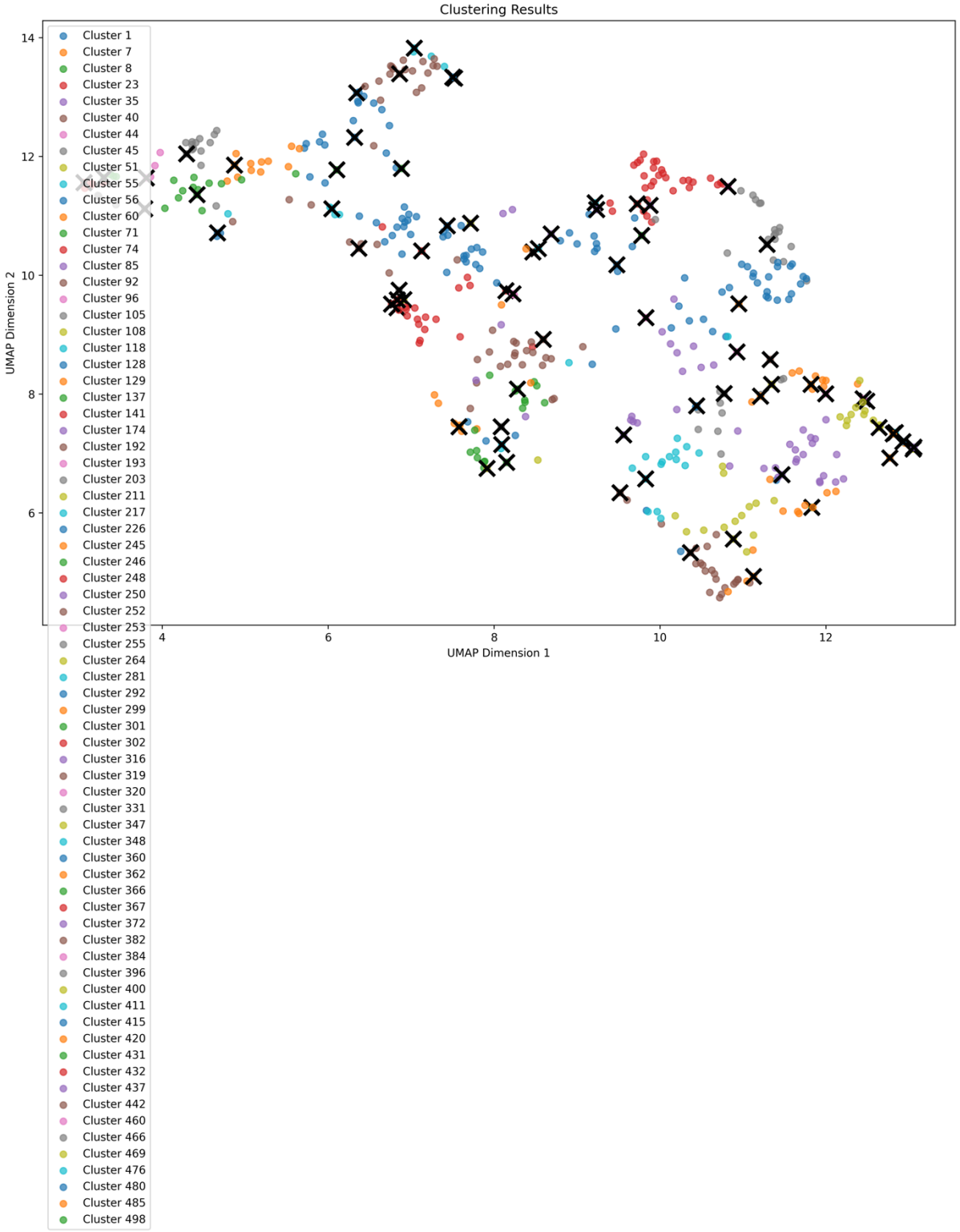
- Digit Representation: The digit labels 0, 1, 2, 3, 4, 5, 7, 8, and 9 were fairly well-represented across the 73 clusters. However, the digit 6 appeared relatively less frequently, indicating that the algorithm may have had a harder time distinguishing this particular digit from others in the dataset.

4.2 Visualization and Cluster Separation

The UMAP-based visualization of the clustering results provided valuable insights into the structure and relationships between the discovered clusters. The majority of the clusters were well-separated and clearly distinguishable from one another, indicating that the AP algorithm was effective in identifying distinct groupings within the high-dimensional MNIST data.

However, the visualization also revealed that some clusters had blurred boundaries or overlapped with each other. This observation suggests that there may have been potential challenges in interpreting the clusters or separating them cleanly, as the boundaries between certain clusters were not as well-defined.

The ability to project the high-dimensional data onto a two-dimensional space using UMAP enabled a comprehensive understanding of the clustering structure, highlighting both the strengths and potential limitations of the Affinity Propagation algorithm in analyzing the MNIST dataset.


Clustering Results
UMAP Dimension 1
UMAP Dimension 2
4
6
8
10
12
14
Cluster 1
Cluster 7
Cluster 8
Cluster 23
Cluster 35
Cluster 40
Cluster 44
Cluster 45
Cluster 51
Cluster 55
Cluster 56
Cluster 60
Cluster 71
Cluster 74
Cluster 85
Cluster 92
Cluster 96
Cluster 105
Cluster 108
Cluster 118
Cluster 128
Cluster 129
Cluster 137
Cluster 141
Cluster 174
Cluster 192
Cluster 193
Cluster 203
Cluster 211
Cluster 217
Cluster 226
Cluster 245
Cluster 246
Cluster 248
Cluster 250
Cluster 252
Cluster 253
Cluster 255
Cluster 264
Cluster 281
Cluster 292
Cluster 299
Cluster 301
Cluster 302
Cluster 316
Cluster 319
Cluster 320
Cluster 331
Cluster 347
Cluster 348
Cluster 360
Cluster 362
Cluster 366
Cluster 367
Cluster 372
Cluster 382
Cluster 384
Cluster 396
Cluster 400
Cluster 411
Cluster 415
Cluster 420
Cluster 431
Cluster 432
Cluster 437
Cluster 442
Cluster 460
Cluster 466
Cluster 469
Cluster 476
Cluster 480
Cluster 485
Cluster 498

V. Challenges and Limitations

5.1 Algorithmic Challenges

5.1.1 Responsibility and Availability Matrix Updates

One of the main challenges in implementing the Affinity Propagation algorithm was correctly translating the mathematical definitions of the responsibility and availability matrix updates into working code. Ensuring the proper implementation of these update rules was crucial for the algorithm to converge correctly.

5.1.2 Convergence Criteria

Determining the appropriate stopping criteria for the iterative updates of the responsibility and availability matrices was not straightforward. We had to balance the need for convergence with the desire to obtain stable and meaningful cluster assignments.

5.2 Computational Limitations

5.2.1 Memory Requirements

Despite using a subset of the MNIST dataset, the memory requirements for storing the responsibility and availability matrices were still significant. This could pose a problem when applying the AP algorithm to larger datasets.

5.2.2 Scalability

The relatively slow convergence speed of the Affinity Propagation algorithm limits its applicability to truly large-scale datasets. Parallelization or the use of approximation techniques may be necessary to improve the algorithm's scalability.

5.3 Inherent Limitations of Affinity Propagation

5.3.1 Cluster Shape Assumptions

The Affinity Propagation algorithm assumes that the clusters have a spherical or Gaussian-like shape. This may limit its ability to discover clusters with more complex or irregular geometries.

5.3.2 Interpretation of Large Cluster Numbers

As the number of discovered clusters increases, the interpretation and analysis of the clustering results can become more challenging. Techniques to summarize and visualize the clusters may be needed to maintain the algorithm's interpretability.

VI. Conclusion

6.1 Summary of Findings

In this project, we successfully implemented the Affinity Propagation clustering algorithm and applied it to the MNIST dataset of handwritten digits. The AP algorithm was able to automatically discover 73 distinct clusters within the data, with varying degrees of cluster purity and separation.

6.2 Significance and Implications

The ability of the Affinity Propagation algorithm to identify meaningful clusters without requiring the number of clusters to be specified in advance is a valuable property, especially when exploring the underlying structure of complex datasets. The insights gained from analyzing the characteristics of the discovered clusters can inform future research and applications of the AP algorithm.

6.3 Future Directions

To further improve the Affinity Propagation algorithm and expand its applicability, future research could focus on the following areas:

- Developing techniques to address the algorithm's computational limitations, such as high memory usage and slow convergence speed.
- Exploring ways to relax the assumption of spherical cluster shapes, allowing the algorithm to discover clusters with more complex geometries.
- Investigating methods to better interpret and summarize the clustering results, especially when the number of discovered clusters is large.
- Combining the Affinity Propagation algorithm with other techniques, such as dimensionality reduction or ensemble methods, to enhance its performance and robustness.

VII. References

1. Brendan J. Frey; Delbert Dueck (2007). "Clustering by passing messages between data points". Science. 315 (5814): 972–976. Bibcode:2007Sci...315..972F. CiteSeerX 10.1.1.121.3145. doi:10.1126/science.1136800. PMID 17218491. S2CID 6502291.

2. Thavikulwat, Precha. "Affinity Propagation: A Clustering Algorithm for Computer-Assisted Business Simulations and Experiential Exercises." Developments in Business Simulation and Experiential Learning 35 (2014): n. pag.

3. https://www.geeksforgeeks.org/affinity-propagation-in-ml-to-find-the-number-of-clusters/