Neighbor Embedding and Clustering using Spectral Methods

1 INTRODUCTION

In this study, we explored the use of spectral methods for clustering. The primary objective of this work was to implement these algorithms from scratch and compare their performance with commonly used versions available in popular libraries. To facilitate visualization, we first applied the T-SNE algorithm to reduce the dimensionality of the data. We then utilized the resulting reduced dimensions to compute eigenvalues and eigenvectors using spectral methods. These eigenpairs were subsequently used to perform clustering using the k-means algorithm. Our experiments were conducted on two different datasets: the Mnist dataset and the Mnist fashion dataset.

2 PROPOSED METHOD

In this chapter, we present the procedures employed to cluster the data. Dimensionality reduction techniques were initially applied, and the resulting dimensions were used to compute the eigenvalues and eigenvectors. These eigenpairs were subsequently utilized to cluster the data using the k-means algorithm.

2.1 Dimensionality Reduction with PCA and T-SNE

The first goal of this study is to perform dimensionality reduction via neighbor embedding, with the aim of visualizing the data and subsequently calculating eigenpairs. To achieve this, we employed a two-step approach, beginning with dimensionality reduction via Principal Component Analysis (PCA). We then applied the T-Distributed Stochastic Neighbor Embedding (T-SNE) algorithm to project the data onto a two-dimensional space. Utilizing PCA as a preliminary step enabled us to accelerate the computation of T-SNE and potentially enhance the quality of the resulting T-SNE mapping. This is due to the fact that PCA removes noise and redundancy from the data, thereby allowing T-SNE to more effectively capture the essential structure of the data. For the above process we used the corresponding algorithms of the Sklearn library.

2.2 Computing Eigenpairs Using Spectral Methods

Having obtained a two-dimensional T-SNE projection of the data, we proceeded to calculate the eigenvectors. To this end, we implemented the following procedure:

(1) Compute the pairwise distances (W) between the data points using the Euclidean distance metric using the following equation:
$$W_{i,j} = \sqrt{\sum_{k=1}^{n} (a_{i,k} - a_{j,k})^2}$$
(2) Calculate the kernel matrix (K) from the pairwise distances (W) by applying Gaussian Kernel (RBF) using the following equation:
$$K = e^{\frac{-W^2}{2\sigma^2}}$$
(3) Generate the diagonal matrix (D) from the kernel matrix using using the following equation:
$$D = \text{diag}\left(\sum K\right)$$
(4) Derive the Laplacian matrix (L) from the kernel and diagonal matrices using the following equation:
$$L = D - K$$
(5) Calculate the eigenvalues and eigenvectors of the Laplacian matrix using the following equation:
$$\texttt{eigvals, eigvecs} = \texttt{numpy.linalg.eig(L)}$$

The complex eigenpairs we calculated were utilized in two separate parts, the real and imaginary, for our experiments. A key consideration we faced was selecting the appropriate eigenvectors to proceed with k-means clustering. To make this decision, we employed the eigenvalues. In general, the larger the eigenvalue, the greater the deviation of the eigenvectors. Through various experiments, we determined that the optimal eigenvectors corresponded to the smallest eigenvalues. After determining the final set of eigenvectors, we standardized them to make them suitable for use as input for the k-means algorithm.

2.3 K-means Clustering

With the eigenvectors in their final form, we implemented the K-means algorithm for clustering. We wrote the algorithm from scratch, following these steps:

(1) Initialize the centroids of the clusters using the $k-means++$ algorithm. This involves randomly selecting one centroid from the data points and then selecting the remaining centroids using a probability distribution based on the distances of the data points to the nearest centroid.
(2) Assign each data point to the cluster with the nearest centroid.
(3) Compute the mean of the points in each cluster to get the new centroids.
(4) Repeat the process of assigning points to clusters and computing new centroids until convergence, which occurs when the centroids do not change from one iteration to the next.
(5) Return the final centroids and a list of cluster assignments for each data point.

The K-means algorithm is an iterative method that can converge to a local optimum, and the final result may be affected by the initial centroids chosen. To mitigate this, the K-means++ algorithm was utilized to initialize the centroids in order to produce more consistent results when running the algorithm multiple times with the same input data. With this method in place, the algorithm was used to perform different types of clustering and the effects of various parameters were explored. The number of clusters was varied and the results were visualized and analyzed using the silhouette metric.

Project in Computational Intelligence - Statistical Learning, Ai, Auth 2022.

3 EXPERIMENTAL EVALUATION

In this section, we will provide detailed information regarding the datasets employed and the preprocessing steps applied to them. Subsequently, we will present the results of the experiments conducted on both datasets using the algorithms developed by us. We will also compare the results obtained from our algorithms with those of commonly used libraries.

3.1 Data

3.1.1 Mnist. This is a dataset of 60,000 ($28x28$) grayscale images of the 10 digits, along with a test set of 10,000 images. More info can be found at the [1]. Due to the limited memory of our system, we could not use the entire data set but a subset consisting of 15000 samples.

3.1.2 Fashion-Mnist. This is a dataset of 60,000 ($28x28$) grayscale images of 10 fashion categories, along with a test set of 10,000 images. This dataset can be used as a drop-in replacement for Mnist. More info can be found at the [2]. Due to the limited memory of our system, we could not use the entire data set but a subset consisting of 15000 samples.

3.2 Data Preprocessing

Both of the datasets used in this study had the same initial dimensions. To prepare them for input to the models, they were reshaped. Table 1 describes the change in data shapes. In addition, the features were standardized by applying the StandardScaler function from the Sklearn library [3], which removes the mean and scales the features to unit variance. Although the pixel values of the images in both datasets were already within a consistent range ($0 - 255$), standardizing the features was deemed beneficial due to the sensitivity of the used algorithms to feature scale.

Table 1: Data Reshaping

Data	Initial Shape	Processed Shape.
X	(15000, 28, 28)	(15000, 784)
y	(15000, 1)	(15000)

3.3 Experiments on Mnist

In the subsequent section, we will provide a thorough description of the experiments conducted on the Mnist dataset.

3.3.1 PCA + T-SNE. We applied dimensionality reduction techniques to reduce the feature space from 768 to 50 dimensions using principal component analysis (PCA). We then employed t-distributed stochastic neighbor embedding (T-SNE) to further transform the data into 2 dimensions for visualization. Figure Fig. 1 illustrates the results of three dimensionality reduction experiments. In the first experiment, we utilized PCA with 2 components, in the second experiment, we applied T-SNE with 2 components, and in the final experiment, we first applied PCA with 50 components, followed by T-SNE with 2 components. In all plots, the points are color-coded according to their actual labels.

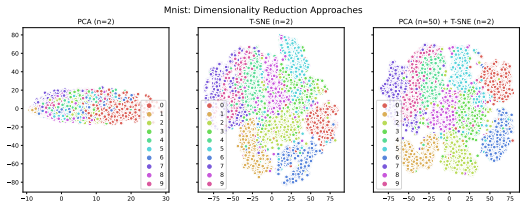

Figure 1: Dimensionality Reduction Approaches on Mnist

As shown in the figure, the approach that combines PCA with T-SNE groups the data more efficiently.

3.3.2 Eigenpairs Computation. After reducing the feature space to two dimensions, we calculated eigenpairs as described above. It is worth noting that in the case of the Mnist dataset, we modified step 2 of the eigenpair calculation, which involves calculating the RBF kernel matrix (K). Specifically, after calculating the pairwise distances, we transformed the resulting array by setting all values below 5 to 0 and all values greater than or equal to 5 to 1. We then continued with the calculation of the diagonal matrix (D), the Laplacian (L), and the eigenpairs using this transformed matrix. This vectorization step was included based on observations from previous studies on the Mnist dataset, which have shown improved performance. From the complex eigenpairs, we retained only the real parts and among the eigenvectors, we only kept those corresponding to the 10 smallest non-zero eigenvalues.

3.3.3 Clustering. Using the final eigenvectors as input for k-means, we performed clustering with a value of $K = 10$. In Fig. 2, we compare the results of our algorithm with those from the Sklearn library. The first plot illustrates the actual grouping of the data, the second plot shows the clustering results obtained from our algorithms, the third plot shows the clustering results obtained from combining our eigenpair calculations with Sklearn's k-means implementation, and finally, the fourth plot displays the results of Sklearn's spectral clustering algorithm. In all of the plots, the x- and y-axes represent the two dimensions obtained from T-SNE for each sample, and the labels indicate the cluster assignment made by the respective algorithm for each sample.

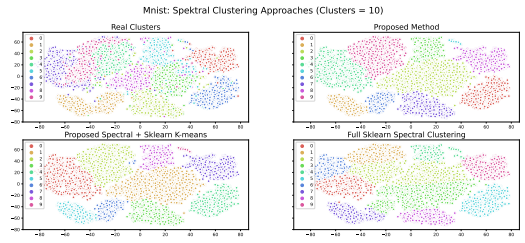

Figure 2: Spectral Clustering Approaches on Mnist

[1] http://yann.lecun.com/exdb/Mnist
[2] https://github.com/zalandoresearch/fashion-Mnist
[3] https://scikit-learn.org/

As illustrated in Fig. 2, the results obtained from the algorithms we implemented are comparable to those produced by Sklearn. To further evaluate the performance of each approach, Table 2 presents the silhouette scores for each method, as well as the respective computational time (reported in seconds) required for each algorithm to execute."

Table 2: Metrics on Mnist

Method	Silhouette	Time
Proposed	0.7979	1289
K-means Sklearn	0.9043	1165
Spectral Clustering Sklearn	0.2248	9

To assess the behavior of our algorithm with respect to the number of clusters, we applied our method to various values of K. Fig. 3 presents the results of 9 experiments where clustering was performed using values of K ranging from 8 to 17, respectively.

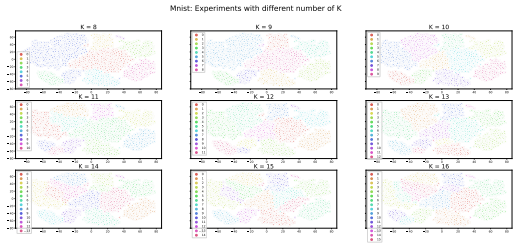

Figure 3: Experiments with different number of K on Mnist

Based on the results presented in Fig. 3, it appears that our algorithm is capable of effectively clustering the data set across a range of values for the number of clusters. The results show consistently clear and well-separated clusters for various K values, indicating that our algorithm is robust to changes in the number of clusters used.

3.4 Experiments on Fashion Mnist

In the next section, we will delve into the experiments conducted on the Fashion Mnist dataset in greater detail. Similar to the Mnist dataset, we employed a similar procedure and as a result, we will not repeat a detailed description of the methodology used. However, we will provide specific information and results specific to the Fashion Mnist dataset.

3.4.1 PCA + T-SNE. As a first step, we applied dimensionality reduction techniques to the Fashion Mnist dataset. Specifically, we used PCA to reduce the number of features from 768 to 200. Subsequently, we employed T-SNE to further reduce the dimensionality to 2 dimensions. Fig. 4 illustrates the impact of the PCA reduction on the subsequent T-SNE algorithm. The figure shows how the visual representation of the dataset changes after PCA and how it can be helpful to improve the T-SNE performance.

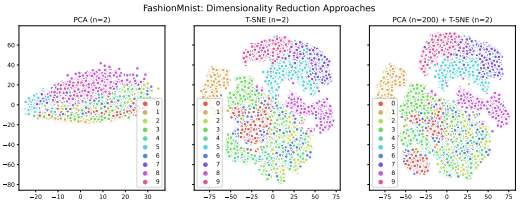

Figure 4: Dimensionality Reduction Approaches on Fashion Mnist

As demonstrated in Fig. 4, the approach of combining PCA with T-SNE appears to cluster the Fashion Mnist dataset more effectively. The use of PCA prior to T-SNE results in a more distinct and clear separation of the clusters in the 2-dimensional space, suggesting that the PCA step helps the T-SNE algorithm to better preserve the local structure of the data.

3.4.2 Eigenpairs Computation. Once the dimensionality of the Fashion Mnist dataset had been reduced to two dimensions, we calculated the eigenpairs. Unlike with the Mnist dataset, for Fashion Mnist, we did not apply a threshold to vectorize the pairwise distances. Instead, we followed the steps outlined in the proposed method section of our study. The (σ) parameter in the RBF Kernel (K) computation step, was determined by setting it equal to 1. We also make experiments by setting (σ) equal to the median of the pairwise distances between data points (W), but this approach didn't work in our case. Additionally, from the complex eigenpairs, we only retained the real parts. Finally, we selected the eigenvectors corresponding to the 10 smallest non-zero eigenvalues for further analysis.

3.4.3 Clustering. Using the final eigenvectors as input for k-means, we performed clustering with a value of K = 10. In Fig. 5, we compare the results of our algorithm with those from the Sklearn library. The plot shows the result of the clustering performed by our algorithms and Sklearn's algorithms.

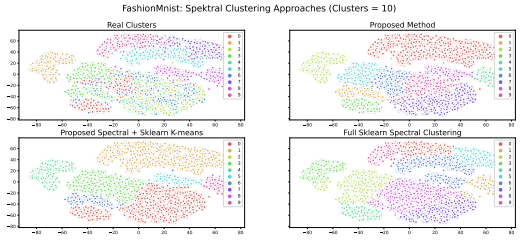

Figure 5: Spectral Clustering Approaches on Fashion Mnist

As can be observed from Fig. 5, the results obtained from the algorithms we implemented for the Fashion Mnist dataset are similar to those produced by Sklearn. To further evaluate the performance of each approach, Table 3 presents the silhouette scores for each method, as well as the respective computational time (reported in seconds) required for each algorithm to execute. This comparison will give us an idea of which algorithm is more efficient or

which one gives us a better trade-off between performance and computational time.

Table 3: Metrics on Fashion Mnist

Method	Silhouette	Time
Proposed	0.7562	636
K-means Sklearn	0.9077	568
Spectral Clustering Sklearn	0.3876	9

To assess the behavior of our algorithm with respect to the number of clusters, we applied our method to various values of K on the Fashion Mnist dataset. Fig. 6 presents the results of 9 experiments where clustering was performed using values of K ranging from 5 to 14, respectively.

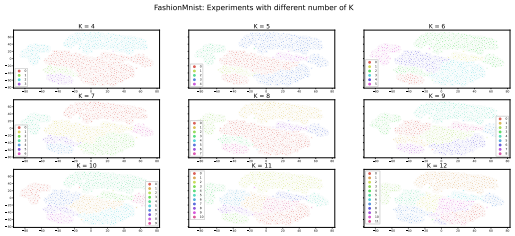

Figure 6: Experiments with different number of K on Fashion Mnist

Based on the results presented in Fig. 6, it appears that our algorithm is capable of effectively clustering the Fashion Mnist dataset across a range of values for the number of clusters. The results show consistently clear and well-separated clusters for various k values, indicating that our algorithm is robust to changes in the number of clusters used. The results also support that our algorithm is capable of effectively clustering the data set regardless of the number of clusters used.