Comparative Analysis of Clustering and Dimensionality Reduction Techniques

1st Zhiling LI
Department of System Design and Intelligent Manufacturing
Southern University of Science and Technology
Shenzhen, China
12112748@mail.sustech.edu.cn

***Abstract*—This report presents a comprehensive study on clustering and dimensionality reduction techniques using the K-means, Soft K-means, PCA, and Linear AutoEncoder methods. We utilize the wheat seed dataset for clustering analysis and compare the performance of these algorithms. Additionally, we explore dimensionality reduction on image data using PCA and Linear AutoEncoder, followed by clustering using Soft K-means. The performance of different methods is compared and analyzed. The report aims to provide insights into the effectiveness of these techniques in handling complex datasets.**

***Index Terms*—K-means, Soft K-means, PCA, Linear Autoencoder, Clustering, Dimensionality Reduction**

I. Introduction

CLUSTERING and dimensionality reduction are two such techniques pivotal in data analysis, reducing complexity for better visualization and understanding. This report focuses on a comprehensive analysis of these techniques, employing K-means and Soft K-means for clustering, alongside Principal Component Analysis (PCA) and Linear AutoEncoder for dimensionality reduction. Using the wheat seed dataset, we conduct an evaluation of the clustering algorithms, aiming to understand their intrinsic data partitioning capabilities. In addition, we apply PCA and Linear AutoEncoder to image data, further assessing the clustering results through Soft K-means with an equivalence of principal component dimensions. This multifaceted approach not only benchmarks the performance of the algorithms but also shows their applicability across different data types.

II. Formulas

A. K-means and Soft K-means

- Optimization method:

$$\min_{\mathbf{m},\mathbf{r}} J(\mathbf{m},\mathbf{r}) = \min_{\mathbf{m},\mathbf{r}} \sum_{n=1}^{N}\sum_{k=1}^{K} r_k^{(n)} \|m_k - x^{(n)}\|^2 \tag{1}$$

- K-means assignment:

$$k^{(n)} = \arg\min_k d(m_k, x^{(n)}) \tag{2}$$

- K-means update:

$$m_k = \frac{\sum_n r_k^{(n)} x^{(n)}}{\sum_n r_k^{(n)}} \tag{3}$$

- Soft K-means assignment:

$$r_k^{(n)} = \frac{\exp(-\beta d(m_k, x^{(n)}))}{\sum_j \exp(-\beta d(m_j, x^{(n)}))} \tag{4}$$

- Soft K-means update:

$$m_k = \frac{\sum_n r_k^{(n)} x^{(n)}}{\sum_n r_k^{(n)}} \tag{5}$$

B. PCA and Linear Autoencoder

- Covariance matrix:

$$C = \frac{1}{N}\sum_{n=1}^{N}(x^{(n)} - \bar{x})(x^{(n)} - \bar{x})^T \tag{6}$$

- Project input vector:

$$z_j = u_j^T x; \quad z = U_{1:M}^T x \tag{7}$$

- Optimization:

$$J(u, z, b) = \sum_n \|x^{(n)} - \tilde{x}^{(n)}\|^2 \tag{8}$$

where the reconstructed $\tilde{x}^{(n)}$ is given by:

$$\tilde{x}^{(n)} = \sum_{j=1}^{M} z_j^{(n)} u_j + \sum_{j=M+1}^{D} b_j u_j \tag{9}$$

- Encoding and decoding:

$$z = f(Wx); \quad \hat{x} = g(Vz) \tag{10}$$

- Optimization:

$$\min_{W,V} \frac{1}{N}\sum_{n=1}^{N}\|x^{(n)} - \hat{x}^{(n)}\|^2 \tag{11}$$

- If g and f are linear, the objective simplifies to:

$$\min_{W,V} \frac{1}{N}\sum_{n=1}^{N}\|x^{(n)} - VWx^{(n)}\|^2 \tag{12}$$

III. Problem Formulation

A. Dataset Description

This project utilizes the Wheat Seed Dataset, comprising 210 data samples. Each sample is described by seven input features, indicative of the geometrical properties of wheat seeds. The dataset also contains labels for three wheat varieties: Kama, Rosa, and Canadian, marked as 1, 2, and 3 respectively. It is crucial to note that, although the labels are provided, they are not employed in the K-means and Soft K-means clustering methods. Instead, these labels serve as a basis for evaluating the performance of the clustering algorithms.

Additionally, the project employs two distinct image types to assess PCA and Linear AutoEncoder in dimensionality reduction. The first is a painting, chosen for its artistic complexity, to test the preservation of artistic features in reduced dimensions. The second is a photograph, selected for its realistic details, providing a contrasting scenario to evaluate these techniques in preserving clarity and realism. The inclusion of both images facilitates a comprehensive analysis of dimensionality reduction methods.

B. K-means and Soft K-means Clustering

The project aims to apply both K-means and Soft K-means clustering methods to the Wheat Seed Dataset. In clustering analysis, the objective is to partition the dataset into groups (or clusters) such that data points in the same cluster are more similar to each other than to those in other clusters.

K-means, a partitional clustering technique, aims to minimize within-cluster variance by iteratively reassigning data points to the nearest cluster centroid and recalculating these centroids.

Soft K-means, an extension of K-means, assigns points to clusters probabilistically, allowing for more nuanced clustering by considering the likelihood of each point belonging to every cluster.

For the initial experiment, we set the number of clusters, K, to 3, reflecting the three varieties of wheat in the dataset. The performance of the clustering methods is assessed without using the label information. Additionally, we experiment with a higher value of K (K=10) to explore the algorithms' behavior in a more complex clustering scenario. We also modify the algorithms by incorporating non-local split-and-merge moves and evaluate their impact.

C. Principal Component Analysis and Linear AutoEncoder

The project investigates dimensionality reduction through Principal Component Analysis (PCA) and Linear AutoEncoder. PCA reorients data into principal components, retaining those characteristics of the dataset that contribute most to its variance. Linear AutoEncoder, utilizing neural networks, compresses and reconstructs data, focusing on minimizing reconstruction error to retain key features.

The project employs these dimensionality reduction techniques on an image dataset and compares their effectiveness in capturing the principal components. The reconstruction quality of both methods is analyzed, and the Soft K-means algorithm is applied for clustering, with K set to the number of dimensions retained by the PCA.

IV. Method and Algorithms

A. K-means Clustering

K-means clustering is a method to partition N observations into K clusters in which each observation belongs to the cluster with the nearest mean. This results in a partitioning of the data space into Voronoi cells.

1) Algorithm Description: The standard K-means algorithm operates through the following iterative process:

i. **Initialization:** Choose K initial cluster centers (centroids) either randomly or based on some heuristic.
ii. **Assignment step:** Assign each data point to the nearest centroid. The 'nearest' is typically determined by the Euclidean distance, forming K partitions of the data.
iii. **Update step:** Recalculate the centroids as the mean of all data points assigned to each cluster, effectively minimizing the within-cluster variance.
iv. Repeat the assignment and update steps until the centroids no longer change significantly, indicating that the clusters have stabilized and the algorithm has converged.

Initial step is set K cluster means $m_1, \ldots, m_K$ to random values. Each data point $x^{(n)}$ is assigned to the nearest mean:

$$k^{(n)} = \arg\min_k d(m_k, x^{(n)})$$

Adjust the model parameters, means are updated to match the sample means of data points they are responsible for:

$$m_k = \frac{\sum_n r_k^{(n)} x^{(n)}}{\sum_n r_k^{(n)}}$$

The objective function for K-means clustering is defined as the sum of squared distances between each data point and its assigned cluster centroid, which K-means aims to minimize:

$$J = \sum_{k=1}^{K} \sum_{x \in S_k} ||x - \mu_k||^2$$

where S_k is the set of data points assigned to cluster k and μ_k is the centroid of cluster k.

2) Convergence: The standard K-means algorithm guarantees convergence to a solution by iteratively reducing within-cluster variance. However, this solution may correspond to a local minimum of the objective function.

3) Optimized centroid initialization method: The algorithm attempts to use **the optimized centroid initialization method**: Randomly determine the first centroid, and loop to find the centroid farthest from all current centroids. This method of initialization **based on distance** effectively optimizes the algorithm effect.

4) Non-local split and merge: To potentially improve the convergence to a global minimum and avoid local minima, non-local split and merge strategies can be employed. By allowing clusters to split and merge, the algorithm can adapt better to the inherent structure of the data and potentially find a more optimal partitioning.

Algorithm 1 K-means Clustering Algorithm

Require: X, the dataset
Require: K, the number of clusters
Require: $max\_iterations$, the iteration maximum limit
  Initialize $centroids$ randomly the dataset X
  // or initialize $centroids$ by largest distance between them
  for $i = 0$ to $max\_iterations$ **do**
    Assign each point in X to the nearest centroid
    Recalculate centroids as the mean of all points assigned to each centroid
    if centroids do not change **then**
      break
    end if
  end for

B. Soft K-means Clustering

Soft K-means clustering is a variant of the K-means algorithm that allows for a probabilistic assignment of data points to clusters, rather than a hard assignment. This method provides a more flexible clustering approach by incorporating the degree of membership for each data point in relation to each cluster.

1) Algorithm Description: The Soft K-means algorithm involves the following steps:

i. **Initialization:** Start by setting K means $\{m_k\}$ to random values, or by the optimized centroid initialization method based on distance.

ii. **Assignment:** Assign each data point $x^{(n)}$ a "degree of assignment" to each cluster mean m_k, reflecting the point's responsibility towards the cluster. The responsibility $r_k^{(n)}$ is calculated as follows:

$$r_k^{(n)} = \frac{\exp(-\beta \cdot d(m_k, x^{(n)}))}{\sum_j \exp(-\beta \cdot d(m_j, x^{(n)}))}$$

where $d(\cdot, \cdot)$ denotes a distance metric between two points, typically the Euclidean distance, and β is a parameter controlling the "softness" of the assignment.

iii. **Update:** Adjust the cluster means based on the responsibilities and the data points:

$$m_k = \frac{\sum_n r_k^{(n)} x^{(n)}}{\sum_n r_k^{(n)}}$$

iv. Repeat the assignment and update steps until convergence is achieved, indicated by minimal changes in responsibilities or means.

2) Convergence: Similar to K-means, Soft K-means is an iterative algorithm that converges to a set of means and responsibilities that locally minimize the objective function. However, due to the "soft" assignment of points to clusters, the algorithm may be more robust to the initial placement of centroids and can potentially reveal overlapping clusters.

3) Responsibilities: The responsibilities $r_k^{(n)}$ represent the extent to which a data point $x^{(n)}$ is associated with a particular cluster k. Unlike hard assignments, these responsibilities allow for a data point to influence the position of multiple cluster means.

4) Softness Parameter β: The parameter β controls the "sharpness" of the cluster boundaries. A high value for β leads to almost hard assignments (similar to standard K-means), while a low value allows for more overlap between clusters, capturing the data's uncertainty and nuances.

Algorithm 2 Soft K-means Clustering Algorithm

Require: X, the dataset
Require: K, the number of clusters
Require: $max\_iterations$, the iteration maximum limit
Require: $beta$, the stiffness parameter
  Initialize $centroids$ from the dataset X
  for $i = 0$ to $max\_iterations$ **do**
    for each point $x^{(n)}$ in X **do**
      for each centroid m_k **do**
        Calculate $d(m_k, x^{(n)})$ as the distance between $x^{(n)}$ and m_k
        $r_k^{(n)} \leftarrow \exp(-beta \cdot d(m_k, x^{(n)}))$
      end for
      Normalize $r_k^{(n)}$ so that $\sum_k r_k^{(n)} = 1$
    end for
    for each centroid m_k **do**
      $m_k \leftarrow \frac{\sum_n r_k^{(n)} x^{(n)}}{\sum_n r_k^{(n)}}$
    end for
    if centroids do not change significantly **then**
      break
    end if
  end for

C. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

1) Algorithm Description: The PCA algorithm can be summarized as follows:

i. **Covariance Matrix:** Compute the covariance matrix C of the data set as:

$$C = \frac{1}{N} \sum_{n=1}^{N} (x^{(n)} - \bar{x})(x^{(n)} - \bar{x})^T$$

where $\bar{x}$ is the mean vector of the data points.

ii. **Eigendecomposition:** Perform eigendecomposition on the covariance matrix C to obtain the matrix of eigenvectors U and the diagonal matrix of eigenvalues Σ:

$$C = U \Sigma U^T$$

where U has columns that are unit-length eigenvectors and is orthogonal, meaning that $U^T U = U U^T = I$.

iii. **Selection of Components:** Select the top M eigenvectors corresponding to the largest eigenvalues to capture most of the variance in the data.

iv. **Projection:** Project the data onto the lower-dimensional subspace spanned by the selected eigenvectors:

$$z_j = u_j^T x; \quad z = U_{1:M}^T x$$

2) Minimizing Reconstruction Error: PCA can also be viewed as a way to minimize the reconstruction error between the data and its projection onto a lower-dimensional space:

$$J(u, z, b) = \sum_n \|x^{(n)} - \tilde{x}^{(n)}\|^2$$

where the reconstructed data point $\tilde{x}^{(n)}$ is given by:

$$\tilde{x}^{(n)} = \sum_{j=1}^{M} z_j^{(n)} u_j + \sum_{j=M+1}^{D} b_j u_j$$

The objective is minimized when the components chosen are the eigenvectors associated with the largest eigenvalues, which represent the directions of maximum variance in the data.

The principal components provide a means of understanding the underlying structure of the data, capturing as much of the data's variability as possible, and reducing the dimensionality without losing significant information.

Algorithm 3 Standard PCA Algorithm

Require: X, the dataset
Require: M, the number of principal components to select
Ensure: Z, the projected data; V_M, the selected eigenvectors
$\mu \leftarrow \text{mean}(X)$
$X_{centered} \leftarrow X - \mu$
$C \leftarrow \frac{1}{N} \times (X_{centered}^T \times X_{centered})$
$[V, D] \leftarrow \text{eig}(C)$
Sort eigenvalues and eigenvectors in descending order
$V_M \leftarrow$ top M eigenvectors
$Z \leftarrow X_{centered} \times V_M$
return Z, V_M

D. Linear AutoEncoder

A Linear AutoEncoder is a type of neural network used for learning a compressed representation of data. It consists of two main components: an encoder and a decoder.

1) Encoder and Decoder: The encoder maps the input data $\mathbf{x}$ into a lower-dimensional representation $\mathbf{z}$, and the decoder attempts to reconstruct the input data from this compressed form $\hat{\mathbf{x}}$. Mathematically, the encoder and decoder functions are defined as:

$$\mathbf{z} = f(\mathbf{W}\mathbf{x}) \tag{13}$$

$$\hat{\mathbf{x}} = g(\mathbf{V}\mathbf{z}) \tag{14}$$

where $\mathbf{W}$ and $\mathbf{V}$ are weight matrices for the encoder and decoder, respectively.

2) Objective: The goal of the AutoEncoder is to minimize the reconstruction error, which is the difference between the original input $\mathbf{x}$ and the reconstructed input $\hat{\mathbf{x}}$. The reconstruction error for all N data points is given by:

$$\min_{\mathbf{W},\mathbf{V}} \frac{1}{2N} \sum_{n=1}^{N} ||\mathbf{x}^{(n)} - \hat{\mathbf{x}}^{(n)}||^2 \tag{15}$$

3) Linear AutoEncoder: If the functions f and g are linear, the AutoEncoder essentially performs the same operation as PCA. The encoder becomes a linear transformation that projects the data onto a lower-dimensional subspace, and the decoder reconstructs the data from this subspace. The objective simplifies to:

$$\min_{\mathbf{W},\mathbf{V}} \frac{1}{2N} \sum_{n=1}^{N} ||\mathbf{x}^{(n)} - \mathbf{V}\mathbf{W}\mathbf{x}^{(n)}||^2 \tag{16}$$

where $\mathbf{W}$ and $\mathbf{V}$ are now the projection matrix and the reconstruction matrix, respectively.

4) Training: During training, the AutoEncoder learns the optimal weights $\mathbf{W}$ and $\mathbf{V}$ that minimize the reconstruction error. For a Linear AutoEncoder, this training process results in learning the principal components of the data.

Algorithm 4 Linear AutoEncoder

Require: X, the input data matrix where each column is a data point
Require: η, the learning rate
Require: $epochs$, the number of iterations for training
Initialize W and V with small random values or using a specific initialization strategy
for $epoch = 0$ **to** $epochs$ **do**
for each data point $x^{(n)}$ in X **do**
// Encoding step
$z^{(n)} \leftarrow Wx^{(n)}$
// Decoding step
$\hat{x}^{(n)} \leftarrow Vz^{(n)}$
// Calculate reconstruction error
$error \leftarrow x^{(n)} - \hat{x}^{(n)}$
// Update the weights
$d\_loss\_X \leftarrow \frac{2}{N} \cdot error$
$d\_loss\_Z \leftarrow d\_loss\_X \cdot W2$
$W_i \leftarrow W + \eta \cdot d\_loss\_W_i$
$b_i \leftarrow V + \eta \cdot d\_loss\_b_i$
end for
// Optionally add a condition to break the loop if the error is below a threshold
end for
return W_2, b_2, W_1, b_1

V. Experiment Results and Analysis

A. K-means Implementation Testing and Analysis

In the implementation of the K-means class, we explored several optimization methods:

Optimizing the method of initializing cluster centroids. We created a **_largest_distance** method to calculate the point that is farthest from the current list of centroids and chose this as the next centroid, until the list of K centroids was completed.

The test results showed that this optimized initialization method effectively avoided clustering errors when random initialization could lead to inaccurate classifications.

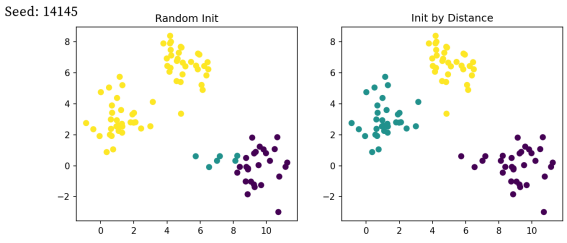

Fig. 1. Optimized K-means clustering with improved initialization method.

To potentially improve convergence to the global minimum and avoid local minima, **non-local split and merge strategies** were employed. These strategies included splitting the largest cluster into two based on certain calculated metrics and merging the two closest clusters. Allowing clusters to split and merge enabled the algorithm to better adapt to the inherent structure of the data and potentially find a more optimized partitioning.

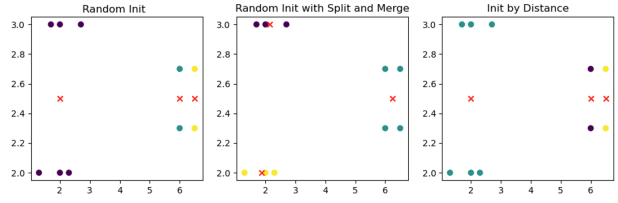

Fig. 2. K-means clustering after non-local split and merge strategies.

B. Soft K-means Implementation Testing and Analysis

The optimization methods mentioned above were still effective in the implementation of the SoftKMeans class:

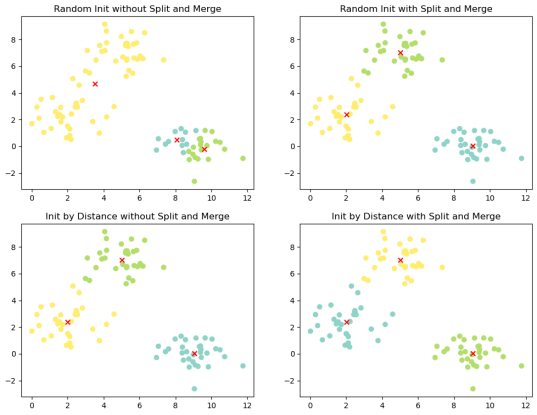

Fig. 3. Soft K-means clustering with different strategies.

Testing the impact of different Beta parameters on the results showed that as the Beta value decreased, the cluster assignments became "softer," and the clustering centers were more likely to approach each other, resulting in a reduction in the number of distinct clustering centers. A Beta value too low might affect accuracy, while increasing Beta value made the cluster assignments "harder," approaching the behavior of the K-means algorithm.

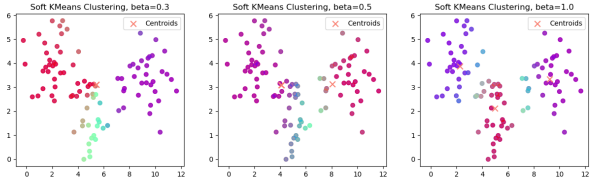

Fig. 4. Soft K-means clustering with different Beta values.

In the above test, in order to compare the classification results, hard assignment was used for the Soft KMeans class during prediction. In order to better demonstrate the effect of the responsibility matrix in the clustering process, responsibility was also applied to visualization to better showcase the effects of Soft KMeans. The test results are shown below:

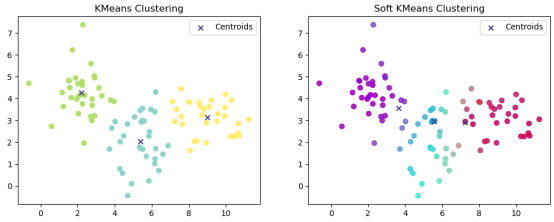

Fig. 5. Visualization of the responsibility matrix in Soft K-means.

C. Clustering Test and Analysis on Wheat Seed Dataset

For the clustering test of the dataset, we constructed test code and serialized the model test result outputs. We first set K to 3 for clustering. We used the $adjusted\_rand\_score$ tool from the $sklearn$ package to analyze prediction accuracy.

When we reconsidered the dataset and set K to 10, we modified both algorithms by adding non-local split and merge moves and then ran the modified algorithms. We built a $momentum$ parameter to assist the analysis. In both the kmeans and softkmeans classes, the calculation of $momentum$ involved replacing the vector coordinates of "point" in the center point calculation method with the Euclidean distance to the center point.

Clustering Results:

TABLE I
CLUSTERING RESULTS FOR K-MEANS WITH $k = 3$

Initialization	Split&Merge	Accuracy Score	Momentum-Sum
Random	No	0.7166	313.2168
Random	Yes	0.7166	313.2168
Optimization	No	0.7103	313.7343
Optimization	Yes	0.7166	313.2168

TABLE II
CLUSTERING RESULTS FOR SOFT K-MEANS WITH $k = 3$

Initialization	Split&Merge	Accuracy Score	Momentum-Sum
Random	No	0.6715	7.7976
Random	Yes	0.6712	7.7882
Optimization	No	0.6715	7.7976
Optimization	Yes	0.6717	7.7646

It can be seen that when the number of k categories is set appropriately ($k = 3$, the true value), the hard clustering approach of KMeans tends to outperform Soft KMeans due to more definitive assignments and less susceptibility to noise.

TABLE III
CLUSTERING RESULTS FOR K-MEANS WITH $k = 10$

Initialization	Split&Merge	Accuracy Score	Momentum-Sum
Random	No	0.3303	193.2378
Random	Yes	0.3224	191.5293
Optimization	No	0.3750	199.1581
Optimization	Yes	0.3112	200.5471

TABLE IV
CLUSTERING RESULTS FOR SOFT K-MEANS WITH $k = 10$

Initialization	Split&Merge	Accuracy Score	Momentum-Sum
Random	No	0.6033	26.0271
Random	Yes	0.6085	26.0117
Optimization	No	0.6033	26.0269
Optimization	Yes	0.6033	26.0266

However, when K is set higher than the actual number of categories ($k = 10$), KMeans' accuracy drops significantly due to the addition of extra categories that do not actually exist, while the effect of Soft KMeans is far better than KMeans. This reflects the better adaptability and dynamic adjustment ability of Soft KMeans' soft assignment to the data point structure.

Furthermore, we can observe that, with the inclusion of non-local split and merge moves, both modified algorithms have the potential for improvement. However, whether improvement occurs is also influenced by random factors.

At the same time, it can be seen that the optimized centroid initialization algorithm (selecting centroids based on the furthest distance principle) can enhance the accuracy of clustering.

D. Dimensionality Reduction Techniques on Image Data

1) PCA and Linear AutoEncoder Processing: In this experiment, an image was processed to discard any transparency channel and transform it into an $N \times 3$ matrix representing RGB values ranging from 0 to 255. Considering the intrinsic nature of the algorithms, the maximum principal component dimensionality for PCA and the Linear AutoEncoder was limited to the channel number, which is three. We tested dimensions $n = 1, 2, 3$ for both PCA and the Linear AutoEncoder.

The PCA and the Linear AutoEncoder yielded results that can be visually compared in the following figures, demonstrating the impact of different dimensionality choices on the reconstructed image quality:

Fig. 6. Comparison of PCA and Linear AutoEncoder processing results with different dimensions.

As the test image used in the above experiment was a work of digital art with less detail in color blocks, a real photograph was chosen for additional test. Moreover, attempt to demonstrate the processing result when $n = 0$. Consistent with the algorithmic prediction, both PCA and the AutoEncoder processed the image to produce a monochromatic picture, where the image's color represents the mean of the original photograph's colors. The comparative observation is illustrated in the figure below:

Fig. 7. Comparison of PCA and Linear AutoEncoder on a real photograph.

The training of the Linear AutoEncoder on a real photograph was conducted with different principal component dimensions $n = 1, 2, 3$. The corresponding loss curves for each dimension setting are depicted in Figure 8. As seen, the loss sharply decreases in the initial epochs, indicating a rapid learning phase, followed by a plateau, suggesting convergence. The difference in the scale of loss values across different

dimensions emphasizes the variation in reconstruction error, with $n = 3$ having the most significant initial loss reduction, indicative of capturing more complex features in higher dimensions.

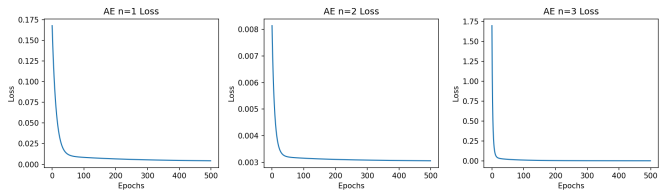

Fig. 8. Loss curves of the Linear AutoEncoder for $n = 1, 2, 3$ when applied to a real photograph, showing the training progression over epochs.

2) Clustering Results with Soft KMeans: Utilizing Soft KMeans, we matched the clustering k values to the principal component dimensions used in PCA. The clustering results are depicted below, showing the effect of color compression in the images:

Fig. 9. Clustering results with $k = 1, 2, 3$ corresponding to PCA dimensions.

Given that PCA and Soft KMeans have different processing effects for $n \geq 3$ and $k \geq 3$, an additional set of images with larger k values was analyzed to observe the variation in the number of color values:

Fig. 10. Clustering results using Soft KMeans with larger k values.

Testing with a real photograph revealed more pronounced noise after clustering. The effect of color compression is significantly more observable:

Fig. 11. Clustering results with $k = 1, 2, 3$ corresponding to PCA dimensions on a real photograph.

VI. Conclusion and Future Problems

This study provided a comparative analysis of clustering and dimensionality reduction techniques, revealing distinct characteristics and performance metrics across various algorithmic implementations. K-means exhibited robust clustering capabilities with definitive assignments, while Soft K-means demonstrated flexibility in handling data points. Dimensionality reduction techniques, PCA and Linear AutoEncoder, proved their efficacy in image data compression while maintaining the integrity of the original data to a commendable extent.

Future research may delve into hybrid models that combine the strengths of these algorithms, potentially enhancing performance. Additionally, exploring the integration of deep learning models for unsupervised tasks could offer new perspectives in clustering and dimensionality reduction. Lastly, extending these techniques to other complex data types such as time series or genomic data could further validate their versatility and applicability in broader data science contexts.

References

[1] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.