Bereket Eshete Kebede
UID - U00827234

M.S. Student
University of Memphis, Fall 2022
COMP 7745 Machine Learning

Implement Dimensionality Reduction using PCA and Clustering using Unsupervised methods Implement Dimensionality Reduction using Auto-encoder and Clustering using Unsupervised methods

Instructor: Prof. Zahangir Alom
Assignment #5 Report

Table of contents

Implement Dimensionality Reduction using PCA and Clustering using Unsupervised methods

1. Introduction

In this section, we will apply PCA on high-dimensional dataset MNIST and reduce them to low dimensions (feature embedding). We will use python effectively to load our dataset, apply PCA on them, and prepare our data for future machine learning models. After reducing the dimension we will use k-means and t-distributed Stochastic Neighbor embedding (tSNE) unsupervised methods for clustering and visualization of input samples.

2. Methodology & Model descriptions

One of the most often used methods of linear dimension reduction is principal component analysis (PCA). It can be used both on its own and as a starting point for further dimension reduction techniques. By projecting the data onto a collection of orthogonal axes, the PCA method alters the data.

For visualizing the feature embedded data using PCA we will use K-means. K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. The objective of K-means is to group similar data points together and discover underlying patterns. To achieve this objective, K-means looks for a fixed number (k) of clusters in a dataset. Another approach for clustering data is t-SNE (Stochastic Neighbor Embedding). t-SNE is a tool to visualize high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence (KL divergence) between the joint probabilities of the low-dimensional embedding and the high-dimensional data.

3. Experiment and Results

For our experiment, we will apply Principal Component Analysis (PCA) on the MNIST dataset to reduce the dimensionality to 50-dimensional space. Then apply K Means clustering and tSNE on the 50-dimensional feature embedding space for clustering and visualization. To achieve PCA and clustering, we are using Tensorflow 2.7 a machine learning python package tool, sklearn python library and NVIDIA TITAN RTX GPU for the computation environment.

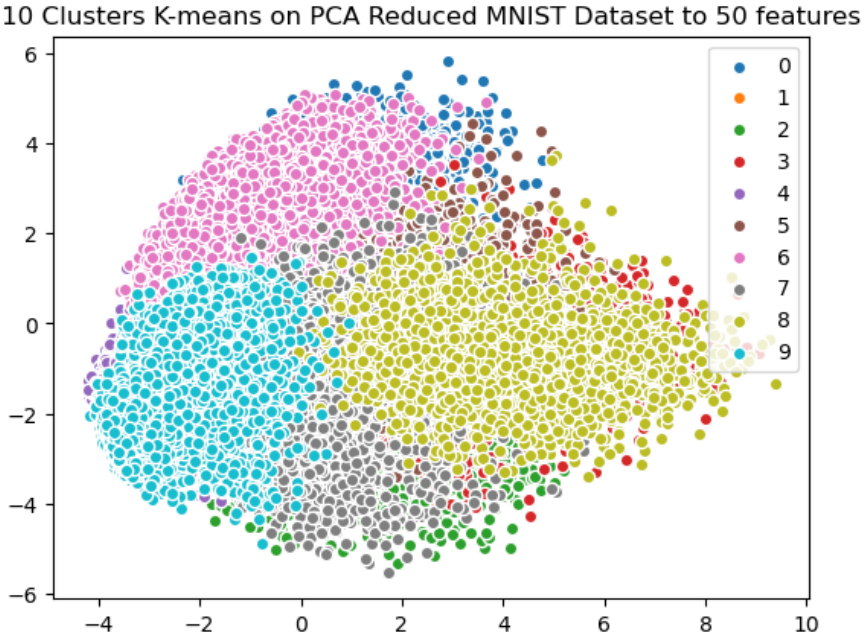

Figure 1. K-means clustering on PCA reduced MNIST dataset to 50 features with k =10.

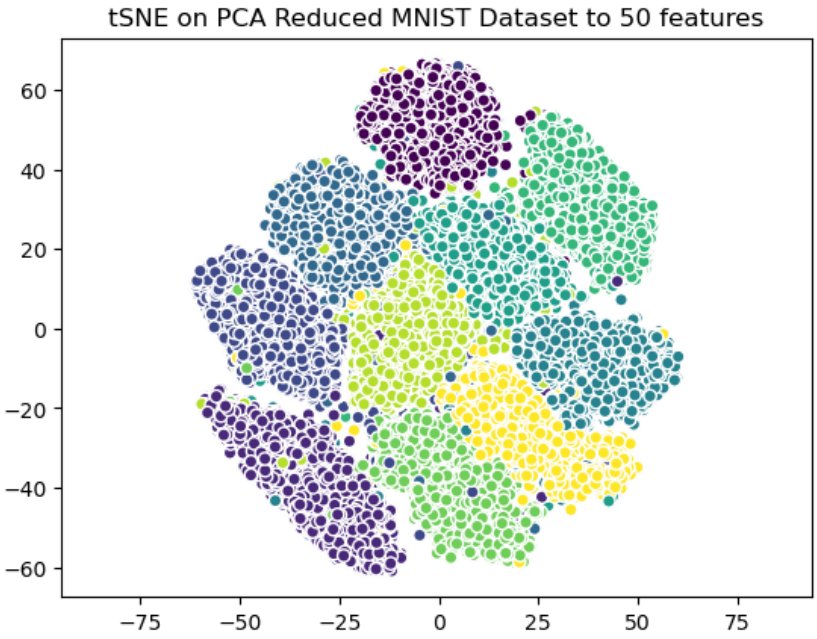

Figure 2. tSNE clustering on PCA reduced MNIST dataset to 50 features, n =2.

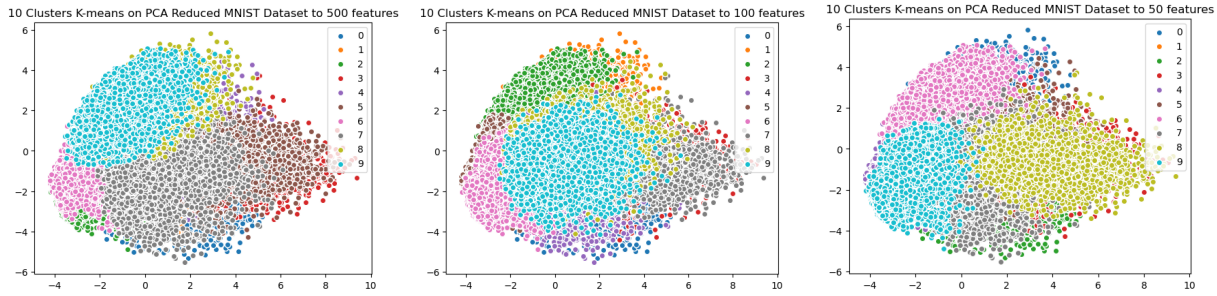

Figure 3. PCA feature embedding for 500, 100 and 50 features.

(a) Database

Dataset : MNIST (MNIST_train.csv)

The MNIST dataset is the “Hello World” for Computer Vision and a great starter for Convolutional Neural Networks. It is a sizable database of handwritten numbers called the Modified National Institute of Standards and Technology database that is frequently used to train different image processing techniques. The database is also frequently used for machine learning training and testing.

Data preparation:

1. Read the samples from MNIST_train.csv
3. Normalize the input samples before applying the dimensionality reduction and clustering approaches.

(b) Training and testing logs

Training and testing code be found it the following link:
https://github.com/bereketeshete/Machine-Learning-Projects/tree/main/Homework%205

(c) Discussion and comparison

While it is simple for us to visualize two or three dimensional data, it is far more difficult for us to visualize high dimensional data once it exceeds three dimensions. Visualization is quite difficult when we need to examine and detect patterns on datasets of dozens or even millions of dimensions. However, dimensionality reduction is a method that can undoubtedly aid in our understanding of the data. Figure 1 illustrates using K-means clustering approach on PCA reduced MNIST dataset to 50 features with k=10, whereas figure 2 illustrates using tSNE clustering on PCA reduced MNIST dataset to 50 features with n =2. These two clustering methods demonstrate that reducing the dimension from 784 to 50 and clustering them helps us visualize the MNIST data pattern.

4. Conclusion

In summary, we used PCA to reduce the high-dimensional MNIST dataset to low dimensions (feature embedding). We efficiently imported our dataset into Python, ran PCA on it, and then prepared our data for upcoming machine learning models. After reducing the dimension, we clustered and visualized the input samples using k-means and t-distributed Stochastic Neighbor embedding (tSNE) unsupervised methods.

5. References

[1]https://medium.com/@samdrinkswater/sequential-model-for-mnist-dataset-using-tensorflow-25e1fab87b48
[2] https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
[3] https://www.kaggle.com/code/parulpandey/visualizing-kannada-mnist-with-t-sne/notebook
[4]https://towardsdatascience.com/dimensionality-reduction-using-t-distributed-stochastic-neighbor-embedding-t-sne-on-the-mnist-9d36a3dd4521

Implement Dimensionality Reduction using Auto-encoder and Clustering using Unsupervised methods

1. Introduction

An autoencoder's job is to represent the data in smaller dimensions while attempting to capture the most significant features and structures in the original data. One neural network that is used in the dimension reduction technique is called AutoEncoder. A lower-dimensional representation that can replicate the original data can be learned by training a neural network to output the same value as the input using an intermediate layer with a dimension smaller than the input dimension. Autoencoders develop efficient data compression and decompression techniques on their own. This means that dimensionality reduction can make use of autoencoders. Images can also be produced from a noise vector using the Autoencoder's Decoder sections. Practical applications of an Autoencoder network include: Denoising, Image Reconstruction, Image Generation and Data Compression & Decompression.

2. Methodology & Model descriptions

To achieve dimension reduction we will use an autoencoder and use clustering for visualization, we are using Tensorflow 2.7 machine learning python package tool and NVIDIA TITAN RTX GPU. A typical auto-encoder is shown in the figure below.

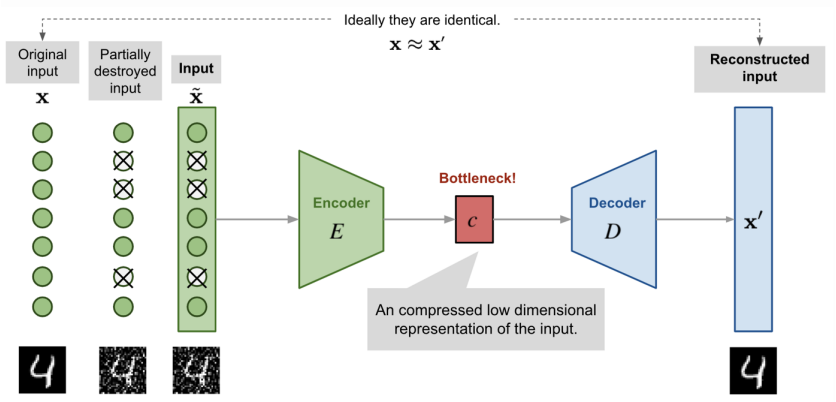

Figure 1. A typical auto-encoder

An autoencoder mainly consists of three main parts; 1) Encoder, which tries to reduce data dimensionality. 2) Code, which is the compressed representation of the data. 3) Decoder, which tries to revert the data into the original form without losing much information.

784→ 500→200→100→50 →100→200→500 →784

Figure 2. Auto-enoder architecture to implement for our experiment

3. Experiment and Results

For our experiment, we will apply an auto-encoder unsupervised method on the MNIST dataset to reduce the dimensionality to 50-dimensional space. Then apply K Means clustering and tSNE on the 50-dimensional feature embedding space for clustering and visualization. To achieve auto-encoder and clustering, we are using Tensorflow 2.7 a machine learning python package tool, sklearn python library and NVIDIA TITAN RTX GPU for the computation environment.

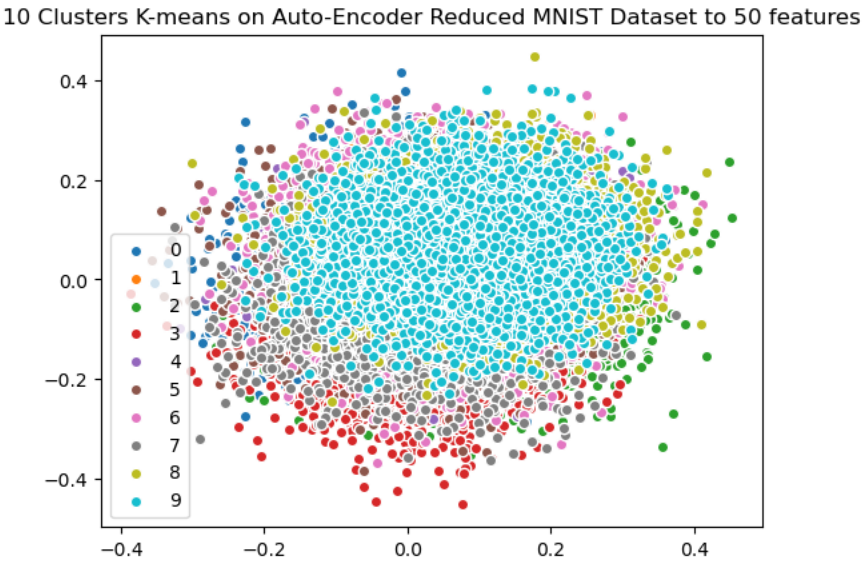

Figure 1. K-means clustering on Auto-encoder reduced MNIST dataset to 50 features with k =10.

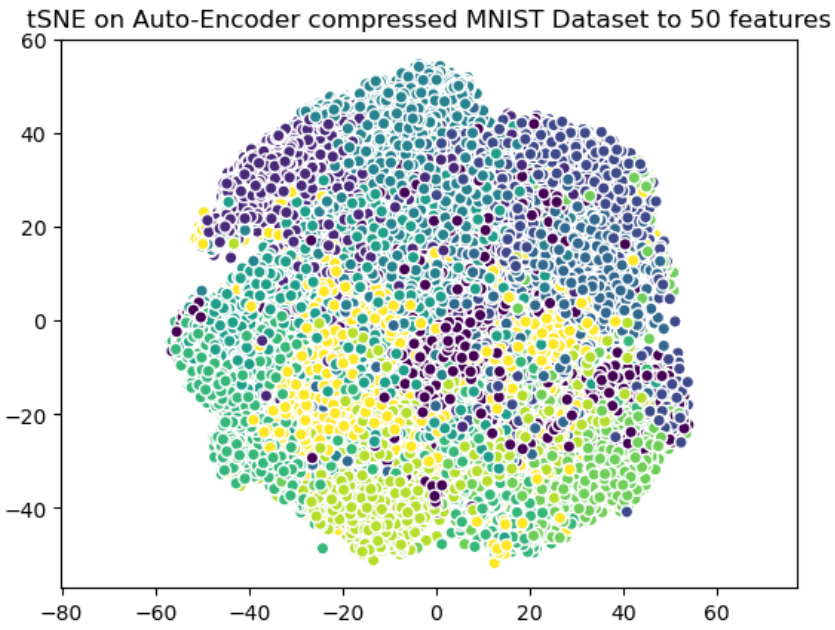

Figure 2. tSNE clustering on Autoencoder reduced MNIST dataset to 50 features, n =2.

(a) Database

Dataset : MNIST (MNIST_train.csv)

Data preparation:

1. Read the samples from MNIST_train.csv
3. Normalize the input samples before applying the dimensionality reduction and clustering approaches.

(b) Training and testing logs

Training and testing code and logs can be found it the following link:
https://github.com/bereketeshete/Machine-Learning-Projects/tree/main/Homework%205

(c) Discussion and comparison

Encoder and decoder sub-models make up an autoencoder. The input is compressed by the encoder, and the decoder makes an effort to reconstruct the input from the encoder's compressed form. The encoder can then be used as a data preparation approach to extract features from raw data so that a separate machine learning model can be trained on it.

Figure 1 illustrates using K-means clustering approach on PCA reduced MNIST dataset to 50 features with k =10, whereas figure 2 illustrates using tSNE clustering on PCA reduced MNIST dataset to 50 features with n =2.

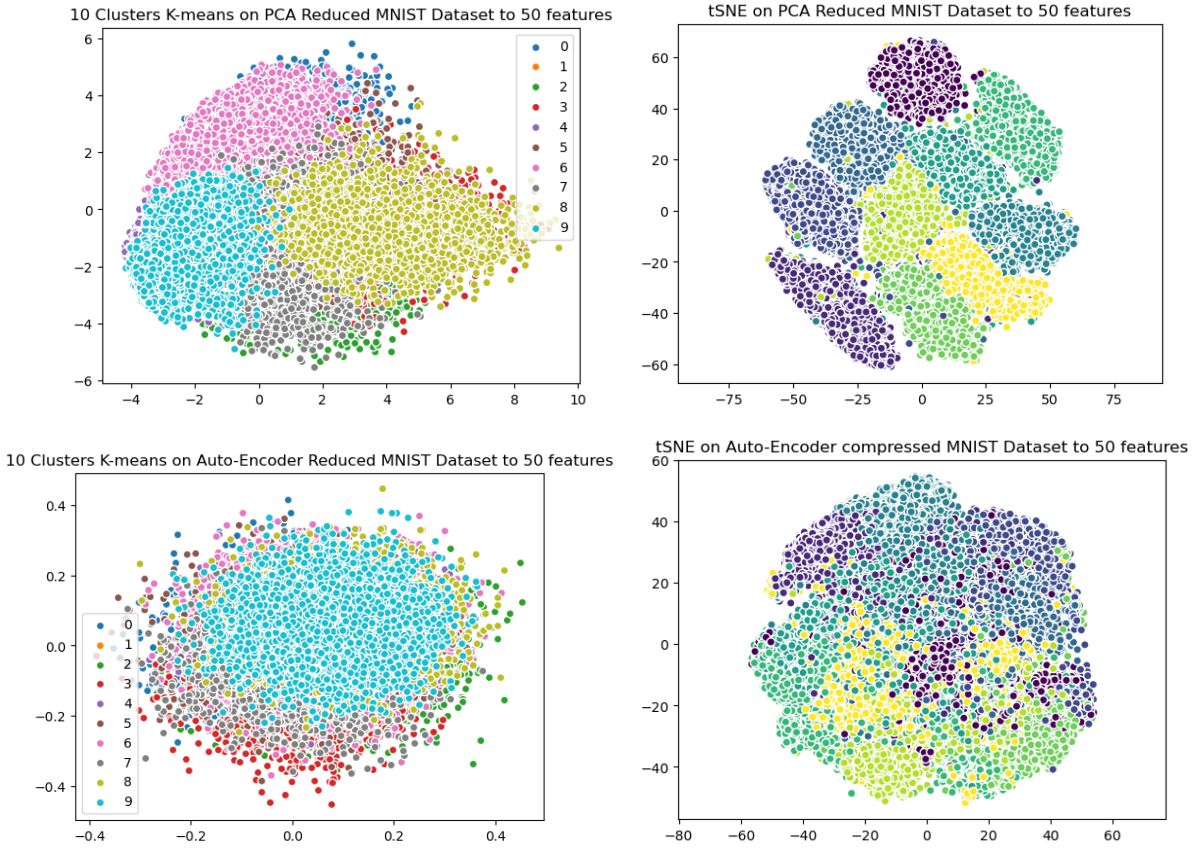

Figure 6. Comparing between pca vs auto-encoder dimension reduction.

From figure 6, we can observe that PCA feature embedding with tSNE clustering gives the best visual result as each cluster is more distinct compared to the remaining three. We implemented t-SNE using sklearn on the MNIST dataset. However, tSNE has its limitations, such as the cost function of t-SNE is non-convex, meaning there is a possibility that we would be stuck in a local minima.

4. Conclusion

In summary, we used an autoencoder to reduce the high-dimensional MNIST dataset to low dimensions (feature embedding). We efficiently imported our dataset into Python, ran an autoencoder on it, and then prepared the MNIST data for upcoming machine learning models. After reducing the dimension, we clustered and visualized the input samples using k-means and t-distributed Stochastic Neighbor embedding (tSNE) unsupervised methods.

5. References

[1]https://medium.com/@samdrinkswater/sequential-model-for-mnist-dataset-using-tensorflow-25e1fab87b48
[2]https://stackoverflow.com/questions/57875581/how-to-get-output-from-a-specific-layer-in-keras-tf-the-bottleneck-layer-in-aut
[3] https://machinelearningmastery.com/autoencoder-for-classification