Project: Dimensionality Reduction and Visualization

Spring 2025

Ezgi Sena Karabacak
Date: April 8, 2025

Introduction

In this project, we investigate the performance of dimensionality reduction techniques for classification tasks using the Fashion-MNIST dataset [1]. The dataset consists of 10,000 grayscale images of size 28×28, representing 10 different clothing categories with 1,000 samples per class. Each image is represented as a flattened vector of 784 features. The objective is to reduce the dimensionality of the data using Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and t-distributed Stochastic Neighbor Embedding (t-SNE), and to analyze the impact of dimensionality on classification accuracy using a Gaussian classifier.

The goal of the project is to assess how reducing the dimensionality of the input data affects classification accuracy when using a quadratic Gaussian classifier. To this end, the following steps are carried out:

- Preprocessing the dataset by splitting it into training and test sets (50% each) using a fixed random seed.
- Applying PCA to learn an optimal set of orthogonal basis vectors and analyzing the effect of varying subspace dimensions on classification performance.
- Using LDA to find class-discriminative linear projections, constrained by the number of classes, and evaluating classification outcomes in lower-dimensional spaces.
- Employing t-SNE to visualize the global structure of the dataset in two dimensions, highlighting how well-separated the classes appear after non-linear embedding.

For each technique, we analyze classification error on both the training and test sets to understand underfitting, overfitting, and the effects of dimensionality. All experiments are implemented in Python using reliable scientific computing libraries, and proper citations are provided for any external tools used. Through these analyses, we aim to gain a deeper understanding of the trade-offs involved in dimensionality reduction and its impact on the separability of classes in high-dimensional data.

Tools Used: All experiments and visualizations were implemented in Python using `NumPy` for numerical computations and `Matplotlib` for plotting.

Data Preprocessing

We first load the Fashion-MNIST dataset. The dataset is split into features and corresponding labels. To visualize an example, we select the 100$^{\text{th}}$ image and display it after reshaping it into a 28×28 matrix. An example image from the dataset is shown in Figure 1.

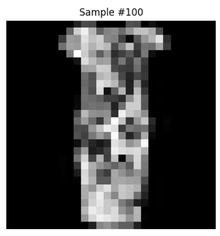

Figure 1: Example image from the Fashion-MNIST dataset.

The dataset is divided class-wise, with 50% of the samples from each class allocated to the training set and the remaining 50% to the test set. This results in 5,000 training and 5,000 test samples. The final data arrays are saved as compressed `.npz` files for further processing.

The preprocessing steps ensure that the subsequent experiments in PCA, LDA, and t-SNE are performed on consistent and reproducible datasets with balanced class distributions.

Principal Component Analysis (PCA)

In this section, we apply Principal Component Analysis (PCA) to reduce the dimensionality of the Fashion-MNIST dataset and examine its effect on classification performance. PCA transforms the original 784-dimensional feature vectors into a lower-dimensional space while preserving as much variance as possible.

Step 1: Centering the Data

The training data was centered by subtracting the mean vector computed across all samples. Figure 2 shows the sample mean image of the training data before centering. We can almost see the silhouettes of some clothing items like pants in this average image.

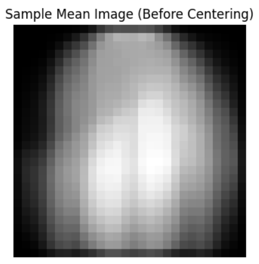

Figure 2: Sample mean image (before centering).

Step 2: PCA and Eigenvalue Analysis

After centering the data, we computed the covariance matrix and extracted its eigenvalues and eigenvectors. The eigenvalues represent the amount of variance captured by each principal component. Figure 3 plots the eigenvalues in descending order.

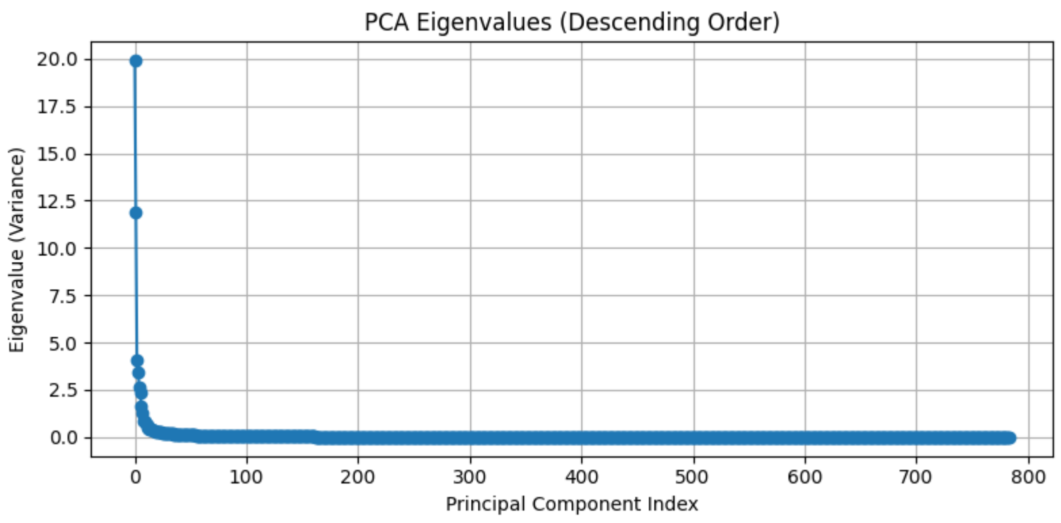

Figure 3: Eigenvalues of the covariance matrix in descending order.

From the plot, we observe that the first few principal components explain most of the variance. After around 40 components, the eigenvalues drop sharply and begin to level off, indicating that higher components contribute very little additional information. Based on this, we can reasonably limit the dimensionality to under 50 components for most analyses.

Step 3: Visualizing PCA Bases

The top 20 eigenvectors (principal components) were reshaped and visualized as images in Figure 4. These bases correspond to directions in the data space along which variance is maximized.

The first few bases capture high-level structures such as clothing outlines and texture patterns, while later bases become more noisy. In the first few images, we can clearly see the outlines of some clothing items, whereas when the order increases, images become more indistinguishable. This aligns with expectations, as the initial components represent major variance directions shared between multiple classes.

Step 4: Dimensionality Reduction and Classification

We projected both the training and test datasets onto subspaces of increasing dimension, ranging from 1 to 350 (with 25 logarithmically spaced values). A Gaussian classifier was trained in each subspace using class-specific maximum likelihood estimates of the mean and covariance matrices. Classification errors were computed for both training and test sets.

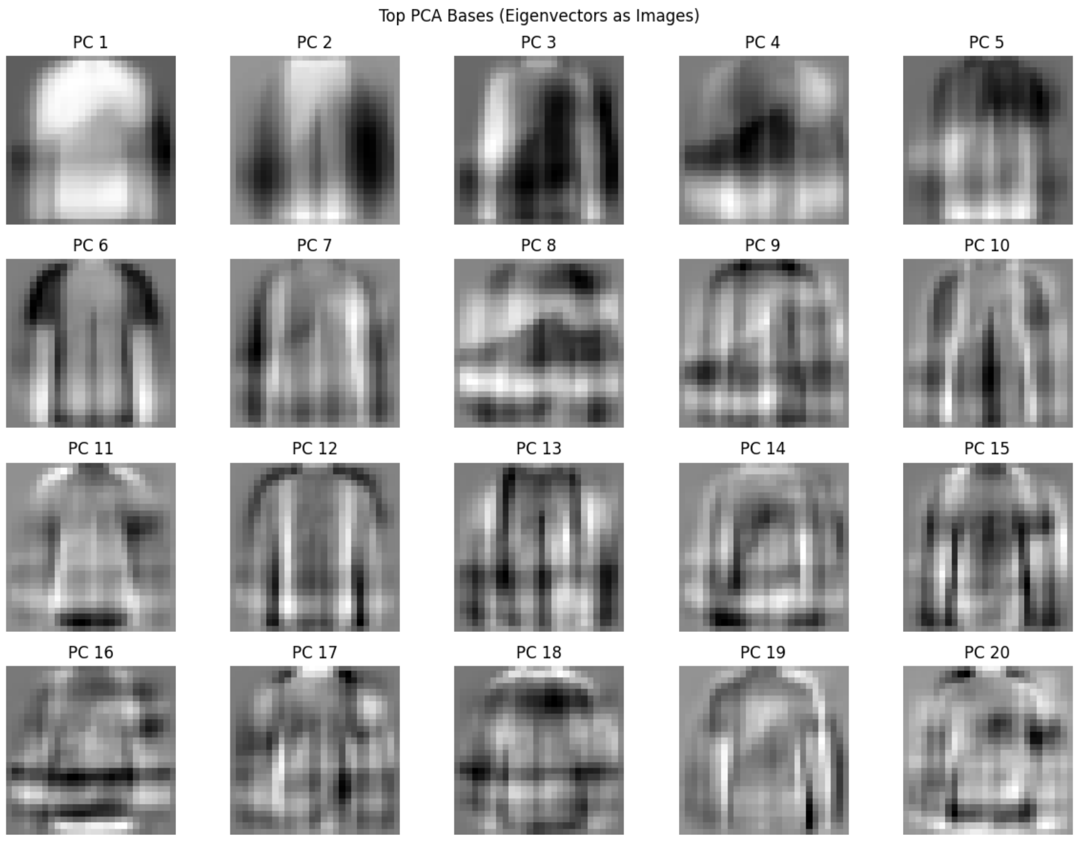

Figure 4: Top 20 PCA basis vectors (eigenvectors) visualized as 28×28 images.

Step 5: Classification Error vs. Dimension

Figure 5 shows the classification error for both the training and test sets as a function of the number of principal components used.

As expected, both training and test errors decrease as more components are included. The training error drops steadily, approaching zero as dimensionality increases. The test error initially decreases and reaches its minimum around 50–100 components. After this point, additional components slightly increase the test error due to overfitting to noise and less relevant features.

Discussion: PCA effectively reduces data dimensionality while retaining critical structure for classification. There is a clear trade-off between dimensionality and generalization. Very low-dimensional representations underfit the data, while very high-dimensional spaces may lead to overfitting. A moderate number of components (e.g., 50–100) provides an optimal balance.

Tools Used: Python was used for all computations, using `NumPy` and `Matplotlib`. Gaussian classification was implemented using `scipy.stats.multivariate_normal`. All PCA steps were performed manually to ensure full understanding and control over the pipeline.

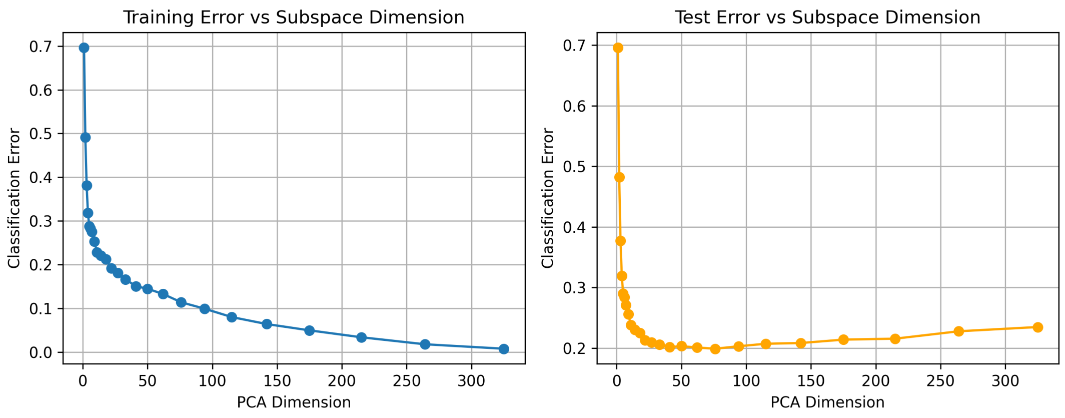

Figure 5: Classification error vs. number of PCA components for training (left) and test (right) sets.

Linear Discriminant Analysis (LDA)

In this section, we apply Linear Discriminant Analysis (LDA) to reduce the dimensionality of the Fashion-MNIST dataset and evaluate its effect on Gaussian classification performance. Unlike PCA, which is unsupervised and maximizes variance, LDA is a supervised method that aims to maximize class separability. For a dataset with C classes, LDA produces at most $C - 1$ components—in our case, 9 components for 10 classes.

Step 1: LDA Basis Vectors

We apply LDA using the `LinearDiscriminantAnalysis` implementation from `scikit-learn`. The resulting basis vectors (scalings) are reshaped into 28×28 images and visualized in Figure 6. These bases represent directions in the original feature space that best separate the classes.

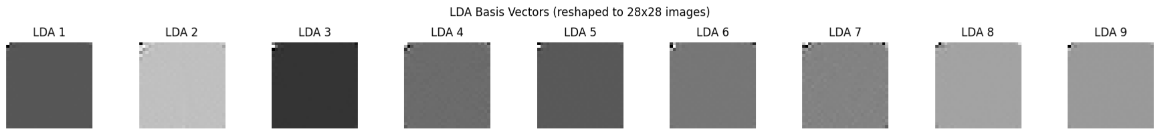

Figure 6: Top 9 LDA basis vectors reshaped to image space (28×28).

Compared to PCA bases, the LDA vectors are less visually interpretable as they are optimized for class discrimination rather than overall variance. Still, some subtle structural patterns can be observed in the corners.

Step 2: Classification Performance

We evaluated classification performance using a Gaussian Naive Bayes classifier in LDA subspaces of dimension 1 through 9. For each subspace, both training and test sets were projected using the fitted LDA transformation. The resulting classification errors are shown in Figure 7.

As expected, both training and test errors decrease as the number of LDA components increases. The improvement is especially significant from 1 to 4 dimensions, and the

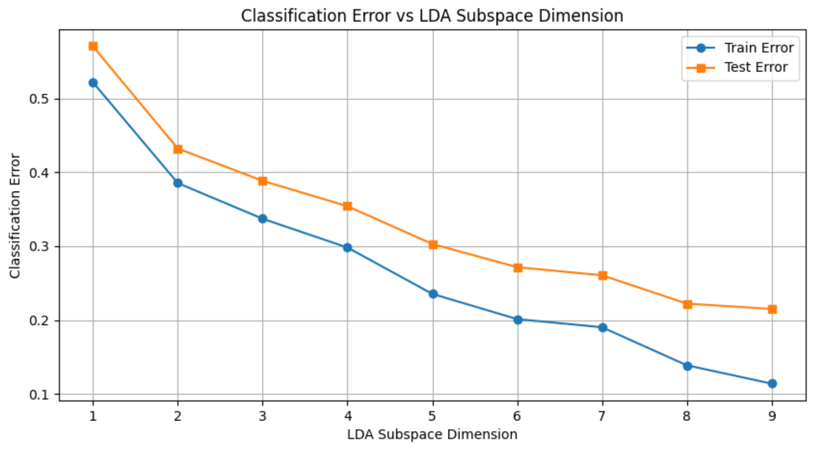

Figure 7: Classification error vs. LDA subspace dimension for training and test sets.

test error begins to plateau around 6 to 9 dimensions. This shows that most of the discriminatory information is captured within a small number of LDA components.

Discussion: LDA performs well in reducing dimensionality while preserving class separability. Although limited to 9 components, it yields competitive classification performance with fewer dimensions compared to PCA. This demonstrates the advantage of incorporating label information in dimensionality reduction.

Tools Used: Python libraries used include `NumPy`, `Matplotlib`, and `scikit-learn`. The classifier used was `GaussianNB` from `sklearn.naive_bayes`. All plots and analysis were produced from scratch, following the steps described in the assignment.

t-SNE Visualization

In this section, we apply t-distributed Stochastic Neighbor Embedding (t-SNE) to visualize the Fashion-MNIST dataset in two dimensions. Unlike PCA and LDA, t-SNE is a non-linear dimensionality reduction technique particularly suited for visualizing high-dimensional data in 2D or 3D. It is designed to preserve local structure in the data and is commonly used to reveal natural groupings or clusters.

Setup and Parameters

We used the `TSNE` implementation from `scikit-learn` to perform 2D embedding of the entire Fashion-MNIST dataset. Prior to applying t-SNE, the dataset was standardized using `StandardScaler`, which ensures that each feature has zero mean and unit variance. Although t-SNE does not require this step, it often results in better local structure preservation.

The following t-SNE parameters were used:

- **n_components = 2**: To reduce the 784-dimensional data to two dimensions for visualization.

- **perplexity = 30**: This controls the balance between local and global structure. A value of 30 is commonly used and works well for medium-sized datasets like this one.
- **init = 'pca'**: PCA-based initialization gives t-SNE a good starting point in the low-dimensional space, leading to faster convergence and more stable results compared to random initialization.
- **max_iter = 1000**: The number of gradient descent iterations. This is sufficient for convergence in most cases. We observed that increasing beyond this point led to minimal visual changes.
- **random_state = 42**: Random seed to ensure reproducibility of the results.

t-SNE does not use an explicit convergence tolerance; instead, it performs a fixed number of optimization steps (`max_iter`). Internally, it uses gradient descent to minimize the Kullback-Leibler divergence between high- and low-dimensional probability distributions, gradually improving the layout over time.

The algorithm is computationally intensive and sensitive to parameter tuning. In our case, the selected setup produced well-separated clusters and completed in a reasonable amount of time.

t-SNE Visualization

The resulting 2D embedding is visualized in Figure 8, with points colored by their class label. Each color corresponds to one of the 10 fashion categories.

Discussion

The plot reveals clear clusters for most of the classes. For example, "Sneaker", "Ankle boot", and "Sandal" appear well-separated from others, likely due to their distinct visual features. However, some overlap is noticeable between classes such as "Shirt", "T-shirt/top", and "Pullover", which tend to have similar appearances. This aligns with observed classification challenges in earlier parts of the project.

Overall, t-SNE provides an intuitive, visual confirmation of the structure in the dataset. While not suitable for classification directly, it is a powerful exploratory tool for understanding how samples from different classes are distributed in the feature space.

Tools Used: Python with `NumPy`, `Matplotlib`, and `scikit-learn`. t-SNE was implemented using `sklearn.manifold.TSNE`. Data was standardized using `StandardScaler` prior to applying t-SNE.

Tools and Libraries Used

All the the code for this project was written in Python. The following open-source libraries were used:

- **NumPy** [2] — for efficient numerical operations and array manipulation.
  https://numpy.org/

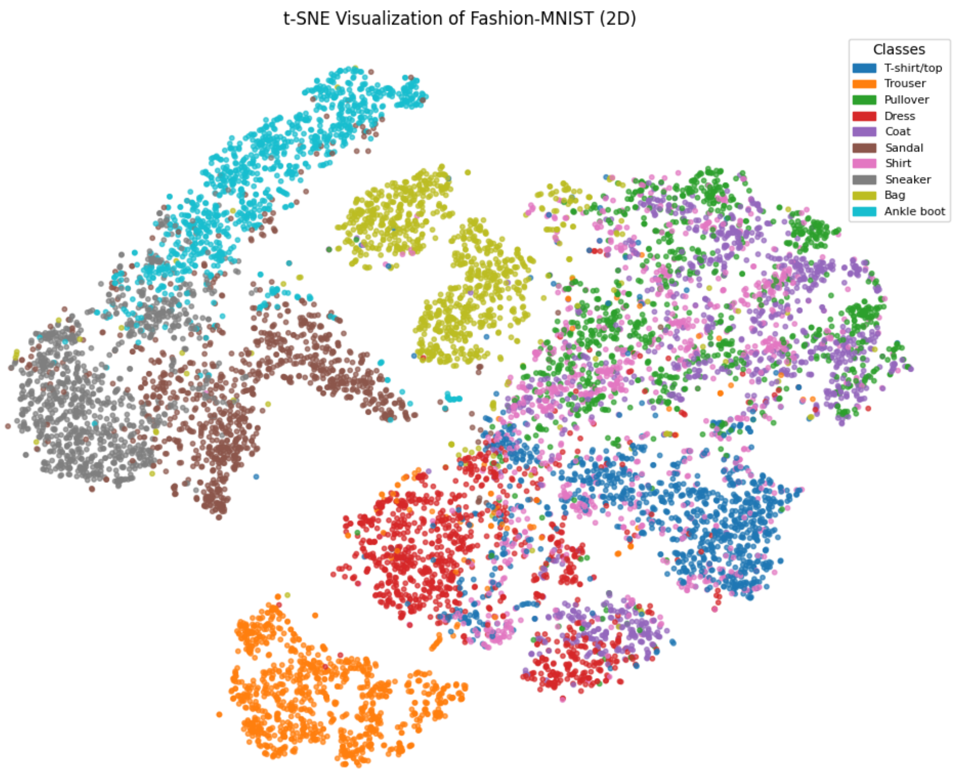

Figure 8: t-SNE 2D visualization of the Fashion-MNIST dataset. Each point represents an image, colored by class label.

- **Matplotlib** [3] — for generating visualizations and plots.
  https://matplotlib.org/
- **scikit-learn** [4] — used for PCA, LDA, t-SNE, and classification (including `LinearDiscriminantA`
  `TSNE`, `GaussianNB`, and `StandardScaler`).
  https://scikit-learn.org/
- **SciPy** [5] — specifically, `scipy.stats.multivariate_normal` was used to implement the Gaussian classifier in PCA space.
  https://scipy.org/

References

[1] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[2] C. R. Harris *et al.*, "Array programming with numpy," 2020.

[3] J. D. Hunter, "Matplotlib: A 2d graphics environment," 2007.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, *et al.*, "Scikit-learn: Machine learning in python," 2011.

[5] P. Virtanen *et al.*, "Scipy 1.0: Fundamental algorithms for scientific computing in python," 2020.