SPRING 2023-2024

GE 461: INTRODUCTION TO DATA SCIENCE

Date: 06.04.2024

DIMENSIONALITY REDUCTION AND VISUALIZATION

SECOND PROJECT REPORT

YİĞİT ALİ KARADOĞAN

21902218

DEPARTMENT OF INDUSTRIAL ENGINEERING

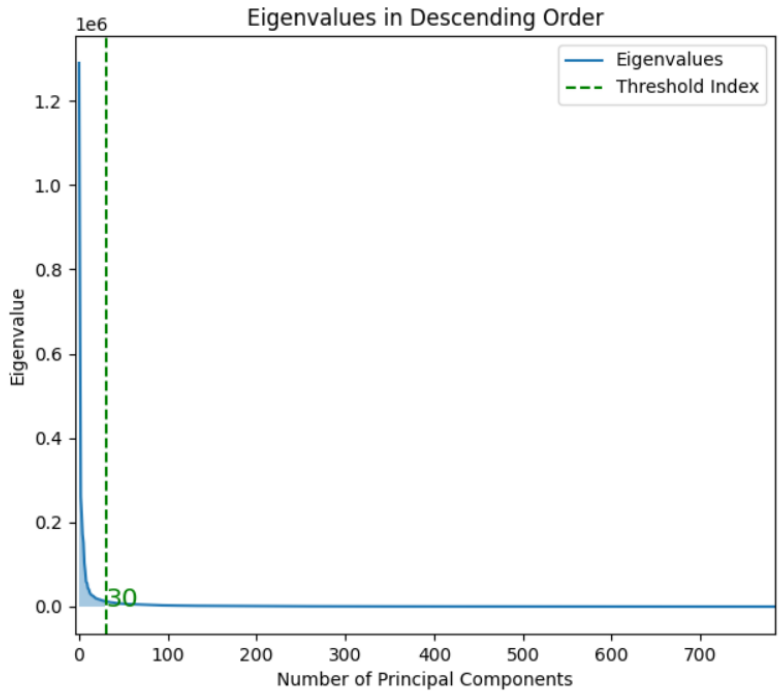
BILKENT UNIVERSITY

06800 ANKARA

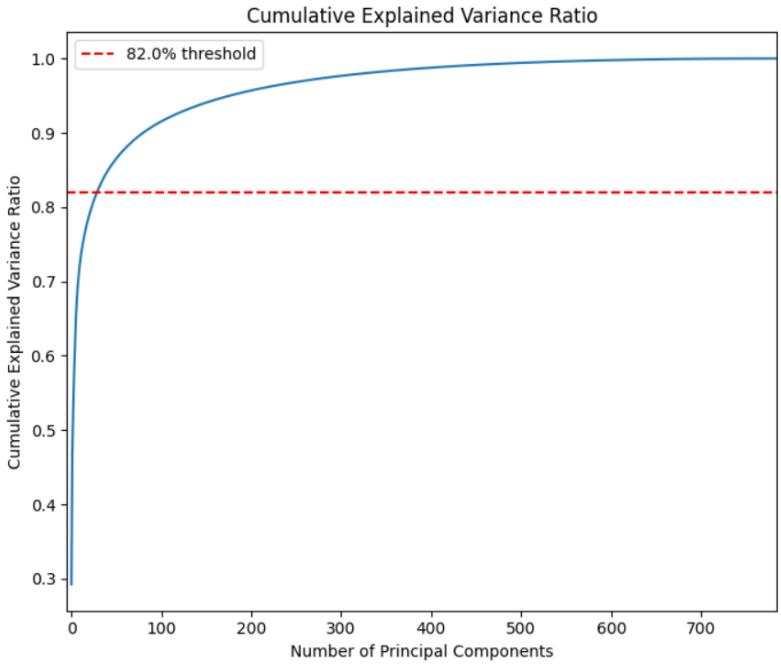
Note: I have included the libraries I used and their implementation details at the conclusion of the report so as not to impede the details of the report.

1. QUESTION 1)

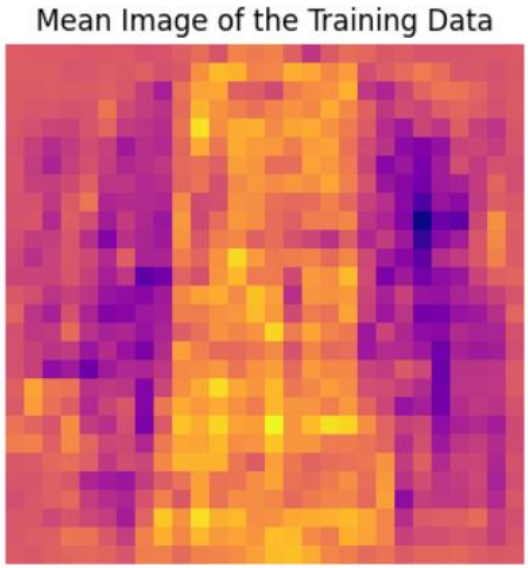
I answered first question by according to the instructions given. To put it plainly, I fitted my centered train data to the PCA using a PCA library. Following that, I employed an implementation of the Quadratic Gaussian Classifier known as "Quadratic Discriminant Analysis." The implementation estimates the MLE according to the formula given in the Project PDF and fits a Gaussian for each of the classes.



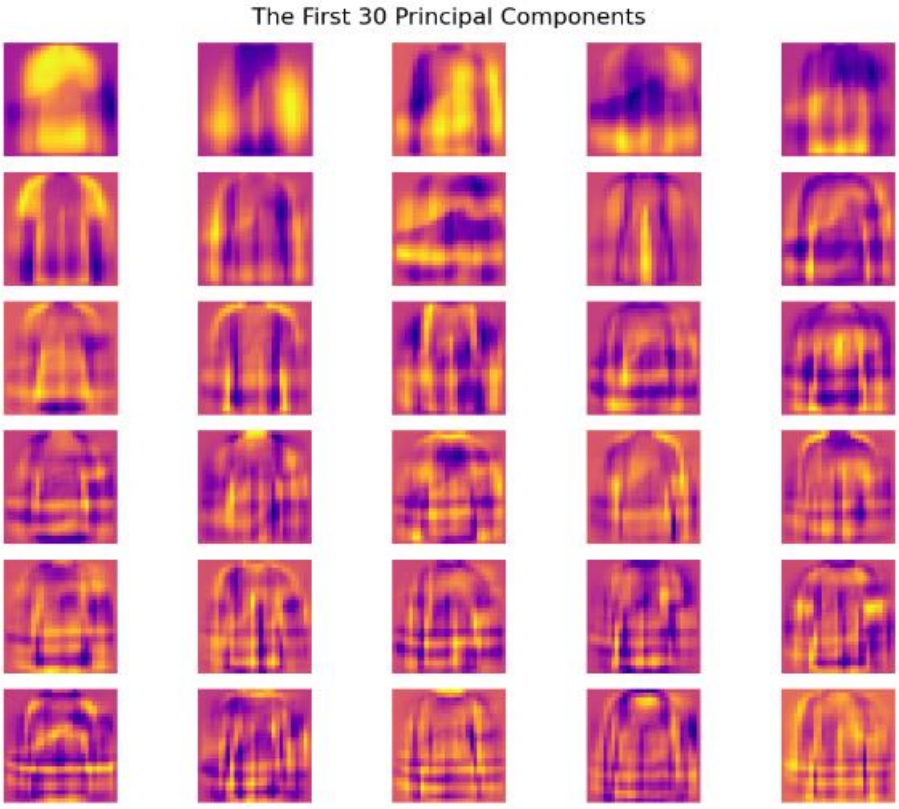
When we plot our eigenvectors, or principal components, based on their eigenvalues, I was able to produce the above figure. With 784 features, I can acquire a maximum of 784 principal components, all of which are presented alongside their corresponding eigenvalues. The amount of variance that our primary components will explain will be reflected in the eigenvalues that are obtained. The graph shows that the eigenvalues of the first couple of principal components have significantly greater values than the other principal components. The decline is not linear, and it happens much more quickly up until the thirtieth principal component. The eigenvalues are gradually approaching 0 after this. For that question, I would thus assume that the first 30 eigenvectors would be a reasonable option based only on the plot. Furthermore, to visualize the amount of variation explained by the first 30 eigenvectors, we can plot a cumulative PVE graph.



It can be observed that the cumulative variances described by 30 eigenvectors account for about 82% of the variation. Because a variance of 82% can be interpreted as a value that will be able to retain the majority of the information gathered from our data and because it uses about 4% of all eigenvectors, which may be considered a sufficient reduction of dimensions for our analysis.

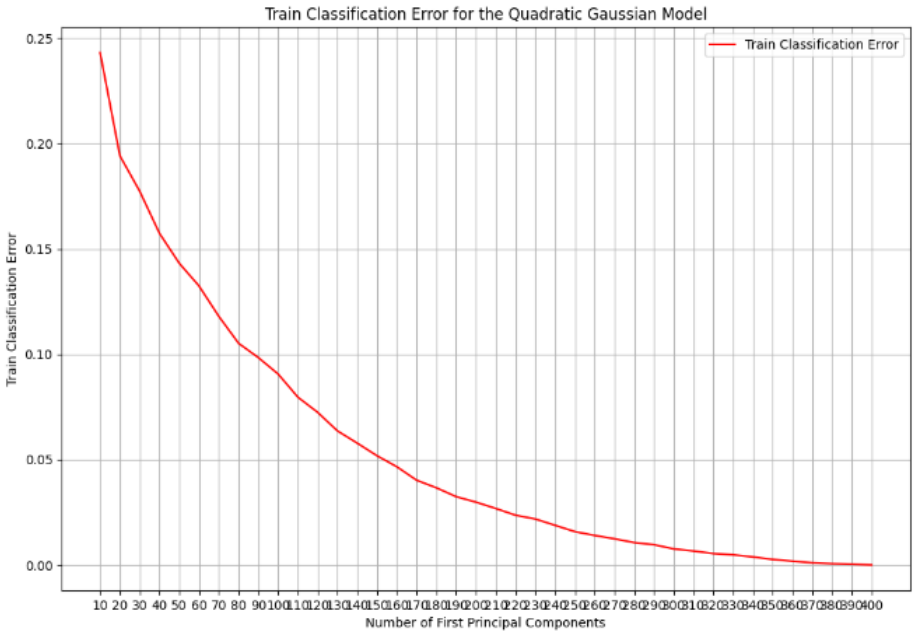


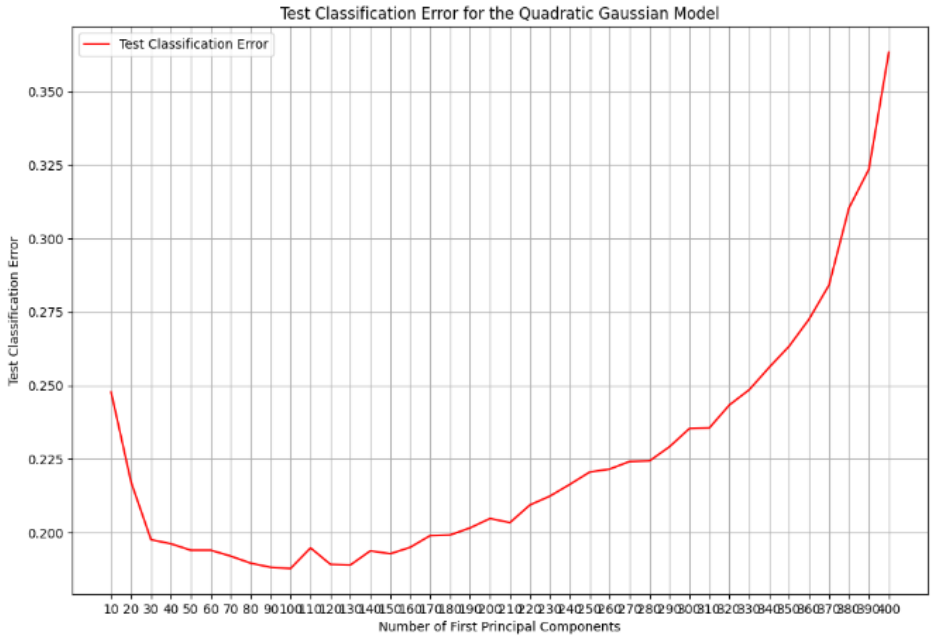
I colored the data using a "plasma" to make it visually appealing. Prior to examining the mean figure, it makes sense to me that, as the mean will represent the assortment of clothing, I should anticipate seeing a fuzzy image that somewhat like the clothing. If we inspect it closely, the white coloring of dresses, coats, shirts can be seen at the centre. Also, we can see the sleeves too. We also see the sneakers toe and the back of the sneakers at the middle left part, and middle right part, respectively.



According to what we know about principal components, the first few principal components hold onto the most variance. This indicates that it is anticipated that the initial components will depict the clothing's more generic structures. We would anticipate that when we decrease to lower-variance valued principal components, they would record more specific information from the pictures rather than generic structures. The eigenvalues of the eigenvectors in the image decrease from top to bottom and from left to right. As we look at the first PC image, we see a generic sneaker and white shirt structure. At the second PC image, trousers's structure is also captured. As we move forward, first set of images resemble each other and to the bottom of the chart, the images become less

and less interpretable, and it is likely that they are capturing small characteristics seen in the acquired samples, including curves and slightly different shapes. Lastly, we also see that most of the images have short-sleeved top which captures most of the variance. It makes sense since coats, t-shirts, pullovers, dresses, shirts contain it as a basis.

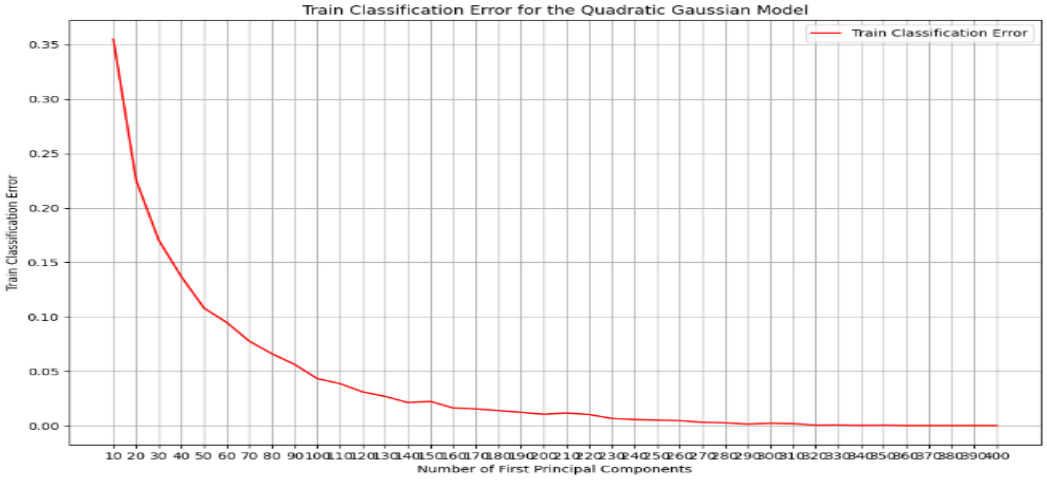


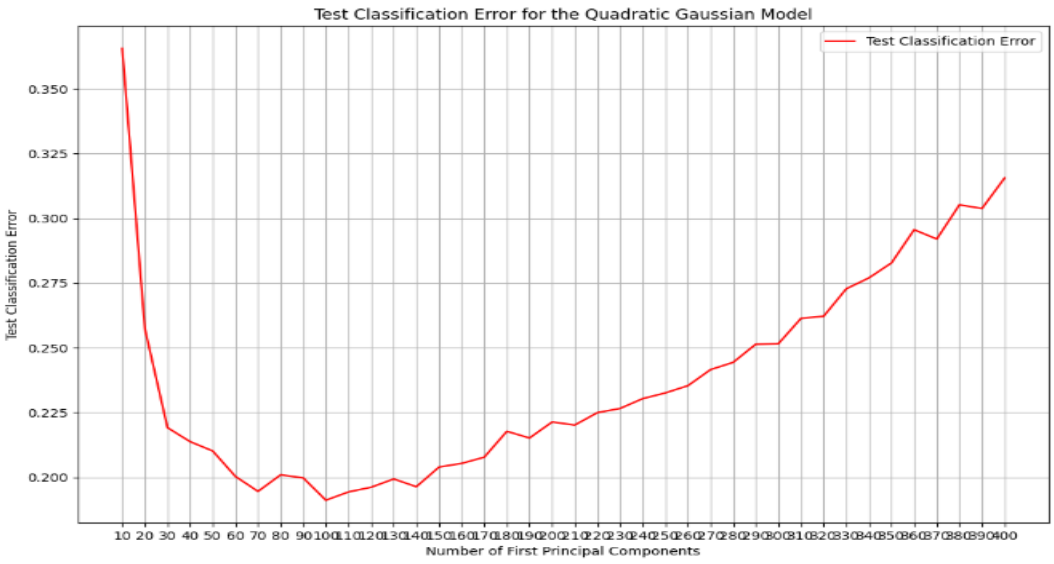


I obtained the numbers above, which display the error percentage, by using eigenvectors in the range of 10 to 400, in increments of 10. I used the half of the data for training, and the other half for testing. The graph for the training dataset shows that as the number of primary components increases, the classification error continuously decreases. This can be explained by the fact that we trained the QGM using the train dataset, which we also used to get the principal components. The train dataset will be more accurately represented by the principal components as we increase their number of dimensions. As the number of dimensions increases, the gaussian model is able to predict it with about 0% error because, up to 400 principal components, it will retain almost all of the information from the train data. In essence, the model predicts the data that it has previously learned.

We can see that this isn't the case for the test dataset. The test dataset differs from the train dataset in that it is not visible to our model; as a result, its error percentages will differ. The test dataset's graph indicates that, with a dimension of 100, the lowest error percentage is roughly 18%. Furthermore, the error percentages do not continuously drop like the train dataset graph does. This indicates that, rather than attempting to capture the underlying generalizable patterns, we eventually start overfitting the data and end up fitting the noise of the train dataset into our model. As a result, after a given dimension, the error percentages rise.

2. QUESTION 2)



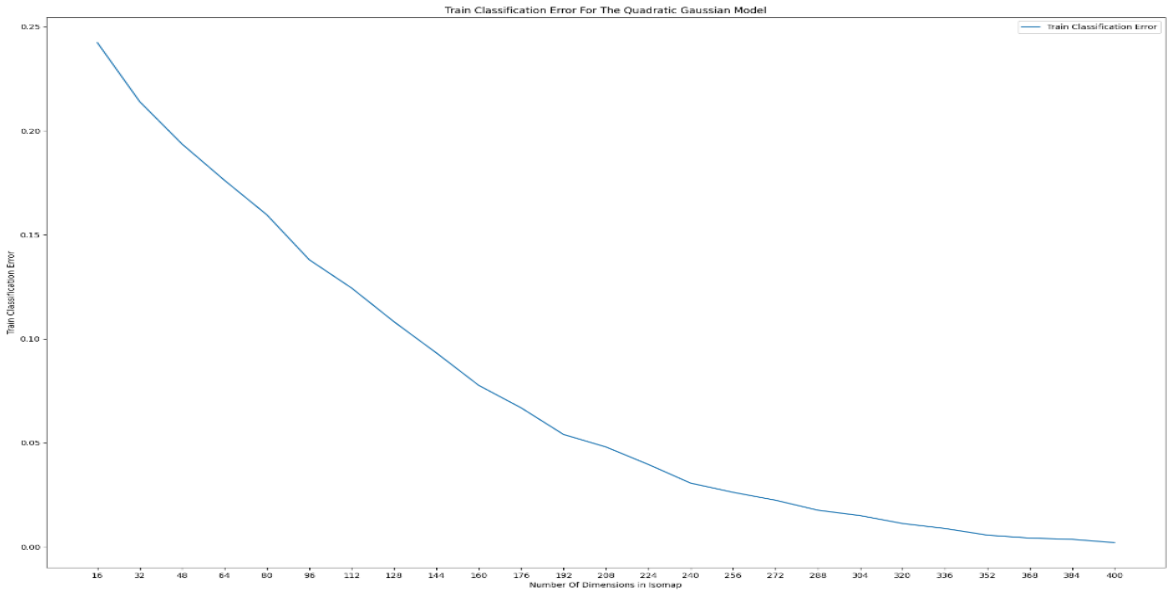


In this question, instead of the bases by using principal components, I generated random matrices and projected the data onto a lower-dimensional subspace by using them. For the training dataset, since random matrices cannot capture the variation as good as first couple of principal components, classification error starts at a higher rate than the one constructed with PCA bases. Similarly to the PCA, this also converges to zero classification error as random matrices increases.

For the test dataset, for the same reason as training dataset, since random matrices cannot capture the variation as good as first couple of principal components, classification error starts at a higher rate than the one constructed with PCA bases. Again, after some point, classification error increases since we start to overfitting data.

3. QUESTION 3)

Isomap's implementation from sklearn.manifold has been utilized for this question. I have only made 16-step increment changes to the n_component parameter between 16 and 400. The number of dimensions is indicated by the parameter. I've used n_neighbours parameter as default value 5 since bigger values make my computer nearly blow up and computationally expensive.

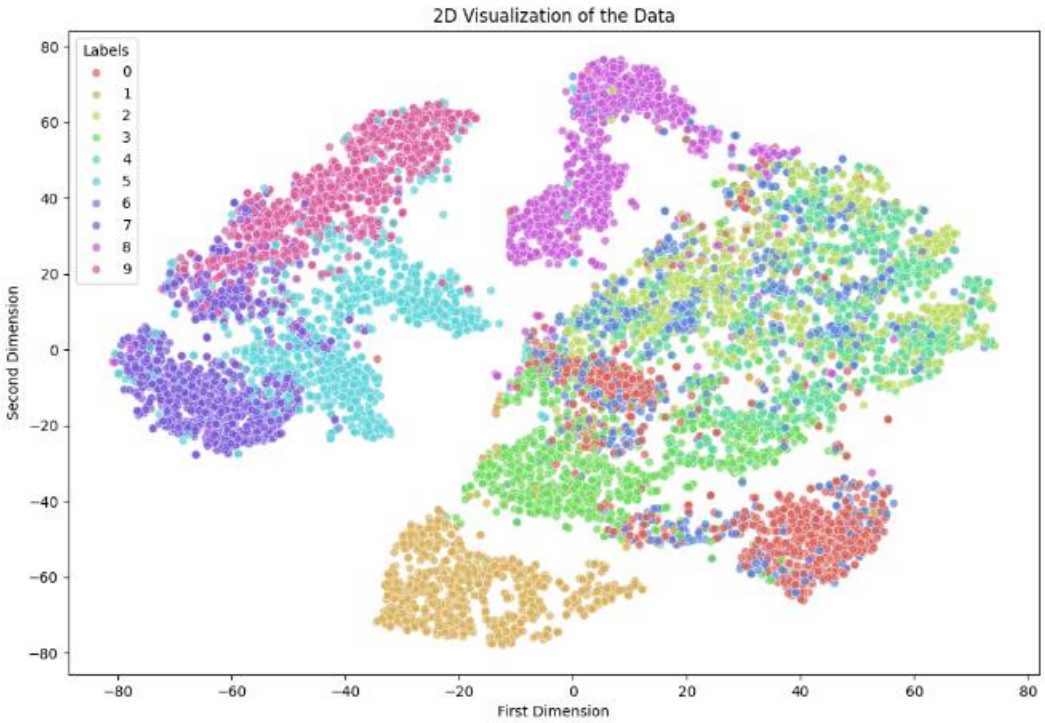


The trajectory have seen using PCA and by the Isomap are similar. For the training, we can observe that our error percentage continuously declines and approaches zero as the number of dimensions increases. Likewise, the test prediction follows the same pattern. When we reach an overfitting point, the error percentage starts to rise once again after first decreasing to a certain amount. At 190 dimensions, the error percentage is around 22.5%, which is the lowest of all the dimensions.

When PCA and Isomap are compared, it is evident that both have the capacity to represent the data in comparable dimensions with a low classification error. However, Isomap required to have more components than PCA to have minimum classification

error. The fundamental distinctions between PCA and Isomap could be the reason behind this. A linear technique for reducing dimensionality is PCA. On the other hand, Isomap is a non-linear technique for reducing dimensionality. As a result, the pattern that is captured by these two approaches will be different. The fact that most clothing shapes have comparable structural elements. PCA may also be successful in capturing these comparatively straightforward and linear dataset structures, and a model with fewer components (dimensions) than Isomap does.

4. QUESTION 4)



I used the fit_transform method on the TSNE object with the centered entire dataset, taking advantage of the t-SNE implementations in the "sklearn.manifold" package. n_components is selected as 2. I decided to use early_exaggeration's and learning_rate's default value. n_iter's default value of 1000 is used as well. Perplexity's default value of 30 is used. I did not increase these because my computer is not that young to handle hard computational tasks.

The high-dimensional relationships between the clothes in a two-dimensional space are represented in the dataset's above t-SNE figure. As a nonlinear dimensionality reduction

method, t-SNE frequently reveals complex correlations between clothing shapes that may go unnoticed by methods like PCA or Isomap.

For example, classes 0, 3, 4, and 6 are visually similar, we can see from the plot that they are clustered within each other. Also, classes 5, 7, and 9 have similar structure since they are basically shoes and it can be seen from the plot as well (to the upper left of the plot). Since class 8 (bag) is not similar to shoes or shirts, it's separate pattern can also be seen.

The plot helps in our understanding of class variability as well. For instance, the presence or absence of sleeves can alter the visualization of a class 0 item (t-shirts/tops). The t-SNE plot indicates that there is variability within class 0 since it shows two distinct clusters, each of which represents whether or not it has sleeves.

All things considered, the t-SNE plot can help us recognize clusters of related clothes and provide an intuitive knowledge of the links between various clothes. It's crucial to remember that the t-SNE plot is only a tool for visualizing the data; in order to statistically examine the data, we must employ other methods.

Tools Used In The Implementations

Python has been the programming language used in this project, along with a number of Python libraries and modules, to train a quadratic Gaussian classifier using the projected dataset and analyze the dataset using PCA, Isomap, and t-SNE [1]. Initially, I loaded the dataset by using the 'os' module to access and modify the paths. Additionally, the dataset was loaded and the "numpy" library was used to manage and wrangle the data in order to optimize linear algebra computations and prepare it for modeling, training, and reporting [2]. I used "matplotlib" to visualize data by displaying graphs and plots that represented the outcomes of PCA, Isomap, and t-SNE [1]. In addition to "matplotlib," I also benefited from the "seaborn" [3] and "pandas" [4] libraries to show the t-SNE results with more visually appealing visualizations. I used the "sklearn" library, notably the "sklearn.decomposition" and "sklearn.manifold" modules for dimensionality reduction and to obtain low-dimensional embeddings of the data by PCA, Isomap, and t-SNE implementations, in order to carry out the actual modeling and evaluation of my machine learning models [5]. To keep track of the model trainings' progress, I loaded the "tqdm" module [6].

References

[1] *Sklearn.manifold.TSNE*. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

[2] *Numpy*. NumPy. (n.d.). https://numpy.org/

[3] *Statistical Data Visualization#Seaborn*. seaborn. (n.d.). https://seaborn.pydata.org/

[4] *Pandas*. pandas. (n.d.). https://pandas.pydata.org/

[5] *SKLEARN.DECOMPOSITION.PCA*. scikit. (n.d.-a). https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[6] Costa-Luis, C. da. (n.d.). *TQDM#*. tqdm documentation. https://tqdm.github.io/