Bilkent University - GE 461 Spring 2022 - Project 2

Full Name: Zübeyir Bodur **ID**: 21702382

1. Software

In this assignment, following tools and methods were used:

- NumPy [1]
- Matplotlib [2]
- SciPy [3]
- Scikit-Learn, which includes the implementations of:
  - PCA, LDA, t-SNE [4]
- Sammon's Mapping [5, 6]

2. Preparing Data

The MNIST data set was split into two as follows, for 1st and 2nd question:

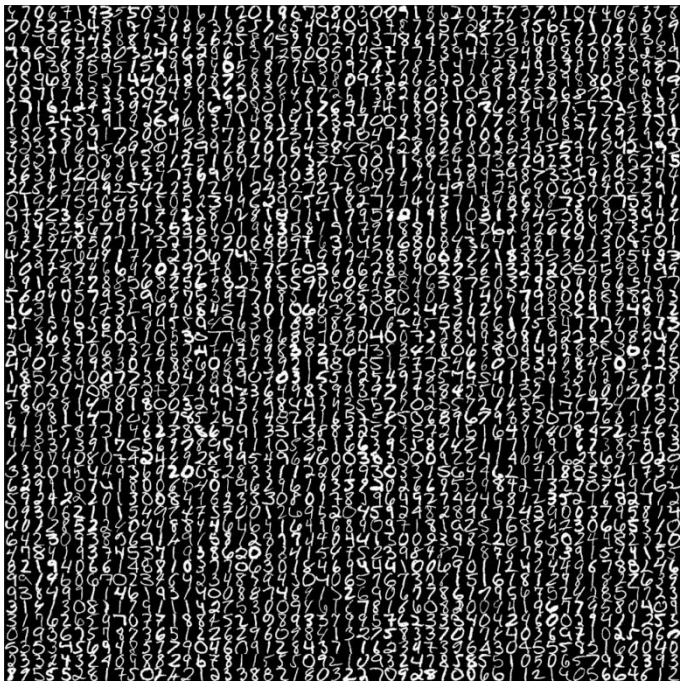

Figure 1: Train data for some run.

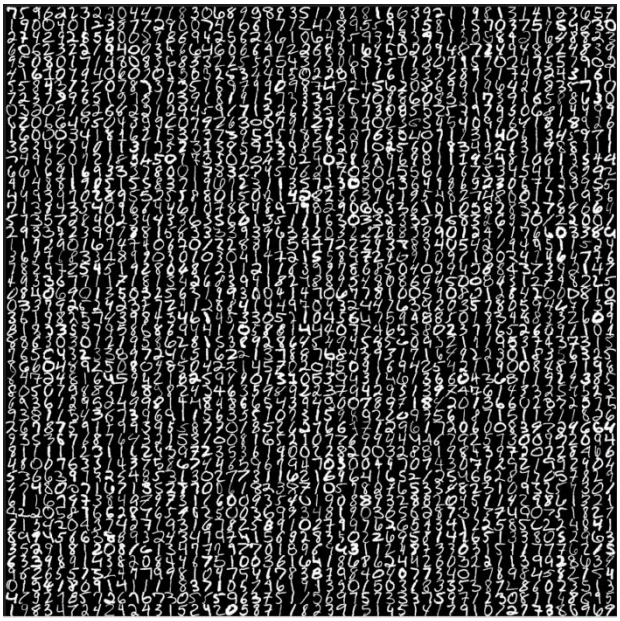

Figure 2: Test data for the same run in Figure 1

3. Question 1 - Principal Component Analysis (PCA)

a) Eigenvalues of the Covariance Matrix

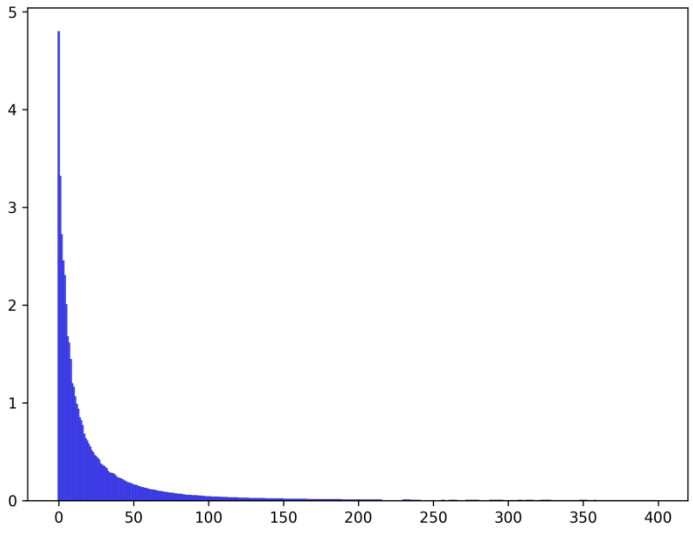

Figure 3: Bar plot for the eigenvalues of the covariance matrix of the training set. Y-axis shows the value, x-axis shows the index i of the eigenvalue λ_i.

From this plot, I can observe that most of the "important" information about the training data is stored within apx. **100** eigenvectors. However, we know that **100 is still a large number for dimensions**, and we will make a trade-off between explained variance and the dimensionality, and will choose such a number so that explained variance is still large, but dimensionality is much smaller.

Hence, by only looking at this plot, I would choose **at most 50 PCA components**.

b) Sample Mean vs Eigenvectors (Bases)

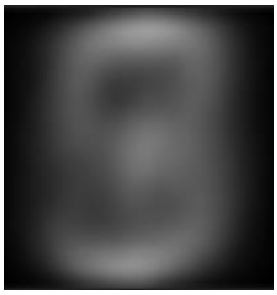

Figure 4: Average of the training set for some training set.

Figure 4 shows a sample mean of the training set. We can observe that the shape contains **all the strokes from all digits**, especially from 0's, 3's and 8's.

In Figure 5, I have displayed the chosen eigenvectors as images **using the training set**. The eigenvectors on the top left have the largest value, and the eigenvectors on the bottom right have the smallest eigenvalues. The eigenvectors of course, **do not strictly represent all digits at once**, which is expected. However, first 4 eigenvectors look like digits (0, 1, 9, 8 and 3), and this is also expected, that **first eigenvectors will be very similar to some of the digits**.

In the satellite image example in the lecture slides, **the important information** provided by eigenvectors also **decreased** as their **eigenvalues decrease**.

It can be observed that, **as the eigenvalues decrease, t**he eigenvectors represent **less and less information and** turn into a **randomly generated grayscale image,** as expected, in Figure 5 as well. It is also expected, that **eigenvectors corresponding the largest eigenvalues best represent the digits**.

While projecting the data, I will use the first 5, 10, 15, …, 200, 205 eigenvectors as bases. Figure 5 shows the top 205 eigenvectors as images.

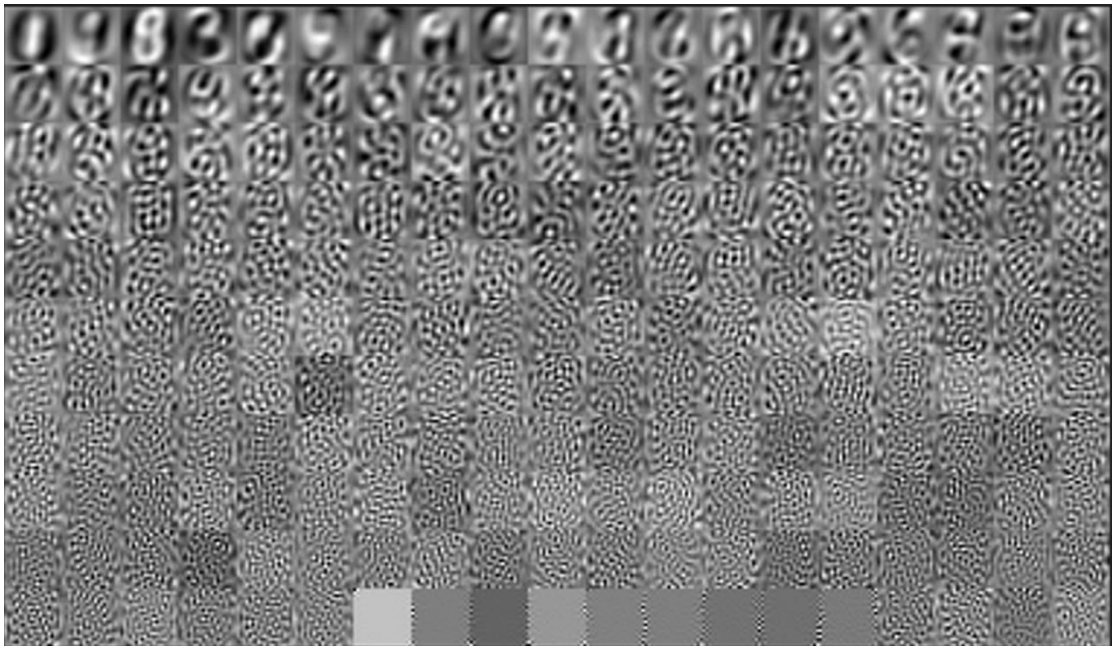

Figure 5: Top 205 eigenvectors obtained from some run of the principal component analysis using the training set.

c) Training Procedure

However, since we have the label information for free, we will also use the label information to train the data, even though PCA is an unsupervised learning method. The implementation will be continued with respect to the methods that were discussed in the paper "Recognizing MNIST Handwritten Data Set Using PCA and LDA" by Sheikh et. al. [7], and "Understanding Gaussian Classifier" by R. Buoy [8].

Even though Buoy uses class posterior probabilities to make predictions [8], for this project, I did not include this while making predictions, because we know that distribution of digits do not tell anything about what the next digit will be, as the digits in MNIST data set are independent from each other.

d) Projections

For the sake of measuring the effect of dimensionality, I projected the data into sub spaces using the first $5i + 5$ PCA components where $i = \overline{0, 40}$.

e) Results

The plot below shows that **curse of dimensionality** just like we discussed in the class.

For the training data, we can **easily say that we are over-fitting our classifier**. Errors as close to 0 % on the training data while dimensionality increases tells us that small differences, such as on the test set, may be considered as a totally different class, and a wrong decision may be made.

For the both data, as we include **too scarce components**, we will **be omitting important bases that needs to be included in our subspace**, as the first 20 components help explain approximately 60 % of the variance in our data.

However, the performance of the classifier i**s not proportionally related with the explained ratio, which can be observed from the test results**. **At some point**, **adding more dimensions (features) increase the classification error**.

i. Classification Error on the Training Data

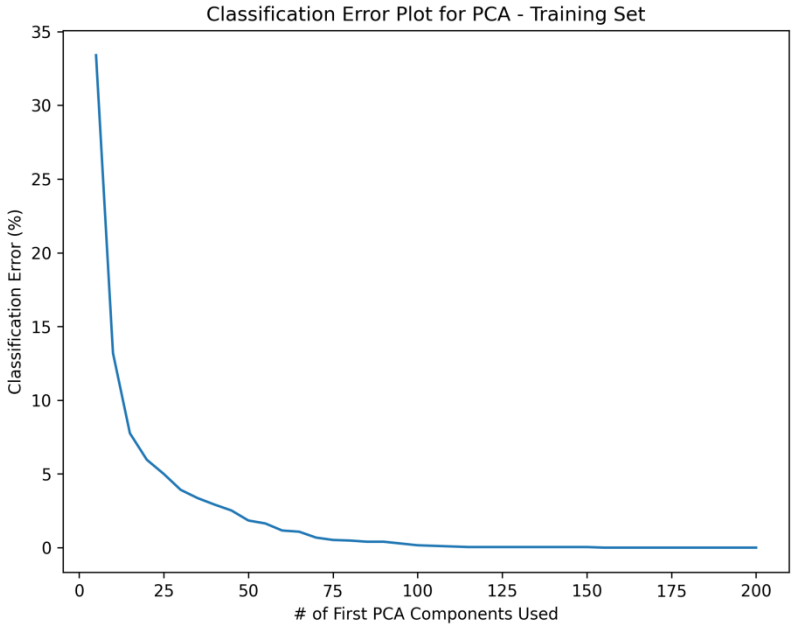

Figure 6: Plot for the classification error vs number of first PCA components included - for the training data.

ii. Classification Error on the Test Data

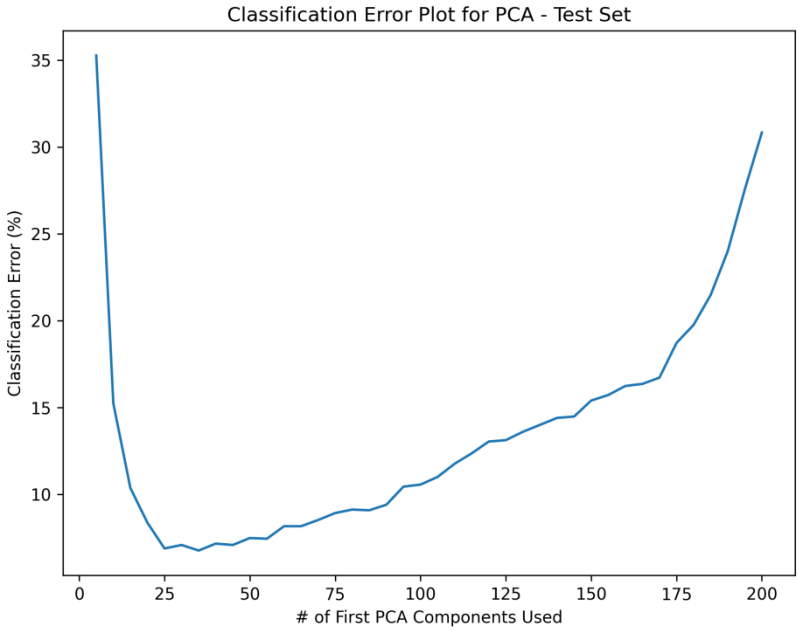

Figure 7: Plot for the classification error vs number of first PCA components included - for the test data.

4. Question 2 - Linear Discriminant Analysis (LDA)

a) Bases (Eigenvectors)

The image below shows the bases eigenvectors obtained by Fisher's LDA.

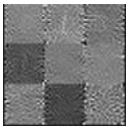

Figure 8: 9 bases, that are 400 dimensional, obtained by LDA.

Unlike PCA, Scikit Learn's (Same as Fisher's) LDA selects the top (n_classes - 1) eigenvectors of the following matrix:

$$S_W^{-1} S_B$$

Where S_W stands for within class covariance matrix, and S_B stands for between class covariance matrix, so that J(W), the Fisher Linear discriminant is minimized.

In PCA, when we show the eigenvectors of the S_B, they are meaningful images as they are **eigenvectors of the covariance matrix of the raw data**, which are **meant to represent the data**.

When we try to do the same thing on this $S_W^{-1}S_B$ matrix, it will not give meaningful representations, as the eigenvectors now are **meant to separate the data, not to represent**. Therefore, the eigenvectors displayed as images **do not represent any valuable information, even though they all contain valuable information**.

b) Projections

For the sake of measuring the effect of dimensionality, I projected the data into sub spaces using the first i LDA components where $i = \overline{1,9}$.

c) Training Procedure

The same classifier was used as in PCA. Even though SciKit learn has its own predict function, in this project, we used LDA to reduce the dimensionality of our data. We can not use SciKit-learn's predict function as it assumes class variances are shared, which we want to avoid while making classifications.

Using the new sub spaces, we will still assume that each class has different variance. With this approach, we will not only have less and less dimensions, but also have clusters that are nicely separated by LDA. Therefore, using the same classifier as in Question 1 should still give good results.

d) Results

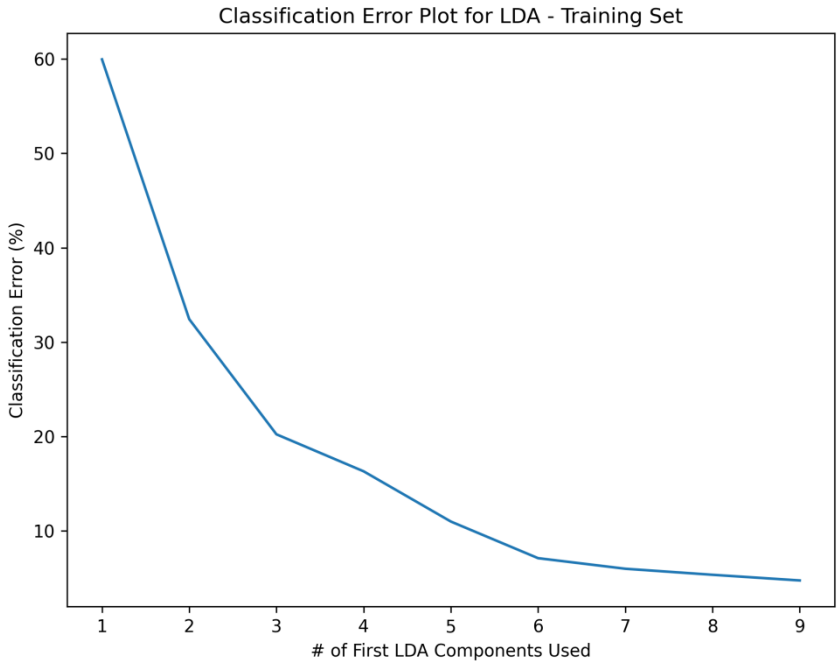

Figure 9: Plot for the classification error vs number of first LDA components included - for the training set.

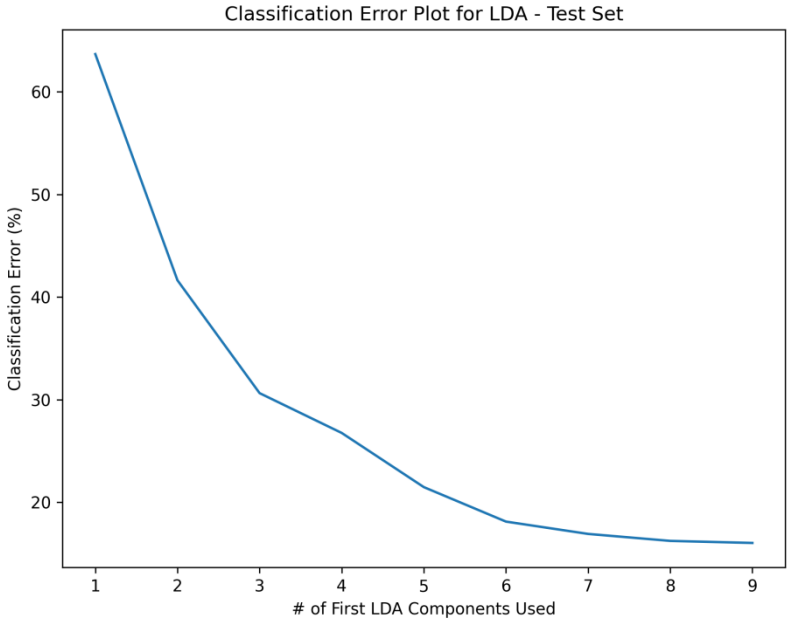

Figure 10: Plot for the classification error vs number of first LDA components included - for the test set.

5. Discussion - Q1 & Q2

a) The Effect of Dimensionality

As explained in 3. e), the dimensionality of the features in a data set have very large effect in the training procedure.

We should not remove too many features, otherwise we will loose valuable information, as we can see from 3. e) and 4. d).

However, **we should not include too many features**, otherwise we will face the curse of dimensionality, and our model will over fit with the training data. For LDA, we did not have too many dimensions, however, therefore this conclusion can only be derived from 3. e).

b) PCA vs LDA

In this data set, **PCA performed better than LDA**.

If we select the first 25 PCA components, and test our model, we will get apx. **7 %** classification error.

On the other hand, if we include all of the LDA components instead, we will have a classification error of apx. **16.2 %**.

We can observe that **PCA is better alternative than LDA** while it **requires some parameter tuning**. LDA performs good if we just include all of the possible dimensions provided by LDA, therefore it does not require any parameter tuning, while under performing the best tuning of PCA.

However, we still do not know what would be the results if we had large number of classes, say 100, for LDA.

A good way to improve the performance of a discriminant analysis could be using Quadratic Discriminant Analysis instead.

6. Question 3 - 2D Visualization

a) Sammon's Mapping

i. Parameter Tuning

1. Original Paper Parameters - Not Used

Algorithm in the original paper requires this parameters to be tuned:

`alpha`: Also known as MF (Magic Factor) in Sammon's original paper [5], this will determine how important the rations of partial derivatives will be in each iteration.

`maxiter`: Maximum number of epochs to learn the optimal 2D mapping.

`epsilon`: The ratio where the algorithm will consider that mapping converged and will stop.

A resource that implements the original algorithm can be found in R. Karlsson's GitHub repository [9]. However, the original algorithm runs too slow. Even the 149x4 Iris dataset takes 1 minute to map. Therefore, more efficient algorithms that look for more efficient ways to compute E metric in the paper were sought [6].

2. External Resource that Implement with Different Heuristics

The following parameters were **used to generate Figure 11:**

`maxhalves = 100`: Unlike the original J. Sammon's implementation, this implementation uses a step halving procedure to make progress, instead of using a Magic Factor called `alpha`. This is the maximum number of halving steps to find a better E value, and the default value seems to find a better E value at each iteration without bombing out. In other words, this parameter should just be large enough for finding a better mapping, increasing linearly will not improve the mapping.

`maxiter = 10 000`: Increasing this parameter was helpful reaching the tolerance in objective function - `tolfun` - threshold. Specifically, at epoch 2794, `tolfun` was exceeded, and mapping was terminated. Likewise, increasing this parameter on its own will not improve the mapping.

`tolfun = 1e-11`: The more this parameter is close to zero, the better the mapping will be. However, for even smaller values of `tolfun`, the training will take hours. However, for very large values of `tolfun`, Sammon's mapping will not converge, it will give a mapping where all clusters are on top of each other.

`init = "pca"`: The initial settings mapping will start. For this result, I have chosen PCA, so that the initial mapping will be the first 2 PCA components.

ii. Results

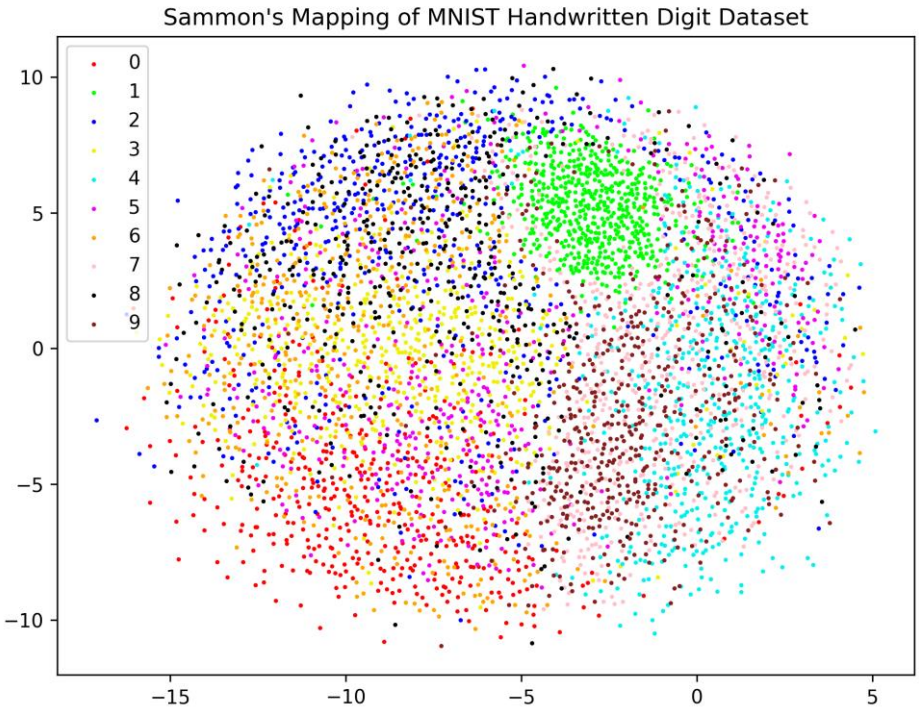

Figure 11: 2D visualization of MNIST data set using Sammon's mapping, with parameters in .

iii. Comments

- 1's (green) are quite separable from other classes.
- 4's can be found in southwest cluster (cyan).
- 5's (magenta) form a big curve, starting from northeast, ending in southwest.
- 2's are mostly in northwest.
- …
- In general, except 5, same classes form clusters that are different than the whole data set. However,
  - **Most clusters intersect with other clusters** in large areas. This would mean that we would not be able to run k-Means to partition this mapping and

predict a point's label if we were to remove the label information. Indeed, even with tuned parameters, Sammon's mapping is not useful for unsupervised learning algorithms such as k-Means.

- **Between cluster** distance is **too small**, or even zero, and **within cluster** distance is **too large**.

- The parameters can still be tuned to improve the clustering, however, it is not a choice, in my opinion, to use Sammon's mapping for a data set with high dimensions and entries. It takes more than hours, to get good results with this method.

b) t-SNE

i. Parameter Tuning

Scikit Learn's TSNE implementation has a lot of paramters, the ones that were tuned, changed from their default values are the following:

`perplexity = 100.0`: It was observed that perplexity parameter is inversely proportional to the within-class distances. Therefore, this parameter was raised as much as possible. It was observed that, with `early_exaggeration = 12.0`, it is safe to set perplexity more than 50.0, highest recommended value by SciKit learn.

`n_iter = 10 000`: The more iterations the better, therefore number of iterations were increased from 1 000 to 10 000.

`learning_rate = 1.0`: Learning rate should be large enough so that the fitting procedure halts, however, should be also small enough that error is minimal. Even though the range is in [10.0, 1000.0], I decided to decrease it as much as possible to get even better results, since I had the computing power to do so.

`metric = 'cosine'`: I have observed that, for MNIST dataset, `cosine` distance metric works best, instead of `euclidean` and `manhattan` distances. Unfortunately, Mahalonobis distance was not

implemented, but if it was available, using that metric would be also fine.

`init = 'pca'`: It is more globally stable than random initialization, so PCA initial embedding will be used.

ii. Results

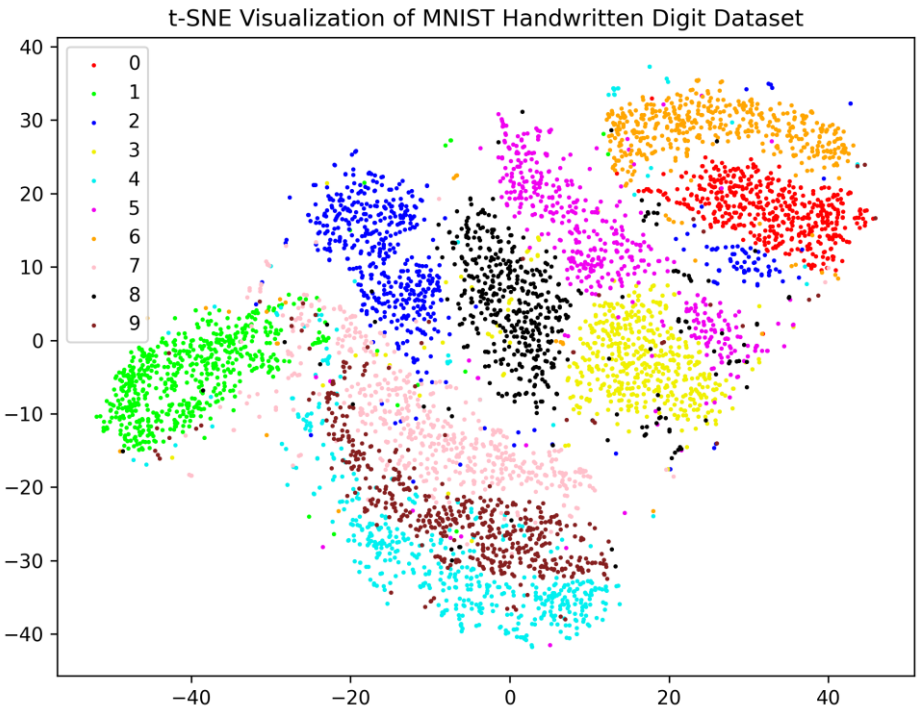

Figure 12: 2D visualization of MNIST data set using t-SNE, with the parameters in 6. b) i).

iii. Comments

Unlike Sammon's mapping, **t-SNE gave stunning results** without needing exhaustive training procedure - it took only 10 to 15 minutes to compute the mapping despite my huge parameters. The main reason could be that **t-SNE is supervised**, as it used the label information.

We can easily identify clusters of our data set using this plot. However, we can still examine the following:

- All classes have a ***master*** cluster.
- **Master** clusters are not necessarily distant from each other.
- Some of the 2's (blue) and 5's (magenta) have **minor** cluster(s)
- Small amount of **4's** (cyan) were **mixed** with **9's** (brown)
- According to L. J. P. van der Mateen, moving along an axis in a cluster may tell us about the changes of patterns [4], however, it is not possible to examine this by just looking at the scatter plot.
- The algorithm suits well for unsupervised learning algorithms such as k-Means, k-NN, and we can also tell this just by comparing Figure 11 and 12.
    - Indeed, L. J. P. van der Mateen's parameters, which he used on MNIST on his original paper, helps form almost perfect clusters [4].

7. References

[1] "NumPy," *NumPy*, 2006. Available: https://numpy.org

[2] "Matplotlip" *Matplotlib*, 2003. Available: https://matplotlib.org

[3] "SciPy", *SciPy*, 2001. Available: https://scipy.org

[4] L. J. P. van der Maaten and G. E. Hinton, "Visualizing High-Dimensional Data Using t-SNE," Journal of Machine Learning Research, vol. 9, pp:2579-2605, November 2008.

[5] J. W. Sammon, "A Nonlinear Mapping for Data Structure Analysis," IEEE Transactions on Computers, vol. C-18, no. 5, pp:401-409, May 1969.

[6] T. Pollard, "Sammon mapping in Python," *GitHub*, Apr. 18, 2014. Available: https://github.com/tompollard/sammon. [Accessed: Apr. 16, 2022].

[7] R. Sheikh, M. Patel and Dr. A. Sinhal, "Recognizing MNIST Handwritten Data Set Using PCA and LDA," 2020, DOI: 10.1007/978-981-15-1059-5_20. Available: https://www.researchgate.net/publication/339559900_Recognizing_MNIST_Handwritten_Data_Set_Using_PCA_and_LDA. [Accessed: Apr. 14, 2022].

[8] R. Buoy, “Understanding Gaussian Classifier,” Jun. 12, 2019. Available: https://medium.com/swlh/understanding-gaussian-classifier-6c9f3452358f . [Accessed: Apr. 16, 2022].

[9] R. Karlsson, “Sammon’s Mapping,” GitHub, 2021. Available: https://github.com/RobinKarlsson/Sammon-Mapping. [Accessed: Apr. 17, 2022].