**Lab I – Dimensionality Reduction**

**Machine Learning II**

**Workshop I**The following are some of the results of the dimensionality reduction lab

1. **Simulate any random rectangular matrix A**

Matrix: [[4 3 4]

[5 2 3]

[7 3 0]]

* **What is the rank and trace of A?**

The rank is: 3

The trace is: 6

* **What is the determinant of A?**

The determinant is: 31

* **Can you invert A? How?**

Yes, if determinant different from 0. With the funtion np.linalg.inv from numpy

* **How are eigenvalues and eigenvectors of A’A and AA’ related?**

Eigenvalue AA´: [-3.5678767 0.99853387 -0.3983991 ]

Eigenvalue A’A: [-3.5678767 0.99853387 -0.3983991 ]

have the same eigenvalues

A'Av = λv

AA'Av = Aλv

* **What interesting differences can you notice between A’A and AA’?**

Size: The matrix A'A has the same number of columns as rows as matrix A has, while AA' has the same number of rows as columns as A has. The same with the Rank.

Type: The matrix A'A is symmetric and positive semidefinite, while AA' is symmetric but not necessarily positive semidefinite.

Eigenvectors: Matrices A'A and AA' have the same eigenvalues, but their corresponding eigenvectors are related by matrix A.

1. **Add a steady, well-centered picture of your face to a shared folder alongside your classmates.**

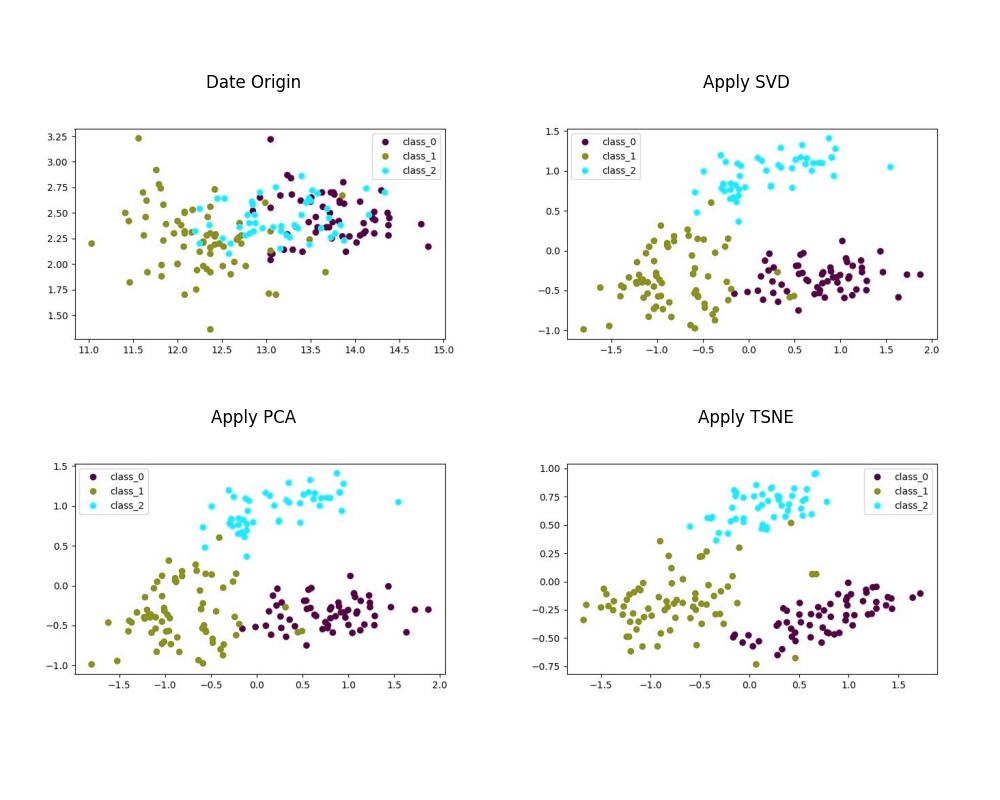
* **How distant is your face from the average? How would you measure it?**

The Euclidean distance from the image to the mean is: 12030.42,

Calculating the MSE: 2208.42

1. **Let’s create the unsupervised Python package**

The following is an example of the application of the unsupervised package with the wine\_data dataset from the sklearn.datasets library



1. **Apply SVD over the picture of your face, progressively increasing the number of singular values used**

* **Is there any point where you can say the image is appropriately reproduced?**

Yes, In my opinion when the image has 32 features

* **How would you quantify how different your photo and the approximation are?**

To determine the similarity we used the SSIM metric from the skimage.metrics library and compared it to the images generated with the following feature quantities

similarity: 1 same imagen, 0 difirent imagen

For imagen with 2 components the similarity is: 0.71

For imagen with 4 components the similarity is: 0.74

For imagen with 8 components the similarity is: 0.80

For imagen with 16 components the similarity is: 0.86

For imagen with 32 components the similarity is: 0.91

For imagen with 64 components the similarity is: 0.95

For imagen with 128 components the similarity is: 0.98

For imagen with 256 components the similarity is: 0.99

note: SSIM compares two images in terms of three characteristics: luminance, contrast and structure. These characteristics are measured using the mean, variance and covariance of the pixels in the images.

1. **Train a naive logistic regression on raw MNIST images to distinguish between 0s and 8s.**

* **What can you tell about the baseline performance?**

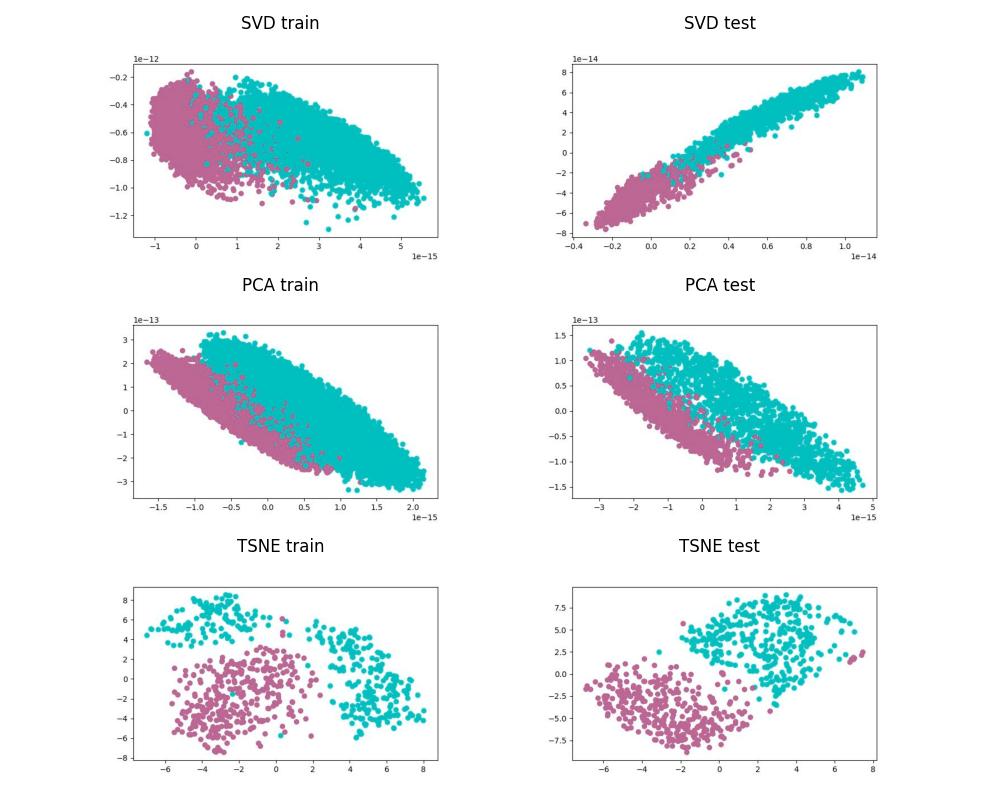
The score property of the LogisticRegression model was used to obtain the accuracy, and the time taken by the algorithm was calculated.

Score Train Mnist 0's & 8's with logist regression: 0.98

Execution time in seconds: 37.8

1. **Now, apply dimensionality reduction using all your algorithms to train the model with only 2 features per image.**

* **Plot the 2 new features generated by your algorithm**

****

* **Does this somehow impact the performance of your model?**

The score property of the LogisticRegression model was used to obtain the accuracy, and the time taken by the algorithm was calculated.

Score for reduction SVD: 0.95

Execution time SVD in seconds: 53.1

Score for reduction PCA: 0.95

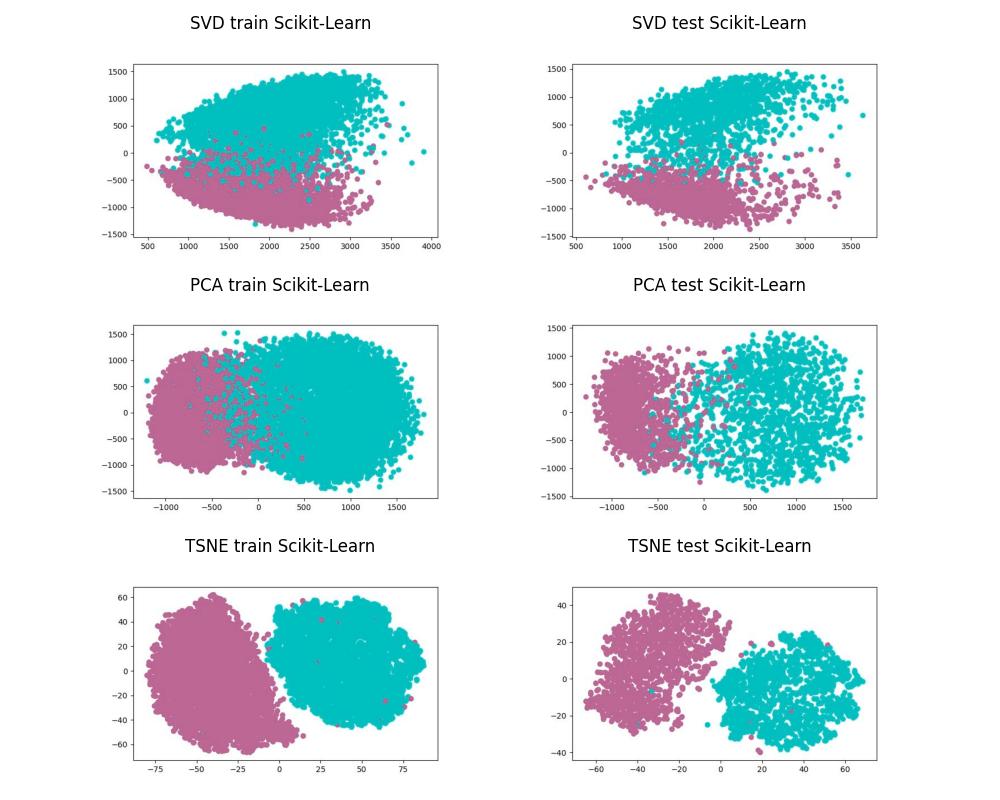
Execution time PCA in seconds: 64.7

Score for reduction TSNE: 0.99

Execution time TSNE in seconds: 38.3

Note: For TSNE, 700 records were used

1. Repeat the process above but now using the built-in algorithms in the Scikit-Learn library.



* How different are these results from those of your implementation? Why?

Score for reduction SVD Scikit-Learn:": 0.95

Execution time SVD Scikit-Learn:": 38.3

Score for reduction PCA Scikit-Learn:": 0.94

Execution time PCA Scikit-Learn:": 36.07

Score for reduction TSNE Scikit-Learn:": 0.93

Execution time TSNE Scikit-Learn:": 83.2

Note: For TSNE Scikit-Learn, all records were used

The results are very similar in terms of accuracy, the unsupervised implementation is slightly better than Scikit-Learn because the exact mathematical definitions were used for dimensionality reduction, the Scikit-Learn library has a better run time and almost equal results because the library is optimized as much as possible.

1. **What strategies do you know (or can think of) in order to make PCA more robust?**

Data standardization: Data standardization is a common technique used in PCA to ensure that all variables have the same scale and are not affected by variability in the data. This helps ensure that the variance in the data is evenly distributed across variables and can make PCA more robust.

Outlier removal: Outliers can have a large impact on the PCA results as they can influence the distance and similarity measures used in the analysis. Therefore, it is important to identify and eliminate outliers before performing PCA.

Cross-validation: Cross-validation is a technique used to assess the robustness of PCA results. Cross-validation can be performed by dividing the data set into a training set and a test set and then comparing the PCA results in both sets. If the results are similar, the PCA is considered more robust.

Testing on different data sets: To evaluate the robustness of the PCA, it is important to test it on different data sets and verify if the results are consistent. If the PCA produces consistent results in different data sets, it can be considered more robust.

1. What are the underlying mathematical principles behind UMAP? What is it useful for?

UMAP (Uniform Manifold Approximation and Projection) is a dimensionality reduction method used to visualize and analyze large datasets in a low dimensional space. UMAP has several applications in the field of machine learning and data mining, such as: Data Visualization, Data Clustering, Time Series Analysis, Spatial Data Analysis.

**mathematical principles**

Riemannian geometry: to model the structure of data. Riemannian geometry is a generalization of Euclidean geometry that is used to model curvature in non-planar spaces. UMAP uses Riemannian geometry to model the curvature of the data and capture the nonlinear structure of the data.

Density functions: to model the distribution of the data. UMAP uses a local density function to measure the density of nearby points and a global density function to measure the density of distant points. UMAP uses these density functions to construct a graph that captures the structure of the data.

Spectral análisis: to reduce the dimensionality of the data. UMAP uses singular value decomposition (SVD) to reduce the dimensionality of the data and heat propagation in a graph to construct a low dimensionality representation space.

Optimization: to find the optimal projection of the data in the low dimensionality representation space. UMAP uses a cost function that measures the discrepancy between the distances in the original space and the low dimensionality representation space. UMAP uses stochastic gradient descent techniques to optimize this cost function.

1. **What are the underlying mathematical principles behind LDA? What is it useful for?**

LDA (Linear Discriminant Analysis) is a dimensionality reduction method used to find a linear combination of variables that maximizes the separation between classes in a data set. LDA has several applications, some of which are described below: Pattern classification, Image recognition, Biomedical data analysis, Market data analysis.

Analysis of Variance (ANOVA): to measure the variance between classes and the variance within classes in a data set. Analysis of variance is used to identify the variables that are most important for class separation.

Probability theory: to model the distribution of variables in each class. LDA assumes that the distribution of variables in each class follows a multivariate normal distribution.

Linear algebra to find a linear combination of the variables that maximizes the separation between the classes. LDA uses singular value decomposition (SVD) techniques to find the eigenvectors that define the optimal linear combination.

Optimization: to find the optimal linear combination that maximizes the separation between classes. LDA uses the Fisher function, which is an objective function that measures the separation between classes.