MATHEMATICAL MODELING AND VISUALIZATION OF MNIST DATA USING EIGEN VALUES AND EIGEN VECTORS

[1]Dr.M. Raja Sekar
Department of Computer Science Engineering (Cyber Security, Data Science, Artificial Intelligence and Data Science)
VNR Vignana Jyoti Institute of Engineering and Technology

[2] P.Veeranjaneyulu
Department of Computer Science Engineering (Cyber Security, Data Science, Artificial Intelligence and Data Science), VNR Vignana Jyoti Institute of Engineering and Technology

[3] G.Ashalatha
Department of Computer Science Engineering (Cyber Security, Data Science, Artificial Intelligence and Data Science), VNR Vignana Jyoti Institute of Engineering and Technology

ABSTRACT:

One of the most basic and probably one of the simplest dimensionality reduction techniques called Principal component analysis. Dimensionality reduction basically, what it does is, if your data points lie in d dimensional space and if you want to transform your data points Xi's , Xi belongs to R^d. We have to reduce d dimensions to d' dimensions such that d'<d, which is called as dimensionality reduction. For MNIST dataset, we have 784 dimensions; if we convert this data in to two dimensions then visualization is very easy. We will apply linear algebraic transformations to convert data from higher dimensions to lower dimension and which is much easier to interpret and understand. We collected MNIST dataset from Kaggle and Kaggle is the phenomenal repository of great datasets.

Keywords: PCA, MNIST dataset, Dimensionality reduction, Geometrical interpretation, Mathematical Interpretation.

DOI Number: 10.14704/nq.2022.20.9.NQ44423 **Neuro Quantology 2022; 20(9):3709-3716**

I. INTRODUCTION

If you have higher dimensional data which you want to visualize, our pair plots, scatter plots and all of them don't work. We can always covert data into two or three dimensions and then visualize [1, 2]. This is certainly one of the most important applications. Principal component analysis is one of the simplest, very fundamental techniques which are extensively used in Data Science. This is one of the actual Data Science techniques which are often used for visualization. So we will see the geometric and mathematical interpretation of Principal component analysis. We can find applications of Principal component analysis in implementing many machine learning algorithms [3], noise filtering, feature engineering and also for feature extraction. It can be considered as non-parametric and un supervised statistical technique [4].

II.GEOMETRIC INTERPRETATION OF EIGEN VALUES AND EIGEN VECTORS

It is easy to visualize data in two dimensions and three dimensions by using scatter plots and four dimensions, five dimensions, and six dimensions by using pairwise plots but in real-time situations [5], we mostly have data set more than ten dimensions. The solution to visualize higher dimensions can be achieved with dimensionality

reduction. Let's take a simple dataset of two dimensions to understand the geometrical interpretation of Principal component analysis. Let me consider a data set with two dimensions for easy interpretation. The results of reducing from higher dimensions to lower dimensions work accurately for original dimensions also. Let's assume I have collected some data and each cross here represents data point Xi and assume we have dataset of students with two features as height and age of the students and assume each feature represents one feature. Here two features corresponding to two dimensions of dataset. Let's see the dataset below in the form of data matrix as well as two dimensional graphs[6,7].

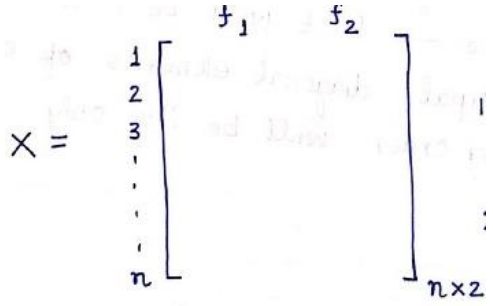


Fig.1. Data Matrix

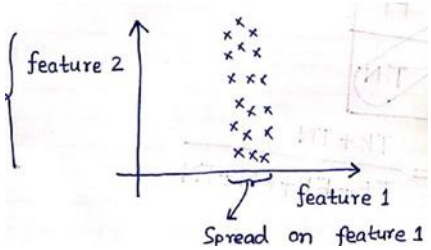


Fig.2. Two dimensional graph

In Figure1, feature1 indicates age of students and feature2 indicates height of the students. If we compare spread of the age and height in a class room we can see that spread of height is more compare to spread of age and which can be seen in Figure2. From Figure2, it is easily interpretable that spread or information or variance on X axis is less than Y axis which means feature1 has less information compare to feature2. There may be situations where we have to represent this data set in one dimension, in such a case we can retain feature2 which holds maximum variance or spread or information. Suppose I want to convert into one dimensional, we would basically take feature f_2 that means our dataset have feature1 and feature2 and we have to create new dataset X' which has to be one dimensional[8,9].

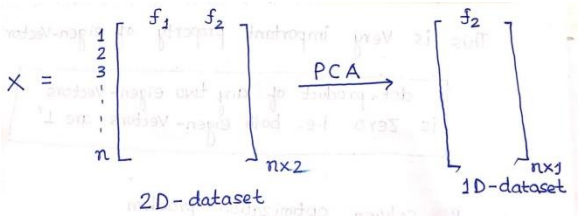


Figure 3. Reducing 2D dataset to 1D-dataset

We would argue that, we just pick feature f2 and just skip whole feature f1 as the spread on feature1 is low which means variability.Figure3 indicates that there is much more wide spread on Y axis (feature2) than on the X axis(feature1). We can see that nx2 matrix is converted into nx1. So what we are doing here, we are preserving the direction with maximal spread or mathematically variance. Now, we will try to understand slightly more complex example. Let's assume, I have two dimensional data. Assume that my dataset has been column standardized. Here X is a two dimensional data in column standardized format. Standardized means, mean{feature1}=mean{feature2}= zero.

Similarly the variance on feature1 equals to variance on feature2 which is equal to 1. Let's assume that my dataset looks like this and each circle represents our data point and feature1 is represented on X-axis and feature 2 is represented on Y-axis[10,11].

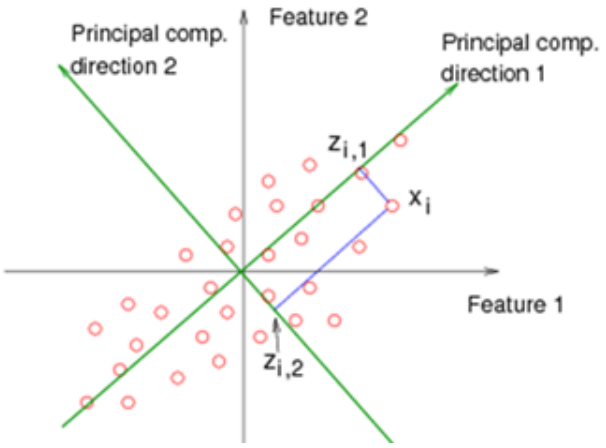


Fig.4. Two dimensional data

It is a two dimensional dataset and we want to transform it into one dimensional. The case which we saw in Fig.2 is very simple because the spread on feature1 and feature2 is very less[12,13]. Now, let's go to slightly tricky case. If you take the data points, what is the spread on feature1, spread on feature1 and feature 2 is fairly wide. Now you

cannot drop either f1 or f2, because there is enough spread on feature1 and feature2. Let's assume Principal component 1 as f1 and let's take one direction which is perpendicular to this and lets call this as f2. Instead of dropping feature1 or feature2, our f1 and f2 are perpendicular; our spread on f2 is very less compare to spread on f1. We have to project all my points xi's onto f1 on which variance is very high then I can successfully convert my two dimensional data into one dimensional data [15, 16].

III. MATHEMATICAL OBJECTIVE FUNCTION OF PCA

For simplicity, let me call F1′ =u1, we know that we represent directions using unit vector. We can make u1 as unit vector.

$||u_1|| = 1$ ----------(1)

Length of the unit vector is 1.

Here u1 is direction on which we project each of our data point. Let's assume, we have a point

$x_i' = \text{proj}_{u_1} x_i$ ----------(2)

and

$D = \{x_i\}_{i=1}^n$ ----------(3)

after projection

$D' = \{x_i'\}_{i=1}^n$ ---------(4)

Now let's project xi'

$x_i' = \text{proj}_{u_1} x_i$ ----------(5)

$x_i' = \frac{u_1 \cdot x_i}{||u_1||}$----------(5)

Now, we know that u1 is a unit vector, which means xi is nothing but

$x_i' = u_1^T x_i$ ----(6)

$x_i' = \text{proj}_{u_1} x_i' = \frac{u_1 \cdot x_i}{||u_1||=1}$-----(7)

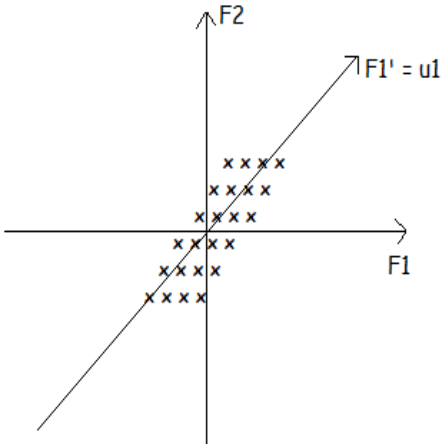


Figure 5: Data in two dimensions with unit vectors

$x_i' = u_1^T x_i$ ----(8)

This is how we represent our dot product and which can be treated as one of the important interpretation of dot product. To find the direction, we have to project the data points along the direction u1. All the points of xi are projected u1. So that after projected transformed points becomes x_i' and each of my xi' is represented in equation 8[17].

So given any point xi, we can convert in the form equation8 using unit vector.

$\bar{x'} = u_1^T \bar{x}$ ----------(9)

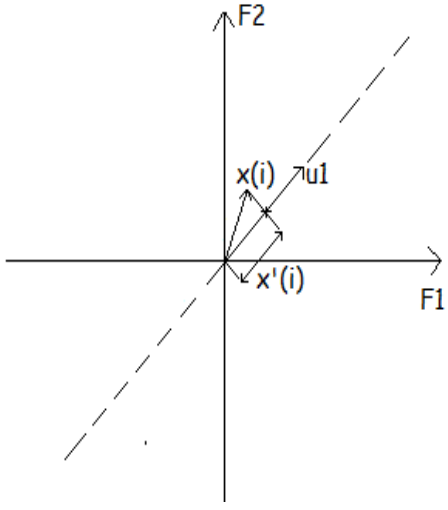


Figure 6: Representation of data points after applying transformation

Find u1 such that

$var\{\text{proj}_{u_1} x_i\}_{i=1}^n$ -----(10)

We have to find u1 such that the variance or spread or information is high.

$var\{u_1^T x_i\}_{i=1}^n$ ----------(11)

$\text{proj}_{u_1} x^i$ ----------(12)

Let's go back and understand write it mathematically, what is the formula for variance, it is the average $\frac{1}{n}\sum_{i=1}^n$ dispersion of each of the points from the mean, that's what variance is.

$$var\{u_1^T x_i\}_{i=1}^n = \frac{1}{n}\sum_{i=1}^n (u_1^T x_i - u_1^T)^2 \text{----------(13)}$$

This is simple formula for variance and here xi is a column vector. Our original dataset is column standardized, so what is column standardized means, it means my mean vector is

$\bar{x} = [0,0,\ldots,0,0]$ ---------(14) because for each column, if mean vector is at 0, then equation 9 becomes zero. Now what is variance?

$$var\{u_1^T x_i\}_{i=1}^n = \frac{1}{n}\sum_{i=1}^n (u_1^T x_i - u_1^T)^2 \text{ ----------(15)}$$

It is average from i=1 to n and now we will write actual mathematical objective that we want to find, we want to maximize the variance

$$max_{u_1} \frac{1}{n}\sum_{i=1}^n (u_1^T x_i)^2 \text{----(16)}$$

We want to find u1, our xi's are part of the data matrix and we have to find u1 such that equation (16) holds good. This is the problem that we have to solve and there is one constraint here, u1 is a unit vector

$$u_1^T u_1 = 1 = ||u_1||^2 \text{ -----(17)}$$

Equation 16 represents objective function and equation 17 represents constraints. This is an optimization problem[18].

IV. MNIST DATASET

Since this paper is about visualizing high dimensional data just like the way IRIS data set data for exploratory data analysis. Let's take another dataset, of course IRIS dataset was a four dimensional dataset and which is not very exciting. Now we want to do data visualization of extremely high dimensions and dimensionality reduction. So the dataset that we will use is called MNIST dataset of characters. It is a large database of handwritten digits. So, each of this is basically a photograph of handwritten digits and so we have characters 0,1,2,…9. Size of each of these images is 28x28.

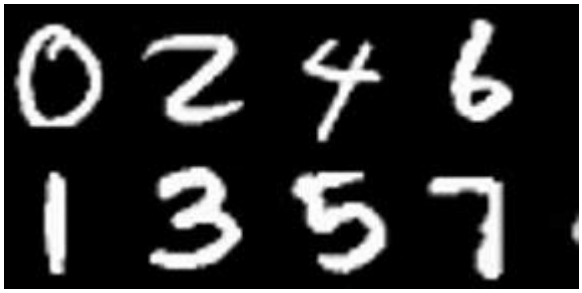


Figure 7: MNIST DATASET

28 pixels are horizontally and 28 pixels are vertically. This is basically computer vision dataset in which we have 60,000 training images where we can have some data points for 5 or for some data points for 0,4 and 7 etc. We are given 10,000 test data points. So we have to build a model using 60,00 training images and 10,000 test images. Let's just use a train dataset 60,000 points where each Xi is basically an image which has 28 pixels horizontally and 28 pixels vertically and Yi could belongs to 0,1,2,3,4,5,6,7,8,9. Objective is the written character into one of the 10 numeric characters. We will try to understand how to represent an image in the form of matrix[19].

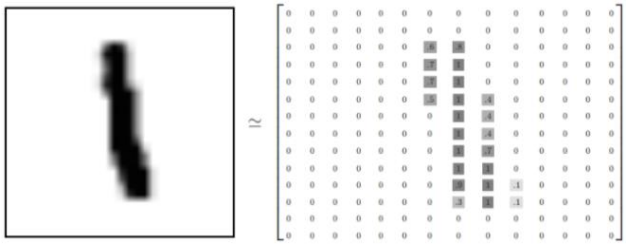


Figure8. Representation of Digit 1 image internally

If it is fully dark, we will give value 1 and if it slightly gray, we will give value between 0 and 1like 0.1 for light gray, 0.4 for dark gray, 0.7 still dark gray. 1 is fully black. If it is white, we can put a value of 0. We have an image which is converted into numerical matrix and size of matrix is 28x28. This matrix is not your data matrix. This is one data point, now the question is since we have converted an image into a data matrix. How do I go to a data vector or data point, this is not data matrix and this is just matrix representation of your image. Let's see how we convert this matrix of pixels into a vector. Given that we have 28

pixels by 28 pixels and we do something called flattening. We convert each image into a data point by using roe flattening or column flattening and resultant column vector will have 784 rows and 1 column which mean we get 784x1 dimensional vector for each data point. Each image or each data point is now 784 dimensional which means we have feature1, feature2, feature3,..., feature 784.Each pixel value can be considered as a feature. Each column vector is converted into row vector by taking transpose and this resultant row vector is the first row of data matrix. In this way all the training images are represented as rows of data matrix. Our training data matrix X is of order 60k by 784 and we have 784 dimensional data set[20].

V. LOADING MNIST DATASET

We collected MNIST dataset from Kaggle and Kaggle is the phenomenal repository of great datasets. This dataset is called digit recognizer dataset on Kaggle and it has two datasets training and testing. We can download mnist_train.csv and mnist_test.csv from Kaggle website. We have to import standard libraries like numpy, pandas and matplotlib which are extensively used in Data Science. We can read CSV file in pandas and where pd is alias of pandas[21].

```
# MNIST dataset downloaded from Kaggle :
#https://www.kaggle.com/c/digit-recognizer/data
# Functions to read and show images.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
d0 = pd.read_csv('E:/AppliedDataScience with Python/train.csv')
```

```
   label  pixel0  pixel1  pixel2  pixel3  pixel4  pixel5  pixel6
pixel7  \
0      1       0       0       0       0       0       0       0
0
1      0       0       0       0       0       0       0       0
0
2      1       0       0       0       0       0       0       0
```

```
     pixel780  pixel781  pixel782  pixel783
0           0         0         0         0
1           0         0         0         0
2           0         0         0         0
3           0         0         0         0
4           0         0         0         0

[5 rows x 785 columns]
```

We can see the pixels from 0 to 783 and also there is a label and total number of columns including label is 785, as we are using head command in python, it is showing that first five rows. Dataset is like this in which first column is label which says what the digit is and all other columns represents pixel values. We have pixels from index 0 to index 783 and these 784 pixels are nothing but our 28x28 pixels. We created a vector of size 784 and total 784 pixels with the label. There is a value of each of these and we can represent this as single vector by going row wise, first it stores all of these rows. I have 10 classes here 0, 1, 2...9. Every row corresponding to label will be stored in a variable called l and we are creating a new variable called d, we are saying drop label and keep every other column that we have. So drop the label column and what we have here is your l stores all class labels and d stores all data. So corresponding to d_i, if I take the i^{th} row of d, its corresponding label will be in l[i]. We just ensured that all my labels are in one variable and the data is in another variable.

```
# save the labels into a variable l.
l = d0['label']

# Drop the label feature and store the pixel data in d.
d = d0.drop("label",axis=1)
```

There are 42,000 data points and each data point has 784 dimensions corresponding to the 28x28 pixels, our l is just the label and label is just one column. I just have 42,000 rows of data with only one column and this is basically sanity check to ensure that everything is going well. Now I can plot a number and see if it makes sense, plot a number as its index 100. In d[100], I have 784 pixel data, in the 100 index of my d. Similarly in l[100], I have what is the actual label. We want to visualize this data, not visualizing in the high dimensional and we just want to visualize 784 dimensional data into a grid. We want to verify that whether the visualization that we noticed is same as the label

```
# display or plot a number.
plt.figure(figsize=(7,7))
idx = 100

grid_data = d.iloc[idx].to_numpy().reshape(28,28)  # reshape from 1
plt.imshow(grid_data, interpolation = "none", cmap = "gray")
plt.show()

print(l[idx])
```

We are using matplotlib and figure size should be 7x7 and matplotlib has a function called imshow which shows images and we are saying the data for this image is the grid data of 28x28 pixels. The map is C which stands for color map and the color map is gray because I want to draw a gray scale image.

VI. CODING

We pick first 15000 images of hand written digits to work on for time efficiency and also we displayed its shape.

```
# Pick first 15K data-points to work on for time-effeciency.


labels = l.head(15000)
data = d.head(15000)

print("the shape of sample data = ", data.shape)
```

```
the shape of sample data =  (15000, 784)
```

Sklearn library is useful for data preprocessing which can standardize the data in which each attribute standard deviation 1 and mean is zero. Here we are importing Standard Scalar from sklearns preprocessing library. Fit transformation function is used for this purpose and also displayed shape of the data which remains same.

```
# Data-preprocessing: Standardizing the data

from sklearn.preprocessing import StandardScaler
standardized_data = StandardScaler().fit_transform(data)
print(standardized_data.shape)
```

After standardizing the data, we are storing it in a variable sample_data and covariance matrix can be calculated by multiplying sample_data.T with sample_data and The shape of variance matrix = (784, 784).

```
#find the co-variance matrix which is : A^T * A
sample_data = standardized_data

# matrix multiplication using numpy
covar_matrix = np.matmul(sample_data.T , sample_data)

print ( "The shape of variance matrix = ", covar_matrix.shape)
```

We can import eigh from scipy.linalg which is useful to find eigen values and eigen vectors of covariance matrix. Shape of the eigen vector is 784x2 and updated shape of the eigen vector is 2x784.

```
from scipy.linalg import eigh
values, vectors = eigh(covar_matrix, eigvals=(782,783))

print("Shape of eigen vectors = ",vectors.shape)
# converting the eigen vectors into (2,d) shape for easyn
vectors = vectors.T

print("Updated shape of eigen vectors = ",vectors.shape)
# here the vectors[1] represent the eigen vector correspo
# here the vectors[0] represent the eigen vector correspo
```

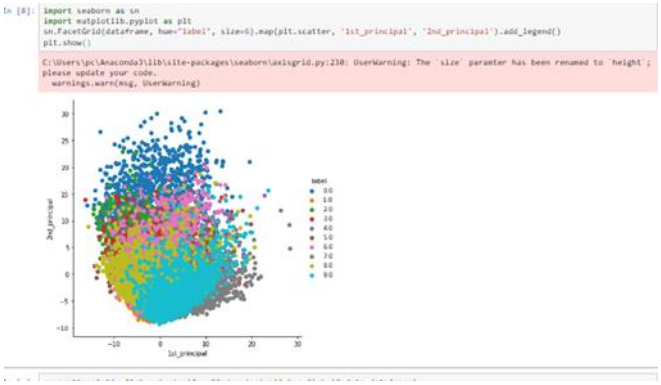
```
Shape of eigen vectors =  (784, 2)
Updated shape of eigen vectors =  (2, 784)
```

We can project the original data sample on the plane with the help of two principal eigen vectors by vector multiplication and here we are multiplying 2x784 vectors with sample_data.T, so that the resultant matrix is of order 2x15000.

Append the labels to 2D projected data and new coordinates are created by vstack. Created a new dataframe for plotting the labeled points with the help of pandas of python.

	1st_principal	2nd_principal	label
0	-5.558661	-5.043558	1.0
1	6.193635	19.305278	0.0
2	-1.909878	-7.678775	1.0
3	5.525748	-0.464845	4.0
4	6.366527	26.644289	0.0

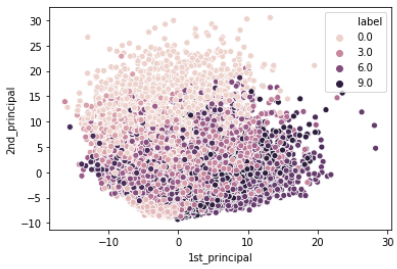
Here sn is alias of seaborn and plt alias of matplotlib.pyplot. FacetGrid is used to display the data for which we gave label is the value of hue parameter. In scatter plot, first principal component is on X axis and second principal component is on Yaxis and also we used legend.



This is the visualization of

```
sn.scatterplot(x="1st_principal",y="2nd_principal",hue="label",data=dataframe)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fa43ed3688>
```



VII. VISUALIZATION

We struggled even to visualize four dimensional dataset When we have four or five or six dimensional dataset, we can use things like pair plot but as soon as this dimensionality increases, even when it becomes 10 dimensional pair plots become almost useless. Now here is the most

interesting thing. How do we now visualize a 784 dimensional dataset.

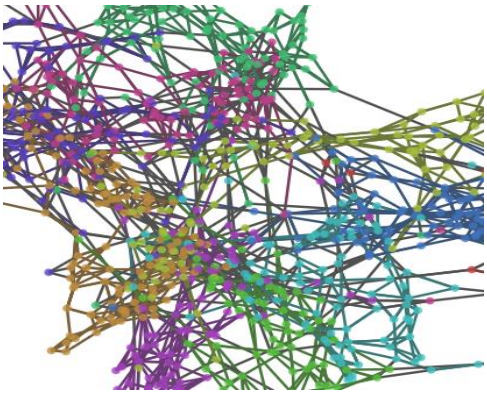


Figure 9 Visualizing 784 dimensional data

In figur10, each dot represents a data point. Each dot represents a 784 dimensional data point; of course we cannot visualize 784 dimensions and we are projecting it into two dimensions, X axis is dimension1 and Y axis dimension2. So what we have done we have taken a data with 784 dimensions and converted this into two dimensions.

From Fig.11, we can see that all the red points are zeros, all blue points are sixes, and dark green is 3 and browns are 1. This is the dimensionality reduction where I am taking a data which is extremely high dimensional and we are converting into a very low dimensional space and when I visualize this data and when we color these points based on Yi, its looks very logical all our four are in one region, all zeros are in another region, all twos are in some other region. So this gives me lot of intuition on how the data in the 784 dimensional space

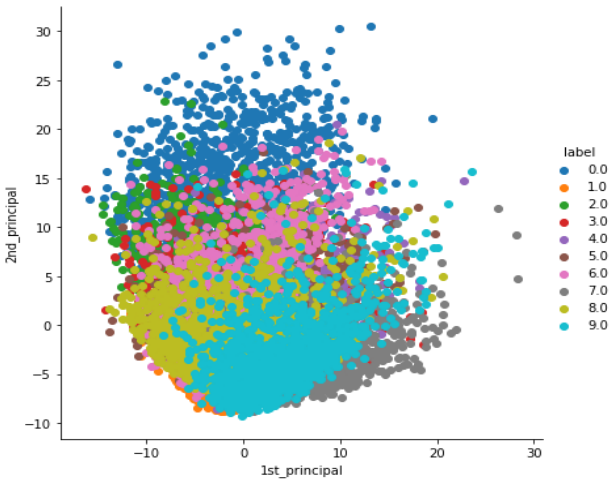


Figure10. Color code of digits with legend

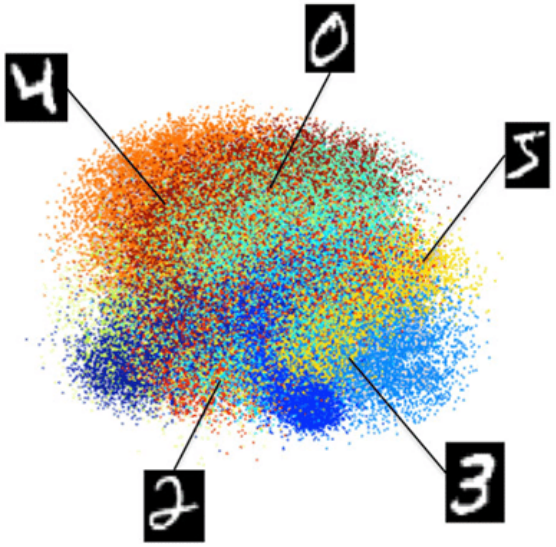


Figure11. Color code of digits without legend

VIII. CONCLUSION

Linear Algebra is very fundamental area in mathematics but with respect to Data Science, we used to solve specific types of problems. Since we have understood eigen values and eigen vectors for Data Science applications and also seen how how eigen values and eigen vectors perform in understanding MNIST dataset but in visualizing this dataset there is no clean separation between digit 6 and digit 0 which can be addressed by t-SNE. Also if both top eigen values are equal, it is difficult reduce the dimension. Our Eigen values and Eigen vectors may not play crucial role when we apply clustered datasets.

IX. REFERENCES

[1] Pearson K. 1901 On lines and planes of closest fit to systems of points in space. Phil. Mag. 2,559–572. (doi:10.1080/14786440109462720)

[2] Hotelling H. 1933 Analysis of a complex of statistical variables into principal components.J. Educ. Psychol. 24, 417–441, 498–520. (doi:10.1037/h0071325)

[3].Cadima J, Cerdeira JO, Minhoto M. 2004 Computational aspects of algorithms for variableselection in the context of principal components. Comp. Stat. Data Anal. 47, 225–236.(doi:10.1016/j.csda.2003.11.001)

[4] Gill PG, Purnell MA, Crumpton N, Brown KR, Gostling NJ, Stampanoni M, Rayfield EJ. 2014 Dietary specializations and diversity in feeding ecology of the earliest stem mammals. Nature 512, 303–305. (doi:10.1038/nature13622)

[5] R Development Core Team. 2015 R: A language and environment for statistical computing. Vienna,Austria: R Foundation for Statistical Computing. See http://www.R-project.org..

[6] Gabriel KR. 1971 The biplot graphical display of matrices with application to principal component analysis. Biometrika 58, 453–467. (doi:10.1093/biomet/58.3.453)

[7] Dr. M Raja Sekar. 2017. Optimization of the Mixed Model Processor Scheduling.International Journal of Engineering Technology and Computer Research (IJETCR). 5(5): 74-79.

[8] Dr. M Raja Sekar. 2017. Fuzzy Approach to Productivity Improvement. International Journal of Computer & Mathematical Sciences. 6(9): 145-149.

[9] Dr. M Raja Sekar et al. 2016. An Effective Atlas-guided Brain image identification using X-rays. International Journal of Scientific & EngineeringResearch. 7(12): 249-258.

[10] Dr. M Raja Sekar. 2017. Fractional Programming with Joint Probability Constraints.International Journal of Innovations & Advancement in Computer Science. 6(9): 338-342.

[11] Dr. M Raja Sekar. Solving Mathematical Problems by Parallel Processors. Current Trends in Technology and Science. 6(4): 734-738.

[12] Cadima J, Jolliffe IT. 2009 On relationships between uncentred and column-centred principalcomponent analysis. Pak. J. Stat. 25, 473–503.

[13] Birnbaum A, Johnstone IM, Nadler B, Paul D. 2013 Minimax bounds for sparse PCA withnoisy high-dimensional data. Ann. Stat. 41, 1055–1084. (doi:10.1214/12-AOS1014)

[14] Johnstone IM, Lu AY. 2009 On consistency and sparsity for principal components analysisin high dimensions. J. Am. Stat. Assoc. 104, 682–693. (doi:10.1198/jasa.2009.0121)

[15] Rao CR. 1958 Some statistical methods for comparison of growth curves. Biometrics 14, 1–17.(doi:10.2307/2527726)

[16] Dr. M Raja Sekar. 2017. Interactive Fuzzy Mathematical Modeling in Design of Multi-Objective Cellular Manufacturing Systems. International Journal of Engineering Technology and Computer Research (IJETCR). 5(5): 74-79.

[17] Dr. M Raja Sekar. 2017. Breast Cancer Detection using Fuzzy SVMs. International Journal for Research in Applied Science & Engineering Technology (IJRASET).5(8): 525-533.

[18] Dr. M Raja Sekaret al. 2017. Areas categorization by operating support Vector machines. ARPN Journal of Engineering and Applied Sciences. 12(15): 4639-4647.

[19] Dr. M Raja Sekar et al. 2017. Tongue Image Analysis For Hepatitis Detection Using GA-SVM. Indian Journal of computer science and Engineering. 8(4).

[20] Boente G, Silibian-Barrera M. 2015 S-estimators for functional principal components. J. Am.Stat. Assoc. 110, 1100–1111. doi:10.1080/01621459.2014.946991).

[21]. Vichi M, Saporta G. 2009 Clustering and disjoint principal component analysis. Comp. Stat.Data Anal. 53, 3194–3208. (doi:10.1016/j.csda.2008.05.028).