DATA MINING AND PREPARATION

MSc INFORMATION SYSTEMS AND SERVICES

SPECIALIZATION: BIG DATA AND ANALYTICS

«PRINCIPAL COMPONENT ANALYSIS (PCA)

SINGULAR VALUE DECOMPOSITION (SVD)»

PANAGIOTAKOPOULOS GEORGIOS (ME2030)

SUPERVISOR

PHILIPPAKIS MICHAEL

Abstract

In the bibliographic review, we undertake the study of extraction techniques and feature selection, dimensionality reduction (Principal Component Analysis, Singular Value Decomposition, Linear Discriminant Analysis), as well as the study of Logistic Regression methods.

Table of contents

Figure Index

1. Feature extraction (PCA-SVD-LDA)-Logistic regression

1.1. Feature Extractions

Nowadays, we are flooded with a large amount of data which we are called to process and draw conclusions. Many times, this data is described by variables in a form not suitable for direct processing, and, despite the increased computing power of today's computers, it is necessary to find a way to take the data and transform it to power the various engineering models. learning that we implement.

Although sometimes the initial input variables are suitable for entering the model and drawing conclusions, nevertheless, several times, their transformation is necessary in order to achieve better performance of the model. In addition, some machine learning algorithms are designed to work only with certain types of input variables, so here too feature extraction can play a decisive role. Existence of many dimensions (variables) in the input data can lead to problems in the machine learning algorithms due to the higher costs and requirements in computing power and memory.

The purpose of exporting feature extractions (variables) is to transform and display the original input data variables into a smaller set of dimensions, while trying to retain most of the information contained in the original data (Nasreen, 2014).

Through the export of features, a new data set is created which results through transformations, combinations and conversion of the original variables into new ones that will feed the model and ensure faster and more accurate results.

Reducing the dimensions resulting from the application of feature extraction techniques, when applied prior to categorization techniques, can help reduce the likelihood of the model being "overfitting", leading to classification errors and decreased performance. Character extraction techniques can also help reduce the negative impact of model performance on the existence of observations that have been incorrectly assigned to a class other than the one to which they actually belong..

A disadvantage of feature extraction techniques is that we can not understand the physical significance of the transformed data, even though they are a linear combination of the originals. Also, we are not able to understand the contribution that an original variable has, after its projection in the new dimensions (Nasreen, 2014).

Some important and widely used dimensionality reduction techniques are:

- *Principal Component Analysis* - PCA
- *Singular Value Decomposition* - SVD
- *Linear Discriminant Analysis* – LDA

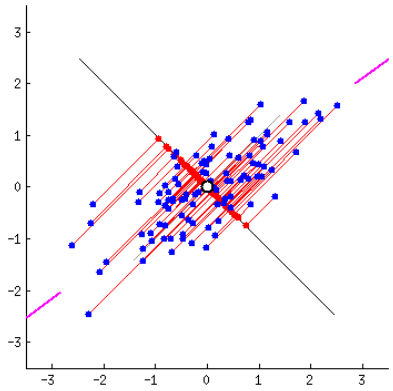
which will be studied next.

2. PCA

Principal Component Analysis is a mathematical process that uses a rectangular transform to convert a set of observations of certain variables that are likely to be related to a set of values of unrelated variables. These unrelated variables are called principal components (Bro & Smilde, 2014).

The principal components are essentially the subset of the original components, which hold the most information about the variability of the data set (Sharma, 2020).

The method is used only when data reduction or exploration is sought, ie it can not be applied to confirmatory analysis. At the same time, this method aims to find linear unrelated rectangular axes which are considered to be known as main components in the field of dimensions. More specifically, the data points on these main components are displayed there. The first component records the largest data variation. Let us intuitively understand the PCA method by adapting it to a two-dimensional data matrix that can be represented by a two-dimensional scatter plot (Wang, 2019).



Given the fact that all the main components are orthogonal to each other, we can use a pair of rectangular axes in two-dimensional space as two components. To record the largest variance of the first component, we rotate the pair of components to make one of them optimally align with the spread of the data points. Then all the points of the data can be displayed in the components, and their projections like the red dots in component 1 are considered to be essentially the resulting representation of the reduced data set. The panel shrank from two-dimensional to one-dimensional, while

maintaining greater variability. The components can be determined by the eigenvalue of the covariance table C. In the final analysis, as a geometric concept of the specific value, is considered the finding of a new system of coordinates of the eigenvectors of the covariance C, through rotations (Wang, 2019).

$$C = W\Lambda W^{-1}$$

In the above equation, the covariance table C, which is of dimension mxm, is decomposed into a table of eigenvectors W of dimension also mxm and a diagonal table of eigenvalues L. The eigenvectors, which are the columns of the table W, are the main components we are looking for. In order to achieve the dimensionality, the points representing data are displayed in the first k main components, as follows::

$$X_k = XW_k$$

Python supports (via numpy library) the decomposition of a table into eigenvalues and eigenvectors (numpy.linalg.eig). Using this function, the following function can be written, which returns the table of eigenvectors X (Sharma, 2020).

```
import numpy as np


def pca(X):
# Data table X
# requires center 0 (0-centered)
n, m = X.shape
assert np.allclose(X.mean(axis=0), np.zeros(m))


# Calculation of the variance table
C = np.dot(X.T, X) / (n-1)
# Decomposition into eigenvalues and eigenvectors
eigen_vals, eigen_vecs = np.linalg.eig(C)
# View in the area of the main components
X_pca = np.dot(X, eigen_vecs)
return X_pca
```

2.1. Fields of implementation of the PCA method

Data Visualization: The data sets that emerge from current applications and need to be processed are often huge. In order to extract essential information from them, the characteristics of the set must be identified which determine this data. The solution of

a problem in which the central factor is its data, arises the need to make an extensive analysis of this data, in order to emerge the correlations between them and / or the distribution of some of these variables. Given that such data are distributed over a large number of variables (or dimensions), their representation in statistical charts is often impossible (Sharma, 2020).

In order to reduce the number of components into which the data is distributed, the application of PCA results in the projection of the data in smaller dimensions, so that it is possible to create graphs in two-dimensional or three-dimensional space, so that it can be understood by humans.

Speeding up the Machine Learning process: In the event that a machine learning algorithm uses multidimensional data, the time required for learning and the necessary controls becomes large, as a result of the slow execution of the machine learning algorithm.

Limiting the dimensions of the data set used by the PCA method, leads to a smaller data set, while accelerating the learning algorithm (Sharma, 2020).

The Principal Component Analysis (PCA) method Uses the principles of Linear Algebra to display the values of a data set described by a number of variables (dimensions) that are likely to be related to the values of a smaller number of unrelated dimensions. (Παπαδουράκης & Μαριάς, 2017). In many cases, the variables that describe a set of data are not all equally useful for drawing conclusions in practice. Thus, we may have some variables that do not contribute enough or behave similarly to other variables in the data set. Thus, these variables that behave in a similar way and do not contribute enough to the variability of the data, can be omitted. The PCA method replaces these variables with new, fewer ones, with each new variable ("principal component") being a linear combination of the variables it replaces. The new dimensions are not related to each other. Using the reduced set of variables to train a machine learning model, it is very likely that it will be able to draw conclusions much faster and more accurately than when we fed it to the original data set. Thus, the PCA method removes the noise and compresses the original data, making the machine learning algorithms more efficient. One such class of algorithms that benefit from the PCA technique for data compression is image recognition and processing algorithms, whereby compressing the original image, which results from the reduction of variables (ie image pixels), we achieve smaller size and faster image processing. Similarly, regression algorithms could conclude faster than processing a data set with a reduced number of dimensions compared to the original.

In addition, it is very useful to visualize our data in order to better understand it and identify any patterns in it. But when a data set is described from more than three dimensions, it is extremely difficult to get an overview of the data behavior, taking into

account all the features at once. Thus, we can apply the PCA method to reduce the number of dimensions to two or three and thus be able to visualize them.

2.1.1. Steps of the method

To apply the PCA method, we perform the following steps (Jaadi, 2020), (Russell, 2011):

1. We define our data set. Suppose our data set is an array of dimensions 3x2:
   $A = \begin{bmatrix} 8 & 10 \\ 12 & 14 \\ 16 & 18 \end{bmatrix}$
   Our data are defined in two columns (dimensions), with values [8, 12, 16] for the first column and [10, 14, 18] for the second column.
2. We subtract the average value of each column from the value of each column. In our case, the vector of the mean value is [12 14]. This is how the data set is made:
   $C = \begin{bmatrix} -4 & -4 \\ 0 & 0 \\ 4 & 4 \end{bmatrix}$
3. We calculate the correlation table that results from our new data table. The diagonal elements of the table are the variance of each column (dimension), while the other elements are the covariance of each column with the other column. The covariance table for our case is:
   $Cov = \begin{bmatrix} 16 & 16 \\ 16 & 16 \end{bmatrix}$
4. Next, we calculate the eigenvalues and the eigenvectors of the covariance table. In our case, the eigenvalues are $\lambda_1 = 32$ and $\lambda_2 = 0$. The eigenvectors that correspond to these eigenvalues are:
   $V = \begin{bmatrix} 0.70710678 & -0.70710678 \\ 0.70710678 & 0.70710678 \end{bmatrix}$
   Each column of the table of eigenvectors is also a principal component. The eigenvector corresponding to the largest eigenvalue, contains the most information (variability) of our data set.
5. We select all or some of the eigenvectors (main dimensions) to create the "feature vector".
6. We transform (display) our data to new dimensions. To do this, we calculate the inverse of the table of components and thus, we have an eigenvector in each line, with the most important being in the first row of the array (array FV^T). Next, we multiply this array by the inverse array of our data from which we have subtracted the mean (C^T). Thus, the table of our transformed data is calculated by the formula: $P = FV^T * C^T$. In our case they are:
   $FV^T = \begin{bmatrix} 0.70710678 & 0.70710678 \\ -0.70710678 & 0.70710678 \end{bmatrix}$

   $C^T = \begin{bmatrix} -4 & 0 & 4 \\ -4 & 0 & 4 \end{bmatrix}$

$$P = \begin{bmatrix} -5.656854254 & 0 & 5.65685425 \\ 0 & 0 & 0 \end{bmatrix}$$

and finally:

$$P^T = \begin{bmatrix} -5.656854254 & 0 \\ 0 & 0 \\ 5.65685425 & 0 \end{bmatrix}$$

In our case we retained both the principal components and projected our data on them. We observe, however, that our initial data with dimensions of 3x2, can be expressed in space of dimensions of 3x1, so only the first principal component is enough in our case to view our data, without loss of variability.

2.2. PCA implementation in Wisconsin Breast Cancer Dataset

The PCA method was used to visualize a data set with measurements of patients with malignant and benign breast cancer. The method was used since the data set has 30 components, so it could not be plotted. The dataset is available at the source:

https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/.

The total contains 569 entries, 212 malignant and 357 benign.

The application that will be presented below uses Python to reduce the dimensions of the data set from 30 to 2, in order to visualize it at the level.

2.2.1. Data processing and PCA implementation

In the first phase, the data is normalized. Normalization is done using the StandardScaler section, which is included in the sklearn library. After the normalized values are obtained, then they are scaled, using the fit_transform function, on the values of the attributes.

With the implementation of StandardScaler, each attribute of the data set will be distributed based on the normal distribution, so that their distribution has an average value of 0 and a standard deviation of 1.

```
# Normalization of feature values
# of the total data set
x = StandardScaler().fit_transform(x)
# normalizing the features
print (np.mean(x),np.std(x))
```

The 30-dimensional data in a two-dimensional space of the two main components should then be displayed.

For this purpose the sklearn library has the PCA module, which in turn has the PCA method. The PCA method takes as a parameter the number of main components we want and exports a dataframe with the values of the main components for all measurements of the data set.

```
# Display at the level of the 2 main components
pca_data = PCA(n_components=2)
PC = pca_data.fit_transform(x)
print(PC)
```

Using the explained_variance_ratio_ function, we can deduce the percentage of primary information retained by each major component, after viewing the data set in a smaller field.

```
# Display a percentage of the original data retained in the main components
print('Percent of data held per principal component: ' + str(pca_data.explained_variance_ratio_))
```

The result is:

```
Percent of data held per principal component: [0.44140602 0.19056427]
```

The above result shows that the first main component retains 44.14% of the original information and the second main component 19.05%. The set of therefore the original information retained is:

44,14% + 19,05% =63,19%

This leads us to the conclusion that 36.81% of the original information is lost.

a. Visualization of results

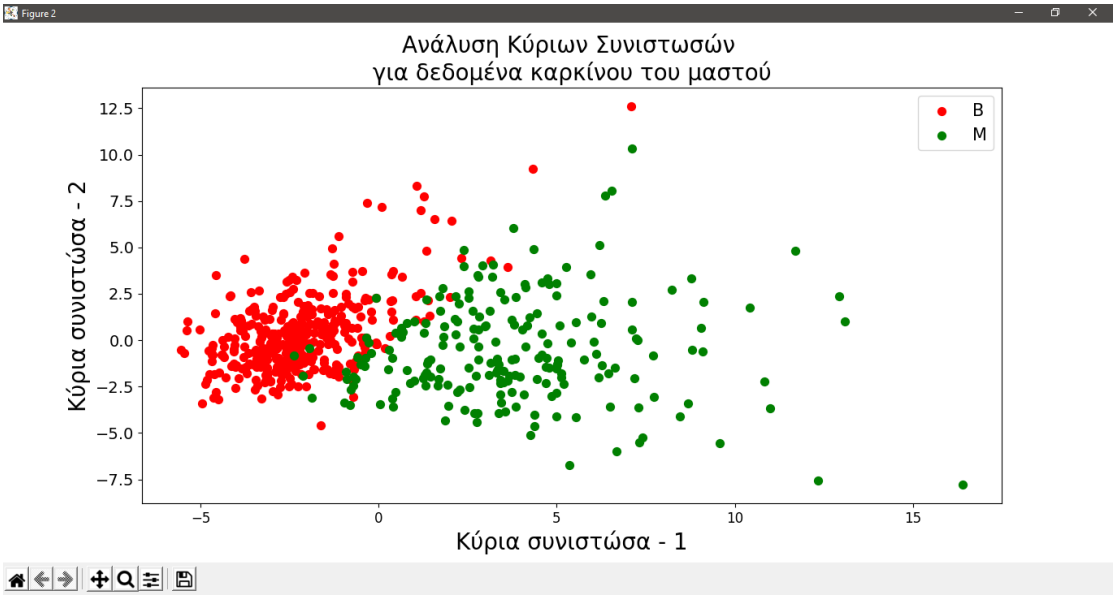
Then the values of the two main components are visualized in two axes. The horizontal corresponds to the first main component and the vertical to the second main component.

The code that implements the visualization is the following:

```
# Diagram of the principal components
plt.figure()
plt.figure(figsize=(10,10))
plt.xticks(fontsize=12)
plt.yticks(fontsize=14)
plt.xlabel('Κύρια συνιστώσα - 1',fontsize=20)
plt.ylabel('Κύρια συνιστώσα - 2',fontsize=20)
plt.title("Ανάλυση Κύριων Συνιστωσών \ηγια δεδομένα καρκίνου του μαστού",fontsize=20)
targets = ['B', 'M']
colors = ['r', 'g']
for target, color in zip(targets,colors):
    indicesToKeep = data['type'] == target
    plt.scatter(PC_df.loc[indicesToKeep, 'PC 1']
               , PC_df.loc[indicesToKeep, 'PC 2'], c = color, s = 50)

plt.legend(targets,prop={'size': 15})
plt.show()
```

The diagram drawn is as follows:



Observing the graph, we conclude that the two measurement classes (Bening (B) and Malignant (M)), when projected in two-dimensional space, are linearly distinct to some extent.

2.2.2. Another application study of the method

In this chapter, we will deal with the study of an application using the PCA method, and specifically for the visualization of a data set. We will deal with a set of data which concerns measurements of various features of a set of wheat grains [1] (Charytanowicz, et al., 2012).

The data set contains values from seven different attributes for each observation, which are: "area", "perimeter", "compactness", "length", "width", "asymmetry coefficient" and "length of kernel groove" . Attribute values are real numbers. Each grain of wheat studied belongs to one of three different varieties (categories) numbered 1, 2 or 3. Due to the fact that the data set is expressed in seven dimensions, it is not easy to visualize it and have an overview of the dispersion of the three categories. Therefore, we can apply the PCA method to display the data in two new dimensions, unrelated to each other, while retaining most of the data variability. Then it will be possible to visualize our data.

The figure below shows the values for the first five observations of our data.

[1] Source: http://archive.ics.uci.edu/ml/datasets/seeds

	area	perimeter	compactness	kernel_length	kernel_width	asymm_coeff	kernel_groove_length	class
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220	1
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	1
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825	1
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	1
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	1

Figure 1. Observations of the seeds data set in its 7 dimensions

Applying the PCA method (using the scikit-learn package in Python programming language) we can view the data in 2 dimensions (principal components). The following figure shows the values from the first five observations after viewing the data in the two principal components.

	PC1	PC2	class
0	0.317047	0.783669	1
1	-0.003386	1.913214	1
2	-0.459443	1.907225	1
3	-0.591936	1.931069	1
4	1.102910	2.068090	1

Figure 2. The Principal Components that were created after the application of the PCA method

We are now able to visualize our data set, something that can be seen in the next Figure.

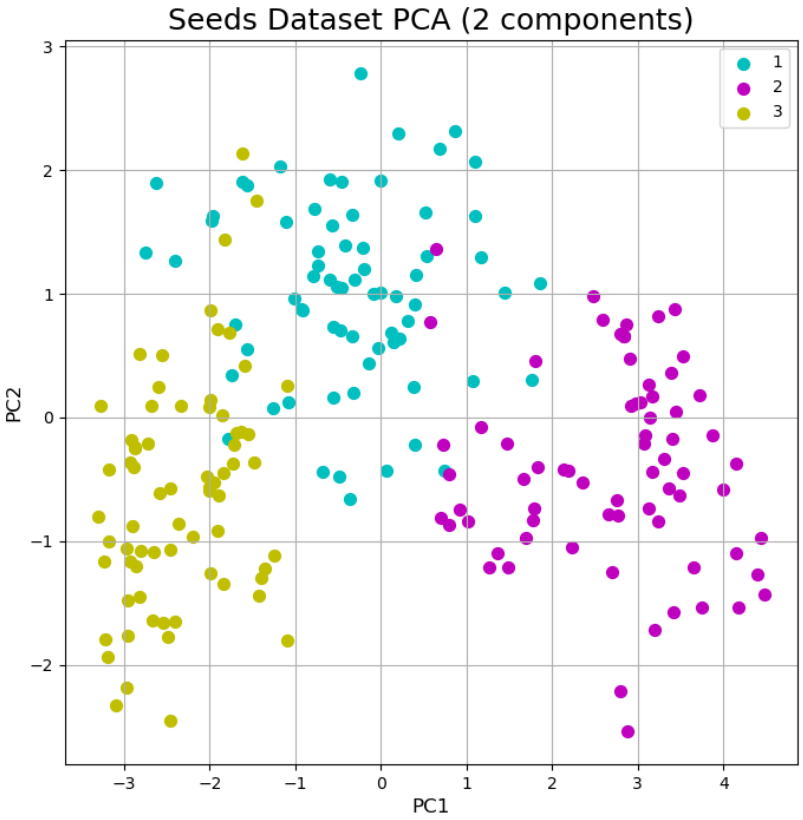


Figure 3. View the seeds data set in the two Principal Components

We observe that the three different categories representing the different varieties of wheat grains in our data set are relatively distinct.

Through this concise study of the Principal Component Analysis method, we conclude that it is a valuable tool in pre-processing datasets for machine learning problems and drawing conclusions. Through this method, we can apply compression to the original data without significant loss of information, thus helping the algorithms to process the data sets faster, but also enabling us to transform data sets and visualize them, understanding the best.

3. Singular Value Decomposition (SVD)

Singular Vaalue Decomposition, known as SVD, is one of the most popular dimensionality reduction techniques in machine learning. It reduces the number of features of a data set by reducing the space dimension from N-dimension to M dimension (N> M). In the field of composite systems SVD is a collaborative table factorization technique.

The analysis in eigenvalues and eigenvectors can be done for any table, square or not. Let A be an array m × n, then the analysis consists of a factorization as follows:(Kalman, 1996)

$$A = U\Sigma V^T \text{ με } U^T U = UU^T = I \text{ και } V^T V = VV^T = I,$$

Where array Σ contains eigenvalues from array A: {σ1, . . . , σr} whose number is equal to the order r of array A. Eigenvalues are all positive. array U is m × m, array V is n × n and array Σ is m × n. More specifically, the last array is divided into blocks as follows:

$$\Sigma = \left[\Sigma_+ O_{r\times(n-r)} O_{(m-r)\times(n-r)}\right], \text{ με } \Sigma_+ = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \sigma_r \end{bmatrix}$$

To find U, V and Σ it is necessary to find the characteristic sizes of arrays A^T A και AA^T. We will have:

$$A^T \text{A} = \text{V}\Sigma^T U^T U\Sigma V^T = \text{V}\Sigma^T \Sigma V^T$$

Given that the table A^TA is symmetric, the rectangular array V includes the eigenvectors of the array A^TA and the diagonal array $\Sigma^T\Sigma$ includes its eigenvalues. It will be more specific:

$$\Sigma = \left[\Sigma_+ O_{r\times(n-r)} O_{(m-r)\times(n-r)}\right], \text{ με } \Sigma_+ = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \sigma_r \end{bmatrix}$$

So, the array Σ_+ contains the positive square roots of the eigenvalues of A^TA. In the same way it is proved that array U contains the eigenvectors of AA^T. The eigenvectors in the rectangular array U are called the left eigenvectors of A, while the eigenvectors in the array V are called the right eigenvectors. The eigenvectors should correspond to the eigenvalues, and in addition there should be a correspondence between the eigenvectors in arrays U and V as follows: $u_i = \frac{1}{\sigma_i}\text{A}v_i$ και $v_i = \frac{1}{\sigma_i}A^T u_i$

The correspondence between eigenvalues and eigenvectors is also expressed in the relations:

$$u_i^T \, \mathrm{A} v_i = v_i^T A^T u_i = \sigma_i$$

Με την ανάλυση σε ιδιάζουσες τιμές έχουν ταυτόχρονα προσδιορισθεί ορθογώνιες βάσεις για τους τέσσερις θεμελιώδεις υπόχωρους του πίνακα Α. Τα διανύσματα του U που αντιστοιχούν στις μη μηδενικές ιδιάζουσες τιμές αποτελούν βάση για το χώρο των στηλών του Α, ενώ τα υπόλοιπα είναι βάση για τον αριστερό μηδενόχωρο του Α. Τα διανύσματα του V που αντιστοιχούν στις μη μηδενικές ιδιάζουσες τιμές αποτελούν βάση για το χώρο στων γραμμών του Α, ενώ τα υπόλοιπα είναι βάση για το μηδενόχωρο του Α.

3.1. SVD++ : Modified Singular Value Decomposition

It is based on the theoretical basis of regular SVD with the addition of an important feature. In reality a user will leave some implicit feedback information, such as historical data. These could be rating or browsing data or even the rating operation which in a way reflects the degree of a user's preference for each latent factor. Therefore, the SVD++ model introduces the implicit feedback information based on SVD. To accomplish that it adds factor vector (y_j) for each item in order to describe the characteristics of the item, regardless of whether it has been evaluated. This impacts the user's factor matrix improving the bias. Thus, the predictive rating of the SVD++ model is: (Zhengzheng Xian, 2017)

$$\widetilde{r}_{ui} = \mu + b_u + b_i + q_i^T \cdot \left(p_u + |R(u)|^{-1/2} \sum_{j \in R(u)} y_j \right)$$

For optimal P and Q we could calculate the regularized (for λ being the regularization parameter) squared error with the following formula:

$$\min_{P,Q} \sum_{r_{ui} \in R} \left[r_{ui} - \mu - b_u - b_i - q_i^T \cdot \left(p_u + |R(u)|^{-1/2} \sum_{j \in R(u)} y_j \right) \right]^2$$
$$+ \lambda \left(b_u^2 + b_i^2 + \|p_u\|^2 + \|q_i\|^2 \right) \Bigg],$$

3.2. Implementation of the Singular Value Decomposition with Python code

The Python language offers a special function for calculating the SVD of a table of values. The svd () function is included in the scipy library (Brownlee, 2018).

In the following example, a 2x2 array is calculated and printed, which results from the reduction of the dimensions of array A:

$$A = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 1 & 2 \end{bmatrix}$$

```
from numpy import array
from scipy.linalg import svd
# matrix
A = array([[2,-1,-1], [-1, 1, 2]])
print(A)
# SVD
U, s, VT = svd(A)
print(U)
print(s)
print(VT)
```

After executing the Python code, the following results were obtained:

```
[[ 2 -1 -1]
 [-1  1  2]]
[[-0.70710678  0.70710678]
 [ 0.70710678  0.70710678]]
[3.31662479 1.        ]
[[-6.39602149e-01  4.26401433e-01  6.39602149e-01]
 [ 7.07106781e-01 -2.77555756e-17  7.07106781e-01]
 [-3.01511345e-01 -9.04534034e-01  3.01511345e-01]]
```

3.3. Solution of the eigenvalue array

$$A=\begin{bmatrix} 2 & -1 & -1 \\ -1 & 1 & 2 \end{bmatrix}$$

A^T arises after I make the rows, columns, so:

$$A^T=\begin{bmatrix} 2 & -1 \\ -1 & 1 \\ -1 & 2 \end{bmatrix}$$

I find A^TA:

$$A^TA=\begin{bmatrix}2 & -1\\ -1 & 1\\ -1 & -2\end{bmatrix} \qquad \begin{bmatrix}2 & -1 & -1\\ -1 & 1 & 2\end{bmatrix}$$

$$A^TA=\begin{bmatrix}2\cdot 2+(-1)\cdot(-1) & 2\cdot(-1)+(-1)\cdot 1 & 2\cdot(-1)+(-1)\cdot 2\\ (-1)\cdot 2+1\cdot(-1) & (-1)\cdot(-1)+1\cdot 1 & (-1)\cdot(-1)+1\cdot 2\\ (-1)\cdot 2+2\cdot(-1) & (-1)\cdot(-1)+2\cdot 1 & (-1)\cdot(-1)+2\cdot 2\end{bmatrix}$$

$$A^TA=\begin{bmatrix}5 & -3 & -4\\ -3 & 2 & 3\\ -4 & 3 & 5\end{bmatrix}$$

We have to diagonalize το A^TA, so:

Det(A^TA-λ_I)=0

$$\begin{vmatrix}5-\lambda & -3 & -4\\ -3 & 2-\lambda & 3\\ -4 & 3 & 5-\lambda\end{vmatrix}=0$$

$$(5\text{-}\lambda)\begin{vmatrix}2-\lambda & 3\\ 3 & 5-\lambda\end{vmatrix}-(-3)\begin{vmatrix}-3 & 3\\ -4 & 5-\lambda\end{vmatrix}-4\begin{vmatrix}-3 & 2-\lambda\\ -4 & 3\end{vmatrix}=0$$

(5-λ)·[(2-λ)·(5-λ)-9]+3·[3·(λ-5)+12]-4[-9+4(2-2λ)]=0

(5-λ)·(10-2λ-5λ+λ^2-5)+3·(3λ-15+12)-4·(-1-4λ)=0

(5-λ)·(1-7λ+λ^2)+3(3λ-3)-4(-1-4λ)=0

(5-λ)·(λ^2-7λ+1)+9·(λ-1)+4·(1+4λ)=0

5λ^2-35λ+5-λ^3+7λ^2-λ+5λ-9+4+16λ=0

-λ^3+12λ^2-11λ=0

-λ·(λ^2-12λ+11)=0

λ(λ^2-11λ-λ+11)=0

λ[λ(λ-11)-(λ-11)]=0

λ[(λ-11)(λ-1)]=0

λ_1=11 λ_2=1 λ_3=0

I find eigenvectors of A^TA :

For λ_1=11 :

$$\begin{bmatrix} 5-11 & -3 & -4 \\ -3 & 2-11 & 3 \\ -4 & 3 & 5-11 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}=\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{bmatrix} -6 & -3 & -4 \\ -3 & -9 & 3 \\ -4 & 3 & -6 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}=\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{matrix} -6x-3y-4z=0 \\ -3x-9+3z=0 \\ -4x+3y-6z=0 \end{matrix}\Bigg]$$

From online solver, I get:

-6x-3y-4z=0

-15y+10z=0

0=0

15y=10z

3y=2z

Y=$\frac{2}{3}z$

-6x-3·$\frac{2}{3}z$-4z=0

-6x-2z-4z=0

X=-z

$$V_I=\begin{pmatrix} x \\ y \\ 7 \end{pmatrix}=7\begin{pmatrix} -1 \\ \frac{2}{3} \\ 1 \end{pmatrix}$$

I normalize by dividing it by the norm:

$$\|v_1\|=\sqrt{1+\frac{4}{9}+1}=\sqrt{\frac{22}{3}}$$

$$x_1=\frac{1}{\sqrt{22}}\begin{bmatrix} -3 & 2 & 3 \end{bmatrix}$$

Similarly for λ=1:

$$\begin{bmatrix} 5-1 & -3 & -4 \\ -3 & 2-1 & 3 \\ -4 & 3 & 5-1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 4 & -3 & -4 \\ 3 & 1 & -3 \\ -4 & 3 & 4 \end{bmatrix}\begin{bmatrix} x \\ y \\ 7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{matrix} 3x - 3y - 47 = 0 \\ -3x + y + 37 = 0 \\ -4x + 3y + 47 = 0 \end{matrix}\Bigg]$$

$$\begin{matrix} 4x - 3y - 4z = 0 \\ -5y = 0 \\ 0 = 0 \end{matrix}\Bigg]$$

y=0

x=z

$V_2=\begin{pmatrix} x \\ y \\ z \end{pmatrix}=\begin{pmatrix} x \\ 0 \\ x \end{pmatrix}=x\cdot\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$ * * I divide by the norm $||v_2||=\sqrt{1+1}=\sqrt{2}$

Where $x_2=\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$

For λ=0:

$$\begin{bmatrix} 5 & -3 & -4 \\ -3 & 2 & 3 \\ -4 & 3 & 5 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{matrix} 5x - 3y - 4z = 0 \\ -3x + 2y + 3z = 0 \\ -4x + 3y + 5z = 0 \end{matrix}\Bigg] \qquad \begin{matrix} 5x - 3y - 4z = 0 \\ y + 3z = 0 \\ 0 = 0 \end{matrix}\Bigg]$$ 5x-3·(-3z)-4z=0 x=-z και y=-3z

$V_3=\begin{pmatrix} x \\ y \\ z \end{pmatrix}=\begin{pmatrix} x \\ 3x \\ -x \end{pmatrix}=x\cdot\begin{pmatrix} 1 \\ 3 \\ -1 \end{pmatrix}$

I normalize by dividing it by the norm:

$||V_3||=\sqrt{11}$, άρα $\chi_3=\frac{1}{\sqrt{11}}\begin{pmatrix} 1 \\ 3 \\ -1 \end{pmatrix}$

The singular values of A are found in its eigenvalues A_A^T.

$\sigma_1=\sqrt{\lambda_1}=\sqrt{11}$, $\sigma_2=\sqrt{\lambda_2}=\sqrt{1}$, $\sigma_3=\sqrt{\lambda_3}=0$

I will not use σ_3 since it is zero.

$$S=\begin{bmatrix} \sqrt{11} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$V_1=\frac{1}{\sigma_1}\quad AV_1=\frac{1}{\sqrt{11}}\begin{bmatrix}2 & -1 & -1\\ -1 & 1 & 1\end{bmatrix}\qquad \frac{1}{\sqrt{22}}\begin{bmatrix}-3\\ 2\\ 3\end{bmatrix}$

$V_1=\frac{1}{11\sqrt{2}}\begin{bmatrix}6-2-3\\ 3+2+6\end{bmatrix}=\frac{1}{11\sqrt{2}}\begin{bmatrix}-11\\ 11\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}-1\\ 1\end{bmatrix}$

$V_2=\frac{1}{\sigma_2}\cdot A\cdot y_2$

$V_2=\frac{1}{\sqrt{2}}\begin{bmatrix}2 & -1 & -1\\ -1 & 1 & 2\end{bmatrix}\begin{bmatrix}1\\ 0\\ 1\end{bmatrix}$

$V_2=\frac{1}{\sqrt{2}}\begin{bmatrix}2-1\\ -1+2\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}1\\ 1\end{bmatrix}$

$V=[x_1 \quad x_2 \quad x_3]$

$V=\begin{bmatrix}\frac{-3}{\sqrt{22}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{11}}\\ \frac{2}{\sqrt{22}} & 0 & \frac{3}{\sqrt{11}}\\ \frac{3}{\sqrt{22}} & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{11}}\end{bmatrix}$

$V=\begin{bmatrix}\frac{-3}{\sqrt{22}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{11}}\\ \frac{2}{\sqrt{22}} & 0 & \frac{3}{\sqrt{1}}\\ \frac{3}{\sqrt{22}} & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{11}}\end{bmatrix}$

$V^T=\begin{bmatrix}\frac{-3}{\sqrt{22}} & \frac{2}{\sqrt{22}} & \frac{3}{\sqrt{22}}\\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{11}} & \frac{3}{\sqrt{11}} & \frac{-1}{\sqrt{11}}\end{bmatrix}$

$A=U\cdot S\ V^T$

$A=\frac{1}{\sqrt{2}}\begin{bmatrix}-1 & 1\\ 1 & 1\end{bmatrix}\begin{bmatrix}\sqrt{11} & 0 & 0\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\frac{-3}{\sqrt{22}} & \frac{2}{\sqrt{22}} & \frac{3}{\sqrt{22}}\\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{11}} & \frac{3}{\sqrt{11}} & \frac{-1}{\sqrt{11}}\end{bmatrix}$

3.3.1. The steps of the method

Singular Value Decomposition - SVD (Τζιρίτας, 2020), (Wikipedia, 2014) can be made for any array with dimensions m x n, but also n x n (square). Let A be an array of dimensions m x n. Then, it is possible to make a factorization of the form:

$A = U * S * V^T$

Array U is rectangular and has dimensions m x m, while array V is rectangular with dimensions n x n. Because the arrays U and V are rectangular, for them it will hold that $U^T = U^{-1}$, so, $U * U^T = U^T * U = I$ and respectively $V^T = V^{-1}$, άρα $V * V^T = V^T * V = I$.

Array S is a diagonal table and its values are non-negative and are usually arranged in descending order. Its values are called "eigenvalues" in array A.

To calculate the arrays U and V, we need to calculate the eigenvalues and eigenvectors of the arrays resulting from the products $A^T * A$ και $A * A^T$. Array U will contain the eigenvectors of the table $A * A^T$ and array V will include the eigenvectors of the array $A^T * A$. Finally, array S will contain the positive square roots of its array values $A^T * A$.

The following is an example of array analysis at Singular Value Decomposition.

Let's define the array:

$$A = \begin{bmatrix} 8 & 10 \\ 12 & 14 \\ 16 & 18 \end{bmatrix}$$

A SVD analysis gives us:

$$U = \begin{bmatrix} -0.38873885 & 0.82596334 & 0.40824829 \\ -0.56013329 & 0.13994297 & -0.81649658 \\ -0.73152772 & -0.5460774 & 0.40824829 \end{bmatrix}$$

$$S = [32.91877338 \quad 0.59528093]$$

$$V^T = \begin{bmatrix} -0.65421495 & -0.75630866 \\ -0.75630866 & 0.65421495 \end{bmatrix}$$

3.3.2. Application study of the method

In this chapter, we will study an application of SVD resolution, specifically image compression. First, we open an image file[2] resolution 720 x 1280 pixels and load it into the computer memory at the greyscale scale as a two-dimensional table. Thus, in this case, our table has dimensions of 720 rows by 1280 columns. Next, we apply the SVD

[2] Source of the image file: https://pixabay.com/photos/dome-skyline-cityscape-urban-5622133/

resolution to it and thus we get the result of the panel U, dimensions 720x720, the one-dimensional table S, 720 elements and the table V, dimensions 1280 x 1280. To be able to return to the original image, ie the original table A, we need to multiply these three tables. It is necessary to convert the one-dimensional S to a diagonal table, and thus acquire dimensions 720x720. Then, we recreate the original image, selecting smaller dimensions each time, ie selecting sub-tables of the above tables and calculating the product. The following images (Figure 1 - Figure 4) show the effect of compressing the image as we select a different number of dimensions to include each time. Of course, the final panel always has dimensions of 720 x 1280, ie the same dimensions as the original image, but it has zero values in the places of the missing dimensions. Thus, we compress the original image. We notice a decrease in image quality as we project it compressed.

Figure 4. The original image in all its dimensions

Figure 5. The image with 360 dimensions

Figure 6. The image with 100 dimensions

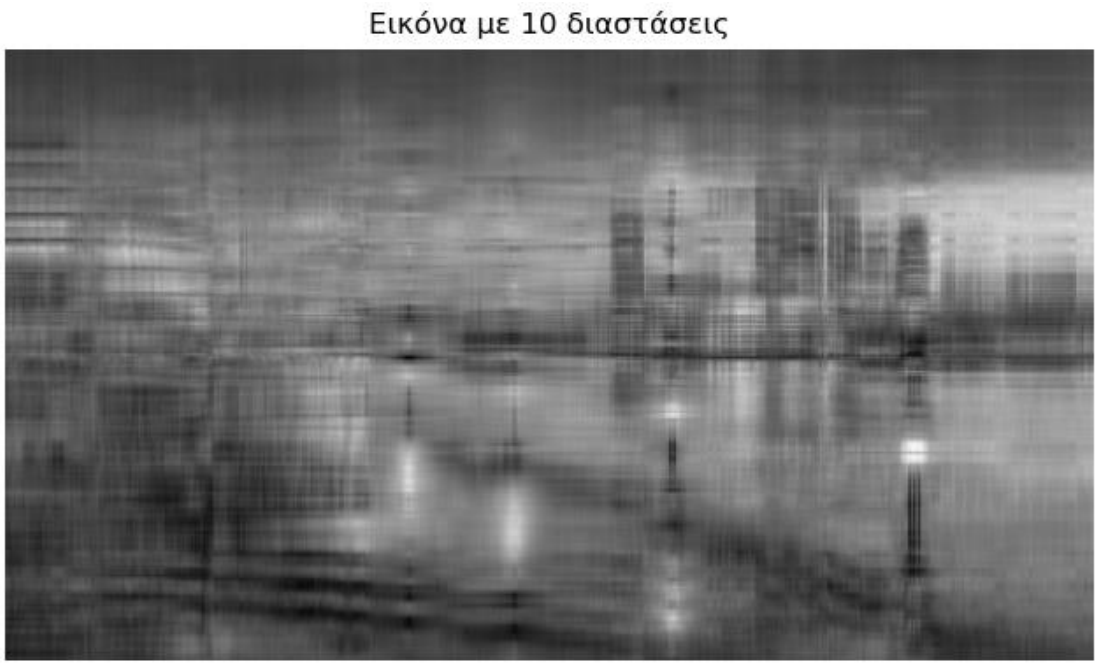


Figure 7. The image with 10 dimensions

We notice that even if we use 50% of the dimensions, the image quality does not decrease significantly, while the image remains usable even with the use of only 100 dimensions. Of course, in the 10 dimensions, the image is now distorted enough that it cannot be used.

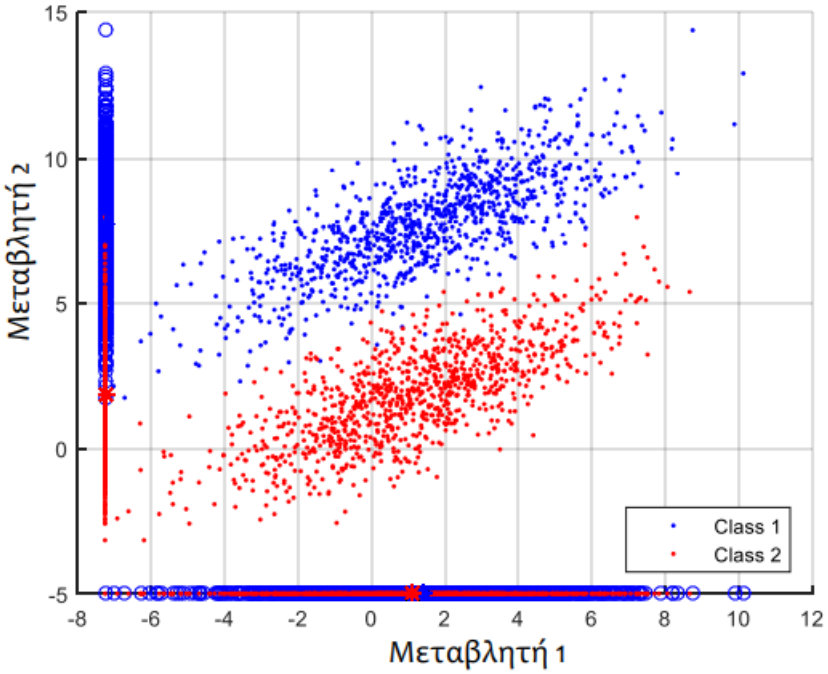
Through this concise study of Singular Value Decomposition Analysis, we can conclude that it is a simple and very useful way to factorize a table and reduce the information contained in it, thus achieving compression in the data. In this way we ensure faster processing of large data sets, without having a significant reduction in the information we can extract from them.

4. Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a method of transforming data belonging to categories (classes), with the aim of (a) better separation of classes and at the same time (b) reducing the dimensionality of data. (Wikipedia, 2021)

The central idea of the LDA is to transform the data in such a way as to maximize the distance between the classes (ie the classes to be remote) and at the same time to minimize the dispersion within the classes (ie the data of each class to be gathered around the their average value). Often the data of two or more categories are not easily linearly separable, and may consist of many variables, from which it may not be easy to find the ones that best separate the categories. The purpose of the LDA is to achieve good separation between categories (classes), while reducing dimensions (variables that make up the data). (Aly A. Farag, 2008)

In the example of the figure, the attempt to reduce features, using only class 1 or only class 2, leads to overlap between classes.

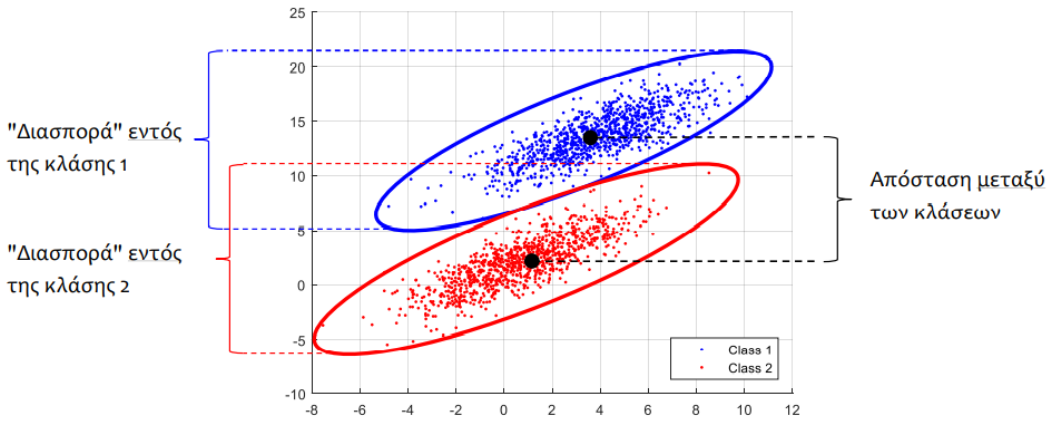


A criterion for good separation between classes (Fisher criterion) is:

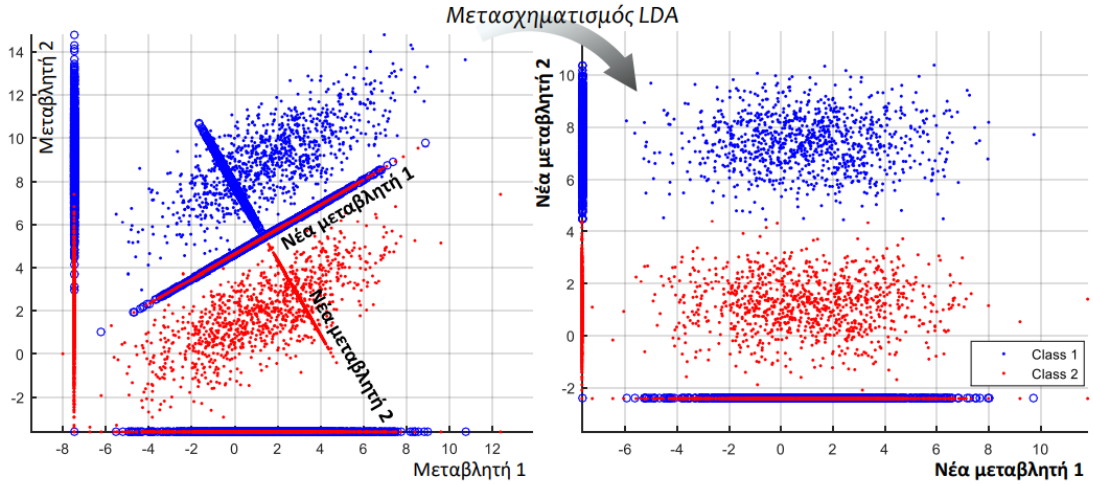
• The distance between the classes should be maximized (ie the mean values of the classes should be distant from each other), and at the same time the dispersion within

each class should be minimized (ie each class should be concentrated around its average value).

Specifically, the Fisher criterion is the ratio of the above two quantities, which when maximized we consider the classes to be better separated.



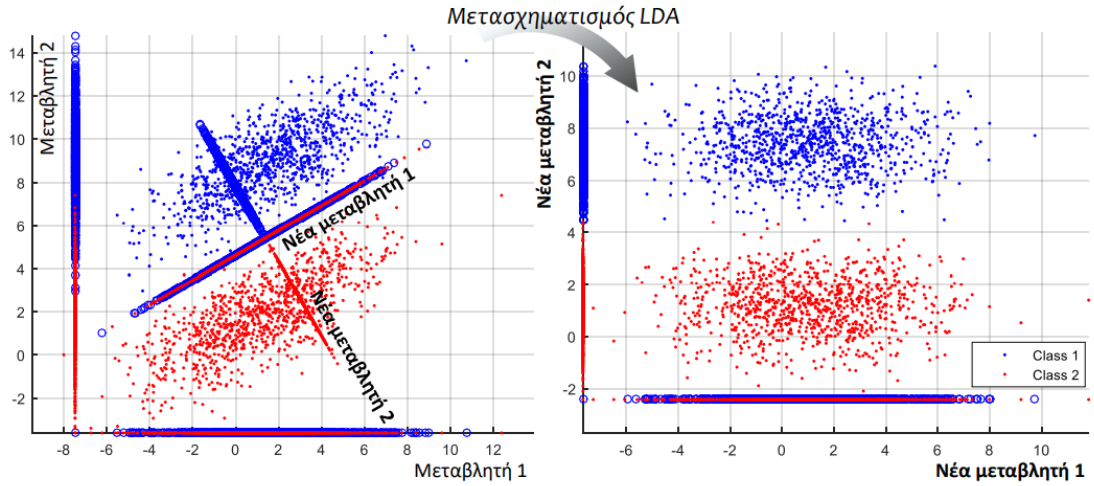
According to the LDA, in order to increase the separation between classes (so that the overlap between them is as small as possible), an appropriate data display address must be found to maximize the Fisher criterion (ie the ratio of the distance between classes to the total dispersion within each class).



LDA Results

By displaying the data in the appropriate address, the ratio of the distance between the classes to the scatter within the classes becomes maximum. Specifically, we

observe that, after the LDA transformation, on the axis of the New Variable 2 the class separation is optimal.



Mathematical process of the LDA

Let 2 classes X1 and X2 be in arrays M*N_k

Where (M=variables, N_k=observations).

The Fisher criterion for class separation is:

$$J = \frac{S_b}{S_w} = \frac{\Delta dispersion\ between\ classes}{Dispersion\ within\ classes}$$

where $S_b = (\mu_1 - \mu_2)^* ((\mu_1 - \mu_2)^T$,

$S_w = S_1 + S_2 = (X_1 - \mu_1)^* ((X_1 - \mu_1)^T + (X_2 - \mu_2)^* ((X_2 - \mu_2)^T +$

where μ_1, μ_2 the average values of the 2 classes and S_1, S_2 the dispersion arrays of the 2 classes.

The purpose of the LDA is to transform data to maximize Fisher J criterion, which occurs when its derivative is reset. After the transformation through table V, it turns out that:

$$\mathrm{J(V)} = \frac{V^T S_b V}{V^T S_w V}$$

The zeroing of the derivative of J (V) leads to the equation of eigenvalues, which we solve with respect to V:

SV=λ*V where S=$S_W^{-1}S_b$ and λ=J(V)

Straight Solution: The maximum eigenvalue λ *, ie the maximum value of the Fisher J criterion, as well as the corresponding eigenvector v * that defines the projection address for optimal class separation, can also be found with a straightforward solution of the equation $Sw\ -1SbV\ =\ \lambda V$, as follows: (Fisher, 2020)

λ*=$((\mu_2\ -\ \mu_1)^T S_W^{-1}(\mu_2\ -\ \mu_1)$ and V=$\frac{v^*}{\sqrt{v^{*T}v^*}}$

The classes after the transformation are:

$Y_1 = V^T X_1, Y_2 = V^T X_2$

5. LDA method

The Linear Discriminant Analysis (LDA) method is a method of dimensional reduction that transforms data belonging to categories (classes). In addition to reducing the size of the original data, the method aims to better separate classes by maximizing the distance between them and minimizing the distance between members of the same class (Καλατζής, 2017).

In order for the method to be successfully implemented, it must (Καλατζής, 2017), (Raschka, 2015):

- The data belong to categories (classes) which are known.
- The values of the data should be numerical and continuous.
- Each variable of the data set follows the normal distribution..
- The variables are statistically independent.

5.1.1. Steps of the method

The process of applying the LDA method is briefly described by the following steps:

a. We calculate the spreadsheet within a class and the spreadsheet between classes.
b. We calculate the eigenvectors and the corresponding eigenvalues of the scatter tables.
c. We classify the eigenvalues and select the larger K.
d. We create a new array that contains the eigenvectors that correspond to the K eigenvalues.
e. We transform the original data by calculating the product of this table with the table of original data.

5.1.2. Application study of the method

Then we will refer in more detail to the above steps (Raschka, 2015), through the study of an application of the method to the Seeds data set (Charytanowicz, et al., 2012) which was also used to study the application of the PCA method. Comments are categorized into three categories (classes).

First, we load the data set and then (Raschka, 2015):

1. We apply normalization to our data. During normalization, subtract the mean value of the corresponding variable from each observation and divide by the standard deviation of the variable.
2. For each class, we calculate the vector m containing the mean of each variable.

$$\boldsymbol{m}_i = \frac{1}{n_i} \sum_{x \in D_i}^{c} \boldsymbol{x}_m$$

Where x is an observation and n is the total number of observations. So we create three vectors with average values.

3. Calculate the spreadsheet within a class $\boldsymbol{S}_W$:

$$\boldsymbol{S}_W = \sum_{i=1}^{c} \boldsymbol{S}_i$$

This array is calculated by summing the individual dispersion of each **i** class:

$$\boldsymbol{S}_i = \sum_{x \in D_i}^{c} (\boldsymbol{x} - \boldsymbol{m}_i)(\boldsymbol{x} - \boldsymbol{m}_i)^T$$

4. Calculate the table of dispersion between classes:

$$\boldsymbol{S}_B = \sum_{i=1}^{c} N_i (\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^T$$

Where m is the mean of all observations in total.

5. We calculate the eigenvectors and eigenvalues for

$$\boldsymbol{S}_w^{-1} \boldsymbol{S}_B$$

so that new linear discriminants emerge.

Having calculated the eigenvectors, we sort them in descending order. The classification is shown in the figure below.

```
326.5819031132867
138.0809896766574
4.857169213671259e-12
2.4014771008651565e-12
1.6399965163856718e-13
1.6399965163856718e-13
2.5841578028465118e-14
```

Figure 1: The calculated eigenvalues

The eigenvectors that have the highest eigenvalues, also contain the largest percentage of information (variability) of the data. We calculate this percentage:

```
[0.7028362027198064,
 0.2971637972801878,
 1.0453103290662306e-14,
 2.9159385767992e-16,
 2.9159385767992e-16,
 5.561360381782612e-17,
 -5.168213640745263e-15]
```

Figure 2: The percentage of variability that each eigenvector contains

We observe that the first two eigenvectors contain almost 100% of the useful information for our data. Therefore, we choose the first two.

6. We create the table W, which contains the eigenvectors that correspond to the 2 eigenvalues we have chosen.

7. Finally, we display our data in the new dimensions by calculating the product of the table of original data with array W:

$$X' = XW$$

The next image shows the projection of our data in the 2 new dimensions that were created.

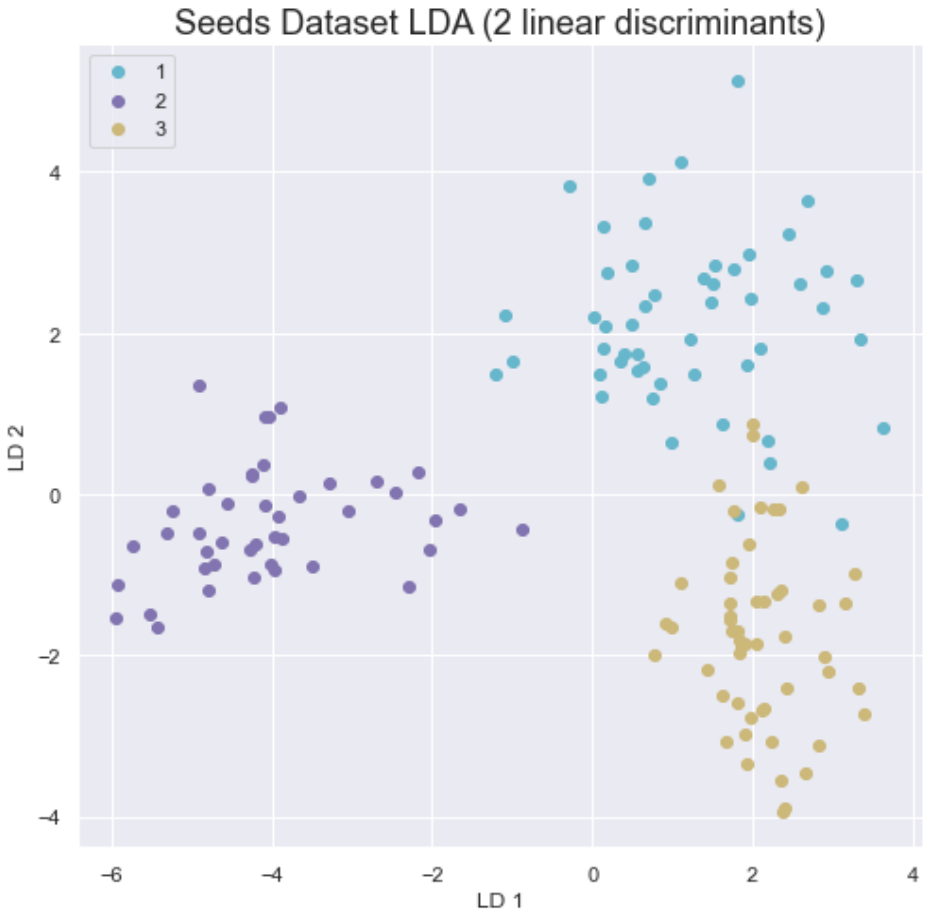


Figure 8. Data visualization in two new dimensions

Conclusion: We notice that class 2 is indeed more separate from the rest, so the algorithm achieves its goal. Classes 1 and 3 are not completely separated. However, compared to the separation of classes immediately after the application of the PCA method, we observe that here the separation is better. The PCA method aims to maximize the retention of data information in the new dimensions, while the LDA method also has the additional goal of maximizing the distance between classes.

6. Logistic Regression

Logistic Regression is a statistical model used to solve the problem of classification into two classes, the positive class and the negative class. It is used when, for example, we want to find the probability that an event will happen or not. The problem is finding a line with the appropriate coefficients (weights) that separates the data set into two classes. The reasoning of the method is based on the concept of odds, which is the ratio of the complementary probabilities of an event. Defining as p the probability of an event occurring, such as a student passing an exam, then (Raschka, 2015):

$$odds\ (p) = \frac{p}{(1-p)}$$

The logarithm of this ratio defines the logit function:

$$\text{logit}\ (p) = \log \frac{p}{(1-p)}$$

The logit function accepts as input values defined in the interval [0-1] and returns values that belong to the set of real numbers. Through this function, we can express the linear correlation between the values of the characteristics (variables) of each observation and the odds:

$$\text{logit}\ (p(y = 1 \mid \boldsymbol{x})) = w_0 x_0 + w_1 x_1 + \cdots + w_m x_m = \sum_{i=0}^{m} w_i x_i = \boldsymbol{w}^T \boldsymbol{x}$$

The probability $p(y = 1 \mid \boldsymbol{x})$ is the conditional probability that an observation belongs to the positive class, given the values of its characteristics. The weights $w_0, w_1 .. w_m$ are the coefficients of the line we are looking for.

The inverse function of the logit function is defined as:

$$f(z) = \frac{1}{1 + e^{-z}}$$

Where z is the linear combination of the weights and the values of the observation characteristics. This function is called a logistic function or "sigmoid" function. The following figure shows the graph of this function.

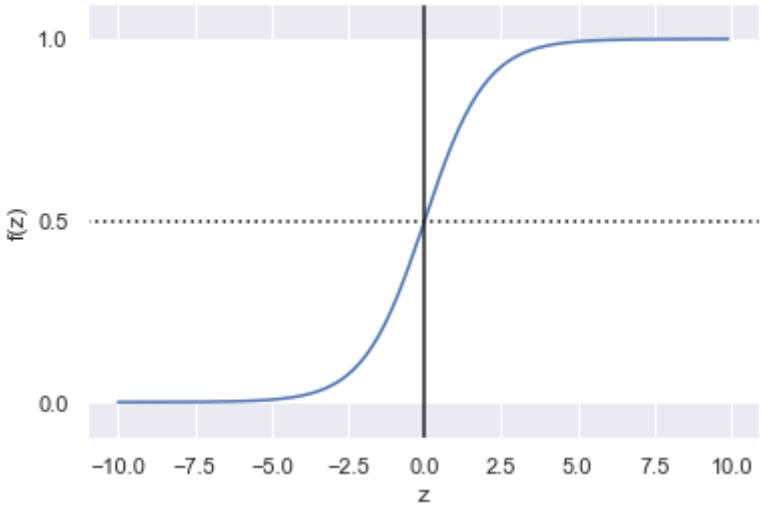


Figure 9. The "sigmoid" function

We observe that as the values of z increase, the value of the function tends to 1, while as the values of z decrease, the value of the function tends to 0. Thus, this function receives as input a value of a real number (z) and corresponds to a value within the interval [0, 1].

The output of the function can be thought of as the probability that the particular observation belongs to the positive class or not. If the value of this probability is> 0.5, then the observation belongs to the positive class.

Thus, finally we can consider as a result of the model that when the value of the observation z is positive, then it belongs to the positive class and when the value is negative, the observation belongs to the negative class.

Bibliography

Abreu, N. G. C. F. M. d., 2011. *Análise do perfil do cliente Recheio e desenvolvimento de um sistema promocional,* Λισαβόνα: ISCTE-IUL.

Aly A. Farag, S. Y. E., 2008. *A Tutorial on Data Reduction - Linear Discriminant Analysis.* [Ηλεκτρονικό]
Available at: https://www.lsv.uni-saarland.de/fileadmin/teaching/dsp/ss15/DSP2016/matdid437773.pdf

Bro, R. & Smilde, K. A., 2014. Principal component analysis. *Royal Society of Chemistry*.

Brownlee, J., 2018. *How to Calculate the SVD from Scratch with Python.* [Ηλεκτρονικό]
Available at: https://machinelearningmastery.com/singular-value-decomposition-for-machine-learning/

Charytanowicz, M. και συν., 2012. *UCI Machine Learning Repository - Seeds Data Set.* [Ηλεκτρονικό]
Available at: http://archive.ics.uci.edu/ml/datasets/seeds
[Πρόσβαση Φεβρουάριος 2021].

DataCadamia, 2021. *Machine Learning - (Univariate|Simple) Logistic regression.* [Ηλεκτρονικό]
Available at: https://datacadamia.com/data_mining/simple_logistic_regression
[Πρόσβαση 2 March 2021].

Fisher, K., 2020. *Kernel Fisher discriminant analysis, Wikipedia.* [Ηλεκτρονικό]
Available at: https://en.wikipedia.org/wiki/Kernel_Fisher_discriminant_analysis
[Πρόσβαση 2021].

Jaadi, Z., 2020. *A step by step explanation of principal component analysis.* [Ηλεκτρονικό]
Available at: https://builtin.com/data-science/step-step-explanation-principal-component-analysis
[Πρόσβαση Φεβρουάριος 2021].

Kalman, D., 1996. A Singularly Valuable Decomposition: The SVD of a Matrix. *THE COLLEGE MATHEMATICS JOURNAL,* 27(1).

Kodinariya, T. & Makwana, P., 2013. Review on Determining of Cluster in K-means Clustering. *International Journal of Advance Research in Computer Science and Management Studies*, pp. 90-95.

Longden, K., 2020. *What is Website Traffic?.* [Ηλεκτρονικό]
Available at: https://activeinternetmarketing.co.uk/what-are-the-different-types-of-website-traffic/

Nasreen, S., 2014. *A Survey Of Feature Selection And Feature Extraction Techniques In Machine Learning*. s.l., Science and Information (SAI).

Raschka, S., 2015. *Python Machine Learning*. Birmingham: Packt Publishing Ltd.

Russell, B., 2011. *A simple principal component analysis example*. [Ηλεκτρονικό] Available at: https://www.cs.toronto.edu/~rgrosse/courses/csc311_f20/tutorials/tut08/tut08_notes.pdf [Πρόσβαση Φεβρουάριος 2021].

Sakar, C., Polat, S., Katircioglu, M. & Kastro, Y., 2019. Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks. *Neural Computing and Applications*, p. 6893–6908.

Santosa, F. & Symes, W. W., 1986. Linear inversion of band-limited reflection seismograms. *SIAM Journal on Scientific and Statistical Computing,* p. 1307–1330.

Sharma, A., 2020. *Principal Component Analysis (PCA) in Python*. [Ηλεκτρονικό] Available at: https://www.datacamp.com/community/tutorials/principal-component-analysis-in-python

Techopedia, 2021 . *Decision Table (DETAB)*. [Ηλεκτρονικό] Available at: https://www.techopedia.com/definition/18829/decision-table-detab [Πρόσβαση 2 March 2021].

Wang, Z., 2019. *PCA and SVD explained with numpy*. [Ηλεκτρονικό] Available at: https://towardsdatascience.com/pca-and-svd-explained-with-numpy-5d13b0d2a4d8

Wikipedia, 2014. *Ανάλυση πίνακα σε ιδιάζουσες τιμές*. [Ηλεκτρονικό] Available at: https://el.wikipedia.org/wiki/Ανάλυση_πίνακα_σε_ιδιάζουσες_τιμές

Wikipedia, 2021. *Linear discriminant analysis*. [Ηλεκτρονικό] Available at: https://en.wikipedia.org/wiki/Linear_discriminant_analysis [Πρόσβαση 3 March 2021].

Zhengzheng Xian, Q. L. G. L. L. L., 2017. *New Collaborative filtering Algorithms based on SVD++and differential privacy,* s.l.: s.n.

Καλατζής, Ι., 2017. *Ανάλυση Γραμμικής Διάκρισης, Τμήμα Μηχανικών Βιοϊατρικής, Πανεπιστήμιο Δυτικής Αττικής*. [Ηλεκτρονικό] Available at: https://medisp.bme.uniwa.gr/eclass/modules/document/file.php/MTMBIT101/ΥΛΙΚΟ Ι. ΚΑΛΑΤΖΗ (DESCRIPTIVE STATISTICS, HYPOTHESIS TESTING, CLUSTERING, PCA, LDA)/5. Ανάλυση Γραμμικής Διάκρισης (LDA)/LDA (Ι.

Kalatzis 2017).pdf
[Πρόσβαση Φεβρουάριος 2021].

Παπαδάκη, Μ., 2012. *Μηχανές διανυσματικής υποστήριξης (SVMs) και εφαρμογές σε πραγματικά σεισμολογικά δεδομένα,* Αθήνα: Εθνικό Μετσόβειο Πολυτεχνείο / Διπλωματική εργασία.

Παπαδουράκης, Γ. & Μαριάς, Κ., 2017. *Η μέθοδος PCA – Ανάλυση Κύριων Συνιστωσών. Τμήμα Μηχανικών Πληροφορικής, ΤΕΙ Κρήτης*. [Ηλεκτρονικό] Available at: https://eclass.teicrete.gr/modules/document/file.php/TP223/Θεωρία 2017/7 PCA.pdf
[Πρόσβαση Φεβρουάριος 2021].

Πετρίδης, Δ., 2015. *Κεφάλαιο 3: Λογιστική Παλινδρόμηση.* [Ηλεκτρονικό] Available at: https://repository.kallipos.gr/bitstream/11419/2128/1/04_chapter03.pdf
[Πρόσβαση 2 March 2021].

Τζιρίτας, Γ., 2020. *Γραμμική Άλγεβρα. Ανάλυση πίνακα σε ιδιάζουσες τιμές. Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης*. [Ηλεκτρονικό] Available at: https://www.csd.uoc.gr/~hy119/tziritas/SVD.pdf
[Πρόσβαση Φεβρουάριος 2021].