Dimensionality Reduction

Feature Extraction

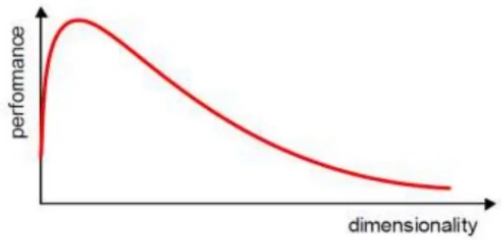
Principal Component Analysis (PCA)

B9DA108

Alexander Victor

Curse of Dimensionality

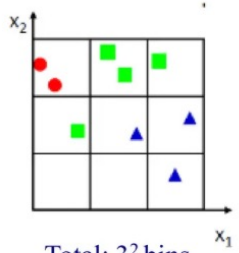
- Increasing the number of features will not always improve classification accuracy.



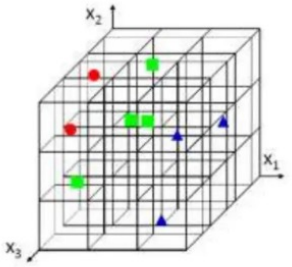
- In practice, the inclusion of more features might actually lead to worse performance.

Total: 3^1 bins

k=3 bins per feature



Total: 3^2 bins



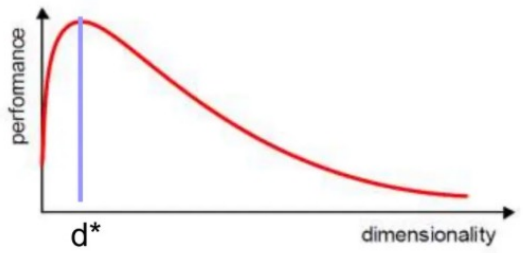
Total: 3^3 bins

- The number of training examples required increases exponentially with dimensionality **D** (i.e., $k^{\mathbf{D}}$).

k: number of bins per feature

Dimensionality Reduction

- What is the objective?
  - Choose an optimum set of features d* of lower dimensionality to improve classification accuracy.



- Different methods can be used to reduce dimensionality:
  - **Feature extraction**
  - **Feature selection**

Dimensionality Reduction (cont'd)

Feature extraction: computes a new set of features from the original features through some transformation f() .

Feature selection: chooses a subset of the original features.

f() could be linear or non-linear

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_D \end{bmatrix} \xrightarrow{f(\mathbf{x})} \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_K \end{bmatrix}$$

K<<D

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_D \end{bmatrix} \rightarrow \mathbf{y} = \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ \cdot \\ \cdot \\ x_{i_K} \end{bmatrix}$$

K<<D

4

Feature Extraction

- Linear transformations are particularly attractive because they are simpler to compute and analytically tractable.

- Given $x \in R^D$, find an K x D matrix T such that:

$$y = Tx \in R^K \text{ where } K<<D$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ . \\ . \\ x_D \end{bmatrix} \xrightarrow{T,\ f(\mathbf{x})} \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_K \end{bmatrix}$$

This is a projection transformation from D dimensions to K dimensions.

Each new feature y_i is a linear combination of the original features x_i

Feature Extraction (cont'd)

- From a mathematical point of view, finding an optimum mapping $\mathbf{y}=f(\mathbf{x})$ can be formulated as an optimization problem (i.e., minimize or maximize an **objective** criterion).

- Commonly used objective criteria:
  - Minimize Information Loss: projection in the lower-dimensional space preserves as much **information** in the data as possible.
  - Maximize Discriminatory Information: projection in the lower-dimensional space increases **class separability**.

Feature Extraction (cont'd)

- Popular linear feature extraction methods:
  - Principal Components Analysis (PCA): Seeks a projection that **minimizes information loss**.
  - Linear Discriminant Analysis (LDA): Seeks a projection that **maximizes discriminatory information.**

- Many other methods:
  - Making features as independent as possible (Independent Component Analysis).
  - Retaining interesting directions (Projection Pursuit).
  - Embedding to lower dimensional manifolds (Isomap, Locally Linear Embedding).

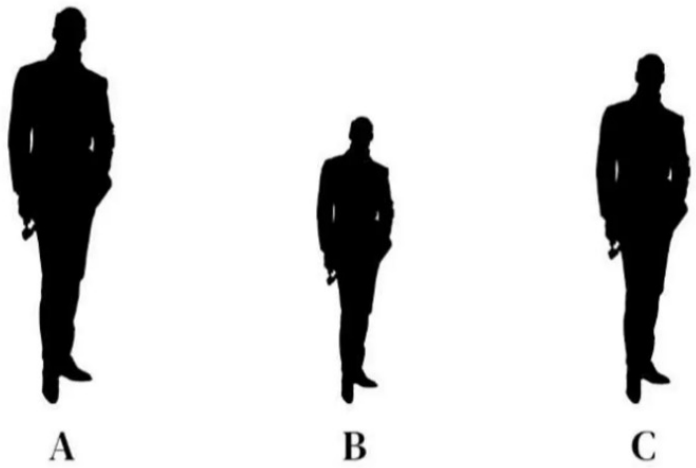
7

Principal Component Analysis (PCA)

Feature Extraction

Intuition behind PCA

Find the tallest person.



Person	Height
A	185
B	145
C	160

I can by seeing person A is the tallest

Intuition behind PCA

Person	Height
A	173
B	172
C	174

Find the tallest person.

It’s tough when they are very similar in height.

What Is Principal Component Analysis?

- Principal component analysis (PCA) is a statistical technique to reduce the dimensionality of complex, high-volume datasets by extracting the principal components that contain the most information and rejecting noise or less important data while preserving all the crucial details.

Principal Component Analysis (PCA)

- PCA comes under the Unsupervised Machine Learning category
- Reducing the number of variables in a data collection while retaining as much information as feasible is the main goal of PCA.
- PCA can be mainly used for Dimensionality Reduction and also for important feature selection.
- Correlated features to Independent features

What Is Principal Component Analysis?

- PCA is a method to reduce the dimensionality of enormous data collections.
- This approach transforms an extensive collection of variables into a smaller group that retains nearly all the data in the larger set.
- Lowering the number of variables in a data set inevitably reduces its precision

What Is Principal Component Analysis?

- The purpose of dimension reduction is to forgo a certain level of precision for simplification.
- Smaller data collections are easier to investigate and visualize.
- This facilitates and accelerates the analysis of data points by machine learning (ML) algorithms.

What Is Principal Component Analysis?

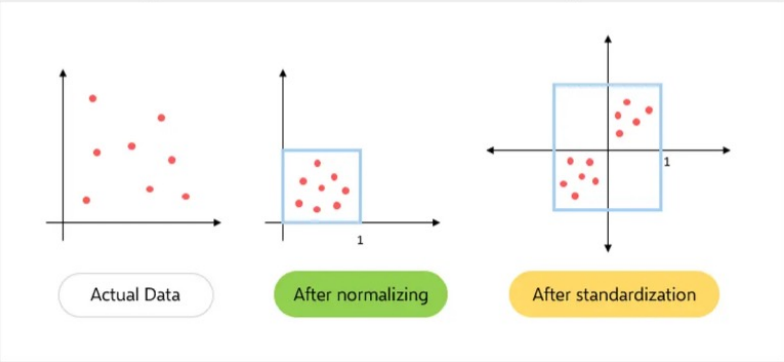
- PCA is designed to limit the total quantity of variables in a data set while maintaining as many details as possible.
- The **altered new features or PCA's results are known as principal components** (PCs) once PCA has been performed.
- The number of PCs is the same as or lesser than the number of original features in the dataset.

characteristics of principal components:

1. The principal component must correspond to the **linear arrangement** of the initial features.
2. The character of these components is **orthogonal**. This indicates there is no relationship within a pair of variables.
3. From 1 to n, the significance of each component diminishes.
    1. This indicates that the number one PC is the most important, while the number “n” PC is the least significant.

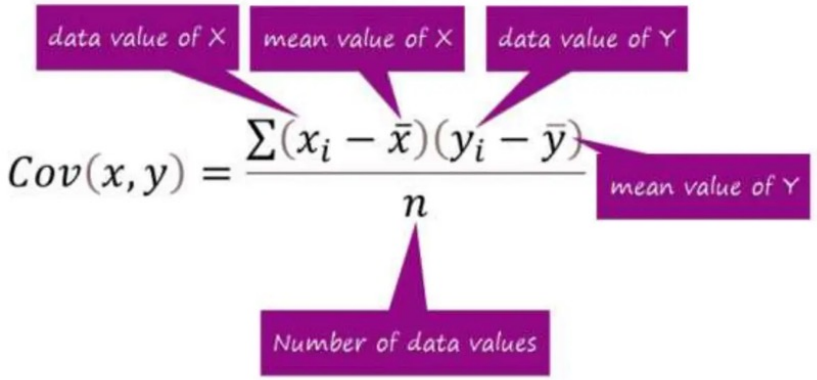
Basic Terminologies of PCA

- **Variance** – for calculating the variation of data distributed across dimensionality of graph
- **Covariance** – calculating dependencies and relationship between features
- **Standardizing data** – Scaling our dataset within a specific range for unbiased output



Basic Terminologies of PCA

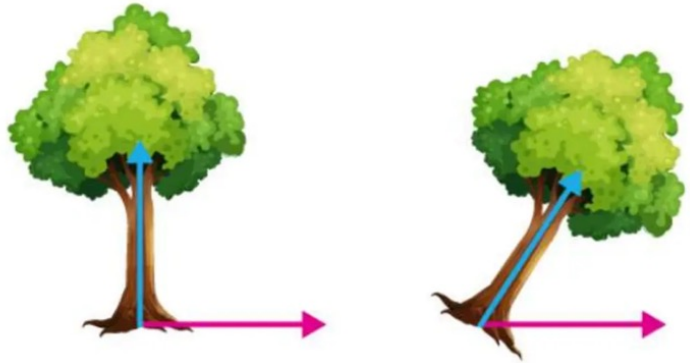
- **Covariance matrix** – Used for calculating interdependencies between the features or variables and also helps in reduce it to improve the performance



Basic Terminologies of PCA

- **EigenValues and EigenVectors** – Eigenvectors' purpose is to find out the largest variance that exists in the dataset to calculate Principal Component.
- Eigenvalue means the magnitude of the Eigenvector.
- Eigenvalue indicates variance in a particular direction and whereas eigenvector is expanding or contracting X-Y (2D) graph without altering the direction.

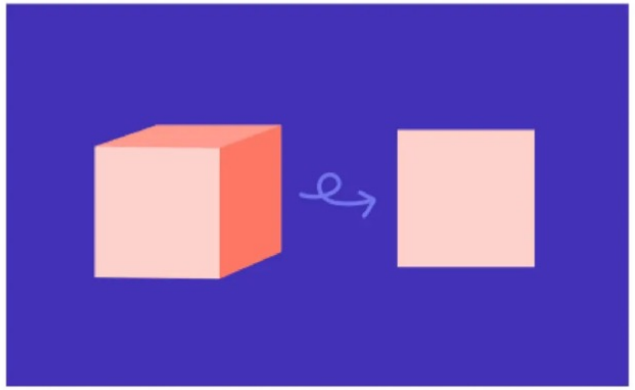
EigenValues and EigenVectors



- In this shear mapping, the **blue arrow changes direction** whereas the pink arrow does not.
- The pink arrow in this instance is an **eigenvector** because of its **constant orientation**.
- The length of this arrow is also unaltered, and its eigenvalue is 1.
- Technically, PC is a straight line that captures the maximum variance (information) of the data.
- PC shows direction and magnitude**. PC are perpendicular to each other.**

Basic Terminologies of PCA

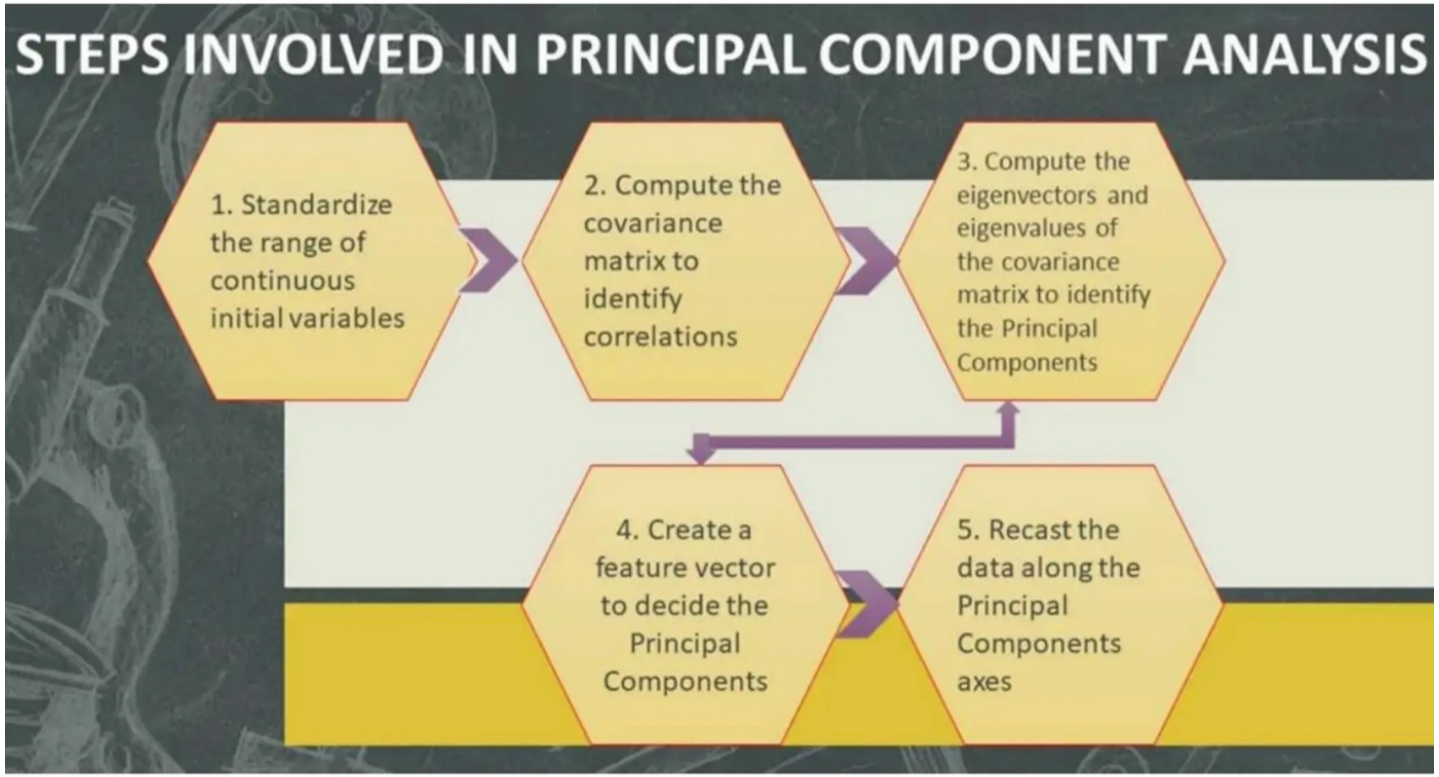
- **Dimensionality Reduction** – Transpose of original data and multiply it by transposing of the derived feature vector.
- Reducing the features without losing information.



How does PCA work?

- The steps involved for PCA are as follows-

1. Original Data
2. Normalize the original data (mean =0, variance =1)
3. Calculating covariance matrix
4. Calculating Eigen values, Eigen vectors, and normalized Eigenvectors
5. Calculating Principal Component (PC)
6. Plot the graph for orthogonality between PCs


STEPS INVOLVED IN PRINCIPAL COMPONENT ANALYSIS
1. Standardize the range of continuous initial variables
2. Compute the covariance matrix to identify correlations
3. Compute the eigenvectors and eigenvalues of the covariance matrix to identify the Principal Components
4. Create a feature vector to decide the Principal Components
5. Recast the data along the Principal Components axes

STEP 1: STANDARDIZATION

- The range of variables is calculated and standardized in this process to analyze the contribution of each variable equally.
- Calculating the initial variables will help you categorize the variables that are dominating the other variables of small ranges.
- This will help you attain biased results at the end of the analysis.

STEP 1: STANDARDIZATION

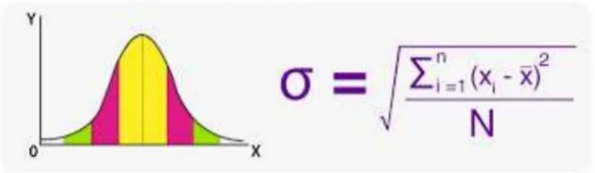
- To transform the variables of the same standard, you can follow the following formula.

$$Z = \frac{VALUE - MEAN}{STANDARD\ DEVIATION}$$

- Where,

$$MEAN = \frac{Sum\ of\ the\ terms}{Total\ number\ of\ terms}$$

Standard Deviation Formula



X= value in a data set N= number of values in the data set

STEP 1: STANDARDIZATION

- **Example:**

LARGE SIZE APPLES	ROTTEN APPLES	DAMAGED APPLES	SMALL APPLES
F1	F2	F3	F4
1	5	3	1
4	2	6	3
1	4	3	2
4	4	1	1
5	5	2	3

STEP 1: STANDARDIZATION

- Calculate the Mean and Standard Deviation for each feature and then, tabulate the same as follows.

	F1	F2	F3	F4
MEAN	3	4	3	2
STANDARD DEVIATION	1.87	1.223	1.87	1

STEP 1: STANDARDIZATION

- after the Standardization of each variable, the results are tabulated below.

F1	F2	F3	F4
-1.0695	0.8196	0	-1
0.5347	-1.6393	1.6042	1
-1.0695	0	0	0
0.5347	0	-1.0695	-1
1.0695	0.8196	-0.5347	1

STEP 2: COVARIANCE MATRIX COMPUTATION

- In this step, you will get to know how the variables of the given **data are varying with the mean value calculated**.
- Any interrelated variables can also be sorted out at the end of this step.
- To segregate the highly interrelated variables, you calculate the covariance matrix with the help of the given formula.

  - **Note: **A covariance matrix is a N x N symmetrical matrix that contains the covariances of all possible data sets.

STEP 2: COVARIANCE MATRIX COMPUTATION

- The covariance matrix of two-dimensional data is, given as follows:

$$\text{Covariance Matrix} = \begin{bmatrix} COV(X,X) & COV(X,Y) \\ COV(Y,X) & COV(Y,Y) \end{bmatrix}$$

$$\text{Covariance} = \frac{Sum\ (X-(Mean\ of\ X)(Y-(Mean\ of\ Y))}{Number\ of\ data\ points}$$

covariance matrix of 3-dimensional data

$$\text{Covariance of data} = \begin{bmatrix} cov(X,X) & cov(Y,X) & cov(Z,X) \\ cov(X,Y) & cov(Y,Y) & cov(Z,Y) \\ cov(X,Z) & cov(Y,Z) & cov(Z,Z) \end{bmatrix}$$

STEP 2: COVARIANCE MATRIX COMPUTATION

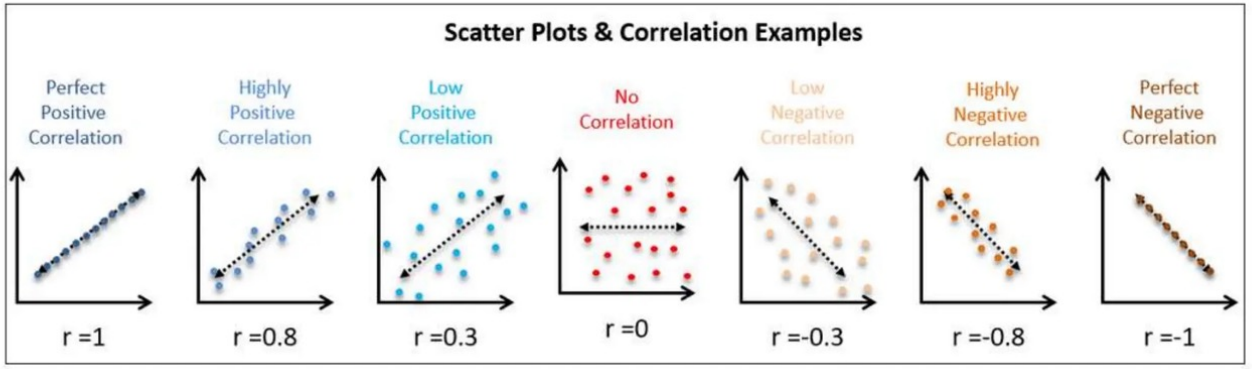
Insights of covariance matrix

- the covariance of a number with itself is its variance **(COV(X, X)=Var(X)),** the values at the top left and bottom right will have the variances of the same initial number.
- Covariance Matrix at the main diagonal will be symmetric concerning the fact that **covariance is commutative (COV(X, Y)=COV(Y, X)).**

STEP 2: COVARIANCE MATRIX COMPUTATION

Insights of covariance matrix

- If the **value of the Covariance Matrix is positive**, :**variables are correlated**. (If X increases, Y also increases and vice versa)
- If the value of the Covariance Matrix **is negative: variables are inversely correlated.** (If X increases, Y also decreases and vice versa).i
- End of this step, we will come to **know which pair of variables are correlated with each other,** so that you might categorize them much easier.



STEP 2: COVARIANCE MATRIX COMPUTATION

- Example: The formula to calculate the covariance matrix of the given example will be:

	F1	F2	F3	F4
F1	VAR(F1)	COV(F1,F2)	COV(F1,F3)	COV(F1,F4)
F2	COV(F2,F1)	VAR(F2)	COV(F2, F3)	COV(F2, F4)
F3	COV(F3,F1)	COV(F3,F2)	VAR(F3)	COV(F3,F4)
F4	COV(F4,F1)	COV(F4,F2)	COV(F4,F3)	VAR(F4)

STEP 2: COVARIANCE MATRIX COMPUTATION

- Since you have already standardized the features, you can consider **Mean = 0 and Standard Deviation=1** for each feature.

VAR(F1) = ((-1.0695-0)² + (0.5347-0)² + (-1.0695-0)² + (0.5347–0)² +(1.069–0)²)/5

- On solving the equation, you get, **VAR(F1) = 0.78**

COV(F1,F2) = ((-1.0695–0)(0.8196-0) + (0.5347–0)(-1.6393-0) + (-1.0695–0)* (0.0000-0) + (0.5347–0)(0.0000-0)+ (1.0695–0)(0.8196–0))/5

- On solving the equation, you get, **COV(F1,F2 = -0.8586)**

covariance matrix

- Similarly solving all the features, the covariance matrix will be,

	F1	F2	F3	F4
F1	0.78	-0.8586	-0.055	0.424
F2	-0.8586	0.78	-0.607	-0.326
F3	-0.055	-0.607	0.78	0.426
F4	0.424	-0.326	0.426	0.78

STEP 4: FEATURE VECTOR

- To determine the principal components of variables, we have to define **eigen value and eigen vectors**.
- Let A be any square matrix. A non-zero vector v is an eigenvector of A if

$$Av = \lambda v$$

  - for some number λ, called the corresponding eigenvalue.

STEP 4: FEATURE VECTOR

- Once you have computed the eigen vector components, define eigen values in descending order (for all variables) and now you will get a list of principal components.

- So, the eigen values represent the principal components and these components represent the direction of data.

STEP 4: FEATURE VECTOR

- This indicates that if the **line contains large variables of large variances**, then there are many data points on the line. Thus, there is **more information** on the line too.

- Finally, these principal components form a line of new axes for easier evaluation of data and also the differences between the observations can also be easily monitored.

STEP 4: FEATURE VECTOR

- **Example:**
- Let ν be a non-zero vector and λ a scalar.

As per the rule,

- $A\nu = \lambda\nu$, then **λ is called eigenvalue** associated with **eigenvector ν** of A.

STEP 4: FEATURE VECTOR

- **Example:**
- Upon substituting the values in **det(A- λI) = 0**, you will get the following matrix.

	F1	F2	F3	F4
F1	0.78 - λ	-0.8586	-0.055	0.424
F2	-0.8586	0.78 - λ	-0.607	-0.326
F3	-0.055	-0.607	0.78 - λ	0.426
F4	0.424	-0.326	0.426	0.78 - λ

STEP 4: FEATURE VECTOR

- When you solve the following the matrix by considering 0 on right-hand side, you can define eigen values as

$$\lambda = 2.11691 , 0.855413 , 0.481689 , 0.334007$$

- Then, substitute each eigen value in $(A-\lambda I)v=0$ equation and solve the same for different eigen vectors v1, v2, v3 and v4.

- For instance, For $\lambda = 2.11691$, solving the above equation using **Cramer's rule**, the values for the v vector are

v1 = 0.515514 v2 = -0.616625 v3 = 0.399314 v4 =0.441098

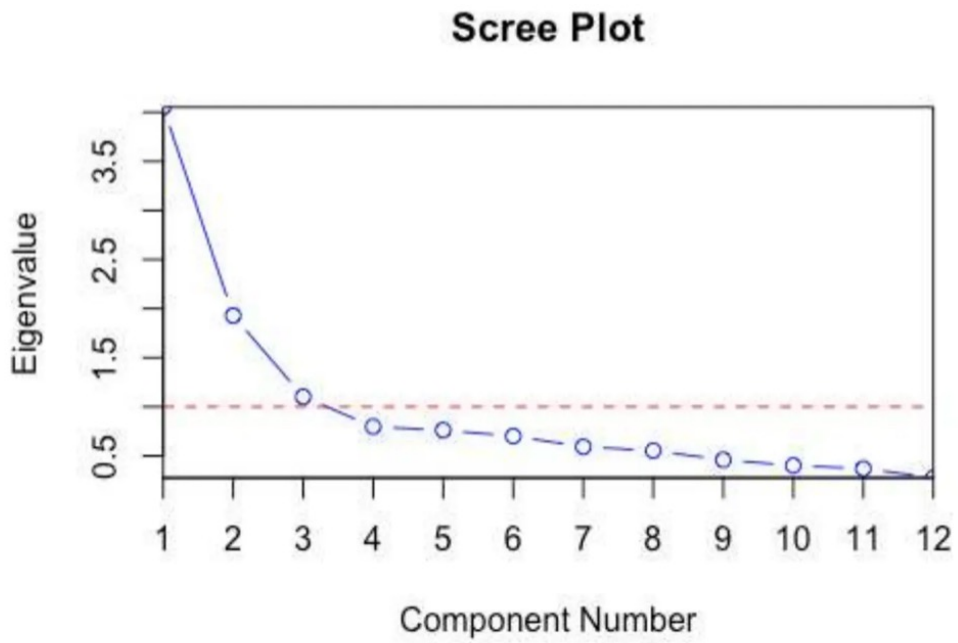
STEP 4: FEATURE VECTOR

- Follow the same process and you will form the following matrix by using the eigen vectors calculated as instructed.

E1	E2	E3	E4
0.515514	−0.623012	0.0349815	−0.587262
-0.616625	0.113105	0.452326	−0.634336
0.399314	0.744256	−0.280906	−0.455767
0.441098	0.212477	0.845736	0.212173

STEP 4: FEATURE VECTOR

- Sort the Eigenvalues in decreasing order.



STEP 4: FEATURE VECTOR

- calculate the sum of each Eigen column, arrange them in descending order and pick up the topmost Eigen values.
- These are your Principal components.

E1	E2
0.515514	–0.623012
-0.616625	0.113105
0.399314	0.744256
0.441098	0.212477

STEP 5: RECAST THE DATA ALONG THE PRINCIPAL COMPONENTS AXES

- Still now, apart from standardization, you haven't made any changes to the original data.
- You have just selected the Principal components and formed a feature vector.
- Yet, the initial data remains the same on their original axes.
- This step aims at the **reorientation of data from their original axes to the ones you have calculated from the Principal components.**
- This can be done by the following formula.

Final Data Set= Standardized Original Data Set * FeatureVector

Example:

- **Standardized Original Data Set =**

F1	F2	F3	F4
-1.0695	0.8196	0	-1
0.5347	-1.6393	1.6042	1
-1.0695	0	0	0
0.5347	0	-1.0695	-1
1.0695	0.8196	-0.5347	1

- FeatureVector =

E1	E2
0.515514	0.744256
0.441098	0.212477
0.399314	0.113105
-0.616625	–0.623012

Example:

- By solving the above equations, you will get the transformed data as follows.

LARGE SIZE APPLES	ROTTEN APPLES
0.4268066978	0.00116114000000012
-0.4234920968	-0.39182856
-0.551342223	-0.7959219
0.4652040128	0.89996329
0.0827279480000002	0.28653037

- Your large dataset is now compressed into a small dataset without any loss of data!

Example in Python

- **Step 1: Load the Iris Data Set**

```
import pandas as pd
import matplotlib.pyplot as plt
# Load dataset into Pandas DataFrame
df = pd.read_csv('iris.csv')
df.rename(columns = {'variety':'target'}, inplace = True)
df.head(10)
```

	sepal.length	sepal.width	petal.length	petal.width	target
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

Step 2: Standardize the Data

- Use **StandardScaler** to help you standardize the data set's features onto unit scale (mean = 0 and variance = 1), which is a requirement for the optimal performance of many machine learning algorithms.
- If you don't scale your data, it can have a negative effect on your algorithm.

```
from sklearn.preprocessing import StandardScaler

features = ['sepal.length', 'sepal.width', 'petal.length', 'petal.width']

# Separating out the features
x = df.loc[:, features].values

# Separating out the target
y = df.loc[:,['target']].values

# Standardizing the features
x = StandardScaler().fit_transform(x)
```

Step 2: Standardize the Data

- Use **StandardScaler** to help you standardize the data set's features onto unit scale (mean = 0 and variance = 1), which is a requirement for the optimal performance of many machine learning algorithms.
- If you don't scale your data, it can have a negative effect on your algorithm.

	sepal length	sepal width	petal length	petal width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Standardization →

	sepal length	sepal width	petal length	petal width
0	-0.900681	1.032057	-1.341272	-1.312977
1	-1.143017	-0.124958	-1.341272	-1.312977
2	-1.385353	0.337848	-1.398138	-1.312977
3	-1.506521	0.106445	-1.284407	-1.312977
4	-1.021849	1.263460	-1.341272	-1.312977

Step 3: PCA Projection to 2D

- The original data has four columns (sepal length, sepal width, petal length and petal width).
- In this section, the code projects the original data, which is four-dimensional, into two dimensions.

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)

principalComponents = pca.fit_transform(x)

principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1', 'principal component 2'])
```

	principal component 1	principal component 2
0	-2.264703	0.480027
1	-2.080961	-0.674134
2	-2.364229	-0.341908
3	-2.299384	-0.597395
4	-2.389842	0.646835

Step 3: PCA Projection to 2D

- Concatenating DataFrame along **axis = 1.**
- finalDf is the final **DataFrame** before plotting the data.

```
finalDf = pd.concat([principalDf, df[['target']]], axis = 1)
finalDf
```

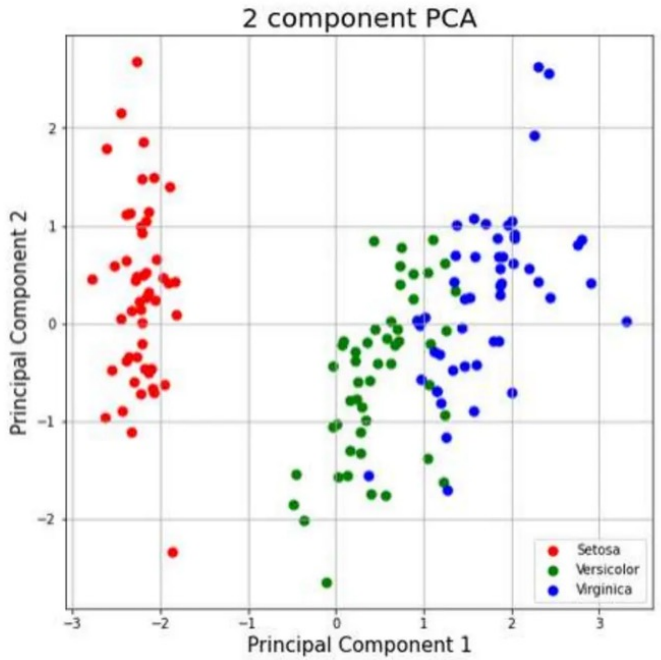
	principal component 1	principal component 2	target
0	-2.264703	0.480027	Setosa
1	-2.080961	-0.674134	Setosa
2	-2.364229	-0.341908	Setosa
3	-2.299384	-0.597395	Setosa
4	-2.389842	0.646835	Setosa

Step 4: Visualize 2D Projection

```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)

targets = ['Setosa', 'Versicolor', 'Virginica']
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['target'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
               , finalDf.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
```

Step 4: Visualize 2D Projection



Explained Variance

- The explained variance tells **you how much information (variance) can** be attributed to each of the principal components.
- By using the attribute **explained_variance_ratio_,** you can see that the first principal component contains 72.96 percent of the variance, and the second principal component contains 22.85 percent of the variance.
- Together, the two components contain 95.71 percent of the information.

```
pca.explained_variance_ratio_
```

```
array([0.72962445, 0.22850762])
```

Perform a Scree Plot of the Principal Components

- A scree plot is like a bar chart showing the size of each of the principal components.
- It helps us to visualize the percentage of variation captured by each of the principal components.
- To perform a scree plot you need to:
  - first of all, create a list of columns
  - then, list of PCs
  - finally, do the scree plot using plt

```
from sklearn.decomposition import PCA

pca = PCA(n_components=4)

principalComponents = pca.fit_transform(x)

principalDf = pd.DataFrame(data = principalComponents
             , columns = ['pc1', 'pc2','pc3','pc4'])
```

```
percent_variance = np.round(pca.explained_variance_ratio_* 100, decimals =2)
columns = ['PC1', 'PC2', 'PC3', 'PC4']
plt.bar(columns,percent_variance, tick_label=columns)
plt.ylabel('Percentate of Variance Explained')
plt.xlabel('Principal Component')
plt.title('PCA Scree Plot')
plt.show()
```

Scree Plot