Machine Learning 🖥

Unsupervised Learning

Principal Component Analysis (PCA)

By ⟹ PRINCE 👑 ❤️

https://github.com/pkvidyarthi/

Dimensionality Reduction Technique or Principal Component Analysis ⟶

- Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning.
- It is one of the popular tools that is used for exploratory data analysis and predictive modeling.
- It is a technique to draw strong patterns from the given dataset by reducing the variances.
- PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.
- **Some Real World applications of PCA are :**
    - Image Processing
    - Movie Recommendation System
    - Optimizing the power allocation in various communication channels.

PCA algorithm is based on some mathematical concepts such as :

- Variance and Covariance
- Eigenvalues and Eigen Factors

Some common terms used in PCA algorithm :

- **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- **Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed.
    - The correlation value ranges from -1 to +1. Here,
    - -1 : If variables are inversely proportional to each other, and
    - +1 : It indicates that variables are directly proportional to each other.
- **Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
- **Eigenvectors:** If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.
- **Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

Application of PCA ⟶

- Computer Vision
- Image Compression, etc
- It can also be used for finding hidden patterns if data has high dimensions. Some fields where PCA is used are Finance, data mining, Psychology, etc.

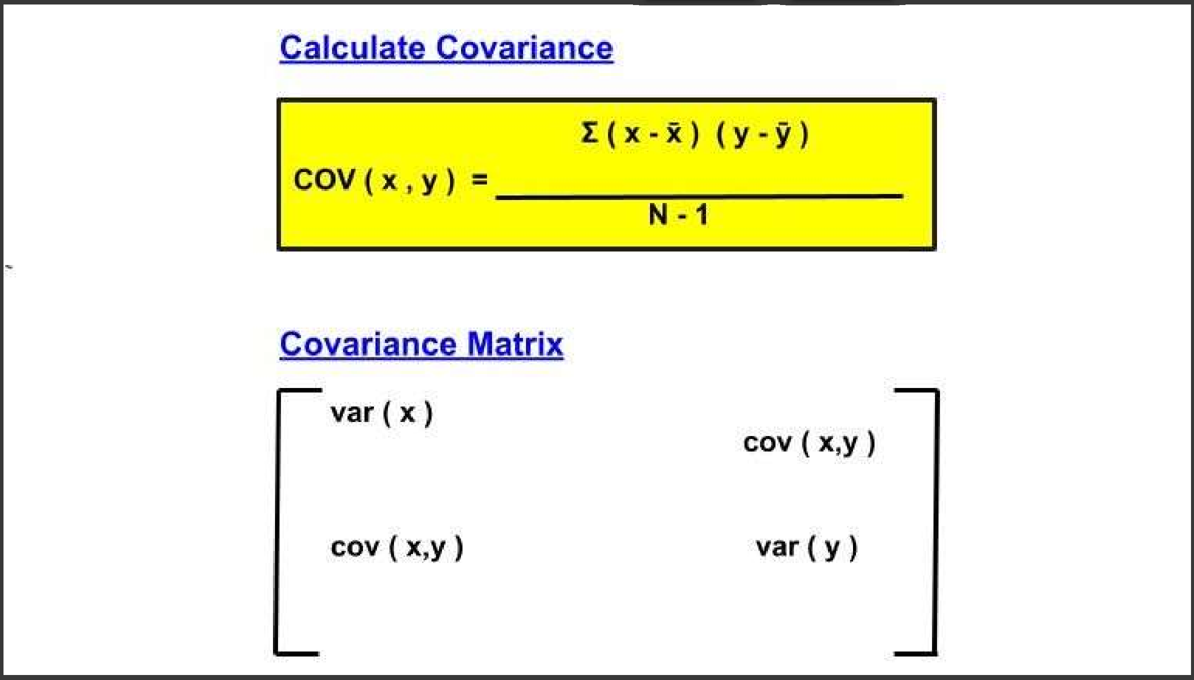
Steps for PCA algorithm

1. **Getting the dataset** ⟶ Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.
2. **Representing data into a structure** ⟶ Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.
3. **Standardizing the data** ⟶ In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance.

If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.

4. **Calculating the Covariance of Z** ⟶ To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.
5. **Calculating the Eigen Values and Eigen Vectors** ⟶ Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.
6. **Sorting the Eigen Vectors** ⟶ In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P*.
7. **Calculating the new features Or Principal Components** ⟶ Here we will calculate the new features. To do this, we will multiply the P* matrix to the Z. In the resultant matrix Z, *each observation is the linear combination of original features. Each column of the Z* matrix is independent of each other.
8. **Remove less or unimportant features from the Principal Components(new dataset)** ⟶ The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out.

Calculate Covariance :



Calculate Covariance

$$\boxed{\mathrm{COV}(x, y) = \frac{\Sigma(x - \bar{x})(y - \bar{y})}{N - 1}}$$

Covariance Matrix

$$\begin{bmatrix} \mathrm{var}(x) & \mathrm{cov}(x,y) \\ \mathrm{cov}(x,y) & \mathrm{var}(y) \end{bmatrix}$$

Eigenvalues in PCA :

- Eigenvectors are unit vectors with length or magnitude equal to 1. They are often referred to as right vectors, which simply means a column vector.
- Eigenvalues are coefficients applied to eigenvectors that give the vectors their length or magnitude.

  So, PCA is a method that:

  - Measures how each variable is associated with one another using a Covariance matrix.
  - Understands the directions of the spread of our data using Eigenvectors.
  - Brings out the relative importance of these directions using Eigenvalues.

Pipeline (Data Pipeline) :

- A machine learning pipeline is a way to control and automate the workflow it takes to produce a machine learning model. Machine learning pipelines consist of multiple sequential steps that do everything from data extraction and preprocessing to model training and deployment.
- Machine learning pipelines are iterative as every step is repeated to continuously improve the accuracy of the model and achieve a successfull algorithm.
- The main objective of having a proper pipeline for any ML model is to exercise control over it.
- A typical machine learning pipeline would consist of the following processes:
  - Data collection
  - Data cleaning
  - Feature extraction (labelling and dimensionality reduction)
  - Model validation
  - Visualisation

The goal for ML is simple: " Make faster and better predictions."

Python Implementation for Principal Component Analysis :

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Mounting Google Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=Tru
```

```
df = pd.read_csv("/content/drive/MyDrive/Dataset/Iris.csv")
```

```
df.shape
```

```
(150, 6)
```

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Id	150.0	75.500000	43.445368	1.0	38.25	75.50	112.75	150.0
SepalLengthCm	150.0	5.843333	0.828066	4.3	5.10	5.80	6.40	7.9
SepalWidthCm	150.0	3.054000	0.433594	2.0	2.80	3.00	3.30	4.4
PetalLengthCm	150.0	3.758667	1.764420	1.0	1.60	4.35	5.10	6.9
PetalWidthCm	150.0	1.198667	0.763161	0.1	0.30	1.30	1.80	2.5

Importing StandardScaler

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()     # Calculates Z-Score
df_std = df.drop(['Id', 'Species'], axis = 1)
df_std = ss.fit_transform(df_std)
```

Importing Principal Component Analysis (PCA)

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 3)
pca = pca.fit(df_std)
#pca
```

```
pca = pca.fit_transform(df_std)
pca
# Output => Set of feature vector (Eigen Values)
```

Importing RandomForestClassifier and Train Test Split

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

Pipeline

```
from sklearn.pipeline import Pipeline
pipe = Pipeline([('scaler', StandardScaler()), ('PCA', PCA(n_components = 2)), ('rf', RandomForestClassifier())])
```

```
x = df.drop(['Id','Species'], axis = 1)
y = df[['Species']]
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.2, random_state = 2)
```

```
pipe.fit(xtrain,ytrain)
```

```
Pipeline(steps=[('scaler', StandardScaler()), ('PCA', PCA(n_components=2)),
                ('rf', RandomForestClassifier())])
```

```
pipe_pred = pipe.predict(xtest)
pipe_pred
```

```
array(['Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica'], dtype=object)
```

Accuracy Score, Confusion Matrix, Classification Report

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
accuracy_score(ytest, pipe_pred)
```

```
0.9
```

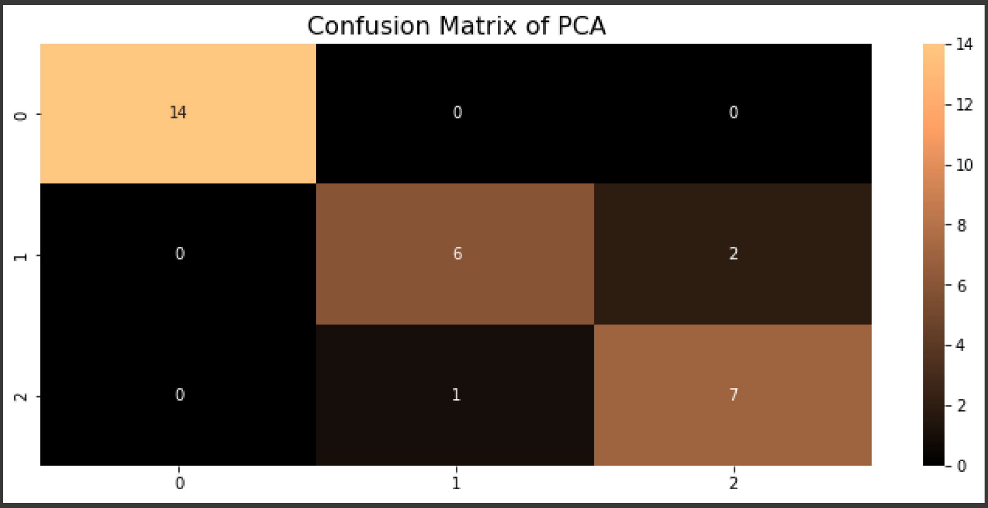
```
print(classification_report(ytest, pipe_pred))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       0.86      0.75      0.80         8
 Iris-virginica       0.78      0.88      0.82         8

       accuracy                           0.90        30
      macro avg       0.88      0.88      0.87        30
   weighted avg       0.90      0.90      0.90        30
```

```
cf = confusion_matrix(ytest, pipe_pred)
cf
```

```
array([[14,  0,  0],
       [ 0,  6,  2],
       [ 0,  1,  7]])
```

```
plt.figure(figsize =(12,5))
plt.title('Confusion Matrix of PCA', fontsize = 16)
sns.heatmap(cf, annot = True, fmt = 'g', cmap = 'copper')
plt.show()
```



✓ 0s completed at 3:22 PM