Principal Component Analysis

Nisha Akole, G V V Sharma*

Contents

***Abstract*—This manual provides a brief description on how to implement Principal Component Analysis from scratch and use it for dimensionality reduction of data.**

1. Objective

Our objective is to implement PCA and reduce dimensionality of data which gives better understanding of data visually.

2. Load Dataset

The dataset used for PCA is available at the following link. Download all the data file in the folder where you want to write code for PCA.

https://github.com/prabhatrai111/Commensal–Radar

```
import numpy as np
import scipy.io as sio
from sklearn.utils.extmath import randomized_svd

mat_contents = sio.loadmat('data_all.mat')
X_data = mat_contents['data_all']
```

*The authors are with the Department of Electrical Engineering, Indian Institute of Technology, Hyderabad 502285 India e-mail: gadepall@iith.ac.in.

3. About PCA

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entitles each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components. There are four main operations in the PCA:

1) Pre-processing
2) Co-variance Matrix
3) Eigen Vectors
4) Feature Vector and Plot

4. Pre-processing

Take the whole dataset ignoring the labels. Feature scaling is an important task in PCA because various scales of feature will affect and we may not get an optimized output. The functions StandardScaler or MinMaxScaler will standardize the features by making mean = 0 and variance = 1. These functions are imported from sklearn.preprocessing. StandardScaler uses formula such as

$$z = (X - \mu)/\sigma$$

where, μ =Mean of data X
σ =Standard Deviation of data X

X = np.matrix(X_data)
$\mu = X.mean(0)$
$\sigma = X.std(0)$
X_std= $(X - \mu)/\sigma$

X.mean(0) and X.std(0) will give columnwise mean and standard deviation of our data matrix.

5. Compute Covariance Matrix

Computing the covariance matrix will help us to understand the relationship between the variables. If variables are highly correlated, they contain redundant information. Covariance matrix will be symmetric with diagonal values as a variance of the

corresponding element. If variables are increasing or decreasing together, then its positive covariance.If one variable is increasing and other is decreasing, sign of covariance will be negative.

```
X_cov = np.cov(X_std)
```

6. Compute Eigenvectors

The diagonal values of covariance matrix are the eigenvalues of a covariance matrix where large eigenvalues correspond to large variance. Eigenvector of covariance matrix is the axes along with data has maximum variation. Eigenvalues and eigevectors can be calculated by taking Singular Value Decomposition(SVD) of a covariance matrix. Eigenvalues are arranged in decreasing order in sigma matrix and its corresponding eigenvector is arranged in U which is left eigenvector or eigenvector of np.matmul(X_cov,X_cov'), where X_cov' is transpose of X_cov.

```
U, Σ, VT = randomized_svd(X_cov,
    n_components=10 ,n_iter= 5, random_state
    =none)
```

n_components will consider first 10 largest eigenvalues. Hence, first 10 eigenvectors will be used for plotting the data and these eigenvectors are known as principal components which carries maximum information.

7. Principal Component and plot

To visualize the data either we can plot it in 2D or 3D. But sometime first 2 or 3 principal components are not sufficient to retain the maximum information from dataset. In such case, no of principal components can be decided by taking ratio of first K eigenvalues to sum of all the eigenvalues. If the ratio is closer to 0.95 then these K principal components are taken. Link for PCA without function is given below:

```
https://github.com/NishaAkole/AI-and-ML/blob/
    master/codes/pca/PCAwithoutFun.py
```

The above whole program can be combined except preprocessing and a single command(i.e using inbuilt function) can do all the

```
https://github.com/NishaAkole/AI-and-ML/blob/
    master/codes/pca/PCAFun.py
```

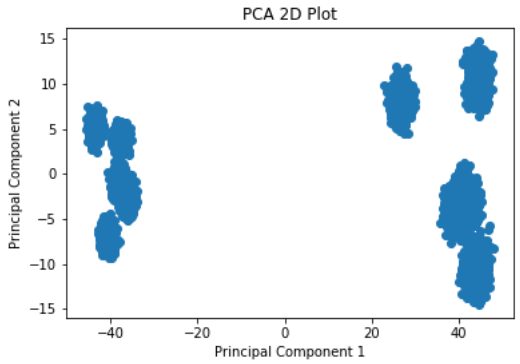

Fig. 1: PCA with Function 2D

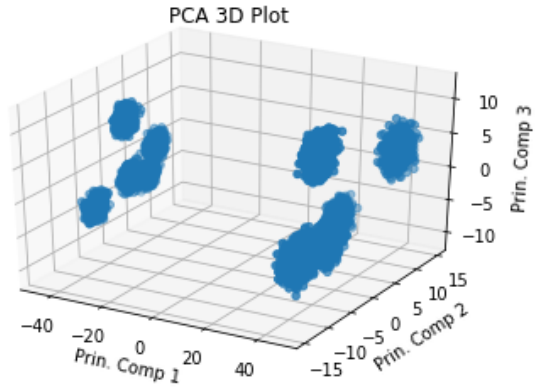

Fig. 2: PCA with Function 3D

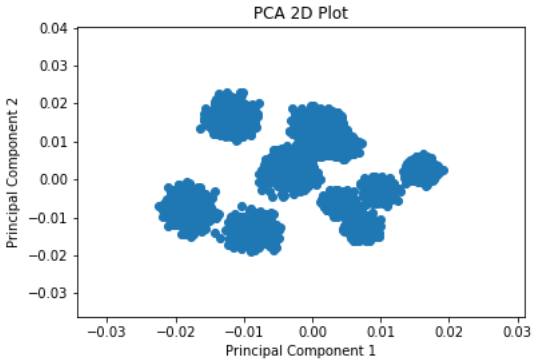

Fig. 3: PCA without Function 2D

8. Figures

Above plots help us to visualize the data. To get more idea on principal component and and amount of information it contain through plot below:

Clearly, only two or three prin. comp does not

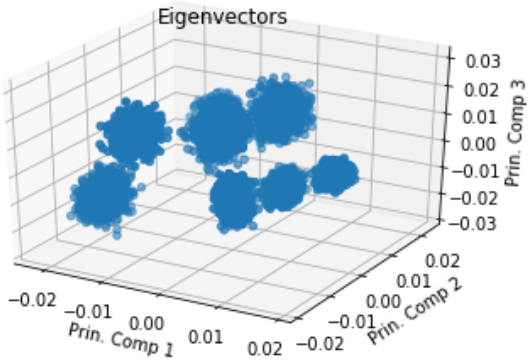

Fig. 4: PCA without Function 3D

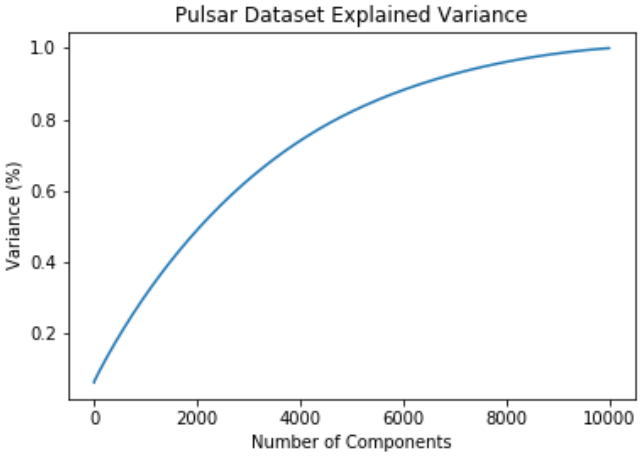

Fig. 5: Information vs No of Dimensions

contain much info. Hence, PCA is not a good approach for dimensionality reduction for this data.