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 into a set of values of linearly uncorrelated variables called principal components.

cov[PC1, PC2] = E[PC1PC2] − E[PC1]E[PC2]=0

Where, PC - 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

Feature scaling is an important task in PCA to get an optimized output. The functions StandardScaler or MinMaxScaler will perform mean normalization by making mean = 0 and variance = 1. Standard-Scaler Function :

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

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

5. Compute Covariance Matrix

$$X\_cov = \begin{bmatrix} var_{PC1} & cov_{PC1,PC2} \\ cov_{PC1,PC2} & var_{PC2} \end{bmatrix} = \begin{bmatrix} \sigma^2_{\text{PC1}} & \sigma_{PC1,PC2} \\ \sigma_{PC1,PC2} & \sigma^2_{\text{PC2}} \end{bmatrix}$$

If variables are highly correlated, they contain redundant information.

$$\mathbf{X\_cov}^{\intercal} = \text{X\_cov}$$

X_cov = np.cov(X_std)

6. Compute Eigenvectors

Eigenvalues and eigevectors can be calculated by Singular Value Decomposition(SVD) of a covariance matrix.

$$X\_cov = U\ \Sigma\ \mathbf{V}^{\intercal}$$

Eigenvalues are arranged in decreasing order in sigma matrix and its corresponding eigenvector is arranged in U.

$$\mathbf{\Sigma}^{\intercal} = \Sigma$$

```
U, Σ, V^T = randomized_svd(X_cov,
    n_components=2 ,n_iter= 5, random_state=
    none)
```

n_components will consider first 2 largest eigenvalues. Hence, first 2 eigenvectors will be used for plotting the data.

7. Principal Component and plot

To visualize the data, only 2 or 3 principal components are needed. To decide K principal components out of n to retain maximum information:

$$\frac{\sum_{i=1}^{k} FirstKDiagonaElements}{\sum AllDiagonalElements} >= 0.99$$

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

8. Figures

From fig. 5, we are not able to reduce the dimensionality below 8000. Hence, PCA is not a good approach for dimensionality reduction for this data.

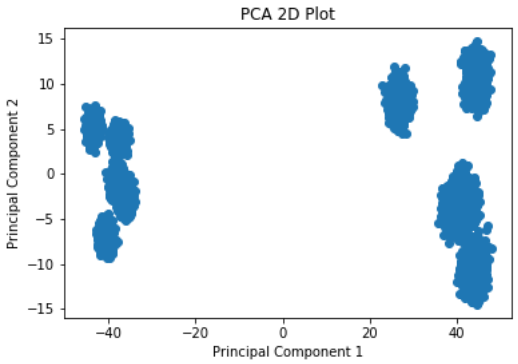

Fig. 1: PCA with Function 2D

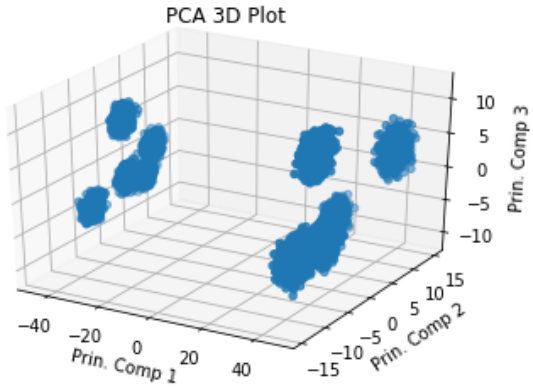

Fig. 2: PCA with Function 3D

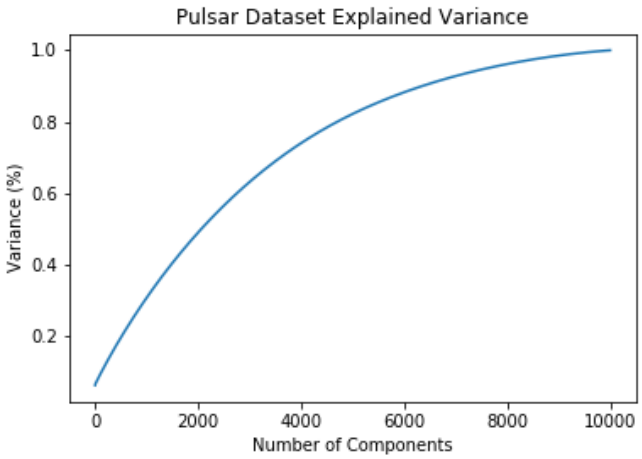

Fig. 3: Information vs No of Dimensions