Princial Component Analysis (PCA)

By SHILPA THOTA

1 Introduction

PCA is used for dimensionality reduction. To understand the concept of PCA we should understand **curse of dimensionality**.

Suppose we have multiple models M_1,M_2.. M_6 and the dataset has 500 features. For each model let us consider a subset of the features are provided which varies the accuracy of the model predicting the output as we try the different combinations. Sometimes we observe even though we increases number of features for the model the accuracy might be less but the model with less number of features might give better accuracy. This is called curse of Dimensionality. The Model M_1 is given 3 features and M_2 is given 6 features and M_3 with 15 features. we see as features increases the accuracy increases, but after this when the M_4 is given 50 features the accuracy starts falling down and M_5 was given 100 features then the accuracy even more reduced. This is because of overfitting the model from M_4 which shows the feature is less significant.

The Model performance degrade occurs as the number of features increases as the model has to evaluate based on all those features.

Two different ways to remove curse of dimensionality

- **Feature Selection** - We take important features and train the model
- **Dimensionality Reduction (PCA)** - Derive the new features from the existing features and then use those features to train the model.

2 Feature Selection Vs Feature Extraction

This technique is used in dimensionality reduction, which prevents the curse of dimensionality, improves the performance of the model, and helps visualize the data.

Feature selection is a process that helps us to select most important feature to better predict the output.

Covariance is the term that represent relationship between input and output.

$$Cov(X,Y) = \sum_{i=1}^{N} \frac{(x-\bar{x})(y-\bar{y})}{N-1}$$

If this is positive then Y increases with increase in X and if it is negative then Y decreases with increase in X. IF the covariance is 0 then there is no relationship between X and Y

If the covariance is very high, then we say that features are very important.

There is also one more technique which is Pearson Correlation.

$$PearsonCorrelation = \frac{cov(X,Y)}{\sigma_x \sigma_y}$$

This value ranges from -1 to +1.The more the value is closer to +1, the more positive correlation between X and Y. similarly, negative correlation if it is closer to -1.

In Feature Extraction, if there are multiple independent features that are very important and it is highly correlated, then we apply some transformation to extract the new feature

3 PCA Geometric Intuition

PCA is used for dimensionality reduction. For instance, we convert 2D to 1D. For doing this, let us project the datapoints on the x-axis. The spread indicates the variance. Now the y-axis feature variance or spread is getting lost as everything is made as 0 when we project the datapoints on the x-axis. In this process, the amount of information is lost. How to prevent that?

In PCA,we will apply a mathematical transformation which is Eigen decomposition matrix there will be a new line $X^{`}$ is created which is projected to the new axis which does have the spread captured for the y-axis.

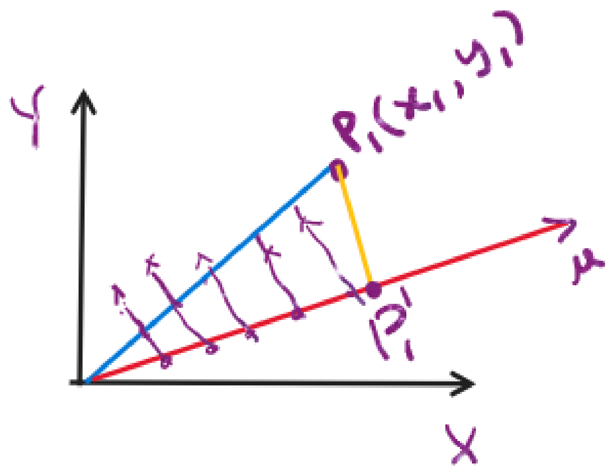

Figure 1: Projection of P_1 on unit vector

Projection is nothing but the shadow of the point in a vector in the other plane or line. We are creating the projection of the line in a different plane so the huge information is not lost.

The main goal of the PCA is to capture the principal component where the least amount of information is lost.

We use the best PC algorithm line and selects 2 best Principal line. A line is called best principal line if it has higher variance. and take it as PC1 and other line as PC2. In 3D then we have PC1, PC2 and PC3.

Variance of PC1 > *Variance of PC2* > *Variance of PC3*

4 Math Intuition behind PCA Algorithm

Let us consider one point $P_1(x_1, y_1)$ and when we do the projection the point becomes $P_1^{'}(x_1, y_1)$ and μ for the unit vector. The projection of P_1 should be projected on unit vector as $P_1^{'}$ Refer to Figure 1

This formula can be given as

$$Proj_{P_1}^{\mu} = \frac{P_1.\mu}{||\mu||}$$

Since the magnitude of vector μ is 1 we can write the above equation as

$$Proj_{P_1}^{\mu} = P_1.\mu$$

The value that is obtained is scalar value. Let us say for each value of P_1 the projected value is $P_1^{'}$ Similarly, for other points, we get $P_0^{'}$, $P_1^{'}$, $P_2^{'}$,... $P_n^{'}$ for n points that are projected on the unit vector. In order to compute the variance we can compute the equation

$$Variance = \sum_{i=1}^{n} \frac{(X_i - \bar{X})^2}{n}$$

The main aim is to find the best unit vector which has maximum variance. This is our **Cost Function**.

There is a technique called Eigen decomposition which is the formation of Eigen vectors and eigen values to find the best unit vector. For this, there are certain steps

- Covariance Matrix between features
- Eigen vectors and Eigen values will be found from the covariance matrix
- Eigen vector which has the maximum magnitude or the eigen value is high. This will capture maximum variance.

Eigen Vectors and Eigen Values can be found from the formula which is linear transformation of Covariance Matrix

$$Av = \lambda v$$

5 Eigen Vectors and Eigen Values

Suppose we have vector and if there is a linear transformation applied on to the vector, then the vector gives the λ multiplied by the vector. This λ has the eigen values.

Eigenvalues and eigenvectors are fundamental concepts in linear algebra that provide insight into the geometric properties of linear transformations.

Given a square matrix A, an eigenvector v is a non-zero vector that, when the matrix A is applied to it, only scales the vector by a constant factor, without changing its direction. This scaling factor is called the eigenvalue λ

$$Av = \lambda v$$

In simple terms,

Eigenvector: A vector whose direction remains unchanged by the transformation.

Eigenvalue: A scalar that indicates how much the eigenvector is stretched or compressed during the transformation.

5.1 Steps to find the Eigen vectors and eigen values

Step 1 is standarizing the data points which will be centered around the center.

Step 2 - We find the Covariance matrix and it is n by n matrix for instance let us say 3 by 3. The covariance matrix A is given by

$$\begin{bmatrix} Var(X) & Cov(X,Y) & Cov(X,Z) \\ Cov(Y,X) & Var(Y) & Cov(Y,Z) \\ Cov(Z,X) & Cov(Z,Y) & Var(Z) \end{bmatrix}$$

The covariance is given by

$$Cov(X,Y) = \sum_{i=1}^{N} \frac{(X_i - \bar{X})(Y_i - \bar{Y})}{N-1}$$

and the *Cov(X,X)* = *Var(X)*

Step - 3 is to find the Eigen values

If the covariance matrix obtained from the above equation is multipled by vector v which is transformation then it gives the λ value which is λ_1 which is PC1 component and λ_2 the PC2 component. If we have N features we get till λ_N

Now if we want to reduce the dimensions we can project the eigen values to the dimensions that we need. Like 3D can be projected to 2D which gives 2 features.

6 Python Implementation

```

 import matplotlib.pyplot as plt
 import seaborn as sns
 import numpy as np
 import pandas as pd
 %matplotlib inline

 # Load the dataset
 from sklearn.datasets import load_breast_cancer

 cancer_dataset = load_breast_cancer()
 cancer_dataset.keys()
print(cancer_dataset.DESCR)
 # we can see the dataset where the output is Malignant and Benign
df = pd.DataFrame(cancer_dataset['data'],columns=cancer_dataset['feature_names'])
df.head()

# We will be extracting the features which has maximum amount of information

### Standardization
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

scaler.fit(df)

```

```python
scaled_data = scaler.transform(df)

### Applying PCA algorithm
from sklearn.decomposition imoprt PCA

### Will extract 2 features
pca = PCA(n_components=2)
data_pca = pca.fit_transform(scaled_data)

data_pca

## To get variance for the 2 features selected
pca.explained_variance_

## Plot the graph
plt.figure(figsize=(8,6))
plt.scatter(data_pca[:,0],data_pca[:,1],c=cancer['target'],cmap='plasma')
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
```

Listing 1: PCA Implementaion