Principal Component Analysis

- Principal component analysis (PCA) is a popular dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

- It is primarily used to transform a high-dimensional dataset into a lower-dimensional representation while preserving the most important information.

- Because smaller data sets are easier to explore and visualize and make analyzing data much easier and faster for machine learning algorithms without extraneous variables to process.

Goal of PCA:

1. Identify Patterns in Data
2. Detect the correlation between variables

3. Reduce the dimension of a d-dimensional datasets by projecting it on a k-dimensional subspace where k<d

4. Find list of Principal axes

Covariance is a statistical measure that quantifies the degree to which two variables change together. In the context of features in a dataset, covariance between features reflects how much two features vary in relation to each other.

•Positive covariance: Indicates that as one feature increases, the other tends to increase as well.

•Negative covariance: Indicates that as one feature increases, the other tends to decrease.

Correlation provides a standardized measure of the strength and direction of the linear relationship between two features,

The main goal of PCA is to identify the directions (principal components) along which the data varies the most.

These principal components are orthogonal to each other, meaning they are uncorrelated.

The first principal component captures the largest amount of variation in the data, followed by the second principal component, and so on.

- PCA is an unsupervised pre-processing task that is carried out before applying any ML algorithm.

- PCA is based on “orthogonal linear transformation” which is a mathematical technique to project the attributes of a data set onto a new coordinate system.

- The attribute which describes the most variance is called the first principal component and is placed at the first coordinate.

•Similarly, the attribute which stands second in describing variance is called a second principal component and so on. In short, the complete dataset can be expressed in terms of principal components

•. Usually, more than 90% of the variance is explained by two/three principal components.

•Principal component analysis, or PCA, thus converts data from high dimensional space to low dimensional space by selecting the most important attributes that capture maximum information about the dataset.

Objectives of PCA

1. The new features are distinct i.e. the covariance between the new features (in case of PCA, they are the principal components) is 0.

2. The principal components are generated in order of the variability in the data that it captures. Hence, the first principal component should capture the maximum variability, the second one should capture the next highest variability etc.

3. The sum of the variance of the new features / the principal components should be equal to the sum of the variance of the original features.

Working of PCA

Step 1: Standardize the dataset.

Step 2: Calculate the covariance matrix for the features in the dataset.

Step 3: Calculate the eigenvalues and eigenvectors for the covariance matrix.

Step 4: Sort eigenvalues and their corresponding eigenvectors.

Step 5: Pick k eigenvalues and form a matrix of eigenvectors.

Step 6: Transform the original matrix.

1. Standardization: PCA begins by standardizing the dataset, where each feature is scaled to have zero mean and unit variance.

This step ensures that all features contribute equally to the analysis and prevents variables with larger scales from dominating the principal components.

2 Covariance matrix: PCA computes the covariance matrix of the standardized dataset. The covariance matrix indicates the relationships and dependencies between pairs of features.

The diagonal elements of the covariance matrix represent the variances of the features.

3. Eigen decomposition: The next step is to perform an eigen decomposition of the covariance matrix. This involves finding the eigenvalues and eigenvectors of the covariance matrix.

The eigenvalues represent the variance explained by each principal component, and the eigenvectors represent the directions or axes of the principal components.

4. Selection of principal components: The principal components are ranked based on their corresponding eigenvalues, with the largest eigenvalue associated with the first principal component, the second largest with the second principal component, and so on.

By selecting the top-k principal components, where k is the desired lower dimensionality, we retain the most significant variation in the data.

5. Projection: Finally, the data is projected onto the selected principal components to obtain the lower-dimensional representation.

This projection is achieved by taking the dot product between the standardized data and the eigenvectors corresponding to the selected principal components.

1. Standardize the Dataset

Assume we have the below dataset which has 4 features and a total of 5 training examples.

f1	f2	f3	f4
1	2	3	4
5	5	6	7
1	4	2	3
5	3	2	1
8	1	2	2

First, we need to standardize the dataset and for that, we need to calculate the mean and standard deviation for each feature.

$$x_{new} = \frac{x - \mu}{\sigma}$$

	f1	f2	f3	f4
μ =	4	3	3	3.4
σ =	3	1.58114	1.73205	2.30217

After applying the formula for each feature in the dataset is transformed as below:

f1	f2	f3	f4
-1	-0.63246	0	0.26062
0.33333	1.26491	1.73205	1.56374
-1	0.63246	-0.57735	-0.17375
0.33333	0	-0.57735	-1.04249
1.33333	-1.26491	-0.57735	-0.60812

2. Calculate the covariance matrix for the whole dataset

The formula to calculate the covariance matrix:

$$cov(x, y) = (1 / n-1) * \sum((x[i] - mean(x)) * (y[i] - mean(y)))$$

Where:

- cov(x, y) represents the covariance between variables x and y.
- n is the number of data points.
- x[i] and y[i] are the values of x and y for each data point.
- mean(x) and mean(y) are the means of variables x and y, respectively.

the covariance matrix for the given dataset will be calculated as below

	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)

Since we have standardized the dataset, so the **mean for each feature is 0** and the standard deviation is 1.

$var(f1) = ((-1.0-0)^2 + (0.33-0)^2 + (-1.0-0)^2 +(0.33–0)^2 +(1.33–0)^2)/4$

var (f1) = 0.8

cov(f1,f2) =
((-1.0–0)*(-0.632456-0) + (0.33–0)*(1.264911-0) + (-1.0–0)* (0.632456-0)+
(0.33–0)*(0.000000 -0)+ (1.33–0)*(-1.264911–0))/4

cov(f1,f2) = -0.25298

In the similar way, we can calculate the other covariances and which will result in the below covariance matrix

	f1	f2	f3	f4
f1	0.8	-0.25298	0.03849	-0.14479
f2	-0.25298	0.8	0.51121	0.4945
f3	0.03849	0.51121	0.8	0.75236
f4	-0.14479	0.4945	0.75236	0.8

3. Calculate eigenvalues and eigen vectors.

Let A be a square matrix (in our case the covariance matrix), v a vector and λ a scalar that satisfies $Av = \lambda v$, then λ is called eigenvalue associated with eigenvector v of A.

Rearranging the above equation,

$Av-\lambda v =0$;

$(A-\lambda I)v = 0$

Since we have already know v is a non- zero vector, only way this equation can be equal to zero, if

$\det(A-\lambda I) = 0$

	f1	f2	f3	f4
f1	0.8 - λ	-0.25298	0.03849	-0.14479
f2	-0.25298	0.8- λ	0.51121	0.4945
f3	0.03849	0.51121	0.8 - λ	0.75236
f4	-0.14479	0.4945	0.75236	0.8 - λ

$\det(A-\lambda I) = 0$

Solving the above equation = 0

λ = 2.51579324 , 1.0652885 , 0.39388704 , 0.02503121

Eigenvectors:

Solving the (A-λI)v = 0 equation for v vector with different λ values:

$$\begin{pmatrix} 0.800000-\lambda & -(0.252982) & 0.038490 & -(0.144791) \\ -(0.252982) & 0.800000-\lambda & 0.511208 & 0.494498 \\ 0.038490 & 0.511208 & 0.800000-\lambda & 0.752355 \\ -(0.144791) & 0.494498 & 0.752355 & 0.800000-\lambda \end{pmatrix} \times \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = 0$$

For λ = *2.51579324, solving the above equation using Cramer's rule, the values for v vector are*

v1 = 0.16195986

v2 = -0.52404813

v3 = -0.58589647

v4 = -0.59654663

Going by the same approach, we can calculate the eigen vectors for the other eigen values.

We can from a matrix using the eigen vectors.

e1	e2	e3	e4
0.161960	-0.917059	-0.307071	0.196162
-0.524048	0.206922	-0.817319	0.120610
-0.585896	-0.320539	0.188250	-0.720099
-0.596547	-0.115935	0.449733	0.654547

eigenvectors(4 * 4 matrix)

4. Sort eigenvalues and their corresponding eigenvectors.

Since eigenvalues are already sorted in this case so no need to sort them again.

5. Pick k eigenvalues and form a matrix of eigenvectors

If we choose the top 2 eigenvectors, the matrix will look like this:

e1	e2
0.161960	-0.917059
-0.524048	0.206922
-0.585896	-0.320539
-0.596547	-0.115935

Top 2 eigenvectors(4*2 matrix)

6. Transform the original matrix.

Feature matrix * top k eigenvectors = Transformed Data

f1	f2	f3	f4
-1.000000	-0.632456	0.000000	0.260623
0.333333	1.264911	1.732051	1.563740
-1.000000	0.632456	-0.577350	-0.173749
0.333333	0.000000	-0.577350	-1.042493
1.333333	-1.264911	-0.577350	-0.608121

(5,4)

*

e1	e2
0.161960	-0.917059
-0.524048	0.206922
-0.585896	-0.320539
-0.596547	-0.115935

(4,2)

=

nf1	nf2
0.014003	0.755975
-2.556534	-0.780432
-0.051480	1.253135
1.014150	0.000239
1.579861	-1.228917

(5,2)

Python Implementation of PCA

1. **Import all the libraries**

2. **Loading Data**

3. **Apply PCA**

- Standardize the dataset prior to PCA.
- Import PCA from sklearn.decomposition.
- Choose the number of principal components.

Let us select it to 3. After executing this code, we get to know that the dimensions of x are (569,3) while the dimension of actual data is (569,30).

Thus, it is clear that with PCA, the number of dimensions has reduced to 3 from 30. If we choose n_components=2, the dimensions would be reduced to 2.

4. Check Components

The principal.components_ provide an array in which the number of rows tells the number of principal components while the number of columns is equal to the number of features in actual data.

We can easily see that there are three rows as n_components was chosen to be 3. However, each row has 30 columns as in actual data.

5. Plot the components (Visualization)

Plot the principal components for better data visualization.

Though we had taken n_components =3, here we are plotting a 2d graph as well as 3d using first two principal components and 3 principal components respectively.

For three principal components, we need to plot a 3d graph.

The colors show the 2 output classes of the original dataset-benign and malignant.
It is clear that principal components show clear separation between two output classes.

For three principal components, we need to plot a 3d graph. x[:,0] signifies the first principal component. Similarly, x[:,1] and x[:,2] represent the second and the third principal component.

Python implementation

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

To implement PCA in Scikit learn, it is essential to standardize/normalize the data before applying PCA.

•PCA is imported from sklearn.decomposition. We need to select the required number of principal components.

•Usually, n_components is chosen to be 2 for better visualization but it matters and depends on data.

•By the fit and transform method, the attributes are passed.

•The values of principal components can be checked using components_

To Visualize PCA

https://setosa.io/ev/principal-component-analysis/

Tutorial:-
https://towardsdatascience.com/feature-extraction-using-principal-component-analysis-a-simplified-visual-demo-e5592ced100a

here pca.components_ gives Principal axes in feature space, representing the directions of maximum variance in the data