TECHNO INDIA
SILIGURI INSTITUTE
OF TECHNOLOGY
A Satyam Roychowdhury Initiative

Dimensionality Reduction

Principal Component Analysis, Linear Discriminant Analysis, Feature Selection, Feature Manipulation and Normalization

What is dimensionality reduction

- Working directly with high-dimensionality data such as image, sensor data, comes with some difficulties.
- High dimensional data is difficult to interpret, visualize, and hence interpret.
- High dimensional data often contain redundant data.
- Dimensionality reduction is a data compression technique without losing variation in data.

What is Principal Component Analysis?

- PCA proposed by Pearson and Hotelling around 100 year ago.
- till today it is popular for data compression and visualization.
- is a very powerful data reduction techniques.
- PCA is used for exploratory data analysis and for making prediction model.
- Data containing n number of features in the given dataset. We need to extract new m features from n features. $m < n$

How to reduce data?

- It is done by projecting each data point onto only first few principal components to obtain lower dimension data while preserving as much of the data's variation possible.
- Principal components are the eigen vector of data's covariance matrix.

Steps of dimension reduction using PCA

1. Standardize the data
2. Compute the covariance matrix
3. Calculate the eigen vectors and eigen values
4. Sort eigen values in descending order and compute the Principal components
5. Reduce the dimension of the dataset.

Data Standardization?

- Transforming the features in a comparable format.
- Convert data I a common format.
- Data standardization is calculating a z-score. It is given by
- $z = \frac{x - \bar{x}}{\sigma}$
- All the data will be zero mean and 1 s.d

```
1 data
```

	Y	X1	X2	X3
0	0	-62.8	-89.5	1.7
1	0	3.3	-3.5	1.1
2	0	-120.8	-103.2	2.5
3	0	-18.1	-28.8	1.1
4	0	-3.8	-50.6	0.9

Original Dataset

	X1	X2	X3
0	-0.690959	-1.855309	-0.019874
1	0.237914	0.107878	-0.582014
2	-1.506005	-2.168049	0.729646
3	-0.062811	-0.469664	-0.582014
4	0.138141	-0.967309	-0.769394

Standardized data

Covariance Matrix

- Covariance always measured between two variables or features.
- $cov(x_1, x_2) = \frac{\sum(x_1 - \overline{x_1})(x_2 - \overline{x_2})}{N-1}$
- It measure only directional relation between two variables not the strength of the relationship between them.

	X1	X2	X3
X1	1.000000	0.640876	0.046741
X2	0.640876	1.000000	-0.350069
X3	0.046741	-0.350069	1.000000

Eigen Values and Eigen Vectors

- Eigen values and Eigen vectors are the linear algebra concept that are required in the determination of principal components from the covariance matrix.
- Eigenvectors helps in finding the new transformation where there is minimum variance.
- For the eigen value v, and the square matrix C, $Cv = \lambda v$
- Eigen vectors v are those vectors which keep the direction same when multiplied by the matrix C.
- Eigen values λ are the scalar of the respective eigenvectors.

Eigen Values and Eigen Vectors

```
1 eigenvalues, eigenvectors = np.linalg.eig(Z)
2 print('Eigen values:\n', eigenvalues)
3 print('Eigen values Shape:', eigenvalues.shape)
4 print('Eigen Vector Shape:', eigenvectors.shape)
```

```
Eigen values:
 [0.24940267 1.71131587 1.03928146]
Eigen values Shape: (3,)
Eigen Vector Shape: (3, 3)
```

```
1 eigenvectors
```

```
array([[ 0.61787894, -0.62471494,  0.47744828],
       [-0.6971437 , -0.71608783, -0.03476891],
       [-0.36361556,  0.31136708,  0.87797168]])
```

How to obtain Principal Components?

- Sort the eigenvalues in descending order.
- The vector corresponding to the highest value is the first component, and the value corresponding to the second highest eigenvalue is the 2nd principal component, and so on.

```
1  # Index the eigenvalues in descending order
2  idx = eigenvalues.argsort()[::-1]
3  idx
```

```
array([1, 2, 0], dtype=int64)
```

```
1  # Sort the eigenvalues in descending order
2  eigenvalues = eigenvalues[idx]
3  eigenvalues
```

```
array([1.71131587, 1.03928146, 0.24940267])
```

```
1  # sort the corresponding eigenvectors accordingly
2  eigenvectors = eigenvectors[:,idx]
```

```
1  eigenvectors
```

```
array([[-0.62471494,  0.47744828,  0.61787894],
       [-0.71608783, -0.03476891, -0.6971437 ],
       [ 0.31136708,  0.87797168, -0.36361556]])
```

Reduce the dimension of the dataset.

- Explained variance is the term that gives us an idea of the amount of the total variance that has been retained by selecting the principal components instead of the original feature space.
- Determine the number of principal components.

```
1  explained_var = np.cumsum(eigenvalues) / np.sum(eigenvalues)
2  explained_var
```

```
array([0.57043862, 0.91686578, 1.        ])
```

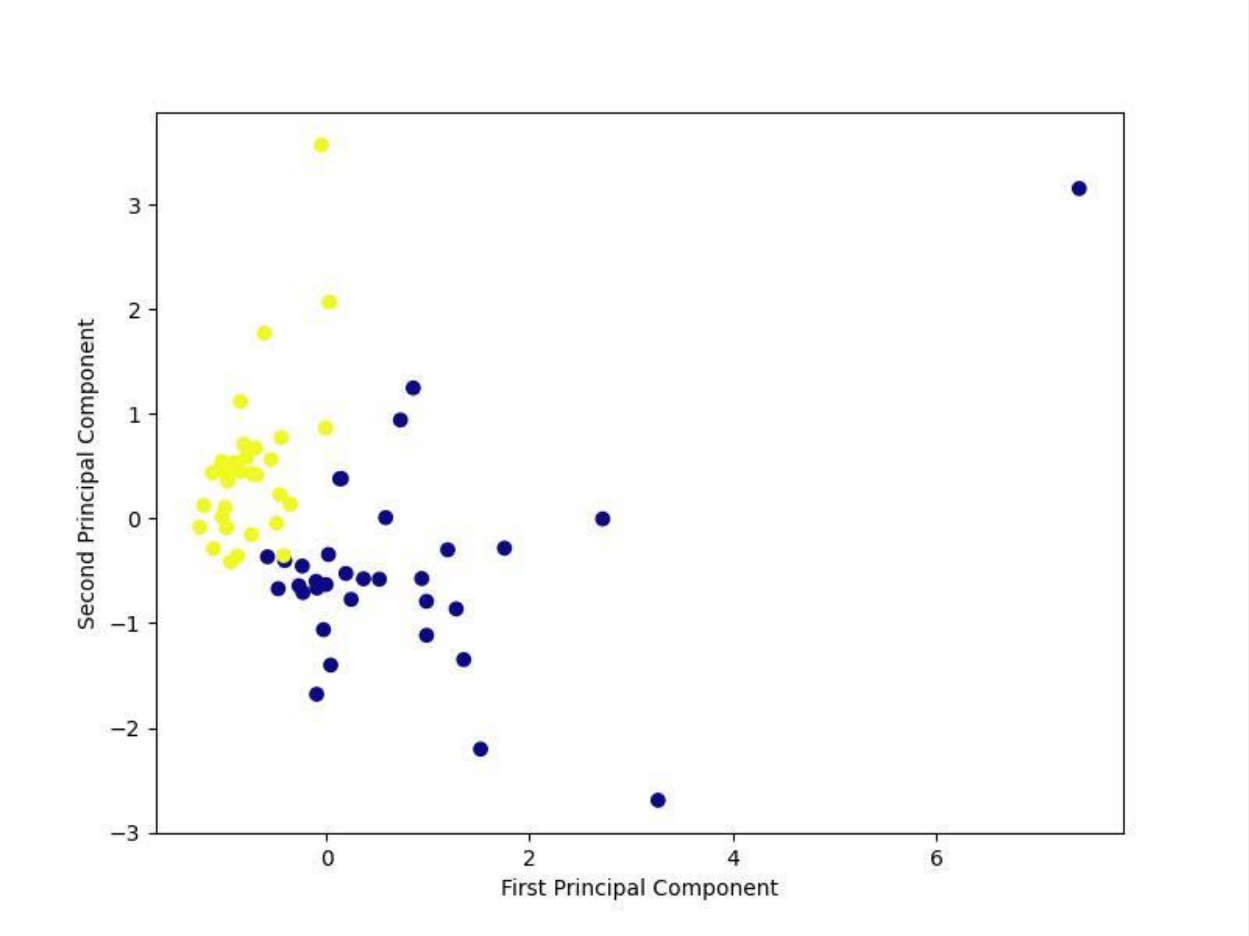
Project the data onto the selected components

- Find the projection matrix, It is a matrix of eigenvectors corresponding to the largest eigenvalues of the covariance matrix of the data.
- It projects the high-dimensional dataset onto a lower-dimensional subspace.

```
1 X_transformed.head()
```

	PCA1	PCA2
0	1.754028	-0.282838
1	-0.407099	-0.401151
2	2.720526	-0.003050
3	0.194339	-0.524651
4	0.366816	-0.575919

Plot the dataset in 2d graph



Linear Discriminant Analysis

- In 1936, Fisher formulated linear discriminant analysis.
- In 1948, C R Rao generalized it for multiple classes.
- It is a supervised classification model for dimensionality reduction.
- LDA project data from a d −dimensional feature space to a d' −dimensional space, thereby maximizing the variability between the classes and reducing the variability within the classes.

What is LDA?

- LDA is a statistical learning technique to categorize data into groups.
- It identifies patterns in features to distinguish between different classes.
- It aims to find a straight line or plan that separates the classes maximizing the distance between classes.
- It minimize the variation of the data within class and maximize variation between classes.

Why LDA?

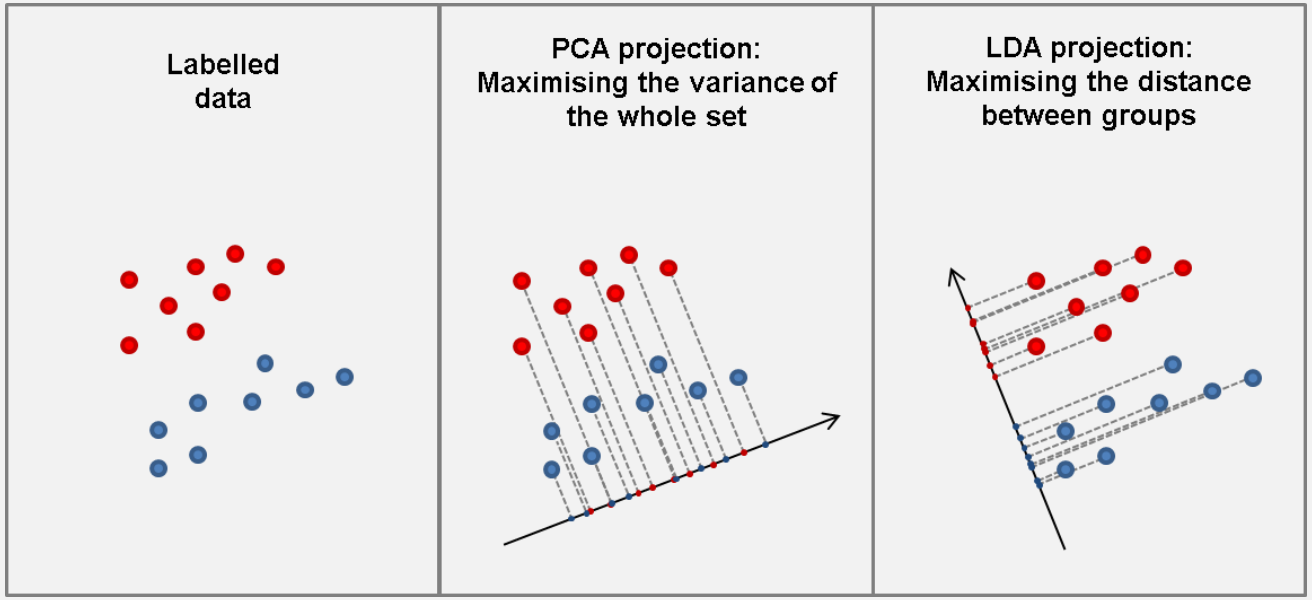
- LDA reduces the number of features in data processing thereby reduce significant computation cost.
- It is used as a linear classifier and works well for multiclass classification problem.

Assumptions of LDA

- The data are linearly separable.
- Data follows multivariate Gaussian distribution.
- Covariance matrix is same for the samples of each class.

How does LDA work?

- The objective is to find out the best projection direction that maximizes the distance between the groups.



Mathematical Intuition behind LDA

- Lets consider we have two classes and d −dimensional samples $\{x_1, x_2, \ldots, x_n\}$ where:
- $n1$ samples coming from the class $c1$ and $n2$ samples coming from the class $c2$
- If x_i is the data point and its projection on the line presented by the unit vector v can be written as $v^T x_i$
- Assume that μ_1 and μ_2 are the mean of the samples of class $c1$ and class $c2$ respectively and $\bar{\mu}_1$ and $\overline{\mu_2}$ are the mean of the classes after projection.

Mathematical Intuition behind LDA

- $\bar{\mu}_1$ and $\bar{\mu}_2$ can be calculated as follows:
- $\bar{\mu}_1 = \frac{1}{n_1}\sum_{x_i \in C1} v^T x_i \ = v^T \mu_1$
- $\bar{\mu}_2 = \frac{1}{n_2}\sum_{x_i \in C2} v^T x_i \ = v^T \mu_2$
- Let consider $\bar{s}_1$ and $\bar{s}_2$ are the scatter for the samples of the class $c1$ and $c1$ after projection.
- As our objective is to maximize variation between class and minimize variation within the class, therefore data will be projected on the line having direction v which maximizes $J(v) = \frac{\bar{\mu}_1 - \bar{\mu}_2}{\bar{s}_1^2 + \bar{s}_2^2}$

Mathematical Intuition behind LDA

- The scatter matrix s_1 and s_2 can be defined as
- $s_1 = \sum_{x_i \in c1}(x_i - \mu_1)(x_i - \mu_1)^T$.
- $s_2 = \sum_{x_i \in c2}(x_i - \mu_2)(x_i - \mu_2)^T$.
- After simplifying the above equation we get scatter within the classes(S_w) and scatter between the class (s_b)
- $s_w = s_1 + s_2$
- $s_b = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$

Mathematical Intuition behind LDA

- Now try to solve the numerator part of $J(v)$
- $J(v) = \frac{\bar{\mu}_1 - \bar{\mu}_2}{\bar{s}_1^2 + \bar{s}_2^2} = \frac{v^T s_b v}{v^T s_w v}$
- To maximize the above equation we need to calculate differentiation w.r.t v
- $\frac{\partial J(v)}{\partial v} = s_b v - \frac{v^T s_b v (s_w v)}{v^T s_w v}$
- $s_b v - \lambda s_w v = 0 \quad where\ \lambda = \frac{v^T s_b v}{v^T s_w v}$

Mathematical Intuition behind LDA

- $s_b v = \lambda s_w v$
- $s_w^{-1} s_b v = \lambda v$
- $Mv = \lambda v \quad where\ M = s_w^{-1} s_b$

Therefore the vector corresponding the highest eigen value will provide maximum value of $J(v)$.

This will provide best solution for LDA.

Feature Selection

- It is intended to reduce the number of input variables to those that are believed to be most useful to a model in order to predict the target variable.
- Feature selection methods can be supervised and unsupervised.
- Unsupervised selection technique ignores the target variable and removes redundant variables using correlation.
- Supervised feature selection techniques are the target variable such as methods that remove irrelevant variables.

Feature Selection

- Supervised Feature selection Algorithm:
- Wrapper: Recursive feature elimination: A machine algorithm is run multiple times with multiple subsets of features and evaluates the performance of the models. The selected features are those subsets that yield the best results.
- Filter:
- Statistical methods
- Intrinsic

Feature manipulation and Normalization

- Feature normalization is mostly needed to eliminate the effect of several quantitative features measured on different scales. If the features are normally distributed, it can be converted into z-scores by centering on the mean and dividing by the standard deviation.
- Feature Scaling is a technique to standardize the independent features present in the data in a fixed range.