Module 6 : Dimensionality Reduction Techniques

Dimensionality reduction plays a pivotal role in data analysis and machine learning, offering a strategic solution to the challenges posed by high-dimensional datasets. As datasets grow in size and complexity, the number of features or dimensions often becomes unwieldy, leading to increased computational demands, potential overfitting, and diminished model interpretability. Dimensionality reduction techniques provide a remedy by capturing the essential information within the data while discarding redundant or less informative features. This process not only streamlines computational tasks but also aids in visualizing data trends, mitigating the risk of the curse of dimensionality, and improving the generalization performance of machine learning models. Dimensionality reduction finds applications across various domains, from image and speech processing to finance and bioinformatics, where extracting meaningful patterns from vast datasets is crucial for making informed decisions and building effective predictive models.

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimensionality reduction technique widely used in data analysis and machine learning. Its primary goal is to transform high-dimensional data into a lower-dimensional representation, capturing the most important information.

Here's the motivation for PCA:

As we aim to identify patterns within datasets, it's desirable for the data to be distributed across each dimension, and we seek independence among these dimensions. Let's revisit some fundamental concepts. Variance serves as a metric for the variability, essentially quantifying the extent to which the dataset is dispersed. In mathematical terms, it represents the average squared deviation from the mean score. The formula employed to calculate variance, denoted as var(x), is expressed as follows.

$$var(x) = \frac{\sum(x_i - \bar{x})^2}{N} \qquad cov(x,y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N}$$

Covariance quantifies the degree to which corresponding elements in two sets of ordered data exhibit similar directional movement. The formula, represented as cov(x, y), captures the covariance between variables x and y. In this context, xi represents the value of x in the ith dimension, while the x bar and y bar denote their respective mean values. Now, let's explore this concept in matrix format. If we have a matrix X with dimensions m*n, containing n data points each with m dimensions, the covariance matrix can be computed as follows:

$$C_x = \frac{1}{n-1}(X - \bar{X})(X - \bar{X})^T \qquad X^T = Transpose\ of\ X$$

Please note that the covariance matrix contains -
1. variance of dimensions as the main diagonal elements
2. covariance of dimensions as the off-diagonal elements

As mentioned earlier, our goal is to ensure that the data is widely dispersed, indicating high variance across its dimensions. Additionally, we aim to eliminate correlated dimensions, meaning that the covariance among dimensions should be zero, signifying their linear independence. Consequently, the aim is to undergo a data transformation so that its covariance matrix exhibits the following characteristics:
1. Significant values as the main diagonal elements.

2. Zero values as the off-diagonal elements.
Hence, the original data points must be transformed to achieve a covariance matrix resembling a diagonal matrix. This process of transforming a matrix into a diagonal matrix is referred to as diagonalization, and it constitutes the primary motivation behind Principal Component Analysis (PCA).

Here's how PCA works:

1. Standardization

Standardize the data when features are measured in diverse units. This entails subtracting the mean and dividing by the standard deviation for each feature. Failure to standardize data with features of varying scales can result in misleading components.

2. Compute the Covariance Matrix

Calculate the covariance matrix as discussed earlier

3. Calculate Eigenvectors and Eigenvalues

Determine the eigenvectors and eigenvalues of the covariance matrix.
$$[Covariance \quad matrix] \cdot [Eigenvector] = [eigenvalue] \cdot [Eigenvector]$$

Eigenvectors represent the directions (principal components), and eigenvalues represent the magnitude of variance in those directions. To understand what eigenvectors and eigenvalues are, you can go through this video:

4. Sort Eigenvalues

Sort the eigenvalues in descending order. The eigenvectors corresponding to the highest eigenvalues are the principal components that capture the most variance in the data.

5. Select Principal Components

Choose the top k eigenvectors (principal components) based on the explained variance needed. Typically, you aim to retain a significant portion of the total variance, like 85%. How explained variance is calculated can be found here.

6. Transform the Data

Now, we can transform the original data using the eigenvectors:

So, if we have m dimensional original n data points then
X : m*n
P : k*m
Y = PX : (k*m)(m*n) = (k*n)
Hence, our new transformed matrix has n data points having k dimensions.

Pros:

1. Dimensionality Reduction:
PCA effectively reduces the number of features, which is beneficial for models that suffer from the curse of dimensionality.

2. Feature Independence:
Principal components are orthogonal (uncorrelated), meaning they capture independent information, simplifying the interpretation of the reduced features.

3. Noise Reduction:
PCA can help reduce noise by focusing on the components that explain the most significant variance in the data.

4. Visualization:
The reduced-dimensional data can be visualized, aiding in understanding the underlying structure and patterns.

Cons:

1. Loss of Interpretability:
Interpretability of the original features may be lost in the transformed space, as principal components are linear combinations of the original features.

2. Assumption of Linearity:
PCA assumes that the relationships between variables are linear, which may not be true in all cases.

3. Sensitive to Scaling:
PCA is sensitive to the scale of the features, so standardization is often required.

4. Outliers Impact Results:
Outliers can significantly impact the results of PCA, as it focuses on capturing the maximum variance, which may be influenced by extreme values.

When to Use:

1. High-Dimensional Data:
PCA is particularly useful when dealing with datasets with a large number of features to mitigate the curse of dimensionality.

2. Collinear Features:
When features are highly correlated, PCA can be effective in capturing the shared information and representing it with fewer components.

3. Visualization:
PCA is beneficial when visualizing high-dimensional data is challenging. It projects data into a lower-dimensional space that can be easily visualized.

5. Linear Relationships:
When the relationships between variables are mostly linear, PCA is a suitable technique.

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) serves as a technique for both dimensionality reduction and classification, aiming to optimize the distinction between various classes within a dataset. LDA is particularly prevalent in supervised learning scenarios where the classes of data points are predetermined. While PCA is considered an "unsupervised" algorithm that disregards class labels, focusing on finding principal components to maximize dataset variance, LDA takes a "supervised" approach. LDA computes "linear discriminants," determining the directions that serve as axes to maximize separation between multiple classes. To delve into the workings of LDA, let's understand how LDA is calculated using an example of the famous "Iris" dataset on the UCI machine learning repository. It contains measurements for 150 iris flowers from three different species.

There are three classes in the Iris dataset:

1. Iris-setosa (n=50)
2. Iris-versicolor (n=50)
3. Iris-virginica (n=50)

There are four features in the Iris dataset:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm

4. petal width in cm

Steps in LDA:

1. We will start off with the computation of the mean vectors mi, (i=1,2,3) of the 3 different flower classes:

```
Mean Vector class 1: [ 5.006  3.418  1.464  0.244]
Mean Vector class 2: [ 5.936  2.77   4.26   1.326]
Mean Vector class 3: [ 6.588  2.974  5.552  2.026]
```

Each vector contains the mean of the 4 features in the dataset for the specific class.

2. Compute the within-class scatter matrix (Sw), which represents the spread of data within each class:

The **within-class scatter** matrix S_W is computed by the following equation:

$$S_W = \sum_{i=1}^{c} S_i$$

where

$$S_i = \sum_{\boldsymbol{x} \in D_i}^{n} (\boldsymbol{x} - \boldsymbol{m}_i)\,(\boldsymbol{x} - \boldsymbol{m}_i)^T$$

(scatter matrix for every class)

and $\boldsymbol{m}_i$ is the mean vector

$$\boldsymbol{m}_i = \frac{1}{n_i} \sum_{\boldsymbol{x} \in D_i}^{n} \boldsymbol{x}_k$$

In our example, it will look like the following:

```
within-class Scatter Matrix:
 [[ 38.9562  13.683    24.614    5.6556]
 [ 13.683   17.035    8.12     4.9132]
 [ 24.614    8.12    27.22     6.2536]
 [  5.6556   4.9132   6.2536   6.1756]]
```

3. Compute the between-class scatter matrix (Sb), which represents the spread between different classes using the following formula:

$$S_B = \sum_{i=1}^{c} N_i(\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^T$$

where
$\boldsymbol{m}$ is the overall mean, and $\boldsymbol{m}_i$ and N_i are the sample mean and sizes of the respective classes.

In our example, it will look like the following:

```
between-class Scatter Matrix:
 [[  63.2121  -19.534   165.1647   71.3631]
 [ -19.534    10.9776  -56.0552  -22.4924]
 [ 165.1647  -56.0552  436.6437  186.9081]
 [  71.3631  -22.4924  186.9081   80.6041]]
```

4. Compute the Eigenvalues and Eigenvectors of Sw-^{1}Sb (similar to PCA). In our case, we have 4 eigenvalues and eigenvectors:

```
Eigenvector 1:
[[-0.2049]
 [-0.3871]
 [ 0.5465]
 [ 0.7138]]
Eigenvalue 1: 3.23e+01
```

```
Eigenvector 2:
[[-0.009 ]
 [-0.589 ]
 [ 0.2543]
 [-0.767 ]]
Eigenvalue 2: 2.78e-01

Eigenvector 3:
[[ 0.179 ]
 [-0.3178]
 [-0.3658]
 [ 0.6011]]
Eigenvalue 3: -4.02e-17

Eigenvector 4:
[[ 0.179 ]
 [-0.3178]
 [-0.3658]
 [ 0.6011]]
Eigenvalue 4: -4.02e-17
```

5. Sort the eigenvectors by decreasing eigenvalues and pick the top k. After sorting the eigenpairs by decreasing eigenvalues, it is now time to construct our d×k dimensional eigenvector matrix (let's call it W) based on the 2 most informative eigenpairs. We get the following matrix in our example:

```
Matrix W:
 [[-0.2049 -0.009 ]
 [-0.3871 -0.589 ]
 [ 0.5465  0.2543]
 [ 0.7138 -0.767 ]]
```

6. Use the matrix W (4 X 2 matrix) to transform our samples onto the new subspace via the equation: Y = X*W, where X is the original data frame in matrix format (150 X 4 matrix in our case) and Y is the transformed dataset (150 X 2 matrix).

Pros:

1. Maximizes Class Separation:
LDA is designed to maximize the separation between different classes, making it effective for classification tasks.

2. Dimensionality Reduction:
Like PCA, LDA can be used for dimensionality reduction, but with the advantage of considering class information.

Cons:

1. Sensitivity to Outliers:
LDA is sensitive to outliers, and the presence of outliers can affect the performance of the method.

2. Assumption of Normality:
LDA assumes that the features within each class are normally distributed, and it may not perform well if this assumption is violated.

3. Requires Sufficient Samples:
LDA may not perform well with a small number of samples per class. Having more samples improves the estimation of class parameters.

When to Use:

1. Classification Tasks
LDA is beneficial when the goal is to classify data into predefined classes.

2. Preserving Class Information:
When the goal is to reduce dimensionality while preserving information that is relevant for discriminating between classes.

3. Normality Assumption Holds:
LDA performs well when the assumption of normal distribution within each class is valid.

4. Supervised Dimensionality Reduction:
When the task requires dimensionality reduction with the guidance of class labels, LDA is a suitable choice.