Feature transformation and Comparison with feature selection, and feature construction:

Feature transformation is simply a function that transforms features from one representation to another.

Feature Transformations, a suite of algorithms designed to alter the internal structure of data to produce mathematically superior *super-columns.*

Feature transformations are a set of matrix algorithms that will structurally alter our data and produce what is essentially a brand-new matrix of data.

Feature Transformations create a new set of features that explain the data-points, perhaps even better, with fewer columns.

Dimension reduction – feature transformations versus feature selection versus feature construction

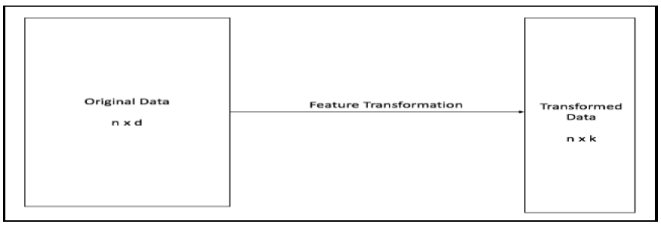
- The toughest part of feature transformations is the suspension of our belief that original feature space is the best.
- Must be open to the fact that there may be other mathematical axes and systems that describe our data just as well with fewer features, or possibly even better.
- Could squish datasets to have fewer columns to describe data in new ways.
- Feature selection processes are limited to only being able to select features from the original set of columns.
- Feature construction is again limited to constructing new columns using simple operations (addition, multiplication, and so on) between a few columns at a time.
- While feature transformation algorithms use these original columns and combine them in useful ways to create new columns that are better at describing the data than any single column from the original dataset.
- Feature transformation methods create new columns using hidden structures in the original datasets to produce an entirely new, structurally different dataset.
- Create brand new columns that are so powerful that we only need a few of them to explain our entire dataset accurately.
- Feature Transformation works by producing new columns that capture the essence (variance) of the data.
- Utilize small bits of information from all columns in every new super-column.

- Feature transformation algorithms are able to *construct* new features by *selecting* best of all columns and combining this latent structure with a few brand-new columns.

Aspect	Feature Selection	Feature Transformations	Feature Construction
Definition	Choosing a subset of the most relevant features.	Changing the representation of features while preserving information.	Creating new features from existing ones.
Example	Univariate methods, recursive feature elimination, model-based methods.	Scaling, normalization, PCA, mathematical transformations.	RBM, Word2Vec, Interaction terms, polynomial features, aggregations.
Purpose	Reduce dimensionality, improve interpretability, speed up training.	Address scale differences, non-linear relationships, or skewed distributions.	Provide additional information, capture complex relationships.
Overlap	Limited overlap, as feature selection focuses on choosing existing features.	Some overlap with feature construction, as transformations may create new features.	Overlaps with transformations, but the primary focus is on creating new features.
Objective	Choose the most relevant features.	Preprocess data for modeling.	Enhance information available to the model.
Methodology	Statistical or model-based criteria.	Standard mathematical operations, scaling, dimensionality reduction.	Based on domain knowledge, data characteristics.
Impact on Dimensionality	Explicitly aims to reduce dimensionality.	Can impact dimensionality depending on the transformation.	Can impact dimensionality based on the nature of the construction.
Use Case	Dealing with high-dimensional data to focus on key features.	Preprocessing step to make data amenable to modeling.	When additional information can be derived from existing features.

Principal Component Analysis (PCA):

Principal Component Analysis is a technique that takes datasets that have several correlated features and projects them onto a coordinate (axis) system that has fewer correlated features. These new, uncorrelated features (referred as super-columns) are called principal components. The principal components serve as an alternative coordinate system to the original feature space that requires fewer features and captures as much variance as possible.



Step by Step Computation of PCA

1. Standardization of the data
2. Computing the covariance matrix
3. Calculating the eigenvectors and eigenvalues
4. Computing the Principal Components
5. Reducing the dimensions of the data set

Details of PCA are as under:

Step 1: Standardization of the data

Missing out on standardization will probably result in a biased outcome. Standardization is all about scaling data in such a way that all variables and their values lie within a similar range. Standardizing data into a comparable range is very important. Post this step, all the variables in the data are scaled across a standard and comparable scale.

$$Z = \frac{variable\ value - mean}{Standard\ Deviation} + \cdots$$

Step 2: Computing the covariance matrix

PCA helps to identify correlation and dependencies among the features. A covariance matrix expresses the correlation between the different variables in the data set. It is essential to identify heavily dependent variables because they contain biased and redundant information. Mathematically, a covariance matrix is a p × p matrix, where p represents dimensions of the data set. Each entry in matrix represents covariance of corresponding variables.

Consider a case where we have a 2-Dimensional data set with variables a and b, the covariance matrix is a 2×2 matrix as shown below:

In the matrix:

- Cov(a, a) represents the covariance of a variable with itself, which is nothing but the variance of the variable 'a'
- Cov(a, b) represents the covariance of the variable 'a' with respect to the variable 'b'.
- And since covariance is commutative, Cov(a, b) = Cov(b, a)

Step 3: Calculating the Eigenvectors and Eigenvalues

Eigenvectors and eigenvalues are the mathematical constructs that must be computed from the covariance matrix in order to determine the principal components of the data set.

Eigenvalues and **Eigenvectors** have following components:

- The **eigenvector** is an array with n entries where n is the number of rows (or columns) of a square matrix. The **eigenvector** is represented as x.
- **Eigenvalues is** a scalar associated with a given linear transformation of a vector space

Step 4: Computing the Principal Components

- Once computed the Eigenvectors and eigenvalues, order them in the descending order, where the eigenvector with the highest eigenvalue is the most significant and thus forms the first principal component.
- The principal components of lesser significances can thus be removed in order to reduce the dimensions of the data.
- The final step in computing the Principal Components is to form a matrix known as the feature matrix that contains all the significant data variables that possess maximum information about the data.

Step 5: Reducing the dimensions of the data set

- Last step in performing PCA is to re-arrange the original data with the final principal components which represent the maximum and the most significant information.
- In order to replace original data axis with newly formed Principal Components, simply multiply transpose of the original data set by the transpose of the obtained feature vector.

Use of PCA for reducedimensionality reduction:

Principal Component Analysis (PCA) is a dimensionality reduction technique commonly used in machine learning and statistics. Its primary goal is to transform a dataset into a new coordinate system, where the features (variables) are represented by a set of linearly uncorrelated variables called principal components. These components are ordered by the amount of variance they capture in the data, with the first principal component capturing the maximum variance.

Step by Step Computation of PCA

1. Standardization of the data
2. Computing the covariance matrix
3. Calculating the eigenvectors and eigenvalues
4. Computing the Principal Components
5. Reducing the dimensions of the data set

Reduce Dimensionality using PCA:

In PCA the original data is projected onto the subspace defined by the selected principal components. This involves multiplying the standardized data by the matrix of selected eigenvectors.

The reduced-dimensional data contains the transformed values along the selected principal components. These new features are linear combinations of the original features, and they capture the most significant information in the data.

The decision of how many principal components to retain depends on the desired level of dimensionality reduction and the amount of variance one wants to preserve. A common approach is to set a threshold for the cumulative explained variance (e.g., retaining 95% of the variance) and select the corresponding number of principal components.

By reducing the dimensionality of the data, PCA can help mitigate the curse of dimensionality, improve computational efficiency, and often enhance the performance of machine learning models by focusing on the most informative features.

PCA and identification of Principal Components

Principal Components (PCs) are the key components obtained through Principal Component Analysis (PCA). They represent a set of new variables that are linear combinations of the original features in a dataset. These new variables are chosen in such a way that they capture the maximum variance in the data.

In the context of PCA, the first principal component (PC1) is the linear combination of the original features that accounts for the most variance in the dataset. The second principal component (PC2) is the linear combination that captures the second most variance, and so on. Each principal component is orthogonal (uncorrelated) to the others, meaning that they represent different directions in the feature space.

Mathematically, if X is the matrix representing the standardized data (where each column corresponds to a feature and each row to an observation), and v_i is the eigenvector associated with the i-th largest eigenvalue of the covariance matrix of X, then the i-th principal component can be calculated as:

$$PC_i = X \cdot v_i$$

Reduce Dimensionality using PCA:

In PCA the original data is projected onto the subspace defined by the selected principal components. This involves multiplying the standardized data by the matrix of selected eigenvectors.

The reduced-dimensional data contains the transformed values along the selected principal components. These new features are linear combinations of the original features, and they capture the most significant information in the data.

The decision of how many principal components to retain depends on the desired level of dimensionality reduction and the amount of variance one wants to preserve. A common approach is to set a threshold for the cumulative explained variance (e.g., retaining 95% of the variance) and select the corresponding number of principal components.

By reducing the dimensionality of the data, PCA can help mitigate the curse of dimensionality, improve computational efficiency, and often enhance the performance of machine learning models by focusing on the most informative features.

Identifying principal components involves finding the eigenvectors and eigenvalues of the covariance matrix of the standardized data. The eigenvectors represent the directions in the feature space, and the eigenvalues indicate the amount of variance along these directions. Here are the key steps in identifying principal components.

Linear Discriminant Analysis (LDA):

Linear Discriminant Analysis (LDA) is a feature transformation technique as well as a supervised classifier. It is commonly used as a preprocessing step for classification pipelines.

The goal of LDA, like PCA, is to extract a new coordinate system and project datasets onto a lower-dimensional space. Main difference between LDA and PCA - instead of focusing on variance of the data as a whole like PCA, LDA optimizes the lower-dimensional space for best class separability. This means that the new coordinate system is more useful in finding decision boundaries for classification models, which is perfect for us when building classification pipelines. The reason that LDA is extremely useful is that separating based on class separability helps us avoid overfitting in our machine learning pipelines. This is also known as preventing the curse of dimensionality. LDA also reduces computational costs. In addition to finding the component axes that maximize the variance of our data (PCA), interested in the axes that maximize the separation between multiple classes (LDA).

LDA and PCA are linear transformation techniques used for dimensionality reduction.

PCA an “unsupervised” algorithm, and its goal is to find the directions (the so-called principal components) that maximize the variance in a dataset.

In contrast to PCA, LDA is “supervised” and computes the directions (“linear discriminants”) that will represent the axes that that maximize the separation between multiple classes.

Summarizing the LDA approach in 5 steps

1. Compute the d-dimensional mean vectors for different classes from dataset.
2. Compute the scatter matrices (in-between-class and within-class scatter matrix).

3. Compute the eigenvectors ($e_1,e_2,...,e_d$) and corresponding eigenvalues ($\lambda_1,\lambda_2,...,\lambda_d$) for the scatter matrices.
4. Sort eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a d×k dimensional matrix W
5. Use this d×k eigenvector matrix to transform the samples onto the new subspace.

   Y=X×W

   (where X is a n×d- dimensional matrix representing n samples, and

   y are transformed n×k-dimensional samples in new subspace).

We will explain with Iris Data Set.

Step 1: Computing d-dimensional mean vectors

Start off with a simple computation of the mean vectors mi, (i=1,2,3) of the 3 different flower classes:

$$\boldsymbol{m}_i = \begin{bmatrix} \mu_{\omega_i\,(\text{sepal length})} \\ \mu_{\omega_i\,(\text{sepal width})} \\ \mu_{\omega_i\,(\text{petal length})} \\ \mu_{\omega_i\,(\text{petal width})} \end{bmatrix}, \text{ with } \quad i = 1, 2, 3$$

Step 2: Computing the Scatter Matrices

Compute the two 4x4-dimensional matrices:

- ❑ The within-class and
- ❑ the between-class scatter matrix.

Within-class scatter matrix S_W

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

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

Between-class scatter matrix S_B

The **between-class scatter** matrix S_B is computed by the following equation:

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

where

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

Step 3: Solving the generalized eigenvalue problem for the matrix $S_W^{-1}S_B$

Solve the generalized eigenvalue problem for the matrix $S_W^{-1}S_B$ to obtain linear discriminants.

If v is an eigenvector of a matrix Σ, we have

$$\Sigma v = \lambda v$$

Here, λ is eigenvalue, and v is also an eigenvector

A quick check that eigenvector-eigenvalue calculation is correct and satisfy equation:

$Av=\lambda v$

where

$A=S_W^{-1}S_B$

v=Eigenvector

λ=Eigenvalue

Step 4: Selecting linear discriminants for the new feature subspace

Interested in merely projecting the data into a subspace that improves the class separability, but also reduces the dimensionality of our feature space. Eigenvectors will form the axes of this new feature subspace. In order to decide which eigenvector(s) we want to drop for our lower-dimensional subspace, we have to take a look at the corresponding eigenvalues of eigenvectors. Eigenvectors with lowest eigenvalues bear least information about the distribution of the data, and those are the ones we want to drop. The common approach is to rank the eigenvectors from highest to lowest corresponding eigenvalue and choose the top k eigenvectors. After sorting eigenpairs by decreasing eigenvalues, construct k×d-dimensional eigenvector matrix W.

Step 5: Transforming the samples onto the new subspace

Interested in merely projecting the data into a subspace that improves the class separability, but also reduces the dimensionality of our feature space. Use matrix W to transform samples onto the new subspace via the equation

$$Y=X\times W$$

where X is a n×d- dimensional matrix representing n samples, and

Y are transformed n×k- dimensional samples in new subspace

Comparison between PCA and LDA:

Principal Component Analysis (PCA):

Principal Component Analysis (PCA) is a dimensionality reduction technique used in statistics and machine learning. Its primary goal is to transform a dataset into a new coordinate system, where the features (variables) are represented by a set of linearly uncorrelated variables called principal components. These principal components are ordered by the amount of variance they capture in the data. PCA identifies the directions (principal components) along which the data varies the most and projects the original data onto these directions.

Linear Discriminant Analysis (LDA):

Linear Discriminant Analysis (LDA) is a supervised dimensionality reduction technique that is commonly used for classification problems. Unlike PCA, which focuses on capturing maximum variance, LDA aims to find the linear combinations of features that best separate different classes in the data. It seeks to maximize the distance between the means of different classes while minimizing the spread (variance) within each class.

Aspect	Principal Component Analysis (PCA)	Linear Discriminant Analysis (LDA)
Objective	Unsupervised; Maximizes overall variance.	Supervised; Maximizes separation between classes.
Supervision	Unsupervised; Does not use class labels.	Supervised; Utilizes class labels.
Variance vs. Discrimination	Maximizes overall variance in the data.	Maximizes the ratio of between-class to within-class variance.
Use Cases	General dimensionality reduction, noise reduction, visualization.	Specifically designed for classification tasks.

Aspect	Principal Component Analysis (PCA)	Linear Discriminant Analysis (LDA)
Mathematical Basis	Involves eigenvectors and eigenvalues of covariance matrix.	Involves scatter matrices and linear discriminants.
Number of Components	Equal to the number of original features.	At most $c-1$ where c is the number of classes.
Data Assumptions	Assumes directions of maximum variance capture important information.	Assumes normal distribution and equal covariance matrices for classes.

Whitening in Principal Component Analysis (PCA) and its need:

Whitening is the process of transforming the data in such a way that the resulting features (principal components) have unit variances and are uncorrelated.

Why Whitening?

Whitening helps in decorrelating the principal components, making them statistically independent. This simplifies further analysis. It ensures that all the principal components have equal importance by giving them unit variances. It can help mitigate issues related to different scales or units in the original data.

Steps for Whitening in PCA:

- Standardize the data: Subtract the mean from each feature to center the data around the origin.
- Compute the covariance matrix: Calculate the covariance matrix of the centered data.
- Eigen decomposition: Find the eigenvalues and eigenvectors of the covariance matrix.
- Diagonalization: Create a diagonal matrix with the square roots of the eigenvalues and transform the data using the eigenvectors.
- Scaling: Scale the transformed data by the inverse of the diagonal matrix to ensure unit variances for each principal component.

Here's the mathematical treatment of whitening in PCA:

1. Standardize the Data:

Subtract the mean of each feature from the data to center it around the origin.

Centered Data: $X_c = X - \mu$

Where X is the original data matrix, and μ is the mean vector of the features.

2. Compute the Covariance Matrix:

Calculate the covariance matrix of the centered data. The covariance matrix is represented as Σ.

Covariance Matrix: $\Sigma = \frac{1}{N} X_c^T X_c$

Where N is the number of data points.

3. Eigen decomposition of the Covariance Matrix:

Find the eigenvalues (λ_i) and eigenvectors (v_i) of the covariance matrix Σ.

$\Sigma vi=\lambda ivi$

4. Diagonalization:

Create a diagonal matrix (D) with the square roots of the eigenvalues (λ_i) and a matrix (P) with the eigenvectors (v_i).

Diagonal Matrix: $D=\text{diag}(\lambda_1,\lambda_2,\ldots,\lambda_n)$

Eigenvector Matrix: $P=[v_1,v_2,\ldots,v_n]$

Whitening Transformation:

Transform the centered data Xc using the eigenvector matrix P and the diagonal matrix D to obtain the whitened data X_w.

Whitened Data: $Xw=X_c \cdot P \cdot D^{-1}$

6. Scaling to Unit Variances:

To ensure that the principal components have unit variances, scale the whitened data by the inverse of the diagonal matrix D.

Whitened Data with Unit Variances: $X'_w=X_w \cdot D^{-1}$

Need of Whitening in PCA:

The whitening transformation involves using the inverse square root of the covariance matrix, making the components more suitable for downstream analyses. Whitening introduces computational costs, and its necessity depends on specific analysis or modeling requirements.

Zero Component Analysis (ZCA) its applications:

ZCA (Zero Component Analysis) is a technique related to Principal Component Analysis (PCA) that is used for data preprocessing and dimensionality reduction. While PCA focuses on decorrelating and reducing the dimensionality of data, ZCA further ensures that the transformed data has a whitened or uncorrelated distribution, meaning that the features are not only decorrelated but also have the same variances. Whitening process un-correlates principal components.

ZCA goes a step further by not only making the components uncorrelated and having unit variances but also by rotating them back to the original feature space. This ensures that the transformed data retains some of the original data's structure and orientation. ZCA Transformation: The ZCA transformation is achieved by rotating the whitened data back to the original feature space using the inverse of the PCA transformation matrix. This process eliminates the correlations while preserving the original data's orientation.

ZCA whitening goes a step further to ensure that the whitened data retains some of the original data's structure and orientation. This is done by rotating the whitened data back to the original feature space.

a. Compute the ZCA transformation matrix Z:

$Z=W \cdot V^T$

where V is a matrix whose columns are eigenvectors of C (covariance matrix)

W is the whitening matrix

This effectively undoes the whitening transformation and restores the original feature space.

b. Apply the ZCA transformation to the whitened data:

$X_{ZCA}=X_{whitened} \cdot Z$

The result X_{ZCA} is the ZCA-whitened data, which retains the original data's orientation while having uncorrelated features and unit variances.

Applications:

1. ZCA whitening is commonly used in image processing, especially for tasks like face recognition and image denoising.
2. It can be beneficial when dealing with natural images, as it helps maintain the spatial structure and texture information.

Limitations and use cases of Principal Component Analysis (PCA):

Principal Component Analysis (PCA):

Principal Component Analysis (PCA) is a dimensionality reduction technique used in statistics and machine learning. Its primary goal is to transform a dataset into a new coordinate system, where the features (variables) are represented by a set of linearly uncorrelated variables called principal components. These principal components are ordered by the amount of variance they capture in the data. PCA identifies the directions (principal components) along which the data varies the most and projects the original data onto these directions.

Limitations:

1. When using PCA for dimensionality reduction, one must address the question of how many principal components (*k*) to use. One possibility is to pick *k* to account for a desired proportion of total variance.
2. One key criticism of PCA is that the transformation is fairly complex, and the results are therefore hard to interpret.
3. PCA is computationally expensive
4. It is difficult to perform PCA in a streaming fashion.

5. Lastly, it is best not to apply PCA to raw counts (word counts, music play counts, movie viewing counts, etc.).

Use Cases of Principal Component Analysis (PCA):

1. One of the coolest applications of PCA is in anomaly detection of time series.
2. PCA is also often used in financial modeling.
3. ZCA is useful as a preprocessing step when learning from images.
4. Many deep learning models use PCA or ZCA as a preprocessing step, though it is not always necessary.