Dimensionality Reduction

Dimensionality reduction in machine learning is the process of reducing the number of input variables (features) in a dataset while retaining essential information. It addresses the "curse of dimensionality" by removing redundant or noisy data, which enhances model performance, reduces computational time, and prevents overfitting. Common techniques include PCA, LDA, and t-SNE.

When working with machine learning models, datasets with too many features can cause issues like slow computation and overfitting. Dimensionality reduction helps to reduce the number of features while retaining key information. It converts high-dimensional data into a lower-dimensional space while preserving important details.

Dimensionality reduction techniques can be broadly divided into two categories:

1. Feature Selection

Feature selection chooses the most relevant features from the dataset without altering them. It helps remove redundant or irrelevant features, improving model efficiency. Some common methods are:

- Filter methods rank the features based on their relevance to the target variable.
- Wrapper methods use the model performance as the criteria for selecting features.
- Embedded methods combine feature selection with the model training process.

2. Feature Extraction

Feature extraction involves creating new features by combining or transforming the original features. These new features retain most of the dataset's important information in fewer dimensions. Common feature extraction methods are:

1. Principal Component Analysis (PCA): Converts correlated variables into uncorrelated principal components hence reducing dimensionality while maintaining as much variance as possible enabling more efficient analysis.
2. Missing Value Ratio: Variables with missing data beyond a set threshold are removed, improving dataset reliability.
3. Backward Feature Elimination: Starts with all features and removes the least significant ones in each iteration. The process continues until only the most impactful features remain, optimizing model performance.
4. Forward Feature Selection: It begins with one feature, adds others incrementally and keeps those improving model performance.
5. Random Forest: Random forest uses decision trees to evaluate feature importance, automatically selecting the most relevant features without the need for manual coding, enhancing model accuracy.
6. Factor Analysis: Groups variables by correlation and keeps the most relevant ones for further analysis.
7. Independent Component Analysis (ICA): Identifies statistically independent components, ideal for applications like 'blind source separation' where traditional correlation-based methods fall short.

PCA and t-SNE

PCA (Principal Component Analysis) is a linear technique that works best with data that has a linear structure. It seeks to identify the underlying principal components in the data by projecting onto lower dimensions, minimizing variance, and preserving large pairwise distances. Read our Principal Component Analysis (PCA) tutorial to understand the inner workings of the algorithms with R examples.

But, t-SNE is a nonlinear technique that focuses on preserving the pairwise similarities between data points in a lower-dimensional space. t-SNE is concerned with preserving small pairwise distances whereas, PCA focuses on maintaining large pairwise distances to maximize variance.

In summary, PCA preserves the variance in the data. In contrast, t-SNE preserves the relationships between data points in a lower-dimensional space, making it quite a good algorithm for visualizing complex high-dimensional data.

The following table can help you compare t-SNE and PCA side by side:

Characteristic	t-SNE	PCA
Type	Non-linear dimensionality reduction	Linear dimensionality reduction
Goal	Preserve local pairwise similarities	Preserve global variance
Best used for	Visualizing complex, high-dimensional data	Data with linear structure

Characteristic	t-SNE	PCA
Output	Low-dimensional representation	Principal components
Use cases	Clustering, anomaly detection, NLP	Noise reduction, feature extraction
Computational intensity	High	Low
Interpretation	Harder to interpret	Easier to interpret

How t-SNE Works

The t-SNE algorithm finds the similarity measure between pairs of instances in higher and lower dimensional space. After that, it tries to optimize two similarity measures. It does all of that in three steps.

1. t-SNE models a point being selected as a neighbor of another point in both higher and lower dimensions. It starts by calculating a pairwise similarity between all data points in the high-dimensional space using a Gaussian kernel. The points far apart have a lower probability of being picked than the points close together.
2. The algorithm then tries to map higher-dimensional data points onto lower-dimensional space while preserving the pairwise similarities.
3. It is achieved by minimizing the divergence between the original high-dimensional and lower-dimensional probability distribution. The algorithm uses gradient descent to minimize the divergence. The lower-dimensional embedding is optimized to a stable state.

The optimization process allows the creation of clusters and sub-clusters of similar data points in the lower-dimensional space, which are visualized to understand the structure and relationships in the higher-dimensional data.