Data Analysis

Missing values and Dimensionality Reduction

CentraleDigitalLab@Nice

Missing Values

Handling missing values is a critical step in preparing data for machine learning models since many **models cannot handle missing values** out of the box.

Missing Values

Handling missing values is a critical step in preparing data for machine learning models since many **models cannot handle missing values** out of the box.

We have three types of missing values:

- Missing Completely At Random (MCAR)
- Missing At Random (MAR)
- Missing Not At Random (MNAR)

Missing Completely At Random (MCAR)

Definition: Data is said to be Missing Completely At Random (MCAR) when the **probability of missingness is the same for all observations**. In other words, the missingness is independent of both observed and unobserved data.

Missing Completely At Random (MCAR)

Definition: Data is said to be Missing Completely At Random (MCAR) when the **probability of missingness is the same for all observations**. In other words, the missingness is independent of both observed and unobserved data.

Example: Suppose you have a dataset of student test scores and due to a clerical error, random pages of the data are lost. The **missing data** is likely MCAR as it **has no relationship with any other data**.

Missing Completely At Random (MCAR)

Definition: Data is said to be Missing Completely At Random (MCAR) when the **probability of missingness is the same for all observations**. In other words, the missingness is independent of both observed and unobserved data.

Solution: **Simple Imputer**, such as using the **mean**, **median**, or **most frequent value** of the feature. You can also use a constant value.

Missing At Random (MAR)

Definition: Data is said to be Missing At Random (MAR) if the probability of missingness is systematic and **can be predicted by other observed data**, **but not by the missing data itself**.

Example: In a health survey, suppose females are less likely to report their weight. The missingness of weight data is systematic and related to the observed gender data but is not related to the weight data itself.

Missing At Random (MAR)

Definition: Data is said to be Missing At Random (MAR) if the probability of missingness is systematic and **can be predicted by other observed data**, **but not by the missing data itself**.

Missing At Random (MAR)

Definition: Data is said to be Missing At Random (MAR) if the probability of missingness is systematic and **can be predicted by other observed data**, **but not by the missing data itself**.

Solution: K-Nearest Neighbors (KNN) Imputation, Iterative Imputer, or Regression Models. The other variables in the dataset predict the missing variable.

Missing At Random (MAR)

Definition: Data is said to be Missing At Random (MAR) if the probability of missingness is systematic and **can be predicted by other observed data**, **but not by the missing data itself**.

K-Nearest Neighbors (KNN) Imputation works as follows:

1. Select the k nearest neighbors based on a distance metric.
2. Calculate the mean, median, mode, or weighted average value of the K neighbours to fill the missing values.

Missing At Random (MAR)

Definition: Data is said to be Missing At Random (MAR) if the probability of missingness is systematic and **can be predicted by other observed data**, **but not by the missing data itself**.

Iterative Imputer works as follows:

1. Fill in missing values with a simple imputation (mean, median, etc)
2. Create a **regression model to predict** missing values in the selected feature based on the values of the other features.
3. **Impute** missing values **with the regressor**.
4. **Iterate** through this process multiple times in a round-robin fashion **until the imputation values converge**.

Missing Not At Random (MNAR)

Definition: Data is Missing Not At Random (MNAR) if the **probability of missingness is related to the missing data itself**, even when controlling for other observed variables.

Example: In a mental health survey, individuals with severe depression might be less likely to respond to questions about mental health. The **missingness** in responses is **related to the unobserved** mental health **data**.

Missing Not At Random (MNAR)

Definition: Data is Missing Not At Random (MNAR) if the **probability of missingness is related to the missing data itself**, even when controlling for other observed variables.

Solution: Advanced Model-Based Imputation such as Random Forests or Bayesian Networks which can capture complex relationships in data and might reveal the hidden structure causing the missingness.

Demo with notebook

01__missing_values.ipynb

Dimensionality Reduction

Reduce the number of columns or variables in our dataset

+

Preserve as much information as possible

Dimensionality Reduction

Let's express the data set as a **matrix X with n rows and m columns**. Each row is a vector x_i that inhabits a mathematical space with m dimensions. **Each dimension intuitively corresponds to a column**.

$$X \in \mathbb{R}^{n \times m}; x_i \in \mathbb{R}^m$$

Dimensionality Reduction

Let's express the data set as a **matrix X with n rows and m columns**. Each row is a vector x_i that inhabits a mathematical space with m dimensions. **Each dimension intuitively corresponds to a column**.

$$X \in \mathbb{R}^{n \times m}; x_i \in \mathbb{R}^m$$

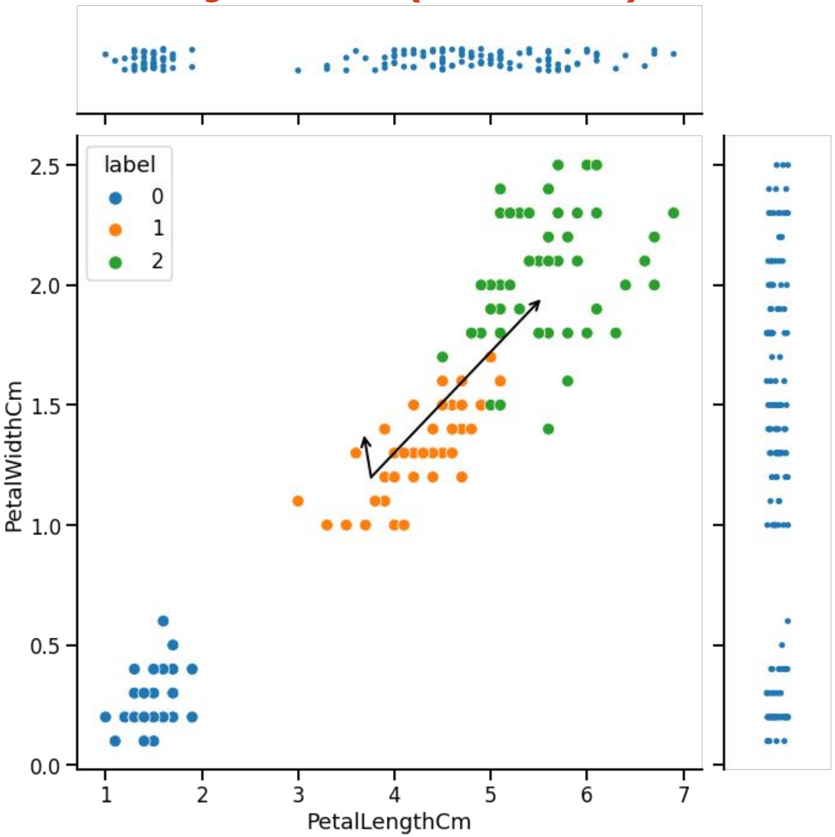
We want to obtain a **new matrix Z** that has the **same number of rows**, but a number of **columns d much smaller than m**.

$$Z \in \mathbb{R}^{n \times d}; d \ll m$$

Principal Components Analysis (PCA)

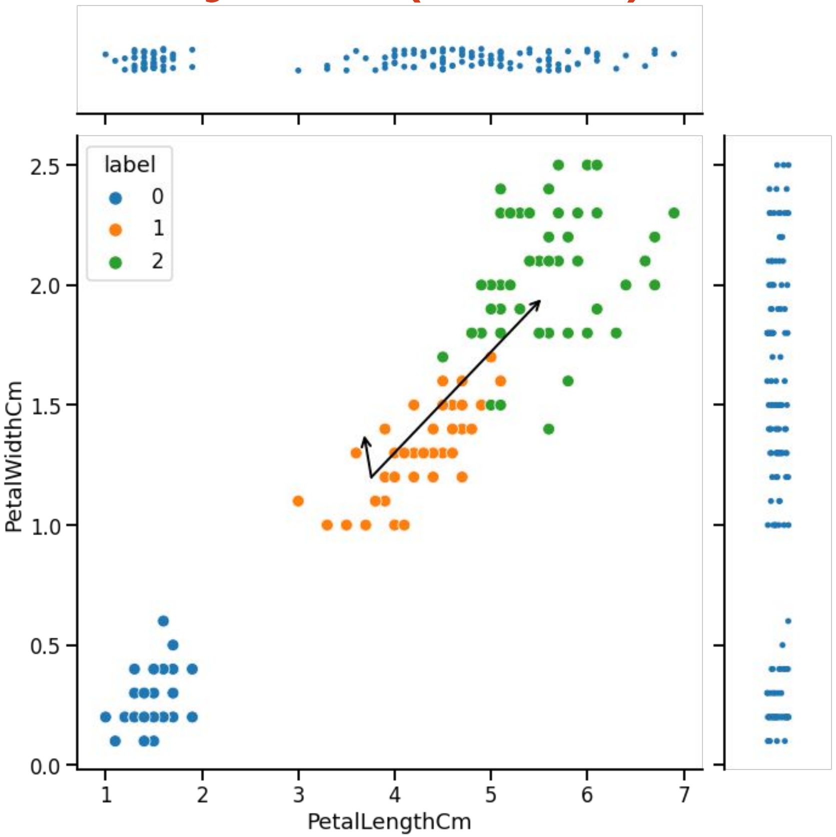
PCA is a statistical procedure that **transforms a set of observations** of possibly correlated variables into a set of values of **linearly uncorrelated variables** called principal components.

This technique is used for dimensionality reduction, visualization, and noise reduction.



Principal Components Analysis (PCA)

PCA is a statistical procedure that **transforms a set of observations** of possibly correlated variables into a set of values of **linearly uncorrelated variables** called principal components.



Principal Components Analysis (PCA)

Pros:

- **Dimensionality Reduction**: PCA allows for the reduction of the dataset dimensionality while retaining the most important information.
- **Improves Model Performance**: By eliminating redundant features, PCA can improve the performance of ML models.
- **Noise Reduction**: PCA can help in noise reduction by isolating and discarding lower-variance components.
- **Visualization**: Helps in visualizing high-dimensional data in a 2D or 3D space which can help in understanding the structure and relations in the data.

Principal Components Analysis (PCA)

Cons:

- **Loss of Interpretability**: The principal components are linear combinations of the original variables, which may result in loss of interpretability.
- **Assumes Linearity**: Assumes that the data structure is linear, which might not always be the case.
- **Sensitive to Scaling**: PCA is sensitive to the scaling of variables, so data needs to be properly scaled (like standardization) before applying PCA.

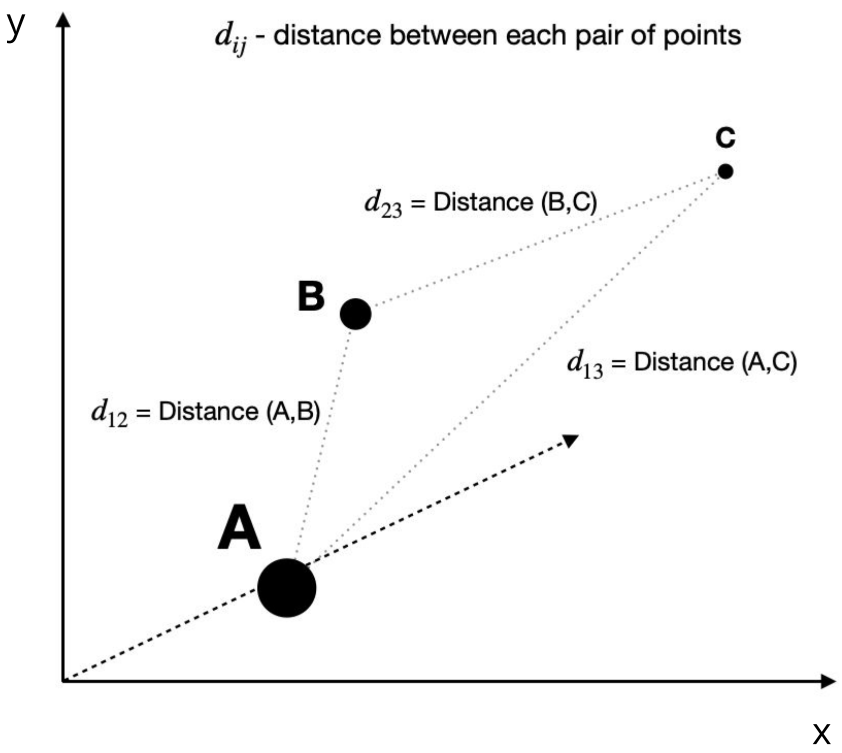
Demo with notebook

02__pca_reduction.ipynb

MDS

Steps used by MDS algorithm

1. The algorithm calculates **distances between each pair of points**.

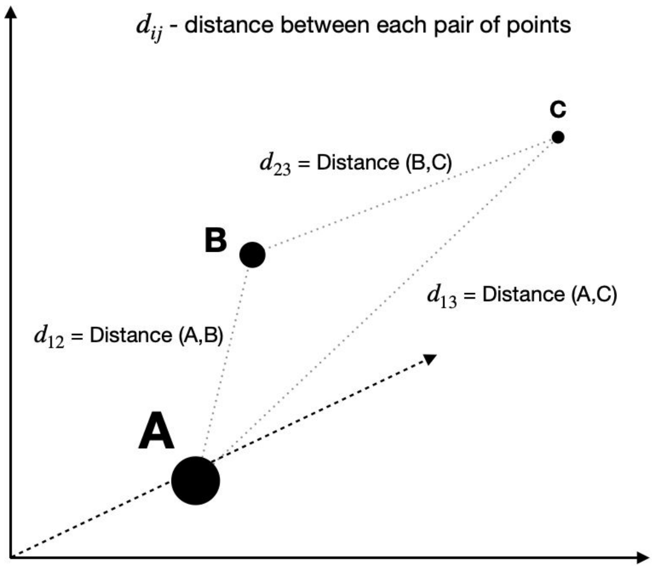


MDS

Steps used by MDS algorithm

1. The algorithm calculates **distances between each pair of points**.

2. With the original distances known, the algorithm attempts to **solve an optimization problem** by finding a set of coordinates in a lower-dimensional space that minimizes the value of Stress



$$Stress_D(x_1, x_2, \ldots, x_N) = \sqrt{\sum_{i \neq j = 1,\ldots,N} \left(d_{ij} - ||x_i - x_j||\right)^2}$$

The goal of the algorithm is to minimize the value of stress.

Where $x_1, \ldots, x_N$ are data points with their new set of coordinates in lower dimensional space.

d_{ij} is the actual distance we have calculated between the two corresponding data points in their original dimensional space.

$||x_i - x_j||$ is the distance between the two corresponding data points in their lower dimensional space.

The closer the value of $||x_i - x_j||$ is to d_{ij} the smaller will be the value of stress.

Demo with notebook

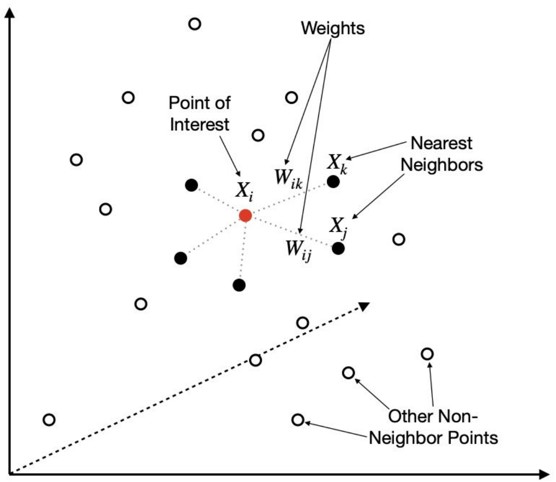
03__mds_reduction.ipynb

LLE

LLE is a **nonlinear technique** that attempts to **preserve local neighborhood relationships** in the lower-dimensional space.

It assumes that each data point and its **neighbors lie on or close to a locally linear patch** of manifold and tries to maintain these local linear relations in the reduced space.

Original High-Dimensional Space



We know the position of the Point of Interest

We know the positions of all the Nearest Neighbors

$$E(W) = \sum_i |X_i - \sum_j W_{ij} X_j|^2$$

The cost function is solved to find the weights, where the sum of weights for each X_i is set to equal to 1

$$\sum_j W_{ij} = 1$$

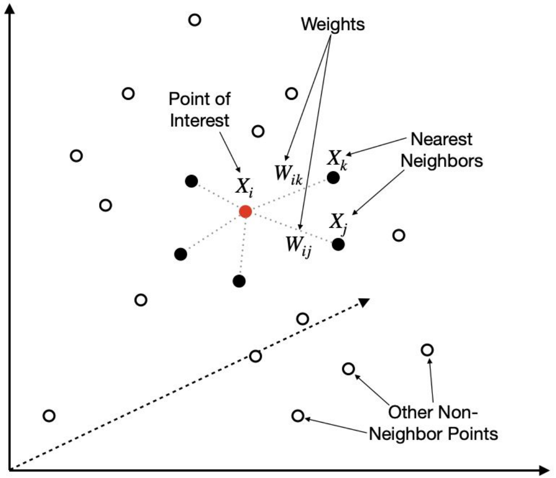
LLE

Steps used by LLE algorithm

1. Use a KNN approach to **find the k nearest neighbors** of every data point. Here, “k” is an arbitrary number of neighbors that you can specify within model hyperparameters.

2. **Construct a weight matrix** where every point has its weights determined by minimizing the error of a cost function E. Every point is a linear combination of its neighbors, which means that **weights for non-neighbors are 0**.

Original High-Dimensional Space



We know the position of the Point of Interest

We know the positions of all the Nearest Neighbors

$$E(W) = \sum_i |X_i - \sum_j W_{ij} X_j|^2$$

The cost function is solved to find the weights, where the sum of weights for each X_i is set to equal to 1

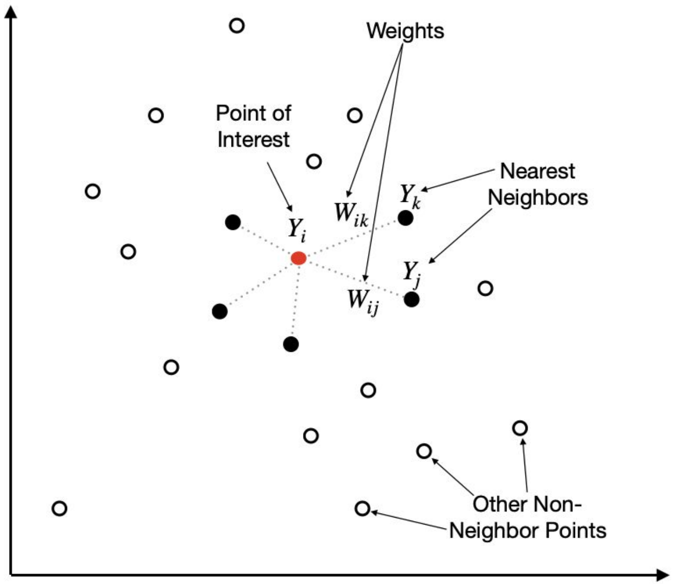
$$\sum_j W_{ij} = 1$$

LLE

Steps used by LLE algorithm

3. **Find the positions** of all the points in the **new lower-dimensional embedding** by minimizing the cost function C. Here we use weights (W) from step 2 and solve for Y.

New Lower-Dimensional Space



We know the weights from the previous step

$$C(Y) = \sum_i |Y_i - \sum_j W_{ij} Y_j|^2$$

The cost function is solved to find the positions of Y_i and its neighbors in the new lower-dimensional space using weights from the previous step.

Demo with notebook

04__lle_reduction.ipynb

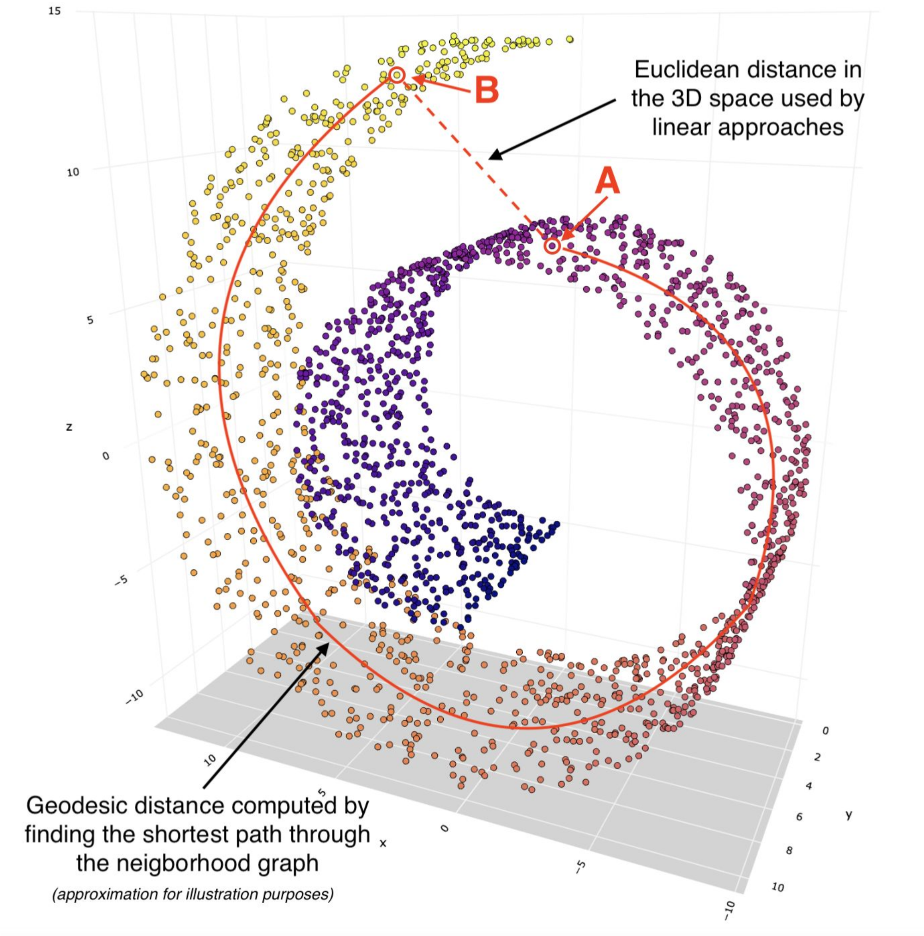
Isomap

Steps used by Isomap algorithm

1. Use a KNN approach to **find the k nearest neighbors** of every data point. Here, “k” is an arbitrary number of neighbors that you can specify within model hyperparameters.

2. Once the neighbors are found, **construct the neighborhood graph** where points are connected to each other if they are each other’s neighbors. Data points that are not neighbors remain unconnected.

(Similar to LLE)

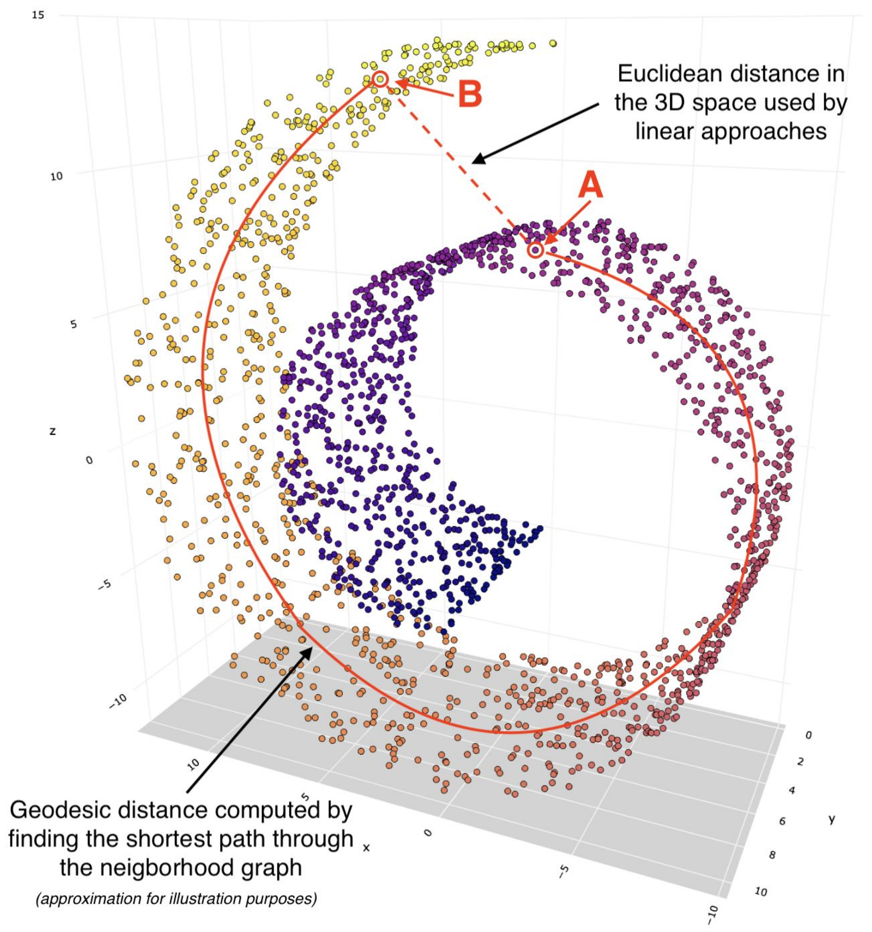


Isomap

Steps used by Isomap algorithm

3. **Compute the shortest path** between each pair of data points (nodes) (Dijkstra's algorithm). Note, this step is also commonly described as finding a **geodesic distance** between points.

4. **Use MDS to compute lower-dimensional embedding**. Given distances between each pair of points are known, MDS places each object into the N-dimensional space (hyperparameter) such that the between-point distances are preserved as well as possible.



Demo with notebook

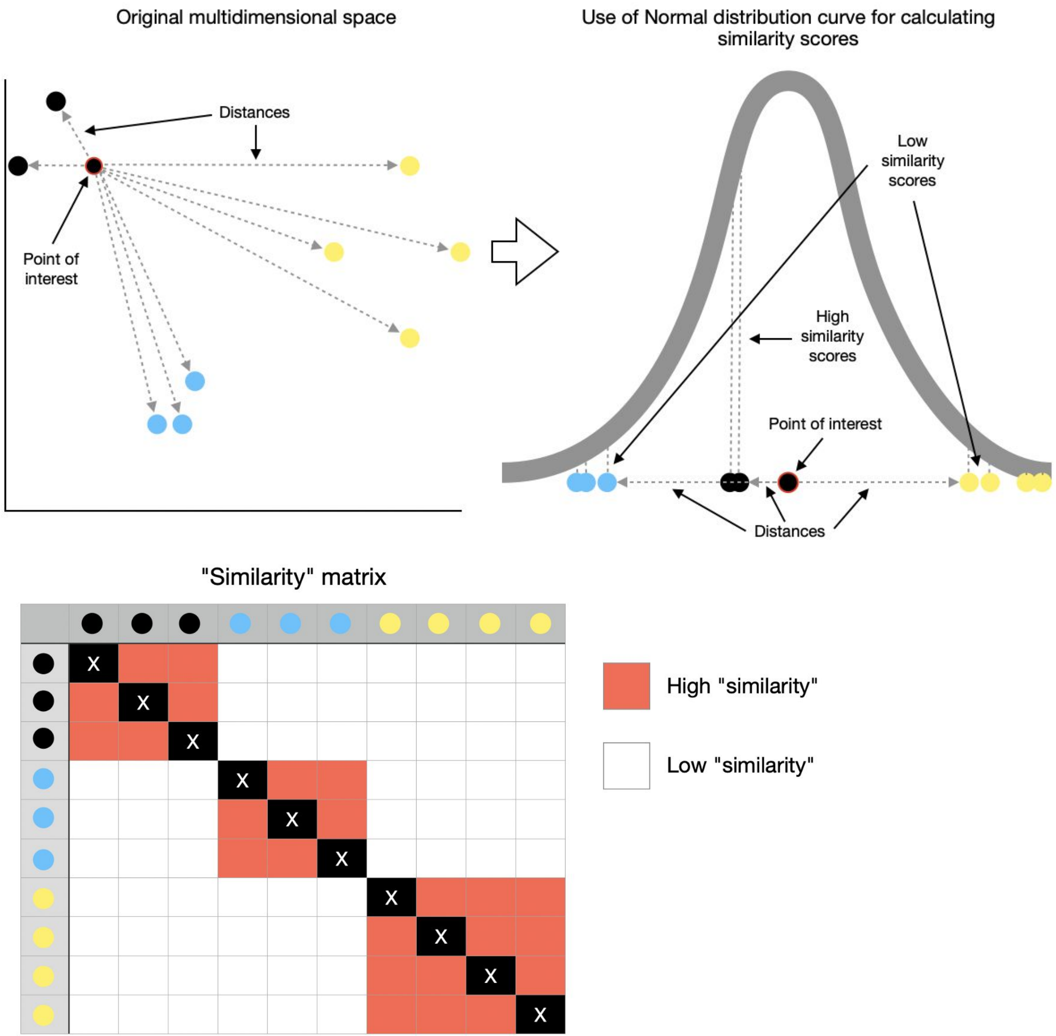
05__isomap_reduction.ipynb

T-SNE

Steps used by t-sne algorithm

1. Determine **the “similarity” of points based on distances between them.** Nearby points are considered “similar,” while distant ones are considered “dissimilar.”

Measuring distances between the point of interest and other points and then **placing them on a Normal curve**. It does this for every point, applying some scaling to account for variations in the density of different regions.

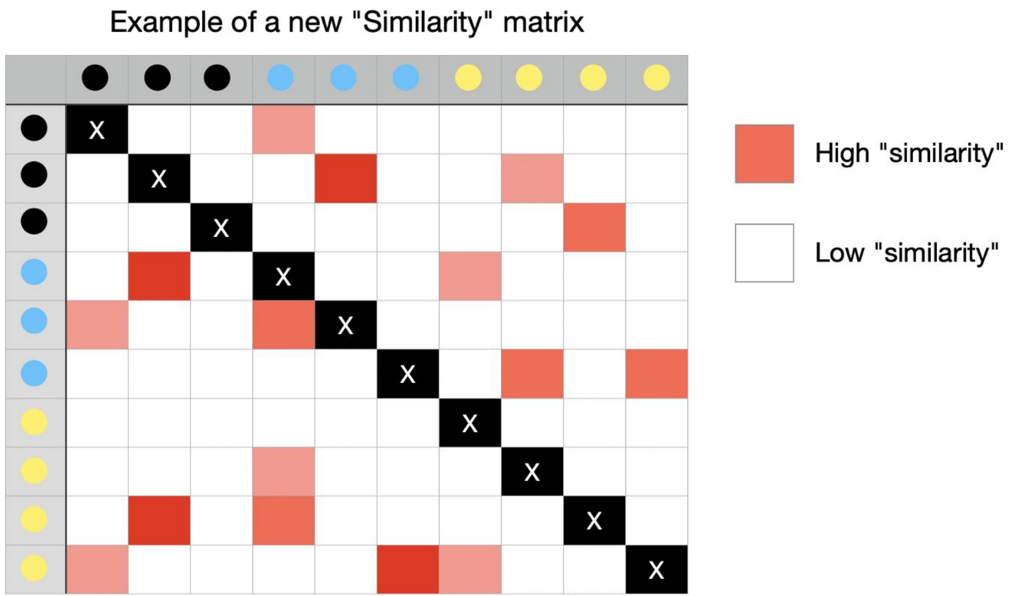


T-SNE

Steps used by t-sne algorithm

2. Randomly **map all the points onto a lower-dimensional space** and calculates “similarities” between points. One difference, though, this time, the algorithm uses **t-distribution** instead of Normal distribution.

3. Make the new “similarity” matrix look like the original one by using an iterative approach (Kullback-Leibler). With each iteration, points **move towards their “closest neighbors”** from the original higher-dimensional space and away from the distant ones.

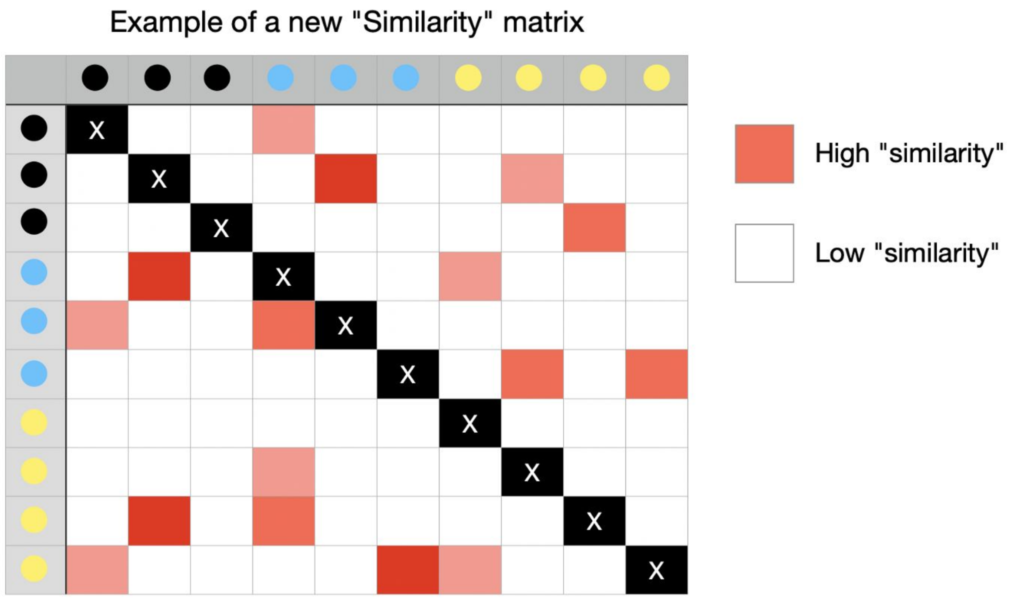


T-SNE

Steps used by t-sne algorithm

2. Randomly **map all the points onto a lower-dimensional space** and calculates “similarities” between points. One difference, though, this time, the algorithm uses **t-distribution** instead of Normal distribution.

3. Make the new “similarity” matrix look like the original one by using an iterative approach (Kullback-Leibler). With each iteration, points **move towards their “closest neighbors”** from the original higher-dimensional space and away from the distant ones.



https://distill.pub/2016/misread-tsne/

Demo with notebook

06__tsne_reduction.ipynb