Analyse et manipulation des données

*DigitalLab@LaPlataforme_*

Tools for data pre-processing

- Descriptive and inferential **statistics tools**
    - Univariate and multivariate analysis

Tools for data pre-processing

- Descriptive and inferential **statistics tools**
  - Univariate and multivariate analysis
- **Data transformations**: indexing, grouping and aggregation

Tools for data pre-processing

- Descriptive and inferential **statistics tools**
  - Univariate and multivariate analysis
- **Data transformations**: indexing, grouping and aggregation
- **Feature Selection**

Tools for data pre-processing

- Descriptive and inferential **statistics tools**
  - Univariate and multivariate analysis
- **Data transformations**: indexing, grouping and aggregation
- **Feature Selection**
- **Combination** of data sets

Tools for data pre-processing

- Descriptive and inferential **statistics tools**
  - Univariate and multivariate analysis
- **Data transformations**: indexing, grouping and aggregation
- **Feature Selection**
- **Combination** of data sets
- **Encoding** of categorical variables

Tools for data pre-processing

- Descriptive and inferential **statistics tools**
  - Univariate and multivariate analysis
- **Data transformations**: indexing, grouping and aggregation
- **Feature Selection**
- **Combination** of data sets
- **Encoding** of categorical variables
- Dimensionality reduction with **PCA**

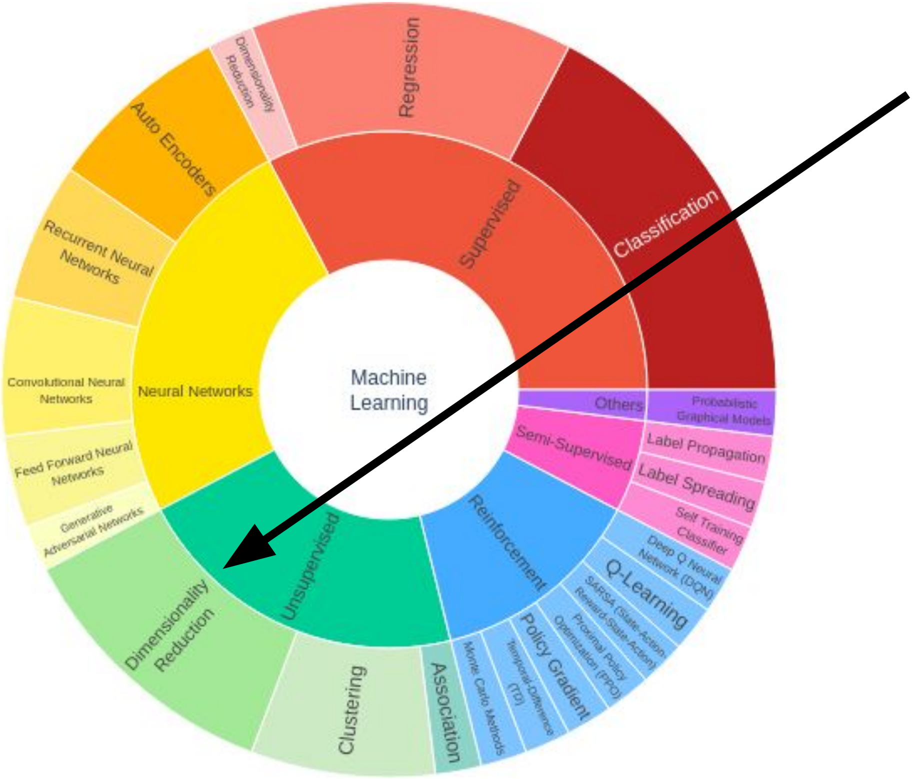
Tools for data pre-processing

- Descriptive and inferential **statistics tools**
  - Univariate and multivariate analysis
- **Data transformations**: indexing, grouping and aggregation
- **Feature Selection**
- **Combination** of data sets
- **Encoding** of categorical variables
- Dimensionality reduction with **PCA**
- Dimensionality reduction with **LDA**

Today we add

- Dimensionality reduction with **MDS**
- Dimensionality reduction with **Isomap**
- Dimensionality reduction with **LLE**
- Dimensionality reduction with **t-sne**

Where are these tools located?



MDS, Isomap, LLE, t-sne, PCA

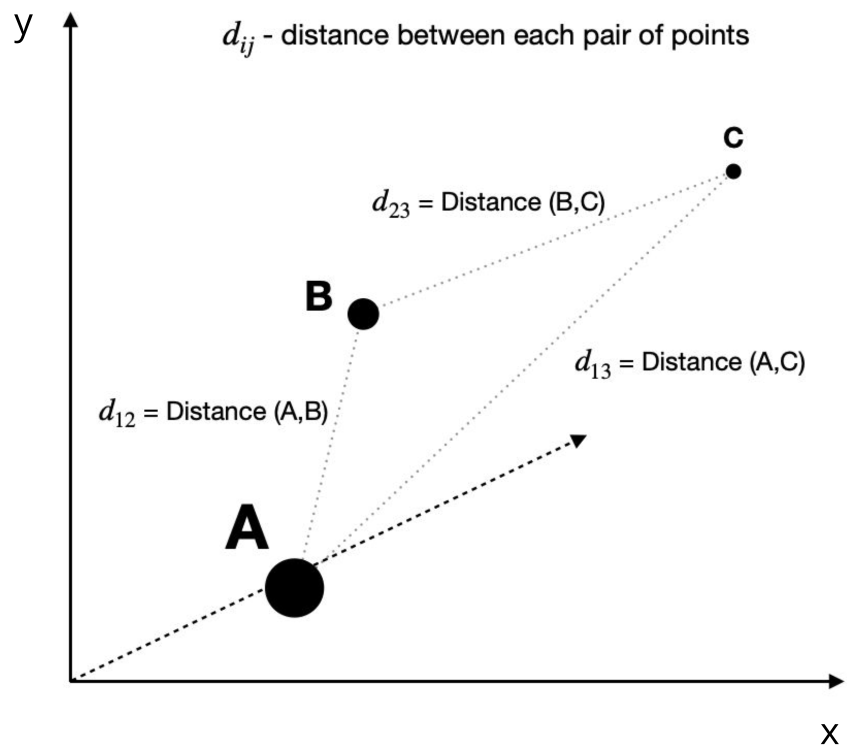
They are **unsupervised machine learning methods**.

Usually unsupervised methods are **used for visualization or encodings** depending on the problem you are trying to solve.

MDS

Steps used by MDS algorithm

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

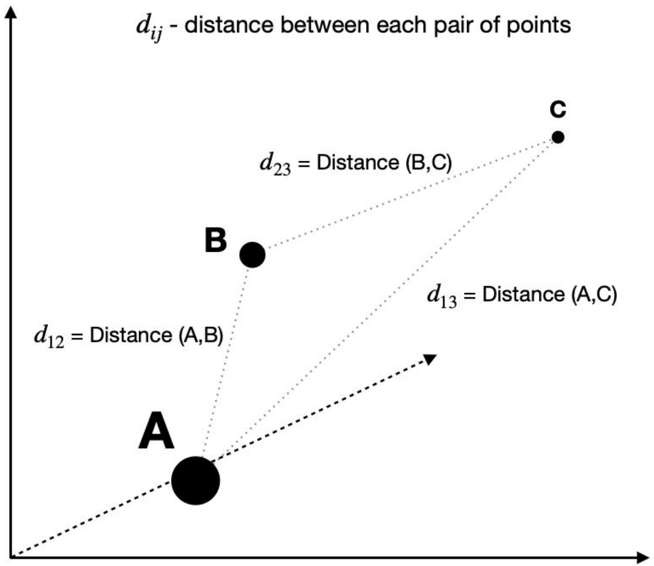


MDS

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 notebook

06_MDS.ipynb

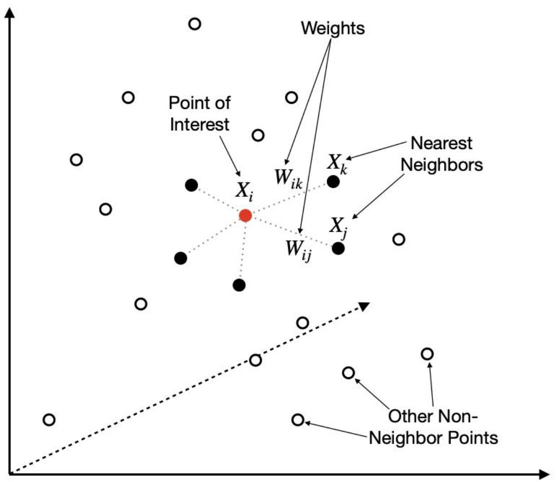
LLE

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

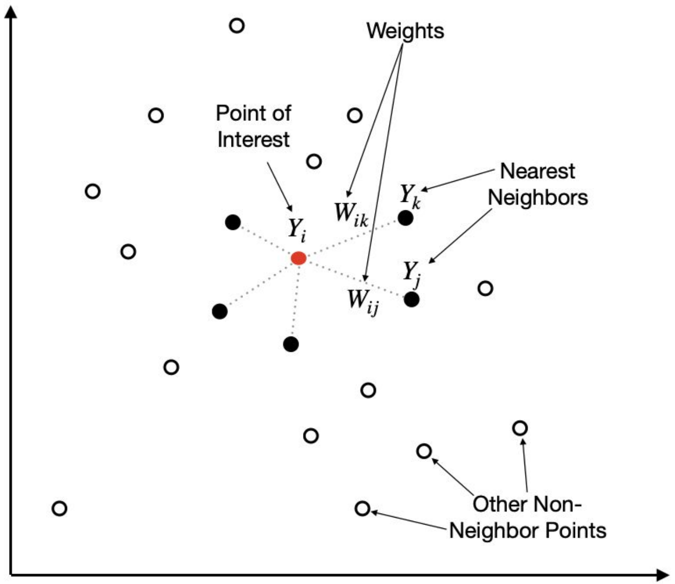
$$\sum_j W_{ij} = 1$$

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 notebook

07_LLE.ipynb

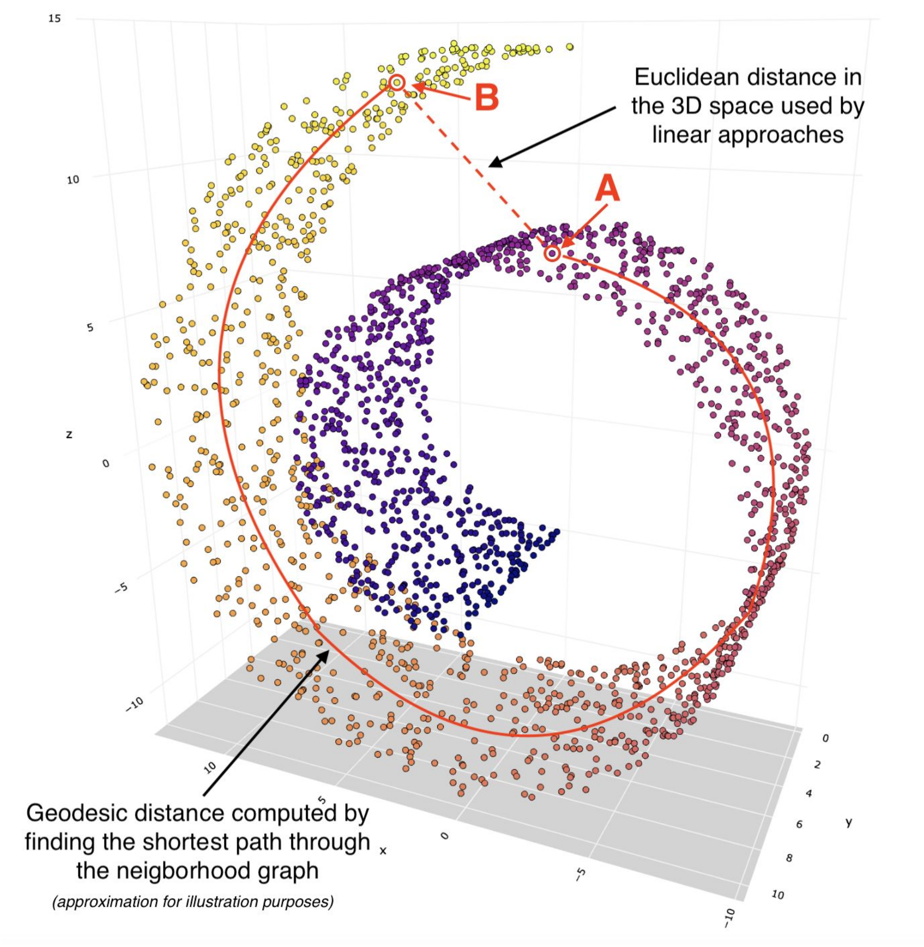
Isomap

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)

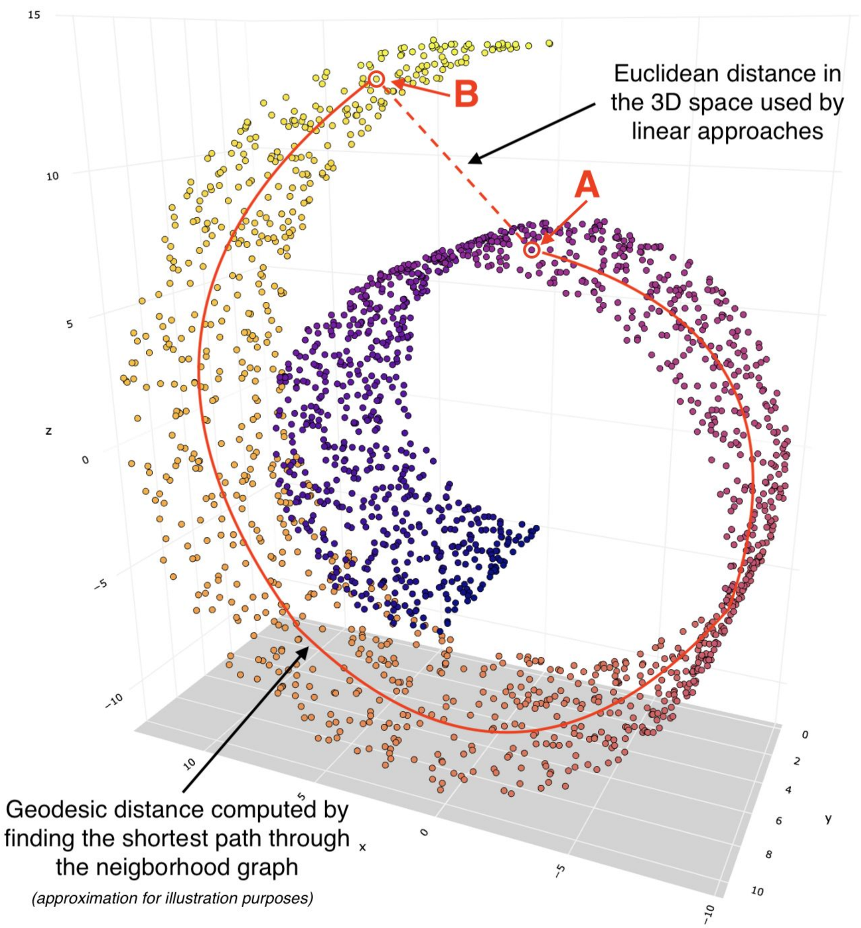


Isomap

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 notebook

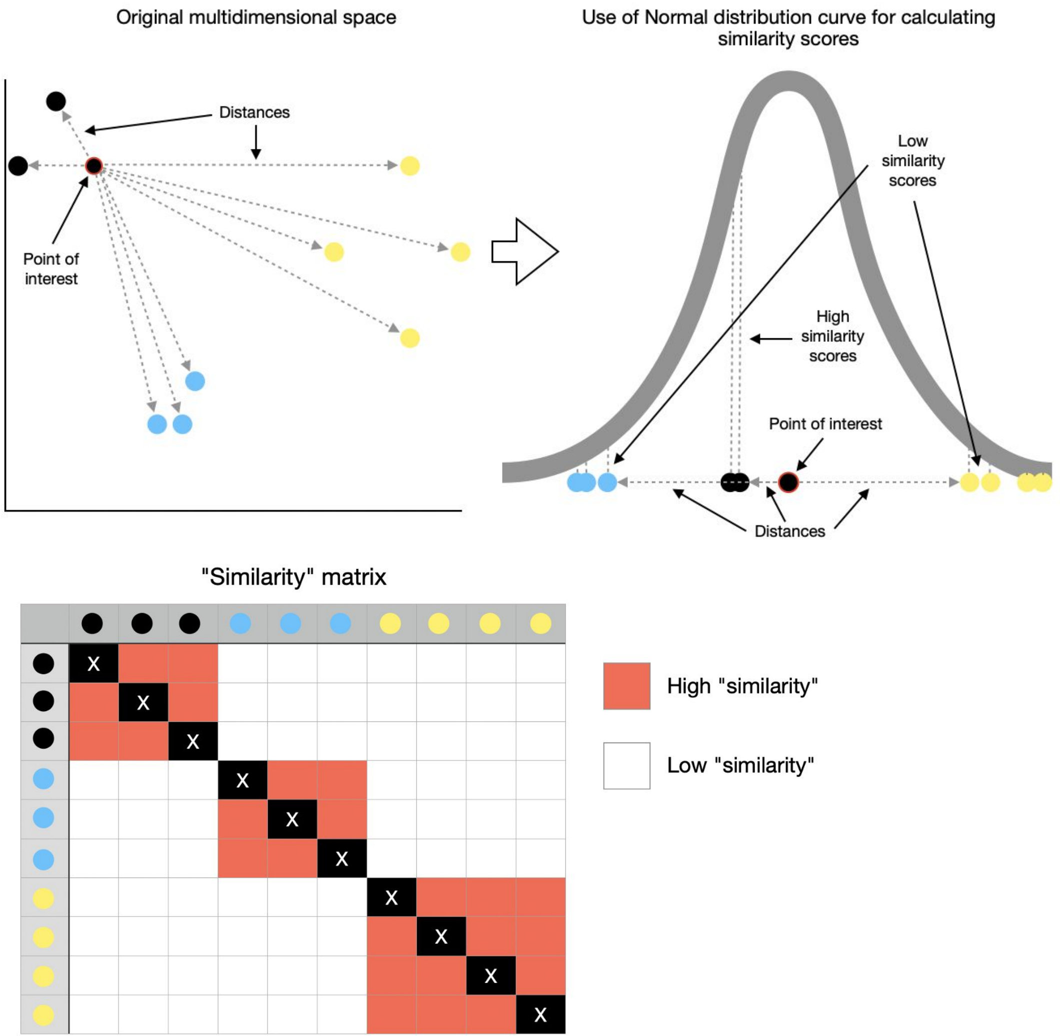
08_Isomap.ipynb

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.



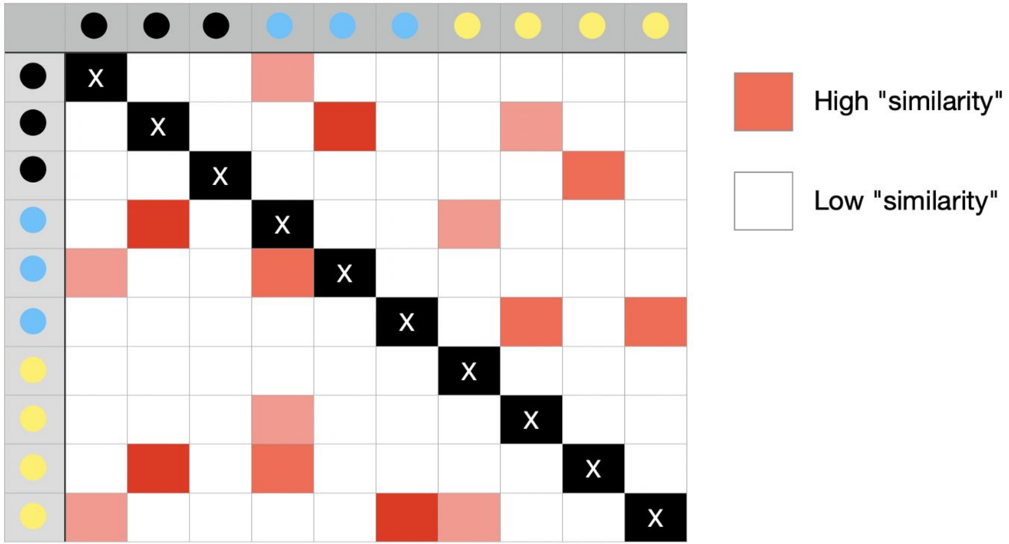
t-sne

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. With each iteration, points **move towards their “closest neighbors”** from the original higher-dimensional space and away from the distant ones.

Example of a new "Similarity" matrix



Demo notebook

09_tsne.ipynb