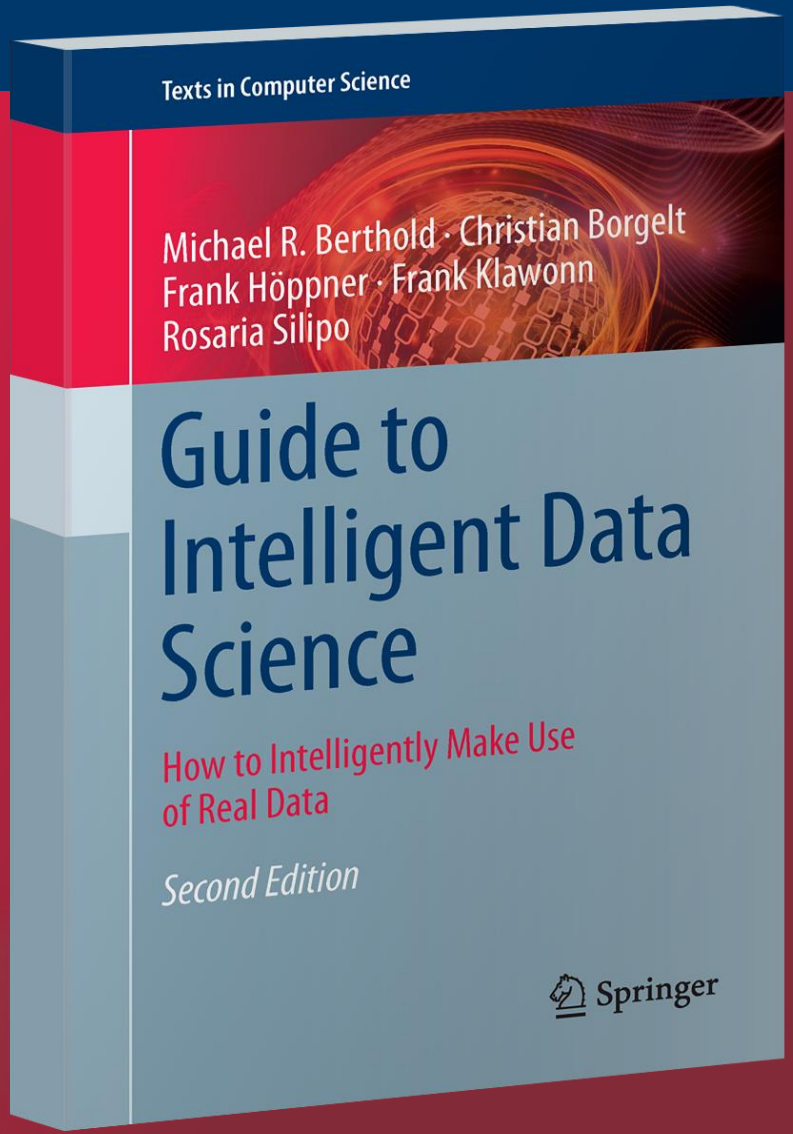
Dimensionality Reduction



Caption

Summary of this lesson

„The more you know, the less you need“
-Yvon Chouinard

Can we condense information?

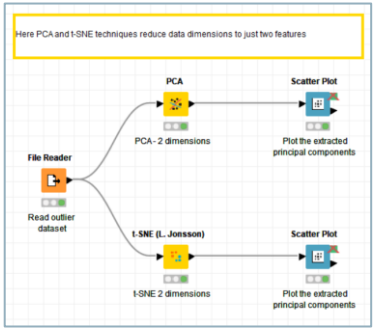
**This lesson refers to chapter 4 of the GIDS book*

Content of this lesson

- **Methods for Dimensionality Reduction**
- Principal Component Analysis (PCA)
- Linear discriminant Analysis (LDA)
- Multidimensional Scaling (MDS)
- t-distributed Stochastic Neighbor Embedding (t-SNE)

Datasets

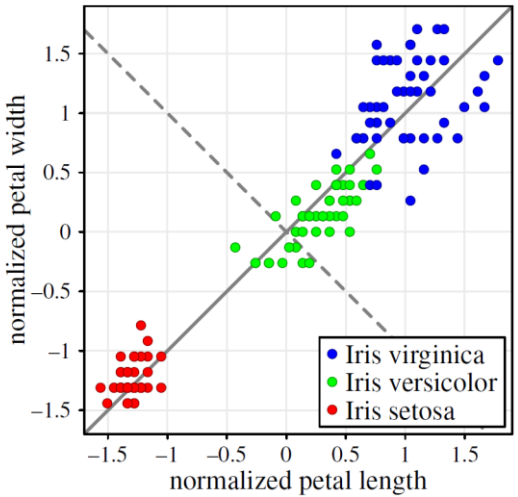
- Datasets used : adult dataset and outliers dataset
- Example Workflows:
  - „Dimensionality Reduction“ https://kni.me/w/EnmfIBCuLOFpYvc2
    - PCA
    - t-SNE



PCA: Principal Component Analysis

Principal Component Analysis: Goal

- Method from statistics to construct a **projection** from the high-dimensional space to a lower-dimensional space
- Uses the **variance** in the data as structure preservation criterion
- Projection must be **to a linear subspace** which preserves as much as possible of the original **variance** of the data



First PC (solid line) in the direction of maximum variance; second PC (dashed line) in the direction of second max. variance and orthogonal to first PC.

Mean and Variance

- (Sample) variance for one numerical attribute:

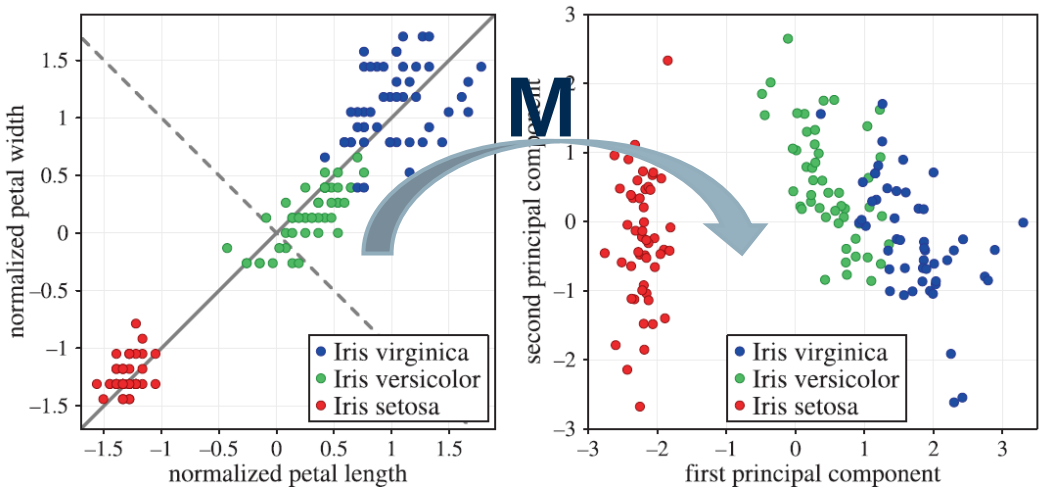
$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2$$

- Where x_i is the i-th sample in the dataset, n the number of samples, and $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the sample mean.
- Variance of a multidimensional data set = Sum of the variances of the attributes
- Covariance Matrix:

$$C = \frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{x}_i - \overline{\boldsymbol{x}})\,(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^T$$

Where $\boldsymbol{x}_i$ is the i-th sample vector in the dataset, n the number of samples, and $\overline{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$ is the mean vector.

Principal Component Analysis: Projection



- The projection can be represented by a matrix $\boldsymbol{M}_{m\times m}$ mapping the data points to the plane by:

$$\boldsymbol{y} = \boldsymbol{M} \cdot (\boldsymbol{x} - \overline{\boldsymbol{x}})$$

- Where $\overline{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the sample mean of $\boldsymbol{x}$, i.e. the vector of m mean values, n the number of data, m the data dimensionality, and ...

- … and $\boldsymbol{M} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m)$
- $\boldsymbol{v}_j$ is the j-th ***principal component***, that is the vector in the direction of the j-th max. variance of the dataset
- With constraints:
  - $\boldsymbol{v}_k \perp \boldsymbol{v}_j \quad for\ k, j = 1, .., m \quad and\ k \neq j$
  - $\|v_i\| = 1$

Solution:

- $\boldsymbol{v}_j$ are the **eigenvectors** of the covariance matrix C of the dataset
- λ_j are the **eigenvalues** of the covariance matrix C of the dataset and the variance associated with each eigenvector

The projection matrix $\boldsymbol{M}_{m\times m}$ is given by $\boldsymbol{M} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$

where the principal components $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ are the normalized eigenvectors of the covariance matrix C of the data

$$C = \frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^T$$

Sorted by the corresponding eigenvalues $\lambda_1 \geq \ldots \geq \lambda_m$

Note.

- λ is called an eigenvalue of a matrix *A*, if there is a non-zero vector **v** such that *A***v** = **v** holds.
- The vector **v** is called eigenvector to the eigenvalue (as you learned in your maths lectures).

- If M moves the data from $\mathbb{R}^m$ to $\mathbb{R}^m$, there is no reduction in dimensionality.
- We look for $M: \mathbb{R}^m \Rightarrow \mathbb{R}^q$ with $q < m$.
- Total variance in the dataset $\lambda_1 + \ldots + \lambda_m$ with $\lambda_1 \geq \ldots \geq \lambda_m$.
- Let's preserve just a fraction of the total variance:

$$\frac{\lambda_1 + \ldots + \lambda_q}{\lambda_1 + \ldots + \lambda_m}$$

- If we use the top $\lambda_1 \geq \ldots \geq \lambda_q$ we lose only a bit of the original variance
- We project the data onto the first q principal components $(\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_q)$ corresponding to the eigenvalues $\lambda_1 \geq \ldots \geq \lambda_q$, thus preserving the given fraction of the data variance

Principal Component Analysis

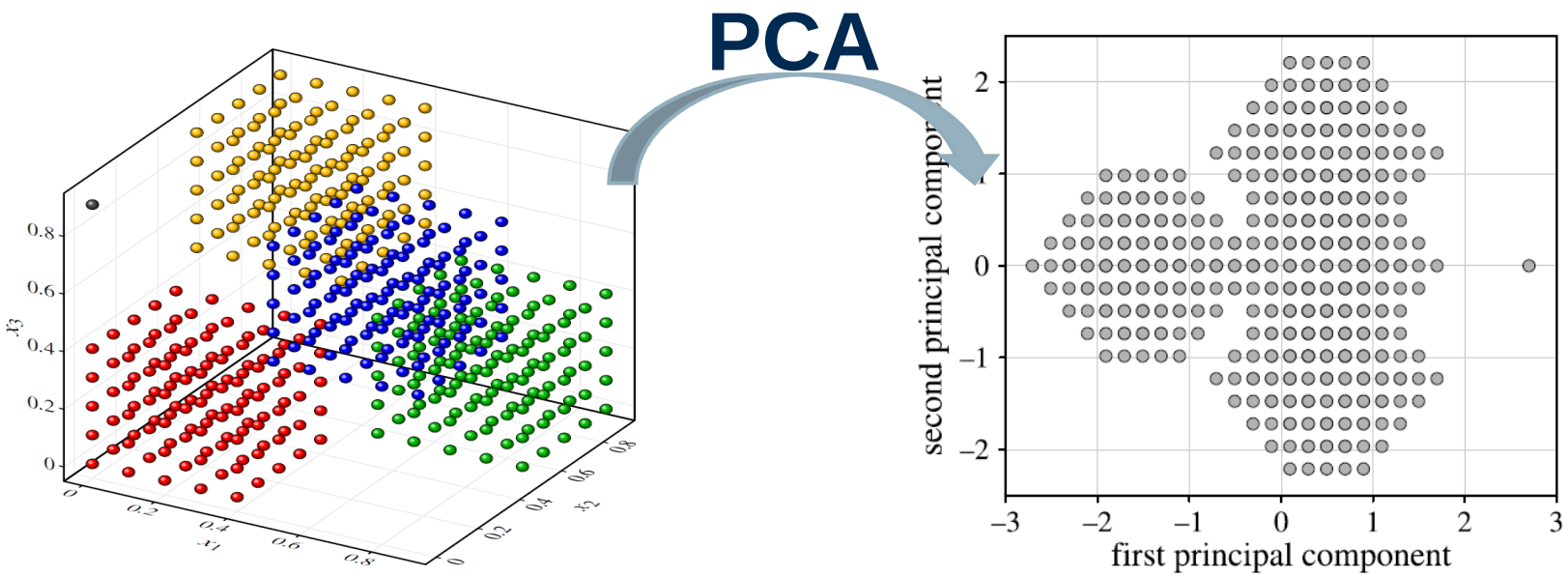
- PCA is a statistical procedure that **orthogonally** transforms the original *n* coordinates of a data set into a new set of *n* coordinates, called principal components
- We calculate the largest eigenvalue λ_1 and the corresponding eigenvector $\boldsymbol{v}_1$ of the covariance matrix C, as the direction of the largest variance in the dataset
- Each succeeding component $\boldsymbol{v}_k$ must follow the direction of the **next largest possible variance** under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding components
- v_1 describes most of the variability in the data, v_2 adds the next big contribution, and so on. In the end, the last v_s do not bring much more information to describe the data.
- Thus, to describe the data we use only the top $q < m$ (i.e., $v_1, v_2, \ldots, v_q$) components with little - if any - loss of information

	Prinicipal Components			
	PC1	**PC2**	**PC3**	**PC4**
Proportion of variance	0.73	0.229	0.0367	0.00518
Cumulative Proportion	0.73	0.958	0.9948	1.0000

Preservation of the variance of the Iris data set depending on the number of principal components.

- **Caveats:**
- Results of PCA are quite difficult to interpret
- Data normalization is required (z-score normalization)
- Only effective on numeric columns

Principal Component Analysis: the „Cube“ data



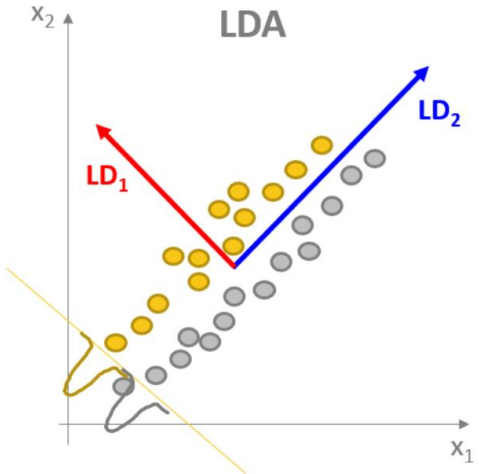
- Impression that the data are uniformly distributed over a grid
- Outlier not represented

- Data are visibly not uniformly distributed
- Outlier is visible

LDA: Linear Discriminant Analysis

Linear Discriminant Analysis: Goal

- LDA is a **classification algorithm** based on class statistics
- It can also be used as a method to construct a **projection** from the high-dimensional space to a lower-dimensional space
- Uses the class **separation** as structure preservation criterion
- Projection must be **to a linear subspace** which preserves as much as possible of the original **separation** of the classes in the data



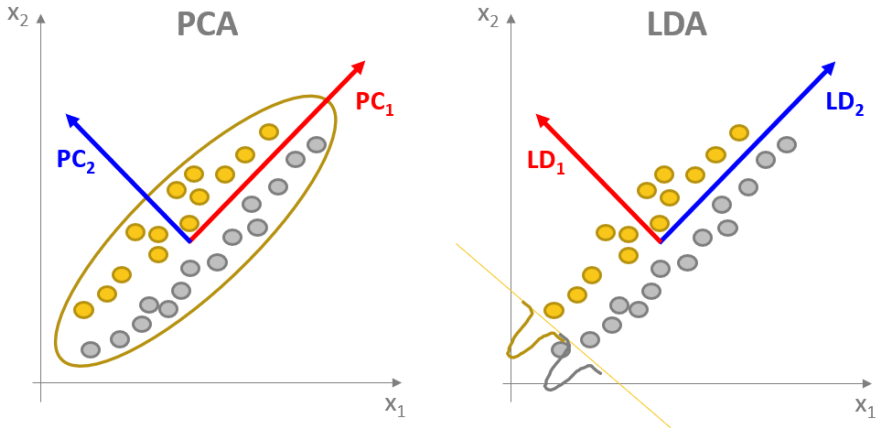
First LD (red line) in the direction of maximum discriminability between classes; second LD (blue line) in the direction of second max. discriminability and orthogonal to first LD.

- LD_1 describes best the class separation in the data, LD_2 adds the next big contribution, and so on. In the end, the last LD_s do not bring much more information to separate the classes.
- Thus, for our classification problem, we could use only the top $\boldsymbol{m < n}$ (i.e., $LD_1, LD_2, \ldots, LD_m$) discriminants with little - if any - loss of information for classification

- **Caveats (as for PCA)**:
- Results of LDA are quite difficult to interpret
- Normalization required
- Only effective on numeric columns

- LDA and PCA are both statistical procedures that **orthogonally** transform the original *n* coordinates of a data set into a new set of *n* coordinates, called linear discriminants (LD) or Principal Components (PC).
- Discriminants maximize the separation between classes, components maximize the variance in the data

- PCA: unsupervised
- LDA: supervised



MDS: Multidimensional Scaling

Multi-Dimensional Scaling

- Is not constructing an explicit mapping from the high-dimensional space into the low-dimensional space
- Only positions data points in the low-dimensional space
- Uses the **distance** between the high dimensional data points as structure preservation criterion (not the variance, not the class separability)

- Input: $(x_1, x_2, \ldots, x_n)$ with $x_i \in \mathbb{R}^m =: X$ (input space)
- Output: $(p_1, p_2, \ldots, p_n)$ with $p_i \in \mathbb{R}^q =: Y$ (output space, usually q= 2 or q=3)

Goal:

- Define a data point p_i for each data object $x_i$such that all distances $d_{ij}^{(Y)}$ between point p_i and all other points p_j in the output space Y are **roughly** the same as the corresponding distances $d_{ij}^{(X)}$ between the original data objects x_i and x_j in the input space X.

$$d_{ij}^{(Y)} = \|p_i - p_j\| \qquad \text{and} \qquad d_{ij}^{(X)} = \|x_i - x_j\|$$

- Distance function requirements:
  - Non negative $d_{ij} \geq 0$
  - Symmetric $d_{ij} = d_{ji}$
  - Positive definite $d_{ii} = 0$
- Usually Euclidean distance
- Starting Point: Distance matrix in input space

$$\left[d_{ij}^{(X)}\right]_{1 \leq i,j \leq n}$$

- Where $d_{ij}^{(X)}$ is the distance between data point x_i and data point x_j in the input space.
- Usually Euclidean distance in X after normalization

- **Objective functions (error measures)** to define the quality of a solution
- **Sum of squared errors**

$$E_0 = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(d_{ij}^{(Y)} - d_{ij}^{(X)} \right)^2$$

- **Normalised sum of squared errors**

$$E_1 = \frac{1}{\sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(d_{ij}^{(X)} \right)^2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(d_{ij}^{(Y)} - d_{ij}^{(X)} \right)^2$$

The normalization factor does not influence the location of the minimum of the objective function

- In contrast to E_0, the value of E_1 does neither depend on the number of data objects nor on the magnitude of the original distances

- **Relative Squared Error**

$$E_2 = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(\frac{d_{ij}^{(Y)} - d_{ij}^{(X)}}{d_{ij}^{(X)}} \right)^2$$

- Mixed absolute and relative error (also called **Stress**)

$$E_3 = \frac{1}{\sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{ij}^{(X)}} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(\frac{d_{ij}^{(Y)} - d_{ij}^{(X)}}{d_{ij}^{(X)}} \right)^2$$

- MDS based on E_3 is called **Sammon Mapping**

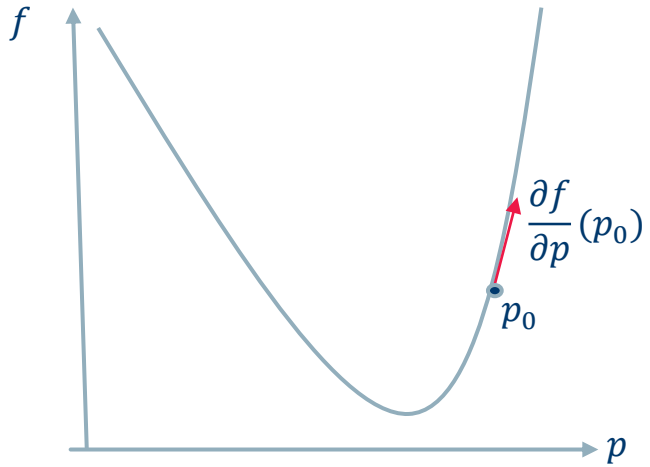
- **How to minimize the error measures?**
- No closed solution
- Non-linear optimization problem with $q \cdot n$ ($2n$ for $q = 2$) parameters to be optimized

- Even for a small dataset like the Iris dataset ($n = 150$), we need to optimize 300 parameters!
- Because we are searching for the best position of p_i in a two-dimensional space (x,y) for each one of the 150 data points

- Heuristic strategy needed ⇒ Typically a **gradient descent** method

The gradient, i.e., the vector of partial derivatives of the objective function with respect to the model parameters, points in the direction of steepest ascend.

- The gradient is the vector of partial derivatives: $[\frac{\partial f}{\partial p_1}, \frac{\partial f}{\partial p_2}, \frac{\partial f}{\partial p_3}, \ldots, \frac{\partial f}{\partial p_k}]$
- The gradient points in the direction of steepest ascend.



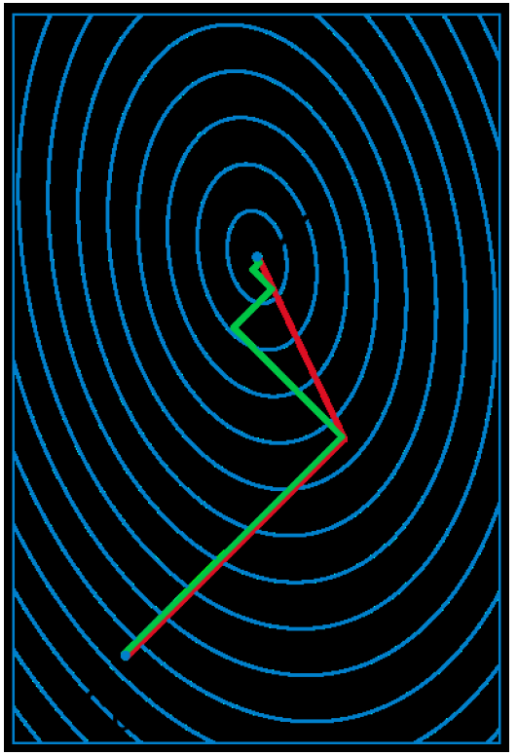
The gradient, i.e., the vector of partial derivatives of the error function with respect to the model parameters, points in the direction of steepest ascend.

1. Start at a random point $p_{i=0}$, i.e. an arbitrary choice of the model parameters
2. Calculate the gradient of the objective function $[\frac{\partial f}{\partial p_1}, \frac{\partial f}{\partial p_2}, \frac{\partial f}{\partial p_3}, \ldots, \frac{\partial f}{\partial p_k}]$
3. From p_i move a certain step in the opposite direction of the gradient (for an error function to minimize) and reach point p_{i+1}
4. Calculate the new value of the objective function in $\mathrm{f}(p_{i+1})$
5. Repeat from 2.

Procedure continues until no more improvements on the objective function $\mathrm{f}(p)$ can be achieved (step 4), a fixed number of gradient steps has been carried out, or p_{i+1} is too close to p_i.

Notes:

- f must be differentiable
- The landscape of the objective function (f vs. P) cannot be plotted already for $k > 2$
- The gradient descent procedure moves in steps along the objective function
- The step size: constant or adaptive?
- Problem of the local minima (maxima)



It is recommended to run a gradient method repeatedly, starting with different initial points to increase the chance to find the global or at least a good local optimum.

- Error measure $E_1 = \frac{1}{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left(d_{ij}^{(X)}\right)^2}\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left(d_{ij}^{(Y)} - d_{ij}^{(X)}\right)^2$
- Gradient of error measure E_1 with respect to one data point $\boldsymbol{y}_k$ in the output space

$$\frac{\partial E_1}{\partial \boldsymbol{y}_k} = \frac{2}{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left(d_{ij}^{(X)}\right)^2}\sum_{j\neq k}\left(d_{kj}^{(Y)} - d_{kj}^{(X)}\right)\frac{\boldsymbol{y}_k - \boldsymbol{y}_j}{d_{kj}^{(X)}}$$

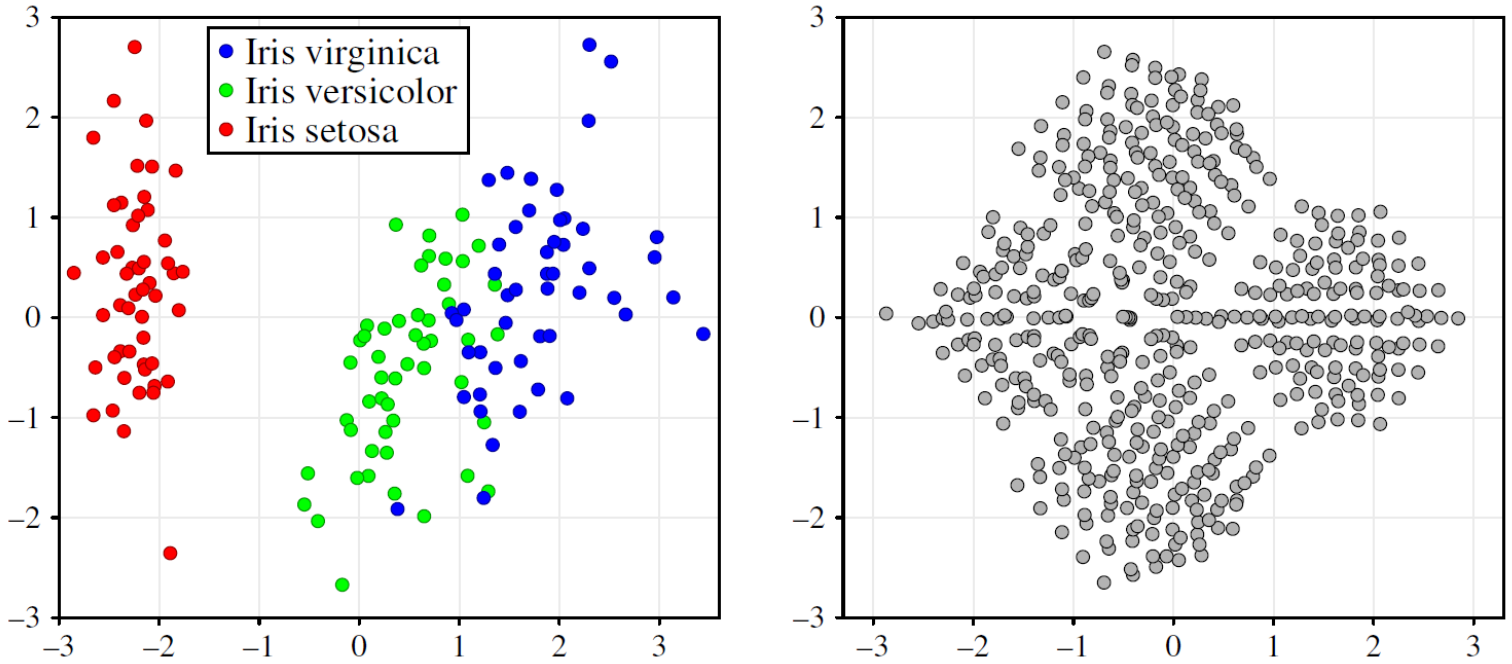
using $$\frac{\partial d_{ij}^{(Y)}}{\partial \boldsymbol{y}_k} = \frac{\partial}{\partial \boldsymbol{y}_k}\|\boldsymbol{y_i} - \boldsymbol{y_j}\| = \begin{cases} \frac{\boldsymbol{y_k} - \boldsymbol{y_j}}{d_{kj}(Y)} & if\ i = k \\ 0 & otherwise \end{cases}$$

Algorithm MDS($\mathcal{D}$)

input:	data set $\mathcal{D} \subset \mathbb{R}^m$ with $\|\mathcal{D}\| = n$ or distance matrix $[d_{i,j}^{(X)}]_{1 \leq i,j \leq n}$
parameter:	dimension q for the representation, stepwidth $\alpha > 0$, stop criterion SC
output:	set Y of n points in $\mathbb{R}^q$
1	Initialize $Y = \{\mathbf{y_1}, \ldots, \mathbf{y_n}\} \subset \mathbb{R}^q$ randomly or better with a PCA projection
2	compute the distances $d_{ij}^{(X)}$
3	**do**
4	Compute $d_{i,j}^{(Y)} = \| \mathbf{y}_i - \mathbf{y}_j \|$ (for all $i, j = 1, \ldots, n$)
5	Compute $\partial E_1 / \partial \mathbf{y_k}$ (for all $k = 1, \ldots, n$)
6	update $\mathbf{y_k}^{\text{new}} = \mathbf{y_k}^{\text{old}} - \alpha \cdot \partial E_1 / \partial \mathbf{y_k}$ (for all $k = 1, \ldots, n$)
7	**while** SC is not satisfied

- **Caveats**:
- Normalization before computing the distances
- Assure that no distance between two data objects is 0 before applying MDS algorithm

Multidimensional Scaling



MDS (Sammon Mapping) for the Iris and the „cube“ datasets

PCA vs. MDS vs. LDA

PCA

- Preserves the variance of the dataset
- Provides explicit mapping to the lower-dimensional space
  - easier insertion of new data
- Lower computational complexity
  - Covariance matrix can be calculated in linear time

MDS

- Preserves the distances between data objects
- Does not provide explicit mapping
  - New data cannot be projected
- Higher computational complexity
  - Quadratic for the pairwise distance
  - Complexity problems partially overcomed by sampling or variations of the error measures

LDA

- Preserves the separation among classes
- Provides explicit mapping to the lower-dimensional space
  - easier insertion of new data
- Lower computational complexity

t-SNE: t-distributed Stochastic Neighbor Embedding

t-distributed Stochastic Neighbor Embedding

- Nonlinear dimensionality reduction technique based on nonlinear relationships among the data points
- Primarily used for visualization from very high-dimensional data to 2 or 3 coordinates
- Works with probabilities of two data points being neighbours
- As for MDS, new data cannot be projected. All data must be used to find the new points.

In the input space:

- For any two data points $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$, probabilities are defined as:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$$

- where $p_{j|i}$ is the probability that object $\boldsymbol{x_i}$ picks object $\boldsymbol{x_j}$ as its neighbor

$$p_{j|i} = \frac{e^{-\frac{\left\|x_i - x_j\right\|^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{-\frac{\left\|x_i - x_k\right\|^2}{2\sigma_i^2}}}$$

- Parameter σ_i becomes smaller if density of points around $\boldsymbol{x_i}$ is high and vice versa

In the output space:

– The corresponding probabilities for points $\boldsymbol{y_i}$ and $\boldsymbol{y_j}$ are defined using t- or Cauchy distribution-like functions

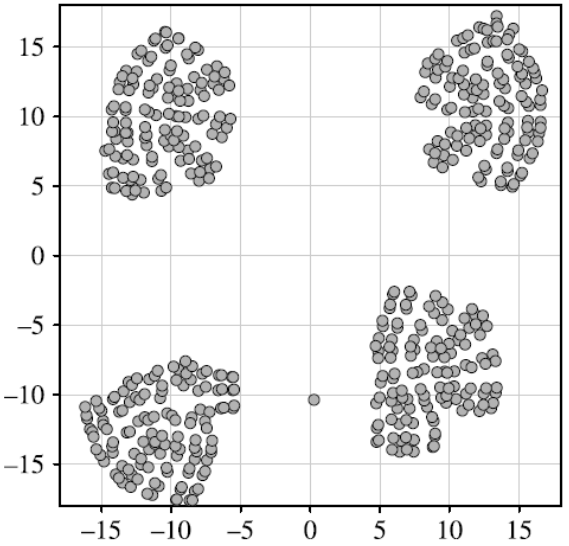
$$q_{i|j} = \frac{(1 + \|\boldsymbol{y_i} - \boldsymbol{y_j}\|^2)^{-1}}{\sum_{k \neq i}(1 + \|\boldsymbol{y_i} - \boldsymbol{y_k}\|^2)^{-1}}$$

– The points $\boldsymbol{y_i}$ are chosen in a way that the probabilities p_{ij} and q_{ij} are as similar as possible according to the **Kullback-Leibler divergence**

$$KL(p||q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

t-SNE is mainly used for visualization because:

- Aggressive dimensionality reduction
- Transformation of likelihood of pairs into visual proximity
- Capability to represent strange shapes of data

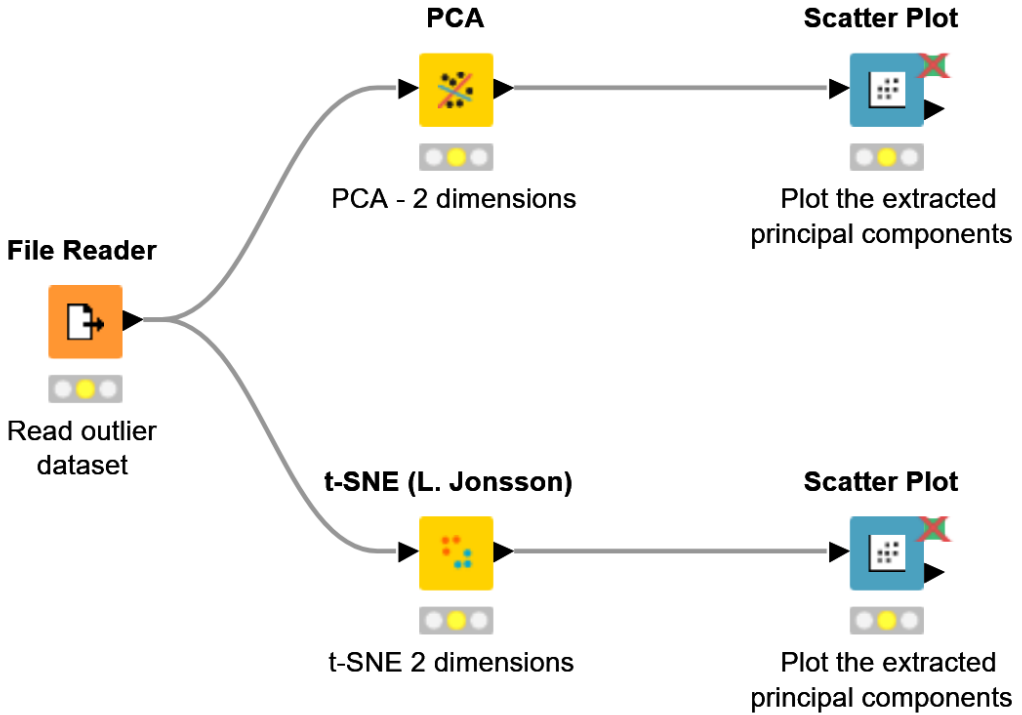


t-SNE applied to the „cube“ dataset. The four original clusters are well preserved in the two-dimensional representation and the outlier is also clearly visible.

Practical Examples

- PCA and t-SNE techniques for data reduction to 2D space

Here PCA and t-SNE techniques reduce data dimensions to just two features



For any questions please contact: email@email.com