Data Preprocessing

Javier Béjar

URL - 2024 Spring Term

CS - MAI

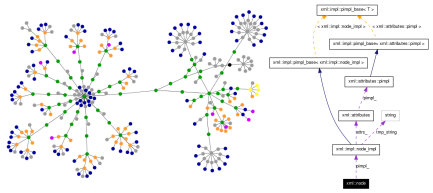
Introduction

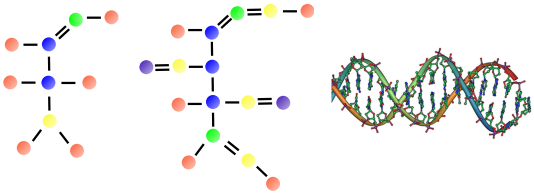
- **Unstructured datasets:**
  - Examples described by a flat set of attributes: attribute-value matrix
- **Structured datasets:**
  - Individual examples described by attributes but with relations among them: sequences (time, spatial...), trees, graphs
  - Sets of structured examples (sequences, graphs, trees)

- Only one table of observations
- Each example represents an instance of the problem
- Each instance is represented by a set of attributes (discrete, continuous)

A	B	C	⋯
1	3.1	a	⋯
1	5.7	b	⋯
0	-2.2	b	⋯
1	-9.0	c	⋯
0	0.3	d	⋯
1	2.1	a	⋯
⋮	⋮	⋮	⋱

- One sequential relation among instances (Time, Strings)
  - Several instances with internal structure
  - Subsequences of unstructured instances
  - One large instance
- Several relations among instances (graphs, trees)
  - Several instances with internal structure
  - One large instance





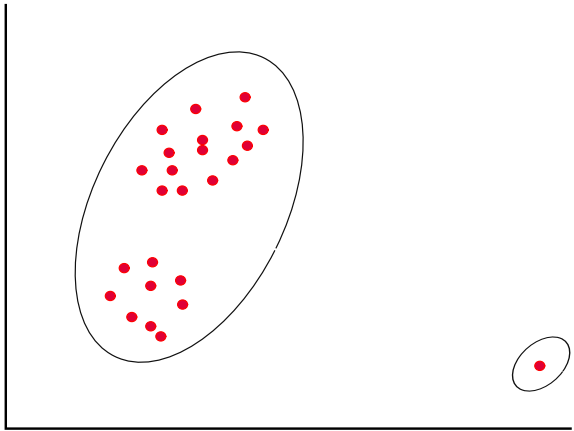
- Endless sequence of data
  - Several streams synchronized
  - Unstructured instances
  - Structured instances
- Static/Dynamic model

- Most unsupervised learning algorithms are specifically fitted for unstructured data
- The data representation is equivalent to a database table (attribute-value pairs)
- Specialized algorithms have been developed for structured data: Graph clustering, Sequence mining, Frequent substructures
- The representation of these types of data is sometimes algorithm dependent

Data Preprocessing

- Usually raw data is not directly adequate for analysis
- The usual reasons:
  - The quality of the data (noise/missing values/outliers)
  - The dimensionality of the data (too many attributes/too many examples)
- The first step of any data task is to assess the quality of the data
- The techniques used for data preprocessing are usually oriented to unstructured data

- **Outliers:** Examples with extreme values compared to the rest of the data
- Can be considered as examples with erroneous values
- Have an important impact on some algorithms

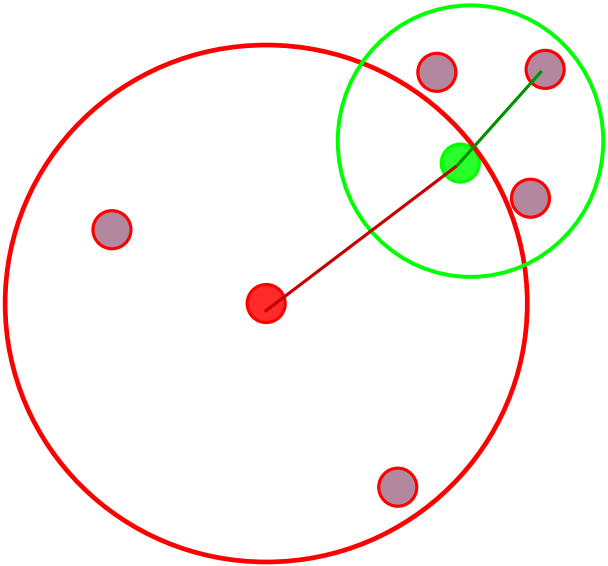


- The exceptional values could appear in all or only a few attributes
- The usual way to correct this problem is to eliminate the examples
- If the exceptional values are only in a few attributes these could be treated as missing values

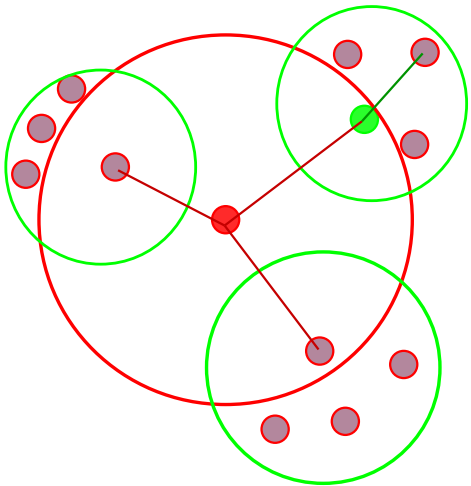
- Assumes a probabilistic distribution for the attributes
- **Univariate**
  - Perform Z-test or *student's* test
- **Multivariate**
  - Deviation method: Reduction in data variance when eliminated
  - Angle based: Variance of the angles to other examples
  - Distance based: Variation of the distance from the mean of the data in different dimensions

- Histogram based: Define a multidimensional grid and discard cells with low density
- Distance based: Distance of outliers to their k-nearest neighbors are larger
- Density based: Approximate data density using Kernel Density estimation or heuristic measures (Local Outlier Factor, LOF)

- LOF quantifies the outlierness of an example adjusting for variation in data density
- Uses the distance of the k-th neighbour $D_k(x)$ of an example and the set of examples that are inside this distance $L_k(x)$
- The **reachability distance** between two data points $R_k(x, y)$ is defined as the maximum between the distance $dist(x, y)$ and the y's k-th neighbour distance



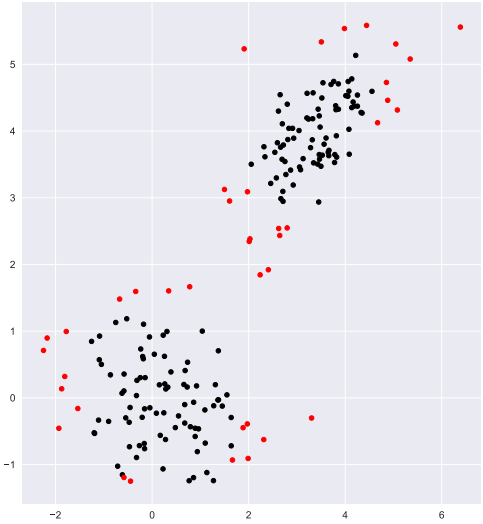
- The **average reachability distance** $AR_k(x)$ with respect of an example's neighbourhood ($L_k(x)$) is defined as the average of the reachability distances to the neighbourhood

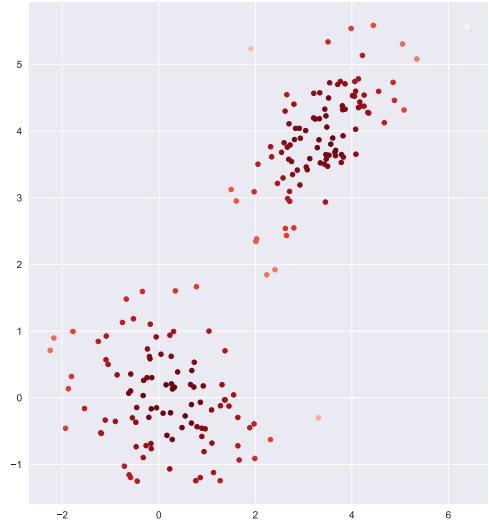


- The **LOF** of an example is computed as the mean ratio between $AR_k(x)$ and the average reachability of its k neighbors:

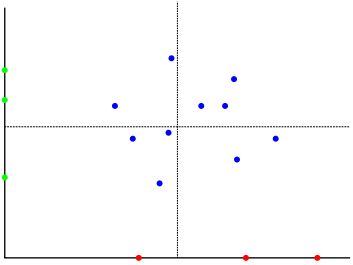
$$LOF_k(x) = \frac{1}{k} \sum_{y \in L_k(x)} \frac{AR_k(x)}{AR_k(y)}$$

- This value ranks all the examples

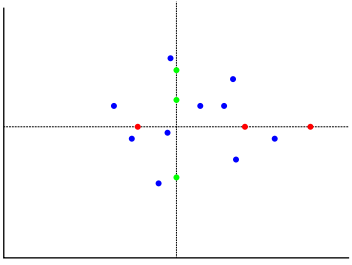

5
4
3
2
1
0
−1
−2
0
2
4
6


5
4
3
2
1
0
−1
−2
0
2
4
6

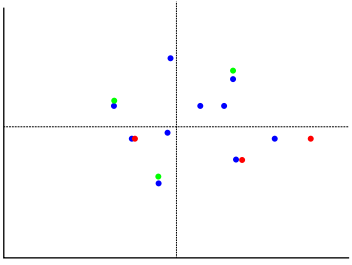
- Missing values appear because of errors or omissions during the gathering of the data
- They can be substituted to increase the quality of the dataset (value imputation)
  - Global constant for all the values
  - Mean or mode of the attribute (global central tendency)
  - Mean or mode of the attribute but only of the k-nearest examples (local central tendency)
  - Learn a model for the data (regression, bayesian) and use it to predict the values
- Problem: changes the statistical distribution of the data



Missing Values



Mean substitution



1-neighbor substitution

- Normalizations are applied to quantitative attributes in order to obtain attributes with similar behaviours
- Possible transformations
  - Rescale the values, so they have the same range
  - Change the moments of the data distributions, so all the attributes have the same
  - Change the data distribution to a well-behaved probability distribution using non-linear transformations

- **Range Normalization:** Transform all the values of the attribute to a preestablished scale (e.g.: [0,1], [-1,1])

$$\frac{x - x_{min}}{x_{max} - x_{min}}$$

- **Standard Score normalization:** Transform the data assumed gaussian distributed to $\mathcal{N}(0, 1)$

$$\frac{x - \mu_x}{\sigma_x}$$

- **Robust Scaling:** Transform using the median and the 50% interquartile range to reduce the effect of possible outliers

- **Quantile Transformation:** Compute a quantized cumulative distribution function of the attribute and transform it using the inverse of the theoretical quantile function of the target distribution
- **Power Transformation:** Apply a non-linear monotonic transformation that maps the values of the feature closer to a gaussian distribution (Box-Cox, Yeo-Johnson)

$$x(\lambda) = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \lambda \neq 0 \\ \log(x) + \sigma^2 & \lambda = 0 \end{cases}$$

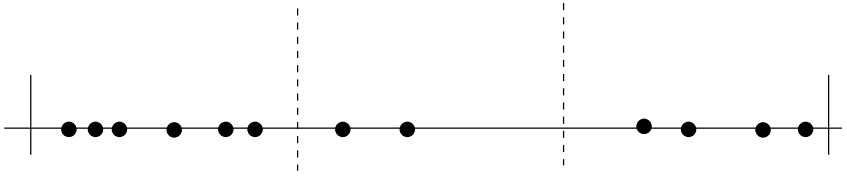
$$x(\lambda) = \begin{cases} \frac{(x+1)^\lambda - 1}{\lambda} & \lambda \neq 0, x \geq 0 \\ \log(x+1) + \sigma^2 & \lambda = 0, x \geq 0 \\ -\frac{(x+1)^{2-\lambda} - 1}{2-\lambda} & \lambda \neq 2, x < 0 \\ -\log(-x+1) + \sigma^2 & \lambda = 0, x < 0 \end{cases}$$

Discretization allows transforming quantitative attributes to qualitative attributes

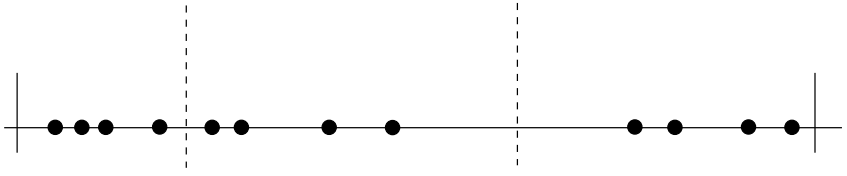
- **Equal size bins:** Pick the number of values and divide the range of data in equal sized bins
- **Equal frequency bins:** Pick the number of values and divide the range of data so each bin has the same number of examples (the size of the intervals will be different)

Discretization allows transforming quantitative attributes to qualitative attributes

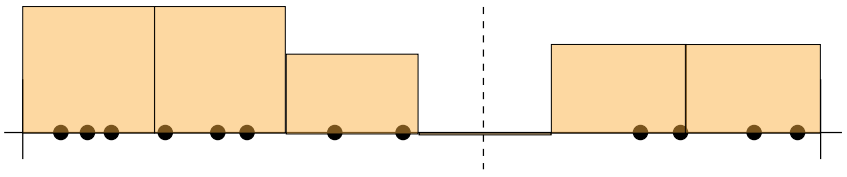
- **Distribution approximation:** Calculate a histogram of the data and fit a kernel function (KDE), the intervals are where the function has its minima
- **Other techniques:** Apply entropy based measures, Minimum Description Length (MDL), clustering



Same size



Same Frequency



Histogram

These two Python Notebooks show some examples of the effect of missing values imputation and data discretization and normalization

- Outliers/Missing Values Notebook (click here to open the notebook in colab)
- Normalization/Discretization Notebook (click here to open the notebook in colab)

If you download the notebook you will be able to use it locally (run jupyter notebook to open the notebooks)

Dimensionality Reduction

- Problems due to the dimensionality of data
  - The computational cost of processing the data
  - The quality of the data
- Elements that define the dimensionality of data
  - The number of examples
  - The number of attributes
- Usually the problem of having too many examples can be solved using sampling.

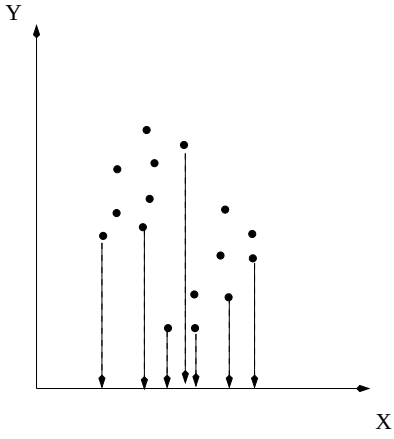
- The number of attributes has an impact on the performance:
  - Poor scalability, computational cost is a function of the number of attributes
  - Inability to cope with irrelevant/noisy/redundant attributes
  - As dimensionality increases similarity/distances become almost meaningless
- Methodologies to reduce the number of attributes:
  - **Dimensionality reduction**: Transforming to a space of less dimensions
  - **Feature subset selection**: Eliminating not relevant attributes

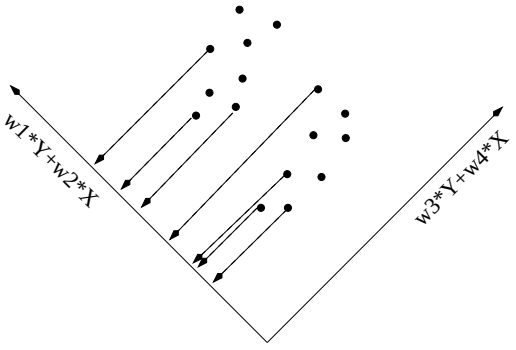
- New dataset that preserves most of the information of the original data but with less attributes
- Many techniques have been developed for this purpose
  - Projection to a space that preserves the statistical distribution of the data (PCA, ICA)
  - Projection to a space that preserves distances among the data (Multidimensional scaling, random projection, nonlinear scaling)

Linear methods

- **Principal Component Analysis:**
  - Data is projected onto a set of orthogonal dimensions (components) that are a linear combination of the original attributes
  - The components are uncorrelated and are ordered by the information they have
  - We assume data follows gaussian distribution
  - Global variance is preserved

Computes a projection matrix where the dimensions are orthogonal (linearly independent) and data variance is preserved





- Principal components: vectors that are the best linear approximation of the data

$$f(\lambda) = \mu + V_q \lambda$$

μ is a location vector in $\mathbb{R}^p$, V_q is a $p \times q$ matrix of q orthogonal unit vectors and λ is a q vector of parameters

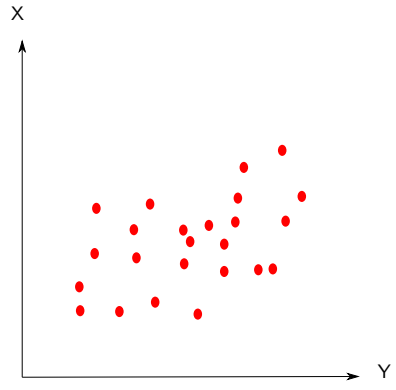
- The reconstruction error for the data is minimized:

$$\min_{\mu, \{\lambda_i\}, V_q} \sum_{i=1}^{N} ||x_i - \mu - V_q \lambda_i||^2$$

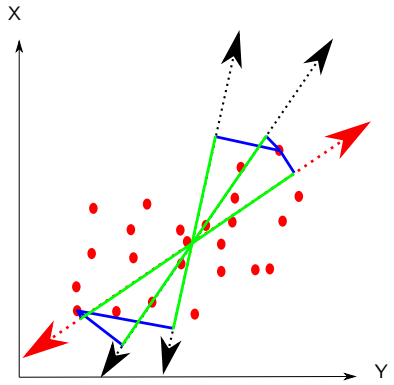
- Assuming $\bar{x} = 0$ we can obtain the projection matrix by **Singular Value Decomposition** of the data matrix X

$$X = UDV^T$$

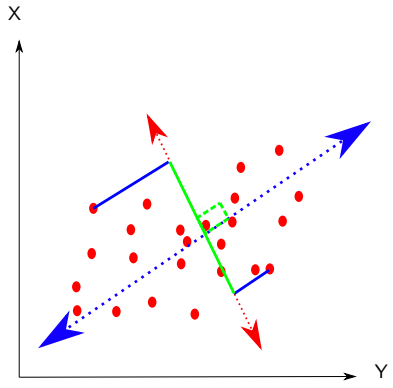
- U is a $N \times p$ orthogonal matrix, its columns are the left singular vectors
- D is a $p \times p$ diagonal matrix with ordered diagonal values called the singular values
- The columns of UD are the principal components
- We can pick the first principal components that account for a percentage of the total variance (e.g. 90%)



Original Data



First component along the maximum variance of the data



Next component maximum variance perpendicular to the other components

- PCA is a linear transformation, this means that if data is linearly separable, the reduced dataset will be linearly separable (given enough components)
- We can use the **kernel trick** to map the original attribute to a space where non-linearly separable data is linearly separable
- Distances among examples are defined as a dot product that can be obtained using a kernel:

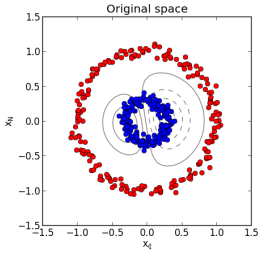
$$d(x_i, x_j) = \Phi(x_i)^T \Phi(x_j) = K(x_i, x_j)$$

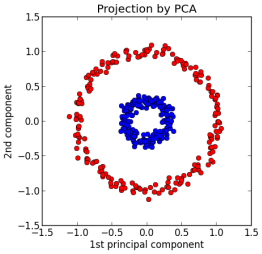
- Different kernels can be used to perform the transformation to the feature space (polynomial, gaussian...)
- The computation of the components is equivalent to PCA but performing the eigen decomposition of the covariance matrix computed for the transformed examples

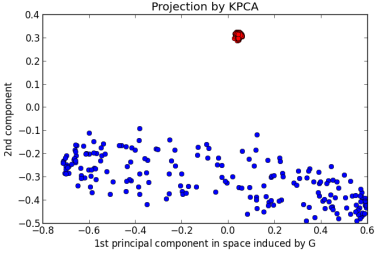
$$C = \frac{1}{M}\sum_{j=1}^{M}\Phi(x_j)\Phi(x_j)^T = \frac{1}{M}\sum_{j=1}^{M}K(x_i, x_j)$$

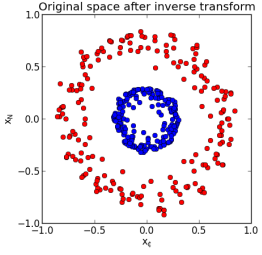
- The components are lineal combinations of features in the feature space

- Pro: Helps to discover patterns that are non-linearly separable in the original space
- Con: Does not give a weight/importance for the new components


Original space
x_{i1}
x_{i2}


Projection by PCA
2nd component
1st principal component


Projection by KPCA
2nd component
1st principal component in space induced by G

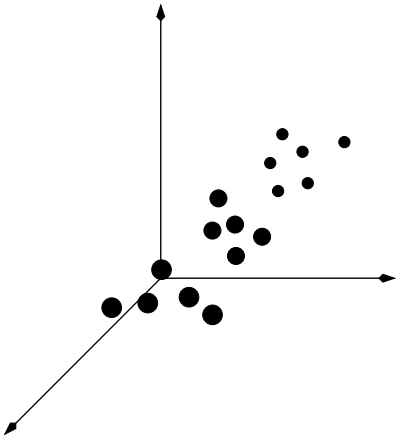

Original space after inverse transform
x_{i1}
x_{i2}

- PCA transforms data to a space of the same dimensionality (all eigenvalues are non-zero)
- An alternative is to solve the minimization problem posed by the reconstruction error using regularization
- A penalization term is added to the objective function proportional to the norm of the eigenvalues matrix

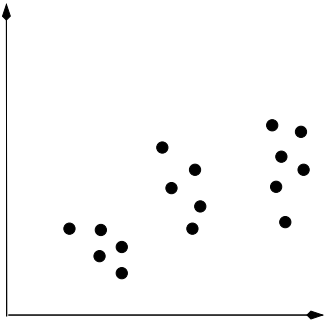
$$\min_{U,V} \|X - UV\|_2^2 + \alpha\|V\|_1$$

- The ℓ-1 norm regularization will encourage sparse solutions (zero eigenvalues)

A transformation matrix maps a dataset from M dimensions to N dimensions preserving pairwise data distances



[MxN]



- **Multidimensional Scaling:** Projects the data to a space with less dimensions preserving the pair distances among the data
- A projection matrix is obtained by optimizing a function of the pairwise distances (stress function)
- The actual attributes are not used in the transformation
- Different objective functions that can be used (least squares, Sammong mapping, classical scaling...)

- Least Squares Multidimensional Scaling (MDS)
- The distortion is defined as the square distance between the original distance matrix and the distance matrix of the new data

$$S_D(z_1, z_2, \ldots, z_n) = \left[\sum_{i \neq i'} (d_{ii'} - \|z_i - z_{i'}\|_2)^2 \right]$$

- The problem is defined as:

$$\arg\min_{z_1, z_2, \ldots, z_n} S_D(z_1, z_2, \ldots, z_n)$$

- Several optimization strategies can be used
- If the distance matrix is euclidean it can be solved using eigen decomposition just like PCA
- In other cases gradient descent can be used with the derivative of $S_D(z_1, z_2, \ldots, z_n)$ and a step α in the following fashion:

  1. Begin with a guess for Z
  2. Repeat until convergence:

$$Z^{(k+1)} = Z^{(k)} - \alpha \nabla S_D(Z)$$

- Sammong Mapping (emphasis on smaller distances)

$$S_D(z_1, z_2, \ldots, z_n) = \left[\sum_{i \neq i'} \frac{(d_{ii'} - \|z_i - z_{i'}\|)^2}{d_{ii'}} \right]$$

- Classical Scaling (similarity instead of distance)

$$S_D(z_1, z_2, \ldots, z_n) = \left[\sum_{i \neq i'} (s_{ii'} - \langle z_i - \bar{z}, z_{i'} - \bar{z} \rangle)^2 \right]$$

- Non metric MDS (assumes a ranking among the distances, non euclidean space)

$$S_D(z_1, z_2, \ldots, z_n) = \frac{\sum_{i,i'} [\theta(||z_i - z_{i'}||) - d_{ii'}]^2}{\sum_{i,i'} d_{ii'}^2}$$

- A random transformation matrix is generated:
  - Rectangular matrix $N \times d$
  - Columns must have unit length
  - Elements are generated from a gaussian distribution
- A matrix generated this way is almost orthogonal
- The projection will preserve the relative distances among pairs of examples
- The Johnson-Lindenstrauss lemma allows picking a number of dimensions to obtain the desired approximation

- This formulation assumes that the data is a sum of unknown positive latent variables
- NMF performs an approximation of a matrix as the product of two matrices

$$V = W \times H$$

- The main difference with PCA is that the values of the matrices are constrained to be positive
- The positiveness assumption helps to interpret the result
  - Eg.: In text mining, a document is an aggregation of topics

- The optimization problem for NMF is defined as given a number of dimensions $k < \min(m, n)$, find the pair of non-negative matrices $W \in \mathbb{R}^{m \times k}$ and $H \in \mathbb{R}^{k \times n}$ that minimize

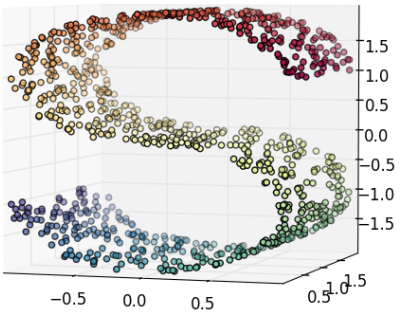
$$f(W, H) = \frac{1}{2}||A - WH||_F^2$$

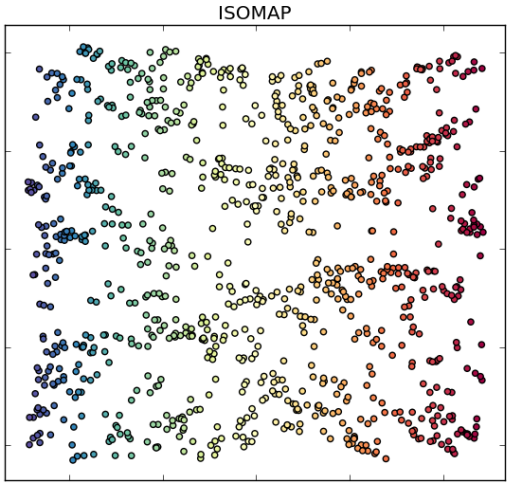
- Given that this is a non-convex problem can be solved using gradient descent or alternate least squares
- There are several variants forcing sparsity or orthogonality of the matrices

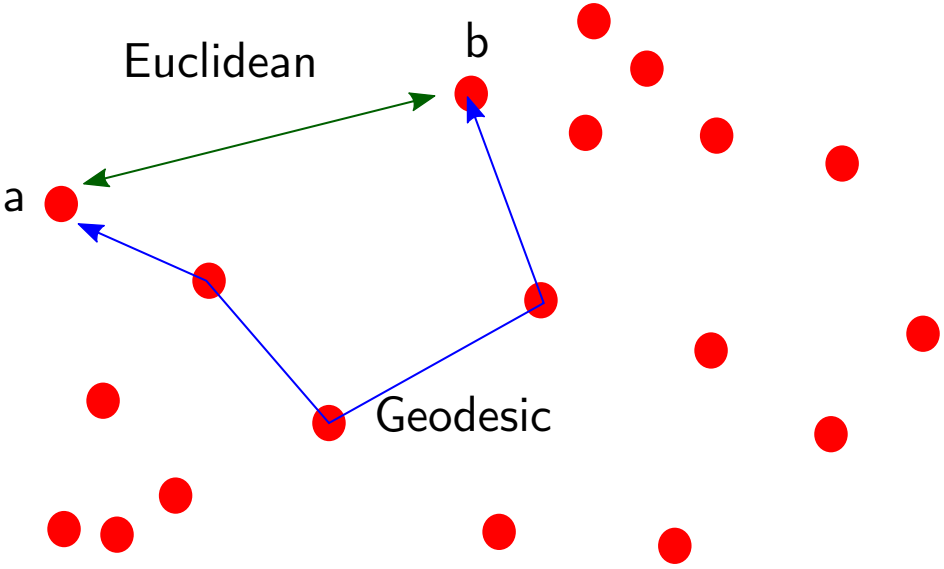
Non linear methods

- The previous methods perform a linear transformation between the original space, and the final space
- For some datasets this kind of transformation is not enough to maintain the information of the original data
- Nonlinear transformations methods:
  - ISOMAP
  - Local Linear Embedding
  - Local MDS
  - t-SNE

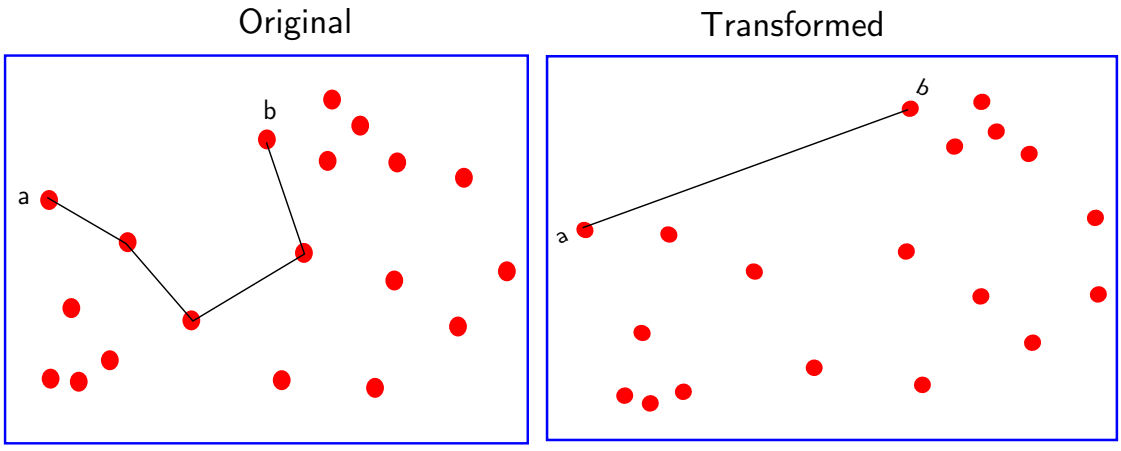
- Assumes a low dimensional dataset embedded in a larger number of dimensions
- The geodesic distance is used instead of the euclidean distance
- The relation of an instance with its immediate neighbors is more representative of the structure of the data
- The transformation generates a new space that preserves neighbourhood relationships


1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−0.5
0.0
0.5
0.5
1.0
1.5

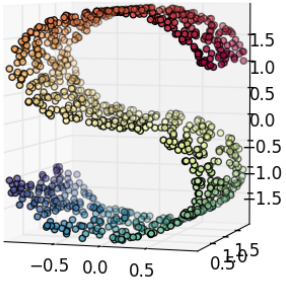

ISOMAP

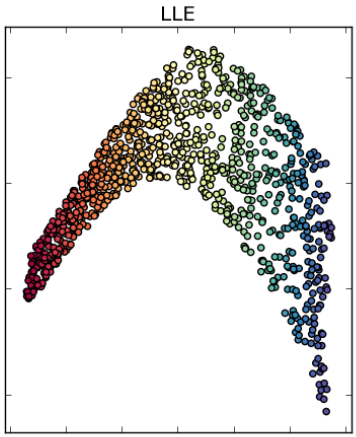

Euclidean
b
a
Geodesic

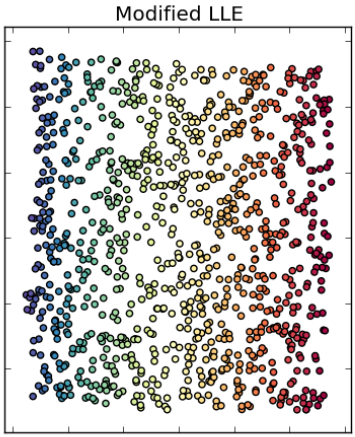
1. For each data point find its k closest neighbours (points at minimal euclidean distance)
2. Build a graph where each point has an edge to its closest neighbours
3. Approximate the geodesic distance for each pair of points by the shortest path in the graph
4. Apply a MDS algorithm to the distance matrix of the graph


Original
a
b
Transformed
a
b

- Performs a transformation that preserves local structure
- Assumes that each instance can be reconstructed by a linear combination of its neighbours (weights)
- From these weights a new set of data points that preserve the reconstruction is computed for a lower dimensional space
- Different variants of the algorithm exist


1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−0.5
0.0
0.5
0.5
0.0
−0.5
−1.0
−1.5


LLE


Modified LLE

1. For each data point find the K nearest neighbours in the original space of dimension p ($\mathcal{N}(i)$)
2. Approximate each point by a mixture of the neighbours:

$$\min_{W_{ik}} \|x_i - \sum_{k \in \mathcal{N}(i)} w_{ik} x_k\|^2$$

   and $\sum_{k \in \mathcal{N}(i)} w_{ik} = 1$ and $K < p$
3. Find points y_i in a space of dimension $d < p$ that minimize:

$$\sum_{i=0}^{N} \|y_i - \sum_{k \in \mathcal{N}(i)} w_{ik} y_k\|^2$$

- Performs a transformation that preserves locality of closer points and puts farther away non neighbor points
- Given a set of pairs of points $\mathcal{N}$ where a pair (i, i') belong to the set if i is among the K neighbours of i' or vice versa
- Minimize the function:

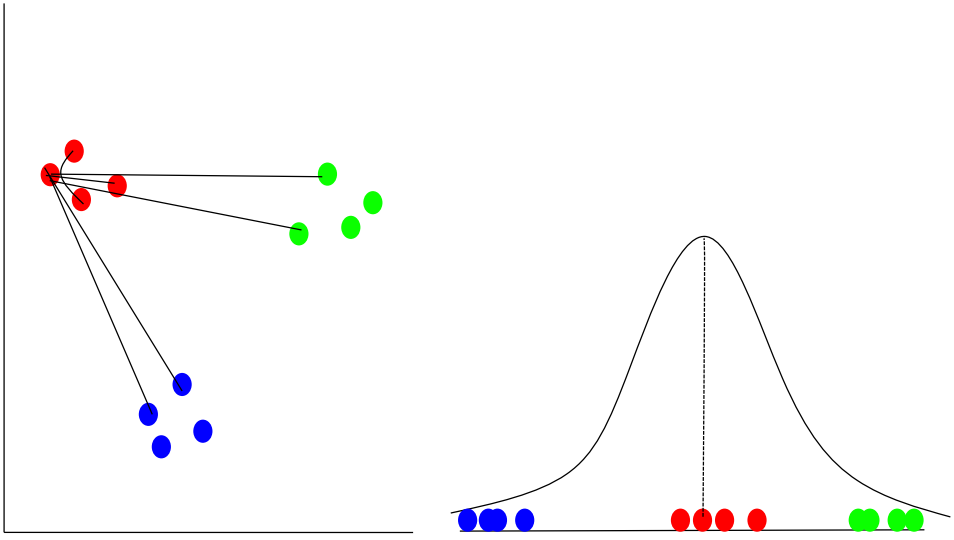
$$S_L(z_1, z_2, \ldots, z_N) = \sum_{(i,i') \in \mathcal{N}} (d_{ii'} - \|z_i - z_{i'}\|)^2 - \tau \sum_{(i,i') \notin \mathcal{N}} (\|z_i - z_{i'}\|)$$

- The parameters τ controls how much the non neighbours are scattered

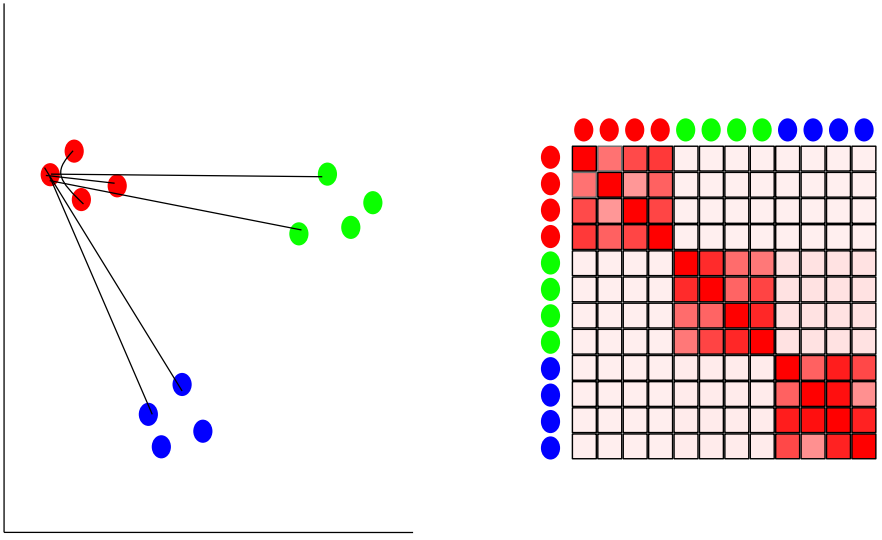
- t-Stochastic Neighbour Embedding (t-SNE)
- Used mainly as visualization tool (2-3 dimensions), not for transforming the data for applying ML algorithms
- Assumes that the distances among examples define a probability distribution that must be preserved in a lower dimensional space
- Many parameters apart from the number of target dimensions, tricky to use (see this link)
- Stochastic algorithm (depends on initialization), it can be initialized to the result of PCA

- t-SNE assumes that distances among examples are defined as a gaussian distribution
- Distances from each example to the rest are scaled to sum one (so it is a proper probability distribution)
- We want to find a projection of the data that preserves this probability distribution on a lower dimensionality space
- Examples are distributed in the new space, and their distance distributions are computed
- Examples are iteratively moved to minimize the Kullback-Leibler distance[1] among the distribution of the neighbours distances in the original and in the projected space

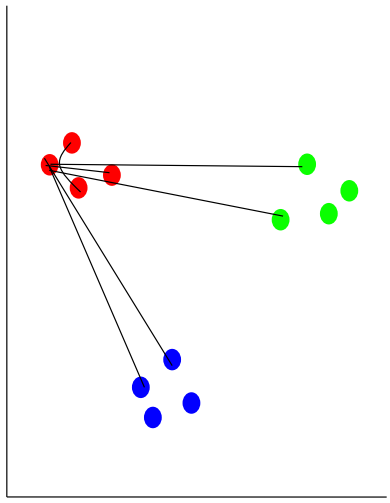
[1] Measure from Information Theory that computes the similarity among data distributions

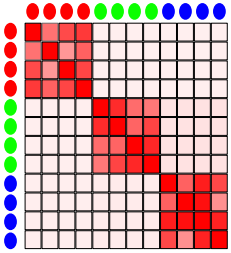


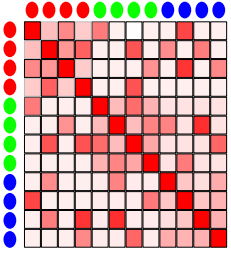
Each example has a distance probability distribution



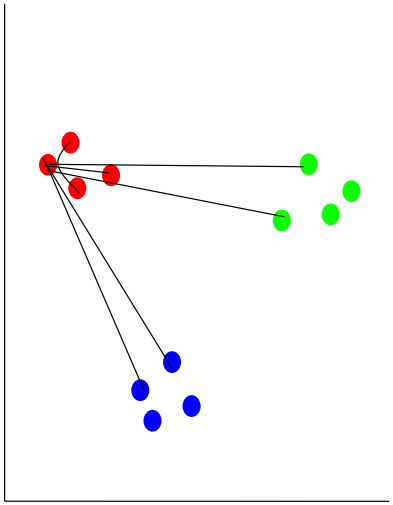
A similarity distribution is obtained

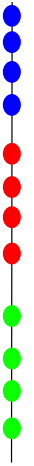


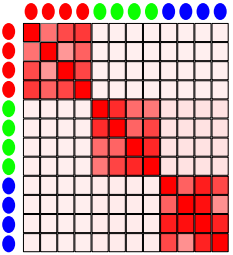


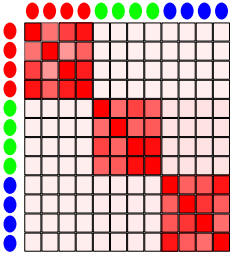


We distribute the data in a lower dimensionality space









Data is moved so the similarity distributions get close

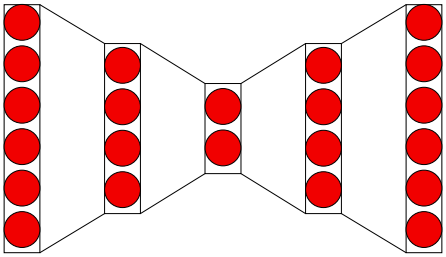
- Autoencoders are fully connected neural networks able to learn a representation from a dataset
- They are trained to reproduce their input on their output using a specific architecture that learn the representation
- An autoencoder is defined by
  - An encoder that can transform the data to a possibly lower dimensional space
  - A decoder that recreates the original data from the representation obtained by the encoder
- Only the encoder needs to be kept after training for performing the transformation

- The training needs to minimize a loss function that measures the difference between the original data and the output of the network.
- Encoder and decoder can be seen as two functions applied to the data (f and g), so the loss would be:

$$\mathcal{L}(x) = L(x, g(f(x)))$$

where L can be the mean squared error

- The architecture of an autoencoder combines:
  - An encoder that has several fully connected layers with progressively fewer units, with the number of units of the last layer as the dimensionality of the resulting space
  - An encoder defined symmetrically, so it increases the number of units until the original number of dimensions is reached

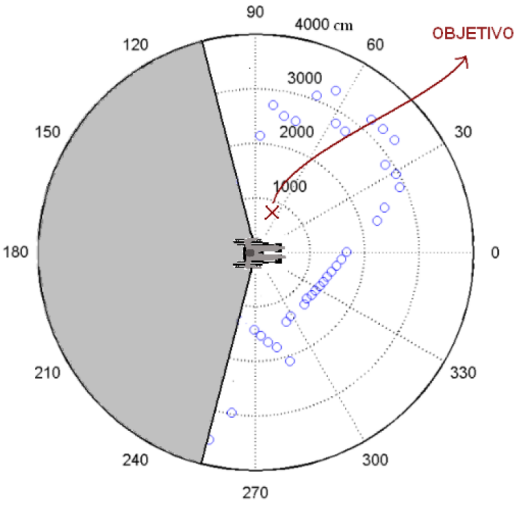


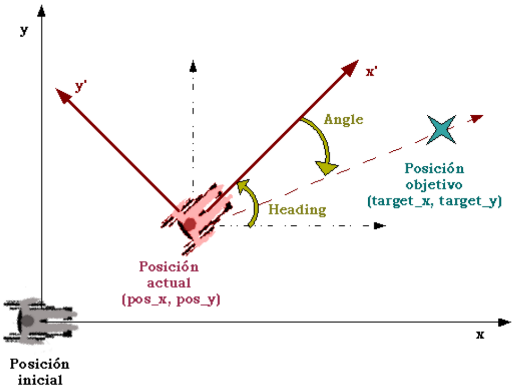
- When linear activation functions are used this architecture is enough to learn a representation that is equivalent to PCA
- Using non-linearities or having a decoder with more dimensionality than the input need for techniques that avoid trivial solutions or overfitting
  - Regularization is a common technique that constraints the values of the parameters of the model or bias the search towards smooth functions
  - Denoising autoencoders change the loss function by using during training corrupted examples, so the network learns how to reconstruct the original data

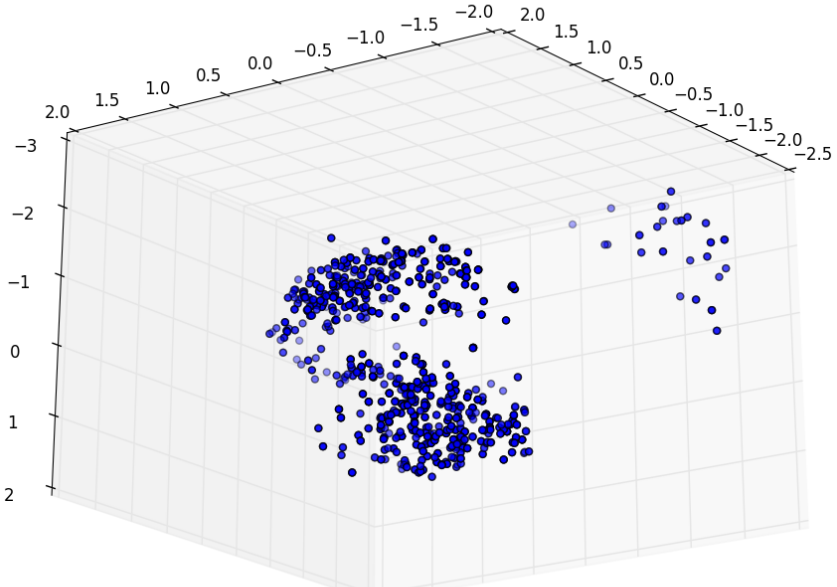
Application: Wheelchair control

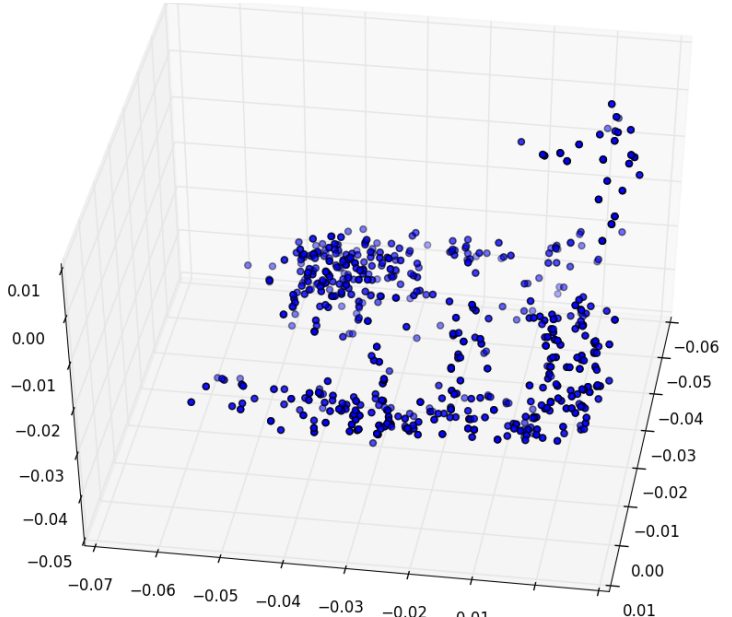
- Wheelchair with shared control (patient/computer)
- Recorded trajectories of several patients in different situations
  - Angle/distance to the goal, Angle/distance to the nearest obstacle from around the chair (210 degrees)
- Characterization about how the computer helps the patients with different handicaps
- Is there any structure in the trajectory data?

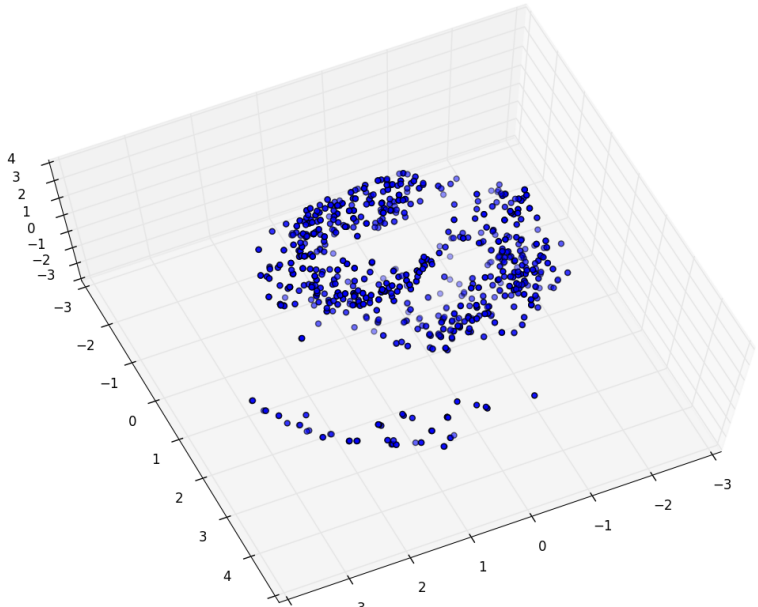

Joystick
PC
Láser

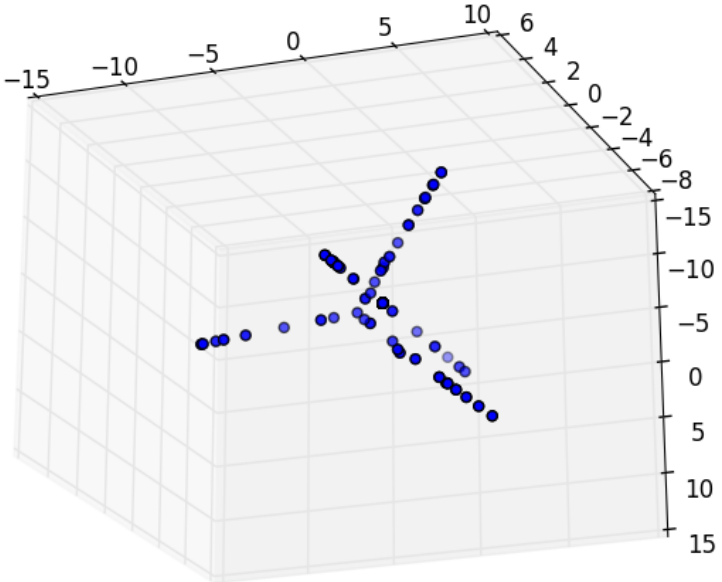

90
4000 cm
OBJETIVO
120
60
3000
150
2000
30
1000
180
0
210
330
240
300
270

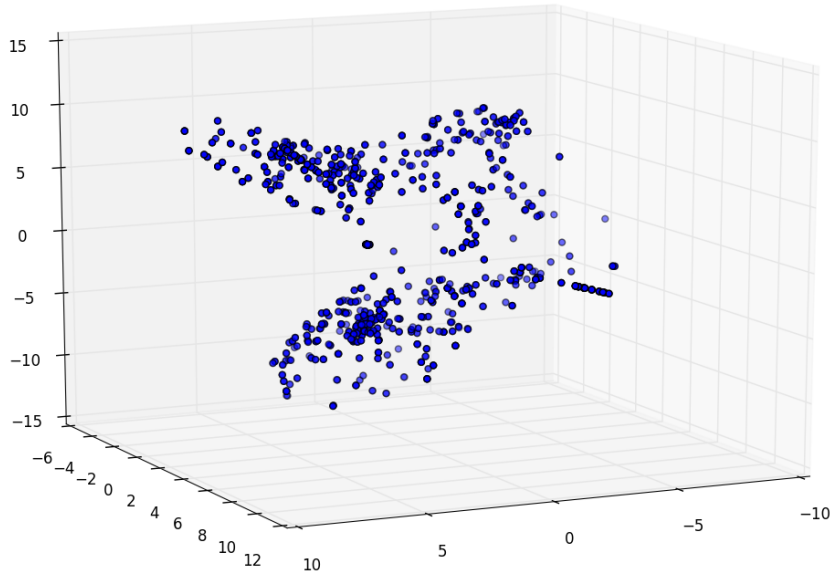

y
y'
x'
Angle
Posición
objetivo
(target_x, target_y)
Heading
Posición
actual
(pos_x, pos_y)
x
Posición
inicial


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3
−2
−1
0
1
2


0.01
0.00
−0.01
−0.02
−0.03
−0.04
−0.05
−0.07
−0.06
−0.05
−0.04
−0.03
−0.02
0.01
−0.06
−0.05
−0.04
−0.03
−0.02
−0.01
0.00
0.01


4
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3
4
−3
−2
−1
0
1
2


−15
−10
−5
0
5
10
6
4
2
0
−2
−4
−6
−8
−15
−10
−5
0
5
10
15


15
10
5
0
−5
−10
−15
−6
−4
−2
0
2
4
6
8
10
12
10
5
0
−5
−10

Attribute Selection

- To eliminate from the dataset all the redundant or irrelevant attributes
- The original attributes are preserved
- Less developed than in Supervised Attribute Selection
  - Problem: An attribute can be relevant or not depending on the goal of the discovery process
- There are mainly two techniques for attribute selection: Wrapping and Filtering

- A model evaluates the relevance of subsets of attributes
- In supervised learning this is easy, in unsupervised learning it is very difficult
- Results depend on the chosen model and on how well this model captures the actual structure of the data

- Clustering algorithms that compute weights for the attributes based on probability distributions
- Clustering algorithms with an objective function that penalizes the size of the model
- Consensus clustering

- A measure evaluates the relevance of each attribute individually
- This kind of measures are difficult to obtain for unsupervised tasks
- The idea is to obtain a measure that evaluates the capacity of each attribute to reveal the structure of the data (e.g.: class separability, similarity of instances in the same class)

- Measures of properties of the spatial structure of the data (Entropy, PCA, laplacian matrix)
- Measures of the relevance of the attributes respect the inherent structure of the data
- Measures of attribute correlation

- The **Laplacian Score** is a filter method that ranks the features respect to their ability of preserving the natural structure of the data.
- This method uses the spectral matrix of the graph computed from the near neighbours of the examples

- The Similarity matrix is usually computed using a gaussian kernel (edges not present have a value of 0)

$$S_{ij} = e^{\frac{||x_i - x_j||^2}{\sigma}}$$

- The Degree matrix is a diagonal matrix where the elements are the sum of the rows of S
- The Laplacian matrix is computed as

$$L = D - S$$

- The score first computes for each attribute r and their values f_r the transformation $\tilde{f}_r$ as:

$$\tilde{f}_r = f_r - \frac{f_r^T D 1}{1^T D 1} 1$$

- and then the score L_r is computed as:

$$L_r = \frac{\tilde{f}_r^T L \tilde{f}_r}{\tilde{f}_r^T D \tilde{f}_r}$$

- This gives a ranking for the relevance of the attributes

These two Python Notebooks show some examples of dimensionality reduction and feature selection

- Dimensionality reduction and feature selection Notebook (click here to open the notebook in colab)
- Linear and non-linear dimensionality reduction Notebook (click here to open the notebook in colab)

If you download the notebook you will be able to use it locally (run jupyter notebook to open the notebooks)