Data Preprocessing

Javier Béjar

BY: ☑ ☑ ☑

CS - MAI

AMLT - 2016/2017
1. Introduction
   - Data Representation

2. Data Preprocessing
   - Outliers
   - Missing Values
   - Normalization
   - Discretization

3. Dimensionality Reduction
   - The curse of dimensionality
   - Component Analysis
   - Multidimensional Scaling
   - Application: Wheelchair control
   - Attribute Selection

4. Similarity functions
Introduction

Data Preprocessing

Dimensionality Reduction

Similarity functions
Data representation

- Unstructured datasets:
  - Examples described by a flat set of attributes: attribute-value matrix

- Structured datasets:
  - Individual examples described by attributes but also having relations among them: sequences (time, spatial, ...), trees, graphs
  - Sets of structured examples (sequences, graphs, trees)
Unstructured data

- Only one table of observations

- Each example represents an instance of the problem

- Each instance is represented by a set of attributes (discrete, continuous)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.1</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5.7</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>-2.2</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-9.0</td>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0.3</td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2.1</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Structured data

- One sequential relation among instances (Time, Strings)
  - Several instances with internal structure (eg: sequences of events)
  - Subsequences of unstructured instances (eg: sequences of complex transactions)
  - One big instance (eg: time series)

- Several relations among instances (graphs, trees)
  - Several instances with internal structure (eg: XML documents)
  - One big instance (eg: social network)
Data Streams

- Endless sequence of data (e.g., sensor data)
  - Several streams synchronized
  - Unstructured instances
  - Structured instances

- Static/Dynamic model
Data representation

- Most of unsupervised learning algorithms are specifically fitted for unstructured data.

- The data representation used is equivalent to a database table (attribute-value pairs).

- More specialized algorithms have been developed to process structured data: Graph clustering, Sequence mining, Frequent substructures.

- The representation of these types of data is sometimes algorithm dependent.
1 Introduction

2 Data Preprocessing

3 Dimensionality Reduction

4 Similarity functions
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 analysis task is to assess the quality of the data
- The techniques used for data preprocessing are usually oriented to unstructured data
Outliers

- Outliers are examples that have values very different from the rest of the dataset.
- They can be considered as examples with erroneous values.
- It could have an important impact on the results of some algorithms.
Outliers

- The different values can happen in all or only a few of attributes
- The usual way to correct this circumstance is to eliminate the examples
- If the exceptional values are only in a few attributes these could be treated as missing values
Outliers Detection

Parametric methods

- Assuming a probabilistic distribution for the attributes (e.g., Univariate Gaussian, Gaussian Mixture Model), test for the probability of the examples, those with low probability can be labeled as outliers.

- Using the upper and lower quantiles of the distribution as decision criteria, discard the observations that are at a specific distance above or below the quantiles.
Outliers Detection
Non parametric methods

- Generate a histogram of the attributes (binning) and discard the examples that are in bins that have a low number of examples.

- Perform kernel density estimation (KDE) and label as outliers examples that belong to low density areas.

- Using a proximity criteria, test for abnormalities in the distribution of the distances to the k-nearest neighbors.
Missing values

- 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:** we are modifying the statistical distribution of the data.
Missing values

- Missing Values
- Mean substitution
- 1-neighbor substitution
Normalization

Normalizations are applied to quantitative attributes in order to eliminate the effect of having different scale measures

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

\[
\frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}
\]

- **Distribution normalization**: We transform the data in order to obtain certain specific statistical distribution with preestablished parameters (Usually a Gaussian distribution with mean 0 and standard deviation 1)

\[
\frac{x - \mu_x}{\sigma_x}
\]
Discretization

Discretization allows to transform quantitative attributes to qualitative attributes

- **Equal size bins**: We choose the number of values that we need and divide the interval in equal bins

- **Equal frequency bins**: We choose the number of values that we need, but we create the intervals so each bean has the same number of examples (the size of the intervals will be different)

- **Distribution approximation**: We calculate a histogram of the data and we fit a function. The intervals are where the function has its minima

- **Other techniques**: We can apply entropy based measures, MDL or even clustering
Discretization

- Same size
- Same Frequency
- Histogram
Python Notebooks

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

- Missing Values Notebook (click here to go to the url)
- Preprocessing Notebook (click here to go to the url)

If you have downloaded the code from the repository you will able to play with the notebooks (run jupyter notebook to open the notebooks)
1. Introduction

2. Data Preprocessing

3. Dimensionality Reduction

4. Similarity functions
The curse of dimensionality

- There are two problems that come from the dimensionality of a dataset
  - The computational cost of processing the data (scalability of the algorithms)
  - The quality of the data (more probability of bad data)
- There are two elements that define the dimensionality of a dataset
  - The number of examples
  - The number of attributes
- Usually the problem of having too many examples can be solved using sampling.
- Attribute reduction has different solutions
Reducing attributes

- Usually the number of attributes of the dataset has an impact on the performance of the algorithms:
  - Because their poor scalability (cost is a function of the number of attributes)
  - Because the inability to cope with irrelevant/noisy/redundant attributes

- There are two main methodologies to reduce the number of attributes of a dataset
  - Transforming the data to a space of less dimensions preserving somewhat the original data (dimensionality reduction)
  - Eliminating the attributes that are not relevant for the goal task (feature subset selection)
Dimensionality Reduction

**Dimensionality reduction**

- We are looking for a new dataset that preserves the information of the original dataset but has less attributes.
- Many techniques have been developed for this purpose:
  - Projection to a space that preserve the statistical model of the data (PCA, ICA)
  - Projection to a space that preserves distances among the data (Multidimensional scaling, random projection, nonlinear scaling)
Component analysis

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

- **Independent Component Analysis:** Transforms the dataset projecting the data to a set of variables statistically independent (all statistical momentums are independent). We assume non gaussian data.
Principal Component Analysis

The model that we want is a projection matrix where the dimensions are orthogonal (linearly independent) and preserve data variance.
Principal Component Analysis

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

\[ f(\lambda) = \mu + V_q \lambda \]

\( \mu \) is a location vector in \( \mathbb{R}^p \), \( V_q \) is a \( p \times q \) matrix of \( q \) orthogonal unit vectors and \( \lambda \) is a \( q \) vector of parameters.

- We want to minimize the reconstruction error for the data:

\[
\min_{\mu, \{\lambda_i\}, V_q} \sum_{i=1}^{N} \left\| x_i - \mu - V_q \lambda_i \right\|^2
\]
Principal Component Analysis

- Optimizing partially for $\mu$ and $\lambda_i$:

$$\mu = \bar{x}$$

$$\lambda_i = V_{iq}^{-1} (x_i - \bar{x})$$

- We can obtain the matrix $V_q$ by minimizing:

$$\min_{V_q} \sum_{i=0}^{N} \| (x_i - \bar{x}) - V_q V_q^T (x_i - \bar{x}) \|^2_2$$
Principal Component Analysis

- Assuming $\bar{x} = 0$ we can obtain the projection matrix $H_q = V_q V_q^T$ 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*.
- $V$ is a $p \times p$ diagonal matrix with ordered diagonal values called the *singular values*.
- The columns of $UD$ are the *principal components*.
- The solution to the minimization problem are the first $q$ principal components.

- The singular values are proportional to the reconstruction error.
Kernel PCA

- 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, ...).
Kernel PCA

- The computation of the components is equivalent to PCA but performing the eigen decomposition of the covariance matrix computed for the transformed examples

\[ C = \frac{1}{M} \sum_{j=1}^{M} \Phi(x_j)\Phi(x_j)^T \]

- **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
Kernel PCA

Original space

Projection by PCA

Projection by KPCA

Original space after inverse transform
Sparse PCA

- PCA transforms data to a space of the same dimensionality (all eigenvalues are non zero)
- An alternative to PCA 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 \(\ell\)-1 norm regularization will encourage sparse solutions (zero eigenvalues)
A transformation matrix transforms a dataset from $M$ dimensions to $N$ dimensions preserving pairwise data distances.
Multidimensional Scaling

- **Multidimensional Scaling**: Projects a dataset 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).
- This means that the actual attributes are not used in the transformation.
- There are different objective functions that can be used (least squares, Sammon mapping, classical scaling, ...).
- The optimization problem is solved by gradient descent.
Multidimensional 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 to solve this problem.

If the distance matrix is euclidean it can be solved using eigen decomposition just like PCA.

In other cases gradient descent can be used using the derivative of $S_D(z_1, z_2, ..., z_n)$ and a step $\alpha$ in the following fashion:

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

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

- Sammon 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)

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

- A random transformation matrix is generated:
  - Rectangular matrix $N \times d$
  - Columns must have unit length
  - Elements are generated from a gaussian distribution (not strictly necessary)
- A matrix generated this way is almost orthogonal (close to a PCA transformation)
- The projection will preserve the relative distances among examples
- The effectiveness usually depends on the number of dimensions (this can be estimated from the number of examples)
Nonnegative Matrix Factorization (NMF)

- NMF performs an approximation of a matrix in the product of two matrices
  \[ V = WH \]

- Similar to PCA, the eigen decomposition transforms the original data matrix in the product of two matrices

- The main difference is that the values of the matrices are constrained to be positive

- The formulation assumes that the data is a sum of unknown positive latent variables

- The positiveness assumption helps to interpret the result
  - Eg.: In text mining, a document is an aggregation of topics

- There are many variants and algorithms
Nonlinear scaling

- 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
**ISOMAP**

- Assumes a low dimensional dataset embedded in a larger number of dimensions
- The geodesic distance is used instead of the euclidean distance
- It is assumed that 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 neighborhood relationships
ISOMAP

Euclidean

Geodesic

a

b
**ISOMAP**

**Algorithm:**

1. For each data point find its k closest neighbors (points at minimal euclidean distance)
2. Build a graph where each point has an edge to its closest neighbors
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
ISOMAP Example

Original

Transformed
Local linear embedding

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

Algorithm:

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

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

where $w_{ik} = 0$ if $k \notin \mathcal{N}(i)$ and $\sum_{i=0}^{N} w_{ik} = 1$ and $K < p$

3. Find points $y_i$ in a space of dimension $d < p$ that minimize:

$$
\sum_{i=0}^{N} \| y_i - \sum_{k=0}^{N} w_{ik} y_k \|^2
$$
Local MDS

- 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 neighbors of $i'$ or viceversa
- 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 $\tau$ controls how much the non neighbors are scattered
Wheelchair control characterization

- 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?
Wheel chair control characterization
Wheel chair control characterization
Wheelchair control characterization

PCA 88 → 3 dimensions
Wheelchair control characterization

SparsePCA 88 $\rightarrow$ 3 dimensions
Wheelchair control characterization

MDS 88 $\implies$ 3 dimensions
Wheelchair control characterization

ISOMAP ($k-n=3$) $88 \Rightarrow 3$ dimensions
Wheelchair control characterization

ISOMAP ($k-n=10$) 88 $\Rightarrow$ 3 dimensions
Unsupervised Attribute Selection

- The goal is to eliminate from the dataset all the redundant or irrelevant attributes.
- The original attributes are preserved.
- The methods for Unsupervised Attribute Selection are less developed than in Supervised Attribute Selection.
- The problem is that 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.
Attribute selection

Wrapping:
- A model to evaluate 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 models capture the actual structure of the data

Filtering:
- A measure assess the relevance of each attribute
- 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 (class separability, similarity of instances in the same class)
Unsupervised feature selection

- The filter methods order the attributes using measures of the structure of the data
  - 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 wrapper methods are more diverse
  - Clustering algorithms that compute weights for the attributes
  - Clustering algorithms with an objective function that penalizes the size of the model
  - Consensus clustering
Laplacian Score

- 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 neighbors of the examples.
- Similarity is usually computed using a gaussian kernel (edges not present have a value of 0).

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

And all edges not present have a value of 0.
The Laplacian matrix is computed from the similarity matrix $S$ and the degree matrix $D$ as:

$$L = S - D$$

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 dimensionality reduction and feature selection

- Dimensionality reduction and feature selection Notebook (click here to go to the url)
- Linear and non linear dimensionality reduction Notebook (click here to go to the url)

If you have downloaded the code from the repository you will able to play with the notebooks (run jupyter notebook to open the notebooks)
1 Introduction

2 Data Preprocessing

3 Dimensionality Reduction

4 Similarity functions
Unsupervised algorithms need similarity/distance to compare examples.

This comparison will be obtained using functions of the attributes of the examples.

Usually we suppose that the examples are embedded in a $N$-dimensional space where it can be defined a similarity/distance.

There are domains where this assumption is not true so some other kind of functions will be needed to represent instances relationships.
Properties of Similarity/Distance functions

The properties of a similarity function are:
1. \( s(p, q) = 1 \iff p = q \)
2. \( \forall p, q \ s(p, q) = s(q, p) \)

The properties of a distance function are:
1. \( \forall p, q \ d(p, q) \geq 0 \) and \( \forall p, q \ d(p, q) = 0 \iff p = q \)
2. \( \forall p, q \ d(p, q) = d(q, p) \)
3. \( \forall p, q, r \ d(p, r) \leq d(q, p) + d(p, r) \)
Distance functions

- **Minkowski metrics** (Manhattan, Euclidean)

\[
d(i, k) = \left( \sum_{j=1}^{d} |x_{ij} - x_{kj}|^r \right)^{\frac{1}{r}}
\]

- **Mahalanobis distance**

\[
d(i, k) = (x_i - x_k)^T \cdot \varphi^{-1} \cdot (x_i - x_k)
\]

where \(\varphi\) is the matrix covariance of the attributes.
Distance functions

- **Chebyshev Distance**

\[ d(i, k) = \max_j |x_{ij} - x_{kj}| \]

- **Camberra distance**

\[ d(i, k) = \sum_{j=1}^{d} \frac{|x_{ij} - x_{kj}|}{|x_{ij}| + |x_{kj}|} \]
Similarity functions

- **Cosine similarity**
  \[ d(i, k) = \frac{x_i^T \cdot x_j}{\|x_i\| \cdot \|x_j\|} \]

- **Pearson correlation measure**
  \[ d(i, k) = \frac{(x_i - \bar{x}_i)^T \cdot (x_j - \bar{x}_j)}{\|x_i - \bar{x}_i\| \cdot \|x_j - \bar{x}_j\|} \]
Similarity functions

Binary data

- **Coincidence coefficient**

\[ s(i, k) = \frac{a_{00} - a_{11}}{d} \]

- **Jaccard coefficient**

\[ s(i, k) = \frac{a_{11}}{a_{00} + a_{01} + a_{10}} \]
This Python Notebook shows examples of using different distance functions.

- Distance functions Notebook ([click here](#) to go to the url)

If you have downloaded the code from the repository you will be able to play with the notebooks (run jupyter notebook to open the notebooks)
In the code from the repository inside subdirectory `DimReduction` you have the Authors python program. The code uses the datasets in the directory `authors` from the datasets zipfile.

- **Auth1** has fragments of books that are novels or philosophy works.
- **Auth2** has fragments of books written in English and books translated to English.

The code transforms the text to attribute vectors and applies different dimensionality reduction algorithms. Modifying the code you can process one of the datasets and choose how the text is transformed into vectors.