

Big Data and Data Mining

Data Mining *Introduction and Preprocessing*

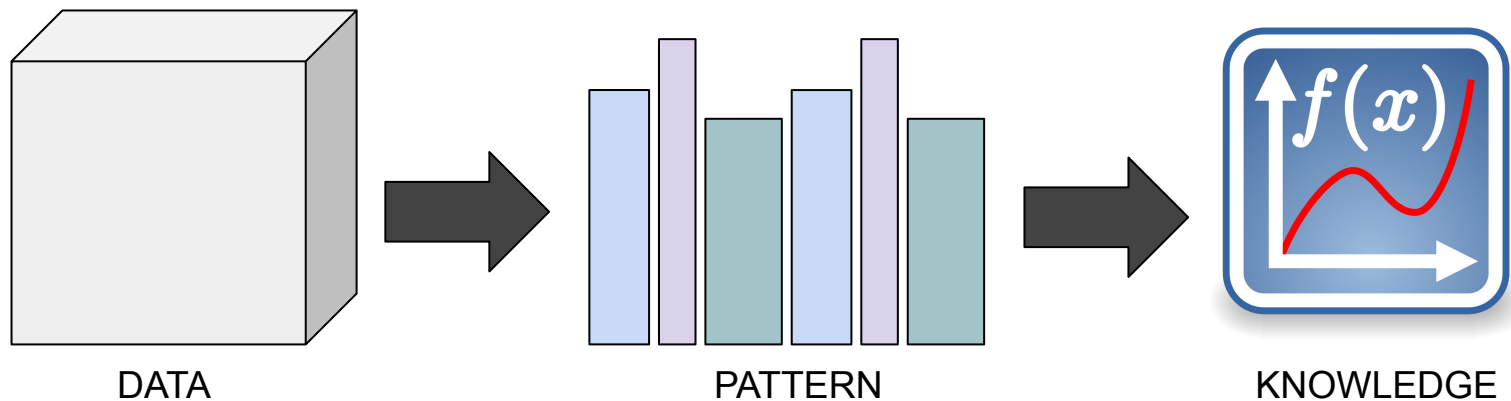
Flavio Bertini

flavio.bertini@unipr.it

Data Mining

*“Data mining is the computing process of discovering patterns in large data sets involving methods at the intersection of **machine learning**, **statistics**, and **database systems**”*

- The goal is the extraction of **patterns** and **knowledge** from large amount of data
- We will see now some example of patterns, knowledge and large amount of data involved in data mining



Patterns

- **Patterns** are regularities in the data, that - approximately - repeat for all of the observations in a predictable manner
- Examples of patterns:
 - In most cases, houses price in € is approximately 2500 times their square meters
 - Frequently, milk and cereals are bought together
 - It has been observed that people buy digital equipment in this order
 - 1) Personal Computer
 - 2) Digital Camera
 - 3) Memory Card

Knowledge

- Once a pattern is discovered, we can understand the logic behind the pattern, so that we can **describe** the pattern and **predict** similar phenomena
- **Knowledge** is a novel understanding on a subject
- Examples discovered knowledge:
 - How house prices are being set depending on square meters
 - Which products should be put closer in the supermarket
 - In which order digital equipment should be advertised in targeted marketing

Large amount of data

- In order to find patterns and deduce knowledge, we need to start from several **observations**
- In Data Mining observations are provided in the form of homogeneous **data** examples:
 - Many examples of house prices and their related size
 - Many examples of groceries transaction with the set of bought items
 - Many examples of sequences of purchases in the digital department of a superstore
- **Large amount** of data is needed in order to extract pattern or knowledge which are statistically significant
 - If we know only the price of three houses it may be due to chance, not a regularity!

Data Mining: Steps

1. **Define the problem:** what is the goal we are trying to achieve?
What knowledge we'd like to extract?
2. **Identify required data:** what data do we need in order to pursue the goal? In this step we collect and understand the data
3. **Prepare and pre-process:** select and cleanse the required data. Format the data to be the input of a *machine learning* algorithm (we will see what machine learning is in a moment)
4. **Model the hypothesis:** select machine learning algorithms and tune parameters to build an accurate “predictive” model
5. **Train and test:** train algorithms using a sample of the data, test it on unseen data
6. **Verify and deploy:** verify final model with stakeholders (final users), prepare visualization and deploy

In this course

- In this course we will assume that:
 1. Goal is already given
 2. Required data is already given
 6. Visualization and deployment vary strongly depending on the stakeholders needs and may go outside the scope of data mining techniques
- This leaves us with 3 steps:
 3. **Prepare and pre-process:** select and cleanse the required data. Format the data to be the input of a machine learning algorithm
 4. **Model the hypothesis:** select machine learning algorithms and tune parameters to build an accurate “descriptive” and then “predictive” model
 5. **Train and test:** train algorithms using a sample of the data, test it on unseen data



Machine Learning

- Machine learning (ML) is at the core of data mining, because it is where the abstraction from data to patterns happens
- **Basic idea:** given many observations, a function with a specified goal (e.g., guess a house's price given its features) is “learned” automatically from the data
- Why humans can't learn patterns directly from the data?
 - Data can be **very large** (many thousands of records) and **high dimensional** (many variables per record)
 - **Non-linear** relations between variables can be very hard to catch
 - Manually coding some algorithms can be incredibly **hard**: how would you code a program to detect cats in a picture, starting from raw pixels?

Example

- We have the data from 10 thousand houses in the U.S.

City	Latitude	Longitude	Rooms	SQM	Price
Los Angeles	33°51'37.0"N	118°08'22.2"W	3	130	420000
Los Angeles	33°50'17.7"N	118°09'12.6"W	2	60	380000
...
Albuquerque	35°14'22.0"N	106°26'26.2"W	2	140	220000
Albuquerque	35°32'23.0"N	106°38'21.2"W	3	150	250000

- Goal:** learn a function (model) that can infer the *Price* given all the other variables, for houses not in the 10 thousand dataset:

City	Latitude	Longitude	Rooms	SQM		Price
Los Angeles	33°49'32.3"N	118°08'44.1"W	5	230	➔	?



Probably, approximately correct!

- Being based on statistics, the patterns and knowledge extracted in data mining is almost **never 100% accurate**
 - many variables could be involved which are not in the data (e.g., a house seller may want to quick sell the house at a lower price)
 - data coming from the real world is affected by noise and randomness (e.g. mistyped price value)
- Machine learning foundation is the P.A.C. theory: **Probably Approximately Correct**
 - **Probably**: the model of the real world abstracted using machine learning **should be correct** with high probability
 - **Approximately**: the model should have low **generalization** error, meaning that it should **approximate the general case** and be accurate with unseen data

ML: common notions

- **Example:** an observation, such as the data of a single house
- **Input variables:** the variables that are given in input for each example (e.g., the square meters)
- **Output variable:** the variable we want to infer on the basis of the input variables (e.g., the house price)
- **Hypothesis** (a.k.a. model): the function from the input variables to the output variable learned by a machine learning algorithm
- **Label:** in supervised tasks (we'll see in a moment), it's the value of the output variable for a certain input instance. It is given from the data, it's considered the true value
- **Prediction:** it's the value of the output variable “guessed” by the hypothesis function. It should be correct with high probability
 - Note that here *prediction* does not stand for *forecast*, time may be not involved at all
 - Any time you predict into the future it is a forecast. All forecasts are predictions, but not all predictions are forecasts, as when you would use regression to explain the relationship between two variables

Notation

- A common notation is to denote with the:
 - capital letter **X** the input data matrix
 - lowercase **y** the output variable vector
 - lowercase **m** the number of examples
 - lowercase **n** the number of variables of the examples

X					y
City	Latitude	Longitude	Rooms	SQM	Price
Los Angeles	33°51'37.0"N	118°08'22.2"W	3	130	420000
Los Angeles	33°50'17.7"N	118°09'12.6"W	2	60	380000
Los Angeles	33°49'32.3"N	118°08'44.1"W	5	230	2500000
...
Albuquerque	35°14'22.0"N	106°26'26.2"W	2	140	220000
Albuquerque	35°32'23.0"N	106°38'21.2"W	3	150	250000

m

n

Supervised techniques

- **Supervised** techniques: a “training set” of data is given to the machine learning algorithm, with a label for each example, representing the *ground truth* for the output value
- It's **supervised** because we tell the algorithm **what should be the output for a set of examples**
 - The output variable is part of the dataset
 - **Example:** the actual price is given, along with the input variables
- Supervised learning has two tasks depending on the type of output variable:
 - **Regression:** when the output variable is a numerical value. For example in the problem of predicting the price of the house
 - **Classification:** when the output variable is a class. The easiest scenario is the binary classification problem where we want to predict if an example belong to a class or not (e.g., is the house located in Los Angeles? Yes or No)



Supervised ML: an example (1)

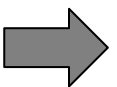
- The houses' prices example is a **supervised** learning task:
 - We have a large training set (data from 10K houses)
 - Among the variables there is the output variable (Price)
 - The output variable is the *ground truth*, because it's the actual price of the house
- The kind of task is a **regression** problem:
 - The **output variable we want to predict is a numerical value**

City	Latitude	Longitude	Rooms	SQM		Price
Los Angeles	33°49'32.3"N	118°08'44.1"W	5	230	➔	?

- Note: given a testing dataset, which is labeled with the actual price but unseen for the learning algorithm, it's easy to measure the **accuracy** or, conversely, the **error**

Supervised ML: an example (2)

- Suppose we want instead to predict if the house is in L.A. or Albuquerque, given all the other variables (including the price)
- The kind of task is a **classification** problem:
 - The **output variable we want to predict is a class value**

Latitude	Longitude	Rooms	SQM	Price		City
33°49'32.3"N	118°08'44.1"W	5	230	2500000		?

- When the classes are 2, it's a **binary** classification problem: output can be either 0 or 1. Example: is the house from L.A.?
- When the classes are more than two, it's a **multi-class** classification problem: **the output is an element of a finite set**. Example: from which city is the house, among these 20 cities?

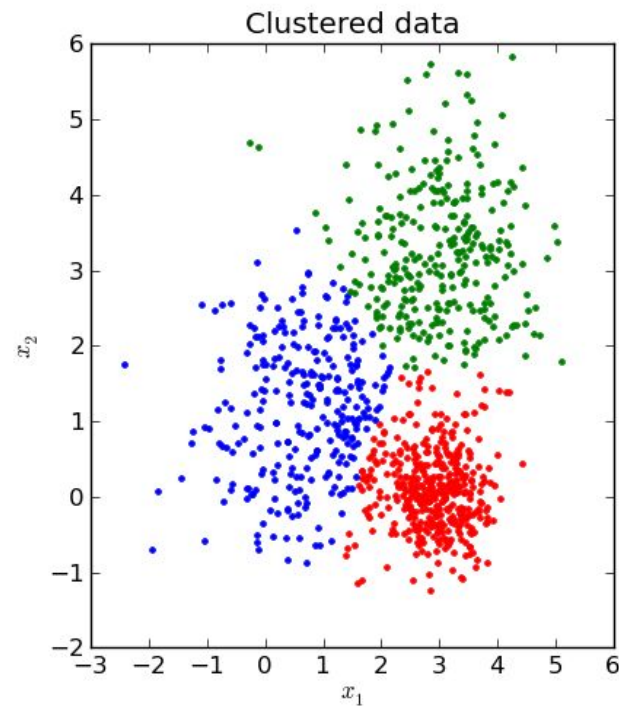
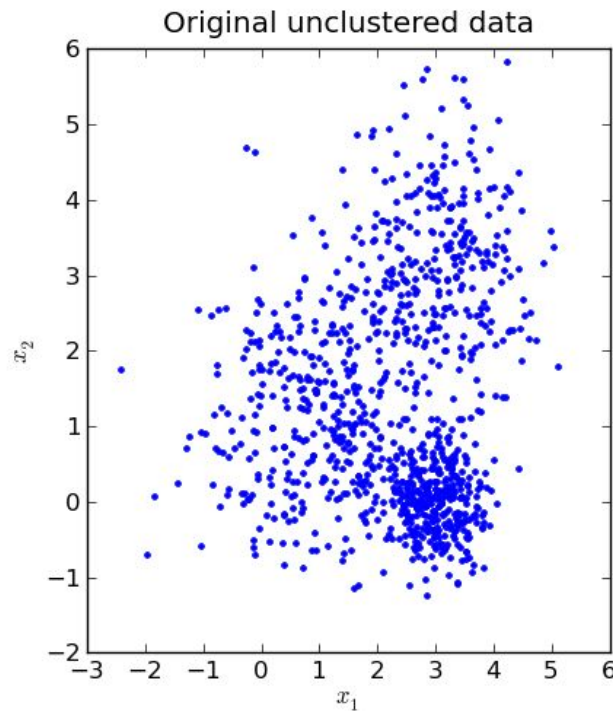


Unsupervised techniques

- **Unsupervised** techniques: there is no given value for the output variable, or there is **no output variable** at all
- Sometimes **there is no goal set** a-priori and we just want to find out regularities and relations inside the data
- Most common tasks (we will see the first two):
 - **Clustering**: split the examples into different groups (clusters). The examples in the same group should be similar, while examples in different groups should be dissimilar
 - **Association rules**: discover interesting relations between variables. E.g.,
 $\{onions, potatoes\} \Rightarrow \{burger\}$ in supermarket sales
 - **Anomaly detection**: find the “outliers” in a set of examples. E.g., from a log of bank transactions find the suspect ones
 - **Generative models**: given many examples generate a new, “synthetic” instance with similar features. E.g., melodies generation, poetry/book automatic writing
 - **Feature extraction**: reduces the number of variables by generating new, characterizing features

Unsupervised ML: an example

- **Clustering** is the most common task among the unsupervised tasks
- In this example our observation have two variables: x_1 and x_2
- There is neither *ground truth* nor labels on the data. The output variable is “*the class the data point belongs to*” but it’s not given in the training



- Note: is this a good clustering? It’s not easy to measure accuracy: without a *ground truth* the correctness is “in the eye of the beholder”

Interdisciplinarity

- One of the reasons of data mining's and machine learning's success are the virtually unlimited fields of **applications**:
 - Machine translation
 - Medical diagnosis
 - Recommendation systems (e.g. Amazon suggesting books)
 - Speech recognition
 - Information Retrieval (learning to rank)
 - Computer vision (e.g. self driving cars)
 - Cyber security (intrusion detection systems)
 - Trading
 - Weather forecast
 - ...

Data pre-processing

Data pre-processing

- Regardless of the specific goal, the pre-processing step is common to all data mining projects
- Why pre-processing? Real world data is generally:
 - **Incomplete**: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
 - E.g., occupation = ""
 - **Noisy**: containing errors or outliers
 - E.g., salary = -100
 - **Inconsistent**: containing discrepancies in code or names
 - E.g., Age = 30, Birthday = "03/07/2001"
 - E.g., Sex = Male, Pregnant = Yes
- The phrase "***garbage in, garbage out***" is particularly applicable in data mining



Pre-processing steps

- Main tasks of data pre-processing are:
 - **Data cleaning**
 - Fill in missing values, smooth noisy data, identify or remove outliers, resolve inconsistencies
 - **Data integration**
 - Integration of multiple databases, data cubes, or files
 - **Data transformation**
 - Machine learning algorithms work on the vector space: text and categories must be “encoded” into vectors of real numbers (e.g., bag of words, one-hot encoding)
 - Very large and very small values must be normalized, otherwise the optimization algorithms under machine learning models will not converge in reasonable time

Data cleaning

- We have seen its importance in **data warehousing**
 - It is said that data cleaning is the number one problem in data warehousing
- Similar issues are also faced in **data mining**
- Common data cleaning **tasks** in data mining are:
 - Fill in **missing** values
 - Identify **outliers** and smooth out noisy data
 - Correct **inconsistent** data
 - Resolve **redundancy** caused by data integration

Why is data dirty?

- **Incomplete/missing data** may come from:
 - “Not applicable” data value when collected
 - Different considerations between the time when the data was collected and when it is analyzed
 - Human/hardware/software problems
- **Noisy data** (incorrect values) may come from:
 - Faulty data collection instruments
 - Human or computer error at data entry
 - Errors in data transmission
- **Inconsistent data** may come from
 - Different data sources
(e.g., to establish the position of a geographic location on a map, common map projections in current use include the Universal Transverse Mercator, the Military Grid Reference System, the United States National Grid, the Global Area Reference System and the World Geographic Reference System)
 - Functional dependency violation (e.g., modify some linked data)
- **Duplicate records** also need data cleaning

Missing data

- Data is not always available
 - e.g., many tuples have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to
 - Equipment malfunction
 - Inconsistent with other recorded data and thus deleted
 - Data not entered due to misunderstanding
 - Certain data may not be considered important at the time of entry
 - Not register history changes of the data
- Missing data may need to be **inferred**

How to handle missing data?

- **Ignore** the tuple: usually done when class label is missing (assuming the tasks in classification) not effective when the percentage of missing values per attribute varies considerably
- Fill in the missing value **manually**: tedious, but could be even infeasible if the value is totally unknown
- Fill in it **automatically** with
 - A **global constant**: e.g., “unknown”, a new class?
 - The attribute **mean** (e.g., replace the missing house price with the average price of all houses)
 - The attribute **mean** for all samples belonging to the **same class** (e.g. replace the missing L.A.’s house price with the average price of the houses in L.A.)
 - The **most probable value**: inference-based such as regression (needs machine learning already)

Noisy data

- Noise: random error or variance in a measured variable
- Incorrect attribute values may due to
 - Faulty data collection instruments
 - Data entry problems
 - Data transmission problems
 - Technology limitation
 - Inconsistency in naming convention
- Other data problems which requires data cleaning
 - Duplicate records
 - Incomplete data
 - Inconsistent data

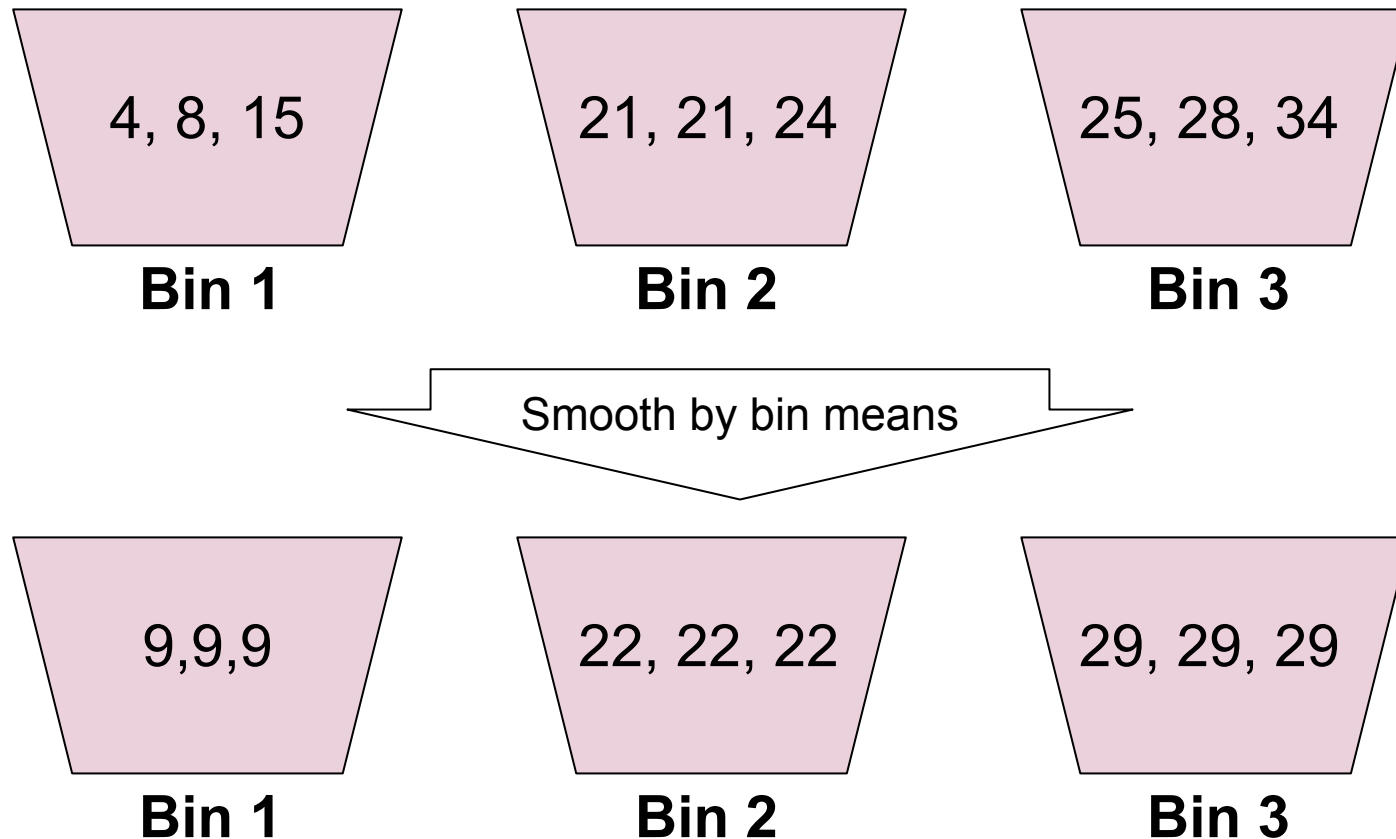


How to handle noisy data

- **Binning:** smooth a sorted data value by consulting its “neighborhood”, that is, the values around it
 - First sort data and partition into (equal-frequency) bins
 - Then we can smooth by bin means
- **Regression:**
 - Smooth by fitting the data into regression functions
- **Clustering:**
 - Detect and remove outliers
- Combined computer and human inspection
 - Detect suspicious values and check by human (e.g., deal with possible outliers)

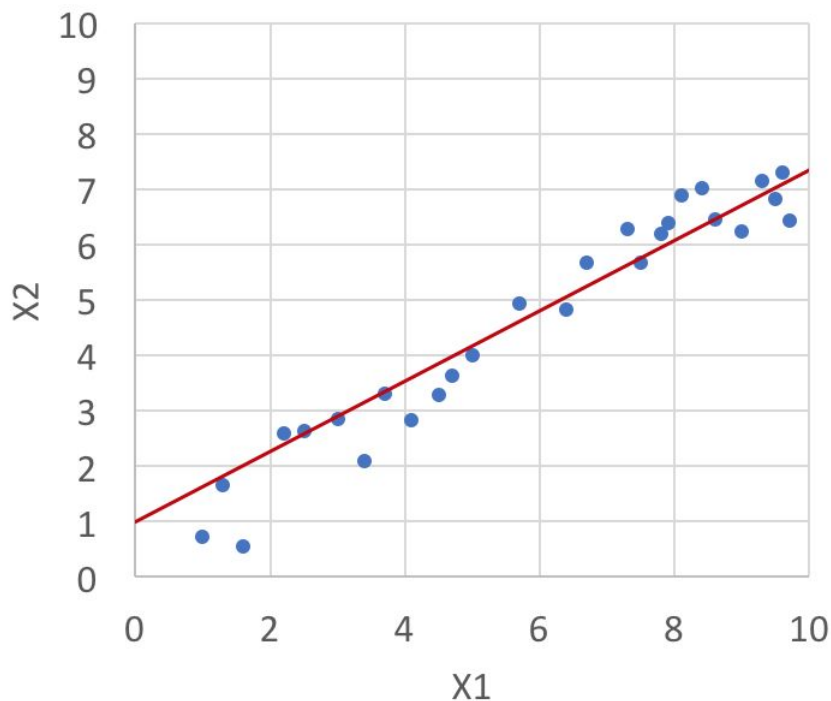
Denoising: Smoothing by bins

- Once the bins are populated, the values are substituted with aggregate values. A typical replacement is the means of the bin
- Because binning methods consider only the values in the same bin, they perform *local smoothing*

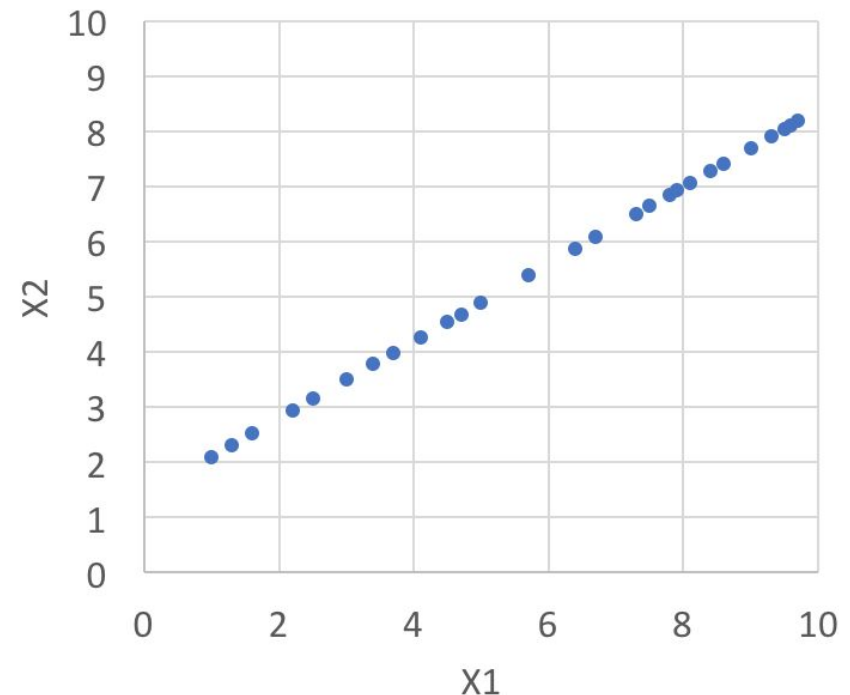


Denoising data: Regression

- Regression: linear regression involves finding the “best” **line** to fit two attributes (or variables) so that one attribute can be used to predict the other
- Regression could be the **final goal** of a data mining project, but it can also be used **a-priori** in the project in order to **smooth** the data



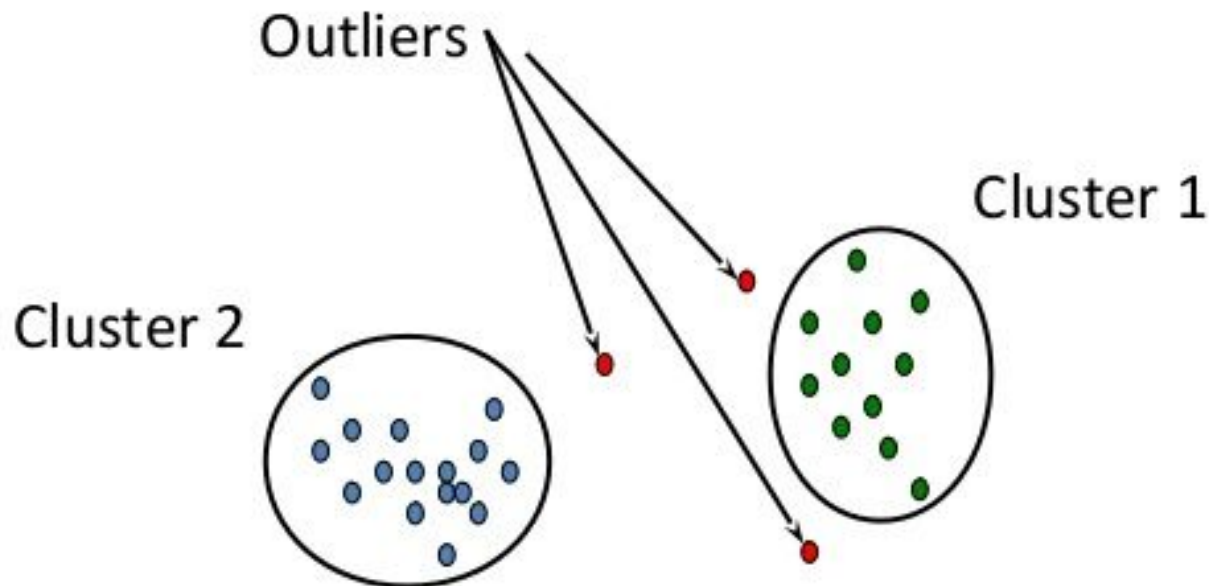
1) Regression



2) Smoothing, moving data points to the regression line

Denoising data: Clustering

- Outliers may be detected by clustering: similar values are organized into groups (clusters)
- Intuitively, **values that fall outside** of the clusters may be considered as outliers (anomalies)
- As for the linear regression method, clustering can be also the ultimate task of a data mining project, but could be applied in the preprocessing task to improve the quality of data



A note on noisy data

- When thinking of data noise we must keep in mind the ultimate tasks and their differences:
 - **Data warehousing:** next step is loading in the DW. Handling noisy data in the pre-processing phase is imperative because the decision maker will have no means of denoising data
 - **Data mining:** next step is machine learning. Machine learning models (even the most simple such as Logistic Regressions) have means of denoising data through “regularization” techniques
- **Practical suggestion:** build a first model without denoising, then reiterate training steps with pre-processing steps

Data integration

- Data mining often requires **data integration**: the merging of data from multiple data stores
- The semantic heterogeneity and structure of data pose great challenges in data integration
- How can we match schema and objects from different sources?
 - E.g., in the table A, the customer identification number is named A.cust_id, while in the table B, the customer identification number is called B.cust_num
- This problem is called the ***Entity Identification Problem***



Entity Identification Problem

- *Schema integration* and *object matching* can be tricky. Both tasks are affected by the **Entity Identification Problem**
 - For example, how can a data analyst or a computer be sure that *customer_id* on one database and *cust_number* in another refer to the same attribute?

Schema A

customer_id	birth	city
109238	07/12/87	Rome
113125	23/08/89	London
159483	28/11/90	Paris
198828	22/12/92	Bristol

Schema B

order_num	cust_num	cost
4982812389	113125	34.50
4982812390	113125	110.02
4982812391	151514	98.49
4982812392	129827	32.40



Entity Identification solution 1/2

- We can use the attribute **metadata**, for example:
 - **Name of the attribute**: we can use distances on string such as the edit distance to measure how much two names are similar
 - **Data type**: are the attributes both strings? Both dates?
 - **Range of values**: if one attribute contains values in the thousands and the other values between -1 and 1, we are probably looking at two different things

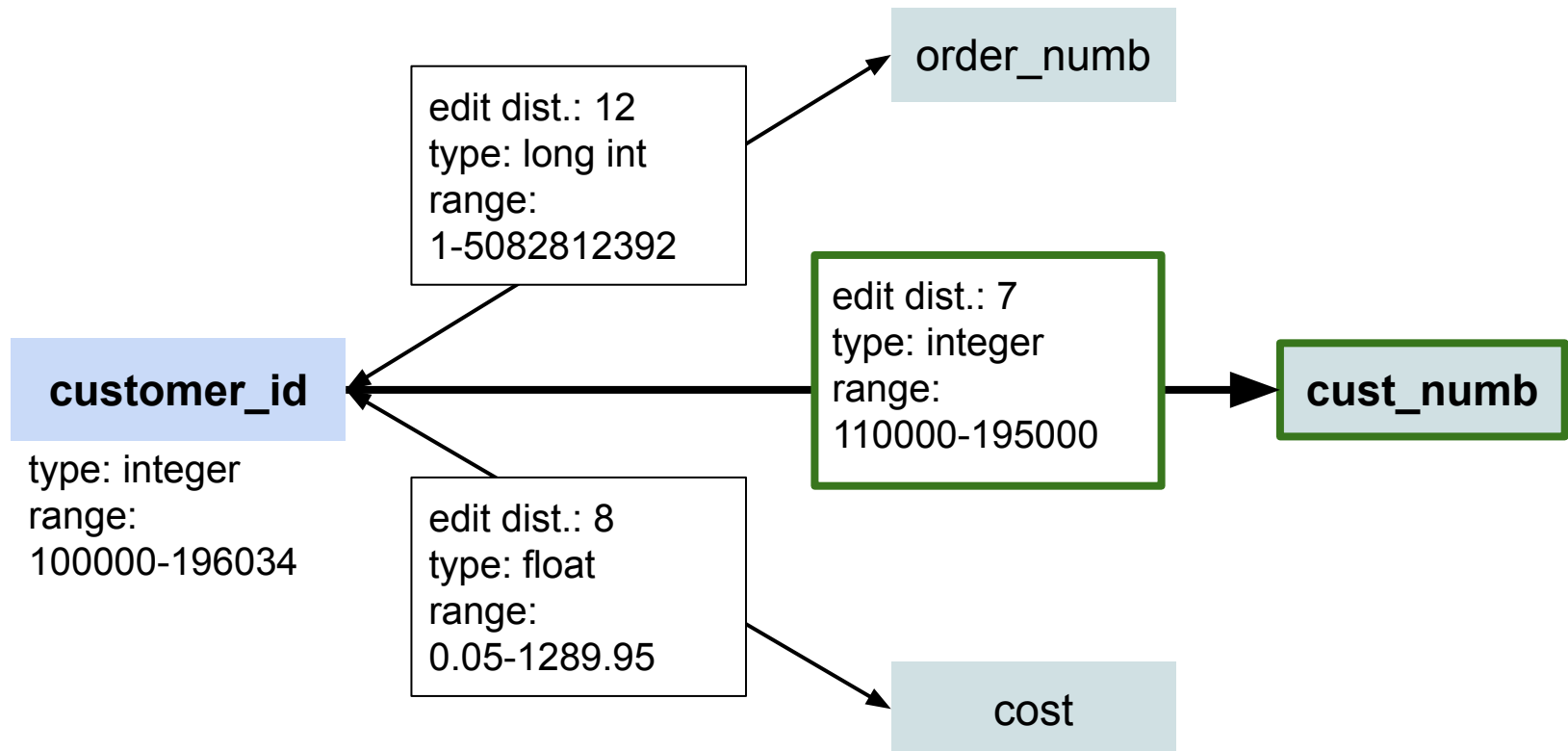
Schema A

customer_id	birth	city
109238	07/12/87	Rome
113125	23/08/89	London
159483	28/11/90	Paris
198828	22/12/92	Bristol

Schema B

order_num	cust_num	cost
4982812389	113125	34.50
4982812390	113125	110.02
4982812391	151514	98.49
4982812392	129827	32.40

Entity Identification solution 2/2

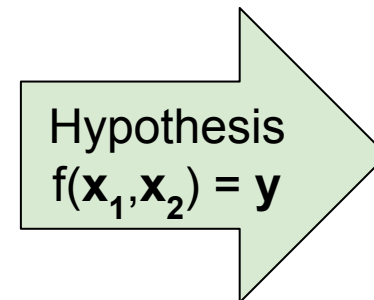


- We select `cust_num` because it has the **least edit distance**, **same data type** and the **most similar values range**
- Keep in mind that it is **not possible** to determine fully automatically the correspondences (with 100% accuracy)

Data transformation

- Recall: the hypothesis is the function modeled by machine learning algorithms, from the input variables to the output variable (e.g., a regression line that fits 2D data points)
- This function is a mathematical function, which takes **numerical values in input**, and output one or more numerical values

SquareMeters x_1	Bedrooms x_2
120	2
200	3
80	2
160	3
45	1
140	1



Price y
380,000
550,000
230,000
500,000
135,000
410,000



Why we need transformation

- **Encoding** problem: what if some variables of our data are **not numerical**?
 - It could be a **categorical** values such as the city
 - It could be a **boolean** yes/no value such as “hasGarage”
 - It could be a **free text** value such as “*Very nice cozy flat near the Centrum shopping mall, friendly neighborhood*”
- **Scaling** problem: very big numerical values are not good in ML
 - Machine learning models are based on **numerical optimization** techniques which minimizes the error between the hypothesis and the training data
 - Optimization algorithms **converge faster** with normalized values (e.g., between 0 and 1 or between -1 and 1)

Encoding solution for text

- We have seen in Information Retrieval how **free text** can be represented as a **vector of numerical variables**:
 - Each variable of the vector is for a specific word
 - The value of a variable x_i is 1 if the ***i*-th word** of the set of all words is present in the text, 0 otherwise
 - This representation is called “**bag-of-words**” (BOW)
 - This is a “**sparse**” representation: the resulting vector for a text has most of the variables set to 0, while only the words present in the text are set to 1

“Very nice flat with wonderful view”

...	<i>apartment</i>	<i>close</i>	<i>cozy</i>	<i>flat</i>	<i>floors</i>	<i>friendly</i>	<i>great</i>	<i>mall</i>	<i>view</i>	...
...	0	0	0	1	0	0	0	0	1	...

- This encoding process that goes from free text to a real-valued vector is called **vectorization**



Encoding solution for categories

- Encoding **categorical** variables is similar to bag-of-word encoding, with some differences:
 - Only **one category** at a time is true (e.g., a house cannot be both in N.Y. and in Miami, or cannot have a garage while not having a garage!)
 - No need to tokenize (split strings into words) or removing stop-words: each unique value will have its own variable (e.g., one variable for each city)

City: New York

...	<i>Albuquerque</i>	<i>Boston</i>	<i>New York</i>	<i>Seattle</i>	<i>Miami</i>	<i>Las Vegas</i>	<i>Chicago</i>	...
...	0	0	1	0	0	0	0	...

- This encoding process is called **one hot encoding**, because **only one** of the generated variables is active (“hot”) while all the other are set to 0

One Hot Encoding: an example

Human-Readable

Pet
Cat
Dog
Turtle
Fish
Cat



Machine-Readable

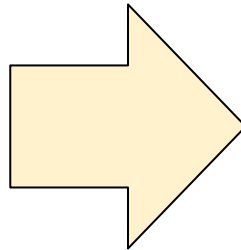
Cat	Dog	Turtle	Fish
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

Scaling - normalization

- The reasons why scaling is important in a data mining project will become more clear when we will see how machine learning models tune the hypothesis function to “fit” the data
- For now, let’s just say it **improves performance and accuracy** of most machine learning models
- Most common scaling range is [0,1], it can be obtained with linear scaling:

$$scale(value) = \frac{value - min(values)}{max(values) - min(values)}$$

SQM	Price
130	420000
60	80000
180	500000
140	220000



SQM	Price
0.583	0.810
0.000	0.000
1.000	1.000
0.666	0.333

Summary

- We've introduced the Data Mining process: it allows to extract new knowledge and patterns from large amount of data
- It consists of several steps, we have covered a preliminary step, which is the pre-processing phase:
 - **Cleaning data:** by removing or replacing erroneous or inconsistent values
 - **Denoising data:** by smoothing values toward central measures and removing outliers
 - **Integrating data:** by matching entities and attributes across different data sources
 - **Transforming data:** by encoding categorical and text data into numerical vectors, and scaling values into fixed ranges
- Data pre-processed, in the form of a **normalized matrix "X"**, of **m** examples and **n** variables, will be the input of machine learning algorithms

References

Data Mining Concepts and Techniques

Authors: Jiawei Han, Micheline Kamber, Jian Pei

