Contents

1. Data

1.1. Data types

Before data mining can be approached effectively, it is necessary to analyze the different types of data. This is because there's no one size fits all and different types of data require different approaches.

By **data**, or **data set** we mean a collection of **data objectes** together with their **attributes**. Data objects are a collection of attributes, capturing its characteristics in its entirety. Data objects are also referred to as: *record*, *point*, *case*, *sample*, *entity*, *instance*.

An attribute is a property or a characteristic of an object that can vary, either from one object to another or from one time to another. Attributes are also referred to as: *variable*, *field*, *characteristic*, *dimension*, *feature*.

There's a distinction between attributes and **attribute values**. The latter are the value or the symbol assigned to the former for a particular object. The same attribute can be matched to different attribute values. Interestingly, the properties of an attribute can differ from the properties of the attribute values associated to the same attribute.

The most intuitive way to arrange data is to use a table. The header of the table represent the attributes, whereas the cells contain the attribute values. The (i, j)-th cell contains the attribute value assigned to the j-th attribute for the i-th object. This requires that the objects are arranged according to a certain order.

Exercise 1.1.1: Arrange an example of a data set in a table.

Solution: The following data set contains information on students. Each row contains, in order: the id of the student, the year in which they are at the moment and their average score.

Student ID	Year	Grade Point Average (GPA)	...
...	...	...	...
1034262	Senior	3.24	...
1052663	Freshman	3.51	...
1082246	Sophomore	3.62	...

□

Attributes are classified into types with respect to the properties that they possess: **nominal**, **ordinal**, **interval**. **ratio**. This four types are arranged in a hierarchy. Each type inherits the properties and the supported operations of the previous one.

Nominal and ordinal attributes are collectively referred to as **categorical** or **qualitative** attributes. As the name suggests, lack most of the properties of numbers. Even when represented by numbers (IDs, for example), they should be treated as symbols. Interval and ratio attributes are collectively referred to as **quantitative** or **numeric** attributes. Quantitative attributes are represented by numbers and have most (if not all) of the properties of numbers.

Type		Description	Transformations	Statistics and tests
Categorical (qualitative)	Nominal	Values are just different names, they have no bearing on the chosen one	Checking whether two values are the same or not, any permutation or rearrangement	mode, entropy, contingency correlation, chi square test
	Ordinal	They can be any name as long as an order relation is preserved	Checking whether a value is greater or less than another, any monotonic function, $\text{new} = f(\text{old})$	median, percentiles, rank correlation, run tests, sign tests

Numeric (quantitative)	Interval	The differences between values are meaningful, they possess both a range and an order	Any scaling and or/stretching, $\text{new} = a \times \text{old} + b$ with a and b constants	mean, standard deviation, Pearson's correlation, t and F tests
	Ratio	Both differences and ratios are meaningful. They lie on an absolute scale and possess both a range and an order	Any scaling, $\text{new} = a \times \text{old}$ with a constant	geometric mean, harmonic mean, percent variation

Note that the properties above aren't necessarely the only ones that those possess. They are the ones that *every* attribute of the kind possesses, but there could be more. Also note the fact that an attribute type supports a certain transformation or statistic does not necessarely entail that computing said transformation or statistic is useful.

Exercise 1.1.2: Provide some examples of data for each type.

Solution:

- *Nominal*: zip codes, employee ID numbers, eye color, gender. If all employee ID numbers are reassigned, it will not make any difference.
- *Ordinal*: taste rankings, school grades, street numbers. Denoting school grades as {"poor", "average", "good"}, as $\{1, 2, 3\}$ or as $\{1, 4, 9\}$ has no difference.
- *Interval*: calendar dates, temperature in Celsius or Fahrenheit. The Fahrenheit and Celsius temperature scales differ in the location of their zero value and the size of a degree (unit). Scaling (say, doubling) or shifting (say, adding one) preserves the meaning.
- *Ratio*: temperature in Kelvin, monetary quantities, counts, age, mass, length, electrical current. Changing unit of measurement (scaling) preserves the meaning, whereas shifting (say, adding one) doesn't.

□

An "orthogonal" way of distinguishing between attributes is whether they are **discrete** or **continuous**, that is, with respect to the number of values that they can take.

Discrete attributes have a finite or countably infinite set of possible values, and are often represented as integer variables. A special case of discrete attributes are **binary** attributes, that can assume only two values (true/false, yes/no, 0/1, ecc...) and are often represented as boolean variables.

Continuous attributes have an uncountably infinite set of possible values, and are often represented as floating-point variables. Practically speaking, especially in the context of computer encoding, the granularity with which a continuous attribute can be measured is limited.

In principle, any nominal, ordinal, interval or ratio attribute can be either discrete or continuous, even though only some combinations are really useful (for example, it's hard to conceive an attribute that is both nominal and continuous). In general, nominal and ordinal attributes are either binary or discrete, whereas interval and ratio attributes are continuous.

Asymmetric attributes are attributes where only the "presence" of a value (a non-null value) is worth recording (for example, words in a document: keeping track of unused words is worthless).

1.2. Dataset types

It is useful to denote three characteristics that every dataset possesses and that have an impact on which data mining technique is used: **dimensionality**, **distribution**, **resolution**.

The dimensionality of a data set is the number of attributes that the objects in the data set possess (not the amount of objects that constitute it). Analyzing data with a small number of dimensions tends to be different from analyzing data with many dimensions. This issue of scalability with respect to dataset dimensions has its own name: *curse of dimensionality*. For this reason, it is often useful to apply techniques of **dimensionality reduction**.

The distribution of a data set is the frequency of occurrence of various values or sets of values for the attributes comprising data objects. Even though many distribution structures are fairly common (gaussian, ecc...) it isn't always possible to employ them to capture every detail of any dataset. This is why data mining algorithms strive to be distribution-agnostic, that is not depending on a specific distribution in order to accomplish their goals.

However, some general aspects of distributions often have a strong impact, such as their **sparsity**. A data set is considered sparse if most attribute values are null values (in the context of numerical attributes, null attribute values are 0s). When attributes are asymmetric, sparsity is a pleasant feature to have since, even though the dataset might consist of many objects, most attribute values do not need to be taken into account. Indeed, some data mining algorithms have a feasible execution time only if the dataset is sparse.

The resolution of a data set is to the level of detail, of "granularity", with which its attribute values are obtained. If the resolution is too fine, a pattern may not be visible or may drown in noise; if the resolution is too coarse, the pattern can disappear. For example, variations in atmospheric pressure on a scale of hours reflect the movement of storms and other weather systems. On a scale of months, such phenomena are not detectable.

Datasets can be grouped with respect to the way that the data is arranged. One possible (yet partial) way of classifying data sets in this regard is to denote it as **record data**, **graph data** or **ordered data**.

Record data is the most intuitive way of arranging data: tables with the attributes as headers and the attribute values stored in the cells. In contrast to how, say, relational databases work, there's no notion of a key (two objects with the same attribute values are not the same object) and there is no relationship between multiple records (each record stands for itself). Also, each record ought to have the same set of attributes, even though some of those values might be unknown.

A special kind of record data is **transaction data**, where each record (transaction) involves a set of items. This type of data is also called **market basket data** because the items in each record can be thought of as the items put in a person's basket at the mall. Transaction data can also be viewed as a set of records whose fields are asymmetric attributes, that is, as the existence of a certain item in each transaction.

If all the attributes in a data set are numerical, it is possible to conceive each data object as a point in a multidimensional space, where each dimension represents a distinct attribute. Therefore, the table structure of record data can be nicely coerced into a $n \times m$ matrix, where each of the m rows represents an object and each of the n columns represents an attribute (or vice versa). This matrix is called a **data matrix** or a **pattern matrix**; being a matrix, it is possible to easily manipulate its data using matrix operations (like column-wise product, for example).

A **sparse data matrix** is a special case of a data matrix where the attributes are of the same type and are asymmetric. As a matter of fact, transaction data is an example of a sparse binary data matrix.

Exercise 1.2.1: Provide examples for each type of record data set.

Solution:
- A generic record data was already introduced in Exercise 1.1.1;
- Transaction data can be approached considering a grocery store. The set of products purchased by a customer in a single shopping trip constitutes a transaction, whereas the individual products that they purchased are the items;
- Any matrix of real values can act as a data matrix;
- A common example of sparse data matrix is document data. Matrices such as these, called **document-term matrices**, contain the number of occurrences of each word in each document. Such attributes are indeed asymmetric, since it is irrelevant to keep track of the absence of a word (i.e. if its occurrency is 0).

id	Transaction
1	Bread, Soda
2	Bread, Soda, Milk
3	Beer, Bread

Length	Width	Depth	Capacity	Pressure
7.21	9.53	9.3	8.66	6.2
4.10	9.56	3.77	6.95	0.39
1.10	7.18	9.28	0.99	7.53

4	Soda, Chocolate, Milk
5	Soda, Milk

	Car	House	Dog	Book	Chair
Document 1	1	7	0	0	0
Document 2	0	0	7	0	0
Document 3	0	4	7	1	1

□

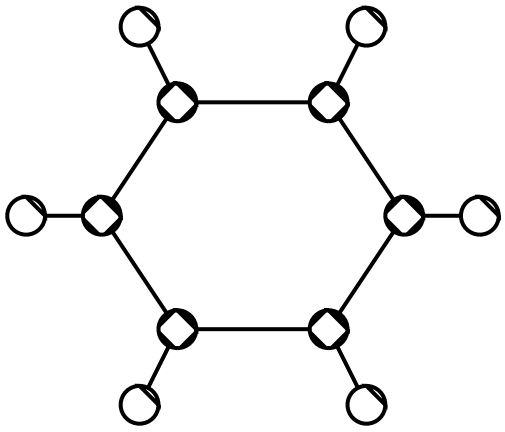
Data arranged into a graph allows to have a more flexible, yet harder to manipulate, data structure. This is particularly useful when there's interest not only in keeping track of the object attributes, but also in the relationship that occurs between objects. That is, when the references between objects can tell something about the data that the objects on their own cannot.

In its most natural form, graph data represents data objects as nodes of the graph and relationships between objects as edges. Traits of said relationships can be encoded as properties of the edges (weight, direction, ecc...). It is also sometimes useful to encode objects as graphs when objects are composite, having subobjects and relationships among each component.

Exercise 1.2.2: Provide examples for each type of graph data set.

Solution:

- Web pages are often represented as graphs, encoding each web page as a node and the existence of a link that allows to move from one web page to another as an edge. This is because the existence of a link that connects two web pages can, for example, change the priority of a search engine when indexing results;
- The structure of chemical compounds can be represented by a graph, where the nodes are atoms and the edges are chemical bonds. A graph representation makes it possible to determine which substructures occur frequently in a set of compounds and to ascertain whether the presence of any of these substructures is associated with the presence or absence of certain chemical properties, such as melting point or heat of formation.



□

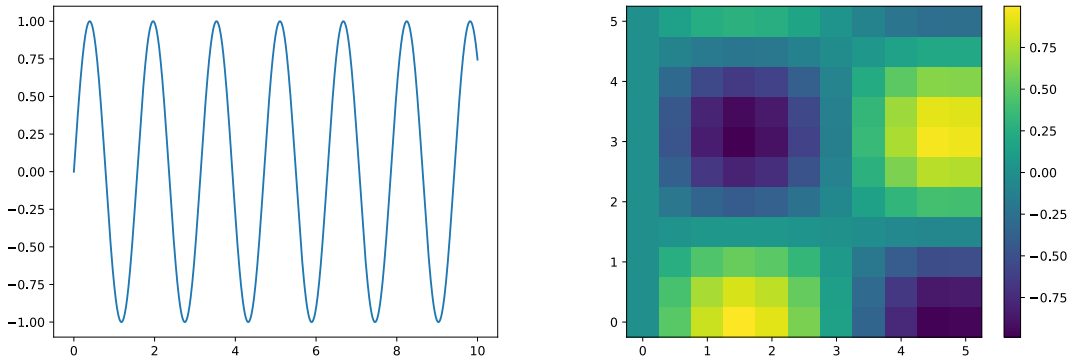
Ordered data is data where there's interest in keeping track of the *order* that each object occupies with respect to the others, be it temporal or spacial. Examples of ordered data include:

- **Sequential transaction data**, which can be thought of as transaction data, extended with a timestamp;
- **Time series data**, a series of measurements of the same phenomena taken in different periods of time. An important aspect of time series data is **time autocorrelation**: measurements taken in time instants close to each other tend to give similar results.
- **Sequence data**, sequences of individual entities where there's interest in the order in which they are arranged;

- **Space data**, a series of measurements of the same phenomena taken in different points in space. An important aspect of space data is **space autocorrelation**: objects that are close to each other (have similar coordinates) tend to also have similar attribute values.

Exercise 1.2.3: Provide examples for each type of ordered data set.

Solution:
- Sequential transaction data can extend the example presented in Exercise 1.2.1 by introducing the time attribute. In this way, it could be possible to investigate relationships, for example, between purchased items and time of the day in which they bought;
- Stock prices over time can be modelled as time series data. it is reasonable to assume that the price of the same stock in consecutive time frames will not change drastically;
- The genetic information of plants and animals can be represented as sequence data in the form of strings on the alphabet $\{A, C, G, T\}$. The order in which they are arranged is indeed the most relevant aspect of the data, since it determines which proteins are built;
- Weather data (precipitation, temperature, pressure) can be represented as spatial data, often displayed with the help of heatmaps. It is indeed to be expected that locations on Earth close to each other will have a similar, for example, average temperature during the day.



□

1.3. Data quality

It is unrealistic to expect that raw data will be perfect, and that it is usable as is. The spuriousness of data can be due to a multitude of reasons, such as:

- Human error;
- Limited precision of measuring devices;
- Biased data collection (the sample does not properly represent its population);
- Missing information (objects have one or more unknown attribute values);
- Duplicate information (an object with the same or almost the same attribute values appears more than once).

The measure of how usable a dataset is is called **data quality**. The approach that data mining takes in tackling poor data quality is twofold: detecting and correcting spurious data objects and/or attribute values so that raw data becomes manageable and designing flexible algorithms that can tolerate imperfections. Preprocessing (raw) data in order to increase its quality is referred to as **data cleaning**.

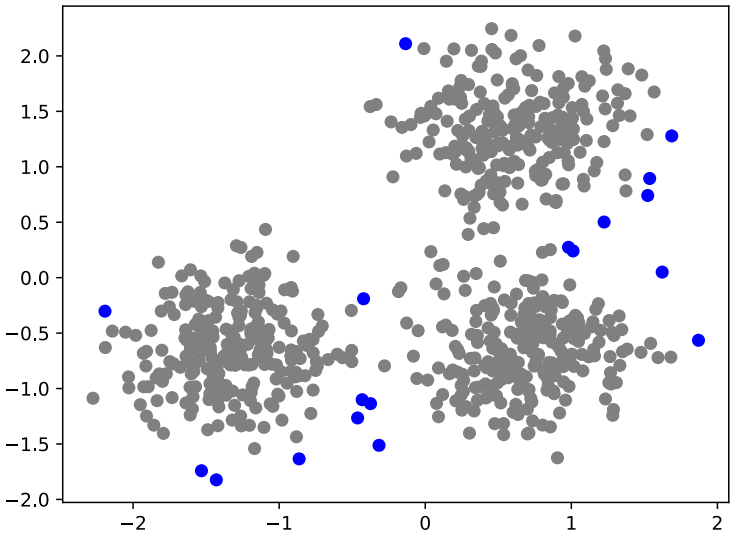
The term **measurement error** refers to any problem resulting from the measurement process. The most commonplace measurement error is a recorded attribute value that differs from its true value. For continuous attributes, the difference between the measured value and true value is called the **error**. The term **data collection error** refers instead to errors such as omitting data objects or attribute values, or inappropriately including a data object that should not be there.

Noise is the random component of a measurement error. With respect to objects, it entails the addition of an object to a dataset where it should not belong, whereas with respect to attribute values it entails a distortion of its original value. Eliminating noise completely is impossible; the only realistic approach is to design algorithms that are noise-tollerant.

Outliers are either data objects whose attribute values differ noticeably with respect to the others or attribute values that differ noticeably from their mean. They differ from noise in the sense that they are legitimate data objects, and therefore their presence in the dataset is justified and a model must account for them. In particular, there are situations where outliers are the interest of the analysis itself (**anomaly detection**).

Exercise 1.3.1: Put outliers from a sample dataset under the spotlight.

Solution: The following dataset has been created using the `make_blobs` function from the Python package `scikit-learn`. The blue dots are the data objects that an algorithm (`DBSCAN`) has classified as outliers.



□

In addition to being spurious, data can also be lackluster. It is therefore expected that some attribute values will be missing. This could be due to, for example, lack of knowledge or not all attributes being applicable to all objects. Regardless, missing values must be taken into account during data analysis, and there are different strategies to accomplish this goal.

One approach consists in discarding all objects that have one or more missing attribute values or, even more drastically, discard all attributes that have one or more missing attribute values. Even though this approach is easy, it might backfire, since even partially known data objects can be valuable for an analysis, and removing them might prevent seeing the bigger picture. Therefore, the approach makes sense only if the missing attribute values are a small fraction of the total (it should be noted that in real datasets this is hardly the case).

A more refined approach consists in estimating missing values from the known ones. This is reasonable when the values vary smoothly, or when the values are concentrated around the average. If the attribute is continuous, then the average attribute value of the nearest neighbors is used; if the attribute is categorical, then the most commonly occurring attribute value can be taken. In some situations, the missing values can simply be ignored, albeit the accuracy of the model might suffer.

The last issue regarding data quality is the presence of duplicate, or almost duplicate, objects. This issue often arises when merging data coming from different sources but that refer to the same entities, where the same

entity appears as object in many sources but its attribute values are slightly different. The process of unifying two or more objects into a single one, so that an entity is referred to just once, is called **entity linking**.

If there are two objects that actually represent a single object, then one or more values of corresponding attributes are usually different, and these inconsistent values must be resolved. This can be done either averaging the two values (with numerical attributes) or picking one of the two. Care should be taken in order to avoid merging two similar objects that represent two different entities (for example, two distinct people having the same name or two transactions containing the same items). It should be noted that, in some circumstances, two or more objects with the same attribute values for each attribute refer to different entities, and therefore the presence of a duplicate is legitimate.