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

□

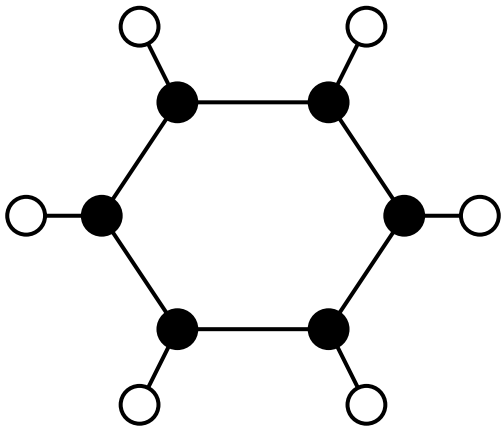
Data arranged into a graph allows to have a more flexible, yet harder to manipulate, data structure. This is particularly useful when there's interest not only in keeping track of the object attributes, but also in the relationship that occurs between objects. That is, when the references between objects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



□

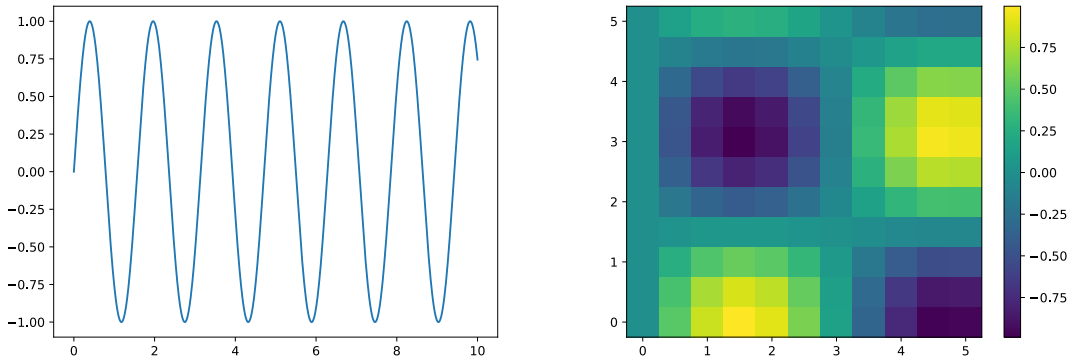
Ordered data is data where there's interest in keeping track of the *order* that each object occupies with respect to the others, be it temporal or spacial. Examples of ordered data include:

- **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**.

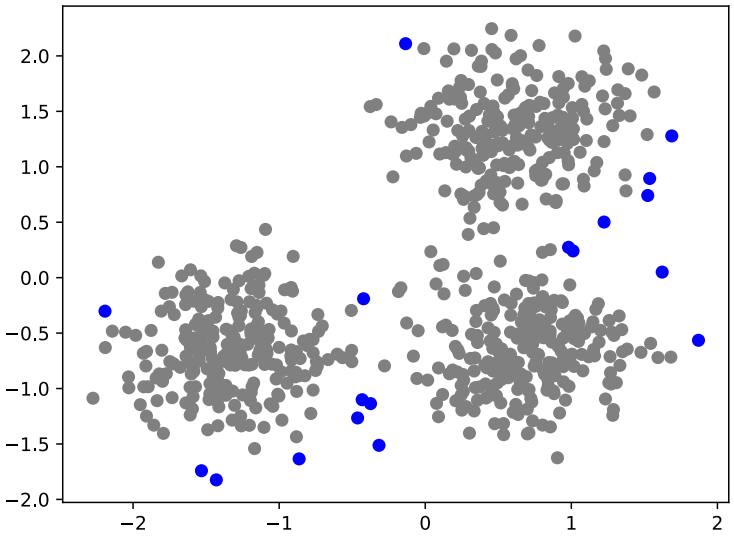
The term **measurement error** refers to any problem resulting from the measurement process. The most commonplace measurement error is a recorded attribute value that differs from its true value. For continuous attributes, the difference between the measured value and true value is called the **error**. 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.

1.4. Proximity measures

Informally, the **similarity** between two data objects is a numerical measure of the degree to which the two data objects are alike. Consequently, the similarity of two data objects that are more alike will be higher, whereas the similarity of two data objects that are less alike will be lower. Similarities often fall in the interval $[0, 1]$, where 0 means no similarity and 1 means complete similarity.

The **dissimilarity** between two objects is a numerical measure of the degree to which the two objects are different. Dissimilarity of two data objects that are more different will be higher, whereas the similarity of two data objects that are less different will be lower. Dissimilarities are generally bounded by 0 on the left, whereas the right boundary varies (common choices are 1 and $+\infty$). Similarity and dissimilarity are together referred to as **proximity**.

Proximity measures can be converted into one another (that is, a similarity into a dissimilarity or vice versa), or can be normalized so that they fall into a chosen range. This is useful, for example, when a certain algorithm can only work with specific proximity measures.

Suppose that there is interest in normalizing a proximity measure d, defined on the interval $[\min_d, \max_d]$, so that it fits in the interval $[0, 1]$. If $\min_d$ and $\max_d$ are both finite, a proximity measure $d' \in [0, 1]$ can be obtained as:

$$d' = \frac{d - \min_d}{\max_d - \min_d}$$

This is an example of a linear transformation, which preserves the relative distances between points.

Note that mapping proximity measures to a different interval by applying a linear transformation might entail a loss of meaning. For example, if $\max_d = +\infty$, the new proximity measure d' will have larger values "compressed" to 1, which isn't necessarely the desired outcome. Even worse, suppose that $d \in [-1, 1]$; if the range is mapped to $[0, 1]$ by taking the absolute value, information of the sign is lost, which may be important for some applications.

Transforming similarities to dissimilarities and vice versa is also relatively straightforward, although the issue of preserving the meaning is still present. If a similarity s is mapped to $[0, 1]$, it can be changed to a dissimilarity d mapped to the same interval as $d = 1 - s$. Vice versa, a dissimilarity $d \in [0, 1]$ can be changed to a similarity $s \in [0, 1]$ as $s = 1 - d$. Another approach is to change a similarity into a dissimilarity or vice versa by simply changing its sign.

In general, any monotonic decreasing function can be used to convert dissimilarities to similarities, or vice versa. Of course, other factors also must be considered when transforming similarities to dissimilarities, or vice versa, or when transforming the values of a proximity measure to a new scale. Preservation of meaning, distortion of scale, and requirements of data analysis tools have been mentioned, but this list is certainly not exhaustive.

The proximity of objects is typically defined by combining the proximities of individual attributes. Therefore, the simplest situation to model is the one where each object is described by a single attribute. The similarity and dissimilarity of an attribute can be computed by exploiting the comparison operators that its class provides.

Nominal attributes only support checking whether two are the same or not. Therefore, the only thing that can be said about two nominal attribute values is whether they have the same value (maximum similarity, minimum dissimilarity) or whether their value differs (minimum similarity, maximum dissimilarity). It is customary to

define similarity between two nominal attribute values as 1 if they are equal and 0 if they are not, and dissimilarities as 1 if they are different and 0 if they are not.

$$d(x,y) = \begin{cases} 0 \text{ if } x = y \\ 1 \text{ if } x \neq y \end{cases} \qquad s(x,y) = \begin{cases} 1 \text{ if } x = y \\ 0 \text{ if } x \neq y \end{cases}$$

Ordinal attributes support checking for equality but also support ordering. Therefore, it is possible to define a notion of "closeness" by mapping each value to consecutive integers starting from 0 or 1. Dissimilarity d can then be defined as the difference between the corresponding values, normalized to $[0, 1]$ with respect to the number of attributes n. Similarity s can be defined from dissimilarity as $s = 1 - d$. Note that this definition entails that the "steps" between each possible value of the attribute are of equal size, but this isn't necessarely the case.

$$d(x,y) = \frac{|x-y|}{n-1} \qquad s(x,y) = 1 - d(x,y)$$

For interval or ratio attributes, the natural measure of dissimilarity between two objects is the absolute difference of their values. This makes sense because both attributes support arbitrary scaling. The similarity of interval or ratio attributes is typically expressed by transforming a dissimilarity into a similarity, as stated previously, by applying any monotonic decreasing function.

$$d(x,y) = |x-y| \qquad s(x,y) = -d(x,y) \text{ or } 1 + \frac{1}{d(x,y)} \text{ or } e^{d(x,y)} \text{ or } \frac{d(x,y) - \min_d}{\max_d - \min_d} \text{ or } \ldots$$

1.4.1. Dissimilarities

The terms "dissimilarity" and "**distance**" are often used interchangeably, even though the latter is a special case of the former. In particular, a distance is defined as any function $d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ that possesses (at least) the following properties:

- $d(x,y) \geq 0$ for all x, y (non negativity);
- $d(x,y) = 0$ if and only if $x = y$;
- $d(x,y) = d(y,x)$ for all x, y (simmetry);
- $d(x,z) \leq d(x,y) + d(y,z)$ for all x, y, z (triangular inequality).

Measures that satisfy all three properties are known as **metrics**. Some sources use the term "distance" only for dissimilarity measures that satisfy these properties, but that practice is often violated.

Given a distance function d and a data set containing m objects, it is possible to construct a **distance matrix** as the $m \times m$ matrix $D[i,j]$ where each cell (i,j) contains the distance between the i-th element of the dataset and the j-th element:

$$D[i,j] = \begin{pmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \ddots & \ddots & \\ d(m,1) & d(m,2) & \ldots & \ldots & 0 \end{pmatrix}$$

The matrix has 0s along the diagonal because of the first property of distances. It also symmetric, due to the third property of distances, hence why only half the matrix is depicted.

The most intuitive notion of distance is the **Euclidean distance**, defined as:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Where n is the number of dimensions and x_i, y_i are the i-th attribute value of objects x and y respectively.

A somewhat simpler distance is the **Manhattan distance** (also known as **taxicab distance** or **city block distance**), defined as:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} |x_i - y_i|$$

Related to the Manhattan distance is the **Hamming distance**, given by the number of elements that are equal between two vectors:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} \delta_{i,j} \text{ where } \delta_{i,j} = \begin{cases} 1 \text{ if } x_i = y_i \\ 0 \text{ otherwise} \end{cases}$$

In the case of binary vectors, this is simply equivalent to the `XOR` between the two.

Exercise 1.4.1.1: Compute three distance matrices of the following dataset, one for each distance (Euclidean, Manhattan, Hamming):

	X	Y
E1	0	2
E2	2	0
E3	3	1
E4	5	1

Solution:

$$\underbrace{\begin{pmatrix} 0 & & & \\ 2.83 & 0 & & \\ 3.16 & 1.41 & 0 & \\ 5.10 & 3.16 & 2.00 & 0 \end{pmatrix}}_{\text{Euclidean}} \quad \underbrace{\begin{pmatrix} 0 & & & \\ 4 & 0 & & \\ 4 & 2 & 0 & \\ 6 & 4 & 2 & 0 \end{pmatrix}}_{\text{Manhattan}} \quad \underbrace{\begin{pmatrix} 0 & & & \\ 2 & 0 & & \\ 2 & 2 & 0 & \\ 2 & 2 & 1 & 0 \end{pmatrix}}_{\text{Hamming}}$$

□

Both the Manhattan and the Euclidean distance are generalized by the **Minkowski distance**:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt[R]{\sum_{i=1}^{n} \left(|x_i - y_i|\right)^R}$$

Where the parameter R influences the nature of the result. In particular:

- When $R = 1$, the Minkowski distance reduces to the Manhattan distance (for this reason, also called **L1 norm**);
- When $R = 2$, the Minkowski distance reduce to the Euclidean distance (for this reason, also called **L2 norm**);
- When $R \to +\infty$, the resulting distance is called the **supremum distance**, also known as the **L_∞ norm** or the **L_{max} norm**. This is given by the maximum difference between any attribute of the objects:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \lim_{R \to +\infty} \sqrt[R]{\sum_{i=1}^{n} \left(|x_i - y_i|\right)^R} = \arg\max_i (x_i - y_i)$$

Note that the Minkowski distance (and, by extension, all the distances from its family) are not invariant with respect to *scaling* and to *shifting*. That is, if $\boldsymbol{x}$, $\boldsymbol{y}$ or both are modified by multiplying its value by a costant and/or by adding a constant the new objects do not have the same value of the Minkwoski distance as they had before.

It should also be noted that the Minkowski distance and its derivatives treat each attribute equally relevant in the computation of the distance. Sometimes, there are situations where one attribute should have more influence on the final result than others. The problem can be solved by introducing, for each i-th attribute, a weight w_i in the computation of the Minkowski distance:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt[R]{\sum_{i=1}^{n} w_i (|x_i - y_i|)^R}$$

1.4.2. Similarities

For similarities, the triangle inequality (or the analogous property) typically does not hold, but symmetry and positivity do. To be explicit, if $s(x, y)$ is the similarity between objects x and y, then the typical properties of similarities are:

- $0 \leq s(x, y) \leq 1$ for all x, y;
- $s(x, y) = 1$ if and only if $x = y$;
- $s(x, y) = s(y, x)$ for all x, y (simmetry);

There is no general analog of the triangle inequality for similarity measures. It is sometimes possible, however, to show that a similarity measure can easily be converted to a metric distance. Also, for specific similarity measures, it is possible to derive mathematical bounds on the similarity between two objects that are similar in spirit to the triangle inequality.

Similarity measures between objects that contain only binary attributes are called **similarity coefficients**, and typically have values between 0 and 1. A value of 1 indicates that the two objects are completely similar, while a value of 0 indicates that the objects are not at all similar. There are many rationales for why one coefficient is better than another in specific instances.

Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two objects that consist of n binary attributes. It is possible to define four frequencies as follows:

$f_{00} =$ number of attributes where x is 0 and y is 0 $\qquad$ $f_{01} =$ number of attributes where x is 0 and y is 1
$f_{10} =$ number of attributes where x is 1 and y is 0 $\qquad$ $f_{11} =$ number of attributes where x is 1 and y is 1

An example of similarity coefficient is **simple matching coefficient** (SMC), defined as:

$$\text{SMC} = \frac{\text{number of matching attribute values}}{\text{number of attributes}} = \frac{f_{11} + f_{00}}{f_{01} + f_{10} + f_{11} + f_{00}}$$

Another similarity coefficient is the **Jaccard coefficient** (J), defined as:

$$\text{J} = \frac{\text{number of non-zero matching attribute values}}{\text{number of non-zero attributes}} = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$$

The difference between the two coefficients is the way that they handle 0-valued attributes. The simple matching coefficient treats both 0 and 1 values equally, and therefore is a reasonable choice when both values are of interest. On the other hand, the Jaccard coeffcient only takes into account 1-valued attributes, and is therefore useful for asymmetric attributes where 0 values are irrelevant.

Exercise 1.4.2.1: Compute both the simple matching coefficient and the Jaccard coefficient for the following binary vectors:

$$x = (1, 0, 0, 0, 0, 0, 0, 0, 0) \qquad y = (0, 0, 0, 0, 0, 0, 0, 0, 1)$$

Solution: Note that:

$$f_{00} = 7 \qquad f_{01} = 2 \qquad f_{10} = 1 \qquad f_{11} = 0$$

Therefore:

$$\text{SMC} = \frac{0 + 7}{2 + 1 + 0 + 7} = 0.7 \qquad \text{J} = \frac{0}{2 + 1 + 0} = 0$$

□

Document-term matrices tend to be very sparse, since it's much more likely for two documents to not share words than for them to be sharing words. Therefore, data such as these need a similarity coefficient that does

not take into account 0-valued attributes, because otherwise the results would be very spurious, but that also works for attributes that aren't binary-valued.

A similarity coefficient having both of these characteristics is the **cosine similarity**. Given two document vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, the cosine similarity between the two is given by:

$$\cos(\boldsymbol{x}, \boldsymbol{y}) = \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{||\boldsymbol{x}|| \ ||\boldsymbol{y}||}$$

Where $\langle\rangle$ denotes the scalar product and $||\ ||$ denotes the norm. The cosine similarity is invariant with respect to scaling, but not with respect to shifting.

The name "cosine" comes from the fact that the cosine similarity between two document vectors is simply the cosine of the angle formed by the two vectors if represented in an hyper-plane. If the cosine similarity of two document vectors is 1, the two are parallel, and therefore share every word, whereas if its 0 the two are perpendicular and therefore share no word.

Exercise 1.4.2.2: Compute the cosine similarity of these two document vectors:

$$\boldsymbol{x} = (3, 2, 0, 5, 0, 0, 0, 2, 0, 0) \qquad \boldsymbol{y} = (6, 4, 0, 10, 0, 0, 0, 4, 0, 0)$$

Solution:

$$\cos(\boldsymbol{x}, \boldsymbol{y}) = \frac{\langle (3, 2, 0, 5, 0, 0, 0, 2, 0, 0), (6, 4, 0, 10, 0, 0, 0, 4, 0, 0) \rangle}{||(3, 2, 0, 5, 0, 0, 0, 2, 0, 0)|| \ ||(6, 4, 0, 10, 0, 0, 0, 4, 0, 0)||} = \frac{84}{84} = 1$$

This means that the two share every word. This can also be hinted at by the fact that the second vector is just two times the first. □

Correlation is frequently used to measure the linear relationship between two sets of values that are observed together. Even though correlation is generally computed with respect to datasets, it's still reasonable to compute it with respect to two objects, treating each attribute as the value of a population. The term "correlation" is sometimes used to mean the relationship between two sets of values that are observed together, even though correlation is one of the possible measures of the latter.

The most common measure of correlation is the **Pearson correlation coefficient**, defined as:

$$\rho = \operatorname{Corr}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\operatorname{Cov}(\boldsymbol{x}, \boldsymbol{y})}{\operatorname{SD}(\boldsymbol{x}) \ \operatorname{SD}(\boldsymbol{y})} = \frac{S_{\boldsymbol{xy}}}{S_{\boldsymbol{x}} S_{\boldsymbol{y}}}$$

Where $\boldsymbol{x}$ and $\boldsymbol{y}$ are two n-dimensional vectors, $\operatorname{Cov}(\boldsymbol{x}, \boldsymbol{y})$ is the covariance between the two and $\operatorname{SD}(\boldsymbol{x})$ and $\operatorname{SD}(\boldsymbol{y})$ are the respective variances. These quantities can be explicited as:

$$S_{\boldsymbol{xy}} = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y}) \qquad S_{\boldsymbol{x}} = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2 \qquad S_{\boldsymbol{y}} = \frac{1}{n-1} \sum_{i=1}^{n} (y_i - \overline{y})^2$$

The value of the coefficient is always in the interval $[-1, 1]$. The closer the value is to ± 1, the better the data is approximately linearly distributed. If the value of the coefficient is close to 0, the worse the distribution of the data can be approximated by a linear function.

Note that, if $\rho \approx 0$, it does not necessarely mean that the two have no correlation; they might, but it's not linearly distributed. A graphical inspection can be obtained through a scatterplot. Is should also be stressed that correlation does not imply causation.

An interesting property of the Pearson coefficient is that it is invariant both with respect to scaling and with respect to shifting.

Exercise 1.4.2.3: Compute the Pearson correlation coefficient between these two vectors:

$$\boldsymbol{x} = (1, 2, 4, 3, 0, 0, 0) \qquad \boldsymbol{y} = (1, 2, 3, 4, 0, 0, 0)$$

Solution: First of all, note that:

$$\overline{x} = \overline{y} = \frac{1 + 2 + 3 + 4 + 0 + 0 + 0}{7} \approx 1.43$$

Then:

$$S_{\boldsymbol{xy}} = \frac{\sum_{i=1}^{7}(x_i - 1.43)(y_i - 1.43)}{6} \approx 2.45 \qquad S_{\boldsymbol{x}} = S_{\boldsymbol{y}} = \frac{\sum_{i=1}^{7}(y_i - 1.43)^2}{6} \approx 1.62$$

Which gives:

$$\rho = \frac{S_{\boldsymbol{xy}}}{S_{\boldsymbol{x}} S_{\boldsymbol{y}}} = \frac{2.45}{1.62 \cdot 1.62} \approx 0.93$$

□

When a linear relationship fails to describe similarity between two objects, an alternative is provided by information theory in the form of **mutual information**. Let X be a discrete random variable, whose possible values are $\{x_1, ..., x_n\}$. The **entropy** of X is defined as:

$$H(X) = -\sum_{i=1}^{n} P(X = x_i) \log_2 P(X = x_i)$$

Where $P(X = x_i)$ denotes the probability that the random variable X takes the value x_i.

Information relates to possible outcomes of an event. The information of an outcome is inversely proportional to its certainty: if an event is certain, it has zero information. Entropy denotes the amount of uncertainty is given by a random variable; when the entropy is 0, there's absolute certainty.

Entropy can be extended to data sets by computing it on the possible attribute values of its data objects. Consider two sets of values, X and Y, which occur in pairs (X, Y). Assume that X and Y are discrete, that is, X can take m distinct values $\{x_1, x_2, ..., x_m\}$ and Y can take n distinct values $\{y_1, y_2, ..., y_n\}$. The **joint entropy** of X and Y is given by:

$$H(X, Y) = -\sum_{i=1}^{n} P(X = x_i \land Y = y_i) \log_2 P(X = x_i \land Y = y_i)$$

Mutual information is used to measure the similarity between two sets of paired values. It's a measure of how much information one set of values provides about the other. If two sets of values are independent the mutual information is 0, because one does not tell anything about the other.

$$I(X, Y) = H(X) + H(Y) - H(X, Y)$$

The previous definitions of similarity were presented under the assumption that all attributes were of the same types; in a more realistic scenario, this isn't the case. It is possible to circumvent the problem by computing the similarity between each attribute separately, using measures appropriate for their respective type, and then combining them together.

The most intuitive approach in this direction consists in computing the average of the similarities between each attribute. Unfortunately, this approach does not work well if some of the attributes are asymmetric. The easiest way to fix this problem is to omit asymmetric attributes from the similarity calculation when their values are 0 for both of the objects whose similarity is being computed. A similar approach also works well for handling missing values.

Let $\boldsymbol{x}$ and $\boldsymbol{y}$ two objects, whose attributes have different types. The aforementioned approach to computing the similarity between the two is summarized in the following algorithm (a similar approach could be taken for dissimilarities):

1. For the i-th attribute, compute the similarity $s_i(\boldsymbol{x}, \boldsymbol{y})$, normalized to $[0, 1]$;
2. Define an indicator variable δ_i for the i-th attribute, whose value is:
   - 0 if the i-th attribute is an asymmetric attribute and both objects have a value of 0, or if the value of the attribute of one of the object is missing;
   - 1 otherwise.
3. Compute the overall similarity as:

$$s(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_{i=1}^{n} \delta_i s_i(\boldsymbol{x}, \boldsymbol{y})}{\sum_{i=1}^{n} \delta_i}$$

Of course, if there's interest in giving different influence to the attributes on the final result, it is possible to introduce weights:

$$s(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_{i=1}^{n} w_i \delta_i s_i(\boldsymbol{x}, \boldsymbol{y})}{\sum_{i=1}^{n} w_i \delta_i}$$