# An overview of data mining

# 1

## 1.1 What's data mining?

Data mining lies at the intersection of computer science, optimization, and statistics, and often appears in other disciplines. Generally, data mining is the process of searching for knowledge in data from different perspectives. Here knowledge can refer to any kinds of summarized or unknown information that are hidden underlying the raw data. For instance, it can be a set of discriminative rules generated from the data collected on some patients of a certain disease and healthy people. These rules can be used for predicting the disease status of new patients.

In general, data mining tasks can be classified into two categories: *descriptive* and *predictive*. Descriptive mining tasks characterize a target data set in concise, informative, discriminative forms. Predictive mining tasks conduct the induction and inference on the current data to make future predictions.

## 1.2 Data mining process models

Data mining is an iterative process that consists of many steps. There are already some generic reference models on the data mining process, such as the Cross Industry Standard Process for Data Mining (CRISP-DM) process model. From a data-centric perspective, these models are structured as sequences of steps to transform the raw data into information or knowledge that is practically useful. As shown in Figure 1.1, a data mining process model typically involves the following phases: data collection, data preprocessing, data modeling, model assessment, and model deployment.

## 1.3 Data collection

The first step in the data mining process is to collect the relevant data according to the analysis goal in the applications. Generally, all the data that are helpful to achieve the objective in the analysis should be included. The key point here is how to define and understand the rather subjective term of "relevant data." Its correct interpretation highly depends on our understanding of the target problem and application background. Although this point will be further illustrated in subsequent chapters, we offer some general remarks here:

- In some cases, people definitely know that some kinds of data are highly relevant to the data mining task at hand. However, the acquisition of such data is very difficult or even impossible due to the device deficiency or cost. For instance, to accurately identify peptides in mass-spectrometry-based shotgun proteomics, it is necessary to generate at least one mass

**Data Mining for Bioinformatics Applications.** http://dx.doi.org/10.1016/B978-0-08-100100-4.00001-6

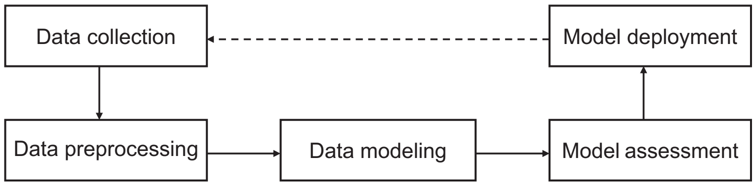

**Figure 1.1** Typical phases involved in a data mining process model.

spectrum for each peptide in the sample. However, due to the limitation of current mass spectrometers, it is not always possible to obtain mass spectra data that can cover all peptides present in the sample.

- On the other hand, the inclusion of new relevant data may significantly change the models and methods in the consequent steps of the data mining process. Furthermore, it is necessary to check thoroughly if the use of more relevant data will boost the performance of data mining procedures.

## 1.4 Data preprocessing

The objective of data preprocessing is twofold: (1) The real-world data are usually low quality; hence preprocessing is used to improve the quality of data, and consequently, the quality of data mining results. (2) In the data modeling step, some specific modeling algorithms cannot operate on the raw data, which should be transformed into some predefined data formats.

There are several general-purpose data preprocessing methods: *data cleaning*, *data integration*, *data reduction,* and *data transformation.*

*Data cleaning*: Real-world data are usually noisy, inconsistent, and incomplete. Data cleaning procedures aim at removing the noise, correcting inconsistencies, and filling in missing values in the data.

*Data integration*: In the data collection phase, data sets from different sources are relevant to the analysis problem. Data integration merges data from different sources into an integrated data set for subsequent data mining analysis. The main objective of data integration is to reduce and avoid redundancies and inconsistencies in the resulting data set.

*Data reduction*: The purpose of data reduction is to generate a new yet smaller representation of the original data set. Generally, the reduced data should contain approximately the same information of the original data that is of primary importance to the analysis target. The most commonly used data reduction technique includes dimension reduction (vertically, reduce the number of features) and sampling (horizontally, reduce the number of samples).

*Data transformation*: Different data mining algorithms may require different forms of data. Data transformation techniques consolidate the original data into forms appropriate for subsequent mining tasks. For instance, data normalization will transform the feature values into a predefined range such as [0.0, 1.0]. Data discretization will replace a continuous feature with a discrete one by dividing numeric values into intervals.

## 1.5 Data modeling

Before discussing the data modeling algorithms, it would be best to explain some terminology. Typically, the data preprocessing step would transform the raw data into a tabular form, in which the columns represent features/variables and rows correspond to samples/instances. For instance, Table 1.1 is a sample data set that has eight samples and five features (class is a special feature for which we are aiming to predict its feature value for a new given sample). The first four features are called *predictive features* and the class feature is the *target feature*. Here the predictive features can be symptoms of some disease, where the value of 1 indicates the existence of a symptom and 0 indicates otherwise. Similarly, the class feature value is 1 if the corresponding person (sample) has the disease.

### *1.5.1 Pattern mining*

Pattern discovery is a core data mining problem, which generates a set of interesting patterns that characterize the data sets in concise and informative forms. Initially, the studies on pattern discovery were dominated by the frequent pattern discovery paradigm, where only frequent patterns were explored. Currently, the issue of frequent pattern discovery has been thoroughly investigated, rendering its limitations well understood. Many alternative pattern discovery formulations are emerging and investigated in the literature. For example, many research efforts impose statistical significance tests over candidate patterns to control the risk of false discoveries.

**Table 1.1 An example data set with eight samples and five features**

| | Feature 1 | Feature 2 | Feature 3 | Feature 4 | Class |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 1 | 0 |
| 3 | 1 | 0 | 0 | 1 | 0 |
| 4 | 1 | 0 | 0 | 0 | 0 |
| 5 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 1 | 1 |
| 7 | 0 | 0 | 1 | 1 | 1 |
| 8 | 1 | 1 | 0 | 0 | 1 |

Here *class* is a special feature representing the category to which each sample belongs.

Let $D = \{t_1, t_2, \ldots, t_n\}$ be a data set, and each $t_i$ is an instance or sample and each sample has $g$ features $f_1, f_2, \ldots, f_g$. For ease of illustration, all features are assumed to be categorical. An item is defined as a feature-value pair $f_i = v$, where $v$ is one of feature values taken from $f_i$. One sample contains an item $f_i = v$ if it takes $v$ as its feature value for the feature $f_i$. The universe of all possible items that occur in the data set is denoted as $I = \{i_1, i_2, \ldots, i_m\}$.

In the example of Table 1.1, "Feature 1 = 0" and "Feature 1 = 1" are two items that are derived from the first feature.

A pattern $p = \{i_1, i_2, \ldots, i_k\}$ is a subset of all items, that is, $p \subseteq I$. $p$ is defined as a $k$-pattern if it is composed of $k$ items. For example, {"Feature 3 = 1," "Feature 4 = 1"} is a 2-pattern since it has two items.

A sample contains a pattern $p$ if all items in $p$ appear in that sample. For instance, the pattern {"Feature 3 = 1," "Feature 4 = 1"} is contained in the first, second, fifth, sixth, and seventh sample, respectively.

To date, there are many different methods for evaluating the interestingness of each pattern. Different interestingness evaluation methods lead to different pattern discovery problems. Generally, it is reasonable to assume that there is a generic interestingness calculation function $E(p)$, which can take different forms. For example, the popular frequent pattern discovery problem is based on the following pattern evaluation approach:

$$E(p) = \frac{|\{t_i | p \subseteq t_i\}|}{n}, \tag{1.1}$$

where $n$ is the number of samples in the data set and $|\cdot|$ represents the size of a set.

In the context of frequent pattern discovery, the value $E(p)$ is defined as the "support" of pattern $p$. Clearly, the support of a pattern is the percentage of samples that contain this pattern in a data set. For instance, the support for pattern {"Feature 3 = 1," "Feature 4 = 1"} in Table 1.1 is 5/8.

A pattern is frequent if its support value is no less than a user-defined threshold. The problem of frequent pattern mining is to find all frequent patterns. Suppose the minimum support threshold is specified to be 0.5, then {"Feature 3 = 1," "Feature 4 = 1"} in Table 1.1 is a frequent pattern since $5/8 > 0.5$.

To effectively identify frequent patterns, a large number of algorithms have been proposed over the last 20 years. Among these methods, the most well-known algorithms are Apriori and FP-growth, which traverse the pattern space in a breadth-first manner and a depth-first manner, respectively.

Apriori iteratively enumerates and checks all frequent patterns based on the antimonotonic property, that is, if a pattern is frequent, then all of its subpatterns must also be frequent. More precisely, the set of frequent patterns of size one is first generated by scanning the data set. When generating frequent patterns of size $k$, the following steps are performed:

**1.** Generate the set of potential frequent patterns of size $k$ by combining two patterns from the set of frequent patterns of size $k-1$. Two patterns are said to be joinable if their first $k-2$ items are the same.

2. Calculate the support of each candidate pattern of size $k$. Prune all the infrequent ones and add all the left to the set of frequent $k$-patterns.
3. Repeat the above steps until no more frequent candidates can be generated.

FP-growth adopts a divide-and-conquer strategy to discover all frequent patterns without candidate generation by constructing a frequent pattern tree (FP-tree). An FP-tree (frequent pattern tree) is a variation of the tree data structure, which is a prefix-tree structure for storing the crucial and compressed information about the support. It is composed of one root labeled as "NULL," a set of item prefix subtrees as the children of the root, and a frequent item header table. Each node in the item prefix subtree has three fields: *item-name*, *count*, and *node-link*, where *item-name* is the item name that this node represents, *count* is the number of samples that contains items in the portion of the path reaching this node, and *node-link* links to the next node in the FP-tree carrying the same item-name, or null if there is none. Each entry in the frequent item header table has two fields: *item name* and *head of node-link*. The *head of node-link* points to the first node in the FP-tree carrying the item name.

After the FP-tree is constructed, FP-growth identifies frequent patterns directly from the FP-tree as follows. Initially, each frequent pattern of size one is used as the suffix pattern. Then, the conditional pattern base of each suffix pattern is constructed as the set of prefix paths in the FP-tree cooccurring with this suffix pattern. From the conditional pattern base, a conditional FP-tree is generated for the suffix pattern. The above procedure is performed recursively on the conditional FP-tree. The so-called pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from the conditional FP-tree.

It is well recognized that the true correlation relationship among items or features may be missed in the support-based pattern discovery framework. Therefore, people have begun to study alternative pattern discovery formulations that use the statistical correlation measures to evaluate the interestingness of patterns. For example, the $\phi$ correlation coefficient (the computation form of the Pearson's correlation coefficient for binary variables) can be adopted for measuring the correlation between two items of a pattern $p=\{i_1, i_2\}$, that is,

$$E(p)=\phi(i_1,i_2)=\frac{\sup(i_1,i_2)-\sup(i_1)\sup(i_2)}{\sqrt{\sup(i_1)\sup(i_2)(1-\sup(i_1))(1-\sup(i_1))}}, \tag{1.2}$$

where $\sup(i_1)$, $\sup(i_2)$, and $\sup(i_1, i_2)$ denote the support values of item(s) $i_1, i_2$, and $\{i_1, i_2\}$, respectively.

The correlation mining can be used for evaluating the association relationship of features rather than items as well. The basic idea is to adopt a correlation measure that is capable of measuring the association for nonbinary categorical features or numeric features.

The pattern discovery problems discussed above are unsupervised; that is, the special-class feature is not considered in the pattern evaluation. Indeed, the discovery of distinguishing characteristics between different classes is one of the most important objectives in data mining as well. The objective of discriminative pattern mining is to find a set of patterns that occur with disproportionate frequency in one class versus others.

To evaluate the discriminative power of one pattern, one common strategy is to evaluate the difference of its support values across different classes. A wide variety of evaluation measures are available for this purpose. For instance, the absolute difference of supports is one of the simplest measures for evaluating the discriminative power in a two-class context:

$$E(p) = |\sup^{(1)}(p) - \sup^{(2)}(p)|, \tag{1.3}$$

where $\sup^{(i)}(p)$ represents the support of pattern $p$ in the set of samples that belong to the $i$th class ($i = 1, 2$). Given a user-specified threshold, if the value of discriminative measure can pass the threshold, then the pattern is claimed to be a discriminative pattern.

In the example data set of Table 1.1, the supports of {"Feature 1 = 1," "Feature 2 = 1"} are 0 and 3/4 in the class labeled as "0" and "1," respectively. According to the formula (1.3), their absolute support difference is $|0 - 0.75| = 0.75$. Clearly, this pattern has an apparent frequency difference between two classes.

Discriminative pattern mining is more computationally challenging than frequent pattern mining. This is because most discriminative measures do not have the antimonotonic property. Therefore, discriminative pattern-mining algorithms usually adopt a two-stage procedure: first generate a set of frequent patterns, and then evaluate the discriminative power of each frequent pattern. Alternatively, one can also design approaches for mining discriminative patterns by discovering significant discriminative patterns in a single step.

### 1.5.2 Supervised predictive modeling: Classification and regression

Supervised learning aims at building a predictive model from the data when a class/target feature is available. There are two types of class features: categorical features and numeric features. The categorical feature can take only nominal values, whereas the numeric feature can take an infinite number of numeric values. Generally, the predictive model is called *classification model* and *regression model* when the class feature is categorical and numeric, respectively. In other words, classification is to predict what category a sample should fall into while regression is the prediction of a numeric value. Here "supervised" means that the algorithm is aware of what to predict because class feature is specified.

In both classification and regression, the training data are used to build the predictive model and the testing data are used to evaluate the performance.

There are numerous classification algorithms in the literature, which belong to different "classifier families." That is, these classification methods arise from different fields within computer science and mathematics. For example, the linear discriminant analysis comes from statistics, rule-based classifiers or decision trees come from artificial intelligence and data mining, and so forth. In a recent study [1], 179 classifiers arising from 17 families were empirically compared over 121 data sets. It shows that the classifiers most likely to be the best are random forest (RF) and support vector machine (SVM). These two methods are briefly introduced in the following.

*RF* is widely used in bioinformatics applications because RF classification models have high prediction accuracy and can provide additional information such as the feature importance. RF is an ensemble of individual decision trees, where each tree in the forest is constructed with a random subset of samples and features. To predict the class of a new sample, every decision tree in the forest casts an unweighted vote for the sample after which the majority vote determines the class of the sample.

*SVM* is one of most well-known classifiers, which has a sound theoretical foundation. In a two-class classification task (one class is defined as the *positive class* and another class is referred to as the *negative class*), the aim of SVM is to find a classification function that is able to distinguish between samples of the two classes in the training data. Geometrically, if the samples from two classes can be linearly separated, the classification function $f(x)$ is linear that corresponds to a separating hyperplane that passes through the middle of the two classes. Once this function is generated, a new data sample $x_n$ can be classified by simply testing the sign of the function. That is, the new data sample will be assigned to the positive class if $f(x_n) > 0$. Otherwise, this sample will be assigned to the negative class.

Under the assumption that samples from two classes are linearly separable, there will be many linear functions that can achieve the above objective. Therefore, SVM requires that this function should be able to maximize the margin between the two classes. Geometrically, the margin corresponds to the shortest distance between the closest data samples to a sample on the hyperplane. This concept of margin enables us to formulate the classifier construction problem as an optimization problem, where the objective is to maximize the margin under the constraints that samples from two classes are separable. It has been argued that the generation of maximum margin hyperplanes can offer good generalization ability (the ability of correct classification of the future data).

The above fundamental formulation of SVM is based on a few unrealistic assumptions such as the data samples from different classes are linearly separable. To make it feasible for real-world classification problems, some extensions have been made. First, the "soft margin" idea is proposed to extend the SVM optimization model so that the classification function allows the existence of a few data samples of the opposite classes that cross the function. In addition, to handle the situations where the training data are not linearly separable, the kernel trick is used to transform the data from the original feature space into another high-dimensional feature space.

For the task of regression, the most popular methods are linear regression and its variants. In linear regression, a linear equation is constructed from the training data with the regression weights as the unknown parameters. Once the regression weights are obtained, the class feature value for a new sample can be forecasted.

In many bioinformatics applications, there are always data sets that have more features than data samples. In this case, it is not feasible to make a prediction using the standard linear regression method. To solve this problem, statisticians introduced so-called shrinkage methods such as *Lasso*, which are variants of linear regression by imposing some additional constraints.

In Lasso regression, the sum of the absolute values of all weights has to be less than or equal to a user-specified parameter. Subject to this constraint, some weight coefficients are forced to be exactly zeros if that parameter is small enough. This nice property makes it possible to eliminate irrelevant features in high-dimensional data sets.

### 1.5.3 Unsupervised descriptive modeling: Cluster analysis

The opposite of supervised learning is the problem of unsupervised learning, where the class feature is unavailable for the data. A fundamental unsupervised learning task is clustering or cluster analysis, which partitions data samples into different groups. The generic clustering criteria are twofold: (1) data samples in the same group are similar and (2) data samples from different groups are dissimilar.

There are many clustering algorithms in the literature. In general, these clustering methods can be classified into different categories. Among existing clustering algorithms, the most widely used methods are partitioning methods and hierarchical methods. The partitioning methods generate a one-level partitioning on data sets whereas the hierarchical methods create a hierarchical decomposition of the data samples.

The $k$-means algorithm is the most popular clustering method, which is a partitioning algorithm that will generate $k$ clusters for a given data set. The number of clusters $k$ is a user-specified parameter. Each cluster is represented by a single data point known as the *centroid*. A centroid is the center of all the data samples in the cluster.

In the $k$-means algorithm, $k$ centroids are randomly generated in the first step. Then, each data sample in the data set is assigned to the closest cluster according the distance between the data sample and the corresponding centroid. After this step, each centroid is updated as the new mean value of all the samples in that cluster.

Since the $k$-means algorithm can handle only numeric data sets, the $k$-modes algorithm extends the $k$-means paradigm to cluster categorical data by using (1) a simple matching distance measure for categorical data, (2) modes instead of means as the centroids for clusters, and (3) a frequency-based method to update modes in the $k$-means fashion to minimize the cost function of clustering.

## 1.6 Model assessment

The assessment of data mining results is extremely important in practice, because it guides the choice of models and measures the quality of the ultimately chosen model. It is important to note that models and patterns learned from the training data by the data mining algorithms are not necessarily valid, which may not be present in the future data set.

To validate the data mining results, there are different strategies for different data mining tasks. For the task of supervised predictive modeling, the evaluation generally uses a testing data set on which the learning algorithm was not trained. For the task of unsupervised descriptive modeling, statistical testing methods are usually adopted to evaluate the significance of discovered patterns or knowledge. The details of different validation methods will be presented in the corresponding chapters.

If the learned knowledge or models do not meet the desired standards, subsequently it is necessary to re-evaluate and change the preprocessing and data mining steps.

## 1.7 Model deployment

Extensive research efforts in data mining have been done on discovering knowledge from the underlying data. Despite such phenomenal success, most of these techniques stop short of the final objective of data mining—providing possible actions to maximize the profit while reducing costs for the end users. Although these techniques are essential to moving the data mining results to the eventual application, they nevertheless require a great deal of expert manuals to postprocess mined knowledge. Overall, the model deployment by taking actions to make a profit for individuals or organizations is the ultimate goal of data mining. For instance, discriminative patterns discovered from the case–control studies must be linked to their biological interpretations to facilitate the clinical validation.

From a computational perspective, making the mined patterns or knowledge *actionable* is critical to facilitate the model deployment. Here, the term *actionable* means that the learned model or patterns should suggest concrete and profitable actions to the decision maker. That is, the user can *do* something to bring direct benefits (increase in profits, reduction in cost, improvement in efficiency, etc.) to the organization’s advantage.

To make the mined model and patterns actionable, one typical strategy is to integrate the data mining process into the decision process. This requires a deep understanding of the application problem and related background knowledge. Therefore, the problem of model deployment is quite complicated and cooperation from multiple disciplines is highly demanded.

## 1.8 Summary

Data mining is already being successfully used in different real-life applications. In this chapter, key data mining concepts and tasks are introduced. For further reading, we suggest two textbooks. For nonprofessionals, Ref. [2] is a good choice as it introduces data mining and machine learning techniques in an easy-to-understand manner. For others who want more technical details, another popular data mining textbook [3] is recommended.

## References

[1] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. 15 (2014) 3133–3181.

[2] P. Harrington, Machine Learning in Action, Manning Publications Co., Greenwich, CT, 2012.

[3] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, third ed., Morgan Kaufmann, Waltham, MA, 2012.