1

Tasks

1.1 Introduction

This chapter discusses the assumptions and requirements of the three major data mining tasks this book focuses on: classification, regression, and clustering. It adopts a machine learning perspective, according to which they are all instantiations of *inductive learning*, which consists in generalizing patterns discovered in the data to create useful *knowledge*. This perfectly matches the predictive modeling view of data mining adopted by this book, according to which the ultimate goal of data mining is delivering models applicable to new data. While the book also discusses tasks that are not directly related to model creation, their only purpose is to make the latter easier, more reliable, and more effective. These auxiliary tasks – attribute transformation, discretization, and attribute selection – are not discussed here. Their definitions are presented in the corresponding chapters.

Inductive learning is definitely the most commonly studied learning scenario in the field of machine learning. It assumes that the learner is provided with *training information* (usually – but not necessarily – in the form of examples) from which it has to derive knowledge via inductive inference. The latter is based on discovering patterns in training information and generalizing them appropriately. The learner is not informed and has no possibility to verify with certainty which of the possible generalizations are correct and can only be hoped, but never guaranteed to succeed.

Inductive learning is the source of many data mining algorithms as well as of their theoretical justifications. This is the area where the domains of machine learning and data mining intersect. But even data mining algorithms that do not originate from machine learning can be usually seen as some explicit or implicit forms of inductive learning. The analyzed data plays the role of training information, and the models derived therefrom represent the induced knowledge. In particular, the three most widely studied and practically exercised data mining tasks, classification, regression, and clustering, can be considered inductive learning tasks. This chapter provides some basic background, terminology, and notation that is common for all of them.

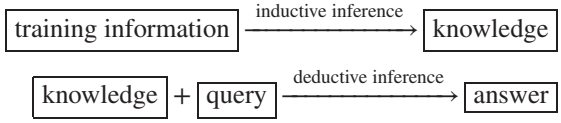
Data Mining Algorithms: Explained Using R, First Edition. Paweł Cichosz.

1.1.1 Knowledge

Whenever discussing any form of learning, including inductive learning, the term "knowledge" is frequently used to refer to the expected result of the learning process. It is unfortunately difficult to provide a satisfactory definition of knowledge, consistent with the common understanding as well as technically useful, without an extensive discussion of psychological and philosophical theories of mind and reasoning, which are undoubtedly beyond the scope of our interest here. It makes sense therefore to adopt a simple indirect surrogate definition which does not try to explain what knowledge is, but explains what purpose it can serve. This purpose of knowledge is *inference*.

1.1.2 Inference

Inference can be considered the process of using some available knowledge to derive some new knowledge. Given the fact that knowledge is both the input and output of inference, the above idea of defining knowledge as something used for inference may appear pretty useless and creating an infinite definition loop. It is not necessarily quite that bad since, in the context of inductive learning, different types of inference are employed when using training information to derive knowledge and when using this derived knowledge. These are *inductive inference* and *deductive inference*, and their role in inductive learning is schematically illustrated below.



1.1.2.1 Inductive inference

It is common to describe inductive inference as a "specific-to-general" reasoning process that uses a number of individual observations to generate laws that match these known observations and can be used to predict currently unknown future observations. This simplified view does not encompass all possible variations of inductive inference, but is perfectly sufficient for our needs. In the diagram presented above, the role of inductive inference is to derive (general) knowledge from the available training information (which can be considered specific knowledge).

Clearly, inductive inference is fallible and there is no guarantee that it would yield true conclusions when applied to true premises. Appropriately designed inductive inference mechanisms can reduce but never eliminate the risk of arriving at wrong conclusions. Actually, in the case of inductive learning, it makes more sense to speak about different quality levels of the induced knowledge rather than of its true or false (correct or incorrect) status. The main effort in inductive learning research is devoted to maximizing the quality of knowledge derived from not necessarily reliable training information via inductive reasoning.

1.1.2.2 Deductive inference

In contrast, deductive inference is infallible and its conclusions are guaranteed to be satisfied whenever its premises are. It does not necessarily have to be a "general-to-specific" reasoning process, although tends to be presented as such when opposed to inductive inference. Actually, it can be used with both general and specific premises, and general and specific conclusions,

although performing deductive inference based on knowledge derived via inductive learning does indeed conform to the popular "general-to-specific" pattern. It is noteworthy, by the way, that the well-known and useful number-theoretical theorem proving scheme called mathematical induction, which does follow a hybrid "specific-and-general-to-general" reasoning path, is in fact a form of deductive inference.

As shown in the above diagram, the induced knowledge is used to deduce an answer to a specified query. If the training information contained a number of known "historical" cases or observations, then typically the query presents one or more new cases or observations some interesting aspects of which remain unknown. The deductive inference process is supposed to supply these missing interesting aspects as an answer.

The infallibility of deductive inference by no means guarantees receiving correct answers to all queries. This would be the case only if the knowledge on which the deduction is based were perfectly correct, which cannot be expected in practice.

1.2 Inductive learning tasks

The three key data mining tasks, classification, regression, and clustering, are all based on the inductive learning paradigm. The essence of each of them is to inductively derive from *data* (representing training information), a *model* (representing knowledge) that has predictive utility, i.e., can be deductively applied to new data. Whereas these tasks, also called *predictive modeling* tasks, by no means exhaust the scope of inductive learning tasks studied in the field of machine learning, they represent the most widely applicable and useful variations thereof from a data mining perspective. They make a number of common assumptions about their input and output and have some common issues that deserve particular attention.

1.2.1 Domain

The *domain*, designated by X, is the set of all entities that are considered in a given inductive learning task. These can be customers, transactions, devices, or whatever is the subject of our interest.

1.2.2 Instances

Any single element of the domain, $x \in X$, is an *instance*. Instances constitute both training information for model creation and queries for model application.

1.2.3 Attributes

Instances, which may be some entities of the real world, are not directly observable. Their observable representation is provided by *attributes*. An attribute is a function $a : X \to A$ that assigns an attribute value to each instance from the domain. Unless discussing a specific example domain, we will assume that there are n attributes defined on the domain X, $a_1 : X \to A_1, a_2 : X \to A_2, \ldots, a_n : X \to A_n$.

Depending on the codomain A, attributes can be divided into different types, which may be treated differently by data mining algorithms. For most algorithms, it is sufficient to distinguish the following three major attribute types:

Nominal. Having a finite number of discrete values with no total order relation.

Ordinal. Having a finite number of discrete values with a total order relation.

Continuous (aka numerical, linear). Having numerical values.

These attribute types are best characterized by the basic relational and arithmetic operations that can be reasonably performed on their values rather than the "physical" representation of their values. Nominal attribute values can only be tested for equality. Ordinal attributes can be tested for both equality and inequality. For continuous attributes, we can perform inequality tests and all arithmetic operations defined for real numbers. It is not untypical to find nominal and ordinal attribute values represented by integer numbers assigned according to some encoding, but this representation does not make them liable for any arithmetics. On the other hand, continuous attributes can actually take only a small number of discrete numerical values, but these values can be treated as real numbers and used for whatever calculation or transformation that can be applied to real numbers.

In many cases, the distinction between attribute types is not quite crisp and the same attribute could be reasonably considered both nominal and ordinal, or both ordinal and continuous. In such a situation, the data miner has to judge or experimentally verify whether adopting a meaningful order relation for a nominal attribute could be helpful or harmful, or whether permitting some arithmetics on numerically represented values of an ordinal attributes might lead to some improvement.

Since instances can only be observable via their attribute values, it is common to identify them with the corresponding attribute value vectors. When speaking of an instance x, we will usually mean the vector of values $a_1(x), a_2(x), \ldots, a_n(x)$.

1.2.4 Target attribute

For the classification and regression tasks, a single attribute is distinguished as the *target attribute*. It represents the property of instances that the created model should be able to predict, based on the other attributes. Inductive learning tasks with a designated target attribute are referred to as *supervised learning* tasks, whereas those with no target attribute are referred to as *unsupervised learning* tasks. The same terms are also used when referring to algorithms for these two types of tasks. The values of the target attribute are assumed to be generally unavailable except for a subset of the domain used for model creation and evaluation.

1.2.5 Input attributes

Some or all of nontarget attributes are considered *input attributes*, the values of which are assumed to be generally available for the whole domain, so that the model can use them for generating its predictions. This general availability does not exclude the possibility of missing values, which is one of the common practical data quality issues.

1.2.6 Training set

Training information is represented by a *training set*, which is a subset of the domain. For any inductive learning task, a set of instances from the domain has to be available. Then the training set $T \subseteq D$ is the set of instances actually used for model creation, where $D \subset X$ denotes the set of all available instances. If there is a distinguished target attribute for a given inductive

learning tasks, its values are assumed to be known on D, and the available instances are called labeled. Selecting a subset of the whole available set of instances D for model creation can be motivated by various reasons, such as computational savings and intention to leave out some data for other purposes, including model evaluation.

There is some ambiguity about the term "training set" that has to be clarified. It is used here in a broader sense, as the whole subset of the domain is used for inductive learning (model creation). When discussing particular data mining algorithms, we will also be using this term in a narrower sense, referring to the subset of instances used for a single algorithm run. Several runs may be required before the final model is obtained (e.g., for algorithm selection, parameter tuning, attribute selection). These runs cannot use the whole available set of instances, since some have to be held out for model evaluation. This is necessary to provide basis for making decisions about the utility of particular algorithms, parameter setups, or attribute subsets, for which these runs are performed.

Since instances are not observable directly, but through their attribute values, when solving inductive learning tasks, we deal actually not with sets of instances, but with *datasets* – which are sets of attribute value vectors describing particular instances. A dataset can be usually thought of as a table with rows corresponding to instances, and columns corresponding to attributes (similarly to a table in a relational database or a spreadsheet). In practice, it is common to simply identify sets of instances with the corresponding datasets.

1.2.7 Model

Inductive learning tasks consist in finding (or generating), based on the provided training set, a *model* h representing knowledge that can be applied to all instances $x \in X$ in an automated way. This is a function that takes an instance as its argument (query) and returns a *prediction* as its value (answer). It can be therefore considered a new attribute, inductively derived from the training data, but defined (i.e., computable) for the whole domain.

What actually has to be predicted depends on the task. It is, in particular, the target attribute for the classification and regression tasks (the class and the target function value, respectively), for which the new attribute represented by the model is an approximation of an existing attribute that is only provided for a limited dataset, but remains unknown for the rest of the domain. For the clustering task, on the other hand, it is an entirely new attribute that represents the similarity structure discovered from the data, i.e., cluster membership assigning arbitrary instances from the domain to one of the similarity-based clusters.

1.2.8 Performance

The quality of predictions provided by a model is called the model's *performance*. It is not a big challenge to achieve good *training performance*, i.e., the quality of predictions generated on the training set. Of much greater interest is the *true performance*, i.e., the expected quality of predictions on the whole domain, including (mostly or entirely) previously unseen instances. It can be estimated in the process of model evaluation, using appropriate performance measures (task-specific) and evaluation procedures (mostly task-independent). Performance measures and evaluation procedures for inductive learning tasks are discussed in Chapters 7, 10, and 14.

1.2.9 Generalization

The true performance is also called the generalization performance, since to predict well on new, previously unseen data, the model has to encompass appropriate generalizations of patterns detected in the training set. Generalization is the essence of inductive learning. The exact generalization mechanisms are largely task- and algorithm-specific, but the common effect can be described in a simplistic way as making predictions for new instances based on their similarity to known training instances. For most algorithms, the simplicity is not determined based on any explicit similarity measure, though, but rather implied by their internal operating mechanisms and model representation.

1.2.10 Overfitting

Poor generalization leads to *overfitting*, which is a nightmare of inductive learning. A model h is considered overfitted to a training set T if there exists another model h' for the same task that performs worse than h on the training set, but performs better on the whole domain (including unseen data). The essence of overfitting is therefore a discrepancy between a model's good training performance (performance on the training set) and its poor true performance (expected performance on the whole domain).

1.2.11 Algorithms

Algorithms that solve an inductive learning task, i.e., generate models based on a given training set, are called *inductive learning algorithms* or *modeling algorithms*. Although an algorithm producing an arbitrarily poor model is formally a learning algorithm, it is natural to restrict one's interest to algorithms that attempt to optimize some explicitly specified or (more typically) implicitly assumed performance measure.

1.2.11.1 Weight-sensitive algorithms

Weight-sensitive modeling algorithms accept a vector of weights w containing a numerical weight $w_x \geq 0$ for each training instance $x \in T$. When a weight vector is specified for a weight-sensitive algorithm, it attempts to optimize the correspondingly weighted version of the performance measure normally assumed. For integer weights, this is roughly equivalent to using a modified training set T_w in which each instance $x \in T$ is replicated w_x times.

1.2.11.2 Inductive bias

Unlike in deductive inference, where all possible conclusions that can be derived are strictly determined by the premises, the training information used for inductive learning only narrows down the space of possible models, but does not strictly determine the model that will be obtained. There may be many models fitting the same set of training instances, and different possible generalizations of the patterns discovered therein. The criteria used by an inductive learning algorithm to select one of them for a given training set, which may be stated explicitly or implied by its operating mechanisms and model representation, are called the *inductive bias*. The inductive bias is not a deficiency of inductive learning algorithms, it is in fact a necessity: it guides the inductive inference process toward (hopefully) the most promising generalizations.

The inductive bias usually takes one or both of the following two forms:

Representation bias. A model representation method is adopted that makes it possible to represent only a small subset of possible models fitting a given dataset.

Preference bias. A preference measure is adopted that favors some models against the others based on their properties.

It is particularly common to see the preference for model simplicity as the inductive bias, which is believed to reduce the risk of overfitting according to Ockham's razor principle.

1.2.12 Inductive learning as search

It is a very useful and insightful perspective to view the inductive learning process as search through the space of possible models, directed by the training information and the inductive bias. The goal of this search is to find a model that fits the training set and can be expected to generalize well. It might appear that the model space should be as rich as possible, as it increases the chance that it actually contains a sufficiently good model. Unfortunately, a rich model space is likely to contain a number of poor models that fit the training set by a mere chance and would not perform well on the whole domain, and the inductive bias used might not be able to avoid choosing one of them. Such situation is referred to as *oversearching* and is one of the most important factors that increases the risk of overfitting.

The search perspective is by most means a conceptual framework that provides useful insights making it easier to understand the inductive learning process and some possible related caveats, but some learning algorithms actually do perform search not only from a conceptual, but also from a technical viewpoint, by applying some general purpose or tailored search techniques to identify the best model.

1.3 Classification

Classification is one of the fundamental cognitive processes used to organize and apply our knowledge about the world. It is common both in everyday life and in business, where we might want to classify customers, employees, transactions, stores, factories, devices, documents, or any other types of instances into a set of predefined meaningful classes or categories. It is therefore not surprising that building classification models by analyzing available data is one of the central data mining tasks that attracted more research interest and found more applications than any other task studied in the field.

The classification task consists in assigning instances from a given domain, described by a set of discrete- or continuous-valued attributes, into a set of classes, which can be considered values of a selected discrete target attribute, also called the target *concept*. Correct class labels are generally unknown, but are provided for a subset of the domain. It can be used to create the classification model, which is a machine-friendly representation of the knowledge needed to classify any possible instance from the same domain, described by the same set of attributes. This follows the general assumptions of inductive learning, of which the classification task is the most common instantiation.

The assumed general unavailability of class labels, but their availability for a given subset of the domain, may seem at first inconsistent, but it is essential for the idea of inductive inference on which all data mining methods are based. It also perfectly corresponds to the

requirements of most practical applications of classification, where the class represents some property of classified instances that is either hard and costly to determine, or (more typically) that becomes known later than is needed. This is why applying a classification model to assign class labels to instances is commonly referred to as *prediction*.

To see more precisely how the classification task instantiates the general inductive learning task, we only need to discuss those aspects of the latter where some classification-specific complements or comments can be added.

1.3.1 Concept

The term "concept" comes from the traditional machine learning terminology and is used to refer to a classification function $c : X \to C$, representing the true assignment of all instances from the domain to a finite set of classes (or categories) C. It can be considered simply a selected target nominal attribute. Concept values will be referred to as *class labels* or *classes*.

A particularly simple, but interesting kind of concepts is that with just a two-element set of classes, which can be assumed to be $C = \{0, 1\}$ for convenience. Such concepts are sometimes called *single concepts*, opposed to *multiconcepts* with $|C| > 2$. Single concepts best correspond to the original notion of concepts, borrowed by machine learning from cognitive psychology. An instance x is said to "belong to" or "be an example of" concept c when $c(x) = 1$. When $c(x) = 0$, the instance is said "not to belong to" or "to be a negative example of" concept c. Classification tasks with single concepts will be referred to as two-class classification tasks.

1.3.2 Training set

The target concept is assumed to be unknown in general, except for some set of instances $D \subset X$ (otherwise no data mining would be possible). Some or all of these available *labeled instances* constitute the *training set* $T \subseteq D$.

Example 1.3.1 As a simple example of a training set for the classification task, consider the classic *weather* data with 14 instances, four input attributes, and one target attribute. The following R code reads this dataset to an R dataframe and summarizes the distribution of attributes (`outlook`, `temperature`, `humidity`, `windy`) and the target concept (`play`).

`dmr.data`

```
weather <- read.table(text="
      outlook temperature humidity   wind play
   1    sunny         hot     high normal   no
   2    sunny         hot     high   high   no
   3 overcast         hot     high normal  yes
   4    rainy        mild     high normal  yes
   5    rainy        cold   normal normal  yes
   6    rainy        cold   normal   high   no
   7 overcast        cold   normal   high  yes
   8    sunny        mild     high normal   no
   9    sunny        cold   normal normal  yes
  10   rainy        mild   normal normal  yes
  11    sunny        mild   normal   high  yes
  12 overcast        mild     high   high  yes
```

```
  13 overcast         hot   normal normal  yes
  14    rainy        mild     high   high   no")

summary(weather)
```

The first four attributes describe weather conditions and the last attribute is the target concept that classifies them as appropriate or inappropriate for playing sports. The extremely small size of this dataset makes it totally unrealistic and unsuitable for any experiments evaluating the performance of classification algorithms, but absolutely perfect for illustrating the calculations needed for their operation. All such calculations, specified by mathematical equations and implemented by illustrative R code, can be easily verified "manually." This is why the *weather* dataset is frequently used in examples presented in chapters on classification in this book.

Example 1.3.2 A modified version of the dataset from the previous example, in which the `temperature` and `humidity` attributes are continuous, will also be used occasionally. This will be referred to as the *weatherc* data. The following R code reads this dataset to an R dataframe and summarizes the attribute distributions.

`dmr.data`

```
weatherc <- read.table(text="
      outlook temperature humidity   wind play
   1    sunny          27       80 normal   no
   2    sunny          28       65   high   no
   3 overcast          29       90 normal  yes
   4    rainy          21       75 normal  yes
   5    rainy          17       40 normal  yes
   6    rainy          15       25   high   no
   7 overcast          19       50   high  yes
   8    sunny          22       95 normal   no
   9    sunny          18       45 normal  yes
  10    rainy          23       30 normal  yes
  11    sunny          24       55   high  yes
  12 overcast          25       70   high  yes
  13 overcast          30       35 normal  yes
  14    rainy          26       85   high   no")

summary(weatherc)
```

1.3.3 Model

A *classification model* $h : X \to C$ produces class predictions for all instances $x \in X$ and is supposed to be a good approximation of the target concept c on the whole domain. Classification models are briefly called *classifiers*, although the latter term sometimes also refers to classification algorithms, used to create classification models.

1.3.3.1 Scoring classifiers

For two-class classification tasks (single concepts), a particular kind of *scoring* classification models deserves special interest. These are the classification models that predict class labels

in a two-step process: they first map instances into real numbers called scores and then they assign one class label (1, by convention) to instances with sufficiently high scores and the other class label (0) to the remaining instances.

More precisely, a scoring model is represented by a scoring function $\pi : X \to \mathcal{R}$ and a labeling function $\lambda : \mathcal{R} \to \{0, 1\}$. The former assigns real-valued scores to all instances from the domain, and the latter converts these scores to class labels using a cutoff rule, such as

$$\lambda(r) = \begin{cases} 1 & \text{if } r \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

where θ is a cutoff value. The model is then the composition of its scoring and labeling functions, $h(x) = \lambda(\pi(x))$.

It is a common convention to consider scoring classification models sharing the same scoring function and differing only in the labeling function (i.e., using different cutoff values) as the same single model, working in different *operating points*. Classification algorithms capable of generating scoring classification models typically create a scoring function and a cutoff value for one *default* operating point, but a number of other operating points can be obtained by using different cutoff values.

Classification models that generate class labels directly, without scoring and labeling functions, are sometimes called *discrete* classifiers.

1.3.3.2 Probabilistic classifiers

A related interesting and useful special kind of classification models are *probabilistic* classifiers, which estimate class probabilities for instances being classified, and then make predictions based on these probabilities. A probabilistic classifier assigns to each instance $x \in X$ and class $d \in C$ a probability estimate $P(d|x)$ of instance x belonging to class d of the target concept c. The estimated class probabilities can be used to generate class labels using the obvious *maximum-probability* rule:

$$h(x) = \arg\max_{d \in C} P(d|x) \tag{1.2}$$

or – under nonuniform misclassification costs – the less obvious minimum-cost rule, as discussed in Section 6.3.3.

For two-class tasks, probabilistic classifiers constitute a particularly common subclass of scoring classifiers, with the estimated probabilities of class 1 for particular instances considered scores, i.e., $\pi(x) = P(1|x)$.

1.3.4 Performance

The exact meaning of "good approximation" of the target concept that is expected from a classification model is established by classification model performance measures, but – informally – we want the model to usually provide correct class labels, as far as possible. Even a model that is wrong in most cases remains a (poor) model, but –needless to say – poor models are not of particular interest in the classification task. The most commonly adopted performance measure is the *misclassification error* which is the fraction of instances from a

dataset or the whole domain misclassified by the model. This and other performance measures for classification models are discussed in Section 7.2.

1.3.5 Generalization

As any inductive model, a classification model should be judged "good" or "poor" not just based on its performance on the training set, but on its (expected) performance on the whole domain. In other words, we care not only and not mainly for the classification accuracy on the training set, but also on new previously unseen instances to which the model could be applied. This requires classification algorithms to not only *discover* relationships between class labels and attribute values in the training set, but also to *generalize* them so that they can be expected to hold on new data.

1.3.6 Overfitting

A classification model is overfitted if another model predicts class labels for the whole domain better despite yielding worse training set predictions. This is typically defined with respect to the misclassification error, but arbitrary performance measures can be used as well. Many classification algorithms include mechanisms supposed to reduce the risk of overfitting.

1.3.7 Algorithms

Algorithms that solve the classification task, i.e., generate classification models based on a supplied training set, are called *classification algorithms*. Two special types of classification algorithms deserve particular interest: weight-sensitive algorithms and cost-sensitive algorithms. They are capable of creating models with particular properties that are sometimes desirable.

1.3.7.1 Weight-sensitive algorithms

A weight-sensitive classification algorithm – like other weight-sensitive modeling algorithms – accepts a vector of weights w containing a numerical weight $w_x \geq 0$ for each training instance $x \in T$. It is not uncommon, though, to have weights assigned solely based on classes, with ω_d for each $d \in C$ being the weight of all training instances of class d, i.e., $w_x = \omega_{c(x)}$. When a weight vector is specified for a weight-sensitive algorithm, it attempts to optimize the correspondingly weighted version of the performance measure normally assumed, where each instance's weight is applied to its contribution to the performance measure. Typically, this is the weighted misclassification error instead of the usual one. For integer weights, this is roughly equivalent to using a modified training set T_w in which each instance $x \in T$ is replicated w_x times.

1.3.7.2 Cost-sensitive algorithms

Cost-sensitive classification algorithms take into account that the severity of misclassifying instances may vary across different true and predicted class combinations, with some being more acceptable than others. Such algorithms accept a misclassification cost specification on

input and adopt the mean misclassification cost as the performance measure to minimize during model creation. Several approaches to achieving cost sensitivity are discussed in Chapter 6.

1.4 Regression

Similar to classification, regression is an inductive learning task that has been extensively studied and can be widely encountered in practical applications. It can be informally characterized as "classification with continuous classes," which means that regression models predict numerical values rather than discrete class labels. This relationship to the classification task makes it possible to describe the regression task by referring to the latter where appropriate and highlighting the differences where necessary.

The term "regression" tends to be sometimes used in a narrow technical meaning referring to statistical algorithms for fitting parametric regression models. We adopt here a broader view in which regression is presented in a completely algorithm-independent way as one of the major data mining tasks, and any algorithm that solves this task will be considered a regression algorithm. This makes regression equivalent to *numerical prediction*, practical instantiations of which are nearly as common as those of classification. In particular, we might want to predict prices, demand, production or sales volumes, resource consumption, physical parameters, etc.

The regression task consists in assigning numerical values to instances from a given domain, described by a set of discrete or continuous-valued attributes. This assignment is supposed to approximate some target function, generally unknown, except for a subset of the domain. This subset can be used to create the regression model, which is a machine-friendly representation of the relationships between the target function and the attributes that makes it possible to predict unknown target function values for any possible instance from the same domain. The regression task adopts therefore the same general scenario of inductive learning that has been presented above for the classification task. In practical applications, the target function represents some interesting property of instances from the domain that is either difficult and costly to determine, or (more typically) that becomes known later than is needed. Subsections below add regression-specific highlights to what has been presented above for inductive learning in general.

1.4.1 Target function

The *target function* $f : X \to \mathcal{R}$ represents the true assignment of numerical values to all instances from the domain. Target function values will briefly be called *target values* or *target labels*.

1.4.2 Training set

The *training set* $T \subseteq D \subset X$ for regression consists of some or all labeled instances for which target function values are available, despite being unknown in general.

Example 1.4.1 As a simple example of a training set for the regression task, consider a modified version of the *weatherc* data, in which the `play` attribute originally representing the target concept is replaced by a new continuous `playability` attribute, which now

represents the target function. This will be referred to as the *weatherr* data. The following R code reads this dataset to an R dataframe and summarizes the attribute distributions.

dmr.data

```
weatherr <- read.table(text="
       outlook temperature humidity   wind playability
   1     sunny          27       80 normal        0.48
   2     sunny          28       65   high        0.46
   3  overcast          29       90 normal        0.68
   4     rainy          21       75 normal        0.52
   5     rainy          17       40 normal        0.54
   6     rainy          15       25   high        0.47
   7  overcast          19       50   high        0.74
   8     sunny          22       95 normal        0.49
   9     sunny          18       45 normal        0.64
  10     rainy          23       30 normal        0.55
  11     sunny          24       55   high        0.57
  12  overcast          25       70   high        0.68
  13  overcast          30       35 normal        0.79
  14     rainy          26       85   high        0.33")

summary(weatherr)
```

1.4.3 Model

A *regression model* $h : X \to \mathcal{R}$ can be used to generate numerical predictions for all instances $x \in X$ and supposed to provide a good approximation of the target function f on the whole domain.

1.4.4 Performance

The exact meaning of "good approximation" is established by regression model performance measures, but – informally – we want the model to usually provide predictions that are not far away from the true target values. One commonly adopted performance measure is the mean sum of squared differences between the true and predicted values, referred to as the *mean square error*. This and other regression performance measures are discussed in Chapter 10.

1.4.5 Generalization

Generalization is no less crucial for regression than for classification. Regression algorithms have to not only *discover* relationships between the target function and attribute values in the training set, but also to *generalize* them so that they can be expected to hold on new data.

1.4.6 Overfitting

Poor generalization leads to overfitting, which is the same serious problem for regression as for the classification and can be defined in the same way. Many regression algorithms include mechanisms supposed to reduce the risk of overfitting.

1.4.7 Algorithms

A *regression algorithm* generates a regression model based on a given training set.

1.5 Clustering

Clustering is an inductive learning task that differs from the classification and regression tasks from the same family by the lack of a predetermined target attribute to be predicted. It can be thought of as classification with autonomously discovered rather than predefined classes, which are based on similarity patterns identified in the data.

The clustering task consists in dividing a set of instances from a given domain, described by a number of discrete or continuous-valued attributes, into a set of clusters based on their similarity, and creating a model that can map arbitrary instances from the same domain to these clusters. This can be considered a superposition of two subtasks:

Cluster formation. The identification of similarity-based groups in the analyzed data.

Cluster modeling. Creating a model for cluster membership prediction.

The latter is clearly a classification task, with clusters identified within the first subtask used as classes. This could be performed, in principle, using any available classification algorithm. It is usually more convenient not to separate these two subtasks, though, and most clustering algorithms handle both cluster formation and cluster modeling. It makes it possible for the criteria used to identify clusters to be subsequently reused for cluster membership prediction.

1.5.1 Motivation

The utility of the clustering task may not be as self-evident as for the classification and regression tasks and deserves some more explanation. Some typical reasons to perform clustering are listed below, along with example applications where they are likely to appear.

- Clustering can provide useful insights about the similarity patterns present in the data, and a clustering model can be considered as knowledge *per se*. Some example applications where this is the case include
  - — customer segmentation,
  - — point of sale segmentation,
  - — document catalog creation.
- Clustering can be performed on a selected subset of *observable* attributes that are easily available for all instances, and used to predict *hidden* attributes that are impossible or difficult to determine for some instances based on cluster membership. This is similar to classification or regression with multiple target attributes with sparingly available values. Such situation occurs in the following example applications:
  - — customer clustering based on socio-demographic attributes used to predict attributes describing purchase behavior,
  - — point-of-sale segmentation based on location, building, and local population features, used to predict attributes describing selling performance.

- Clustering performed on a set of "normal" instances can be used for anomaly detection, by issuing alerts for new instances that do not fit any existing cluster. This is a possible approach to various diagnostics applications, such as
  - — network traffic clustering, used for intrusion detection,
  - — credit card transaction clustering, used for fraud detection,
  - — sensor signal clustering, used for device fault detection.
- Clustering can be used as a domain decomposition method for some further data mining tasks, which may be easier to perform within homogeneous clusters. Example applications include
  - — customer clustering based on socio-demographic and purchase history attributes, and classification with respect to loyalty within clusters,
  - — customer clustering based on socio-demographic and purchase history attributes, and predicting reaction to incentives within clusters,
  - — credit card account clustering based on cardholder socio-demographic attributes and transaction history attributes, and classification with respect to fraud likelihood within clusters,
  - — product clustering based on technical specification and usage attributes, and demand forecasting within clusters,
  - — used vehicle clustering and price prediction within clusters.

To more formally define the clustering task, we only need to slightly modify the classification task definition wherever these two differ.

1.5.2 Training set

The *training set* $T \subseteq D$ is a subset of the available dataset $D \subset X$ used to create a clustering model.

Unlike for the classification and regression tasks, training instances are not normally assumed to be labeled by any target attribute values, since no target attribute is considered for the clustering task. Particular instantiations of the clustering task may adopt some other assumptions, though. In particular, one of the typical clustering usage scenarios assumes that the set of attributes is divided into subsets of *observable* and *hidden* attributes. If this is the case, only the former are assumed to be available for the whole domain, but the dataset D consists of instances for which the latter are known as well.

Example 1.5.1 As a simple example of a training set for the clustering task, consider a modified version of the *weatherc* data, in which the `play` attribute originally representing the target function is dropped, as demonstrated by the following R code. This will be referred to as the *weathercl* data.

`dmr.data`

```
weathercl <- weatherc[,-5]
```

1.5.3 Model

The clustering task consists in creating, based on the provided training set, a model $h : X \to C_h$ that is computable for all $x \in X$ and maps them into a model-specific set of clusters C_h. This is formally nearly the same requirement as for classification models, except for an apparently small but substantial difference consisting in the set of "classes" C_h not being predetermined, but identified as part of model creation. It is therefore more instructive to think of the clustering model creation process as of cluster identification (i.e., determining C_h) rather than cluster modeling (i.e., determining h for given C_h), since once the set of clusters is determined, their representation usually makes the actual mapping function straightforward to obtain.

Without a predetermined target attribute, the clustering model is not required to approximate any kind of target concept or function. This deprives the model creation process from any explicit and objective guidance, which is so essential for the classification and regression tasks, making clustering an unsupervised inductive learning task. The only requirement adopted for the clustering task is to identify clusters based on similarity patterns observed in the set of training instances. This can only be stated informally when discussing the general task formulation and is explicitly or implicitly formalized by specific clustering algorithms.

1.5.4 Crisp vs. soft clustering

As presented above, a clustering model represents the so-called *crisp* clustering in which all clusters are disjoint, i.e., every instance is assigned to exactly one cluster. This is the same as with classes in the classification task. Departures from this view of clustering, however, are not quite uncommon. Unlike "objectively" existing, predefined classes, which serve the purpose of separating distinct types of entities, clusters formed by a clustering algorithm extract similarity patterns that may not be strong enough to justify definite distinctions. This is why sometimes *soft* or *fuzzy* clustering models are considered that may assign a single instance to multiple clusters at some membership level.

1.5.5 Hierarchical clustering

A variation of the clustering task receiving special attention requires that the clustering model be *hierarchical*. A hierarchical clustering model can be thought of as a set of ordinary (flat) clustering models organized in a tree structure. Each internal tree node represents both a flat clustering model and a cluster of the flat clustering model from its parent node. The root node represents a special level-0 cluster covering the whole domain. Leaves represent just clusters, with no further clustering models assigned to them. The model in the root node is applied to the whole domain and maps it to level-1 clusters. These clusters correspond to descendant nodes with subsequent models that partition them into subclusters, etc.

1.5.6 Performance

Given the unsupervised nature of the clustering task, clustering model performance can be hardly evaluated in a truly objective and application-independent way. Still there is a number of clustering model performance measures that may be helpful in judging the suitability of a given model for a given application. These are presented in Chapter 14.

1.5.7 Generalization

As for other inductive learning models, a clustering model is expected, in principle, to generalize relationships discovered in the training set and make them applicable to the whole domain. In other words, we care not only and not mainly for fitting all the similarity patterns in the training set, but also for capturing those that would hold for new previously unseen instances to which the model could be applied. On the other hand, the importance of this form of generalization required for the clustering task tends to be underestimated sometimes, with all or most of the attention paid to achieving the best possible match between the identified set of clusters and the training set. This is sufficient for applications where similarity patterns captured by the clustering model are not supposed to be applied for prediction.

1.5.8 Algorithms

Algorithms that generate clustering models based on a given training set are called *clustering algorithms*. Since the clustering task in its general formulation does not specify any strict requirements for the exact way of capturing the similarity patterns in the data by the clustering algorithm, different algorithms take substantially different approaches. They may be roughly categorized as follows:

(Dis)similarity-based clustering. Using a predefined or user-specified explicit measure of instance similarity to drive the cluster formation and modeling processes.

Probabilistic clustering. Using probability distributions and probabilistic inference to drive the cluster formation and modeling processes.

Conceptual clustering. Using a (usually symbolic) conceptual cluster representation to drive the cluster formation and modeling processes.

The scope of clustering algorithms presented in this book is limited to (dis)similarity-based clustering.

1.5.9 Descriptive vs. predictive clustering

The definition of clustering presented in this section and assumed later in this book's clustering chapters adopts a predictive modeling perspective. It is not uncommon to see practical applications of clustering that focus on descriptive modeling only, though. The capability of predicting cluster membership for new instances is neither needed nor used whenever the purpose of clustering is just to discover and present similarity patterns in the data. Several practical implementations of clustering algorithms do not provide the prediction functionality, making it possible to determine cluster membership for training instances only.

1.6 Practical issues

The definitions of inductive learning tasks presented above are somewhat idealized. For practical tasks some compromises are often necessary. They are mostly related to imperfect data, which may not provide full or reliable instance descriptions.

1.6.1 Incomplete data

The above descriptions of attributes as functions mapping instances to attribute values might suggest that the values of all attributes are available for all instances. This is usually not the case in practice, where for some instances some attribute values may be missing. Such an incomplete dataset can either be "repaired" in a preprocessing phase, or handled in some special way by modeling algorithms.

1.6.2 Noisy data

Similarly, having described attributes (including the target attribute) as functions mapping instances to attribute values or classes, we might expect the available attribute values to be always perfectly reliable. It is often not the case in practice, where attribute values (including instance target labels) can be corrupted by some noise. Sometimes incorrect attribute values can be corrected or unreliable instances filtered out during a preprocessing phase, but usually the presence of noise has to be accepted as unavoidable. Moreover, some noise not only cannot be usually eliminated, but also in many cases it cannot be even detected. To take the simplest example, do two instances with exactly the same attribute values but different class labels result from noise or rather from an insufficient set of attributes which cannot fully differentiate instances from different classes? Such questions can be often asked, but rarely answered, unless we accept a somewhat evasive answer that both hypotheses represent simply two different views on the same phenomenon. The fact is that all useful data mining algorithms have to assume the risk of data being affected by noise and not blindly trust any apparent patterns. This limited confidence in data is actual at the heart of good generalization.

1.7 Conclusion

Many data mining algorithms, both those originating from machine learning and those developed in the field of statistics, are based on the inductive learning paradigm. The three most common data mining tasks, classification, regression, and clustering, follow this paradigm particularly directly and can be therefore called inductive learning tasks. This chapter has provided some background information, assumptions, and terminology that apply to all of them. The entirety of this book is devoted to algorithms solving these tasks and to closely related techniques used to improve model quality.

The formulations of the inductive learning tasks presented in this chapter and subsequently adopted throughout the book are simple and generic. Their more specific or enhanced versions are sometimes studied in the literature and employed in applications where necessary. These may adopt special assumptions about the properties of the target concept or the target function for the classification and regression tasks (such as the number of classes, relationships among classes, or target function value distribution), fulfill special requirements for cluster formation and representation methods for clustering (such as soft clusters), or adjust to special domain and training set properties (such as the number and types of attributes, or the number and availability of training instances). Leaving such interesting and useful extensions beyond the scope of this book is a regrettable necessity, dictated by the adopted level of detail, precision, and R code illustration coverage. Extending the scope of the book would require either compromising on the former, or making its size unmanageable for a single author.

This is also one additional reason why other data mining tasks and the corresponding algorithms that definitely deserve attention are not included in the book. This applies, in particular, to association and temporal pattern discovery, geospatial data analysis, time series analysis, or survival analysis. While some of these tasks can be viewed as forms of classification or regression, they are much better handled by dedicated algorithms. In some cases, the latter tend to be more mathematically refined than those presented in the book and do not fit the "maximum usefulness at minimum complexity" principle adopted here. Some of these tasks also make substantially different assumptions about data representation, making the instance-attribute scheme introduced in this chapter inapplicable or awkward. This is why such tasks and algorithms would require a considerably different form of presentation and including them would make the book not only overly large or overly superficial, but also inconsistent.

1.8 Further readings

The three major data mining tasks introduced in this chapter are discussed in most data mining books which cover predictive modeling (e.g., Abu-Mostafa *et al*., 2012, Cios *et al*., 2007, Han *et al*., 2011, Hand *et al*., 2001, Tan *et al*., 2013, Witten *et al*., 2011). While they may use partially different terminology and notation, emphasize different aspects of these tasks, or present different motivation and application examples, they ultimately arrive at the same basic assumptions and requirements. Kohavi and Provost (1998) collect concise definitions for some of the most commonly used terms.

While all the three tasks can be presented as instantiations of inductive learning, it is classification learning that has received most attention in the machine learning literature (Cichosz, 2007; Mitchell, 1997). This learning task, also referred to as concept learning, is also one of the major topics of machine learning theoretical work. Most of this work has been done within the scope of the computational learning theory, which focuses on the learnability of concept classes and characterizing their hardness, deriving requirements for the number and quality of training instances, and establishing the properties and performance bounds of specific learning algorithms (e.g., Kearns and Vazirani, 1994, Valiant, 1984, Vapnik, 1998).

However, a complementary approach that highlights different types of inductive inference used to derive models from training information, possibly augmented with background knowledge, also received some attention and brought insightful results (Michalski, 1983). Viewing inductive learning as searching the model space for the most justified generalizations of training data, as proposed by Mitchell (1982) for classification learning, remains a valid view of other data mining tasks. This is also the case for the idea of bias as a necessary component of any inductive learning process (Haussler, 1988; Mitchell, 1980).

The classification and clustering tasks in text domains, where instances are text documents or messages of any kind, become text mining tasks. While some of general-purpose classification and clustering algorithms handle text data quite well, after transforming it to an appropriate representation, there are also several dedicated text mining algorithms as well as more specific text mining tasks that do not have their general data mining counterparts (Aggarwal and Zhai, 2012; Weiss, 2010).

The *weather* data first presented in this chapter and then used several times throughout the book comes from Quinlan (1986) and is quite popular in the machine learning and data mining literature (e.g., Witten *et al*. 2011).

References

Abu-Mostafa YS, Magdon-Ismail M and Lin HT 2012 *Learning from Data*. AMLBook.

Cichosz P 2007 *Learning Systems (in Polish)* 2nd edn. WNT.

Cios KJ, Pedrycz W, Swiniarski RW and Kurgan L 2007 *Data Mining: A Knowledge Discovery Approach*. Springer.

(eds. Aggarwal CC and Zhai CX) 2012 *Mining Text Data*. Springer.

Han J, Kamber M and Pei J 2011 *Data Mining: Concepts and Techniques* 3rd edn. Morgan Kaufmann.

Hand DJ, Mannila H and Smyth P 2001 *Principles of Data Mining*. MIT Press.

Haussler D 1988 Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence* **36**, 177–221.

Kearns M and Vazirani U 1994 *An Introduction to Computational Learning Theory*. MIT Press.

Kohavi R and Provost F 1998 Glossary of terms: Editorial for the special issue on applications of machine learning and the knowledge discovery process. *Machine Learning* **30**, 271–274.

Michalski RS 1983 A theory and methodology of inductive learning In *Machine Learning: An Artificial Intelligence Approach* (ed. Michalski RS, Carbonell JG and Mitchell TM) Tioga.

Mitchell T 1980 The need for biases in learning generalizations. Technical Report CBM-TR-117, Rutgers University, New Brunswick, NJ.

Mitchell T 1997 *Machine Learning*. McGraw-Hill.

Mitchell TM 1982 Generalization as search. *Artificial Intelligence* **18**, 203–226.

Quinlan JR 1986 Induction of decision trees. *Machine Learning* **1**, 81–106.

Tan PN, Steinbach M and Kumar V 2013 *Introduction to Data Mining* 2nd edn. Addison-Wesley.

Valiant L 1984 A theory of the learnable. *Communications of the ACM* **27**, pp. 1134–1142.

Vapnik VN 1998 *Statistical Learning Theory*. Wiley.

Weiss SM 2010 *Fundamentals of Predictive Text Mining*. Springer.

Witten IH, Frank E and Hall MA 2011 *Data Mining: Practical Machine Learning Tools and Techniques* 3rd edn. Morgan Kaufmann.