PART I

CHAPTER 1

WHAT IS KNOWLEDGE DISCOVERY?

Knowledge discovery is a semiautomated process of extracting useful information from collections of data that are too big to be investigated manually. By *semiautomated* we mean that we use computer-based tools for the discovery process but that guidance by an analyst is indispensable. The information retrieved by the discovery process usually takes on the form of actionable or explanatory patterns often referred to as *models*. There are many different types of models. For instance, we have models that are represented as *if–then–else* rules as well as models that implement artificial neural networks. All models have the desirable property that they tend to ignore unnecessary detail and summarize the major trends in data. A model can represent or summarize terabytes of data and therefore provides access to information or knowledge hidden in large amounts of data. In this book we deal with one particular type of model called a *support vector machine*. Support vector machines represent a powerful new class of models invented by Vladimir Vapnik in the early 1990s. They have been shown to be competitive with artificial neural networks and outperform them in many cases.

A term that is often associated with knowledge discovery is *data mining*. Data mining can be considered a specific kind of knowledge discovery process that aims at extracting information from databases. Data mining is often referred to as *knowledge discovery in databases* (KDD).

Knowledge discovery is a highly interdisciplinary undertaking ranging from domain analysis, data cleansing, and visualization to model evaluation and deployment (see Figure 1.1). However, at the core of the knowledge discovery process is a discovery algorithm that performs some kind of pattern recognition and constructs models of the data encountered. The discovery algorithms we are concerned with in

Knowledge Discovery with Support Vector Machines, by Lutz Hamel

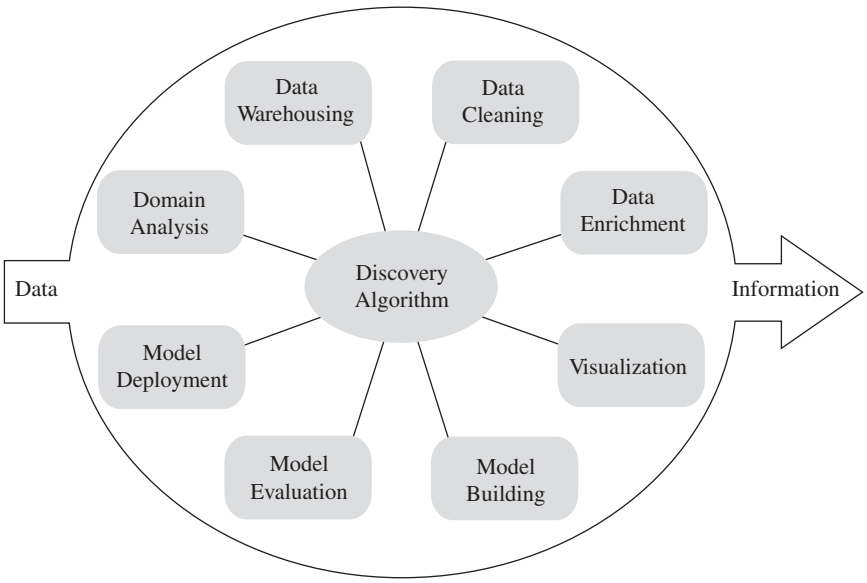

FIGURE 1.1 The knowledge discovery process: from data to information.

this book are based on *machine learning*. Let us start by defining what we mean by machine learning.

1.1 MACHINE LEARNING

Phenomena whose behavior we can observe exist all around us. Consider, for example, the orbits of the planets around the sun or the timing of the tides. The central question in machine learning is: *Can we use computers to discover and describe patterns based on these behaviors?* The answer to this question is a resounding "yes" and it is the topic of the remainder of the book.

Perhaps the easiest way to describe phenomena is through classification. Here, a particular object either belongs to a class of objects or it does not. When we see a cat, we easily recognize that it belongs to the class of mammals, and when we see a crow, we recognize that it belongs to the class of birds. Abstractly speaking, we can imagine that there exists some process in connection with some phenomenon that labels objects as *true* if they belong to the class in question or *false* if they do not belong to the class. In our case, we have $mammal(\text{cat}) = true$ and $mammal(\text{crow}) = false$, as well as $bird(\text{cat}) = false$ and $bird(\text{crow}) = true$. Here, $mammal$ and $bird$ are processes that provide the labels for any object according to the class of mammals and the class of birds, respectively. Typically, classifications are not as easy as mammals and birds, and in general we do not have access to the processes that label the objects. We can only observe the consequences of these processes: the observable labels for each object. The goal of machine learning then is to compute a suitable model for

a labeling process that approximates the original process as closely as possible. The following definition states this more formally.

Definition 1.1 (Machine Learning)
Given:

- *A data universe* X
- *A sample set* S*, where* $S \subset X$
- *Some target function (labeling process)* $f : X \to \{true, false\}$
- *A labeled training set* D*, where* $D = \{(x, y) \mid x \in S \text{ and } y = f(x)\}$

Compute a function $\hat{f} : X \to \{true, false\}$ *using* D *such that*

$$\hat{f}(x) \cong f(x) \tag{1.1}$$

for all $x \in X$.

Let us take a look at this definition in more detail. The data universe X is the set of objects of interest. For example, this might be a set of celestial objects viewable in a photograph taken through a telescope; it could also be a set of persons who visited a particular web page; or it could be a collection of proteins whose function in the cell and three-dimensional structure are known. The sample set S is a subset of the data universe. In general, the sample set is necessary since most collections of objects we are interested in tend to be very large or perhaps infinite, and building models can be very slow for large data universes and impossible for infinite data universes. Therefore, the sample set S acts as a representative of the data universe in order to make the process of building models tractable. The target function f is the process that provides the observable labels. It is assumed that f is able to provide a suitable value in $\{true, false\}$ for any element in X when that element is observed. Thus, even though we have no direct access to the process itself, we are always able to observe the labels this process assigns to the elements in the data universe. For example, when we interpret a photograph, a target function f might label celestial objects viewable in the photograph according to whether or not they are stars. We use this property of the target function to construct the training set D by observing the labels for objects in the sample set S. As an aside, machine learning that makes use of labeled training data is referred to as *supervised learning*. There are other types of machine learning, referred to as *unsupervised learning*, that do not need labeled training data. Finally, equation (1.1) in our definition of machine learning formally states that learning can be viewed as computing the function $\hat{f}$ as an approximation to or a model of the original process f based on the training examples in D. That is, the result of machine learning is a model of the original labeling function. However, out of convenience we often say that $\hat{f}$ is a model of the training data D. This is compatible with the formal view expressed in (1.1) because the elements in the training set are input–output pairs of the original labeling function, $(x, y) \in f$ with $x \in S$ and $y = f(x)$, and this means that modeling the function f and modeling the training data D are one and the same.

The names of the labels in $\{true, false\}$ are arbitrary; instead of *true* and *false* we could have used T and F, 0 and 1, or $blue$ and $green$. The important fact is that this set contains two distinct labels: one for class membership and one for nonmembership. We can also consider classification problems with more than two possibilities. The only difference from our definition of machine learning above would be that the codomain of the original labeling process f and its model $\hat{f}$ is a set that includes an appropriate number of distinct labels.

Once we have a model of the original labeling process, two interesting things can be accomplished. First, we can use the model to compute or *predict* the label of an element in the data universe X without having to observe this element. Second, the model can provide some insight into the original labeling process. That is, a model possesses some *explanatory* ability. Consider the following scenario, where the data universe X represents all the customers of a bank. Now, assume that a model $\hat{f}$ classifies the customers according to who is likely to default on a mortgage (*true*) and who is not (*false*). The bank can now use this model to predict which of its customers are likely to default on their mortgage payments before the event is observable, and is able to take actions such as offering refinance or debt management options. The bank can also use the model to discover which features of the data universe X are most relevant to the prediction; that is, the model can tell the bank the characteristics of a bank customer who is likely to default. These characteristics can take on the form of multiple maxed-out credit cards or perhaps a large, high-interest home equity loan.

1.2 STRUCTURE OF THE UNIVERSE X

As varied as the objects in a data universe may be, they can usually be described by a collection of *features* or *attributes*. The most common way to represent a set of objects is as a table where each feature is given as a table column and each object is a row in the table. Table 1.1 is a table representing a subset S of the data universe X of all objects with legs. We have five objects in this set. Each object in S is described by four features:

1. *Legs*: the number of legs the object has
2. *Wings*: yes if the object has wings; otherwise, no
3. *Fur*: yes if the object has fur; otherwise, no
4. *Feathers*: yes if the object has feathers; otherwise, no

When we apply a labeling process such as *mammal* to an object in S (e.g., Cat), we actually apply the labeling process to the feature set of that object. The name of the object does not carry any information; it is the description or representation of that object that matters during classification. That is, *mammal*(Cat) is shorthand for *mammal*(4, no, yes, no). If we had called our cat "Jup," *mammal*(Jup) would still be shorthand for *mammal*(4, no, yes, no) because the nature of the object did not change. Therefore, we ignore the names of the objects and view our set S as a subset of the cross-product of our features; that is, S is a subset of all possible object descriptions

TABLE 1.1 Simple Feature Table for a Small Number of Objects

	Legs	Wings	Fur	Feathers
Cat	4	no	yes	no
Crow	2	yes	no	yes
Frog	4	no	no	no
Bat	4	yes	yes	no
Barstool	3	no	no	no

that we can generate given our four features. In our case we have

$$S \subset \text{Legs} \times \text{Wings} \times \text{Fur} \times \text{Feathers}, \tag{1.2}$$

and the description of Cat is a member of S according to Table 1.1, (4, no, yes, no) $\in$ S. Since we view S as a subset of our data universe X, it follows that

$$X \subseteq \text{Legs} \times \text{Wings} \times \text{Fur} \times \text{Feathers}. \tag{1.3}$$

Each object in our data universe X is described by four features.

We construct the training data set D by applying the target function *mammal* to each object in S:

$$\begin{aligned} \mathit{mammal}(4, \text{no}, \text{yes}, \text{no}) &= \mathit{true}, \\ \mathit{mammal}(2, \text{yes}, \text{no}, \text{yes}) &= \mathit{false}, \\ \mathit{mammal}(4, \text{no}, \text{no}, \text{no}) &= \mathit{false}, \\ \mathit{mammal}(4, \text{yes}, \text{yes}, \text{no}) &= \mathit{true}, \\ \mathit{mammal}(4, \text{no}, \text{no}, \text{no}) &= \mathit{false}. \end{aligned}$$

The training data can also be represented as a table and is shown in Table 1.2. Here we dropped the names of the objects from the table altogether since they do not add any information. It is typical that in this representation of the training data the class label is made into an additional feature often called the *dependent attribute*.

Looking at the training data we see an interesting pattern emerging, in that being a mammal seems to be highly correlated with having fur. So perhaps a reasonable model $\hat{f}$ for the labeling process *mammal* is

$$\hat{f}(\mathit{legs}, \mathit{wings}, \mathit{fur}, \mathit{feathers}) \equiv \textbf{if } \mathit{fur} = \text{ yes } \textbf{then } \mathit{true} \textbf{ else } \mathit{false}. \tag{1.4}$$

In other words, given any object in our data universe the model tests the input value *fur*, and if it is set to yes it will return *true*; otherwise, it will return *false*. If our training set is representative, our model will approximate the original labeling process over the entire data universe:

$$\hat{f}(x) \cong \mathit{mammal}(x) \tag{1.5}$$

TABLE 1.2 Training Data as a Table

Legs	Wings	Fur	Feathers	Mammal[a]
4	no	yes	no	*true*
2	yes	no	yes	*false*
4	no	no	no	*false*
4	yes	yes	no	*true*
3	no	no	no	*false*

[a]The label observed for each object in the table.

for all $x \in X$. Here we used a pattern found in the training set to construct a model and inferred that this model will approximate the labeling process *mammal* over the rest of the data universe. This type of reasoning is called *inductive learning*.

1.3 INDUCTIVE LEARNING

Our definition of machine learning (Definition 1.1) expresses an *inductive process* where, given a limited amount of data in the form of a training set, we try to induce a function that approximates the original labeling process over the entire data universe. That is, we *generalize* from specific instances in the training set D to the entire data universe X. We call this *inductive learning*. At the heart of inductive learning lies the assumption that the training set is an accurate representation of the entire universe. This assumption is formalized in the following hypothesis.

Inductive Learning Hypothesis *Any function found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over unobserved examples.*

The intricacies of inductive learning can be illustrated by the *black swan problem*. Consider Figure 1.2. Here the set X denotes the universe of all possible swans (i.e., black and white swans) and the set D denotes the training set for a machine learning algorithm. From this training set a learning algorithm might infer a model in which all swans are white, or more formally,

$$\hat{f}(x) = white \tag{1.6}$$

for all $x \in X$. This is clearly only an approximation to the original process,

$$f : X \rightarrow \{white, black\}, \tag{1.7}$$

which labels most of the swans white but also labels some swans black. Our model $\hat{f}$ would be a poor choice for answering scientific questions on the color of swans. On the other hand, if 99% of the swans in the world are white, our model has an accuracy of 99% when evaluated over the entire data universe. This means that it is a pretty good model if we want an approximation of the color of swans.

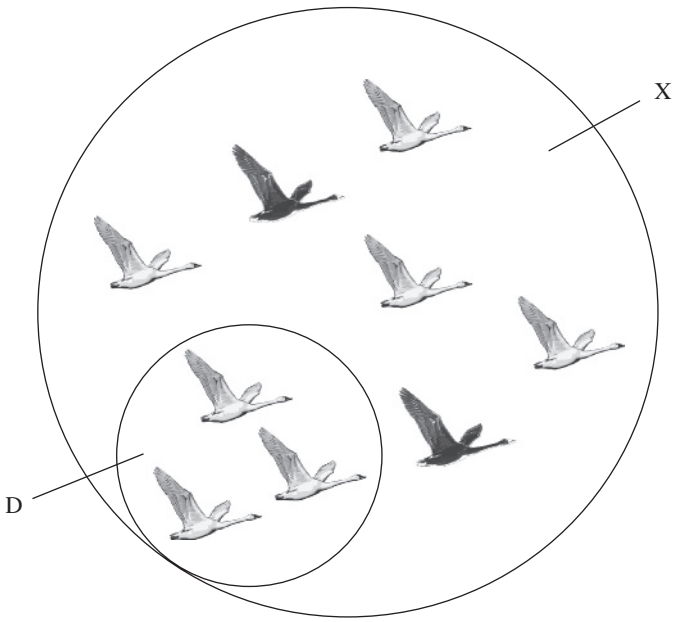

FIGURE 1.2 Data universe X and training set D for the *black swan problem*.

The question of inductive learning and whether or not a training set is a good representation of a data universe depends on the application of the ensuing models and is not a clear-cut proposition. It is desirable, however, to construct a training set as representative of the data universe as possible; that is, it is desirable to construct a training set that is "sufficiently large." In our case we should try to include at least one black swan in the training data set. Sophisticated techniques from statistical sampling theory can be used to ensure that the training data are "large enough." However, ultimately there will always be some uncertainty about the objects captured in our training set. We will study techniques that help us to evaluate some of this uncertainty and, with it, the generalization ability and expected accuracy of our models.

1.4 MODEL REPRESENTATIONS

Since we want the approximation $\hat{f}$ of the target function f to be computable, we are interested in appropriate representations of the models $\hat{f}$. Typically, we consider two types of model representations:

1. Transparent representations (or transparent models)

   a. If–then–else rules
   b. Decision trees

2. Nontransparent representations (or nontransparent models)

   a. The weights on the connections between the elements in an artificial neural network
   b. The linear combination of vectors in support vector machines

Transparent models are representations that can be interpreted by human beings unaided; nontransparent models cannot be interpreted unaided. For example, we can interpret if–then–else rules very easily just by looking at the rule text. On the other hand, we can examine the weights in an artificial neural network without ever fully understanding exactly how the neural network stores its learned information in these weights.

The representation of models is an important topic because it dictates how well we can model certain target functions. In machine learning theory this is referred to as *language bias*. Consider a data table as a model representation. In this case the model representation simply mirrors the training set and therefore memorizes all the objects in the training set. This means that this model will have perfect knowledge of the objects in the training set, but it will fail to produce any meaningful results for objects not in its table. Assume for a moment that our model is represented by the training data for all objects with legs given in Table 1.2. This model can certainly answer questions on objects, such as (4, yes, yes, no), that are in the table simply by looking up the object that matches the features of the query and returning the value stored in the dependent attribute as the answer:

$$(4, \text{yes}, \text{yes}, \text{no}) \mapsto \textit{true}. \tag{1.8}$$

But this model cannot answer questions on objects that are not in its table. Consider

$$(2, \text{no}, \text{no}, \text{no}) \mapsto ? \tag{1.9}$$

This means that our model does not generalize beyond the objects found in its table and therefore is a poor choice as an approximation of the original labeling process over the data universe.

The limitations of this model are due to the fact that we chose the training data table as our model implementation. Now, consider another type of representation: The model consists of a constant. Regardless of what type of object the model is handed, it will always generate the same constant response. We have seen this above in the swan example, where the model always produces the response *white*. If we pick the constant to be the majority label in the training set, in our case the label *false*, this simple model will make mistakes on the training set. However, if the training set is an accurate representation of the data universe as a whole, we can expect that the model will have the same or similar accuracy on the data universe as for the training set. Thus, we can say that the model does generalize to a certain extent; it at least encodes the majority label in its simple structure and uses this single piece of information to assign labels to unobserved objects.

Model representations such as decision trees, neural networks, and support vector machines fall somewhere in between the two extremes above. The algorithms that give rise to more sophisticated model representations discover regularities that relate objects to their corresponding labels, and these regularities are then encoded in appropriate model representations.

In the previous discussion we have seen a simple decision rule model for our *mammal* target function that captured the regularity or pattern that being a mammal and having fur seems to be highly correlated. It is interesting to observe that, in general, transparent model representations lag in performance compared to nontransparent model representations. The constraint that a model is interpretable by people unaided seems to interfere with the modeling process, in that a transparent model is not able to classify certain phenomena as effectively as are nontransparent models.

EXERCISES

1.1 Explain in your own words what is meant by the statement *a model generalizes well*.

1.2 Briefly explain what *inductive learning* means.

1.3 Consider the training set given in Table 1.2. Write a program that will detect the perfect correlation between the *fur* attribute and the *mammal* labels and outputs this as a model along the lines of equation (1.4).

1.4 Write a program that can accept any training data set of the form given in Table 1.2 and that computes a majority label model. You can assume that the last attribute of the training table is always the dependent attribute.

1.5 Consider a large set of a variety of objects and make that your data universe X. Now consider the labeling function $bird: X \rightarrow \{true, false\}$ that labels each object in X as a bird (or not). Design a model $\hat{f}: X \rightarrow \{true, false\}$ that could be implemented on a computer that approximates the original function *bird*. What is your feature set? Now take a subset D of X as your training data. Analyze where and how your model makes mistakes when it is applied to D and/or to X.

1.6 Consider a naturally occurring phenomenon around you. Construct a classification model for it using machine learning. What is the data universe? What is the feature set? What are the labels? Can you estimate the accuracy of your model?

BIBLIOGRAPHIC NOTES

A readable introduction to machine learning from a computer science perspective is Mitchell's book [54]. Our definitions of machine learning and the inductive learning

hypothesis closely follow Mitchell. A comprehensive and recent overview of the field of machine learning and pattern recognition is [10]. A more statistical view of machine learning can be found in books by Hastie et al. [36] and Gentle [34]. Quinlan's C4.5 and Breiman's et al. CART decision tree algorithms are described in detail in [62] and [16], respectively. An excellent description of neural networks is Bishop's book [9]. An older but interesting collection of papers dealing with the knowledge discovery process is [31]. A particularly gentle introduction to data mining is [2]. Data mining from the perspective of particular application areas such as customer support is discussed in [8]. Data preparation and data warehousing are discussed in [61] and [43], respectively. Perhaps the best known formalization of knowledge discovery and data mining is the CRISP methodology (http://www.crisp-dm.org). The earliest reference to the *black swan problem* we are aware of is [60].