About this book

If you want to make it right,
make it wrong first.

What it is about

This book is about *knowledge discovery*. There are many excellent books on machine learning and data mining. And there are many excellent books covering many particular aspects of these areas. Even though all the knowledge we are concerned with in computer science is *relational*, relational or logic machine learning or knowledge discovery is not that common. Accordingly, there are fewer textbooks on this issue.

This book strongly emphasises knowledge: what it is, how it can be represented, and, finally, how new knowledge can be discovered from what is already known plus a few new observations. Our interpretation of knowledge is based on the notion of "discernability"; all the methods discussed in this book are presented within the same paradigm: we take "learning" to mean acquiring the ability to discriminate between different things. Because things are different if they are not equal, we use a "natural" equivalence to group similar things together and distinguish them from differing things. Equivalence means to have something in common. According to the portion of commonness between things there are certain degrees of equality: things can be exactly the same, they can be the same in most cases or aspects, they can be roughly the same, not really the same, and they can be entirely different. Sometimes, they are even incomparable.

There are several well-known ways of describing similarities between sets of objects. If we arrange all our objects by their (measurable) properties, then their mutual distance to each other reflects their similarity. And if there are two different objects that have a zero distance, we have to find another property that will distinguish them. If all the objects are described by a set of features, then similarity means something like the number of features in which they agree. The utility of a feature for finding new knowledge is its information content. Because

features induce equivalence relations, many features create many such relations. And by intersecting them, we gain a very fine-grained partitioning of the universe in many, many small classes of, well, equivalent or equal or similar things. Finally, we can describe objects and concepts by logic formulae or theories. Then, knowledge discovery means refining our set of formulae so that we are able to deduce something that we were not able to infer before.

Every paradigm is dedicated a chapter on its own.

How it is organised

This book tries to illustrate the common ideas behind several different approaches to machine learning or knowledge discovery. If you take a look at a set of books each of which specialises in any of these areas, you will find idiosyncratic notation in each of them. This does not really help in understanding the common processes and the parts in which they differ. And it is important to understand the differences between them to gain knowledge about them. It is also the differences that make one or another paradigm more suitable in a certain domain. Therefore it is important to be able to see them clearly. As a consequence this textbook has a leitmotif: we will always be speaking about simple geometric shapes like △, ●, or ■ – their differences, their common properties and how to construct different concepts like "grey boxes" or "things with at most n corners". If you consider this as a plus for reading this book, then I hope you consider the following a plus as well. To stress the common characteristics of the theories, we need a common language. As a result, I have tried to find a more or less consistent notation or notational principle (like "a $\cup$ is to a $\subseteq$ as $\sqcup$ is to $\sqsubseteq$; and $\longrightarrow$ is to $\Longrightarrow$ as $\vdash$ is to $\models$"). I think it is a nice idea to use the same notation throughout a book covering several topics that usually use different notations. But the downside is that the result is another nomenclature. I beg the reader's forgiveness for using Greek letters, upright and slanted function names, fraktur characters, and symbols you will never see elsewhere (unless you attend my classes).

The language used in this book is English with a German accent. In addition, I have tried to find a delicate balance between informal written text and a rather formal and exact notation. The text explains what all the formulae are about – and the formulae are there to have an indisputable and solid foundation for describing things. Additionally, there are many examples. As mentioned above, there is the running example of geometric shapes. But there are others from everyday life, some famous examples, and also some rather surprising ones that require very special knowledge in areas that not all readers will be familiar with.

However, if you are, I hope they are even more illustrative (did you ever notice that it takes three variables in Lambda-calculus to define exclusive disjunction?).

Then, there are exercises. They are for self-control only; solutions are not provided. You may understand the questions as just a few hints of directions for further thinking. I labelled the questions with marks from ◇ to ◆. The ◇ exercises should not require more than just a few minutes of thinking, some reading in other books, or performing simple calculations using the formulae from the text. ◈ exercises require a bit more thinking, for example, simple proofs or some questions that require a deeper understanding. Exercises with a ◆ mark are questions that go beyond the scope of the book. They might require longer proofs, some thinking about how to overcome hidden problems in the methods described in the text, or even writing a small program for solving longer calculations. All marks just represent the *suggested* effort that is worth spending on the question. It does not say anything about the actual hardness of the problem or the time you should spend on it.

Finally, there are "knowledge boxes." They are small grey boxes like this:

Box of knowledge
A box of knowledge summarises the relevant results of a section in a punchline, preferably in prose. By reading them alone, you ought to be able to tell someone else what this book is about and even explain the most important concepts in your own words.

Thanks to:

Helmar, who taught me to ask the right questions; Ivo, who was the first to introduce me to the beauty of formal thinking; and Bernhard, with whom I discovered the combination of both.

Alexander, Jonghwa, and Peter for friendship, help, and support.

All researchers I met during the past 15 years for their inspiration, discussion, clarification, and criticism.

All students who by their bravery and willingness to pass the exams contributed to all the previous versions.

David Tranah from CUP and Ali Jaoua, Simon Parsons, Andrzej Skowron, Harrie de Swart, George Tourlakis, and Michael Winter for inspiring discussion, useful comments, and proof reading.

Chapter 1
Introduction

Knowledge discovery, machine learning, data mining, pattern recognition, and rule invention are all about algorithms that are designed to extract knowledge from data and to describe patterns by rules.

One of the cornerstones of *(traditional) artificial intelligence* is the assumption that

> Intelligent behaviour requires rational, knowledge-based decisive and active processes.

These processes include the acquisition of new knowledge, which we call *machine learning* or *knowledge discovery*. However, when talking about *knowledge-based systems* we first need to explain what we mean by *knowledge*. If we try to define learning by intelligence, we need to explain intelligence, and if we want to explain intelligence, we need to explain knowledge. Bertrand Russell (1992, 1995) has given a very precise and in our case very helpful (and actually entirely sufficient) definition of *knowledge*:

> Knowledge is the ability to discriminate things from each other.

As a consequence, learning means acquiring the ability to recognise and differentiate between different things. Thus, the process of knowledge acquisition is a process that is initiated and (autonomously) run by a system whose purpose is to learn by itself. L. G. Valiant (1984) said that

> Learning means acquiring a program without a programmer.

To us, it means:

Learning as discovery of knowledge
Learning means acquiring the ability to discriminate different things from each other without being told about every single instance.

1.1 Motivation

Like any other computer science or artificial intelligence discipline, machine learning research evolves in many dimensions. This includes the interpretation of the learning process as a data processing technique, a rule discovery tool, or a model of cognitive processes. Machine learning (or an aspect of it) can also be described in terms of its successful application in the real world (whatever one might consider a successful result).

1.1.1 Different kinds of learning

Engineering and Theory: As an engineer or computer scientist who seeks patterns in data sets, one might ask how to extract knowledge from huge databases and how to make knowledge elicitation as efficient as possible. If, on the other hand, you are interested in the theory of computation, then it would be much more interesting to see if there are fundamental limitations of learning in terms of complexity or in terms of the problem classes.

Data-Driven Learning and Conceptual Learning: Data-driven learning means to take all data (or, rather, observations) without much further information about it and try to extract as much knowledge as possible. This means that data-driven learning focuses on what one can learn from the supplied data. However, quite often we already have a more or less precise image of what we think the world is like, i.e., an underlying model of the data. We then use a set of known concepts that a learning algorithm uses to describe unknown target concepts.

The difference is that in data-driven learning one tries to identify clusters of similar objects from a set of observations. On the other hand, *conceptual learning* supplies our algorithm with background knowledge about the world. As an example, data-driven learning may help in the discovery of classes like *mammals* and *birds*. Using knowledge about *habitats* and *domestication*, conceptual learning is able to describe penguins and polar bears by their habitat and it can tell a dog from a wolf by their domestication.

Clustering or Classification and Scientific Discovery: Engineers are often faced with huge sets of data, and the larger the sets are, the less

is known about the (hidden) structures they may hold. In the course of developing growing data warehouses, some system operators are in need of handling petabytes of data. With too much data around and too little knowledge about them, one needs to devise algorithms that efficiently group similar cases into the same classes.

Classification means assigning a new unseen case one of the given class labels, but scientific discovery is rather concerned with finding new subclasses or relational dependencies between them. If such class hierarchies and dependencies are expressed in terms of rules, then the invention of a new concept and its description is what we call *scientific discovery*.

Algorithms and Cognitive Processes: Similar to the engineering–theory dichotomy, one can also consider the algorithmic issues in data mining or understand machine learning as a metaphor for human learning. For example, many data sets can be explained using decision trees – but using modules of artificial neural networks one can evaluate psychological models of human problem solving, too.

Data mining is a multi-stage (business) process leading to the automated detection of regularities in data that are useful in new situations. Particularly in the context of very large data sets, knowledge can be compared to a gem that is very well hidden under a huge mountain of rock-hard data. Knowledge discovery requires the extraction of

1. *implicit* (but hidden), previously *unknown*
2. and potentially *useful information* from
3. *data*

or even the search for relationships and global patterns that exist in databases.

1.1.2 Applications

Nowadays, knowledge discovery has become a very common technique in the (semantic) web. Its popularity in computational biology – especially in genetics or large-array marker scans – is still increasing. For example, sequencing of the genetic code allows us to understand the encoded protein – given that we can understand how tertiary structures of proteins develop during folding. Similarly, pattern recognition on marker arrays help in the identification of genomic defects and diseases, and the spatial properties of molecules can be expressed using a language of relations. It would be very interesting to explain – in terms of molecule structures – why some chemicals are carcinogenic while others are not (Muggleton et al. 1992, 1998).

Finding patterns and making up rules from them is a very popular research topic: nearly all observations consist of complex patterns from which we try to abstract by generalising. Sometimes, observations yield similarities that help us form class representatives (both a penguin and a sparrow are birds, but it is the sparrow that is a prototypical bird). Spam classifiers take emails as patterns of word occurrences and use rules defined by filters to keep your mailbox clean.

Recently, data mining has become one of the most important application areas of knowledge discovery. Just recall how it was a few years ago when you tried to find some information in data collections: the first problem was that we did not have enough data available – that is, we knew what kind of information we were looking for and the amount of available data could be easily surveyed. The result was that we could not find the information we needed simply because it was not there. The problem of second generation information retrieval systems was slightly different. With that, there was enough data available, but how could we effectively or efficiently find it? Early solutions required data items to be tagged with metadata until more powerful indexing and search methods were developed.

Example 1.1 Information retrieval in the World Wide Web is a very clear example. In the early 1990s, the Web was so small that personal link lists (and those of others) were sufficient for exhaustive searching. The next step introduced search engines like Yahoo (with manually tagged indices) and AltaVista and many other services competing for the largest search indices. Today, we simply "*Google*" the Web for information.

With an ever increasing amount of data available, the next question became how to integrate it all. The answer was data warehouses. In 2005, Yahoo was reported to maintain a 100-PetaByte data warehouse and AT&T was using two data warehouses with a total 1.2 ExaByte of data. The question that arose then was, what kind of information is hidden within all these data? And this is what *knowledge discovery* is all about. Commercially, it is referred to as *data mining* – because this is what we do to find some "gems", that is, important pieces of information, in the huge pile of data. There is a small difference, though: we use the term "knowledge discovery" to describe the process of extracting new knowledge from a set of data about that set of data. This means that the acquisition of new knowledge requires us to build a new model of the data. "Data mining" refers mostly to the extraction of parts of information with respect to a *given* model. One crucial problem is the interpretation of correlation: if two things correlate, it does not necessarily mean there exists some kind of causal dependency between

them. This is where *relational* knowledge discovery enters the game: here, the primary interest is in the *relations between relations* and not the relations between *objects*.

Last, but not least, knowledge discovery, data mining, and machine learning are tools that can be used in any situation where the problem we are faced with is ill-posed. So if we are not able to devise an efficient deterministic algorithm that solves our problem within a predefined instance space, we simply give it a try and let the machine learn by itself.

In quite a few cases, the results are surprisingly good; in other cases, they are not. But then, we can at least blame the machine for being a bad learner rather than blame ourselves for being bad programmers.

1.2 Related disciplines

To us, knowledge discovery means learning how to discriminate between different objects. There are other disciplines that have slightly different understandings of the term "knowledge". But most of them also develop techniques that somehow correspond to what we call learning.

1.2.1 Codes and compression

The motivation behind coding is representing a *meaningful message* by a suitable sequence of symbols. The representation process is called *encoding* and maps *plain text (symbols)* or *(source) messages* onto *codes* that are other symbol strings. A one-to-one substitution of source symbols onto code symbols is called *enciphering*, and the result a *cipher*. The reverse processes of reconstructing the original message from a code (cipher) is called *decoding* (*deciphering*).

We assume the reader is familiar with the sequence of *Fibonacci numbers*. Yet, if asked, no one will ever reply by saying "Fibonacci numbers? Sure! 1, 1, 2, 3, 5, 8, . . ." So, even if you are able to memorise the entire sequence of Fibonacci numbers, you have not *learned* anything about them because learning would require the ability of (intensionally) explaining a *concept*.[1] This also relates to *data compression*: The less compressible our data, the more information they contain and the harder it is to learn and compress them further. Any

[1] Note the difference between *intentional* and *intensional*. Explaining something intentionally means to explain it with a certain intention in mind. But explaining something intensionally (as opposed to extensionally) means to explain it by abstract concepts instead of concrete examples.

good learner needs to be able to *compress* data by giving a rule that can describe them.

Example 1.2 RLE compression. One of the simplest methods of encoding and compressing a stream of symbols is *run length encoding*. Consider the alphabet $\Sigma = \{\square, \bigcirc\}$. Then, strings will contain repetitive occurrences of each symbol. For example,

A reasonable way of compressing such a sequence would be to precede each symbol by the number of times it occurs before another symbol appears. And if we assume that each sequence starts with a $\square$, we can even drop the symbol itself, because after every sequence of $\square$s there can only be a sequence of $\bigcirc$s, and vice versa. So, the above string can be compressed to

$$2.1.1.5.4.2.1.3.3.1.\cdots$$

which is certainly much shorter but requires a larger alphabet of symbols and a special delimiter symbol ".". ●

Finding (optimal, shortest) codes (that is, *encoding functions*) without the need for delimiters or any additional symbols is the topic of coding theory.

The notion of compression enters the game at two different points: first, a message that cannot be compressed further is free of any redundancies. Then, the message string itself must have maximum entropy and all the symbols occurring in the message are more or less of the same probability. However, they do *not* encode the same "amount of information": changing one symbol can result in just a small change after decoding, but it can also scramble the entire encoded message into a meaningless sequence of symbols. Second, a strong concept requires a complex representation language that basically is the same as finding a good *code*.

One example that we shall encounter is the encoding of messages (or hypotheses) into strings of symbols ("genomic codes"). The field of coding theory and compression has become a huge discipline of its own, which is why we refer to MacKay (2003). If you are more interested in coding and cryptography, consult Welsh (1988); and for the advanced reader we recommend Chaitin (1987), Kolmogorov (1965), and Li and Vitanyi (1993).

Example 1.3 There is a crucial difference between encoding the numbers $0, 1, 2, \ldots, 255$ using a three-digit decimal representation $x \cdot 10^2 + y \cdot 10^1 + z$ with $x, y, z \in \{0, 1, 2, \ldots, 9\}$ and by using a binary representation with eight bits $w \in \mathbf{2}^8$.

In both cases, changing one symbol creates an error of at least 1. The least dramatic change in the decimal representation is to replace z by $z+1$ or $z-1$, and in the binary representation one simply flips the least significant (that is, the last) bit. Things are different if we take a look at the maximum error possible in each representation formalism. For the binary representation, the maximum error corresponds to the value of the most significant bit – and that is 128. But for the decimal representation it is $9 \cdot 10^2 = 900$!

Imagine that the integer we want to encode has the value "three". Then, **10000011** − **00000011** is "one-hundred thirty-one minus three", which equals $2^8 = 128$. But the maximum error in the decimal representation is the result of maximising the difference on the most significant decimal place: 903 − 003 is "nine-hundred".

If you argue that 900 is not within the range of integers covered by our example, then imagine that $x \in \{0, 1, 2\}$ and $x + y + z \leq 255$. Then $203 - 003 = 200$ is still nearly twice as much as the maximum error of the binary representation.

It is clear that a sequence of symbols that can be compressed pretty well (e.g., by RLE) obviously contains some kind of redundancy. If we know that the next five symbols we receive are □s, then the sender's effort in transmitting □□□□□ is simply a waste of time (and, as we shall see, a waste of bandwidth).

Machine learning and coding
Learning means finding an optimal code to describe observations.

Coding and its role in cryptography are far beyond the scope of this book, but they are indispensable for *information theory* as well. The basic idea behind information theory is that the average randomness of a sequence (and thus its reverse redundancy) is a measure of the complexity of the system that emits the messages.

1.2.2 Information theory

There is much confusion about the term "information". Together with entropy, complexity, and probability, terminology often gets in the way of a proper understanding. To avoid misconceptions from the very beginning, we first and foremost need to make clear one crucial point:

Information
Information is not so much about *what* is being said, but rather what *could* be said.