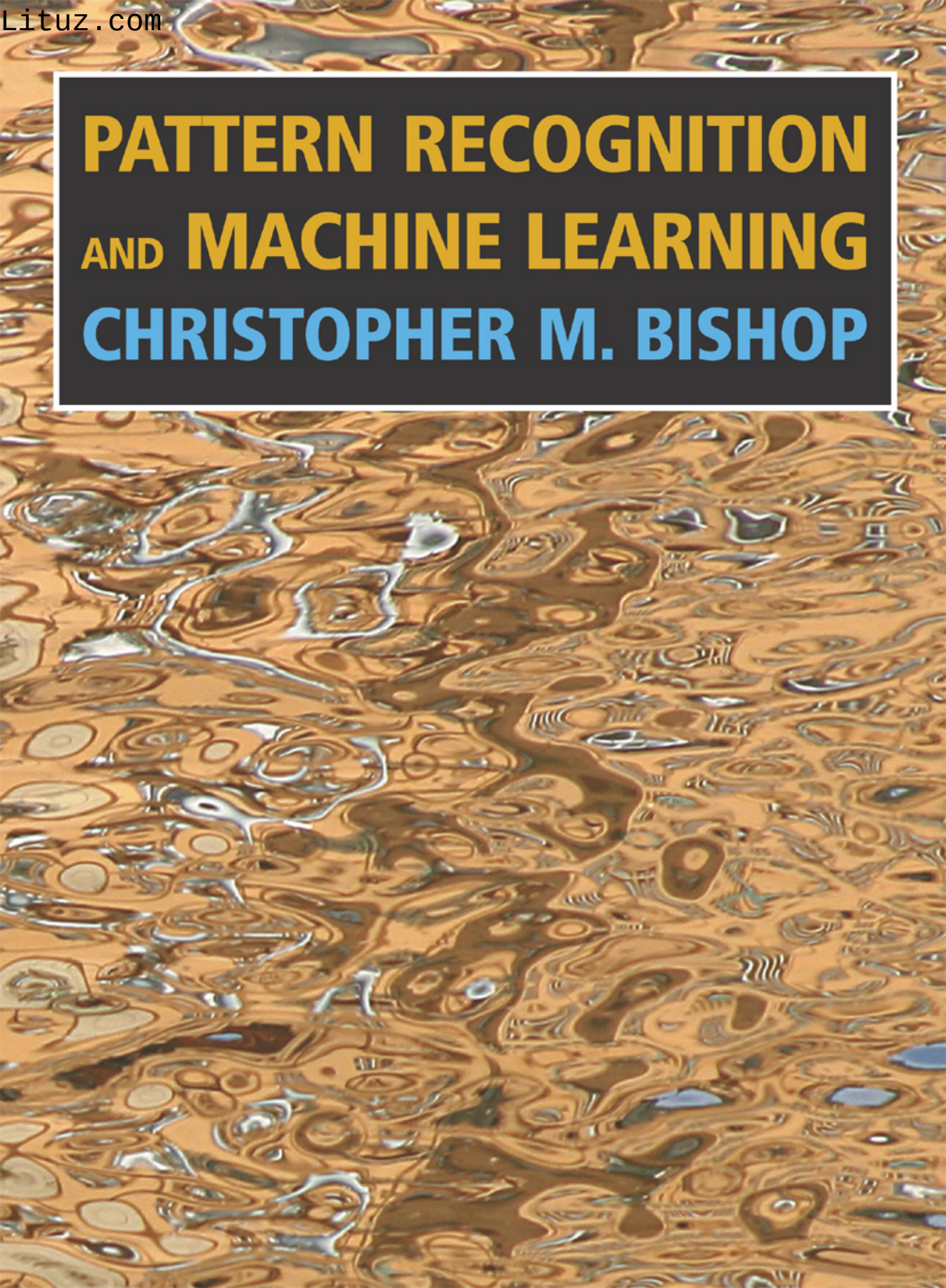

PATTERN RECOGNITION
AND MACHINE LEARNING
CHRISTOPHER M. BISHOP

Information Science and Statistics

Information Science and Statistics

Akaike and Kitagawa: The Practice of Time Series Analysis.
Bishop: Pattern Recognition and Machine Learning.
Cowell, Dawid, Lauritzen, and Spiegelhalter: Probabilistic Networks and Expert Systems.
Doucet, de Freitas, and Gordon: Sequential Monte Carlo Methods in Practice.
Fine: Feedforward Neural Network Methodology.
Hawkins and Olwell: Cumulative Sum Charts and Charting for Quality Improvement.
Jensen: Bayesian Networks and Decision Graphs.
Marchette: Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint.
Rubinstein and Kroese: The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation, and Machine Learning.
Studený: Probabilistic Conditional Independence Structures.
Vapnik: The Nature of Statistical Learning Theory, Second Edition.
Wallace: Statistical and Inductive Inference by Minimum Massage Length.

Christopher M. Bishop

Pattern Recognition and Machine Learning

Springer

Christopher M. Bishop F.R.Eng.
Assistant Director
Microsoft Research Ltd
Cambridge CB3 0FB, U.K.
cmbishop@microsoft.com
http://research.microsoft.com/~cmbishop

Library of Congress Control Number: 2006922522

ISBN-10: 0-387-31073-8
ISBN-13: 978-0387-31073-2

Printed on acid-free paper.

Printed in Singapore. (KYO)

9 8 7 6 5 4 3 2 1

springer.com

This book is dedicated to my family:

Jenna, Mark, and Hugh

Total eclipse of the sun, Antalya, Turkey, 29 March 2006.

Preface

Pattern recognition has its origins in engineering, whereas machine learning grew out of computer science. However, these activities can be viewed as two facets of the same field, and together they have undergone substantial development over the past ten years. In particular, Bayesian methods have grown from a specialist niche to become mainstream, while graphical models have emerged as a general framework for describing and applying probabilistic models. Also, the practical applicability of Bayesian methods has been greatly enhanced through the development of a range of approximate inference algorithms such as variational Bayes and expectation propagation. Similarly, new models based on kernels have had significant impact on both algorithms and applications.

This new textbook reflects these recent developments while providing a comprehensive introduction to the fields of pattern recognition and machine learning. It is aimed at advanced undergraduates or first year PhD students, as well as researchers and practitioners, and assumes no previous knowledge of pattern recognition or machine learning concepts. Knowledge of multivariate calculus and basic linear algebra is required, and some familiarity with probabilities would be helpful though not essential as the book includes a self-contained introduction to basic probability theory.

Because this book has broad scope, it is impossible to provide a complete list of references, and in particular no attempt has been made to provide accurate historical attribution of ideas. Instead, the aim has been to give references that offer greater detail than is possible here and that hopefully provide entry points into what, in some cases, is a very extensive literature. For this reason, the references are often to more recent textbooks and review articles rather than to original sources.

The book is supported by a great deal of additional material, including lecture slides as well as the complete set of figures used in the book, and the reader is encouraged to visit the book web site for the latest information:

http://research.microsoft.com/∼cmbishop/PRML

Exercises

The exercises that appear at the end of every chapter form an important component of the book. Each exercise has been carefully chosen to reinforce concepts explained in the text or to develop and generalize them in significant ways, and each is graded according to difficulty ranging from ($\star$), which denotes a simple exercise taking a few minutes to complete, through to ($\star\star\star$), which denotes a significantly more complex exercise.

It has been difficult to know to what extent these solutions should be made widely available. Those engaged in self study will find worked solutions very beneficial, whereas many course tutors request that solutions be available only via the publisher so that the exercises may be used in class. In order to try to meet these conflicting requirements, those exercises that help amplify key points in the text, or that fill in important details, have solutions that are available as a PDF file from the book web site. Such exercises are denoted by **www**. Solutions for the remaining exercises are available to course tutors by contacting the publisher (contact details are given on the book web site). Readers are strongly encouraged to work through the exercises unaided, and to turn to the solutions only as required.

Although this book focuses on concepts and principles, in a taught course the students should ideally have the opportunity to experiment with some of the key algorithms using appropriate data sets. A companion volume (Bishop and Nabney, 2008) will deal with practical aspects of pattern recognition and machine learning, and will be accompanied by Matlab software implementing most of the algorithms discussed in this book.

Acknowledgements

First of all I would like to express my sincere thanks to Markus Svensén who has provided immense help with preparation of figures and with the typesetting of the book in LaTeX. His assistance has been invaluable.

I am very grateful to Microsoft Research for providing a highly stimulating research environment and for giving me the freedom to write this book (the views and opinions expressed in this book, however, are my own and are therefore not necessarily the same as those of Microsoft or its affiliates).

Springer has provided excellent support throughout the final stages of preparation of this book, and I would like to thank my commissioning editor John Kimmel for his support and professionalism, as well as Joseph Piliero for his help in designing the cover and the text format and MaryAnn Brickner for her numerous contributions during the production phase. The inspiration for the cover design came from a discussion with Antonio Criminisi.

I also wish to thank Oxford University Press for permission to reproduce excerpts from an earlier textbook, *Neural Networks for Pattern Recognition* (Bishop, 1995a). The images of the Mark 1 perceptron and of Frank Rosenblatt are reproduced with the permission of Arvin Calspan Advanced Technology Center. I would also like to thank Asela Gunawardana for plotting the spectrogram in Figure 13.1, and Bernhard Schölkopf for permission to use his kernel PCA code to plot Figure 12.17.

Many people have helped by proofreading draft material and providing comments and suggestions, including Shivani Agarwal, Cédric Archambeau, Arik Azran, Andrew Blake, Hakan Cevikalp, Michael Fourman, Brendan Frey, Zoubin Ghahramani, Thore Graepel, Katherine Heller, Ralf Herbrich, Geoffrey Hinton, Adam Johansen, Matthew Johnson, Michael Jordan, Eva Kalyvianaki, Anitha Kannan, Julia Lasserre, David Liu, Tom Minka, Ian Nabney, Tonatiuh Pena, Yuan Qi, Sam Roweis, Balaji Sanjiya, Toby Sharp, Ana Costa e Silva, David Spiegelhalter, Jay Stokes, Tara Symeonides, Martin Szummer, Marshall Tappen, Ilkay Ulusoy, Chris Williams, John Winn, and Andrew Zisserman.

Finally, I would like to thank my wife Jenna who has been hugely supportive throughout the several years it has taken to write this book.

Chris Bishop
Cambridge
February 2006

Mathematical notation

I have tried to keep the mathematical content of the book to the minimum necessary to achieve a proper understanding of the field. However, this minimum level is nonzero, and it should be emphasized that a good grasp of calculus, linear algebra, and probability theory is essential for a clear understanding of modern pattern recognition and machine learning techniques. Nevertheless, the emphasis in this book is on conveying the underlying concepts rather than on mathematical rigour.

I have tried to use a consistent notation throughout the book, although at times this means departing from some of the conventions used in the corresponding research literature. Vectors are denoted by lower case bold Roman letters such as $\mathbf{x}$, and all vectors are assumed to be column vectors. A superscript T denotes the transpose of a matrix or vector, so that $\mathbf{x}^{\mathrm{T}}$ will be a row vector. Uppercase bold roman letters, such as $\mathbf{M}$, denote matrices. The notation $(w_1, \ldots, w_M)$ denotes a row vector with M elements, while the corresponding column vector is written as $\mathbf{w} = (w_1, \ldots, w_M)^{\mathrm{T}}$.

The notation $[a, b]$ is used to denote the *closed* interval from a to b, that is the interval including the values a and b themselves, while (a, b) denotes the corresponding *open* interval, that is the interval excluding a and b. Similarly, $[a, b)$ denotes an interval that includes a but excludes b. For the most part, however, there will be little need to dwell on such refinements as whether the end points of an interval are included or not.

The $M \times M$ identity matrix (also known as the unit matrix) is denoted $\mathbf{I}_M$, which will be abbreviated to $\mathbf{I}$ where there is no ambiguity about it dimensionality. It has elements I_{ij} that equal 1 if $i = j$ and 0 if $i \neq j$.

A functional is denoted $f[y]$ where $y(x)$ is some function. The concept of a functional is discussed in Appendix D.

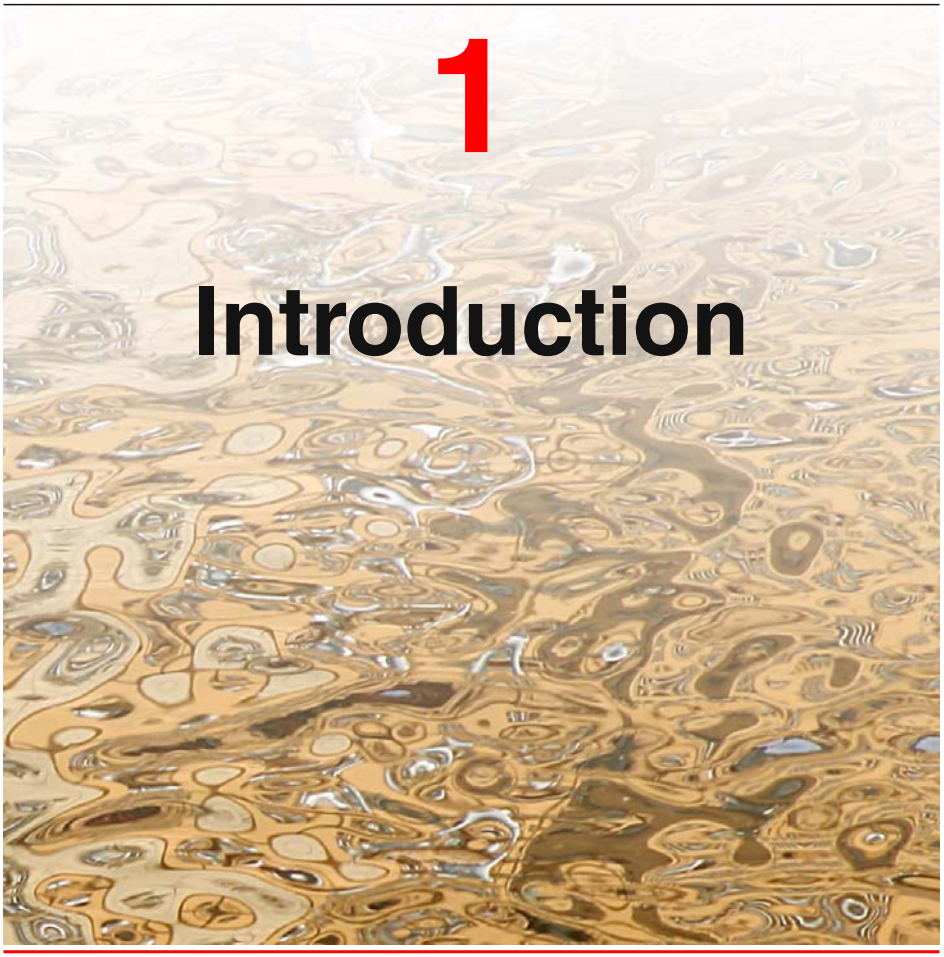
The notation $g(x) = O(f(x))$ denotes that $|f(x)/g(x)|$ is bounded as $x \to \infty$. For instance if $g(x) = 3x^2 + 2$, then $g(x) = O(x^2)$.

The expectation of a function $f(x, y)$ with respect to a random variable x is denoted by $\mathbb{E}_x[f(x, y)]$. In situations where there is no ambiguity as to which variable is being averaged over, this will be simplified by omitting the suffix, for instance

$\mathbb{E}[x]$. If the distribution of x is conditioned on another variable z, then the corresponding conditional expectation will be written $\mathbb{E}_x[f(x)|z]$. Similarly, the variance is denoted $\mathrm{var}[f(x)]$, and for vector variables the covariance is written $\mathrm{cov}[\mathbf{x}, \mathbf{y}]$. We shall also use $\mathrm{cov}[\mathbf{x}]$ as a shorthand notation for $\mathrm{cov}[\mathbf{x}, \mathbf{x}]$. The concepts of expectations and covariances are introduced in Section 1.2.2.

If we have N values $\mathbf{x}_1, \ldots, \mathbf{x}_N$ of a D-dimensional vector $\mathbf{x} = (x_1, \ldots, x_D)^{\mathrm{T}}$, we can combine the observations into a data matrix $\mathbf{X}$ in which the n^{th} row of $\mathbf{X}$ corresponds to the row vector $\mathbf{x}_n^{\mathrm{T}}$. Thus the n, i element of $\mathbf{X}$ corresponds to the i^{th} element of the n^{th} observation $\mathbf{x}_n$. For the case of one-dimensional variables we shall denote such a matrix by $\mathbf{x}$, which is a column vector whose n^{th} element is x_n. Note that $\mathbf{x}$ (which has dimensionality N) uses a different typeface to distinguish it from $\mathbf{x}$ (which has dimensionality D).

Contents



1 Introduction

The problem of searching for patterns in data is a fundamental one and has a long and successful history. For instance, the extensive astronomical observations of Tycho Brahe in the 16^{th} century allowed Johannes Kepler to discover the empirical laws of planetary motion, which in turn provided a springboard for the development of classical mechanics. Similarly, the discovery of regularities in atomic spectra played a key role in the development and verification of quantum physics in the early twentieth century. The field of pattern recognition is concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories.

Consider the example of recognizing handwritten digits, illustrated in Figure 1.1. Each digit corresponds to a 28×28 pixel image and so can be represented by a vector $\mathbf{x}$ comprising 784 real numbers. The goal is to build a machine that will take such a vector $\mathbf{x}$ as input and that will produce the identity of the digit $0, \ldots, 9$ as the output. This is a nontrivial problem due to the wide variability of handwriting. It could be

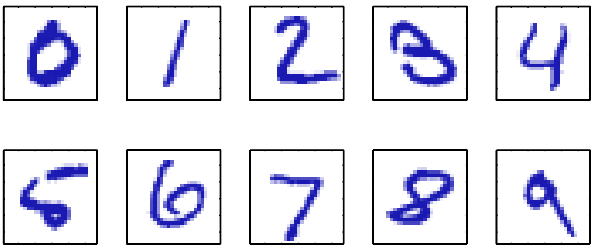


Figure 1.1 Examples of hand-written digits taken from US zip codes.

tackled using handcrafted rules or heuristics for distinguishing the digits based on the shapes of the strokes, but in practice such an approach leads to a proliferation of rules and of exceptions to the rules and so on, and invariably gives poor results.

Far better results can be obtained by adopting a machine learning approach in which a large set of N digits $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ called a *training set* is used to tune the parameters of an adaptive model. The categories of the digits in the training set are known in advance, typically by inspecting them individually and hand-labelling them. We can express the category of a digit using *target vector* $\mathbf{t}$, which represents the identity of the corresponding digit. Suitable techniques for representing categories in terms of vectors will be discussed later. Note that there is one such target vector $\mathbf{t}$ for each digit image $\mathbf{x}$.

The result of running the machine learning algorithm can be expressed as a function $\mathbf{y}(\mathbf{x})$ which takes a new digit image $\mathbf{x}$ as input and that generates an output vector $\mathbf{y}$, encoded in the same way as the target vectors. The precise form of the function $\mathbf{y}(\mathbf{x})$ is determined during the *training* phase, also known as the *learning* phase, on the basis of the training data. Once the model is trained it can then determine the identity of new digit images, which are said to comprise a *test set*. The ability to categorize correctly new examples that differ from those used for training is known as *generalization*. In practical applications, the variability of the input vectors will be such that the training data can comprise only a tiny fraction of all possible input vectors, and so generalization is a central goal in pattern recognition.

For most practical applications, the original input variables are typically *preprocessed* to transform them into some new space of variables where, it is hoped, the pattern recognition problem will be easier to solve. For instance, in the digit recognition problem, the images of the digits are typically translated and scaled so that each digit is contained within a box of a fixed size. This greatly reduces the variability within each digit class, because the location and scale of all the digits are now the same, which makes it much easier for a subsequent pattern recognition algorithm to distinguish between the different classes. This pre-processing stage is sometimes also called *feature extraction*. Note that new test data must be pre-processed using the same steps as the training data.

Pre-processing might also be performed in order to speed up computation. For example, if the goal is real-time face detection in a high-resolution video stream, the computer must handle huge numbers of pixels per second, and presenting these directly to a complex pattern recognition algorithm may be computationally infeasible. Instead, the aim is to find useful features that are fast to compute, and yet that

also preserve useful discriminatory information enabling faces to be distinguished from non-faces. These features are then used as the inputs to the pattern recognition algorithm. For instance, the average value of the image intensity over a rectangular subregion can be evaluated extremely efficiently (Viola and Jones, 2004), and a set of such features can prove very effective in fast face detection. Because the number of such features is smaller than the number of pixels, this kind of pre-processing represents a form of dimensionality reduction. Care must be taken during pre-processing because often information is discarded, and if this information is important to the solution of the problem then the overall accuracy of the system can suffer.

Applications in which the training data comprises examples of the input vectors along with their corresponding target vectors are known as *supervised learning* problems. Cases such as the digit recognition example, in which the aim is to assign each input vector to one of a finite number of discrete categories, are called *classification* problems. If the desired output consists of one or more continuous variables, then the task is called *regression*. An example of a regression problem would be the prediction of the yield in a chemical manufacturing process in which the inputs consist of the concentrations of reactants, the temperature, and the pressure.

In other pattern recognition problems, the training data consists of a set of input vectors $\mathbf{x}$ without any corresponding target values. The goal in such *unsupervised learning* problems may be to discover groups of similar examples within the data, where it is called *clustering*, or to determine the distribution of data within the input space, known as *density estimation*, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of *visualization*.

Finally, the technique of *reinforcement learning* (Sutton and Barto, 1998) is concerned with the problem of finding suitable actions to take in a given situation in order to maximize a reward. Here the learning algorithm is not given examples of optimal outputs, in contrast to supervised learning, but must instead discover them by a process of trial and error. Typically there is a sequence of states and actions in which the learning algorithm is interacting with its environment. In many cases, the current action not only affects the immediate reward but also has an impact on the reward at all subsequent time steps. For example, by using appropriate reinforcement learning techniques a neural network can learn to play the game of backgammon to a high standard (Tesauro, 1994). Here the network must learn to take a board position as input, along with the result of a dice throw, and produce a strong move as the output. This is done by having the network play against a copy of itself for perhaps a million games. A major challenge is that a game of backgammon can involve dozens of moves, and yet it is only at the end of the game that the reward, in the form of victory, is achieved. The reward must then be attributed appropriately to all of the moves that led to it, even though some moves will have been good ones and others less so. This is an example of a *credit assignment* problem. A general feature of reinforcement learning is the trade-off between *exploration*, in which the system tries out new kinds of actions to see how effective they are, and *exploitation*, in which the system makes use of actions that are known to yield a high reward. Too strong a focus on either exploration or exploitation will yield poor results. Reinforcement learning continues to be an active area of machine learning research. However, a

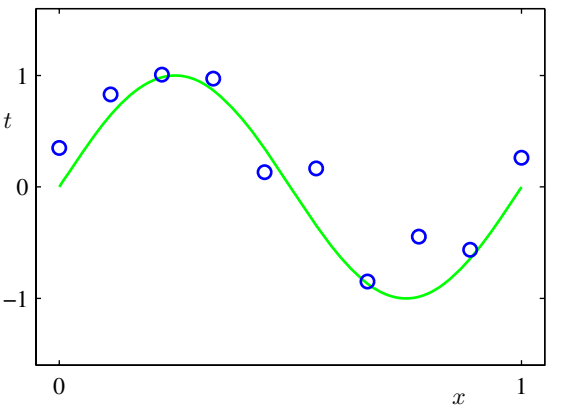


Figure 1.2 Plot of a training data set of $N = 10$ points, shown as blue circles, each comprising an observation of the input variable x along with the corresponding target variable t. The green curve shows the function $\sin(2\pi x)$ used to generate the data. Our goal is to predict the value of t for some new value of x, without knowledge of the green curve.

detailed treatment lies beyond the scope of this book.

Although each of these tasks needs its own tools and techniques, many of the key ideas that underpin them are common to all such problems. One of the main goals of this chapter is to introduce, in a relatively informal way, several of the most important of these concepts and to illustrate them using simple examples. Later in the book we shall see these same ideas re-emerge in the context of more sophisticated models that are applicable to real-world pattern recognition applications. This chapter also provides a self-contained introduction to three important tools that will be used throughout the book, namely probability theory, decision theory, and information theory. Although these might sound like daunting topics, they are in fact straightforward, and a clear understanding of them is essential if machine learning techniques are to be used to best effect in practical applications.

1.1. Example: Polynomial Curve Fitting

We begin by introducing a simple regression problem, which we shall use as a running example throughout this chapter to motivate a number of key concepts. Suppose we observe a real-valued input variable x and we wish to use this observation to predict the value of a real-valued target variable t. For the present purposes, it is instructive to consider an artificial example using synthetically generated data because we then know the precise process that generated the data for comparison against any learned model. The data for this example is generated from the function $\sin(2\pi x)$ with random noise included in the target values, as described in detail in Appendix A.

Now suppose that we are given a training set comprising N observations of x, written $\mathbf{x} \equiv (x_1, \ldots, x_N)^{\mathrm{T}}$, together with corresponding observations of the values of t, denoted $\mathbf{t} \equiv (t_1, \ldots, t_N)^{\mathrm{T}}$. Figure 1.2 shows a plot of a training set comprising $N = 10$ data points. The input data set $\mathbf{x}$ in Figure 1.2 was generated by choosing values of x_n, for $n = 1, \ldots, N$, spaced uniformly in range $[0, 1]$, and the target data set $\mathbf{t}$ was obtained by first computing the corresponding values of the function