TURING 图灵原版计算机科学系列

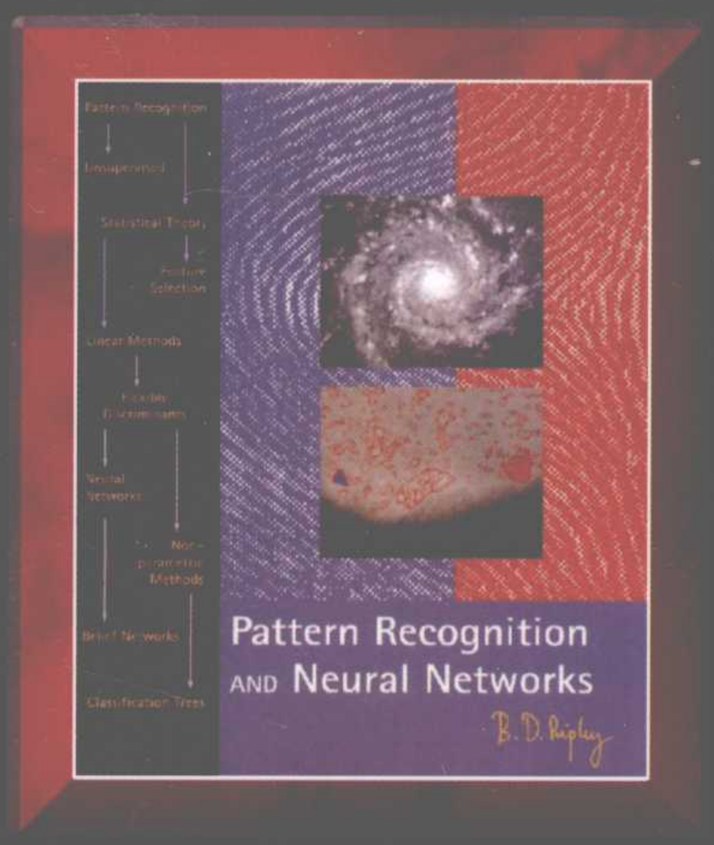
CAMBRIDGE

Pattern Recognition and Neural Networks

模式识别与神经网络

（英文版）

[英] B. D. Ripley 著



人民邮电出版社
POSTS & TELECOM PRESS

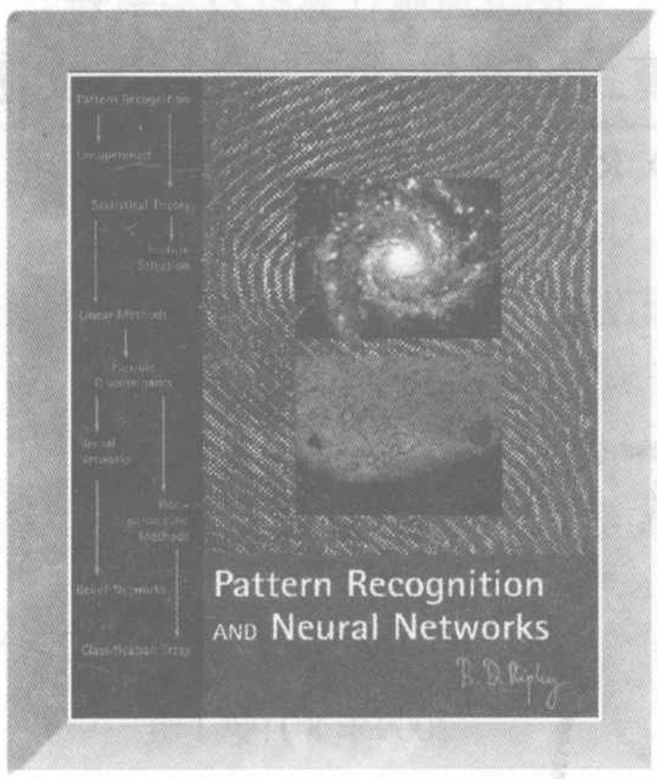
TURING 图灵原版计算机科学系列

Pattern Recognition and Neural Networks

模式识别与神经网络

(英文版)

[英] B. D. Ripley 著



人民邮电出版社
北 京

图书在版编目（CIP）数据

模式识别与神经网络＝Pattern Recognition and Neural Networks：英文/（英）里普利（Ripley, B. D.）著. —北京：人民邮电出版社，2009.8

（图灵原版计算机科学系列）

ISBN 978-7-115-21064-7

Ⅰ. 模… Ⅱ. 里… Ⅲ. ①模式识别－教材－英文 ②神经网络－教材－英文 Ⅳ. 0235 TP183

中国版本图书馆CIP数据核字（2009）第100526号

内 容 提 要

本书是模式识别和神经网络方面的名著，讲述了模式识别所涉及的统计方法、神经网络和机器学习等分支。书的内容从介绍和例子开始，主要涵盖统计决策理论、线性判别分析、弹性判别分析、前馈神经网络、非参数方法、树结构分类、信念网、无监管方法、探寻优良的模式特性等方面的内容。

本书可作为统计与理工科研究生课程的教材，对模式识别和神经网络领域的研究人员也是极有价值的参考书。

图灵原版计算机科学系列

模式识别与神经网络（英文版）

◆ 著 [英] B. D. Ripley

责任编辑 杨海玲

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号

邮编 100061 电子函件 315@ptpress.com.cn

网址 http://www.ptpress.com.cn

北京铭成印刷有限公司印刷

◆ 开本：800×1000 1/16

印张：26

字数：499千字 2009年8月第1版

印数：1-2 000册 2009年8月北京第1次印刷

著作权合同登记号 图字：01-2009-2916号

ISBN 978-7-115-21064-7/TP

定价：69.00元

读者服务热线：（010）51095186 印装质量热线：（010）67129223

反盗版热线：（010）67171154

Preface

Pattern recognition has a long and respectable history within engineering, especially for military applications, but the cost of the hardware both to acquire the data (signals and images) and to compute the answers made it for many years a rather specialist subject. Hardware advances have made the concerns of pattern recognition of much wider applicability. In essence it covers the following problem:

> 'Given some examples of complex signals and the correct decisions for them, make decisions automatically for a stream of future examples.'

There are many examples from everyday life:

> Name the species of a flowering plant.
> Grade bacon rashers from a visual image.
> Classify an X-ray image of a tumour as cancerous or benign.
> Decide to buy or sell a stock option.
> Give or refuse credit to a shopper.

Many of these are currently performed by human experts, but it is increasingly becoming feasible to design automated systems to replace the expert and either perform better (as in credit scoring) or 'clone' the expert (as in aids to medical diagnosis).

Neural networks have arisen from analogies with models of the way that humans might approach pattern recognition tasks, although they have developed a long way from the biological roots. Great claims have been made for these procedures, and although few of these claims have withstood careful scrutiny, neural network methods have had great impact on pattern recognition practice. A theoretical understanding of how they work is still under construction, and is attempted here by viewing neural networks within a statistical framework, together with methods developed in the field of *machine learning*.

One of the aims of this book is to be a reference resource, so almost all the results used are proved (and the remainder are given references to complete proofs). The proofs are often original, short and I believe

show insight into why the methods work. Another unusual feature of this book is that the methods are illustrated on examples, and those examples are either real ones or realistic abstractions. Unlike the proofs, the examples are not optional!

Those hardy perennials, the 'exclusive or' and 'two spirals' problems, do not appear in this book.

The formal pre-requisites to follow this book are rather few, especially if no attempt is made to follow the proofs. A background in linear algebra is needed, including eigendecompositions. (The singular value decomposition is used, but explained.) A knowledge of calculus and its use in finding extrema (such as local minima) is needed, as well as the simplest notions of asymptotics (Taylor series expansions and $O(n)$ notation). Graph theory is used in Chapter 8, but developed from scratch. Only a first course in probability and statistics is assumed, *but* considerable experience in manipulations will be needed to follow the derivations without writing out the intermediate steps. The glossary should help readers with non-technical backgrounds.

A graduate-course knowledge of statistical concepts will be needed to appreciate fully the theoretical developments and proofs. The sections on examples need a much less mathematical background; indeed a good overview of the state of the subject can be obtained by skimming the theoretical sections and concentrating on the examples. The theory and the insights it gives are important in understanding the relative merits of the methods, and it is often very much harder to show that an idea is unsound than to explain the idea.

Several chapters have been used in graduate courses to statisticians and to engineers, computer scientists and physicists. A core of material would be Sections 2.1–2.3, 2.6, 2.7, 3.1, 3.5, 3.6, 4.1, 4.2, 5.1–5.4, 6.1–6.4, 7.1–7.3 and 9.1–9.4, supplemented by material of particular interest to the audience. For example, statisticians should cover 2.4, 2.5, 3.3, 3.4, 5.5, 5.6 and are likely to be interested in Chapter 8, and a fuller view of neural networks in pattern recognition will be gained by adding 3.2, 4.3, 5.5–5.7, 7.6 and 8.4 to the core.

Acknowledgements

This book was originally planned as a joint work with Nils Lid Hjort (University of Oslo), and his influence will be immediately apparent to those who have seen Hjort (1986), a limited circulation report. My own interest in neural networks was kindled by the invitation from Ole Barndorff-Nielsen and David Cox to give a short course at SemStat in 1992, which resulted in Ripley (1993). The book was planned and parts were written during a six-month period of leave at the programme on 'Computer Vision' at the Isaac Newton Institute

for the Mathematical Sciences in Cambridge (England); discussions with the participants helped shape my impressions of leading-edge pattern recognition problems. Discussions with Lionel Tarassenko, Wray Buntine, John Moody and Chris Bishop have also helped to shape my treatment. I was introduced to the machine-learning literature and its distinctive goals by Donald Michie. Several people have read and commented on chapters, notably Phil Dawid, Francis Marriott and Ruth Ripley. I am grateful to Lionel Tarassenko and his co-authors for the cover picture of outlier detection in a mammogram (from Tarassenko *et al.*, 1995).

Parts of this book have been used as source material for graduate lectures and seminar courses at Oxford, and I am grateful to my students and colleagues for feedback; present readers will appreciate the results of their insistence on more details in the mathematics.

Some of the software used is supplied with Venables & Ripley (1994).

The examples were computed within the statistical system S-Plus of MathSoft Inc., using software developed by the author and other contributors to the library of software for that system (notably Trevor Hastie and Rob Tibshirani).

It has been a pleasure to work with CUP staff on the design and production of this volume; especial thanks go to David Tranah, the editor for this project who also contributed many aspects of the design.

B. D. Ripley
Oxford, June 1995

Notation

The notation used generally follows the standard conventions of mathematics and statistics. Random variables are usually denoted by capital letters; if X is a random variable then x denotes its value. Often bold letters denote vectors, so $\mathbf{x} = (x_i)$ is a vector with components $x_i, i = 1, \ldots, m$, with m being deduced from the context.

$\mathscr{D}$	is the 'doubt' report.
E	denotes expectation. A suffix denotes the random variable or distribution over which the averaging takes place.
$I(A)$	is the indicator function of event A, one if A happens, otherwise zero.
$\mathsf{N}_p\{\mu, \Sigma\}$	denotes a normal distribution in p dimensions.
$O(g(n))$	$f(n) = O(g(n))$ means $\lvert f(n)/g(n)\rvert$ is bounded as $n \to \infty$.
$O_p(g(n))$	$X_n = O_p(g(n))$ means given $\epsilon > 0$ there is a constant B such that $\mathsf{Pr}\{\lvert X_n/g(n)\rvert > B\} < \epsilon$ for all n.
$\mathscr{O}$	is the outlier report.
$p(x)$	denotes a probability density function.
$\mathsf{Pr}\{A\}$	denotes the probability of an event A.
$\mathsf{Pr}\{A \mid B\}$	denotes the conditional probability of A given B.
$\mathbb{R}^m$	m-dimensional Euclidean space.
X^T	denotes the transpose of a matrix X.
θ	a parameter or vector of parameters.
$\widehat{\theta}, \widetilde{\theta}$	a parameter estimate.
$[\,]_+$	the positive part, the maximum of the expression and zero.
$\lfloor\,\rfloor$	the integer part (rounding down). The *floor* function.
$\lceil\,\rceil$	the nearest integer (rounding up). The *ceiling* function.

Contents

1

Introduction and Examples

This book is primarily about *pattern recognition*, which covers a wide range of activities from many walks of life. It is something which we humans are particularly good at; we receive data from our senses and are often able, immediately and without conscious effort, to identify the source of the data. For example, many of us can

recognize faces we have not seen for many years, even in disguise,
recognize voices over a poor telephone line,
as babies recognize our mothers by smell,
distinguish the grapes used to make a wine, and sometimes
even recognize the vineyard and year,
identify thousands of species of flowers and
spot an approaching storm.

Science, technology and business has brought to us many similar tasks, including

diagnosing diseases,
detecting abnormal cells in cervical smears,
recognizing dangerous driving conditions,
identifying types of car, aeroplane, . . . ,
identifying suspected criminals by fingerprints and DNA profiles,
reading Zip codes (US postal codes) on envelopes,
reading hand-written symbols (on a penpad computer),
reading maps and circuit diagrams,
classifying galaxies by shape,
picking an optimal move or strategy in a game such as chess,
identifying incoming missiles from radar or sonar signals,
detecting shoals of fish by sonar,
checking packets of frozen peas for 'foreign bodies',
spotting fake 'antique' furniture,

deciding which customers will be good credit risks and
spotting good opportunities on the financial markets.

Humans can (and do) do some of the tasks quite well, but the technological pressure is to build machines which can perform such tasks more accurately or faster or more cheaply than humans, or even to release humans from drudgery. There are also purely technological tasks such as reading bar codes at which humans are poor. *Pattern recognition* is the discipline of building such machines:

> 'It is felt that the decision-making processes of a human being are somewhat related to the recognition of patterns; for example the next move in a chess game is based upon the present position on the board, and buying or selling stocks is decided by a complex pattern of information. The goal of pattern recognition research is to clarify these complicated mechanisms of decision-making processes and to automate these functions using computers. However, because of the complex nature of the problem, most pattern recognition research has been concentrated on more realistic problems, such as the recognition of Latin characters and the classification of waveforms.'
> (Fukunaga, 1990, p. 1)

Since the best humans can perform many of these tasks very well, even better than the best machines, it has been of great interest to understand how we do so, and this is of independent scientific interest. So there has for many years been an interchange of ideas between engineers building pattern recognition systems and psychologists and physiologists studying human and animal brains. Twice this has led to great enthusiasm about machines influenced by ideas from psychology and biology. The first was in the late 1950s with the *perceptron*, the second in the mid 1980s over *neural networks*. Both rapidly left their biological roots, and were studied by mathematical techniques against engineering performance goals as pattern recognizers. This book is not about the impact of the study of neural networks as models of animal brains, but discusses what are more accurately (but rarely) called *artificial* neural networks which have been developed by a community which was originally biologically motivated (although many 'neural network' methods were not). Thus for the purposes of this book, a neural network is a method which arose or was popularized by the neural network community and has been or could be used for pattern recognition. Many of the originators of the current wave of interest were more careful in their terminology; whereas Hopfield (1982) did talk about neural networks, Rumelhart & McClelland (1986) used the term 'parallel distributed processing', and 'connectionist' has also been popular (for example, see Hinton, 1989a).

Marginal notes such as this replace footnotes and offer explanation, sidelines, and opinion.

Many of the ideas had arisen earlier in the pattern recognition context, but without the seductive titles had made little impact.

One characteristic of human pattern recognition is that it is mainly *learnt*. We cannot describe the rules we use to recognize a particular face, and will probably be unable to describe it well enough for anyone else to use the description for recognition. On the other hand, botanists can give the rules they use to identify flowering plants.

Most learning involves a *teacher*. If we try enough different wines from unlabelled bottles, we may well discover that there are common groupings, and that one group has the aroma of gooseberries (if the latter have been experienced). But we will need a teacher to tell us that the common factor is that they were made (in part) from the *sauvignon blanc* grape. The discovery of new groupings is called *unsupervised* pattern recognition. A more common mode of learning both for us and for machines is to be given a collection of labelled examples, known as the *training set*, and from these to distil the essence of the grouping. This is *supervised* pattern recognition and is used to classify future examples into one of the same set of classes (or say it is none of these).

Gooseberries are the fruits of the species *Ribes grossularia.*

We should never underestimate the power of simply remembering some or all of the examples and comparing test examples with our memory.

There is a subject known as *machine learning* which has emerged from the artificial intelligence and computer science communities. It too is concerned with distilling structure from labelled examples, although the labels are usually 'true' and 'false'.

> 'Machine Learning is generally taken to encompass automatic learning procedures based on logical or binary operations, that learn a task from a series of examples.'
>
> 'Machine Learning aims to generate classifying expressions simple enough to be understood easily by humans. They must mimic human reasoning sufficiently well to provide insight into the decision process. Like statistical approaches, background knowledge may be exploited in development, but operation is assumed without human intervention.'
>
> (Michie *et al.*, 1994, p. 2)

This stresses the need for a comprehensible explanation, which is needed in some but not all pattern recognition tasks. We have already noted that we cannot explain our identification of faces, and to recognize Zip codes no explanation is needed, just speed and accuracy.

This quotation mentions statistical approaches, and statistics is the oldest of the disciplines concerned with automatically finding structure in examples. As in the quotation, statistics is often thought of as being less automatic than the other disciplines, but this is largely an artefact of its greater age; its current research frontiers are very much concerned with replacing the human choice of methods by computation. Furthermore, statistics encompasses what the community of statisticians do, of whom your author is one!

1.1 How do neural methods differ?

Assertions are often made that neural networks provide a new approach to computing, involving analog (real-valued) rather than digital signals and massively parallel computation. For example, Haykin (1994, p. 2) offers a definition of a neural network adapted from Aleksander & Morton (1990):

> 'A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:
>
> 1. Knowledge is acquired by the network through a learning process.
> 2. Interneuron connection strengths known as synaptic weights are used to store the knowledge.'

Many neural networks are excluded by this definition, including those of Kohonen. One could ask how a machine comes to have 'natural' properties.

In practice the vast majority of neural network applications are run on single-processor digital computers, although specialist parallel hardware is being developed (if not yet massively parallel). However, all the other methods we consider use real signals and can be parallelized to a considerable extent; it is far from clear that neural network methods will have an advantage as parallel computation becomes common, although they are frequently so slow that they need a speed-up. (Parallelization on real hardware has proved to be non-trivial; see Pitas, 1993 and Przytula & Prasanna, 1993.) We will argue that a large speed-up can be achieved by designing better learning algorithms using experience borrowed from other fields.

The traditional methods of statistics and pattern recognition are either *parametric* based on a family of models with a small number of parameters, or *non-parametric* in which the models used are totally flexible. One of the impacts of neural network methods on pattern recognition has been to emphasize the need in large-scale practical problems for something in between, families of models with large but not unlimited flexibility given by a large number of parameters. The two most widely used neural network architectures, *multi-layer perceptrons* and *radial basis functions* (RBFs), provide two such families (and several others already existed in statistics).

The name 'multi-layer perceptrons' is confusing; they are not multiple layers of perceptrons. We call them feed-forward neural nets.

Another difference in emphasis is on *'on-line'* methods, in which the data are not stored except through the changes the learning algorithm has made. The theory of such algorithms is studied for a very long stream of examples, but the practical distinction is less clear, as this stream is made up either by repeatedly cycling through the training set or by sampling the training examples (with replacement). In contrast, methods which use all the examples together are called *'batch'* methods.

It is often forgotten that there are intermediate positions, such as using small batches chosen from the training set.

1.2 The pattern recognition task

Except in Chapter 9 we will be exclusively concerned with supervised pattern recognition. Thus we are given a set of K pre-determined classes, and assume (in theory) the existence of an oracle that could correctly label each example which might be presented to us. When we receive an example, some measurements are made, known as *features*, and these data are fed into the pattern recognition machine, known as the *classifier*. This is allowed to report

Someone else may have made the measurements for us.

'this example is from class ℓ' or
'this example is from none of these classes' or
'this example is too hard for me'.

The second category are called *outliers* and the third *rejects* or *'doubt'* reports. Both can have great importance in applications. Suppose we have a medical diagnosis aid. We would want it to report any patient who apparently had an unknown disease, and we would also want it to ask the opinion of a senior doctor if there was real doubt. Often rejects are referred to a more expensive second tier of classification, perhaps a human or (as in Zip code recognition) a slower but more powerful method or even (as in analytical chemistry) for more expensive measurements to be made. Many pattern recognition systems always make a firm classification, but this seems to us more often to be bad design than a conscious decision that a firm decision was necessary.

It may help to know which classes are plausible.

The primary assessment of a system will be by its performance; a Zip code recognition system might be required to reject less than 2% of the examples and mis-read less than 0.5% of the remainder. In medical diagnosis we will be more interested in some errors than others, in particular in missing a disease, so the errors will need to be weighted. There may be a cost trade-off between rejection and error rate.

This might be unrealistic for hand-written addresses, and is well beyond current performance levels.

The other aspect of performance stressed in the quote from Michie *et al.* (1994, p. 2) is the power of explanation. Users need to have confidence in the system before it will be adopted. No one really cares if an odd letter is mis-routed, but patients do care if they are mis-diagnosed, and when a civilian airliner is mistaken for an enemy aircraft, questions are raised. So for some tasks 'black boxes' are unacceptable whatever their performance advantage (possibly even if they appear perfect on test). The methods of Chapters 7 and 8 are often found to be more acceptable for such tasks.

Some tasks are slightly different. We (and medics) often think of medical diagnosis as deciding which disease a patient has, but this ignores the possibility of two or more concurrent diseases; what we should really be asking is whether the patient has this disease for each of a range of diseases. This can be thought of as a compound decision, the classes each being a subset of the diseases, but it is normally helpful to make use of special structure within the classes.

Design issues

Although most of this book is about designing the pattern recognition machine, often the most important aspect of design is to choose the right features. If the wrong things are measured (or, more often these days with digital data, if the data are condensed too much) the task may be unachievable. Much of the enhanced success of Zip-code recognition systems has come from better features (for example, Simard *et al.*, 1993) rather than through more complicated classifiers. Sometimes good features can be found by training a classifier on a large number of features and extracting good ones (for example, by the methods of Chapters 9 and 10), but most often problem-specific insights are used.

In a few problem domains very specific rules are known which can be used to design a classifier; as an extreme example compilers can classify C programs as correct or invalid without needing to see any previous programs. Such information is often in the form of a formal *grammar*, and systems based on specifying such grammars are often called *syntactic* pattern recognition systems (Fu, 1982; Gonzalez & Thomason, 1978), but are of very restricted application. Allowing stochastic grammars in which the structure is given but the probabilities are learnt allows a little more flexibility. Chou (1989) gives an example of recognizing typeset mathematical expressions using a stochastic grammar.

In the vast majority of applications no structural assumptions are made, all the structure in the classifier being learnt from data. In the pattern recognition literature this is known as *statistical pattern recognition.* The training set is regarded as a sample from a population of possible examples, and the statistical similarities of each class extracted, or more precisely the significant differences between classes are found. A parametric or non-parametric model is constructed for the distribution of features for examples from each class, and statistical decision theory used to find an optimal classification. This is sometimes known (Dawid, 1976) as the *sampling paradigm.*

Another view, the *diagnostic paradigm*, goes back in the statistical literature at least to Cox (1958), and was developed in medical applications by Jerome Cornfield. This said that we were not interested in what the classes looked like, but only given an example in what the distribution over classes is *for similar examples*. The main method of this approach became known as *logistic discrimination* (Anderson, 1982), but was never widely known even in statistics and (as far as we could ascertain) appears in no pattern recognition text. This is the main approach of the neural network school.

When humans are learning concepts, we are often able to ask questions or to seek the classifications of examples which we synthesize (this being a paradigm of experimental science). Alternatively, we may describe our understanding to an expert, who will then supply a counter-example. Can we allow our machines to do the same? The idea has occurred in machine learning (Angluin, 1987, 1988, 1993), but apparently only for learning logical concepts.

We will sometimes have qualitative knowledge about the task in hand; we might know that only the sign of one of the features was material, or that the probability of a positive outcome was increasing in some continuous feature. Of course we should design the classifier to agree with such information, which Abu-Mostafa (1990, 1993, 1995a, b, c) calls 'hints'. Sometimes this is easy (just use the sign of the feature) but it can be very difficult (as in monotonicity). Generally hints (if true) help to avoid over-fitting to the training set, and this seems to be the real explanation of the gains in exchange-rate performance observed by Abu-Mostafa (1995a).

Method tuning and checking

All methods have some knobs which can be tweaked. Sometimes taking the class of the nearest training-set example is regarded as a fully automatic method, but we need to specify the metric used to find the nearest. (If the answer is 'use Euclidean distance' we still have to specify the units of measurement.)

How should those knobs be set? The most obvious way is to choose them to maximize performance. One thing we should *not* do is to evaluate the performance on a *test set* and choose the best-performing classifier, since we will then have no way to measure the true performance. We can keep back another test set, called a *validation set*, and use the performance on that to set the knobs. However, to obtain a sensitive measure of the performance, the validation set will

Note that this is not the procedure called *cross-validation*, despite the misuse of that term in the neural networks literature.