Algorithmic Learning in a Random World

Vladimir Vovk
University of London
Egham, United Kingdom

Alexander Gammerman
University of London
Egham, United Kingdom

Glenn Shafer
Rutgers Business School
Newark, NJ, USA

Vladimir Vovk
Computer Learning Research Centre,
Dept. of Computer Science
Royal Holloway
University of London
Egham, Surrey TW2 0EX
UK

EMAIL: vovk@cs.rhul.ac.uk

Alexander Gammerman
Computer Learning Research Centre,
Dept. of Computer Science
Royal Holloway
University of London
Egham, Surrey TW2 0EX
UK

EMAIL: alex@cs.rhul.ac.uk

Glenn Shafer
Dept. of Accounting and Information Systems
Rutgers Business School,
Newark and New Brunswick
180 University Avenue
Newark NJ 07102

EMAIL: gshafer@andromeda.rutgers.edu

Library of Congress Cataloging-in-Publication Data

Vovk, Vladimir.
Algorithmic Learning in a Random World / by Vladimir Vovk, Alexander Gammerman, and Glenn Shafer.
p.cm.
Includes Bibliographical references and index.

ISBN 0-387-00152-2 (HC) e-ISBN 0-387-25061-1 (eBK) Printed on acid-free paper.
ISBN-13: 978-0387-00152-4 (HC) e-ISBN-13: 978-038-725061-8 (eBK)

1. Prediction theory. 2. Algorithms. 3. Stochastic processes. I. Gammerman, A. (Alexander) II. Shafer, Glenn, 1946- III. Title.

QA279.2.V68 2005
519.2'87—dc22

2005042556

Printed in the United States of America. (BS/DH)

9 8 7 6 5 4 3 2 1 SPIN 10900581 (HC) / 11399810 (eBK)

springeronline.com

Contents

Preface

This book is about prediction algorithms that learn. The predictions these algorithms make are often imperfect, but they improve over time, and they are *hedged*: they incorporate a valid indication of their own accuracy and reliability. In most of the book we make the standard assumption of randomness: the examples the algorithm sees are drawn from some probability distribution, independently of one another. The main novelty of the book is that our algorithms learn and predict simultaneously, continually improving their performance as they make each new prediction and find out how accurate it is. It might seem surprising that this should be novel, but most existing algorithms for hedged prediction first learn from a training data set and then predict without ever learning again. The few algorithms that do learn and predict simultaneously do not produce hedged predictions. In later chapters we relax the assumption of randomness to the assumption that the data come from an on-line compression model. We have written the book for researchers in and users of the theory of prediction under randomness, but it may also be useful to those in other disciplines who are interested in prediction under uncertainty.

This book has its roots in a series of discussions at Royal Holloway, University of London, in the summer of 1996, involving AG, Vladimir N. Vapnik and VV. Vapnik, who was then based at AT&T Laboratories in New Jersey, was visiting the Department of Computer Science at Royal Holloway for a couple of months as a part-time professor. VV had just joined the department, after a year at the Center for Advanced Study in Behavioral Sciences at Stanford. AG had become the head of department in 1995 and invited both Vapnik and VV to join the department as part of his program (enthusiastically supported by Norman Gowar, the college principal) of creating a machine learning center at Royal Holloway. The discussions were mainly concerned with Vapnik's work on support vector machines, and it was then that it was realized that the number of support vectors used by such a machine could serve as a basis for hedged prediction.

Our subsequent work on this idea involved several doctoral students at Royal Holloway. Ilia Nouretdinov has made several valuable theoretical contributions. Our other students working on this topic included Craig Saunders, Tom Melluish, Kostas Proedrou, Harris Papadopoulos, David Surkov, Leo Gordon, Tony Bellotti, Daniil Ryabko, and David Lindsay. The contribution of Yura Kalnishkan and Misha Vyugin to this book was less direct, mainly through their work on predictive complexity, but still important. Thank you all.

GS joined the project only in the autumn of 2003, although he had earlier helped develop some of its key ideas through his work with VV on game-theoretic probability; see their joint book – *Probability and Finance: It's Only a Game!* – published in 2001.

Steffen Lauritzen introduced both GS and VV to repetitive structures. In VV's case, the occasion was a pleasant symposium organized and hosted by Lauritzen in Aalborg in June 1994. We have also had helpful conversations with Masafumi Akahira, Satoshi Aoki, Peter Bramley, John Campbell, Alexey Chervonenkis, Philip Dawid, José Gonzáles, Thore Graepel, Gregory Gutin, David Hand, Fumiyasu Komaki, Leonid Levin, Xiao Hui Liu, George Loizou, Zhiyuan Luo, Per Martin-Löf, Sally McClean, Boris Mirkin, Fionn Murtagh, John Shawe-Taylor, Sasha Shen', Akimichi Takemura, Kei Takeuchi, Roger Thatcher, Vladimir V'yugin, David Waltz, and Chris Watkins.

Many ideas in this book have their origin in our attempts to understand the mathematical and philosophical legacy of Andrei Nikolaevich Kolmogorov. Kolmogorov's algorithmic notions of complexity and randomness (especially as developed by Martin-Löf and Levin) have been for us the main source of intuition, although they almost disappeared in the final version. VV is grateful to Andrei Nikolaevich for steering him in the direction of compression modeling and for his insistence on its independent value.

We thank the following bodies for generous financial support: EPSRC through grants GR/L35812, GR/M14937, GR/M16856, and GR/R46670; BBSRC through grant 111/BIO14428; MRC through grant S505/65; the Royal Society; the European Commission through grant IST-1999-10226; NSF through grant 5-26830.

University of London (VV, AG, and GS) — *Vladimir Vovk*
Rutgers University (GS) — *Alexander Gammerman*
July 2004 — *Glenn Shafer*

List of principal results

1

Introduction

In this introductory chapter, we sketch the existing work in machine learning on which we build and then outline the contents of the book.

1.1 Machine learning

The rapid development of computer technology during the last several decades has made it possible to solve ever more difficult problems in a wide variety of fields. The development of software has been essential to this progress. The painstaking programming in machine code or assembly language that was once required to solve even simple problems has been replaced by programming in high-level object-oriented languages. We are concerned with the next natural step is this progression – the development of programs that can *learn*, i.e., automatically improve their performance with experience.

The need for programs that can learn was already recognized by Alan Turing (1950), who argued that it may be too ambitious to write from scratch programs for tasks that even humans must learn to perform. Consider, for example, the problem of recognizing hand-written digits. We are not born able to perform this task, but we learn to do it quite robustly. Even when the hand-written digit is represented as a gray-scale matrix, as in Fig. 1.1, we can recognize it easily, and our ability to do so scarcely diminishes when it is slightly rotated or otherwise perturbed. We do not know how to write instructions for a computer that will produce equally robust performance.

The essential difference between a program that implements instructions for a particular task and a program that learns is adaptability. A single learning program may be able to learn a wide variety of tasks: recognizing hand-written digits and faces, diagnosing patients in a hospital, estimating house prices, etc.

Fig. 1.1. A hand-written digit

Recognition, diagnosis, and estimation can all be thought of as special cases of prediction. A person or a computer is given certain information and asked to predict the answer to a question. A broad discussion of learning would go beyond prediction to consider the problems faced by a robot, who needs to act as well as predict. The literature on machine learning, has emphasized prediction, however, and the research reported in this book is in that tradition. We are interested in algorithms that learn to predict well.

Learning under randomness

One learns from experience. This is as true for a computer as it is for a human being. In order for there to be something to learn there must be some stability in the environment; it must be governed by constant, or evolving only slowly, laws. And when we learn to predict well, we may claim to have learned something about that environment.

The traditional way of making the idea of a stable environment precise is to assume that it generates a sequence of examples randomly from some fixed probability distribution, say Q, on a fixed space of possible examples, say $\mathbf{Z}$. These mathematical objects, $\mathbf{Z}$ and Q, describe the environment.

The environment can be very complex; $\mathbf{Z}$ can be large and structured in a complex way. This is illustrated by the USPS data set from which Fig. 1.1 is drawn (see Appendix B). Here an example is any 16×16 image with 31 shades of gray, together with the digit the image represents (an integer between 0 to 9). So there are $31^{16\times 16} \times 10$ (this is approximately 10^{382}) possible examples in the space $\mathbf{Z}$.

In most of this book, we assume that each example consists of an *object* and its *label*. In the USPS dataset, for example, an object is a gray-scale matrix like the one in Fig. 1.1, and its label is the integer between 0 and 9 represented by the gray-scale matrix.

In the problem of recognizing hand-written digits and other typical machine-learning problems, it is the space of objects, the space of possible gray-scale images, that is large. The space of labels is either a small finite set (in what is called *classification problems*) or the set of real numbers (*regression problems*).

When we say that the examples are chosen randomly from Q, we mean that they are independent in the sense of probability theory and all have the distribution Q. They are independent and identically distributed. We call this the *randomness assumption*.

Of course, not all work in machine learning is concerned with learning under randomness. In learning with expert advice, for example, randomness is replaced by a game-theoretic set-up (Vovk 2001a); here a typical result is that the learner can predict almost as well as the best strategy in a pool of

possible strategies. In reinforcement learning, which is concerned with rational decision-making in a dynamic environment (Sutton and Barto 1998), the standard assumption is Markovian. In this book, we will consider extensions of learning under randomness in Chaps. 7–9.

Learning under unconstrained randomness

Sometimes we make the randomness assumption without assuming anything more about the environment: we know the space of examples $\mathbf{Z}$, we know that examples are drawn independently from the same distribution, and this is all we know. We know nothing at the outset about the probability distribution Q from which each example is drawn. In this case, we say we are *learning under unconstrained randomness*. Most of the work in this book, like much other work in machine learning, is concerned with learning under unconstrained randomness.

The strength of modern machine-learning methods often lies in their ability to make hedged predictions under unconstrained randomness in a *high-dimensional* environment, where examples have a very large (or infinite) number of components. We already mentioned the USPS data set, where each example consists of 257 components (16×16 pixels and the label). In machine learning, this number is now considered small, and the problem of learning from the USPS dataset is sometimes regarded as a toy problem.

1.2 A shortcoming of the existing theory

Machine learning has made significant strides in its study of learning under unconstrained randomness. We now have a wide range of algorithms that often work very well in practice: decision trees, neural networks, nearest neighbors algorithms, and naive Bayes methods have been used for decades; newer algorithms include support vector machines and boosting, an algorithm that is used to improve the quality of other algorithms.

From a theoretical point of view, machine learning's most significant contributions to learning under unconstrained randomness are comprised by *statistical learning theory*. This theory, which began with the discovery of VC dimension by Vapnik and Chervonenkis in the late 1960s and was partially rediscovered independently by Valiant (1984), has produced both deep mathematical results and learning algorithms that work very well in practice (see Vapnik 1998 for a recent review).

Given a "training" set of examples, statistical learning theory produces what we call a *prediction rule* – a function mapping the objects into the labels. Formally, the value taken by a prediction rule on a new object is a *simple prediction* – a guess that is not accompanied by any statement concerning how accurate it is likely to be. The theory does guarantee, however, that as the training set becomes bigger and bigger these predictions will become more

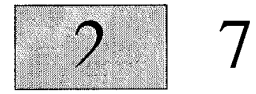

Fig. 1.2. In the problem of digit recognition, we would like to attach lower confidence to the prediction for the image in the middle than to the predictions for the images on the left and the right

and more accurate with greater and greater probability: they are *probably approximately correct*.

How probably and how approximately? This question has not been answered as well as we might like. This is because the theoretical results that might be thought to answer it, the bounds that demonstrate arbitrarily good accuracy with sufficiently large sizes of the training set, are usually too loose to tell us anything interesting for training sets that we actually have. This happens in spite of the empirical fact that the predictions often perform very well in practice. Consider, for example, the problem of recognizing hand-written digits, which we have already discussed. Here we are interested in giving an upper bound on the probability that our learning algorithm fails to choose the right digit; we might like this probability to be less than 0.05, for example, so that we can be 95% confident that the prediction is correct. Unfortunately, typical upper bounds on the probability of error provided by the theory, even for relatively clean data sets such as the USPS data set we have discussed, are greater than 1; bounds less than 1 can usually be achieved only for very straightforward problems or with very large data sets. This is true even for newer results in which the bound on the accuracy depends on the training set (as in, e.g., Littlestone and Warmuth 1986, Floyd and Warmuth 1995; cf. §10.1).

The hold-out estimate of confidence

Fortunately, there are less theoretical and more effective ways of estimating the confidence we should have in predictions output by machine-learning algorithms, including those output by the algorithms proposed by statistical learning theory. One of the most effective is the oldest and most naive: the "hold-out" estimate. In order to compute this estimate, we split the available examples into two parts, a training set and a "test" set. We apply the algorithm to the training set in order to find a prediction rule, and then we apply this prediction rule to the test set. The observed rate of errors on the test set tells us how confident we should be in the prediction rule when we apply it to new examples (for details, see §10.1).

The contribution of this book

When we use a hold-out sample to obtain a meaningful bound on the probability of error, or when we use an error bound from statistical learning theory, we are *hedging* the prediction – we are adding to it a statement about how strongly we believe it. In this book, we develop a different way of producing hedged predictions. Aside from the elegance of our new methods, at least in comparison with the procedure that relies on a hold-out sample, the methods we develop have several important advantages.

As already mentioned in the preface, we do not have the rigid separation between learning and prediction, which is the feature of the traditional approaches that makes hedged prediction feasible. In our basic learning protocol learning and prediction are blended, yet our predictions are hedged.

Second, the hedged predictions produced by our new algorithms are much more confident and accurate. We have, of course, a different notion of a hedged prediction, so the comparison can be only informal; but the difference is so big that there is little doubt that the improvement is real from the practical point of view.

A third advantage of our methods is that the confidence with which the label of a new object is predicted is always tailored not only to the previously seen examples but also to that object.

1.3 The on-line transductive framework

The new methods presented in this book are quite general; they can be tried out, at least, in almost any problem of learning under randomness. The framework in which we introduce and study these methods is somewhat unusual, however. Most previous theoretical work in machine learning has been in an *inductive* and *off-line* framework: one uses a batch of old examples to produce a prediction rule, which is then applied to new examples. We begin instead with a framework that is *transductive*, in the sense advocated by Vapnik (1995, 1998), and *on-line*: one makes predictions sequentially, basing each new prediction on all the previous examples instead of repeatedly using a rule constructed from a fixed batch of examples.

On-line learning

Our framework is *on-line* because we assume that the examples are presented one by one. Each time, we observe the object and predict the label. Then we observe the label and go on to the next example. We start by observing the first object x_1 and predicting its label y_1. Then we observe y_1 and the second object x_2, and predict its label y_2. And so on. At the nth step, we have observed the previous examples

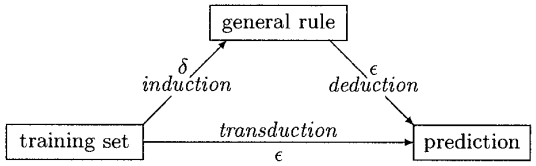

Fig. 1.3. Inductive and transductive prediction

$$(x_1, y_1), \ldots, (x_{n-1}, y_{n-1})$$

and the new object x_n, and our task is to predict y_n. The quality of our predictions should improve as we accumulate more and more old examples. This is the sense in which we are learning.

Transduction

Vapnik's distinction between induction and transduction, as applied to the problem of prediction, is depicted in Fig. 1.3. In *inductive prediction* we first move from examples in hand to some more or less general rule, which we might call a prediction or decision rule, a model, or a theory; this is the *inductive step*. When presented with a new object, we derive a prediction from the general rule; this is the *deductive step*. In *transductive prediction*, we take a shortcut, moving from the old examples directly to the prediction about the new object.

Typical examples of the inductive step are estimating parameters in statistics and finding a "concept" (to use Valiant's 1984 terminology) in statistical learning theory. Examples of transductive prediction are estimation of future observations in statistics (see, e.g., Cox and Hinkley 1974, §7.5) and nearest neighbors algorithms in machine learning.

In the case of simple predictions the distinction between induction and transduction is less than crisp. A method for doing transduction, in our on-line setting, is a method for predicting y_n from $x_1, y_1, \ldots, x_{n-1}, y_{n-1}, x_n$. Such a method gives a prediction for any object that might be presented as x_n, and so it defines, at least implicitly, a rule, which might be extracted from $x_1, y_1, \ldots, x_{n-1}, y_{n-1}$ (induction), stored, and then subsequently applied to x_n to predict y_n (deduction). So any real distinction is really at a practical and computational level: do we extract and store the general rule or not?

For hedged predictions the difference between transduction and induction goes deeper. We will typically want different notions of hedged prediction in the two frameworks. Mathematical results about induction typically involve two parameters, often denoted ϵ (the desired accuracy of the prediction rule) and δ (the probability of achieving the accuracy of ϵ), whereas results about transduction involve only one parameter, which we will denote ϵ (in this book,

the probability of error we are willing to tolerate); see Fig. 1.3. A detailed discussion can be found in Chap. 10, which also contains a historical perspective on the three main approaches to hedged prediction (inductive, Bayesian, and transductive).

On-line/off-line and transduction/induction compromises

When we work on-line, we would want to use a general rule extracted from $x_1, y_1, \ldots, x_{n-1}, y_{n-1}$ only once, to predict y_n from x_n. After observing x_n and then y_n, we have a larger dataset, $x_1, y_1, \ldots, x_n, y_n$, and we can use it to extract a new, possibly improved, general rule before trying to predict y_{n+1} from x_{n+1}. So from a purely conceptual point of view, induction seems silly in the on-line framework; it is more natural to say that we are doing transduction, even in cases where the general rule is easy to extract. As a practical matter, however, the computational cost of a transductive method may be high, and in this case, it may be sensible to compromise with the off-line or inductive approach. After accumulating a certain number of examples, we might extract a general rule and use it for a while, only updating it as frequently as is practical.

The methods we present in this book are most naturally described and are most amenable to mathematical analysis in the on-line framework. So we work out our basic theory in that framework, and this theory can be considered transductive. The theory extends, however, to the transductive/inductive compromise just described, where a general rule is extracted and used for a period of time before it is updated (see §4.1).

The theory also extends to relaxations of the on-line protocol that make it close to the off-line setting, and this is important, because most practical problems have at least some off-line aspects. If we are concerned with recognizing hand-written zip codes, for example, we cannot always rely on a human teacher to tell us the correct interpretation of each hand-written zip code; why not use such an *ideal teacher* directly for prediction? The relaxation of the on-line protocol considered in §4.3 includes "slow teachers", who provide the feedback with a delay, and "lazy teachers", who provide feedback only occasionally. In the example of zip codes recognition, this relaxation allows us to replace constant supervision by using returned letters for teaching or by occasional lessons.

1.4 Conformal prediction

Most of this book is devoted to a particular method that we call "conformal prediction". When we use this method, we predict that a new object will have a label that makes it similar to the old examples in some specified way, and we use the degree to which the specified type of similarity holds within

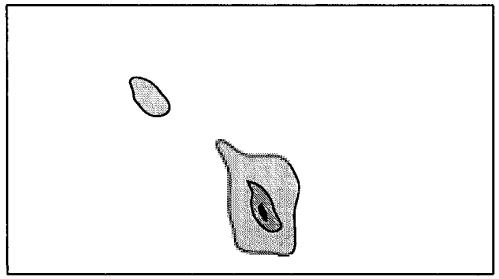

Fig. 1.4. An example of a nested family of prediction sets (casual prediction in black, confident prediction in dark gray, and highly confident prediction in light gray)

the old examples to estimate our confidence in the prediction. Our conformal predictors are, in other words, "confidence predictors".

We need not explain here exactly how conformal prediction works. This is the topic of the next chapter. But we will explain informally what a confidence predictor aims to do and what it means for it to be valid and efficient.

Nested prediction sets

Suppose we want to pinpoint a target that lies somewhere within a rectangular field. This could be an on-line prediction problem; for each example, we predict the coordinates $y_n \in [a_1, a_2] \times [b_1, b_2]$ of the target from a set of measurements x_n.

We can hardly hope to predict the coordinates y_n exactly. But we can hope to have a method that gives a subset Γ_n of $[a_1, a_2] \times [b_1, b_2]$ where we can be confident y_n lies. Intuitively, the size of the *prediction set* Γ_n should depend on how great a probability of error we want to allow, and in order to get a clear picture, we should specify several such probabilities. We might, for example, specify the probabilities 1%, 5%, and 20%, corresponding to *confidence levels* 99%, 95%, and 80%. When the probability of the prediction set failing to include y_n is only 1%, we declare 99% confidence in the set (highly confident prediction). When it is 5%, we declare 95% confidence (confident prediction). When it is 20%, we declare 80% confidence (casual prediction). We might also want a 100% confidence set, but in practice this might be the whole field assumed at the outset to contain the target.

Figure 1.4 shows how such a family of prediction sets might look. The casual prediction pinpoints the target quite well, but we know that this kind of prediction can be wrong 20% of the time. The confident prediction is much bigger. If we want to be highly confident (make a mistake only for each 100th example, on average), we must accept an even lower accuracy; there is even a completely different location that we cannot rule out at this level of confidence.

In principle, a confidence predictor outputs prediction sets for all confidence levels, and these sets are nested, as in Fig. 1.4.

There are two important desiderata for a confidence predictor:

- They should be *valid*, in the sense that in the long run the frequency[1] of error does not exceed ϵ at each chosen confidence level $1-\epsilon$.
- They should be *efficient*, in the sense that the prediction sets they output are as small as possible.

We would also like the predictor to be as conditional as possible – we want it to take full account of how difficult the particular example is.

Validity

Our conformal predictors are always valid. Fig. 1.5 shows the empirical confirmation of the validity for one particular conformal predictor that we study in Chap. 3. The solid, dash-dot and dotted lines show the cumulative number of errors for the confidence levels 99%, 95%, and 80%, respectively. As expected, the number of errors made grows linearly, and the slope is approximately 20% for the confidence level 80%, 5% for the confidence level 95%, and 1% for the confidence level 99%.

As we will see in Chap. 2, a precise discussion of the validity of conformal predictors actually requires that we distinguish two kinds of validity: conservative and exact. In general, a conformal predictor is conservatively valid: the probability it makes an error when it outputs a $1-\epsilon$ set (i.e., a prediction set at a confidence level $1-\epsilon$) is no greater than ϵ, and there is little dependence between errors it makes when predicting successive examples (at successive *trials*, as we will say). This implies, by the law of large numbers, that the long-run frequency of errors at confidence level $1-\epsilon$ is about ϵ or less. In practice, the conservativeness is often not very great, especially when n is large, and so we get empirical results like those in Fig. 1.5, where the long-run frequency of errors is very close to ϵ. From a theoretical point of view, however, we must introduce a small element of deliberate randomization into the prediction process in order to get exact validity, where the probability of a $1-\epsilon$ set being in error is exactly ϵ, errors are made independently at different trials, and the long-run frequency of errors converges to ϵ.

Efficiency

Machine learning has been mainly concerned with two types of problems:

- Classification, where the label space $\mathbf{Y}$ is a small finite set (often binary).
- Regression, where the label space is the real line.

[1] By "frequency" we usually mean "relative frequency".

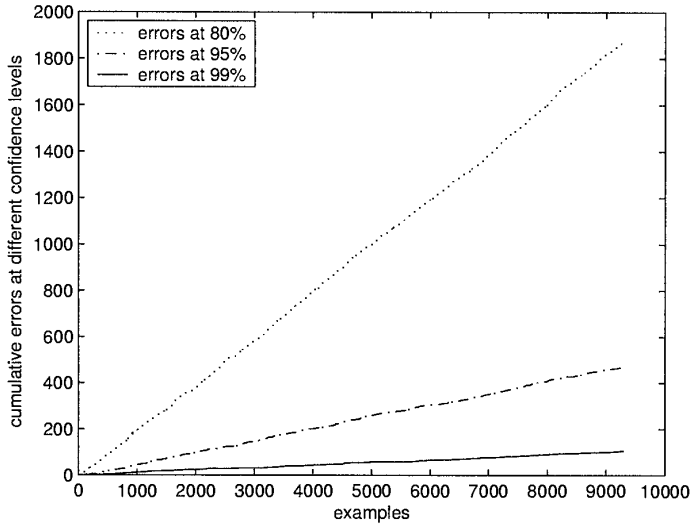

Fig. 1.5. On-line performance of a conformal predictor ("the 1-nearest neighbor conformal predictor", described in Chap. 3) on the USPS data set (9298 handwritten digits, randomly permuted) for the confidence levels 80%, 95%, and 99%. The figures in this book are not too much affected by statistical variation (due to the random choice of the permutation of the data set)

In classification problems, a natural measure of efficiency of confidence predictors is the number of multiple predictions – the number of prediction sets containing two or more labels, at different confidence levels. In regression problems, the prediction set is often an interval of values, and a natural measure of efficiency of such a prediction is the length of the interval. In §2.3 we use the median length of the convex closures of prediction sets as a measure of efficiency of a sequence of predictions.

As we will see in Chap. 2, there are many conformal predictors for any particular on-line prediction problem, whether it is a classification problem or a regression problem. Indeed, we can construct a conformal predictor from any method for scoring the similarity (conformity, as we call it) of a new example to old ones. All these conformal predictors, it turns out, are valid. Which is most efficient – which produces the smallest prediction sets in practice – will depend on details of the environment that we may not know in advance.

In Chap. 3 we show that there exists a "universal" randomized conformal predictor, making asymptotically no more multiple predictions than any valid confidence predictor. This asymptotic result may, however, have limited relevance to practical prediction problems (as discussed in the next section).

Conditionality

The goal of conditionality can be explained with a simple example discussed by David Cox (1958b). Suppose there are two categories of objects, "easy" (easy to predict) and "hard" (hard to predict). We can tell which objects belong to which category, and the two categories occur with equal probability; about 50% of the objects we encounter are easy, and 50% hard. We have a prediction method that applies to all objects, hard and easy, and has error rate 5%. We do not know what the error rate is for hard objects, but perhaps it is 8%, and we get an overall error rate of 5% only because the rate for easy objects is 2%. In this situation, we may feel uncomfortable, when we encounter a hard object, about appealing to the average error rate of 5% and saying that we are 95% confident of our prediction.

Whenever there are features of objects that we know make the prediction easier or harder, we would like to take these features into account – to condition on them. This is done by conformal predictors almost automatically: they are designed for specific applications so that their predictions take fullest possible account of the individual object to be predicted. What is not achieved automatically is the validity separately for hard and easy objects. It is possible, for example, that if a figure such as Fig. 1.5 were constructed for easy objects only, or for hard objects only, the slopes of the cumulative error lines would be different. We would get the correct slope if we average the slope for easy objects and the slope for hard objects, but we would ideally like to have the "conditional validity": validity for both categories of objects. As we show in §4.5, this can be achieved by modifying the definition of conformal predictors. In fact, the conditional validity is handled by a general theory that also applies when we segregate examples not by their difficulty but by their time of arrival, as when we are using an inductive rule that we update only at specified intervals.

Flexibility of conformal predictors

A useful feature of our method is that a conformal predictor can be built on top of almost any machine-learning algorithm. The latter, which we call the *underlying algorithm*, may produce hedged predictions, simple predictions, or simple predictions complemented by ad hoc measures of confidence; our experience is that it is always possible to transform it into a conformal predictor that inherits its predictive performance but is, of course, valid, just like any other conformal predictor. In this book we explain how to build conformal predictors using such methods as nearest neighbors, support vector machines, bootstrap, boosting, neural networks, decision trees, ridge regression, logistic regression, and any Bayesian algorithm (see §§2.3, 3.1, 4.2).

1.5 Probabilistic prediction under unconstrained randomness

There are many ways to do classification and regression under unconstrained randomness and for high-dimensional examples. Conformal predictors, for example, combine good theoretical properties with high accuracy in practical problems. It is true that the environment has to be benign, in some sense, for any learning method to be successful, but there are no obvious insurmountable barriers for classification and regression. The situation changes if we move to the harder problem of *probabilistic prediction*: that of guessing the probability distribution for the new object's label. Features of data that can reasonably be expected in typical machine-learning applications become such barriers.

For simplicity, we will assume in this section that the label is binary, 0 or 1. In this case the probabilistic prediction for the label of the new object boils down to one number, the predicted probability that the label is 1.

The problem of probabilistic prediction is discussed in Chaps. 5, 6, and 9. Probabilistic prediction is impossible in an important sense, but there are also senses in which it is possible. So this book gives more than one answer to the question "Is probabilistic prediction possible?" We start with a "yes" answer.

Universally consistent probabilistic predictor

Stone (1977) showed that a nearest neighbors probabilistic predictor (whose probabilistic prediction is the fraction of objects classified as 1 among the k nearest neighbors of the new object, with a suitably chosen k) is *universally consistent*, in the sense that the difference between the probabilistic prediction and the true conditional probability given the object that the label is 1 converges to zero in probability. The only essential assumption is randomness[2]; there are no restrictive regularity conditions.

Stone's actual result was more general, and it has been further extended in different directions. One of these extension is used in Chap. 3 for constructing a universal randomized conformal predictor.

Probabilistic prediction using a finite data set

The main obstacle in applying Stone's theorem is that the convergence it asserts is not uniform. The situation that we typically encounter in practice is that we are given a set of examples and a new object and we would like to estimate the probability that the label of the new object is 1. It is well understood that in this situation the applicability of Stone's theorem is very

[2]The other assumption made by Stone was that the objects were coming from a Euclidean space; since "Euclidean" is equivalent to "Borel" in the context of existence of a universally consistent probabilistic predictor, this assumption is very weak.

limited (see, e.g., Devroye et al. 1996, §7.1). In Chap. 5 we give a new, more direct, formalization of this observation.

We say that a data set consisting of old examples and one new object is *diverse* if no object in it is repeated (in particular, the new object is different from all old objects). The main result of §5.2 asserts that any nontrivial (not empty and not containing 0 and 1) prediction interval for the conditional probability given the new object that the new label is 1 is inadmissible if the data set is diverse and randomness is the only assumption.

The assumption that the data set is diverse is related to the assumption of a high-dimensional environment. If the objects are, for example, complex images, we will not expect precise repetitions among them.

Venn prediction

The results of Chap. 5 show that it is impossible to estimate the true conditional probabilities under the conditions stated; that chapter also contains a result that it is impossible to find conditional probabilities that are as good (in the sense of the algorithmic theory of randomness) as the true probabilities. If, however, we are prepared to settle for less and only want probabilities that are "well calibrated" (in other words, have a frequentist justification), a modification of conformal predictors which we call Venn predictors will achieve this goal, in a very strong non-asymptotic sense. This is the subject of Chap. 6, which is one of the longest in this book. The main problem that we have to deal with in this chapter is that one cannot guarantee that miscalibration will not happen: everything can happen (perhaps with a small probability) for finite sequences and typical probability distributions. But in the case of Venn predictors, any evidence against calibration translates into evidence, at least as strong, against the assumption of randomness; therefore, we expect Venn predictors to be well calibrated as long as we accept the hypothesis of randomness. A significant part of the chapter is devoted to the ways of testing calibration and randomness.

1.6 Beyond randomness

In this book we also consider testing the assumption of randomness and alternatives to this assumption. The most radical alternative is introduced in Chaps. 8 and 9 under the name of "on-line compression modeling".

Testing randomness

This is the topic of Chap. 7. We start it by adapting the mathematical apparatus developed in the previous chapters to testing the assumption of randomness. The usual statistical approach to testing (sometimes called

the "Neyman–Pearson–Wald" theory) is essentially off-line: in the original Neyman–Pearson approach (see, e.g., Lehmann 1986), the sample size is chosen *a priori*, and in Wald's (1947) sequential analysis, the sample size is data-dependent but still at some point a categorical decision on whether the null hypothesis is rejected or not is taken (with probability one). The approach of §7.1 is on-line: we constantly update the strength of evidence against the null hypothesis of randomness. Finding evidence against the null hypothesis involves gambling against it, and the strength of evidence equals the gambler's current capital. For further details and the history of this approach to testing, see Shafer and Vovk 2001. The main mathematical finding of §7.1 is that there exists a wide family of "exchangeability martingales", which can be successfully applied to detecting lack of randomness.

Low-dimensional dynamic models

The ability to test the assumption of randomness immediately provides opportunities for extending the range of stochastic environments to which one can apply the idea of conformal prediction. In §7.2 we consider the simple case where we are given a parametric family of transformations one of which is believed to transform the observed data sequence into a random sequence. If the parameter is a vector in a low-dimensional linear space, we can often hope to be able to detect lack of randomness of the transformed data sequence for most values of the parameter as the number of observed examples grows. When the range of possible values of the parameter becomes very narrow, conformal prediction can be used.

Islands of randomness

When we are willing to make the assumption of randomness, or some version of this assumption as described in the previous subsection, about a data sequence, it usually means that this data sequence was obtained from a much bigger sequence by careful filtering. When observing the real world around us, we cannot hope that a simple model such as randomness will explain much, but the situation changes if we, e.g., discard all observations except the results of fair coin tosses.

In §7.3, we briefly discuss the case where randomness can appear as a property of only relatively small subsequences of the full data sequence. Such a "big picture" is of great interest to philosophers (see, e.g., Venn 1866). Once we know that some subsequence is random (this knowledge can be based on an initial guess and then using as severe tests as we can think of to try and falsify this guess; §7.1 provides the means for the second stage), we can apply the theory developed under the assumption of randomness to make predictions.

On-line compression models

As we will see in Chap. 8, the idea of conformal prediction generalizes from learning under randomness, where examples are independent and identically distributed, to "on-line compression models". In an on-line compression model, it is assumed that the data can be summarized in way that can be updated as new examples come in, and the only probabilities given are backward probabilities – probabilities for how the updated summary might have been obtained.

On-line compression models derive from the work of Andrei Kolmogorov. They open a new direction for broadening the applicability of machine-learning methods, giving a new meaning to the familiar idea that learning can be understood as information compression.

In Chap. 8 we consider in detail three important on-line compression models (Gaussian, Markov, exchangeability) and their variants. In Chap. 9 we extend the idea of Venn prediction to on-line compression modeling and apply it to a new model, which we call the "hypergraphical model".

1.7 Bibliographical remarks

Each chapter of this book ends with a section entitled "Bibliographical remarks", or similarly. These sections are set in a small font and may use mathematical notions and results not introduced elsewhere in the book.

Turing suggested the idea of machine learning in his paper published in *Mind* as an approach to solving his famous "imitation game" (Turing 1950, §1).

A recent empirical study of various bounds on prediction accuracy is reported in Langford 2004. It found the hold-out estimate to be a top performer.

Mitchell (1997, §8.6) discusses advantages and disadvantages of inductive and transductive approaches to making simple predictions. The near-synonyms for "transductive learning" used in that book are "lazy learning" and "instance-based learning".

2

Conformal prediction

In this chapter we formally introduce conformal predictors. After giving the necessary definitions, we will prove that when a conformal predictor is used in the on-line mode, its output is valid, not only in the asymptotic sense that the sets it predicts for any fixed confidence level $1 - \epsilon$ will be wrong with frequency at most ϵ (approaching ϵ in the case of smoothed conformal predictors) in the long run, but also in a much more precise sense: the error probability of a smoothed conformal predictor is ϵ at every trial and errors happen independently at different trials. In §2.4 we will see that conformal prediction is indispensable for achieving this kind of validity. The basic procedure of conformal prediction might look computationally inefficient when the label set is large, but in §2.3 we show that in the case of, e.g., least squares regression (where the label space $\mathbb{R}$ is uncountable) there are ways of making conformal predictors much more efficient.

2.1 Confidence predictors

The conformal predictors we define in this chapter are confidence predictors – they make a range of successively more specific predictions with successively less confidence. In this section we define precisely what we mean by a confidence predictor and its validity.

Assumptions

We assume that Reality outputs successive pairs

$$(x_1, y_1), (x_2, y_2), \dots , \tag{2.1}$$

called *examples*. Each example (x_i, y_i) consists of an *object* x_i and its *label* y_i. The objects are elements of a measurable space $\mathbf{X}$ called the *object space* and the labels are elements of a measurable space $\mathbf{Y}$ called the *label space*.

We assume that $\mathbf{X}$ is non-empty and that $\mathbf{Y}$ contains at least two essentially different elements[1]. When we need a more compact notation, we write z_i for (x_i, y_i). We set

$$\mathbf{Z} := \mathbf{X} \times \mathbf{Y}$$

and call $\mathbf{Z}$ the *example space*. Thus the infinite data sequence (2.1) is an element of the measurable space $\mathbf{Z}^\infty$.

When we say that the objects are *absent*, we mean that $|\mathbf{X}| = 1$. In this case x_i do not carry any information and do not need to be mentioned; we will then identify $\mathbf{Y}$ and $\mathbf{Z}$.

Our standard assumption is that Reality chooses the examples independently from some probability distribution Q on $\mathbf{Z}$ – i.e., that the infinite sequence $z_1, z_2, \ldots$ is drawn from the *power probability distribution* Q^∞ in $\mathbf{Z}^\infty$. Most of the results of this book hold under this *randomness assumption*, but usually we need only the slightly weaker assumption that the infinite data sequence (2.1) is drawn from a distribution P on $\mathbf{Z}^\infty$ that is *exchangeable*. The statement that P is exchangeable means that for every positive integer n, every permutation π of $\{1, \ldots, n\}$, and every measurable set $E \subseteq \mathbf{Z}^n$,

$$P\{(z_1, z_2, \ldots) \in \mathbf{Z}^\infty : (z_1, \ldots, z_n) \in E\} = P\left\{(z_1, z_2, \ldots) \in \mathbf{Z}^\infty : (z_{\pi(1)}, \ldots, z_{\pi(n)}) \in E\right\} .$$

Every power distribution is exchangeable, and under a natural regularity condition ($\mathbf{Z}$ is a Borel space), any exchangeable distribution on $\mathbf{Z}^\infty$ is a mixture of power distributions; for details, see §A.5. In our mathematical results, we usually use the randomness assumption or the exchangeability assumption depending on which one leads to a stronger statement.

Simple predictors and confidence predictors

We assume that at the nth trial Reality first announces the object x_n and only later announces the label y_n. If we simply want to predict y_n, then we need a function

$$D : \mathbf{Z}^* \times \mathbf{X} \to \mathbf{Y} . \tag{2.2}$$

We call such a function a *simple predictor*, always assuming it is measurable. For any sequence of old examples, say $x_1, y_1, \ldots, x_{n-1}, y_{n-1} \in \mathbf{Z}^*$, and any new object, say $x_n \in \mathbf{X}$, it gives $D(x_1, y_1, \ldots, x_{n-1}, y_{n-1}, x_n) \in \mathbf{Y}$ as its prediction for the new label y_n.

As we explained in §1.4, however, we have a more complicated notion of prediction. Instead of merely choosing a single element of $\mathbf{Y}$ as our prediction

[1]Formally, the σ-algebra on $\mathbf{Y}$ is assumed to be different from $\{\emptyset, \mathbf{Y}\}$. It is convenient to assume that for each pair of distinct elements of $\mathbf{Y}$ there is a measurable set containing only one of them; we will do this without loss of generality, and then our assumption about $\mathbf{Y}$ is that $|\mathbf{Y}| > 1$.

for y_n, we want to give a range of more or less precise predictions, each labeled with a degree of confidence. We want to give subsets of $\mathbf{Y}$ large enough that we can be confident that y_n will fall in them, while also giving smaller subsets in which we are less confident. An algorithm that predicts in this sense requires an additional input $\epsilon \in (0, 1)$, which we call the *significance level*; the complementary value $1 - \epsilon$ is called the *confidence level*. Given all these inputs, say

$$x_1, y_1, \ldots, x_{n-1}, y_{n-1}, x_n, \epsilon ,$$

an algorithm of the type that interests us, say Γ, outputs a subset

$$\Gamma^\epsilon(x_1, y_1, \ldots, x_{n-1}, y_{n-1}, x_n)$$

of $\mathbf{Y}$. (We position ϵ as a superscript instead of placing it with the other arguments.) We require this subset to shrink as ϵ is increased: Γ must satisfy

$$\Gamma^{\epsilon_1}(x_1, y_1, \ldots, x_{n-1}, y_{n-1}, x_n) \subseteq \Gamma^{\epsilon_2}(x_1, y_1, \ldots, x_{n-1}, y_{n-1}, x_n) \tag{2.3}$$

whenever $\epsilon_1 \geq \epsilon_2$. Intuitively, once we observe the *incomplete data sequence*

$$x_1, y_1, \ldots, x_{n-1}, y_{n-1}, x_n \tag{2.4}$$

and chose the significance level ϵ, Γ predicts that

$$y_n \in \Gamma^\epsilon(x_1, y_1, \ldots, x_{n-1}, y_{n-1}, x_n) , \tag{2.5}$$

and the smaller ϵ is the more confident the prediction is. According to condition (2.3), we are more confident in less specific predictions.

Formally, we call a measurable function

$$\Gamma : \mathbf{Z}^* \times \mathbf{X} \times (0, 1) \to 2^{\mathbf{Y}} \tag{2.6}$$

($2^{\mathbf{Y}}$ is the set of all subsets of $\mathbf{Y}$) that satisfies (2.3), for all significance levels $\epsilon_1 \geq \epsilon_2$, all positive integers n, and all incomplete data sequences (2.4), a *confidence predictor* (or *deterministic confidence predictor*). The requirement that Γ be measurable means that for each n the set of sequences $\epsilon, x_1, y_1, \ldots, x_n, y_n$ satisfying (2.5) is a measurable subset of $(0, 1) \times (\mathbf{X} \times \mathbf{Y})^n$.

Validity

Let us begin with an intuitive explanation of the notions of exact and conservative validity. For each significance level ϵ, we want to have confidence $1 - \epsilon$ in our prediction (2.5) about y_n. This means that the probability of the prediction being in error – the probability of the event (2.5) not happening – should be ϵ. Moreover, since we are making a whole sequence of predictions, first for y_1, then for y_2, and so on, we would like these error events to be independent. If these conditions are met no matter what exchangeable probability

distribution P governs the sequence of examples, then we say that the confidence predictor Γ is "exactly valid", or, more briefly, "exact". This means that making errors with Γ is like getting heads when making independent tosses of a biased coin whose probability of heads is always ϵ. If the probabilities for errors are allowed to be even less than this, then we say that the confidence predictor Γ is "conservatively valid" or, more briefly, "conservative".

When we make independent tosses of a biased coin whose probability of heads is always ϵ, the frequency of heads will converge to ϵ with probability one – this is the law of large numbers. So the frequency of errors at significance level ϵ for an exactly valid confidence predictor converges to ϵ with probability one. As we will see, confidence predictors sometimes have this asymptotic property even when they are not exactly valid. So we give the property a name of its own; we call a confidence predictor that has it "asymptotically exact". Similarly, we call a confidence predictor for which the frequency of errors is asymptotically no more than ϵ (for which the upper limit of the frequency of errors is at most ϵ) with probability one "asymptotically conservative".

In order to restate these definitions in a way sufficiently precise to exclude possible misunderstandings, we now introduce a formal notation for the errors Γ makes when it processes the data sequence

$$\omega = (x_1, y_1, x_2, y_2, \dots) \tag{2.7}$$

at significance level ϵ. Whether Γ makes an error on the nth trial can be represented by a number that is one in case of an error and zero in case of no error:

$$\operatorname{err}_n^\epsilon(\Gamma, \omega) := \begin{cases} 1 & \text{if } y_n \notin \Gamma^\epsilon(x_1, y_1, \dots, x_{n-1}, y_{n-1}, x_n) \\ 0 & \text{otherwise} , \end{cases} \tag{2.8}$$

and the number of errors during the first n trials is

$$\operatorname{Err}_n^\epsilon(\Gamma, \omega) := \sum_{i=1}^{n} \operatorname{err}_i^\epsilon(\Gamma, \omega) . \tag{2.9}$$

It is enlightening to think about the error counts $\operatorname{err}_n^\epsilon$ and $\operatorname{Err}_n^\epsilon$ in the context of the protocol followed as the examples are presented and the predictions are made:

$\operatorname{Err}_0^\epsilon := 0$ for all $\epsilon \in (0,1)$;
FOR $n = 1, 2, \dots$:
  Reality outputs $x_n \in \mathbf{X}$;
  Predictor outputs $\Gamma_n^\epsilon \subseteq \mathbf{Y}$ for all $\epsilon \in (0,1)$;
  Reality outputs $y_n \in \mathbf{Y}$;
  $\operatorname{err}_n^\epsilon := \begin{cases} 1 & \text{if } y_n \notin \Gamma_n^\epsilon \\ 0 & \text{otherwise} \end{cases}$ for all $\epsilon \in (0,1)$;
  $\operatorname{Err}_n^\epsilon := \operatorname{Err}_{n-1}^\epsilon + \operatorname{err}_n^\epsilon$ for all $\epsilon \in (0,1)$
END FOR.

This is a game protocol, and we can consider strategies for both players, Reality and Predictor. We have assumed that Reality's strategy is randomized; her moves are generated from an exchangeable probability distribution P on $\mathbf{Z}^\infty$. A confidence predictor Γ is, by definition, a measurable strategy for Predictor.

If ω is drawn from an exchangeable probability distribution P, the number $\operatorname{err}_n^\epsilon(\Gamma, \omega)$ is the realized value of a random variable, which we may designate $\operatorname{err}_n^\epsilon(\Gamma, P)$. Formally, the confidence predictor Γ is *exactly valid* if for each ϵ,

$$\operatorname{err}_1^\epsilon(\Gamma, P), \operatorname{err}_2^\epsilon(\Gamma, P), \dots \tag{2.10}$$

is a sequence of independent Bernoulli random variables with parameter ϵ – i.e., if it is a sequence of independent random variables each of which has probability ϵ of being one and probability $1 - \epsilon$ of being zero – no matter what exchangeable distribution P we draw ω from. Unfortunately, the notion of exact validity is vacuous for confidence predictors.

Theorem 2.1. *No confidence predictor is exactly valid.*

The notion of conservative validity is more complex; now we only require that $\operatorname{err}_n^\epsilon(\Gamma, P)$ be *dominated in distribution* by a sequence of independent Bernoulli random variables with parameter ϵ. Formally, the confidence predictor Γ is *conservatively valid* if for any exchangeable probability distribution P on $\mathbf{Z}^\infty$ there exists a probability space with two families

$$(\xi_n^{(\epsilon)} : \epsilon \in (0,1), n = 1, 2, \dots), \quad (\eta_n^{(\epsilon)} : \epsilon \in (0,1), n = 1, 2, \dots)$$

of $\{0,1\}$-valued random variables such that:

- for a fixed ϵ, $\xi_1^{(\epsilon)}, \xi_2^{(\epsilon)}, \dots$ is a sequence of independent Bernoulli random variables with parameter ϵ;
- for all n and ϵ, $\eta_n^{(\epsilon)} \le \xi_n^{(\epsilon)}$;
- the joint distribution of $\operatorname{err}_n^\epsilon(\Gamma, P)$, $\epsilon \in (0,1)$, $n = 1, 2, \dots$, coincides with the joint distribution of $\eta_n^{(\epsilon)}$, $\epsilon \in (0,1)$, $n = 1, 2, \dots$.

(It might have been natural to also require that $\xi_n^{(\epsilon_1)} \ge \xi_n^{(\epsilon_2)}$ whenever $\epsilon_1 \ge \epsilon_2$, but it is easy to check that the inclusion of this condition leads to an equivalent definition.)

To conclude, we define asymptotic validity. The confidence predictor Γ is *asymptotically exact* if for any exchangeable probability distribution P on $\mathbf{Z}^\infty$ generating examples $z_1, z_2, \dots$ and any significance level ϵ,

$$\lim_{n\to\infty} \frac{\operatorname{Err}_n^\epsilon(\Gamma, P)}{n} = \epsilon \tag{2.11}$$

with probability one. It is *asymptotically conservative* if for any exchangeable probability distribution P on $\mathbf{Z}^\infty$ and any significance level ϵ,

$$\limsup_{n\to\infty} \frac{\mathrm{Err}^\epsilon_n(\Gamma, P)}{n} \le \epsilon \tag{2.12}$$

with probability one.

Proposition 2.2. *An exact confidence predictor is asymptotically exact. A conservative confidence predictor is asymptotically conservative.*

This proposition is an immediate consequence of the law of large numbers.

Randomized confidence predictors

We will also be interested in randomized confidence predictors, which depend, additionally, on elements of an auxiliary probability space. The main advantage of randomization in this context is that, as we will see, there are many randomized confidence predictors that are exactly valid. Formally, we define a *randomized confidence predictor* to be a measurable function

$$\Gamma : (\mathbf{X} \times [0,1] \times \mathbf{Y})^* \times (\mathbf{X} \times [0,1]) \times (0,1) \to 2^{\mathbf{Y}} \tag{2.13}$$

which, for all significance levels $\epsilon_1 \ge \epsilon_2$, all positive integer n, and all incomplete data sequences

$$x_1, \tau_1, y_1, \dots, x_{n-1}, \tau_{n-1}, y_{n-1}, x_n, \tau_n \ ,$$

where $x_i \in \mathbf{X}$, $\tau_i \in [0,1]$ and $y_i \in \mathbf{Y}$ for all i, satisfies

$$\Gamma^{\epsilon_1}(x_1, \tau_1, y_1, \dots, x_{n-1}, \tau_{n-1}, y_{n-1}, x_n, \tau_n) \subseteq \Gamma^{\epsilon_2}(x_1, \tau_1, y_1, \dots, x_{n-1}, \tau_{n-1}, y_{n-1}, x_n, \tau_n) \ .$$

We will always assume that $\tau_1, \tau_2, \dots$ are random variables that are independent (between themselves and of anything else) and distributed uniformly in $[0,1]$; this is what one expects to get from a random number generator.

We define $\mathrm{err}^\epsilon_n(\Gamma, \omega)$ and $\mathrm{Err}^\epsilon_n(\Gamma, \omega)$ as before, only now they also depend on the τ's. In other words, $\mathrm{err}^\epsilon_n(\Gamma, \omega)$ is defined by (2.8) with x_i now being *extended objects* $x_i \in \mathbf{X} \times [0,1]$; $\mathrm{Err}^\epsilon_n(\Gamma, \omega)$ is defined, as before, by (2.9).

In general, many definitions for randomized confidence predictors are special cases of the corresponding definitions for deterministic confidence predictors: the latter should just be applied to the *extended object space* $\mathbf{X} \times [0,1]$ (whose elements consist of both the object and the random number to be used at a given trial).

If Γ is a randomized confidence predictor, P is an exchangeable distribution on $\mathbf{Z}^\infty$, and $n \in \mathbb{N}$,

$$\mathrm{err}^\epsilon_n(\Gamma, (x_1, \tau_1, y_1, x_2, \tau_2, y_2, \dots))$$

and

$$\mathrm{Err}^\epsilon_n(\Gamma, (x_1, \tau_1, y_1, x_2, \tau_2, y_2, \dots))$$

where $(x_1, y_1), (x_2, y_2), \dots$ are drawn from P and $\tau_1, \tau_2, \dots$ are drawn independently from $\mathbf{U}^\infty$, the uniform distribution on $[0,1]^\infty$, are random variables, which will be denoted $\mathrm{err}^\epsilon_n(\Gamma, P)$ and $\mathrm{Err}^\epsilon_n(\Gamma, P)$, respectively. We say that Γ is *exactly valid* if for each $\epsilon \in (0,1)$, (2.10) is a sequence of independent Bernoulli random variables with parameter ϵ.

We are not really interested in the notion of conservative validity for randomized confidence predictors.

2.2 Conformal predictors

We start by defining the concept of a nonconformity measure. Intuitively, this is a way of measuring how different a new example is from old examples. There are many different nonconformity measures, and each one defines a conformal predictor and a smoothed conformal predictor.

Bags

The order in which old examples appear should not make any difference, and in order to formalize this point, we need the concept of a bag (also called a multiset). A *bag* of size $n \in \mathbb{N}$ is a collection of n elements some of which may be identical; a bag resembles a set in that the order of its elements is irrelevant, but it differs from a set in that repetition is allowed. To identify a bag, we must say what elements it contains and how many times each of these elements is repeated. We write $\wr z_1, \dots, z_n \int$ for the bag consisting of elements $z_1, \dots, z_n$, some of which may be identical with each other.

Although the elements of a bag are not ordered, there is an ordering in our notation, and we will make use of it. The bag $\wr z_1, \dots, z_{20} \int$ is the bag we get from $z_1, \dots, z_{20}$ when we ignore their order, but because we have identified the bag using the elements in a certain order, we can manipulate it using our knowledge of this order. We can, for example, talk about the bag we get when we remove z_6 (while leaving any other z_i that might be equal to z_6); this is a bag of size 19 – the bag $\wr z_1, \dots, z_5, z_7, \dots, z_{20} \int$.

We write $\mathbf{Z}^{(n)}$ for the set of all bags of size n of elements of a measurable space $\mathbf{Z}$. The set $\mathbf{Z}^{(n)}$ is itself a measurable space. It can be defined formally as the power space $\mathbf{Z}^n$ with a nonstandard σ-algebra, consisting of measurable subsets of $\mathbf{Z}^n$ that contain all permutations of their elements. We write $\mathbf{Z}^{(*)}$ for the set of all bags of elements of $\mathbf{Z}$ (the union of all the $\mathbf{Z}^{(n)}$).

Nonconformity and conformity

As we have already said, a nonconformity measure is a way of scoring how different a new example is from a bag of old examples. Formally, a *nonconformity measure* is a measurable mapping

$$A : \mathbf{Z}^{(*)} \times \mathbf{Z} \to \overline{\mathbb{R}} ;$$

to each possible bag of old examples and each possible new example, A assigns a numerical score indicating how different the new example is from the old ones.

It is easy to invent nonconformity measures, especially when we already have methods for prediction at hand:

- If the examples are merely numbers ($\mathbf{Z} = \mathbb{R}$) and it is natural to take the average of the old examples as the simple predictor of the new example, then we might define the nonconformity of a new example as the absolute value of its difference from the average of the old examples. Alternatively, we could use instead the absolute value of its difference from the median of the old examples.
- In a regression problem where examples are pairs of numbers, say $z_i = (x_i, y_i)$, we might define the nonconformity of a new example (x, y) as the absolute value of the difference between y and the predicted value $\hat{y}$ calculated from x and the old examples.

We will discuss this way of detecting nonconformity further at the end of this section. But whether a particular function on $\mathbf{Z}^{(*)} \times \mathbf{Z}$ is an appropriate way of measuring nonconformity will always be open to discussion, and we do not need to enter into this discussion at this point. In our general theory, we will call *any* measurable function on $\mathbf{Z}^{(*)} \times \mathbf{Z}$ taking values in the extended real line a nonconformity measure.

Given a nonconformity measure A, a sequence $z_1, \ldots, z_l$ of examples, and an example z, we can score how different z is from the bag $\{z_1, \ldots, z_l\}$; namely, $A(\{z_1, \ldots, z_l\}, z)$ is called the *nonconformity score* for z.

Of course, instead of looking at functions that we feel measure nonconformity, we could look at functions that we feel measure conformity. We call such a function, say B, a conformity measure, and we can use it to define *conformity scores* $B(\{z_1, \ldots, z_l\}, z)$. Formally, a *conformity measure* is a measurable function of the type $\mathbf{Z}^{(*)} \times \mathbf{Z} \to \overline{\mathbb{R}}$ (so there is no difference between conformity measures and nonconformity measures as mathematical objects). If we begin in this way, then nonconformity appears as a derivative idea. Given a conformity measure B we can define a nonconformity measure A using any strictly decreasing transformation, say $A := -B$ or perhaps (if B takes only positive values) $A := 1/B$. Given our goal, prediction, beginning with conformity might seem the more natural approach. As we will explain shortly, our strategy for prediction is to predict that a new label will be among the labels that best make a new example conform with old examples, and it is more natural to emphasize the labels that we include in the prediction (the most conforming ones) rather than the labels that we exclude (the most nonconforming ones). But in practice, it is often more natural to begin with nonconformity measures. For example, when we compare a new example with an average of old examples, we will usually first define a distance between the two rather than devise a way to measure their closeness. For this reason, we emphasize nonconformity rather than conformity. Doing so is consistent with tradition in mathematical statistics, where test statistics are usually defined so as to measure discrepancy rather than agreement.

We sometimes find it convenient to consider separately how a nonconformity measure deals with bags of different sizes: if A is a nonconformity measure, for each $n = 1, 2, \ldots$ we define the function

$$A_n : \mathbf{Z}^{(n-1)} \times \mathbf{Z} \to \overline{\mathbb{R}} \tag{2.14}$$

as the restriction of A to $\mathbf{Z}^{(n-1)} \times \mathbf{Z}$. The sequence $(A_n : n \in \mathbb{N})$, which we abbreviate to (A_n) when there is no danger of confusion, will also be called a nonconformity measure. Analogous conventions will be used for conformity measures.

p-values

Given a nonconformity measure (A_n) and a bag $\{z_1, \ldots, z_n\}$, we can compute the nonconformity score

$$\alpha_i := A_n\left(\{z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n\}, z_i\right) \tag{2.15}$$

for each example z_i in the bag. Because a nonconformity measure (A_n) may be scaled however we like, the numerical value of α_i does not, by itself, tell us how unusual (A_n) finds z_i to be. For that, we need a comparison of α_i to the other α_j. A convenient way of making this comparison is to compute the fraction

$$\frac{|\{j = 1, \ldots, n : \alpha_j \geq \alpha_i\}|}{n} . \tag{2.16}$$

This fraction, which lies between $1/n$ and 1, is called the *p-value* for z_i. It is the fraction of the examples in the bag as nonconforming as z_i. If it is small (close to its lower bound $1/n$ for a large n), then z_i is very nonconforming (an outlier). If it is large (close to its upper bound 1), then z_i is very conforming.

If we begin with a conformity measure (B_n) rather than a nonconformity measure, then we can define the p-value for z_i by

$$\frac{|\{j = 1, \ldots, n : \beta_j \leq \beta_i\}|}{n} ,$$

where the β_j are the conformity scores. This gives the same result as we would obtain from (2.16) using a nonconformity measure (A_n) obtained from (B_n) by means of a strictly decreasing transformation.

Definition of conformal predictors

Every nonconformity measure determines a confidence predictor. Given a new object x_n and a level of significance, this predictor provides a prediction set

that should contain the object's label y_n. We obtain the set by supposing that y_n will have a value that makes (x_n, y_n) conform with the previous examples. The level of significance determines the amount of conformity (as measured by the p-value) that we require.

Formally, the *conformal predictor determined by a nonconformity measure* (A_n) is the confidence predictor Γ obtained by setting

$$\Gamma^\epsilon(x_1, y_1, \dots, x_{n-1}, y_{n-1}, x_n) \tag{2.17}$$

equal to the set of all labels $y \in \mathbf{Y}$ such that

$$\frac{|\{i = 1, \dots, n : \alpha_i \geq \alpha_n\}|}{n} > \epsilon \,, \tag{2.18}$$

where

$$\begin{aligned} \alpha_i &:= A_n(\langle (x_1, y_1), \dots, (x_{i-1}, y_{i-1}), (x_{i+1}, y_{i+1}), \dots, (x_{n-1}, y_{n-1}), \\ &\qquad (x_n, y)\rangle, (x_i, y_i)), \qquad i = 1, \dots, n-1, \\ \alpha_n &:= A_n(\langle (x_1, y_1), \dots, (x_{n-1}, y_{n-1})\rangle, (x_n, y)) \,. \end{aligned} \tag{2.19}$$

In general, a *conformal predictor* is a conformal predictor determined by some nonconformity measure.

The left-hand side of (2.18) is the p-value of (x_n, y) in the bag consisting of it and the old examples (cf. (2.16)). So our prediction with significance level ϵ (or confidence level $1 - \epsilon$) is that the value of y_n will make (x_n, y_n) have a p-value greater than ϵ when it is bagged with the old examples. We are 98% confident, for example, that we will get a value for y_n that gives (x_n, y_n) a p-value greater than 0.02. In other words, we are 98% confident that

$$\frac{\text{number of examples that conform worse or the same as } (x_n, y_n)}{n}$$

will exceed 0.02.

If A is a conformity measure, the conformal predictor determined by A is defined by (2.18) with "$\geq$" replaced by "$\leq$".

Validity

Proposition 2.3. *All conformal predictors are conservative.*

It follows by Proposition 2.2 that a conformal predictor is asymptotically conservative. Of course, more can be said. Using the law of the iterated logarithm instead of the law of large numbers, we can strengthen (2.12) to

$$\limsup_{n\to\infty} \frac{\mathrm{Err}_n^\epsilon(\Gamma, \omega) - n\epsilon}{\sqrt{2\epsilon(1-\epsilon) n \ln\ln n}} \leq 1 \,.$$

We will also state two finite-sample implications of Proposition 2.3: Hoeffding's inequality (see p. 287) implies that, for any positive integer N and any constant $\delta > 0$,

$$P\{\omega : \mathrm{Err}_N^\epsilon(\Gamma, \omega) \geq N(\epsilon + \delta)\} \leq \mathrm{e}^{-2N\delta^2} \,;$$

the central limit theorem implies that, for any constant c,

$$\limsup_{N\to\infty} P\left\{\omega : \mathrm{Err}_N^\epsilon(\Gamma, \omega) \geq N\epsilon + c\sqrt{N}\right\} \leq \frac{1}{\sqrt{2\pi}} \int_{\frac{c}{\sqrt{\epsilon(1-\epsilon)}}}^\infty \mathrm{e}^{-u^2/2} \mathrm{d}u \,.$$

For a graphical illustration of asymptotic conservativeness, see Fig. 1.5 on p. 10.

Smoothed conformal predictors

In this section we introduce a modification of conformal predictors which will allow us to simplify and strengthen Proposition 2.3. The *smoothed conformal predictor determined by the nonconformity measure* (A_n) is the following randomized confidence predictor Γ: the set

$$\Gamma^\epsilon(x_1, \tau_1, y_1, \dots, x_{n-1}, \tau_{n-1}, y_{n-1}, x_n, \tau_n)$$

consists of the $y \in \mathbf{Y}$ satisfying

$$\frac{|\{i = 1, \dots, n : \alpha_i > \alpha_n\}| + \tau_n |\{i = 1, \dots, n : \alpha_i = \alpha_n\}|}{n} > \epsilon \,, \tag{2.20}$$

where α_i are defined, as before, by (2.19). The left-hand side of (2.20) is called the *smoothed p-value*.

The main difference of (2.20) from (2.18) is that in the former we treat the borderline cases $\alpha_i = \alpha_n$ more carefully. Instead of increasing the p-value by $1/n$ for each $\alpha_i = \alpha_n$, we increase it by a random amount between 0 and $1/n$.

When n is not too small, it is typical for almost all $\alpha_1, \dots, \alpha_n$ to be different, and then there is very little difference between conformal predictors and smoothed conformal predictors.

Proposition 2.4. *Any smoothed conformal predictor is exactly valid.*

This proposition will be proved in Chap. 8 (as a special case of Theorem 8.1 on p. 193). It immediately implies Proposition 2.3: if a smoothed conformal predictor Γ and a conformal predictor $\Gamma^\dagger$ are constructed from the same nonconformity measure, the latter's errors $\mathrm{err}_n^\dagger$ never exceed the former's errors err_n, $\mathrm{err}_n^\dagger \leq \mathrm{err}_n$. It also implies

Corollary 2.5. *Every smoothed conformal predictor is asymptotically exact.*

A general scheme for defining nonconformity

There are many different ways of defining nonconformity measures; we will consider them more systematically in the following two chapters, and here we will only explain the most basic approach, which in the next section will be illustrated in the case of regression, $\mathbf{Y} = \mathbb{R}$.

Suppose we are given a bag $\{z_1, \dots, z_n\}$ and we want to estimate the nonconformity of each example z_i in the bag, as in (2.15). (It is clear that the values (2.15) determine the nonconformity measure, and we will often define nonconformity measures by specifying the nonconformity scores (2.15).)

There is a natural solution if we are given a simple predictor (2.2) whose output does not depend on the order in which the old examples are presented. The simple predictor D then defines a prediction rule $D_{\{z_1,\dots,z_n\}} : \mathbf{X} \to \mathbf{Y}$ by the formula

$$D_{\{z_1,\dots,z_n\}}(x) := D(z_1, \dots, z_n, x) .$$

A natural measure of nonconformity of z_i is the deviation of the predicted label

$$\hat{y}_i := D_{\{z_1,\dots,z_n\}}(x_i) \tag{2.21}$$

from the true label y_i. In this way any simple predictor, combined with a suitable measure of deviation of $\hat{y}_i$ from y_i, leads to a nonconformity measure and, therefore, to a conformal predictor.

The simplest way of measuring the deviation of $\hat{y}_i$ from y_i is to take the absolute value $|y_i - \hat{y}_i|$ of their difference as α_i. We could try, however, to somehow "standardize" $|y_i - \hat{y}_i|$ taking into account typical values we expect the difference between y_i and $\hat{y}_i$ to take given the object x_i. Yet another approach is to take $\alpha_i := |y_i - \hat{y}_{(i)}|$, where $\hat{y}_{(i)}$ is the *deleted prediction*

$$\hat{y}_{(i)} := D_{\{z_1,\dots,z_{i-1},z_{i+1},\dots,z_n\}}(x_i)$$

computed by applying to x_i the prediction rule found from the data set with the example z_i deleted. The rationale behind this deletion is that z_i, even if it is an outlier, can influence the prediction rule $D_{\{z_1,\dots,z_n\}}$ so heavily that $\hat{y}_i$ will become close to y_i, even though y_i can be very far from $\hat{y}_{(i)}$.

More generally, the prediction rule $D_{\{z_1,\dots,z_n\}}$ (or $D_{\{z_1,\dots,z_{i-1},z_{i+1},\dots,z_n\}}$) may map $\mathbf{X}$ to some *prediction space* $\hat{\mathbf{Y}}$, not necessarily coinciding with $\mathbf{Y}$ (e.g., $\hat{\mathbf{Y}}$ can be the set of all probability distributions on $\mathbf{Y}$). An *invariant simple predictor* is a function D that maps each bag $\{z_1, \dots, z_n\}$ of each size n to a prediction rule $D_{\{z_1,\dots,z_n\}} : \mathbf{X} \to \hat{\mathbf{Y}}$ and such that the function

$$(\{z_1, \dots, z_n\}, x) \mapsto D_{\{z_1,\dots,z_n\}}(x)$$

of the type $\mathbf{Z}^{(n)} \times \mathbf{X} \to \hat{\mathbf{Y}}$ is measurable for all n. A *discrepancy measure* is a measurable function $\Delta : \mathbf{Y} \times \hat{\mathbf{Y}} \to \mathbb{R}$. Given an invariant simple predictor D and a discrepancy measure Δ, we define functions A_n, $n = 1, 2, \dots$, as follows: for any $((x_1, y_1), \dots, (x_n, y_n)) \in \mathbf{Z}^*$, the values

$$\alpha_i = A_n\big(\{(x_1, y_1), \dots, (x_{i-1}, y_{i-1}), (x_{i+1}, y_{i+1}), \dots, (x_n, y_n)\}, (x_i, y_i)\big) \tag{2.22}$$

are defined by the formula

$$\alpha_i := \Delta\left(y_i, D_{\{(x_1,y_1),\dots,(x_n,y_n)\}}(x_i)\right) \tag{2.23}$$

or the formula

$$\alpha_i := \Delta\left(y_i, D_{\{(x_1,y_1),\dots,(x_{i-1},y_{i-1}),(x_{i+1},y_{i+1}),\dots,(x_n,y_n)\}}(x_i)\right) . \tag{2.24}$$

It can be checked easily that in both cases (A_n) form a nonconformity measure.

2.3 Ridge regression confidence machine

In this section we will implement conformal prediction based on (2.23) and (2.24) concentrating on the case where the underlying simple predictor is ridge regression or nearest neighbors regression, which are two of the most standard regression algorithms. In the case of regression there is an obvious difficulty in implementing the idea of conformal prediction: it appears that to form a prediction set (2.17) we need to examine each potential classification y (cf. (2.19)). We will see, however, that there often is a feasible way to compute (2.17); in particular, this is the case for ridge regression and nearest neighbors regression. (We will make little effort to optimize the computational resources required, so "feasible" essentially means "avoiding examining infinitely many cases" here. Much faster algorithms will be constructed in §4.1.)

Least squares and ridge regression

Ridge regression and its special case, least squares, are among the most widely used regression algorithms. Least squares is the classical algorithm (going back to Gauss and Legendre), and ridge regression is its modification proposed in the 1960s. In this subsection we will describe least squares and ridge regression as simple predictors; for further details, see, e.g., Montgomery et al. 2001.

Suppose $\mathbf{X} = \mathbb{R}^p$ (objects are vectors consisting of p attributes), $\mathbf{Y} = \mathbb{R}$ (we are dealing with the problem of regression), and we are given a training set[2] $z_1, \dots, z_n$. To approximate the data, the ridge regression procedure recommends calculating the value $w \in \mathbb{R}^p$ where

$$a\|w\|^2 + \sum_{i=1}^{n} (y_i - w \cdot x_i)^2 \to \min \tag{2.25}$$

[2]It would be more correct to say "training sequence" or "training bag", since we do not assume that all z_i are different, but we will use a more familiar term (as we already did in Chap. 1).

is attained; a is a nonnegative constant called the *ridge parameter*. The ridge regression prediction $\hat{y}$ for the label y of an object x is then $\hat{y} := w \cdot x$. Least squares is the special case corresponding to $a = 0$.

We can naturally represent the ridge regression procedure in a matrix form. Let Y_n be the (column-) vector

$$Y_n := (y_1, \dots, y_n)' \tag{2.26}$$

of the labels and X_n be the $n \times p$ matrix formed from the objects

$$X_n := (x_1, x_2, \dots, x_n)' \tag{2.27}$$

Now we can represent the ridge regression procedure (2.25) as

$$a\|w\|^2 + \|Y_n - X_n w\|^2 \to \min , \tag{2.28}$$

or

$$Y_n' Y_n - 2w' X_n' Y_n + w'(X_n' X_n + aI_p)w \to \min .$$

Taking the derivative in w we obtain

$$2(X_n' X_n + aI_p)w - 2X_n' Y_n = 0 ,$$

or

$$w = (X_n' X_n + aI_p)^{-1} X_n' Y_n . \tag{2.29}$$

Standard statistical textbooks mainly discuss the case $a = 0$ (least squares). It is easy to see, however, that not only least squares is a special case of ridge regression, but ridge regression can be reduced to least squares as well: the solution of (2.28) for a general $a \geq 0$ can be found as the solution to the least squares problem

$$\|\overline{Y} - \overline{X} w\|^2 \to \min ,$$

where $\overline{Y}$ is Y_n extended by adding p 0s on top and $\overline{X}$ is X_n extended by adding the $p \times p$ matrix $\sqrt{a} I_p$ on top.

Basic RRCM

In this subsection we consider the conformal predictor determined by the nonconformity measure (2.22)–(2.23), with Δ the Euclidean distance and D the ridge regression procedure. Therefore, α_i are now the absolute values of the *residuals* $e_i := y_i - \hat{y}_i$, where $\hat{y}_i$ is the ridge regression prediction for x_i based on the training set $x_1, y_1, \dots, x_n, y_n$. Two slightly more sophisticated approaches will be considered in the following subsection.

From (2.29) we can see that the ridge regression prediction for an object x is

$$x' w = x'(X_n' X_n + aI_p)^{-1} X_n' Y_n ; \tag{2.30}$$

therefore, the predictions $\hat{y}_i$ for the objects x_i are given by

$$\hat{Y}_n := (\hat{y}_1, \dots, \hat{y}_n)' = X_n(X_n' X_n + aI_p)^{-1} X_n' Y_n .$$

The matrix

$$H_n := X_n(X_n' X_n + aI_p)^{-1} X_n' \tag{2.31}$$

is called the *hat matrix* (since it transforms the y_i into the hatted form $\hat{y}_i$) and plays an important role in the standard regression theory. This matrix, as well as $I_n - H_n$, is symmetric and idempotent when $a = 0$ (remember that a symmetric matrix M is *idempotent* if $MM = M$). Therefore, the vector of nonconformity scores $(\alpha_1, \dots, \alpha_n)'$ can be written in the form

$$|Y_n - H_n Y_n| = |(I_n - H_n) Y_n| .$$

Now suppose that we know the incomplete data sequence (2.4), we are given a significance level ϵ, and we want to compute the prediction set output by the conformal predictor determined by the nonconformity scores $\alpha_i = |e_i|$. Let y be a possible label for x_n and $Y := (y_1, \dots, y_{n-1}, y)'$. Note that $Y = (y_1, \dots, y_{n-1}, 0)' + (0, \dots, 0, y)'$ and so the vector of nonconformity scores can be represented as $|A + By|$ where

$$A = (I_n - H_n)(y_1, \dots, y_{n-1}, 0)'$$

and

$$B = (I_n - H_n)(0, \dots, 0, 1)' .$$

Therefore, each $\alpha_i = \alpha_i(y)$ varies piecewise-linearly as we change y. It is clear that the p-value $p(y)$ (defined to be the left-hand side of (2.18)) corresponding to y can only change at points where $\alpha_i(y) - \alpha_n(y)$ changes sign for some $i = 1, \dots, n-1$. This means that we can calculate the set of points y on the real line that have the p-value $p(y)$ exceeding ϵ rather than having to try all possible y, leading us to a feasible prediction algorithm.

For each $i = 1, \dots, n$, let

$$S_i := \{y : \alpha_i(y) \geq \alpha_n(y)\} = \{y : |a_i + b_i y| \geq |a_n + b_n y|\} , \tag{2.32}$$

where a_i and b_i are the components of A and B ($A = (a_1, \dots, a_n)'$ and $B = (b_1, \dots, b_n)'$). Each set S_i (always closed) will either be the real line, the union of two rays, a ray, an interval, a point (which is a special case of an interval), or empty. Indeed, as we are interested in $|a_i + b_i y|$ we can assume $b_i \geq 0$ for $i = 1, \dots, n$ (if necessary, multiply both a_i and b_i by -1). If $b_i \neq b_n$ then $\alpha_i(y)$ and $\alpha_n(y)$ are equal at two points (which may coincide):

$$-\frac{a_i - a_n}{b_i - b_n} \quad \text{and} \quad -\frac{a_i + a_n}{b_i + b_n} ; \tag{2.33}$$

in this case, S_i is an interval (possibly a point) or the union of two rays. If $b_i = b_n \neq 0$ then $\alpha_i(y) = \alpha_n(y)$ at the only point

$$-\frac{a_i + a_n}{2b_i} \tag{2.34}$$

(and S_i is a ray) unless $a_i = a_n$, in which case $S_i = \mathbb{R}$. If $b_i = b_n = 0$, S_i is either $\emptyset$ or $\mathbb{R}$.

To calculate the p-value $p(y)$ for any potential label y of x_n, we count how many S_i include y and divide by n:

$$p(y) = \frac{|\{i = 1, \dots, n : y \in S_i\}|}{n} .$$

It is clear that as y increases $p(y)$ can only change at the points (2.33) and (2.34) and so for any threshold ϵ we can find the explicit representation of the set of y for which $p(y) > \epsilon$ as the union of finitely many intervals and rays. The following algorithm gives a slightly easier to describe implementation of this idea; it arranges the points (2.33) and (2.34) into an increasing sequence $y_{(1)}, \dots, y_{(m)}$, adds $y_{(0)} := -\infty$ and $y_{(m+1)} := \infty$, and then computes $N(j)$, the number of i such that $(y_{(j)}, y_{(j+1)}) \subseteq S_i$, for $j = 0, \dots, m$, and $M(j)$, the number of i such that $y_{(j)} \in S_i$, for $j = 1, \dots, m$. The algorithm is given a (small) set of significance levels ϵ_k, $k = 1, \dots, K$, and outputs the corresponding nested family of prediction sets $\Gamma_n^{\epsilon_k}$, $k = 1, \dots, K$.

ALGORITHM RRCM (RIDGE REGRESSION CONFIDENCE MACHINE)

$C := I_n - X_n(X_n'X_n + aI_p)^{-1}X_n'$, X_n being defined by (2.27);
$A = (a_1, \dots, a_n)' := C(y_1, \dots, y_{n-1}, 0)'$;
$B = (b_1, \dots, b_n)' := C(0, \dots, 0, 1)'$;
FOR $i = 1, \dots, n$:
  IF $b_i < 0$ THEN $a_i := -a_i$; $b_i := -b_i$ END IF
END FOR;
$P := \emptyset$;
FOR $i = 1, \dots, n$:
  IF $b_i \neq b_n$ THEN add (2.33) to P END IF;
  IF $b_i = b_n \neq 0$ AND $a_i \neq a_n$ THEN add (2.34) to P END IF
END FOR;
sort P in ascending order obtaining $y_{(1)}, \dots, y_{(m)}$;
set $y_{(0)} := -\infty$ and $y_{(m+1)} := \infty$;
$N(j) := 0$, $j = 0, \dots, m$;
FOR $i = 1, \dots, n$:
  FOR $j = 0, \dots, m$:
    IF $|a_i + b_i y| \geq |a_n + b_n y|$ for $y \in (y_{(j)}, y_{(j+1)})$
    THEN $N(j) := N(j) + 1$
    END IF
  END FOR
END FOR;
$M(j) := 0$, $j = 1, \dots, m$;
FOR $i = 1, \dots, n$:

  FOR $j = 1, \dots, m$:
    IF $|a_i + b_i y_{(j)}| \geq |a_n + b_n y_{(j)}|$
    THEN $M(j) := M(j) + 1$
    END IF
  END FOR
END FOR;
FOR $k = 1, \dots, K$:
  $\Gamma_n^{\epsilon_k} := \left(\cup_{j:N(j)/n > \epsilon} (y_{(j)}, y_{(j+1)})\right) \cup \{y_{(j)} : M(j)/n > \epsilon\}$
END FOR.

This algorithm is run by Predictor at each trial $n = 1, 2, \dots$ of the on-line prediction protocol (given on p. 20).

Let us suppose that the number p of attributes is constant. It is clear that Algorithm RRCM requires computation time $O(n^2)$. A simple modification of the algorithm, however, reduces the required computation time to $O(n \log n)$.

First, it is clear that

$$A = (y_1, \dots, y_{n-1}, 0)' - X_n \left[(X_n'X_n + aI_p)^{-1}X_n'(y_1, \dots, y_{n-1}, 0)'\right]$$

and

$$B = (0, \dots, 0, 1)' - X_n \left[(X_n'X_n + aI_p)^{-1}X_n'(0, \dots, 0, 1)'\right]$$

can be computed in time $O(n)$. Sorting P can be done in time $O(n \log n)$ (see, e.g., Cormen et al. 2001, Part II). Therefore, it suffices to show that the two double loops (computing N and computing M) in Algorithm RRCM can be implemented in time $O(n)$. Instead of computing the array $N(j)$, $j = 0, \dots, m$, directly, we can first compute $N'(j) := N(j) - N(j-1)$, $j = 0, \dots, m$, with $N(-1) := 0$; it is easy (takes time $O(n)$) to compute N from N'. Analogously, we can compute $M'(j) := M(j) - M(j-1)$, $j = 1, \dots, m$, with $M(0) := 0$, instead of M. To find N' and M' in time $O(n)$, initialize $N'(j) := 0$, $j = 0, \dots, m$, $M'(j) := 0$, $j = 1, \dots, m$, and for each example $i = 1, \dots, n$ do the following:

- if S_i (see (2.32)) is empty, do nothing;
- if S_i contains only one point, $S_i = \{y_{(j)}\}$, set $M'(j) := M'(j) + 1$ and $M'(j+1) := M'(j+1) - 1$ (assignments that do not make sense, such as $M'(j+1) := M'(j+1) - 1$ for $j = m$, are simply ignored);
- if S_i is an interval $[y_{(j_1)}, y_{(j_2)}]$ and $j_1 < j_2$, set $M'(j_1) := M'(j_1) + 1$, $M'(j_2 + 1) := M'(j_2 + 1) - 1$, $N'(j_1) := N'(j_1) + 1$, $N'(j_2) := N'(j_2) - 1$;
- if S_i is a ray $(-\infty, y_{(j)}]$, set $M'(1) := M'(1) + 1$, $M'(j+1) := M'(j+1) - 1$, $N'(0) := N'(0) + 1$, $N'(j) := N'(j) - 1$;
- if S_i is a ray $[y_{(j)}, \infty)$, set $M'(j) := M'(j) + 1$, $N'(j) := N'(j) + 1$;
- if S_i is the union $(-\infty, y_{(j_1)}] \cup [y_{(j_2)}, \infty)$ of two rays such that $j_1 < j_2$, set $M'(1) := M'(1) + 1$, $M'(j_1 + 1) := M'(j_1 + 1) - 1$, $M'(j_2) := M'(j_2) + 1$, $N'(0) := N'(0) + 1$, $N'(j_1) := N'(j_1) - 1$, $N'(j_2) := N'(j_2) + 1$;

- if S_i is the real line $(-\infty, \infty)$, set $M'(1) := M'(1) + 1$ and $N'(0) := N'(0) + 1$.

Two modifications

The algorithm developed in the previous subsection can be easily modified to allow two alternative ways of computing nonconformity scores. To simplify formulas, we assume $a = 0$ in this subsection (i.e., we will consider least squares). A *least squares confidence machine* (LSCM) is an RRCM with the ridge parameter a set to 0.

First we consider the special case of (2.24) where α_i is defined to be the absolute value $|y_i - \hat{y}_{(i)}|$ of the *deleted residual* $e_{(i)} := y_i - \hat{y}_{(-i)}$, where $\hat{y}_{(-i)}$ is the least squares prediction for y_i computed from x_i based on the training set $x_1, y_1, \ldots, x_{i-1}, y_{i-1}, x_{i+1}, y_{i+1}, \ldots, x_n, y_n$. It is well known in statistics that to compute the deleted residuals $e_{(i)}$ we do not need to perform n regressions; they can be easily computed from the usual residuals $e_i = |y_i - \hat{y}_i|$ by the formula

$$e_{(i)} = \frac{e_i}{1 - h_{ii}} , \tag{2.35}$$

where h_{ii} is the ith diagonal element of the hat matrix H. (For a proof, see Montgomery et al. 2001, Appendix C.7.)

Let us call the conformal predictor determined by the nonconformity scores $\alpha_i := |e_{(i)}|$ the *deleted LSCM*. Algorithm LSCM will implement the deleted LSCM if A and B are redefined as follows:

$$a_i := \frac{a_i}{1 - h_{ii}}, \quad b_i := \frac{b_i}{1 - h_{ii}}, \quad i = 1, \ldots, n .$$

It is clear that each

$$h_{ii} = x_i'(X_n'X_n)^{-1}x_i$$

can be computed from $(X_n'X_n)^{-1}$ in time $O(1)$ (again assuming that the number p of attributes is constant), and so the deleted LSCM can also be implemented in time $O(n \log n)$.

Another natural modification of LSCM is half-way between the LSCM and the deleted LSCM: the nonconformity scores are taken to be

$$\alpha_i := \frac{|e_i|}{\sqrt{1 - h_{ii}}} . \tag{2.36}$$

We will explain the motivation behind this choice momentarily, but first describe how to implement the *studentized LSCM* determined by these nonconformity scores. The implementation is just Algorithm LSCM with A and B redefined as

$$a_i := \frac{a_i}{\sqrt{1 - h_{ii}}}, \quad b_i := \frac{b_i}{\sqrt{1 - h_{ii}}}, \quad i = 1, \ldots, n .$$

The computation time is again $O(n \log n)$.

Now we explain the standard motivation for studentized LSCM. (Remember that this motivation has no bearing on the validity of the constructed conformal predictor; in particular, the smoothed version of this confidence predictor will make errors independently with probability ϵ at each significance level ϵ regardless whether the assumption of normal noise we are about to make is satisfied or not; cf. §10.3.) Imagine that the labels y_i are generated from the deterministic objects x_i in the following way:

$$y_i = w \cdot x_i + \xi_i , \tag{2.37}$$

where ξ_i are independent normal random variables with the mean 0 and same variance σ^2 (random noise). Set $\xi := (\xi_1, \ldots, \xi_n)'$. Since the vector of residuals is $e = (I_n - H_n)Y_n$ (see above), we obtain

$$e = (I_n - H_n)(X_n w + \xi) = (I_n - H_n)\xi \tag{2.38}$$

for any fixed w (the true parameters); therefore, the covariance matrix of the residuals is

$$\mathrm{var}(e) = \mathrm{var}((I_n - H_n)\xi) = (I_n - H_n)\mathrm{var}(\xi)(I_n - H_n)' = \sigma^2(I_n - H_n) ,$$

since $\mathrm{var}(\xi) = \sigma^2 I_n$ and $I_n - H_n$ is symmetric and idempotent. We can see that the variance of e_i is $(1 - h_{ii})\sigma^2$, and the scaling of the residuals e_i by dividing by $\sqrt{1 - h_{ii}}$ will equalize their variances (and even their distributions, since by (2.38), e_i are normally distributed).

Notice that according to our motivational model (2.37) the level of noise ξ_i does not depend on the observed object x_i (the variance of ξ_i remains the same, σ^2). Even in this case, it may be useful to scale residuals. If we suspect that noise can be different in different parts of the object space, heavier scaling may become necessary for satisfactory prediction.

Dual form ridge regression

Least squares and ridge regression procedures can only deal with situations where the number of parameters p is relatively small since they involve inverting a $p \times p$ matrix. They are carried over to high-dimensional problems using the so-called "kernel trick", introduced in machine learning by Vapnik (see, e.g., Vapnik 1995, 1998). (The formulas arrived at by using the kernel trick coincide with those obtained by means of Gaussian processes and reproducing kernel Hilbert spaces; see §2.7.)

We first state the ridge regression procedure in the "dual form". The traditional statistical approach to dualization is to use the easy-to-check matrix equality

$$X_n(X_n'X_n + aI_p)^{-1} = (X_nX_n' + aI_n)^{-1}X_n , \tag{2.39}$$

which can be equivalently rewritten as

$$(X_n'X_n + aI_p)^{-1}X_n' = X_n'(X_nX_n' + aI_n)^{-1} .$$

(To see that (2.39) is true, multiply it by $(X_nX_n' + aI_n)$ on the left and $(X_n'X_n + aI_p)$ on the right.) Using (2.39), we can rewrite the ridge regression prediction for an object x based on examples $(x_1, y_1, \dots, x_n, y_n)$ as

$$\hat{y} = Y_n'X_n(X_n'X_n + aI_p)^{-1}x = Y_n'(X_nX_n' + aI_n)^{-1}X_nx , \tag{2.40}$$

where the $n \times p$ matrix X_n and vector Y_n are defined as before. The crucial property of the representation $\hat{y} = Y_n'(X_nX_n' + aI_n)^{-1}X_nx$ is that it depends on the objects $x_1, \dots, x_n, x$ only via the scalar products between them. In particular, if the object space $\mathbf{X}$ is mapped into another Euclidean space (called the *feature space*) $\mathbf{H}$, $F : \mathbf{X} \to \mathbf{H}$, and ridge regression is performed in the feature space, the prediction (2.40) can be written in the form

$$\hat{y} = Y_n'(K_n + aI_n)^{-1}k_n , \tag{2.41}$$

where K_n is the matrix with elements $(K_n)_{i,j} := \mathcal{K}(x_i, x_j)$, k_n is the vector with elements $(k_n)_i := \mathcal{K}(x, x_i)$, and $\mathcal{K}$ is the *kernel*, defined by

$$\mathcal{K}(x^{(1)}, x^{(2)}) := F(x^{(1)}) \cdot F(x^{(2)}) \tag{2.42}$$

for all $x^{(1)}, x^{(2)} \in \mathbf{X}$. The hat matrix in the dual representation is

$$H_n = X_n(X_n'X_n + aI_p)^{-1}X_n' = (X_nX_n' + aI_n)^{-1}X_nX_n' ,$$

and if ridge regression is carried out in the feature space, this becomes

$$H_n = (K_n + aI_n)^{-1}K_n . \tag{2.43}$$

Now it is easy to represent the RRCM algorithm in the kernel form: the only difference from Algorithm RRCM is that now C is defined as $I_n - H_n$ with H_n given by (2.43).

The computation time of the kernel form of the RRCM algorithm is $O(n^2)$. This can be seen from the well-known (see, e.g., Henderson and Searle 1981, (8)) and easy-to-check formula

$$\begin{pmatrix} K & k \\ k' & \kappa \end{pmatrix}^{-1} = \begin{pmatrix} K^{-1} + dK^{-1}kk'K^{-1} & -dK^{-1}k \\ -dk'K^{-1} & d \end{pmatrix} , \tag{2.44}$$

where K is a square matrix, k a vector, κ a number, and

$$d := \frac{1}{\kappa - k'K^{-1}k} .$$

Indeed, by this formula $(K_n + aI_n)^{-1}$ can be updated from the previous trial of the on-line learning protocol in time $O(n^2)$, and both

$$A = (y_1, \dots, y_{n-1}, 0)' - (K_n + aI_n)^{-1}(K_n(y_1, \dots, y_{n-1}, 0)')$$

and

$$B = (0, \dots, 0, 1)' - (K_n + aI_n)^{-1}(K_n(0, \dots, 0, 1)')$$

can be computed in time $O(n^2)$. There are some conditions on the validity of formula (2.44), but they are satisfied in our context: the theorem on normal correlation (see, e.g., Shiryaev 1996, Theorem II.13.2) implies that d is well-defined (and positive) whenever the matrix $\left(\begin{smallmatrix} K & k \\ k' & \kappa \end{smallmatrix}\right)$ is positive definite; the latter condition is satisfied when the formula is used for updating $(K_n + aI_n)^{-1}$, $a > 0$, to $(K_{n+1} + aI_{n+1})^{-1}$ (in which case $K = K_n + aI_n$, $k = k_n$, and $\kappa = \mathcal{K}(x_n, x_n) + a$).

It is easy to see that the above construction also works in the case where $\mathbf{H}$ is an arbitrary Hilbert space. The essence of the kernel trick is that one does not need to consider the feature space in explicit form (Vapnik 1998, §10.5.2). It is clear that any kernel (2.42) is symmetric,

$$\mathcal{K}(x^{(1)}, x^{(2)}) = \mathcal{K}(x^{(2)}, x^{(1)}), \quad \forall x^{(1)}, x^{(2)} \in \mathbf{X} , \tag{2.45}$$

and nonnegative definite,

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\mathcal{K}(x^{(i)}, x^{(j)})a_ia_j \geq 0,$$
$$\forall x^{(1)}, \dots, x^{(m)} \in \mathbf{X}, \forall a_1, \dots, a_m \in \mathbb{R} . \tag{2.46}$$

(To see that (2.46) is true, notice that

$$\|a_1F(x^{(1)}) + \dots + a_mF(x^{(m)})\|^2$$
$$= \left(a_1F(x^{(1)}) + \dots + a_mF(x^{(m)})\right) \cdot \left(a_1F(x^{(1)}) + \dots + a_mF(x^{(m)})\right) \geq 0 .)$$

It turns out that the opposite statement is also true: any function $\mathcal{K} : \mathbf{X}^2 \to \mathbb{R}$ that is symmetric and nonnegative definite can be represented in the form (2.42). There are many proofs of this result, but one of the simplest arguments is "probabilistic": any symmetric nonnegative definite matrix is the covariance matrix of a set of (zero-mean) normal random variables; the Daniell–Kolmogorov (Kolmogorov 1933a) theorem then immediately implies that any symmetric nonnegative definite function $\mathcal{K}(x^{(1)}, x^{(2)})$ is the "infinite covariance matrix" of a zero-mean Gaussian random field $\xi(x)$, $x \in \mathbf{X}$:

$$\mathcal{K}(x^{(1)}, x^{(2)}) = \mathbb{E}\left(\xi(x^{(1)})\xi(x^{(2)})\right) .$$

The last equality is a special case of (2.42) since the zero-mean finite-variance random variables with dot product

$$\xi \cdot \eta := \mathbb{E}(\xi\eta)$$

form a Hilbert space (not necessarily separable; see, e.g., Shiryaev 1996, §II.11).

Nearest neighbors regression

Least squares and ridge regression are just two of the standard regression algorithms; conformal predictors can be implemented in a feasible way for nonconformity measures based on many other regression algorithms. Such an implementation is especially simple in the case of the nearest neighbors algorithm.

The idea of the k-nearest neighbors algorithms, where k is the number of "neighbors" taken into account, is as follows. Suppose the object space $\mathbf{X}$ is a metric space (for example, the usual Euclidean distance is often used if $\mathbf{X} = \mathbb{R}^p$). To give a prediction for a new object x_n, find the k objects $x_{i_1}, \dots, x_{i_k}$ among the known examples that are nearest to x_i in the sense of the chosen metric (assuming, for simplicity, that there are no ties). In the problem of classification, the predicted classification $\hat{y}_n$ is obtained by "voting": it is defined to be the most frequent label among $y_{i_1}, \dots, y_{i_k}$. In regression, we can take, e.g., the mean or the median of $y_{i_1}, \dots, y_{i_k}$.

We will only consider the version of the k-nearest neighbors regression (k-NNR) where the prediction $\hat{y}$ for a new object x based on the training set (x_i, y_i), $i = 1, \dots, n$, is defined to be the arithmetic mean of the labels of the k nearest neighbors of x among $x_1, \dots, x_n$. It will be easy to see that the more robust procedure where arithmetic mean is replaced by median also leads to a feasible conformal predictor.

Consider the special case of the nonconformity scores (2.24) where $\alpha_i := |y_i - \hat{y}_{(i)}|$ and $\hat{y}_{(i)}$ is the k-NNR prediction for x_i based on the training set (x_j, y_j), $j = 1, \dots, i-1, i+1, \dots, n$. The conformal predictor determined by this nonconformity measure (*k-NNR conformal predictor*) is implemented by the RRCM algorithm with the only modification that a_i and b_i are now defined as follows (we assume that $n > k$ and that all distances between the objects are different):

- a_n is the minus arithmetic mean of the labels of x_n's k nearest neighbors and $b_n = 1$;
- if $i < n$ and x_n is among the k nearest neighbors of x_i, a_i is x_i's label minus the arithmetic mean of the labels of those nearest neighbors with x_n's label set to 0, and $b_i = -1/k$;
- if $i < n$ and x_n is not among the k nearest neighbors of x_i, a_i is x_i's label minus the arithmetic mean of the labels of x_i's k nearest neighbors, and $b_i = 0$.

(It is obvious that the nonconformity score for the ith example is $\alpha_i = |a_i + b_i y|$, where y is the label for x_n that is being tried.)

Instead of measuring distance in the original example space $\mathbf{X}$ we can measure it in the feature space, which corresponds to using the function

$$\begin{aligned}\rho(x^{(1)}, x^{(2)}) &:= \|F(x^{(1)}) - F(x^{(2)})\|^2 \\ &= \left(F(x^{(1)}) - F(x^{(2)})\right) \cdot \left(F(x^{(1)}) - F(x^{(2)})\right) \\ &= \mathcal{K}(x^{(1)}, x^{(1)}) + \mathcal{K}(x^{(2)}, x^{(2)}) - 2\mathcal{K}(x^{(1)}, x^{(2)})\end{aligned}$$

as the distance (remember that in the case of k-NNR conformal prediction we are only interested in the distance up to a monotonic transformation).

Experimental results

In this subsection we empirically test the validity and evaluate the efficiency of some of the conformal predictors introduced earlier. We start from RRCM, reporting only results for $a = 1$ on the randomly permuted Boston Housing data set (this data set and the rationale for shuffling, counteracting possible deviations from exchangeability, are described in Appendix B). A dummy attribute always taking value 1 (to allow a non-zero intercept) was added to each example, and at each trial each attribute was linearly scaled for the known objects to span the interval $[-1, 1]$ (or $[0, 0]$, if the attribute took the same value for all known objects, as described in Appendix B). Figures 2.1–2.3 show the performance of RRCM in regard of its efficiency. In Fig. 2.1, the solid line shows, for each $n = 1, \dots, 506$, the median $M_n^{99\%}$ of the widths of the convex hulls $\operatorname{co} \Gamma_i^{1\%}$ of the prediction sets $\Gamma_i^{1\%}$, $i = 1, \dots, n$, at confidence level 99%; similarly, the dashed line shows $M_n^{95\%}$ and the dash-dot line shows $M_n^{80\%}$. Figure 2.2 presents more detailed information for the performance at the confidence level 95%: not only the median $M_n^{95\%}$ of the widths of $\operatorname{co} \Gamma_i^{5\%}$, $i = 1, \dots, n$, but also the upper and lower quartiles of those widths; the cumulative number of errors at this confidence level is also given. Both Figs. 2.1 and 2.2 give, for comparison, a graph reflecting the performance of the ridge regression procedure as a simple predictor: the dotted line shows the sequence $2A(n)$, where $A(n)$ is the median of the sequence $|y_i - \hat{y}_i|$, $i = 1, \dots, n$, of distances between the true label y_i and the prediction $\hat{y}_i$ given by the ridge regression procedure according to (2.30). This line lies well below the other lines, the main reason being that we considered confidence levels of 80% and above; lowering the plank to 50% leads to near coincidence (Fig. 2.3). The cumulative error lines are close to straight lines with correct slopes (Fig. 2.4), although because of the small sample size the imperfections due to statistical fluctuations are more noticeable than in Fig. 1.5 on p. 10.

In fact, the RRCM's performance was not sensitive to moderate changes in a: e.g., running the algorithm for $a = 0$ (LSCM), whether modified (deleted or studentized LSCM) or not, produced virtually identical figures.

The quality of prediction can be improved by using non-linear methods. Figure 2.5 shows the performance of the kernel RRCM with the second-order polynomial kernel

$$\mathcal{K}(x^{(1)}, x^{(2)}) := \left(1 + x^{(1)} \cdot x^{(2)}\right)^2$$

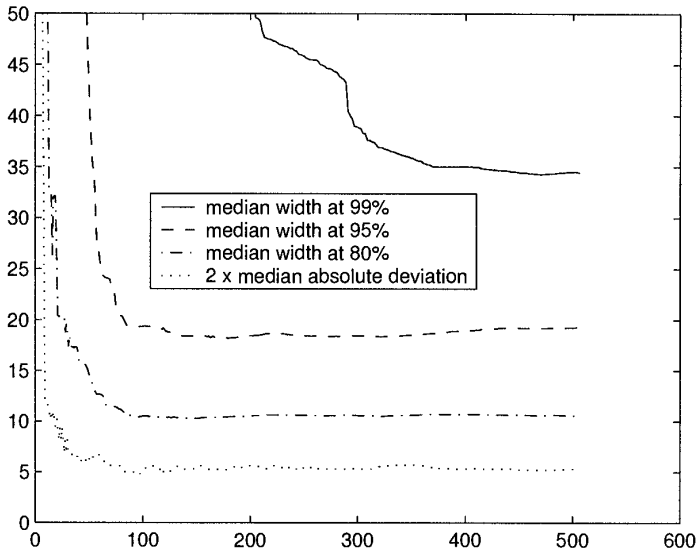

Fig. 2.1. The on-line performance of RRCM on the randomly permuted Boston Housing data set (of size 506)

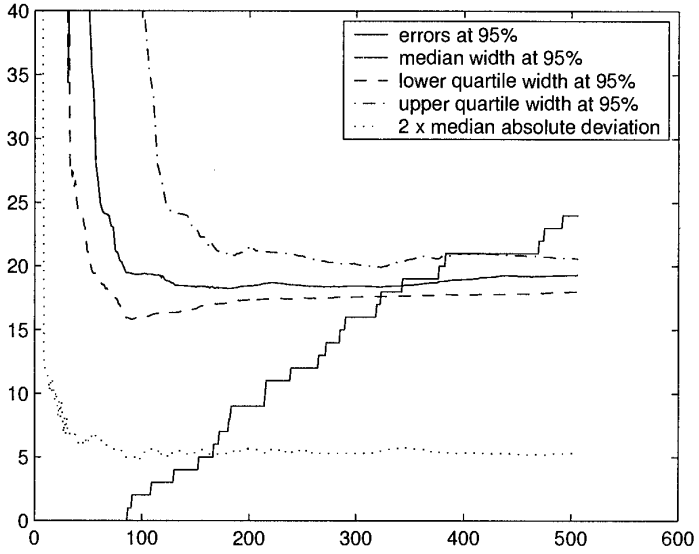

Fig. 2.2. The on-line performance of RRCM on the randomly permuted Boston Housing data set at the confidence level 95%

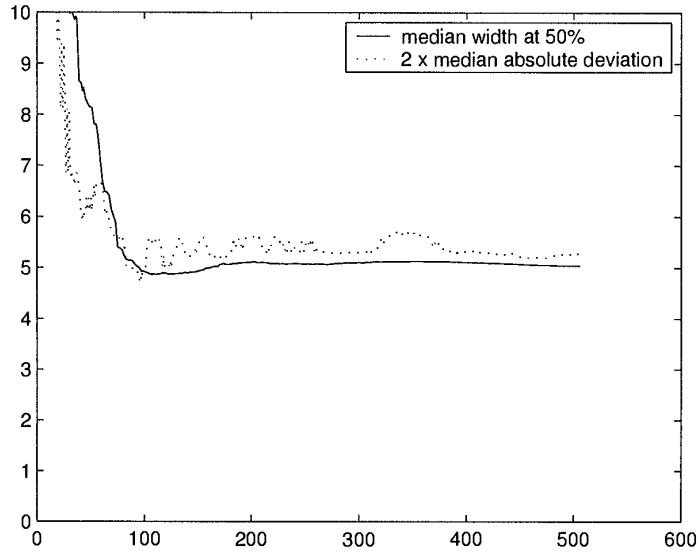

Fig. 2.3. The on-line performance of RRCM on the randomly permuted Boston Housing data set at the confidence level 50%

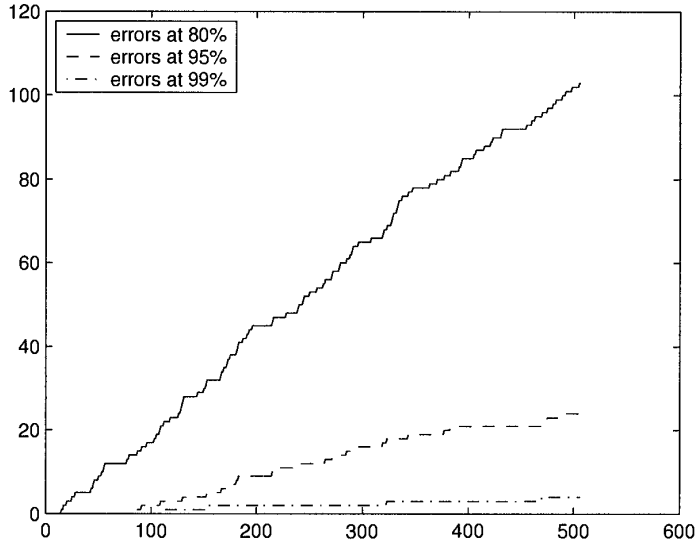

Fig. 2.4. The cumulative numbers of errors at the given confidence levels for RRCM run on-line on the randomly permuted Boston Housing data set

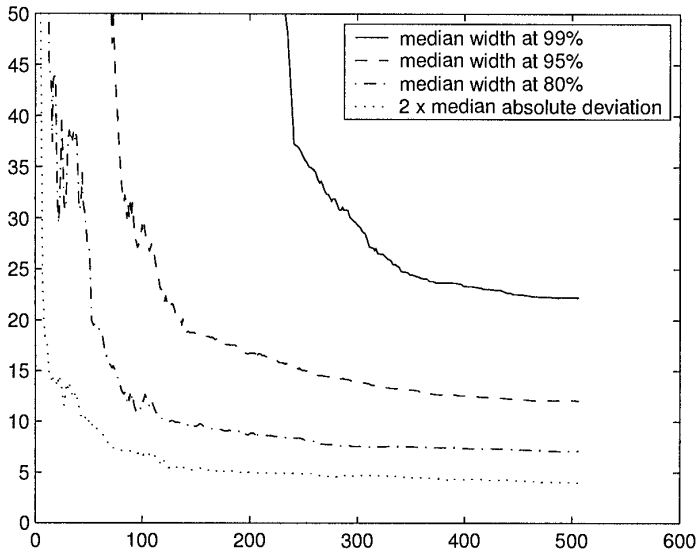

Fig. 2.5. The on-line performance of kernel RRCM on the randomly permuted Boston Housing data set

using the same format as Fig. 2.1.

The performance of the 1-NNR conformal predictor (as described in the preceding subsection) is shown, in the same format, in Fig. 2.6. The 1-NNR procedure performs reasonably well as a simple predictor (as the dotted line shows), but the prediction intervals it produces are much worse than those produced by more advanced methods.

2.4 Are there other ways to achieve validity?

In this section we will see that conformal predictors are essentially the only confidence predictors in a very natural class that satisfy our strong non-asymptotic property of validity.

Let us say that a confidence predictor is *invariant* if $\Gamma^\epsilon(z_1,\dots,z_{n-1},x_n)$ does not depend on the order in which $z_1,\dots,z_{n-1}$ are listed. Since we assume exchangeability, the invariant confidence predictors constitute a natural class (see, e.g., the description of the "sufficiency principle" in Cox and Hinkley 1974; later in this book, however, we will also study confidence predictors that are not invariant, such as inductive and Mondrian conformal predictors in Chap. 4).

If Γ_1 and Γ_2 are (deterministic) confidence predictors, we will say that Γ_1 is *at least as good as* Γ_2 if, for any n and any $\epsilon \in (0,1)$,

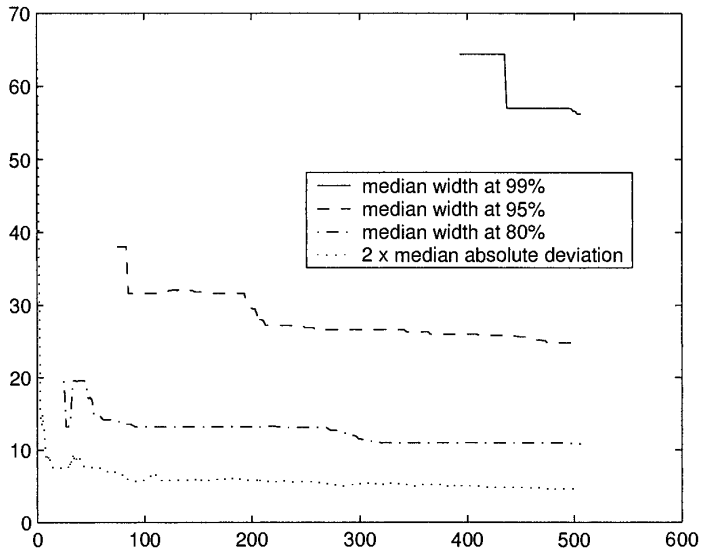

Fig. 2.6. The on-line performance of the 1-NNR conformal predictor on the randomly permuted Boston Housing data set

$$\Gamma_1^\epsilon(x_1,y_1,\dots,x_{n-1},y_{n-1},x_n) \subseteq \Gamma_2^\epsilon(x_1,y_1,\dots,x_{n-1},y_{n-1},x_n)$$

holds for almost all $x_1,y_1,x_2,y_2,\dots$ generated by any exchangeable distribution on $\mathbf{Z}^\infty$.

The following proposition asserts that invariant conservatively valid confidence predictors are conformal predictors or can be improved to become conformal predictors.

Theorem 2.6. *Suppose* $\mathbf{Z}$ *is a Borel space. Let* Γ *be an invariant conservatively valid confidence predictor. Then there is a conformal predictor that is at least as good as* Γ.

This proposition will be proved in §2.6 (p. 48). It is interesting that the proof will not use the fact that the random variables $\xi_1^{(\epsilon)},\xi_2^{(\epsilon)},\dots$ from the definition of Γ's conservative validity are independent. This observation leads to the following simple result, which we state in terms of randomized confidence predictors.

Proposition 2.7. *Suppose* $\mathbf{Z}$ *is Borel. If an invariant randomized confidence predictor* Γ *at each significance level* $\epsilon \in (0,1)$ *makes an error with probability* ϵ *at each trial and under any exchangeable distribution on* $\mathbf{Z}^\infty$*, then it makes errors at different trials independently, at each significance level* $\epsilon \in (0,1)$ *and under any exchangeable distribution on* $\mathbf{Z}^\infty$.

2.5 Conformal transducers

There are two convenient ways to represent a conformal predictor: as a confidence predictor and as a "transducer". So far we have been using the first way; the goal of this section is to introduce the second way, which is simpler mathematically, and to discuss connections between the two. This will be needed in, e.g., Chap. 7, and can be skipped for now.

A *randomized transducer* is a function f of the type $(\mathbf{X} \times [0,1] \times \mathbf{Y})^* \to [0,1]$. It is called "transducer" because it can be regarded as mapping each input sequence $(x_1, \tau_1, y_1, x_2, \tau_2, y_2, \dots)$ in $(\mathbf{X} \times [0,1] \times \mathbf{Y})^\infty$ into the output sequence $(p_1, p_2, \dots)$ of *p-values* defined by $p_n := f(x_1, \tau_1, y_1, \dots, x_n, \tau_n, y_n)$, $n = 1, 2, \dots$. We say that f is an *exactly valid randomized transducer* (or just *exact randomized transducer*) if the output p-values $p_1 p_2 \dots$ are always distributed according to the uniform distribution $\mathbf{U}^\infty$ on $[0,1]^\infty$, provided the input examples $z_n = (x_n, y_n)$, $n = 1, 2, \dots$, are generated by an exchangeable probability distribution on $\mathbf{Z}^\infty$ and the numbers $\tau_1, \tau_2, \dots$ are generated independently from the uniform distribution $\mathbf{U}$ on $[0,1]$.

We can extract exact randomized transducers from nonconformity measures: given a nonconformity measure A, for each sequence

$$(x_1, \tau_1, y_1, \dots, x_n, \tau_n, y_n) \in (\mathbf{X} \times [0,1] \times \mathbf{Y})^*$$

define

$$f(x_1, \tau_1, y_1, \dots, x_n, \tau_n, y_n) := \frac{|\{i : \alpha_i > \alpha_n\}| + \tau_n |\{i : \alpha_i = \alpha_n\}|}{n} , \quad (2.47)$$

where α_i, $i = 1, 2, \dots$, are computed from $z_i = (x_i, y_i)$ using A by the usual (cf. (2.19), p. 26) formula

$$\alpha_i := A_n(\lfloor z_1, \dots, z_{i-1}, z_{i+1}, \dots, z_n \rfloor, z_i) .$$

Each randomized transducer f that can be obtained in this way will be called a *smoothed conformal transducer*.

Proposition 2.8. *Each smoothed conformal transducer is an exact randomized transducer.*

This proposition is a special case of Theorem 8.1 (p. 193), which will be proved in §8.7.

In a similar way we can define (deterministic) *conformal transducers* f: given a nonconformity measure A, for each sequence $(z_1, \dots, z_n) \in \mathbf{Z}^*$ set

$$f(z_1, \dots, z_n) := \frac{|\{i : \alpha_i \geq \alpha_n\}|}{n} ,$$

where α_i are computed as before. In general, a (deterministic) *transducer* is a function f of the type $\mathbf{Z}^* \to [0,1]$; as before, we associate with f a mapping from $z_1 z_2 \dots$ to the *p-values* $p_1 p_2 \dots$ ($p_n := f(z_1, \dots, z_n)$). We say that f is a *conservatively valid transducer* (or *conservative transducer*) if there exists a probability space with two sequences ξ_n and η_n, $n = 1, 2, \dots$, of $[0,1]$-valued random variables such that:

- the sequence $\xi_1, \xi_2, \dots$ is distributed as $\mathbf{U}^\infty$;
- for each n, $\eta_n \leq \xi_n$;
- the joint distribution of the sequence of p-values produced by f coincides with the joint distribution of $\eta_1, \eta_2, \dots$, provided the examples $z_1, z_2, \dots$ are generated from an exchangeable distribution on $\mathbf{Z}^\infty$.

The following implication of Proposition 2.8 is obvious:

Corollary 2.9. *Each conformal transducer is conservative.*

We can fruitfully discuss confidence transducers even in the case of general example spaces $\mathbf{Z}$, not necessarily products $\mathbf{X} \times \mathbf{Y}$ of object and label spaces. But in the latter case we can associate a confidence predictor $\Gamma = f'$ with each confidence transducer f defining (2.17) (p. 26) as

$$\{y \in \mathbf{Y} : f(x_1, y_1, \dots, x_{n-1}, y_{n-1}, x_n, y) > \epsilon\} . \quad (2.48)$$

Vice versa, with any confidence predictor Γ we can associate the confidence transducer $f = \Gamma'$ defined by

$$f(x_1, y_1, \dots, x_n, y_n) := \sup \{\epsilon : y_n \in \Gamma^\epsilon(x_1, y_1, \dots, x_{n-1}, y_{n-1}, x_n)\} . \quad (2.49)$$

Letting x_i in (2.48) and (2.49) range over the extended object space $\mathbf{X} \times [0,1]$, we obtain the definition of the randomized confidence predictor f' associated with a randomized confidence transducer f and the definition of the randomized confidence transducer Γ' associated with a randomized confidence predictor Γ. The definition (2.49) of p-values associated with a confidence transducer agrees with the definitions given earlier (see (2.16) and the left-hand sides of (2.18) and (2.20)).

We will see in the next subsection that the expositions of the theory of hedged prediction in terms of conformal transducers and conformal predictors are essentially equivalent. But first we slightly strengthen Proposition 2.4.

A randomized confidence predictor is *strongly exact* if, for any exchangeable probability distribution on $\mathbf{Z}^\infty$ and any sequence $(\epsilon_1, \epsilon_2, \dots) \in (0,1)^\infty$ of significance levels, the sequence of random variables $\mathrm{err}_n^{\epsilon_n}(\Gamma, P)$, $n = 1, 2, \dots$, is distributed as the product $\mathbf{B}_{\epsilon_1} \times \mathbf{B}_{\epsilon_2} \times \cdots$ of Bernoulli distributions with parameters $\epsilon_1, \epsilon_2, \dots$. It can be defined in a similar way what it means for a confidence predictor Γ to be strongly conservative.

Theorem 8.1 will also imply the following proposition.

Proposition 2.10. *Any smoothed conformal predictor is strongly exact.*

Normalized confidence predictors and confidence transducers

To obtain full equivalence between confidence transducers and confidence predictors, a further natural restriction has to be imposed on the latter: they will be required to be "normalized". This is a mild restriction since each confidence predictor can be normalized in such a way that its quality does not suffer.

Formally, the *normal form* Γ_{norm} of a confidence predictor Γ is defined by

$$\Gamma^\epsilon_{\text{norm}}(x_1, y_1, \dots, x_n) := \bigcup_{\epsilon' > \epsilon} \Gamma^{\epsilon'}(x_1, y_1, \dots, x_n) \,. \tag{2.50}$$

We say that Γ is *normalized* if $\Gamma_{\text{norm}} = \Gamma$. These definitions are also applicable to randomized confidence predictors, in which case x_i range over the extended object space $\mathbf{X} \times [0,1]$. The following proposition lists some basic properties of the operation $_{\text{norm}}$ and normalized confidence predictors.

Proposition 2.11. *All conformal predictors and smoothed conformal predictors are normalized. For any confidence predictor Γ (randomized or deterministic):*

1. *Γ_{norm} is* at least as good as *Γ, in the sense that*

$$\Gamma^\epsilon_{\text{norm}}(x_1, y_1, \dots, x_n) \subseteq \Gamma^\epsilon(x_1, y_1, \dots, x_n)$$

   for all $x_1, y_1, \dots, x_n$ and ϵ;
2. *$(\Gamma_{\text{norm}})_{\text{norm}} = \Gamma_{\text{norm}}$;*
3. *Γ is normalized if and only if the set*

$$\{\epsilon : y_n \in \Gamma^\epsilon(x_1, y_1, \dots, x_n)\}$$

   is open in $(0,1)$ for all $x_1, y_1, \dots, x_n$;
4. *If Γ is exact (resp. conservative, resp. strongly exact), then Γ_{norm} is exact (resp. conservative, resp. strongly exact).*

Proof. All (smoothed) conformal predictors are normalized because all inequalities involving ϵ in (2.18) and (2.20) are strict. Properties 1 and 2 are obvious. Property 3 follow from the following restatement of the definition of Γ_{norm}:

$$y_n \in \Gamma^\epsilon_{\text{norm}}(x_1, y_1, \dots, x_n)$$

if and only if

$$\exists \epsilon' > \epsilon : y_n \in \Gamma^{\epsilon'}(x_1, y_1, \dots, x_n) \,.$$

Property 4 follows from

$$\text{err}^\epsilon_n(\Gamma_{\text{norm}}) = \inf_{\epsilon' > \epsilon} \text{err}^{\epsilon'}_n(\Gamma) \,. \qquad \square$$

The next proposition asserts the equivalence of confidence transducers and normalized confidence predictors (in particular, the equivalence of conformal transducers and conformal predictors). Remember that f' is the confidence predictor associated with a confidence transducer f and Γ' is the confidence transducer associated with a confidence predictor Γ.

Proposition 2.12. *For each confidence transducer f, the confidence predictor f' is normalized and $f'' = f$. For each normalized confidence predictor Γ, $\Gamma'' = \Gamma$. If $\Gamma = f'$ is the normalized confidence predictor associated with a confidence transducer $f = \Gamma'$,*

$$\text{err}^\epsilon_n(\Gamma, (z_1, z_2, \dots)) = \mathbb{I}_{f(z_1,\dots,z_n) \le \epsilon} \tag{2.51}$$

for any data sequence $z_1, z_2, \dots$. If a randomized confidence transducer f is exact, f' is strongly exact. If a randomized confidence predictor Γ is strongly exact, Γ' is exact. If f is a (smoothed) conformal transducer, f' is a (smoothed) conformal predictor. If Γ is a (smoothed) conformal predictor, Γ' is a (smoothed) conformal transducer.

Proof. Most of the statements in the proposition are obvious. Equality (2.51) follows from the definition of f'. This equality implies that the confidence transducer Γ' is exact for any strongly exact randomized confidence predictor Γ. Indeed, the p-values p_n output by Γ' have the uniform distribution $\mathbf{U}$ on $[0,1]$, and it remain to apply the following simple fact: if a sequence $p_1, p_2, \dots$ of random variables distributed as $\mathbf{U}$ is such that, for any sequence $(\epsilon_1, \epsilon_2, \dots) \in (0,1)^\infty$, the random variables $\mathbb{I}_{p_n \le \epsilon_n}$, $n = 1, 2, \dots$, are independent, then the random variables p_n themselves are independent (see, e.g., Shiryaev 1996, the theorem in §II.5, p. 179). $\square$

2.6 Proofs

Proof of Theorem 2.1

The proof will show that no confidence predictor satisfies even the property of being *weakly exact*, where the requirement that $\text{err}^\epsilon_n(\Gamma)$ be independent for different n is dropped and the exchangeability assumption is replaced by the randomness assumption. Moreover, we will see that even for a fixed $n \in \mathbb{N}$ it is impossible to have the probability of $\text{err}^\epsilon_n(\Gamma) = 1$ equal to ϵ for all $\epsilon \in (0,1)$.

We may assume that the examples $z_1, z_2, \dots$ are generated from a power distribution Q^∞ such that the probability distribution Q on $\mathbf{Z}$ is concentrated on the set $\{(x, y^{(1)}), (x, y^{(2)})\} \subseteq \mathbf{Z}$, for some arbitrarily fixed $x \in \mathbf{X}$ and $y^{(1)}, y^{(2)} \in \mathbf{Y}$ (remember that we assumed $|\mathbf{X}| \ge 1$ and $|\mathbf{Y}| > 1$). Therefore, we assume, without loss of generality, that $\mathbf{Z} = \{0,1\}$. Fix $n \in \mathbb{N}$ and suppose that $\text{err}^\epsilon_n(\Gamma) = 1$ with probability ϵ for all $\epsilon \in (0,1)$.

For each k let $f(k)$ be the probability that $\text{err}^\epsilon_n(\Gamma, (z_1, \dots, z_n)) = 1$ (we drop $z_{n+1}, z_{n+2}, \dots$ from our notation since err^ϵ_n does not depend on them)

where $(z_1, \dots, z_n) \in \{0,1\}^n$ is drawn from the uniform distribution on the set of all binary sequences of length n with k 1s. Since the expected value of $f(k)$ is ϵ w.r. to any binomial distribution on $\{0, 1, \dots, n\}$, the standard completeness result (see, e.g., Lehmann 1986, §4.3) implies that $f(k) = \epsilon$ for all $k = 0, 1, \dots, n$. Therefore, $\epsilon\binom{n}{k}$ is an integer for all k and ϵ, which cannot be true.

Proof of Theorem 2.6

Let $\epsilon \in (0,1)$, $n \in \mathbb{N}$, Q be a probability distribution on $\mathbf{Z}$, and $\eta_n^{(\epsilon)}$ and $\xi_n^{(\epsilon)}$ be the random variables from the definition of the conservative validity of Γ corresponding to the exchangeable probability distribution Q^∞. For each sequence of examples $(z_1, \dots, z_n) \in \mathbf{Z}^n$ let $f(z_1, \dots, z_n)$ be the conditional probability under Q^∞ that $\xi_n^{(\epsilon)} = 1$ given $\eta_i^{(\epsilon)} = \mathrm{err}_i^{(\epsilon)}(\Gamma, (z_1, z_2, \dots))$, $i = 1, \dots, n$. For each bag $B \in \mathbf{Z}^{(n)}$ let $f(B)$ be the arithmetic mean of $f(z_1, \dots, z_n)$ over all $n!$ orderings of B. We know that the expected value of $f(B)$ is ϵ under any Q^n, and this, by the completeness of the statistic that maps data sequences $(z_1, \dots, z_n)$ to bags $\{\!|z_1, \dots, z_n|\!\}$ (see Lehmann 1986, §4.3; since $\mathbf{Z}$ is Borel, it can as well be taken to be $\mathbb{R}$), implies that $f(B) = \epsilon$ for almost all (under any Q^∞) bags B. Let us only consider such bags. Define $S(B, \epsilon)$ as the bag of elements z of B such that Γ makes an error at significance level ϵ at trial n when fed with the elements of B ordered in such a way that the nth example is z (since Γ is invariant, whether an error is made depends only on which element is last, not on the ordering of the first $n-1$ elements). It is clear that

$$\epsilon_1 \le \epsilon_2 \Longrightarrow S(B, \epsilon_1) \subseteq S(B, \epsilon_2)$$

and

$$\frac{|S(B, \epsilon)|}{n} \le \epsilon \,.$$

Therefore, the conformal predictor determined by the conformity measure

$$A_n(B \setminus \{\!|z|\!\}, z) := \inf\{\epsilon : z \in S(B, \epsilon)\}$$

is at least as good as Γ.

Proof of Proposition 2.7

The proof is similar to the proof of the previous proposition but more complicated since it depends on the proof of Proposition 2.4. Let $\epsilon \in (0,1)$ and, for each $n = 1, 2, \dots$ and each bag $B \in \mathbf{Z}^{(n)}$, let $f(B)$ be the probability that $\mathrm{err}_n^\epsilon(\Gamma) = 1$ when Γ is supplied with the elements of B in a random order (each order having the same probability $1/n!$) and with random numbers $\tau_1, \dots, \tau_n$ distributed independently according to $\mathbf{U}$. Since the expected value of $f(B)$ is ϵ for any power distribution Q^∞ on $\mathbf{Z}^\infty$ generating the examples, the same completeness argument shows that $f(B) = \epsilon$ for almost all bags B. It remains to combine this with the invariance of Γ and the proof of Theorem 8.2 (generalization of Proposition 2.4) in §8.7.

2.7 Bibliographical and historical remarks

Conformal prediction

Conformal predictors were first described by Vovk et al. (June 1999) and Saunders et al. (1999). The independence of errors in the on-line mode was proved in Vovk 2002b.

The idea of a rudimentary conformal predictor based on Vapnik's support vector machine was described by Gammerman et al. (January 1997) and originated at the meeting (mentioned on p. XV) between Gammerman, Vapnik and Vovk in the summer of 1996. Having had worked for a long time on the algorithmic theory of randomness (the paper Vovk and V'yugin 1993 being most relevant), Vovk realized that Chervonenkis's old idea (dating from June 1966, according to Chervonenkis 2004) that a small number of support vectors translates into confident predictions (cf. (10.6)) can be used for making hedged predictions. Let us consider the problem of classification ($|\mathbf{Y}|$ is finite and small). From the point of view of the algorithmic theory of randomness, we can make a confident prediction for the label y_n of the new object x_n given a training set $z_1, \dots, z_{n-1}$ if the algorithmic randomness deficiency is small for only one possible extension $(z_1, \dots, z_{n-1}, (x_n, y))$, $y \in \mathbf{Y}$; we can then output the corresponding y as a confident prediction. (In the case of regression, $\mathbf{Y} = \mathbb{R}$, a confident prediction for x_n is possible if the algorithmic randomness deficiency is small for a narrow range of $y \in \mathbf{Y}$.)

Remark The reader who is not familiar with the algorithmic theory of randomness (which is not used in this book outside the end-of-chapter remarks) can consult Kolmogorov 1983, Martin-Löf 1966, V'yugin 1994, Li and Vitányi 1997. In the literature on algorithmic randomness the word "algorithmic" is often omitted, but we will always keep it, to avoid confusion with several other, unrelated, notions of randomness used in this book. (In particular, there are no obvious connections between algorithmic randomness and the assumption of randomness discussed in the previous chapter.) The algorithmic notion randomness formalizes the intuitive notion of typicalness: an object $\omega \in \Omega$ is regarded as typical of a probability distribution P on Ω (we will also say "under P" or "w.r. to P") if there is no reason to be surprised when told that ω was drawn randomly from P. Using the notion of a universal Turing machine, it is possible to introduce the notion of *algorithmic randomness deficiency*, formalizing the degree of deviation from typicalness. For further details, see p. 218.

There are two very different approaches to defining algorithmic randomness deficiency: Martin-Löf's (1966) and Levin's (1976, 1984; simplified in Gacs 1980 and Vovk and V'yugin 1993). Kolmogorov's (1968) original definition is a special case of Martin-Löf's, but becomes a special case of Levin's (as simplified in Vovk and V'yugin 1993) if the plain Kolmogorov complexity in it is replaced by prefix complexity. Martin-Löf's definition is more intuitive, being a universal version of the standard statistical notion of p-value, but Levin's definition often leads to more elegant mathematical results.

The paper by Gammerman et al. (1997) was based on Levin's definition, which made it difficult to understand. Conformal predictors, which appeared in Vovk et al. 1999 and Saunders et al. 1999, were the result of replacing Levin's definition of algorithmic randomness by Martin-Löf's definition in Gammerman et al. 1997.

After the notion of conformal predictor crystallized, the connection with the algorithmic theory of randomness started to disappear; in particular, in order to obtain the strongest possible results, we replaced the algorithmic notion of randomness with statistical tests. As we said earlier, in this book we hardly ever mention algorithmic theory of randomness outside the end-of-chapter remarks. This evolution does not look surprising: e.g., we argued in Vovk and Shafer 2003 that the algorithmic notions of randomness and complexity are powerful sources of intuition, but for stating mathematical results in their strongest and most elegant form it is often necessary to "translate" them into a non-algorithmic form.

For the information on the many precursors of conformal prediction, see §10.2; Kei Takeuchi's definition is especially close to ours.

A version (more sophisticated but less precise, involving arbitrary constants) of Theorem 2.6 was stated and proved in Nouretdinov et al. 2003. An analogous result was stated by Takeuchi for his version of conformal predictors.

Least squares and ridge regression

The least squares procedure was invented independently by Gauss and Legendre and first published by Legendre in 1805 (for details, see, e.g., Plackett 1972, Stigler 1981, 1986a). The term "hat matrix" was introduced by John W. Tukey (see Hoaglin and Welsch 1978). The ridge regression procedure was first described in detail by Arthur E. Hoerl and Robert W. Kennard (1970a, 1970b). The idea came from Hoerl's (1959) ridge analysis, a method of examining high-dimensional quadratic response surfaces (for details, see Hoerl 1985). The link between ridge analysis and ridge regression is provided by Hoerl's 1962 paper.

Deleted residuals are also known as PRESS and predicted residuals, and the nonconformity scores (2.36) differ from "internally studentized residuals" only by a factor that does not affect the conformal predictor's output. We did not consider "externally studentized residuals"; for details and history, see, e.g., Cook and Weisberg 1982 (§2.2.1).

The RRCM was developed by Ilia Nouretdinov and published in Nouretdinov et al. 2001a.

Kernel methods

Kernel methods have their origins in the Hilbert–Schmidt theory of integral equations (see Mercer 1909). The fundamental fact that each symmetric nonnegative definite function has representation (2.42) can be proved by many different methods: see, e.g., Mercer 1909 (that paper, however, proves a slightly different result, "Mercer's theorem", about continuous kernels and an integral analogue of condition (2.46)) and Aronszajn 1950 (Aronszajn's proof is based on Moore's idea; it is reproduced in Wahba 1990). For recent expositions, see, e.g., Schölkopf and Smola 2002; Cristianini and Shawe-Taylor 2000; Shawe-Taylor and Cristianini 2004.

There are several approaches to the kernel ridge regression; the three main ones appear to be the following:

- the approach adopted in this book: the objects are mapped to an arbitrary (not necessarily functional or separable) Hilbert space and the prediction rule is chosen from among the continuous linear functionals on that space; the main equation (2.41) can be obtained, for example, using the Lagrange method analogously to Vapnik's (1998) derivation of SVM (see Saunders et al. 1998); the approach based on the equality (2.39) is standard in statistics;
- the approach based on functional Hilbert spaces with bounded evaluation functionals (called *reproducing kernel Hilbert spaces*; the prediction rule is chosen from among the elements of such a space; see, e.g., Wahba 1990);
- the approach based on Gaussian processes: one assumes that the labels y_i are obtained from zero-mean normal random variables with covariances $\operatorname{cov}(y_i, y_j)$ defined in terms of $K(x_i, x_j)$; (2.41) can then be obtained as the expected value of the x's label. In geostatistics this approach is known as kriging; for further details, see Cressie 1993 and p. 273.

Formula (2.44), which we used for the fast updating of the inverse matrix in the kernel RRCM, may have been first explicitly given by Banachiewicz (1937a, 1937b); further references and history can be found in Henderson and Searle 1981. There are similar updating formulas (going back to Gauss 1823 and also reviewed in Henderson and Searle 1981) that could be used in the case of RRCM, but the need for speeding up computations is less pressing for RRCM since the matrix to be inverted is always of the constant size $p \times p$.