ROYAL HOLLOWAY UNIVERSITY OF LONDON

COMPUTER LEARNING RESEARCH CENTRE

DEPARTMENT OF COMPUTER SCIENCE

Small and Large Scale Probabilistic Classifiers with Guarantees of Validity

IVAN PETEJ

Supervisor:

Prof. Vladimir VOVK

June 10, 2018

A THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Declaration of Authorship

I hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Ivan PETEJ June 10, 2018

Abstract

This thesis addresses and expands research in probabilistic prediction with a particular emphasis on generating forecasts which are well calibrated. Chapter 1 describes standard techniques in machine learning and outlines two methods - conformal prediction and Venn prediction - both which serve as important building blocks for the remainder of the results in this thesis. Chapter 2 introduces the field of probabilistic machine learning and highlights some of the advantages and challenges of the methods developed to date. Chapter 3 proposes a new method of probabilistic prediction which is based on conformal prediction - a machine learning method for generating prediction sets that are guaranteed to have a specified coverage probability. The method is applied to the standard USPS data set with encouraging results. Chapter 4 focuses on the study of Venn prediction, concentrating on binary prediction problems. Venn predictors produce probability-type predictions for the labels of test objects which are guaranteed to be well calibrated under the standard assumption that the observations are generated independently from the same distribution. A new class of Venn predictors is introduced, called Venn–Abers predictors, which are based on the idea of isotonic regression. Promising empirical results are demonstrated both for Venn–Abers predictors and for their more computationally efficient simplified version. Chapter 5 studies theoretically and empirically a method of turning machine learning algorithms into probabilistic predictors that, as the Venn-Abers predictors described in the preceding chapter, automatically enjoy a property of validity (perfect calibration) but are computationally more efficient. The price to pay for perfect calibration is that these probabilistic predictors produce imprecise probabilities. When these imprecise probabilities are merged into precise probabilities, the resulting predictors, while losing the theoretical property of perfect calibration, are shown to be consistently more accurate than the existing methods in empirical studies.

Acknowledgements

I would like to thank Prof. Volodya Vovk for his invaluable help and guidance during the course of this thesis. I would also like to thank my wife Katie and my family for their immense support and encouragement.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	**A**rtificial **N**eural **N**etworks
CP	**C**onformal **P**rediction
CVAP	**C**ross **V**enn–**A**bers **P**redictors
CSD	**C**umulative **S**um **D**iagram
DIR	**D**irect **I**sotonic **R**egression
GCM	**G**reatest **C**onvex **M**inorant
GLM	**G**eneralised **L**inear **M**odel
IID	**I**ndependent **I**dentically **D**istributed
IVAP	**I**nductive **V**enn–**A**bers **P**redictors
LR	**L**ogistic **R**egression
MBL	**M**ean **B**rier **L**oss
NB	**N**aïve **B**ayes
NN	**N**eural **N**etworks
PAV	**P**air **A**djacent **V**iolators
PET	**P**robability **E**stimation **T**rees
RMSE	**R**oot **M**ean **S**quare **E**rror
SVM	**S**upport **V**ector **M**achines
SVA	**S**implified **V**enn–**A**bers
VA	**V**enn–**A**bers
VC	**V**apnik–**C**hervonenkis
VP	**V**enn **P**rediction

Chapter 1

Introduction

1.1 Notion of Machine Learning

The advances in computing technology over the past several decades have been truly staggering. The emergence of ever faster and increasingly efficient computer hardware, the rapid spread of electronic communications and increasing use of artificial intelligence have been some of the more notable developments recently integrated into our everyday lives.

Despite the impressive array of applications around us today which make use of this new technology and a recent emergence of computer devices which appear to exhibit a form of intelligent learning, the field of artificial intelligence is not new. The notion of intelligent, or ”thinking” machines has been proposed as early as 1950 by Alan Turing [77] who formalised the concept in what became famously known as the Turing Test. In this test, an intelligent machine is the one able to convince a human being by answering questions behind a closed screen that it is not a machine but a human instead.

Shortly following Turing’s work, Samuel [65] recognised that it would be challenging or potentially impossible to create a thinking machine which could pass Turing’s test by programming it to always follow a predefined set of instructions. Just as humans don’t always follow a set of instructions but rather learn from experience, Samuel suggested that in order to begin to resemble a human, computers should be able to do so too. He suggested that a key starting point in creating intelligent machines is an ability to program them in such a way that they learn and adapt with experience. This idea led to a whole new field of “machine learning” focused on developing computer algorithms which are able to learn and adapt. The rapid development of machine learning over the past 50 years coupled with advances in computer speed and technology has resulted in algorithms that can help us diagnose diseases, automatically drive a vehicle, detect fraud and automatically classify images, to name but a few.

In the most general setting, a machine learning algorithm operates by building a model of the world from a set of input observations in order to make predictions expressed as an output. A traditional way of classifying learning algorithms is by the way they learn. One of the most widely used machine learning methods is known as *supervised learning* [65]. In this setting, a learning algorithm is presented with a number of *examples*. Each example consists of two components: an *object* which contains certain *features* or *attributes* that describe it and the *label* which represents an outcome. The goal of a supervised learning algorithm is to learn how to predict a label when presented with a particular object. It does so by learning from a set of examples in which both objects and labels are given (known as the *training set*) in order to predict unknown labels for objects which did not form part of the training set (known as as the *test set*).

One relatively well known example of a supervised learning problem consists of recognition of hand written digits. Here the objects are digital images of hand-written single digits and the labels are numbers they represent. The learning algorithm attempts learn a rule which maps the input space of objects represented by digital images, into labels, in this case the actual numbers the images represent. Due to the fact that labels are discrete (numbers ranging from 0 to 9) this problem is referred to as *classification*. In some other supervised problems, where the labels are continuous rather than discrete, the problem is referred to as *regression* (an example would be house price prediction where the objects contain features of a house such as size, location, etc. and the labels are house prices).

The second type of learning is known as *unsupervised learning* [76]. Here the algorithm is presented with examples which contain only objects without labels. The goal of an unsupervised learning algorithm is to detect similarity of different examples and to group them accordingly. For instance, finding groups of individuals with similar shopping habits and separating them into clusters can be done by means of unsupervised learning.

The third traditional method of machine learning is known as *reinforcement learning* [74]. Here the algorithm learns on the basis of behaviour based on feedback from the environment. In contrast to supervised learning, where the algorithm tries to maximise some form of predictive accuracy, the goal of a reinforcement learning algorithm is to maximise a certain utility function over time. The algorithm does not need to be told which actions to take but rather discovers which actions yield the greatest reward by trial and error. This action may not just affect the immediate reward, but also all subsequent rewards. To date, successful reinforcement learning algorithms have demonstrated their ability to play games such as chess and Go, in some cases beating experienced human opponents.

One further important distinction in machine learning is in how frequently algorithms learn. In the traditional way of supervised learning described above, the algorithms learn from a training set. Learning is performed once and subsequently the algorithm is tested on yet unseen data - this is known as the *batch* mode of learning [76]. Other types of algorithms learn continuously, with each new example. This type of learning is known as *online* [9]. Rather than just learning once from a fixed batch of examples, data in the form of examples is presented to an online learning algorithm in sequential order and the algorithm aims to improve its accuracy at each step. By definition reinforcement learning is online, whereas the supervised and unsupervised learning algorithms can be either batch or online in nature.

The focus of this thesis will be on probabilistic prediction of supervised classification algorithms. The following sections will introduce the terminology and some commonly used supervised classification algorithms to date.

1.1.1 Terminology

As mentioned in the introduction above, supervised learning algorithms learn from past experience which are formalised in the form of examples. In the remainder of this thesis we will refer to a set of all possible examples as $\mathbf{Z}$. Each individual example consists of an object and the corresponding outcome or label. We refer to the set of all possible objects as

$\mathbf{X}$ and the corresponding labels as $\mathbf{Y}$ with the Cartesian product $\mathbf{Z} := \mathbf{X} \times \mathbf{Y}$ representing a set of all possible examples.

Whereas $\mathbf{Z}$ represents the set of examples, each individual example (often referred to as an *observation*) is represented by $z = (x, y)$ where x is generally multidimensional. The individual dimensions of objects represent particular *inputs* or *features* and the dimension of labels represent possible *outcomes*. In the case of recognition of hand-written digits described above, $\mathbf{X}$ (the space of possible objects) is represented by digital images measuring 16 by 16 pixels and 31 different shades of grey. Each object in this example consists of a vector of length $16 \times 16 = 256$ rows, each row containing a value representing one of 31 different possible shades of grey. This results in a total objects space $\mathbf{X}$ of $31^{16\times16}$ (approximately 10^{357}). It is not unusual for the object space to be significantly larger than this in practice. The space of labels $\mathbf{Y}$ is generally smaller, in this case it consists of a single number which can vary from 0 to 9. The goal of a machine learning algorithm therefore is to learn a mapping rule which can classify an object into one of nine different possible labels. This problem is therefore known as a *multiclass* classification problem. Sometimes, there are only two possible classes (example are classification of emails into spam/not spam and other similar true/false problems), in which case the problem is referred to as a *binary* classification problem.

In order for an algorithm to be able to learn, there generally needs to be some stability in the underlying environment that generates the data. The traditional way of describing a stable environment is to assume that individual examples are drawn at random from some fixed probability distribution, we refer to as Q, on the fixed example space $\mathbf{Z}$. The standard assumption is that we do not know the actual probability distribution Q from which each example is drawn, all we know is that the examples are drawn independently from it – they are independently and identically distributed *(i.i.d)*. This type of learning is known as *learning under unconstrained randomness*.

To date, machine learning has made significant advances in ability to learn in such an environment. One of the most significant theoretical advances came in the form of statistical learning theory developed by Vapnik and Chervonenkis in the late 1960s

[17]. The discovery of the Vapnik–Chervonenkis (VC) dimension, or a measure of capacity/complexity of a space of functions that can be learned by a statistical classification algorithm, set machine learning on a firm theoretical footing and resulted in the development of algorithms which today perform very well in practice. The theorem was, for the first time, able to provide an upper bound for the probability of the error a machine learning algorithm will experience on the test set as a function of the size of the training set and the complexity of the algorithm (its VC dimension) under the *i.i.d* assumption. The following section will describe popular supervised learning algorithms to date, some of which will be used in the remainder of this thesis.

1.1.2 Supervised Learning Algorithms

One of the very first and most basic supervised machine learning algorithms, developed by Rosenblatt [63] in the 1950s, is called the *perceptron*. Based on the idea of linear regression, the perceptron is designed for binary classification problems. The perceptron is a function which maps the input space **x** into an output $h(\mathbf{x})$ such that:

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x}) \tag{1.1}$$

where **w** is a vector of real-valued weights. In other words a set of objects **X**, with each individual object **x** represented by an m dimensional vector (with m representing the number of input features or attributes) is multiplied by a corresponding m dimensional weight vector **w** in order to obtain the function $h(\mathbf{x})$ which can be either 0 or 1, depending on the sign of the value of (1.1) and which serves as the algorithms' prediction for the binary label. The goal of the perceptron algorithm is to learn a set of weights which serve to emphasize or de-emphasize the influence of the individual input variables and which result in $h(\mathbf{x})$ that approximates the actual binary labels **y** (represented by 0 or 1) as close as possible. The perceptron algorithm is a linear separation algorithm - if the set of examples from which the perceptron algorithm learns (the training set) is linearly separable then it can be shown that the perceptron algorithm eventually converges to a solution. On the contrary, if the data is not linearly separable then the perceptron is unable to converge, although it can still provide an estimate for **y**.

Despite its relative simplicity, the perceptron algorithm was successful in solving some simple image classification tasks and remains in use until today. More importantly, the perceptron also formed a basic building block for one of the more successful machine learning methods developed since, called Artificial Neural Networks (ANN) [51] [56]. ANN consist of interconnected group of nodes, each node represented by a perceptron, also known as the neuron. In part inspired by the network of neurons in a brain, a neural network consists of multiple layers of perceptrons, with perceptron outputs from previous layers forming inputs to the next. In general, an ANN is defined by the choice of interconnections between different layers of perceptions/neurons, the weights of the interconnections (which are updated during the learning process) and the individual activation functions which convert a neuron weighted input into its output activation. ANN learns by optimising over a possible set of individual neuron weights and activation functions which results in the minimum error between predicted and true classes given a set of objects in the training set. A key advance in the development of ANN came with the discovery of the back-propagation algorithm [64], which allowed for a far greater speed of training and desired accuracy. The advantage of ANN over simple perceptrons is that they can be trained to solve problems which are far more complex and do not need to be linearly separable. As a result, they have been successfully applied to a variety of tasks such as computer vision and speech recognition, to name but a few.

The simple perceptron which forms the basic building block for ANN algorithms utilises regression techniques in order to derive a solution. One further algorithm with a similar underlying principle is based on the idea of logistic regression [22]. In contrast to the perceptron, the logistic regression uses a non-linear rather than a linear function of mapping inputs to labels. In particular, a logistic regression algorithm tries to find the set of weights **w** such that:

$$h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x}) \tag{1.2}$$

yields a likelihood for the class label to equal to 1, where $\theta(s) = \frac{e^s}{1+e^s}$. Generally, the threshold between the two classes is set at $h(\mathbf{x}) = 0.5$. One of the key advantages of logistic regression is that it allows for modelling of non-linear relationships between inputs and outcomes. Logistic regression has successfully been applied to date to a variety of machine learning problems such as diagnostics in medicine, economics and social sciences.

Both the linear regression which forms the basis for the perceptron method of classification and the logistic regression are special cases of broader family of models known as Generalised Linear Models (GLM) [54] which have their roots in statistics. These models all use regression but allow for response variables that have error distribution models other than a normal distribution. In a generalized linear model (GLM), each outcome $\mathbf{y}$ of the set of dependent variable is assumed to be generated from a particular distribution in the exponential family, a large range of probability distributions that includes the normal, binomial, Poisson and gamma distributions. The use of perceptrons within GLM falls under the assumption of a normal distribution whereas logistic regression falls under the binomial distribution family.

A very powerful supervised learning algorithm, known as Support Vector Machine (SVM), was developed by Vladimir Vapnik [21] in the 1990s. The particular advantage of SVM over standard linear classifiers is in the way that it learns. This difference can be illustrated by considering a simple linearly separable binary classification problem. In contrast to the perceptron, which during the learning process can converge at *any* solution which separates examples belonging one of two classes, the support vector machine framework searches for an *optimal* separation instead. This solution is said to maximise the margin between two classes by maximising the distance between points belonging to different classes which lie closest to the separating hyperplane known as *support vectors*. In doing so, the SVM reaches a solution to the binary classification problem by utilising what is known as *structural* risk minimisation, in contrast to *empirical* risk minimisation as is the case for a number of other standard machine learning algorithms. Furthermore, SVM can be used for mutidimensional learning problems which are not linearly separable through use of *kernels* which transform the input space into the one which is linearly separable. This relative power and flexibility of SVM has resulted in their widespread use in solving some of the more challenging machine learning problems since their discovery in the 1990s.

Another area of machine learning utilises *decision trees* [12] as a supervising learning method. Decision trees work on the basis of passing a set of input variables through a sequence of internal *nodes* which test individual inputs and branch into a set of *leaves* which eventually lead to a prediction for the output variable. Each individual internal node is labeled with a test for one specific input variable and each subsequent branch of an individual node corresponds to one possible value of the given input variable. The leaf nodes of a decision tree lead to a given class (or a real value in case of regression). Despite their relative simplicity, decision trees can be successfully trained utilising an *information gain* (closely related to the concept of entropy from information theory) in order to decide which feature to split on at each step in building the tree. A number of different decision tree optimisation techniques have been developed to date [61] which helped decision tees become a relatively versatile technique able to cope with some challenging problems. They are popular due to their transparency, ability to deal with variety of types of input variables and their comparative computational efficiency.

A further subset of machine learning is based on Bayes' Theorem [3]. One of the most popular, termed *naïve Bayes*, assumes an independence of input features to derive the probability of a given class label. Amongst some of the simplest machine learning techniques, naïve Bayes algorithms have been successfully used in a variety of supervised learning problems to date, as an example in email text categorisation.

One of the advantages naïve Bayes technique is that it's non-parametric, i.e. it's use does not require any input parameters. Another largely non-parametric technique used for classification is known as the *k-nearest neighbours algorithm* (k-NN) [19]. Here the input consists of the k closest training examples in the feature space. The output is a class membership with classification performed by a majority vote of its neighbours. Each example is assigned the class most common among its k nearest neighbours (where k is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbour.

The list of algorithms provided above is by no means exhaustive. Over the past few years the field of machine learning is rapidly evolving and novel techniques are being continually developed. The methods introduced above form a useful basis for the main results developed during the course of this thesis.

1.2 Probabilistic Machine Learning

Despite their widespread use, many supervised machine learning classification algorithms only provide discrete class labels with no probabilistic output associated with such labels. In many real world problems, it is desirable to have probabilistic answers instead of discrete case answers. For example, in medical applications, it may be preferable to make a decision based on probabilistic predictions of the patients state, rather than be given a discrete chosen action for each patient. More formally we are interested in the conditional probability:

$$Q_{\mathbf{Y}|\mathbf{X}}(y \mid x) \tag{1.3}$$

for $\mathbf{Z} := \mathbf{X} \times \mathbf{Y}$, $y \in \mathbf{Y}$ and $x \in \mathbf{X}$. In addition, it may be desirable to know not only the predicted class label but the degree of confidence in the prediction. The following section will introduce an important theoretical result which states that generating true conditional probabilities for class labels is in fact impossible under an assumption of unconstrained randomness unless test objects are identical repetitions of those observed in the training set. We will also introduce a recently developed technique which allows for estimation of probabilities which are not true conditional probabilities but are "well-calibrated" under the same theoretical assumptions.

1.2.1 Probabilistic learning under unconstrained randomness

When predicting probabilities for class labels one can distinguish between the case of asymptotic results (when the data set is assumed infinite), or more practical cases which deal with the finite set of objects and labels. In one of the earlier works Stone [72] showed that a predictor based on nearest neighbours under binary classification (whose probabilistic prediction is the fraction of objects classified as 1 amongst the k nearest neighbours of a test object) is *universally consistent* meaning that the difference between the predicted and true conditional probabilities converges to zero under the assumption of unconstrained randomness. Although this result is important in a theoretical sense, practically it is less useful as this convergence is not uniform – in other words it is true under an asymptotic case when the number of examples tends to infinity.

In practice, we are dealing with finite data sets and we would like to estimate the true conditional probability of a new object based on our finite world experience. This problem has been described in significant detail by Vovk, Gammerman and Shafer in [84]. In particular, the authors show (see e.g. Chapter 5, Theorem 5.2) that any non trivial (not empty and not containing 0 or 1) prediction interval for the conditional probability is impossible under the assumption of unconstrained randomness for finite data sets where the data sets are *diverse* (i.e. those data sets in which the objects are not precise repetitions of each other). Given that many of real life machine learning problems contain diverse data sets, we are faced with a situation that the best we can do is to generate probabilities which are as close to true values, but never perfect. In order to partially overcome this theoretical limitation, Vovk, Gammerman and Shafer [84] describe two significant novel machine learning methods:

- *Conformal prediction* - with which it is possible to not just output a given label but assign a *confidence* in that prediction
- *Venn prediction* - a framework for *multiprobability* prediction which results in "well-calibrated" or valid probabilities that perform well in finite data sets

A valid or "well-calibrated" probabilistic predictor is one which gets the probabilities approximately right on average. Each of the frameworks is described further below.

1.2.2 Conformal prediction

In the traditional supervised machine learning setting we are given a set of objects with the goal of predicting labels. Conformal prediction expands on the traditional methods by the ability to make "hedged" predictions - in other words, the ability to provide a degree of confidence in the prediction as well as the prediction itself. The conformal framework measures the degree of similarity of a new object to the existing examples in the training set – the degree of similarity (known as the "conformity measure") is then used to assign the confidence in the prediction for a given label. More specifically, instead of producing a single element of $\mathbf{Y}$ as the predicted output, a conformal predictor gives a range of more or less precise predictions, each with a certain degree of confidence. The algorithm which predicts in this manner does so at a specified *significance level* $\epsilon \in (0, 1)$, with a degree of *confidence* $1 - \epsilon$. Given a set of *n* examples and a significance level ϵ:

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

the conformal predictor Γ outputs a subset

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

of **Y**. Γ must satisfy

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

whenever $\epsilon_1 \geqslant \epsilon_2$, *i.e.* the larger the subset of possible labels, the more confident the prediction is. The main assumption behind conformal predictors is that the set of examples $(z_1, z_2, \ldots, z_n)$ are generated independently from some fixed probability distribution P.

The two main criteria in assessing conformal predictions are:

- *Validity* - which means that the frequency of errors which a conformal predictor makes does not exceed ϵ at each chosen confidence interval $1 - \epsilon$,
- *Predictive efficiency* - which implies that the prediction sets output by conformal predictors should be as small as possible.

It can be shown (see [84], Chapter 2) that conformal predictors are *conservatively valid* under the exchangeability assumption, meaning that the frequency of errors is asymptotically no greater than ϵ with probability of one.

Other methods of producing predictions with a measure on confidence rely on either statistical learning theory [80] (which allows us to estimate with respect to some confidence level the upper bound on the probability of error in our prediction) or methods which use Bayes' theorem [3] in order to estimate the distribution of labels for a given object (Bernardo and Smith [7]). The drawback of the former method is that for most practical problems the confidence intervals are too large to tell us anything useful (see Vovk, Gammerman and Shafer [84], p. 249), whereas the latter methods require a prior distribution to be specified, which if not correct, leads to predictions which are not guaranteed to be valid (see Melluish et al. [50]).

To date the conformal prediction framework has been successfully applied to a variety of important problems. Examples include medicine (see Gammerman et al. [32], Bellotti et al. [4], Papadopolous et al. [58]), active learning (Ho and Wechsler [41]) and change detection in data streams (Ho and Wechsler [40]). A further direction explored more recently (developed as part of this thesis) describes a framework for calibration of conformal predictors which can yield conditional probabilities (Vovk et al. [87]). This work has the potential to further the application of this robust and non-parametric technique to problems involving probabilistic class membership estimation.

1.2.3 Venn prediction

Venn predictors (VP), described in detail in [84] are multiprobability predictors which yield a separate probability distribution for each class label given a test object which, when combined, can lead to well-calibrated probabilities. The set of Venn prediction outputs can be summarized by lower and upper bounds for the conditional probability of the new example belonging to each one of the possible classes.

Venn predictors work by dividing the examples in the training set into categories, classifying the test object into one of the categories, and then using the frequencies of labels in the chosen category as probabilities for the new object's label. In computing the frequencies of labels in the category containing the test object, the test object is added to the training set examples already in that category. Since at the time of prediction we do not yet know the new object's label, the frequencies are computed several times, once for each label the new object might have. Each set of frequencies is interpreted as a probability distribution for the new object's unknown label. Venn Predictors therefore produce several probability distributions for the new label rather than a single one. It can be shown (see Vovk, Gammerman and Shafer [84], Chapter 6) that such resulting probability distributions are guaranteed to contain well-calibrated probabilities (up to statistical fluctuations) in a non-asymptotic sense, providing examples are generated randomly from some fixed unknown probability distribution.

As is the case with conformal predictors, the main desiderata for Venn predictors are validity, predictive efficiency and computational efficiency. To date, the VP framework has been successfully combined with the k-nearest neighbours algorithm (Dashevski and Luo [25]) as well as support vector machines (Zhou *et al.* [100]) and neural networks (Papadopolous [57]) leading to promising new methods in predicting class-membership probabilities.

1.3 Main contributions

The aim of this thesis is to develop further methods for obtaining class probability estimates from classifiers which are well calibrated while retaining good computational and informational efficiency. Chapter 2 which follows will describe a range of currently available methods, both parametric and non-parametric. It will outline advantages as well as current limitations of each. The remainder of this thesis will then focus on novel methods of probabilistic classification based on conformal and Venn prediction. These novel methods offer improvements relative to other probabilistic classification techniques to date.

The following list summarises the original results that were obtained during the course of work on this thesis with the corresponding chapters for these findings given following each bullet point.

- Chapter 3 describes a novel method of probabilistic prediction which consists of calibration of p-values into probabilities using a novel criteria of efficiency of conformal prediction.

- Chapter 4 focuses on the study of Venn prediction. Probabilistic predictions produced by Venn predictors are guaranteed to be well calibrated under the standard assumption that the observations are generated independently from the same distribution. A new class of Venn predictors is introduced called Venn–Abers predictors, based on the idea of isotonic regression.

- Chapter 5 develops two further versions of Venn-Abers predictors that automatically enjoy a property of validity but are computationally more efficient and demonstrate promising empirical results.

1.4 Publications

The following is a list of publications submitted as part of the work on this thesis:

- a journal paper by Vovk, Petej and Fedorova [87] describing a method of converting p-values into probabilistic predictors (also available in *arXiv* format at [88]).
- a conference paper by Vovk and Petej [86] introducing Venn-Abers predictors (also available in *arXiv* format at [85]).
- a journal paper by Vovk, Petej and Fedorova [89] introducing a computationally more efficient version of Venn-Abers predictors with applications to large data-sets (also available in *arXiv* format at [90]).
- a journal paper by Vovk, Nouretdinov, Fedorova, Petej and Gammerman [93] which describes optimal conformity measures for various criteria of efficiency of set-valued classification as well as probabilistic criteria of efficiency under the binary classification setting

1.5 Thesis summary

The remainder of this thesis is organised as follows: Chapter 2 introduces the field of probabilistic machine learning and highlights some of the advantages and challenges of the methods developed to date. Chapter 3 describes a new method of probabilistic prediction based on conformal prediction with promising empirical results on the USPS data set. Chapter 4 introduces Venn-Abers predictors (VAPs) and demonstrates their theoretical ability of producing probabilistic forecasts which can be applied on top of standard machine learning algorithms resulting in a probabilistic prediction which is well calibrated under an exchangeability assumption. The method is tested on standard datasets and promising results are demonstrated in comparison to existing methods of calibrating probabilities. Chapter 5 extends on this work and describes two further versions of Venn-Abers predictors, namely Inductive Venn-Abers Predictors (IVAPs) and Cross Venn-Abers Predictors (CVAPs) that also enjoy theoretical guarantees of validity (i.e. they are well-calibrated) however with far greater computational efficiency than the standard

Venn-Abers predictors. Their advantage is demonstrated with further empirical results. Chapter 6 summarises the main findings of the thesis and offers scope for future work. There are also two separate appendices: Appendix A describes used data sets and Appendix B presents a Matlab implementation of Venn-Abers prediction (VAP) and their computationally more efficient version, IVAP and CVAP.

Chapter 2

Literature review

This chapter introduces the field of probabilistic machine learning and highlights some of the advantages and challenges of the methods developed to date. The aim is to summarise the standard approaches, both parametric and non-parametric which will serve as a basis of comparison for the methods developed in the remainder of this thesis.

2.1 Introduction

The standard methods of classification can be divided into two types - statistical classification and machine learning - which have their origins in statistics and computer science, respectively. Classification outputs provided by statistical classification and some machine learning algorithms are generated in the form of membership probabilities which, in addition to assigning an example to a particular class, also reflect the probabilistic confidence that an observation belongs to that particular class. Other machine learning algorithms, such as Support Vector Machines (SVM) [80] or Artificial Neural Networks (ANN) [56] only generate normalised scores and are known as "scoring classifiers". In contrast to membership probabilities, these scores do not correctly reflect assessment uncertainty. For such learners, classifier scores need to be transformed into suitable class membership probabilities and this is generally achieved through use of one or more suitable calibration methods.

Due to the popularity and relative success SVM and ANN for classification, a significant body of literature to date has been devoted to development of reliable calibration methods for those and other scoring classifier algorithms. Section 2.2 reviews the main results in this field to date. Section 2.3 describes statistical classifiers and some machine learning algorithms which classify by means of membership probabilities. We will show that research to date suggests that even though such algorithms can be used directly to derive probabilistic predictions, they often yield inappropriate estimates which may need further calibration.

2.2 Calibration of scoring classifier algorithms

There are several existing approaches of obtaining class probability estimates from scoring classifiers. They can broadly be summarised into four main types: calibration via Mapping, Bayes' rule, calibration via Assignment values and Bagging. Each of the methods is briefly summarised below.

2.2.1 Mapping

One of the most widely used methods for mapping of scores to membership probabilities for SVM in a binary classification setting (where classes $\mathbf{Y} \in \{0, 1\}$) is based on a logistic regression calibration approach. Here, the aim is to model the log odds $g(x)$ represented by the conditional probabilities:

$$g(x) = \log \left[\frac{P(y = 1 \mid x)}{P(y = 0 \mid x)} \right] \tag{2.1}$$

using the standard notation introduced in Chapter 1 with y representing a given class and x representing the corresponding object. Using the complement $P(y = 1 \mid x) = 1 - P(y = 0 \mid x)$ leads to the term for deriving the calibrated conditional probability:

$$P(y = 1 \mid x) = \frac{e^{g(x)}}{1 + e^{g(x)}} = \frac{1}{1 + e^{-g(x)}} \tag{2.2}$$

In one of the earliest works Platt [59] applied a linear function such that $g(x) = Ax + B$ with scalar parameters A and B. By using this function, the calibration is fitted with a sigmoidal shape. The particular parametric form of this model was inspired by observing the empirical data - the relationship between SVM scores and the empirical probabilities appears to be sigmoidal for many datasets (for example, the Adult dataset from the UCI Repository [31]).

The search for the mapping function g is done via optimisation. The parameters A and B are derived in sample such that the negative log-likelihood of the data is minimized. Platt has shown empirically that this method yields probability estimates that are at least as accurate as ones obtained by training an SVM specifically for producing accurate class membership probability estimates (see Vapnik [80]), while being faster.

Zhang [98] extended Platt's method by using piecewise logistic regression instead of the full logistic regression, building on the idea of Bennett [5]. In contrast to Platt's model the log odds are not regarded as a linear function of membership values (i.e. $Ax + B$), but as a piecewise linear function with four different knots. Using four knots leads to a separation of membership values into the three areas - 1) obvious decision for negative class, 2) hard to classify and 3) obvious decision for positive class. In each of these three areas the log odds are fitted separately and independently as a linear function. The authors evaluated piecewise logistic regression calibration and standard logistic regression calibration over standard text categorization collections with three classifiers (SVM, naïve Bayes and logistic regression classifier), and observed that piecewise logistic regression performs significantly better than the full logistic regression method in the log-loss metric.

While the parametric methods above have been shown to be relatively successful in application to a wide range of datasets, there is no guarantee that such logistic regression calibration techniques are always valid. Motivated by this fact, Zadrozny and Elkan [97] proposed a non-parametric method for obtaining calibrated two-class probability estimates based on isotonic regression [2] which can be applied to any classifier that produces a ranking of examples. Assuming that the classifier ranks examples correctly and the mapping from scores into probabilities is non-decreasing, they apply a commonly used method for computing isotonic regression based on the pair-adjacent violators algorithm (PAVA) (see Ayer *et.al* [2]) to convert SVM scores into probabilities. This algorithm finds the stepwise-constant isotonic function that best fits the data according to a mean-square error criterion. The isotonic regression technique has been shown to exhibit good performance characteristics for a range of binary class datasets. Furthermore, the authors applied the technique to multiclass problems, by first separating the problem into a number of binary problems, calibrating the scores from each binary classifier using PAVA and combining them to obtain multiclass probabilities, with some promising results.

An alternative multi class calibration method was studied by Gebel and Weihs [34]. They propose a direct multi-class calibration procedure for margin based classifiers which combines the binary outcomes to assessment probabilities of multiple classes in a single step. The method is based upon the Dirichlet distribution and was shown to yield some competitive results, especially for data sets with balanced class distributions. Additional work in directly estimating class membership probability for any class in multiclass classification without decomposition and combination was done by Takahashi [75].

A further method of calibrating classifier scores into class membership for the binary case was proposed by Langord and Zadrozny [46]. The method is based upon a technique named the *Probing reduction* which is a general method for converting any classifier learner into a probability estimator. The method relies on the observation that probability estimates (in a Bayesian sense) can be extracted from preferences over bets at different odds ratios. Their proposed algorithm is shown to satisfy certain strong optimality guarantees - good performance with respect to classification implies good performance with respect to class probability estimation. The authors tested the probing algorithm on several datasets with several classifier learning algorithms and found strong performance compared to other common methods for obtaining class membership probability estimates, such as bagging in decision trees and calibration using Platt's sigmoid function for SVM.

An alternative method of mapping SVM classifier scores into class membership probabilities, which is closely related to concepts in quantum detection theory, was proposed by Crammer and Globerson [24]. In contrast to the standard SVM interpretation, which assumes that inputs can be transformed into a higher dimensional space such that positive and negative points correspond to different classes, in this work the authors present a different view of class separation, which incorporates both the concepts of margin maximization and probabilistic modelling. The approach assumes that classes instead correspond to orthogonal linear subspaces in feature space. This assumption can be used in many domains where the existence or absence of a feature is the key predictor of its class identity, rather than its exact value or its relation to values of other features. For example, in document classification there may be subsets of words (or linear combinations of word counts) whose appearance indicates the document topic. In image classification, a set of pixels may be indicative of image content regardless of their exact intensity ratios. An alternative statement of the problem is that there exists a linear transformation of feature space such that a unique subset of coordinates is active in each class. In order to measure the degree to which a given input point belongs to a given subspace the authors use a projection operator which measures what fraction of the point's norm lies in a subspace. The outputs of the projection operators have a natural interpretation as probabilities, and these probabilities are linear functions of the model parameters (the projection matrices). The authors compare the performance of their proposed method to the closely related second order kernels SVM, and show that it achieves improved performance on a handwritten digit classification task, while providing meaningful probabilistic outputs.

One of the more recent works proposes a novel method of combining multi class support vector machines with linear ensemble methods to yield class posterior probabilities (Guermeur [37]). The inspiration for the study is based upon the works of Breiman [11] which deals with multivariate regression. The method requires the normalisation of SVM outputs so that their outputs are non-negative and sum to one which is obtained by applying a polytomous logistic regression. The normalised outputs are then combined with linear ensemble methods in which the loss function is chosen such that the overall outputs are class posterior probability estimates.

2.2.2 Calibration via Bayes' rule

In contrast to the previously described calibration methods whose aim is to directly map membership values to calibrated probabilities, Bayesian methods can be applicable for a calibration of *unnormalized* scores instead. In contrast to many direct mapping approaches, the Bayesian method uses two steps to calculate membership probabilities. At first, the positive class scores are split into two groups according to their true class, so that probabilities $P(s_+ \mid y)$ (s_+ is a positive class score) for the score given a particular class $y \in \{-1, +1\}$ can be derived.

The second step involves determination of class membership probabilities by application of Bayes' Theorem to class conditional probabilities and class priors. While class priors can easily be estimated from the training set (i.e. by measuring the proportion of positively classified examples), the crucial point in this way of calibration is the choice of the distribution type for the class conditional probability $P(s_+ \mid y)$. Two different approaches are presented by Bennett [5], the standard assumption of a Gaussian distribution and a further idea using the asymmetric Gaussian and asymmetric Laplace distributions. According to Bennett it is more justifiable to use an asymmetric distribution for class conditional probabilities than a symmetric one. The study suggests that scores have a different distributional behaviour in the area between the modes of the two distributions compared to the respective other side. The area between the modes contains the scores of those observations which are difficult to classify, while the respective other portions stand for the observations for which classification is easier. This conclusion leads to the separation of scores into the three areas 1) obvious decision for the negative class, 2) hard to classify and 3) obvious decision for the positive class. The authors analysed the experimental performance of these models over the outputs of two text classifiers. The analysis demonstrates that the asymmetric Laplace model is theoretically attractive (introducing few new parameters while increasing flexibility) while computationally efficient.

2.2.3 Calibration via Assignment values

Calibration methods described based on Bayes' rule consist of partitioning and separate calibration and the partitioning forms the basis for the following independent determination of calibrated membership probabilities. While the calibration method by Bennett [5] partitions the unnormalized scores for a chosen class according to their true class, Garczarek [33] partitions the membership values according to their *assignment* instead. The idea of this method is to model the membership values for the assigned classes in each partition separately as Beta distributed random variables. The calibration procedure transforms these normalized membership values for each partition to new Beta random variables with optimal parameters and regards them as membership probabilities.

2.2.4 Binning

One of the shared characteristics of the techniques mentioned above is that, except for the isotonic regression approach of Zadrozny and Elkan [97], all other approaches rely on parametric assumptions. One further non-parametric approach which has been applied to date is by means of *binning* (see Zadrozny and Elkan [96] and Drish [30]). In binning, the training examples are sorted according to their scores and the sorted set is divided into a number of subsets of equal size, called bins. For each bin it is possible to compute lower and upper boundary scores. Any test example can then be placed it in a bin according to its score. The corrected probability that a sample belongs to a given class is the fraction of training examples in the bin that actually belong to that class. A difficulty of the binning method is that the number of bins is often chosen by cross-validation. If the dataset is small, or highly unbalanced, cross-validation is not likely to indicate the optimal number of bins. Also, the size of the bins is fixed and the position of the boundaries is chosen arbitrarily. If the boundaries are such that we average together the labels of examples that clearly should have different probability estimates, the binning method will fail to produce accurate probability estimates. As such, despite the fact that the method is not-parametric in nature, its success highly depends on the individual characteristics of the studied data set.

2.3 Calibration of probabilistic classifier algorithms

As described in the introductory section, there are a number of well-known classifying machine learning techniques which produce a probabilistic confidence estimate for a given class (examples include decision trees, logistic regression, naïve Bayes classification and random forests). However, for a large number of such algorithms, there are no guarantees the probabilistic outputs they produce are well calibrated (see Cohen and Goldszmitd [18]). A number of authors studied suitable methods of recalibration of these outputs, in order to improve their accuracy and quality.

In one of the frequently cited works, Zadrozny and Elkan [96] compared a number of different calibration techniques (a total of 10 different methods) on a large and challenging dataset, using naïve Bayes and decision trees as classifying algorithms. Their findings suggest that binning can be used to successfully improve naïve Bayesian probability estimates, while a technique called *smoothing* is useful in calibrating the probabilities associated with decision tree outputs. In particular, they propose a new technique, which they refer to as *curtailment*, which builds on the method of smoothing used by Provost and Domingos [60], in which they aim to avoid the problem of overfitting by ceasing a search in the decision tree as soon as a node is reached that has less than v examples, where v is a parameter of the method. The major reported advantage of this method is that it produces relatively small and hence understandable decision trees, while still giving high-resolution, well-calibrated probability estimates.

An alternative approach of improving class membership estimates for decision tress involves the use of a technique called *bagging*. Bagging basically aggregates decision tree predictions (by voting or averaging) from classifiers learned on multiple bootstraps of data. It has been shown to improve accuracy (Breiman [10]) and even for large and unbalanced datasets, lead to an improvement in the quality of probabilistic estimates (Chawla and Cieslak [16]). Other techniques which can improve the performance of decision tree classifiers involve *boosting* (Niculescu-Mizil and Caruana [55]), which varies the significance of individual training examples according to how difficult they are to classify and *randomisation* (Dietterich [29]), which randomises the internal decisions of the learning algorithm. Bagging, boosting and randomisation are examples of what are known as *ensemble learning methods*.

Other similar more recent works on calibration of decision tree outputs involve the study by Alvarez *et.al.* [1]. Their method provides smooth class probability estimates, without any modification of the tree when the data consists of attributes which are purely numerical. The technique relies on the distance to the decision boundary induced by the decision tree which is computed on the training sample. It is then used as an input for a one dimensional kernel-based density estimator, which provides an estimate of the class membership probability. This geometric method is reported to give good results even with pruned trees.

A Bayesian approach to calibrating class membership probabilities in classification trees was proposed by Cano *et.al.* [14]. In this work the authors apply a Bayesian technique to induce classification trees. Their calibration of class membership probabilities is based upon a modified technique of Bayesian model averaging (BMA) (see Wasserman [94]). The authors also introduce a new approach to define non-uniform priors over the parameters of the models. The authors tested their model on 27 different UCI datasets against a commonly used decision tree inducer *C4.5* (Quinlan [61]), and found an improvement in terms of accuracy in the overall class probability estimates.

A recent study by Böstrom [8] investigated the effect of two commonly employed class probability estimates, *Laplace estimate* and *m-estimate* to random forests of single probability estimation trees (PET) (developed by Provost and Domingos [60]). Their experiment with 34 datasets from the UCI repository suggests that estimating class probabilities using the relative class frequency significantly outperforms or is clearly ahead of both the Laplace estimate and the m-estimate with respect to overall accuracy. Hence, these results strongly suggest that a non-corrected probability estimate should be used in random forests of PETs, in contrast to what previously has been commonly employed.

2.4 Conclusion

This chapter offers a brief review existing methods for class membership calibration. of Despite a wide range of reported methods to date, one common issue faced amongst the majority is that they are based upon parametric techniques.

The remaining chapters in this thesis will introduce novel non-parametric techniques for producing class membership probabilities which are well calibrated and represent an extension of the conformal and Venn prediction frameworks introduced in Chapter 1.

.

Chapter 3

From conformal to probabilistic prediction

This chapter proposes a new method of probabilistic prediction, which is based on conformal prediction. It relies on the idea of idea of transforming p-values into probabilities using probabilistic criteria of efficiency for conformity measures. The method is applied to the standard USPS data set and gives encouraging results.

3.1 Introduction

In essence, conformal predictors introduced in Chapter 1 output systems of p-values: to each potential label of a test object a conformal predictor assigns the corresponding p-value, and a low p-value is interpreted as the label being unlikely. It has been argued, especially by Bayesian statisticians, that p-values are more difficult to interpret than probabilities; besides, in decision problems probabilities can be easily combined with utilities to obtain decisions that are optimal from the point of view of Bayesian decision theory. In this chapter we will apply the idea of transforming p-values into probabilities (used in a completely different context in, e.g., [82], Section 9, and [66]) to conformal prediction: the p-values produced by conformal predictors will be transformed into probabilities.

This chapter (whose findings are also reported in [87]) is based on the recently published work [92] which observed that some criteria of efficiency for conformal prediction (called "probabilistic criteria") encourage using the conditional probability $Q(y \mid x)$ as the conformity score for an observation (x, y), Q being the data-generating distribution. In this chapter we extend this observation to label-conditional predictors (Section 3.2).

Next we imagine that we are given a conformal predictor Γ that is nearly optimal with respect to a probabilistic criterion (such a conformal predictor might be an outcome of a thorough empirical study of various conformal predictors using a probabilistic criterion of efficiency). Essentially, this means that in the limit of a very large training set the p-value that Γ outputs for an observation (x, y) is a monotonic transformation of the conditional probability $Q(y \mid x)$ (Theorem 1 in Section 3.3).

Finally, we transform the p-values back into conditional probabilities using the distribution of p-values in the test set (Section 3.5). Following [82] and [66], we will say that at this step we *calibrate* the p-values into probabilities.

In Section 3.6 we give an example of a realistic situation where use of the techniques developed in this chapter improves on a standard approach. The performance of the probabilistic predictors considered in that section is measured using standard loss functions, logarithmic and Brier (Section 3.4).

It should be noted that in the process of transforming p-values into probabilities suggested in this chapter we lose a valuable feature of conformal prediction, its automatic validity. Our hope, however, is that the advantages of conformal prediction will translate into accurate probabilistic predictions.

3.2 Criteria of efficiency for label-conditional conformal predictors and transducers

Let $\mathbf{X}$ be a measurable space (the *object space*) and $\mathbf{Y}$ be a finite set equipped with the discrete σ-algebra (the *label space*); the *observation space* is defined to be $\mathbf{Z} := \mathbf{X} \times \mathbf{Y}$. A *conformity measure* is a measurable function A that assigns to every sequence $(z_1, \dots, z_l) \in \mathbf{Z}^*$ of observations a same-length sequence $(\alpha_1, \dots, \alpha_l)$ of real numbers and that is equivariant with respect to permutations: for any l and any permutation π of $\{1, \dots, l\}$,

$$(\alpha_1, \dots, \alpha_l) = A(z_1, \dots, z_l) \Longrightarrow \left(\alpha_{\pi(1)}, \dots, \alpha_{\pi(l)}\right) = A\left(z_{\pi(1)}, \dots, z_{\pi(l)}\right).$$

The *label-conditional conformal predictor* determined by A is defined by

$$\Gamma^{\epsilon}(z_1, \dots, z_l, x) := \{y \mid p^y > \epsilon\}, \tag{3.1}$$

where $(z_1, \dots, z_l) \in \mathbf{Z}^*$ is a training sequence, x is a test object, $\epsilon \in (0,1)$ is a given *significance level*, and for each $y \in \mathbf{Y}$ the corresponding *label-conditional p-value* p^y is defined by

$$\begin{aligned} p^y := & \frac{\left|\{i = 1, \dots, l+1 \mid y_i = y \ \& \ \alpha_i^y < \alpha_{l+1}^y\}\right|}{\left|\{i = 1, \dots, l+1 \mid y_i = y\}\right|} \\ & \qquad + \tau \frac{\left|\{i = 1, \dots, l+1 \mid y_i = y \ \& \ \alpha_i^y = \alpha_{l+1}^y\}\right|}{\left|\{i = 1, \dots, l+1 \mid y_i = y\}\right|}, \end{aligned} \tag{3.2}$$

where τ is a random number distributed uniformly on the interval $[0,1]$ and the corresponding sequence of *conformity scores* is defined by

$$(\alpha_1^y, \dots, \alpha_l^y, \alpha_{l+1}^y) := A(z_1, \dots, z_l, (x, y)).$$

It is clear that the system of *prediction sets* (3.1) output by a conformal predictor is nested, namely decreasing in ϵ.

The *label-conditional conformal transducer* determined by A outputs the system of p-values $(p^y \mid y \in \mathbf{Y})$ defined by (3.2) for each training sequence $(z_1, \dots, z_l)$ of observations and each test object x.

Four criteria of efficiency

Suppose that, besides the training sequence, we are also given a test sequence, and would like to measure on it the performance of a label-conditional conformal predictor or transducer. As usual, let us define the performance on the test set to be the average performance (or, equivalently, the sum of performances) on the individual test observations. We focus on the following four criteria of efficiency for individual test observations; all the criteria will work in the same direction: the smaller the better.

- The sum $\sum_{y \in \mathbf{Y}} p^y$ of the p-values; referred to as the *S criterion*. This is applicable to conformal transducers (i.e., the criterion is ϵ-independent).

- The size $|\Gamma^\epsilon|$ of the prediction set at a significance level ϵ; this is the *N criterion*. It is applicable to conformal predictors (ϵ-dependent).
- The sum of the p-values apart from that for the true label: the *OF* ("observed fuzziness") *criterion* (ϵ-dependent).
- The number of false labels included in the prediction set Γ^ϵ at a significance level ϵ; this is the *OE* ("observed excess") *criterion* (ϵ-independent).

The last two criteria are simple modifications of the first two.

Remark 1. Equivalently, the S criterion can be defined as the arithmetic mean $\frac{1}{|\mathbf{Y}|}\sum_{y\in\mathbf{Y}} p^y$ of the p-values; the proof of Theorem 1 below will show that, in fact, we can replace arithmetic mean by any mean ([39], Section 3.1), including geometric, harmonic, etc.

3.3 Optimal idealized conformity measures for a known probability distribution

In this section we consider the idealized case where the probability distribution Q generating independent observations $z_1, z_2, \dots$ is known. In this section we assume, for simplicity, that the set $\mathbf{Z}$ is finite and that $Q(\{z\}) > 0$ for all $z \in \mathbf{Z}$.

An *idealized conformity measure* is a function $A(z, Q)$ of $z \in \mathbf{Z}$ and $Q \in \mathcal{P}(\mathbf{Z})$ (where $\mathcal{P}(\mathbf{Z})$ is the set of all probability measures on $\mathbf{Z}$). We will sometimes write the corresponding conformity scores as $A(z)$, as Q will be clear from the context. The *idealized smoothed label-conditional conformal predictor* corresponding to A outputs the following prediction set $\Gamma^\epsilon(x)$ for each object $x \in \mathbf{X}$ and each significance level $\epsilon \in (0,1)$. For each potential label $y \in \mathbf{Y}$ for x define the corresponding *label-conditional p-value* as

$$\begin{aligned} p^y = p(x,y) := {} & \frac{Q(\{(x',y) \mid x' \in \mathbf{X} \ \& \ A((x',y),Q) < A((x,y),Q)\})}{Q_{\mathbf{Y}}(\{y\})} \\ & + \tau \frac{Q(\{(x',y) \mid x' \in \mathbf{X} \ \& \ A((x',y),Q) = A((x,y),Q)\})}{Q_{\mathbf{Y}}(\{y\})} \end{aligned} \tag{3.3}$$

(this is the idealized analogue of (3.2)), where $Q_{\mathbf{Y}}$ is the marginal distribution of Q on $\mathbf{Y}$ and τ is a random number distributed uniformly on $[0,1]$. The prediction set is

$$\Gamma^{\epsilon}(x) := \{y \in \mathbf{Y} \mid p(x,y) > \epsilon\}\,. \tag{3.4}$$

The *idealized smoothed label-conditional conformal transducer* corresponding to A outputs for each object $x \in \mathbf{X}$ the system of p-values $(p^y \mid y \in \mathbf{Y})$ defined by (3.3); in the idealized case we will usually use the alternative notation $p(x,y)$ for p^y.

Four idealized criteria of efficiency

In this subsection we will apply the four criteria of efficiency that we discussed in the previous section to the idealized case of infinite training and test sequences; since the sequences are infinite, they carry all information about the data-generating distribution Q. We will write $\Gamma^{\epsilon}_A(x)$ for the $\Gamma^{\epsilon}(x)$ in (3.4) and $p_A(x,y)$ for the $p(x,y)$ in (3.3) to indicate the dependence on the choice of the conformity measure A. Let U be the uniform probability measure on the interval $[0,1]$.

An idealized conformity measure A is:

- *S-optimal* if $\mathbb{E}_{(x,\tau)\sim Q_{\mathbf{X}}\times U} \sum_y p_A(x,y) \le \mathbb{E}_{(x,\tau)\sim Q_{\mathbf{X}}\times U} \sum_y p_B(x,y)$ for any idealized conformity measure B, where $Q_{\mathbf{X}}$ is the marginal distribution of Q on $\mathbf{X}$;
- *N-optimal* if $\mathbb{E}_{(x,\tau)\sim Q_{\mathbf{X}}\times U} |\Gamma^{\epsilon}_A(x)| \le \mathbb{E}_{(x,\tau)\sim Q_{\mathbf{X}}\times U} |\Gamma^{\epsilon}_B(x)|$

  for any idealized conformity measure B and any significance level ϵ;
- *OF-optimal* if

$$\mathbb{E}_{((x,y),\tau)\sim Q\times U} \sum_{y'\neq y} p_A(x,y') \le \mathbb{E}_{((x,y),\tau)\sim Q\times U} \sum_{y'\neq y} p_A(x,y')$$

  for any idealized conformity measure B;
- *OE-optimal* if

$$\mathbb{E}_{((x,y),\tau)\sim Q\times U} |\Gamma^{\epsilon}_A(x) \setminus \{y\}| \le \mathbb{E}_{((x,y),\tau)\sim Q\times U} |\Gamma^{\epsilon}_B(x) \setminus \{y\}|$$

for any idealized conformity measure B and any significance level ϵ.

The *conditional probability (PC) idealized conformity measure* is

$$A((x,y),Q) := Q(y \mid x).$$

An idealized conformity measure A is a (label-conditional) *refinement* of an idealized conformity measure B if

$$B((x_1,y)) < B((x_2,y)) \Longrightarrow A((x_1,y)) < A((x_2,y))$$

for all $x_1, x_2 \in \mathbf{Z}$ and all $y \in \mathbf{Y}$. Let $\mathcal{R}(\mathrm{PC})$ be the set of all refinements of the PC idealized conformity measure. If C is a criterion of efficiency (one of the four discussed above), we let $\mathcal{O}(C)$ stand for the set of all C-optimal idealized conformity measures.

Theorem 1. $\mathcal{O}(\mathrm{S}) = \mathcal{O}(\mathrm{OF}) = \mathcal{O}(\mathrm{N}) = \mathcal{O}(\mathrm{OE}) = \mathcal{R}(\mathrm{PC})$.

Proof. We start from proving $\mathcal{R}(\mathrm{PC}) = \mathcal{O}(\mathrm{N})$. Fix a significance level ϵ. A smoothed confidence predictor at level ϵ is defined as a random set of observations $(x,y) \in \mathbf{Z}$; in other words, to each observation (x,y) is assigned the probability $P(x,y)$ that the observation will be outside the prediction set. Under the restriction that the sum of the probabilities $Q(x,y)$ of observations (x,y) outside the prediction set (defined as $\sum_x Q(x,y)P(x,y)$ in the smoothed case) is bounded by $\epsilon Q_{\mathbf{Y}}(y)$ for a fixed y, the N criterion requires us to make the sum of $Q_{\mathbf{X}}(x)$ for (x,y) outside the prediction set (defined as $\sum_x Q_{\mathbf{X}}(x)P(x,y)$ in the smoothed case) as large as possible. It is clear that the set should consist of the observations with the smallest $Q(y \mid x)$ (by the usual Neyman–Pearson argument: cf. [48], Section 3.2). This argument in fact also shows that $\mathcal{O}(\mathrm{N}) \subseteq \mathcal{R}(\mathrm{PC})$.

Next we show that $\mathcal{O}(\mathrm{N}) \subseteq \mathcal{O}(\mathrm{S})$. Let an idealized conformity measure A be N-optimal. By definition,

$$\mathbb{E}_{x,\tau} \left|\Gamma^{\epsilon}_{A}(x)\right| \le \mathbb{E}_{x,\tau} \left|\Gamma^{\epsilon}_{B}(x)\right|$$

for any idealized conformity measure B and any significance level ϵ. Integrating over $\epsilon \in (0,1)$ and swapping the order of integrals and expectations,

$$\mathbb{E}_{x,\tau} \int_0^1 |\Gamma_A^\epsilon(x)| \,\mathrm{d}\epsilon \le \mathbb{E}_{x,\tau} \int_0^1 |\Gamma_B^\epsilon(x)| \,\mathrm{d}\epsilon. \tag{3.5}$$

Since

$$|\Gamma^\epsilon(x)| = \sum_{y \in \mathbf{Y}} 1_{\{p(x,y) > \epsilon\}},$$

we can rewrite (3.5), after swapping the order of summation and integration, as

$$\mathbb{E}_{x,\tau} \sum_{y \in \mathbf{Y}} \left(\int_0^1 1_{\{p_A(x,y) > \epsilon\}} \,\mathrm{d}\epsilon \right) \le \mathbb{E}_{x,\tau} \sum_{y \in \mathbf{Y}} \left(\int_0^1 1_{\{p_B(x,y) > \epsilon\}} \,\mathrm{d}\epsilon \right).$$

Since

$$\int_0^1 1_{\{p(x,y) > \epsilon\}} \,\mathrm{d}\epsilon = p(x,y),$$

we finally obtain

$$\mathbb{E}_{x,\tau} \sum_{y \in \mathbf{Y}} p_A(x,y) \le \mathbb{E}_{x,\tau} \sum_{y \in \mathbf{Y}} p_B(x,y).$$

Since this holds for any idealized conformity measure B, A is S-optimal.

The argument in the previous paragraph in fact shows that $\mathcal{O}(\mathrm{S}) = \mathcal{O}(\mathrm{N}) = \mathcal{R}(\mathrm{PC})$. Indeed, that argument shows that

$$\sum_{y \in \mathbf{Y}} p(x,y) = \int_0^1 |\Gamma^\epsilon(x)| \,\mathrm{d}\epsilon,$$

and so to optimize a conformity measure in the sense of the S criterion it suffices to optimize it in the sense of the N criterion for all ϵ simultaneously (which can, and therefore should, be done). More generally, for any continuous increasing function ϕ we have

$$\begin{aligned}\sum_{y \in \mathbf{Y}} \phi(p(x,y)) = \sum_{y \in \mathbf{Y}} \int_0^1 1_{\{\phi(p(x,y)) > \epsilon\}} \,\mathrm{d}\epsilon &= \int_0^1 \sum_{y \in \mathbf{Y}} 1_{\{p(x,y) > \phi^{-1}(\epsilon)\}} \,\mathrm{d}\epsilon \\ &= \int_0^1 \left| \Gamma^{\phi^{-1}(\epsilon)}(x) \right| \,\mathrm{d}\epsilon = \int \left| \Gamma^{\epsilon'}(x) \right| \phi'(\epsilon') \,\mathrm{d}\epsilon',\end{aligned}$$

which proves Remark 1.

The equality $\mathcal{O}(\mathrm{S}) = \mathcal{O}(\mathrm{OF})$ follows from

$$\mathbb{E}_{x,\tau} \sum_{y} p(x,y) = \mathbb{E}_{(x,y),\tau} \sum_{y' \neq y} p(x,y') + \frac{1}{2},$$

where we have used the fact that $p(x,y)$ is distributed uniformly on $[0,1]$ when $((x,y),\tau) \sim Q \times U$ (see [84] and [92]).

Finally, we notice that $\mathcal{O}(\mathrm{N}) = \mathcal{O}(\mathrm{OE})$. Indeed, for any significance level ϵ,

$$\mathbb{E}_{x,\tau} |\Gamma^{\epsilon}(x)| = \mathbb{E}_{(x,y),\tau} |\Gamma^{\epsilon}(x) \setminus \{y\}| + (1-\epsilon),$$

again using the fact that $p(x,y)$ is distributed uniformly on $[0,1]$ and so $\mathbb{P}_{(x,y),\tau}(y \in \Gamma^{\epsilon}(x)) = 1-\epsilon$. $\square$

3.4 Criteria of efficiency for probabilistic predictors

Given a training set $(z_1, \dots, z_l)$ and a test object x, a probabilistic predictor outputs a probability measure $P \in \mathcal{P}(\mathbf{Y})$, which is interpreted as its probabilistic prediction for the label y of x; we let $\mathcal{P}(\mathbf{Y})$ stand for the set of all probability measures on $\mathbf{Y}$. The two standard ways of measuring the performance of P on the actual label y are the *logarithmic* (or *log*) *loss* $-\ln P(\{y\})$ and the *Brier loss*

$$\sum_{y' \in \mathbf{Y}} \left(1_{\{y'=y\}} - P(\{y'\})\right)^2,$$

where 1_E stands for the indicator of an event E: $1_E = 1$ if E happens and $1_E = 0$ otherwise. The efficiency of probabilistic predictors will be measured by these two loss functions.

Suppose we have a test sequence $(z_{l+1}, \dots, z_{l+k})$, where $z_i = (x_i, y_i)$ for $i = l+1, \dots, l+k$, and we want to evaluate the performance of a probabilistic predictor (trained on a training sequence $z_1, \dots, z_l$) on it. In the next section we will use the *average log loss*

$$-\frac{1}{k} \sum_{i=l+1}^{l+k} \ln P_i(\{y_i\})$$

and the *standardized Brier loss*

$$\sqrt{\frac{1}{k\,|\mathbf{Y}|}\sum_{i=l+1}^{l+k}\sum_{y'\in\mathbf{Y}}\left(1_{\{y'=y_i\}}-P_i(\{y'\})\right)^2},$$

where $P_i \in \mathcal{P}(\mathbf{Y})$ is the probabilistic prediction for x_i. Notice that in the binary case, $|\mathbf{Y}| = 2$, the average log loss coincides with the mean log error and the standardized Brier loss coincides with the root mean square error.

3.5 Calibration of p-values into conditional probabilities

We can use a hold-out set for calibration (say nonparametric, using monotonic regression, as in [97] in a related context). This might be too wasteful, but still we should run experiments. In this section we will discuss an alternative approach: how to calibrate p-values using the test set.

The argument of this section will be somewhat heuristic, and we will not try to formalize it. Fix $y \in \mathbf{Y}$. Suppose that $q := P(y \mid x)$ has an absolutely continuous distribution with density f when $x \sim Q_{\mathbf{X}}$. (In other words, f is the density of the image of $Q_{\mathbf{X}}$ under the mapping $x \mapsto P(y \mid x)$). This assumption contradicts the assumption made earlier that $\mathbf{Z}$ is finite. For the CP idealized conformity measure, we can rewrite (3.3) as

$$p(q) := \int_0^q q' f(q') dq' \Big/ D \,, \tag{3.6}$$

where $D := Q_{\mathbf{Y}}(\{y\})$; alternatively, we can set $D := \int_0^1 q' f(q') dq'$ to the normalizing constant ensuring that $p(1) = 1$. To see how (3.6) is a special case of (3.3) for the CP idealized conformity measure, notice that the probability that $Y = y$ and $P(Y \mid X) \in (q', q' + dq')$, where $(X, Y) \sim f$, is $q' f(q') dq'$. In (3.6) we write $p(q)$ rather than p^y since p^y depends on y only via q.

We are more interested in the inverse function $q(p)$, which is defined by the condition

$$p = \int_0^{q(p)} q' f(q') dq' \Big/ D \,.$$

Algorithm 1 Conformal-type probabilistic predictor

```
Input: training sequence $(z_1, \dots, z_l) \in \mathbf{Z}^l$
Input: calibration sequence $(x_{l+1}, \dots, x_{l+k}) \in \mathbf{X}^k$
Input: test object $x_0$
Output: probabilistic prediction $P \in \mathcal{P}(\mathbf{Y})$ for the label of $x_0$
  for $y \in \mathbf{Y}$ do
      for each $x_i$ in the calibration sequence find the p-value $p_i^y$ by (3.2)
          (with $l + i$ in place of $l + 1$)
      let $g_y$ be the antitonic density on $[0, 1]$ fitted to $p_{l+1}^y, \dots, p_{l+k}^y$
      find the p-value $p_0^y$ by (3.2) (with $0$ in place of $l + 1$)
      for each $y \in \mathbf{Y}$, set $P'(\{y\}) := g_y(1)/g_y(p_0^y)$
  for each $y \in \mathbf{Y}$ set $P(\{y\}) := P'(\{y\})/\sum_{y'} P'(\{y'\})$
```

When $q \sim f$, we have

$$\mathbb{P}(p(q) \le a) = \mathbb{P}(q \le q(a)) = \int_0^{q(a)} f(q')dq'.$$

Therefore, when $q \sim f$, we have

$$\mathbb{P}(a \le p(q) \le a + da) = \int_{q(a)}^{q(a+da)} f(q')dq' \approx \frac{1}{q(a)} \int_{q(a)}^{q(a+da)} q' f(q')dq' = \frac{Dda}{q(a)},$$

and so

$$q(c) \approx D \Big/ \frac{\mathbb{P}(c \le p(q) \le c + dc)}{dc}.$$

This gives rise to the algorithm given as Algorithm 1, which uses real p-values (3.2) instead of the ideal p-values (3.3). The algorithm is transductive in that it uses a training sequence of labelled observations and a calibration sequence of unlabelled objects (in the next section we use the test sequence as the calibration sequence); the latter is used for calibrating p-values into conditional probabilities. Given all the p-values for the calibration sequence with postulated label y, find the corresponding antitonic density $g(p)$ (remember that the function $q(p)$ is known to be monotonic, namely isotonic) using Grenander's estimator (see [36] or, e.g., [27], Chapter 8). Use $D/g(p)$ as the calibration function, where $D := g(1)$ is chosen in such a way that a p-value of 1 is calibrated into a conditional probability of 1. (Alternatively, we could set D to the fraction of observations labelled as y in the training sequence; this approximates setting $D := Q_{\mathbf{Y}}(\{y\})$.) The probabilities produced by this procedure are not guaranteed to lead to a probability measure: the sum over y can be different from 1 (and this phenomenon has been observed in our experiments). Therefore, in the last line of Algorithm 1 we normalize the calibrated p-values to obtain genuine probabilities.

Remark 2. The topic of this chapter is how to transform conformal predictors into probabilistic predictors. Moving in the opposite direction, from probabilistic to conformal predictors, seems to be much easier: given a probabilistic predictor, a natural conformity measure α_i for an observation $z_i = (x_i, y_i)$ in a sequence $z_1, \dots, z_n$ is the probability $\alpha_i := P(y_i)$, where P is the probabilistic prediction for the label of x_i found using that probabilistic predictor from $z_1, \dots, z_n$ (or $z_1, \dots, z_{i-1}, z_{i+1}, \dots, z_n$) as the training set.

3.6 Experiments

In our experiments we use the standard USPS data set of hand-written digits. The size of the training set is 7291, and the size of the test set is 2007; however, instead of using the original split of the data into the two parts, we randomly split all available data (the union of the original training and test sets) into a training set of size 7291 and test set of size 2007. (Therefore, our results somewhat depend on the seed used by the random number generator, but the dependence is minor and does not affect our conclusions at all; we always report results for seed 0.)

A powerful algorithm for the USPS data set is the 1-Nearest Neighbour (1-NN) algorithm using tangent distance [68]. However, it is not obvious how this algorithm could be transformed into a probabilistic predictor. On the other hand, there is a very natural and standard way of extracting probabilities from support vector machines, which we will refer to it as *Platt's algorithm* in this chapter: it is the combination of the method proposed by Platt [59] with pairwise coupling [95] (unlike our algorithm, which is applicable to multi-class problems directly, Platt's method is directly applicable only to binary problems). In this section we will apply our method to the 1-NN algorithm with tangent distance and compare the results to Platt's algorithm as implemented in the function `svm` from the `e1071` R package (for our multi-class problem this function calculates probabilities using the combination of Platt's binary method and pairwise coupling).

TABLE 3.1: The performance of the two algorithms, Platt's (with the optimal values of parameters) and the conformal-type probabilistic predictor based on 1-Nearest Neighbour with tangent distance

algorithm	average log loss	standardized Brier loss
optimized Platt	0.06431	0.05089
conformal-type 1-NN	0.04958	0.04359

TABLE 3.2: The performance of Platt's algorithm with the polynomial kernels of various degrees for the cost parameter $C = 10$

degree	average log loss	standardized Brier loss
1	0.12681	0.07342
2	0.09967	0.06109
3	0.06855	0.05237
4	0.11041	0.06227
5	0.09794	0.06040

There is a standard way of turning a distance into a conformal predictor ([84], Section 3.1): namely, the conformity score α_i of the ith observation in a sequence of observations can be defined as

$$\frac{\min_{j:y_j \neq y_i} d(x_i, x_j)}{\min_{j \neq i: y_j = y_i} d(x_i, x_j)}, \tag{3.7}$$

where d is the distance; the intuition is that an object is considered conforming if it is close to an object labelled in the same way and far from any object labelled in a different way.

Table 3.1 compares the performance of the conformal-type probabilistic predictor based on the 1-NN conformity measure (3.7), where d is tangent distance, with the performance of Platt's algorithm with the optimal values of its parameters. The conformal predictor is parameter-free but Platt's algorithm depends on the choice of the kernel. We chose the polynomial kernel of degree 3 (since it is known to produce the best results: see [80], Section 12.2) and the cost parameter $C := 2.9$ in the case of the average log loss and $C := 3.4$ in the case of the standardized Brier loss (the optimal values in our experiments). (Reporting the performance of Platt's algorithm with optimal parameter values may look like data snooping, but it is fine in this context since we are helping our competitor.) Table 3.2 reports the performance of Platt's algorithm as function of the degree of the polynomial kernel with the cost parameter set at $C := 10$ (the dependence on C is relatively mild, and $C = 10$ gives good performance for all degrees that we consider).

3.7 Conclusion

This chapter has proposed a way to turn conformal predictors into probabilistic ones. Even though the initial results look premising, one of the main drawbacks of the technique is that its success is highly dependent upon the choice of a particular conformity measure and its efficiency under the probabilistic criteria. Indeed, it may be difficult to know a priori which particular measure is the most suitable for a given problem and this may dependent upon the type od dataset. Furthermore, the technique also suffers from a degree of computational inefficiency due to separate treatment of different classes at the stage of calibrating p-values. The following chapter will consider a different method of probabilistic prediction which relies on Venn prediction and which hopes to overcome some of the above challenges.

Chapter 4

Venn-Abers predictors

This chapter focuses on the study of Venn prediction, concentrating on binary prediction problems. Venn predictors produce probability-type predictions for the labels of test objects which are guaranteed to be well calibrated under the standard assumption that the observations are generated independently from the same distribution. This chapter offers a simple formalization and proof of this property. A new class of Venn predictors is introduced, called Venn–Abers predictors, which are based on the idea of isotonic regression. Promising empirical results are demonstrated both for Venn–Abers predictors and for their more computationally efficient simplified version.

4.1 Introduction

Chapter 2 discussed a number of methods of calibrating a variety of machine learning algorithms into better probabilistic classifiers. Despite a degree of success achieved by methods to date, one of the potential drawbacks is that many of them rely on parametric assumptions. One has to often therefore select a certain calibrating method to the particular problem and dataset at hand. It would be instructive to consider a method which relies on as few parametric assumptions as possible but which produces probabilistic predictions which are well calibrated. This chapter introduces a novel method which aims to achieve both of these requirements.

The chapter builds on the work Venn prediction to define a natural class of Venn predictors, called Venn-Abers predictors (VAP) (with the "Abers" part formed by the initial letters of the authors' surnames of the paper by Ayer *et al.* [2] introducing the underlying technique). The latter (introduced in Section 4.3) are defined on top of a wide class of classification algorithms, which we will refer to as "scoring classifiers" in this chapter; each scoring classifier can be automatically transformed into a Venn-Abers predictor, and this transformation is referred to as the "Venn–Abers method". Because of its theoretical guarantees, this method can be used for improving the calibration of probabilistic predictions.

The definition of Venn–Abers predictors was prompted by research carried out by Lambrou *et al.* [45], which demonstrated that the method of calibrating probabilistic predictions introduced by Zadrozny and Elkan [97] introduced in Chapter 2 (an adaptation of the isotonic regression procedure of [2]) does not always achieve its goal and sometimes leads to poorly calibrated predictions. Another paper reporting the possibility for the Zadrozny–Elkan method to produce grossly miscalibrated predictions is [42]. The Venn–Abers method is a simple modification of Zadrozny and Elkan's method; being a special case of Venn prediction, it overcomes the problem of potentially poor calibration.

As illustrated in Chapter 1, Venn predictors (introduced by Vovk *et al.* in [91] and discussed in detail in [84], Chapter 6) are an important class of multiprobability predictors. This chapter describes them in greater detail. The chapter also states an important property of *validity* of Venn predictors: they are automatically well calibrated. In some form this property of validity has been known: see, e.g., [84], Theorem 6.6. However, this known version is complicated, whereas the version described here (Theorem 2 below) is much simpler and the intuition behind it is more transparent. The same section shows (by Theorem 3) that Venn prediction is essentially the only way to achieve this new property of validity.

We show in Theorem 2 in Section 4.2 that Venn predictors are perfectly calibrated. The price to pay, however, is that Venn predictors are multiprobabilistic predictors, in the sense of issuing a set of probabilistic predictions instead of a single probabilistic prediction. Section 4.5 explores the efficiency of Venn–Abers predictors empirically using the fundamental log loss function and another popular loss function, square loss. To apply these loss functions, we need, however, probabilistic predictions rather than multiprobabilistic predictions, therefore Section 4.4 defines natural minimax ways of replacing the latter with the former.

Finally, Section 4.5 explores the empirical predictive performance of the most natural version of the original Zadrozny–Elkan method, the Venn–Abers method, and the latter's simplified version, which is not only simpler but also more efficient computationally. The methods are applied to nine benchmark data sets from the UCI repository [31] and six standard scoring classifiers, and for each combination of a data set and classifier we evaluate the predictive performance of each method. The results show that the Venn–Abers and simplified Venn–Abers methods usually improve the performance of the underlying classifiers, and in these sets of experiments they work better than the original Zadrozny–Elkan method.

4.2 Venn predictors

We consider *observations* $z = (x, y)$ consisting of two components: an *object* $x \in \mathbf{X}$ and its *label* $y \in \mathbf{Y}$. In this chapter we are only interested in the binary case and for concreteness set $\mathbf{Y} := \{0, 1\}$. We assume that $\mathbf{X}$ is a measurable space, so that observations are elements of the measurable space that is the Cartesian product $\mathbf{Z} := \mathbf{X} \times \mathbf{Y} = \mathbf{X} \times \{0, 1\}$.

A *Venn taxonomy* A is a measurable function that assigns to each $n \in \{2, 3, \ldots\}$ and each sequence $(z_1, \ldots, z_n) \in \mathbf{Z}^n$ an equivalence relation $\sim$ on $\{1, \ldots, n\}$ which is equivariant in the sense that, for each n and each permutation π of $\{1, \ldots, n\}$,

$$(i \sim j \mid z_1, \ldots, z_n) \Longrightarrow (\pi(i) \sim \pi(j) \mid z_{\pi(1)}, \ldots, z_{\pi(n)}),$$

where the notation $(i \sim j \mid z_1, \ldots, z_n)$ means that i is equivalent to j under the relation assigned by A to $(z_1, \ldots, z_n)$. The measurability of A means that for all n, i, and j the set $\{(z_1, \ldots, z_n) \,|\, (i \sim j \mid z_1, \ldots, z_n)\}$ is measurable. Define

$$A(j \mid z_1, \ldots, z_n) := \{i \in \{1, \ldots, n\} \mid (i \sim j \mid z_1, \ldots, z_n)\}$$

to be the equivalence class of j. Let $(z_1, \ldots, z_l)$ be a training sequence of observations $z_i = (x_i, y_i)$, $i = 1, \ldots, l$, and x be a test object. The *Venn predictor* associated with a given

Venn taxonomy A outputs the pair (p_0, p_1) as its prediction for x's label, where

$$p_y := \frac{|\{i \in A(l+1 \mid z_1, \dots, z_l, (x,y)) \mid y_i = 1\}|}{|A(l+1 \mid z_1, \dots, z_l, (x,y))|}$$

for both $y \in \{0,1\}$ (notice that the denominator is always positive). Intuitively, p_0 and p_1 are the predicted probabilities that the label of x is 1; of course, the prediction is useful only when $p_0 \approx p_1$. The *probability interval* output by a Venn predictor is defined to be the convex hull $\operatorname{conv}(p_0, p_1)$ of the set $\{p_0, p_1\}$; we will sometimes refer to the pair (p_0, p_1) or the set $\{p_0, p_1\}$ as the *multiprobabilistic prediction*.

Validity of Venn predictors

Let us say that a random variable P taking values in $[0,1]$ is *perfectly calibrated* for a random variable Y taking values in $\{0,1\}$ if

$$\mathbb{E}(Y \mid P) = P \qquad \text{a.s.} \tag{4.1}$$

Intuitively, P is the prediction made by a probabilistic predictor for Y, and perfect calibration means that the probabilistic predictor gets the probabilities right, at least on average, for each value of the prediction. A probabilistic predictor for Y whose prediction P satisfies (4.1) with an approximate equality is said to be well calibrated [26], or unbiased in the small [53, 26].

A *selector* is a random variable taking values 0 or 1.

Theorem 2. *Let $(X_1, Y_1), \dots, (X_l, Y_l), (X, Y)$ be IID (independent identically distributed) random observations. Fix a Venn predictor V and an $l \in \{1, 2, \dots\}$. Let (P_0, P_1) be the output of V given $(X_1, Y_1, \dots, X_l, Y_l)$ as the training set and X as the test object. There exists a selector S such that P_S is perfectly calibrated for Y.*

Intuitively, at least one of the two probabilities output by the Venn predictor is perfectly calibrated. Therefore, if the two probabilities tend to be close to each other, we expect them (or, say, their average) to be well calibrated.

In the proof of Theorem 2 and later in the chapter we will use the notation $\lbag a_1, \dots, a_n \rbag$ for bags (in other words, multisets); the cardinality of the set $\{a_1, \dots, a_n\}$ might well be smaller than n (because of the removal of all duplicates in the bag). Intuitively, $\wr a_1, \ldots, a_n \int$ is the sequence $(a_1, \ldots, a_n)$ with its ordering forgotten. We will sometimes refer to the bag $\wr z_1, \ldots, z_l \int$, where $(z_1, \ldots, z_l)$ is the training sequence, as the training set (although technically it is a multiset rather than a set).

Proof of Theorem 2. Take $S := Y$ as the selector. Let us check that (4.1) is true even if we further condition on the observed bag $\wr (X_1, Y_1), \ldots, (X_l, Y_l), (X, Y) \int$ (so that the remaining randomness consists in generating a random permutation of this bag). We only need to check the equality $\mathbb{E}(Y \mid P = p) = p$, where P is the average of 1s in the equivalence class containing (X, Y), for the ps which are the percentages of 1s in various equivalence classes (further conditioning on the observed bag is not reflected in our notation). For each such p, $\mathbb{E}(Y \mid P = p)$ is the average of 1s in the equivalence classes for which the average of 1s is p; therefore, we indeed have $\mathbb{E}(Y \mid P = p) = p$. □

The following simple corollary of Theorem 2 gives a weaker property of validity, which is sometimes called "unbiasedness in the large" [53, 26].

Corollary 1. *For any Venn predictor V and any $l = 1, 2, \ldots$,*

$$\mathbb{P}(Y = 1) \in \left[\mathbb{E}\left(\underline{V}(X; X_1, Y_1, \ldots, X_l, Y_l)\right), \mathbb{E}\left(\overline{V}(X; X_1, Y_1, \ldots, X_l, Y_l)\right)\right], \tag{4.2}$$

where $(X_1, Y_1), \ldots, (X_l, Y_l), (X, Y)$ are IID observations and $[\underline{V}(\ldots), \overline{V}(\ldots)]$ is the probability interval produced by V for the test object X based on the training sequence $(X_1, Y_1, \ldots, X_l, Y_l)$.

Proof. It suffices to notice that, for a selector S such that $P = P_S$ ((P_0, P_1) being the output of V) satisfies the condition of perfect calibration (4.1),

$$\mathbb{P}(Y = 1) = \mathbb{E}(Y) = \mathbb{E}(\mathbb{E}(Y \mid P_S)) = \mathbb{E}(P_S) \in \left[\mathbb{E}\,\underline{V}, \mathbb{E}\,\overline{V}\right],$$

where the arguments of $\underline{V}$ and $\overline{V}$ are omitted.

The following is a direct argument: First notice that in (4.2) we can replace the left-hand side by the expectation of the arithmetic mean of $Y_1, \ldots, Y_l, Y$ and the right-hand side by the (one-element set consisting of the) expectation of the VP prediction p_Y with Y as the postulated classification for X. Now suppose that the bag

$\wr(X_1, Y_1), \dots, (X_l, Y_l), (X, Y)\int$ has already been generated and it only remains to be decided which element of the bag is the test observation. Then the expectation on the left-hand side becomes a constant (the arithmetic mean of $Y_1, \dots, Y_l, Y$ is now known), and the expectation on the right-hand side becomes the average over the equivalence classes of the percentage of 1s in each equivalence class; the two sides clearly coincide.

$\square$

Unbiasedness in the large (4.2) is easy to achieve even for probabilistic predictors if we do not care about other measures of quality of our predictions: for example, the probabilistic predictor ignoring the xs and outputting k/l as its prediction, where k is the number of 1s in the training sequence of size l, is unbiased in the large. Unbiasedness in the small (4.1) is also easy to achieve if we allow multiprobabilistic predictors: consider the multiprobabilistic predictor ignoring the xs and outputting $\{k/(l+1), (k+1)/(l+1)\}$ as its prediction. The problem is how to achieve predictive efficiency (making our prediction as relevant to the test object as possible without overfitting) while maintaining validity.

Our following result, Theorem 3, will say that under mild regularity conditions unbiasedness in the small (4.1) holds only for Venn predictors (perhaps weakened by adding irrelevant probabilistic predictions) and, therefore, implies all other properties of validity, such as the more complicated one given in [84, Chapter 6].

To state Theorem 3 we need a few further definitions. Let us fix the length l of the training sequence for now. A *multiprobabilistic predictor* is a function that maps each sequence $(z_1, \dots, z_l) \in \mathbf{Z}^l$ to a subset of $[0, 1]$ (not required to be measurable in any sense). Venn predictors are an important example for this chapter. Let us say that a multiprobabilistic predictor is *invariant* if it is independent of the ordering of the training set $(z_1, \dots, z_l)$. An *invariant selector* for an invariant multiprobabilistic predictor F is a measurable function $f : \mathbf{Z}^{l+1} \to [0, 1]$ such that $f(z_1, \dots, z_{l+1})$ does not change when $z_1, \dots, z_l$ are permuted and such that $f(z_1, \dots, z_{l+1}) \in F(z_1, \dots, z_l)$ for all $(z_1, \dots, z_{l+1})$. (It is natural to consider only invariant predictors and selectors under the IID assumption because of the principle of sufficiency [23, Chapter 2]). We say that an invariant multiprobabilistic predictor F is *invariantly perfectly calibrated* if it has an invariant selector f

such that

$$\mathbb{E}\big(Y \mid f(Z_1,\dots,Z_l,(X,Y))\big) = f(Z_1,\dots,Z_l,(X,Y)) \text{ a.s.} \tag{4.3}$$

whenever $Z_1,\dots,Z_l,(X,Y)$ are IID observations.

Theorem 3. *If an invariant multiprobabilistic predictor F is invariantly perfectly calibrated, then it contains a Venn predictor V in the sense that both elements of $V(Z_1,\dots,Z_l)$ belong to $F(Z_1,\dots,Z_l)$ almost surely provided $Z_1,\dots,Z_l$ are IID.*

Proof. Let f be an invariant selector of F satisfying the condition (4.3) of being invariantly perfectly calibrated. By definition,

$$\mathbb{E}\big(Y - f(Z_1,\dots,Z_l,(X,Y)) \mid f(Z_1,\dots,Z_l,(X,Y))\big) = 0 \text{ a.s.},$$

which implies

$$\mathbb{E}\big((Y - f(Z_1,\dots,Z_l,(X,Y)))1_{\{f(Z_1,\dots,Z_l,(X,Y))\in[a,b]\}}\big) = 0 \text{ a.s.} \tag{4.4}$$

for all intervals $[a,b]$ with rational end-points. The expected value in (4.4) can be obtained in two steps: first we average

$$(y'_{l+1} - f(z'_1,\dots,z'_{l+1}))1_{\{f(z'_1,\dots,z'_{l+1})\in[a,b]\}}$$

over the orderings $(z'_1,\dots,z'_{l+1})$ of each bag $\wr z_1,\dots,z_{l+1}\int$, where $z_i = (x_i,y_i)$ and $z'_i = (x'_i,y'_i)$, and then we average over the bags $\wr z_1,\dots,z_{l+1}\int$ generated according to $z_i := Z_i$, $i = 1,\dots,l$, and $z_{l+1} := (X,Y)$. The first operation is discrete: the average over the orderings of $\wr z_1,\dots,z_{l+1}\int$ is the arithmetic mean of $(y_i - p_i)1_{\{p_i\in[a,b]\}}$ over $i = 1,\dots,l+1$, where $p_i := f(\dots,z_i)$ and the dots stand for $z_1,\dots,z_{i-1}$ and $z_{i+1},\dots,z_{l+1}$ arranged in any order (since f is invariant, the order does not matter). By the completeness of the statistic that maps a data sequence of size $l+1$ to the corresponding bag [48, Section 4.3], this average is zero for all $[a,b]$ and almost all bags. Without loss of generality we assume that this holds for all bags.

Define a Venn taxonomy A as follows: given a sequence $(z_1,\dots,z_{l+1})$, set $i \sim j$ if $p_i = p_j$ where p is defined as above. It is easy to check that the corresponding Venn predictor satisfies the requirement in Theorem 3. □

Remark 3. The invariance assumption in Theorem 3 is essential. Indeed, suppose $l > 1$ and consider the multiprobabilistic predictor whose prediction for the label of the test observation does not depend on the objects and is

$$\begin{cases} \{k/l, (k+1)/l\} & \text{if } y_1 = 0 \\ \{(k-1)/l, k/l\} & \text{if } y_1 = 1, \end{cases}$$

where k is the number of 1s among the labels of the l training observations. This non-invariant predictor is perfectly calibrated (see below) but does not contain a Venn predictor (if it did, such a Venn predictor, being invariant, would always output the one-element multiprobabilistic prediction $\{k/l\}$, which is impossible). Let us check that this non-invariant predictor is indeed perfectly calibrated, even given the union of the training set and the test observation (i.e., given the bag of size $l+1$ obtained from the training sequence by joining the test observation and then forgetting the ordering). Take the selector such that the selected probabilistic predictor is

$$\begin{cases} k/l & \text{for sequences of the form } 0\dots0 \\ (k+1)/l & \text{for sequences of the form } 0\dots1 \\ (k-1)/l & \text{for sequences of the form } 1\dots0 \\ k/l & \text{for sequences of the form } 1\dots1. \end{cases}$$

For a binary sequence of labels of length $l+1$ with m 1s the probabilistic prediction P for its last element will be, therefore,

$$\begin{cases} m/l & \text{for sequences of the form } 0\dots0 \\ m/l & \text{for sequences of the form } 0\dots1 \\ (m-1)/l & \text{for sequences of the form } 1\dots0 \\ (m-1)/l & \text{for sequences of the form } 1\dots1. \end{cases}$$

The conditional probability that $Y = 1$ (Y being the label of the last element) given $P = p$ (and given m) is

$$\frac{\binom{l-1}{m-1}}{\binom{l}{m}} = \frac{m}{l}$$

when $p = m/l$ and is

$$\frac{\binom{l-1}{m-2}}{\binom{l}{m-1}} = \frac{m-1}{l}$$

when $p = (m-1)/l$; in both cases we have perfect calibration.

4.3 Venn–Abers predictors

We say that a function f is *increasing* if its domain is an ordered set and $t_1 \le t_2 \Rightarrow f(t_1) \le f(t_2)$. As described in the introductory chapter, many machine-learning algorithms for classification are in fact *scoring classifiers*: when trained on a training sequence of observations and fed with a test object x, they output a *prediction score* $s(x)$; we will call $s : \mathbf{X} \to \mathbb{R}$ the *scoring function* for that training sequence. The actual classification algorithm is obtained by fixing a threshold c and predicting the label of x to be 1 if and only if $s(x) \ge c$ (or if and only if $s(x) > c$). Alternatively, one could apply an increasing function g to $s(x)$ in an attempt to "calibrate" the scores, so that $g(s(x))$ can be used as the predicted probability that the label of x is 1.

Fix a scoring classifier and let $(z_1, \dots, z_l)$ be a training sequence of observations $z_i = (x_i, y_i)$, $i = 1, \dots, l$. The most direct application [97] of the method of isotonic regression [2] to the problem of score calibration is as follows. Train the scoring classifier on the training sequence and compute the score $s(x_i)$ for each training observation (x_i, y_i), where s is the scoring function for $(z_1, \dots, z_l)$. Let g be the increasing function on the set $\{s(x_1), \dots, s(x_l)\}$ that maximizes the likelihood

$$\prod_{i=1}^{l} p_i, \quad \text{where } p_i := \begin{cases} g(s(x_i)) & \text{if } y_i = 1 \\ 1 - g(s(x_i)) & \text{if } y_i = 0. \end{cases} \tag{4.5}$$

Such a function g is indeed unique [2, Corollary 2.1] and can be easily found using the "pair-adjacent violators algorithm" (PAVA, described in detail in the summary of [2] and

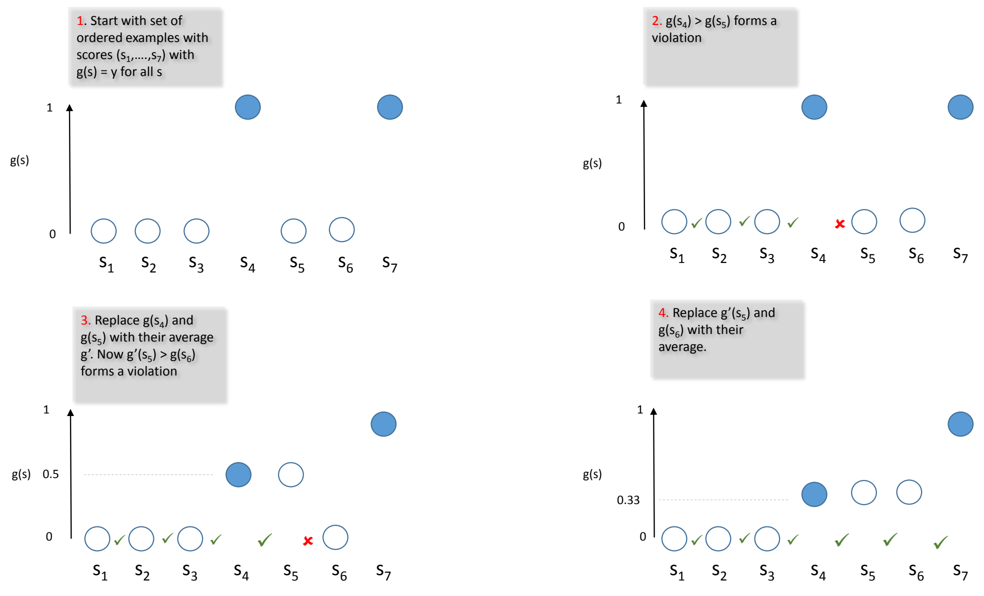
1. Start with set of ordered examples with scores (s1,....,s7) with g(s) = y for all s
2. g(s4) > g(s5) forms a violation
3. Replace g(s4) and g(s5) with their average g'. Now g'(s5) > g(s6) forms a violation
4. Replace g'(s5) and g(s6) with their average.
g(s)
1
0
0.5
0.33
s1
s2
s3
s4
s5
s6
s7

FIGURE 4.1: An example of the pair-adjacent violator algorithm (PAVA)

in [13, Section 1.2]; see also the proof of Lemma 1 below). We will say that g is the *isotonic calibrator* for $((s(x_1), y_1), \dots, (s(x_l), y_l))$. To predict the label of a test object x, the direct method finds the closest $s(x_i)$ to $s(x)$ and outputs $g(s(x_i))$ as its prediction (in the case of ties our implementation of this method used in Section 4.5 chooses the smaller $s(x_i)$; however, ties almost never happen in our experiments). We will refer to this as the *direct isotonic-regression* (DIR) method. An illustration of the PAVA algorithm is shown in Figure 4.1.

The direct method is prone to overfitting as the same observations $z_1, \dots, z_l$ are used both for training the scoring classifier and for calibration without taking any precautions. The *Venn–Abers predictor* corresponding to the given scoring classifier is the multiprobabilistic predictor that is defined as follows. Try the two different labels, 0 and 1, for the test object x. Let s_0 be the scoring function for $(z_1, \dots, z_l, (x, 0))$, s_1 be the scoring function for $(z_1, \dots, z_l, (x, 1))$, g_0 be the isotonic calibrator for

$$\big((s_0(x_1), y_1), \dots, (s_0(x_l), y_l), (s_0(x), 0)\big), \tag{4.6}$$

Algorithm 2 Venn–Abers predictor

```
Input: training sequence $(z_1, \dots, z_l)$
Input: test object $x$
Output: multiprobabilistic prediction $(p_0, p_1)$
  for $y \in \{0, 1\}$ do
      set $s_y$ to the scoring function for
        $(z_1, \dots, z_l, (x, y))$
      set $g_y$ to the isotonic calibrator for
            $(s_y(x_1), y_1), \dots, (s_y(x_l), y_l), (s_y(x), y)$
      set $p_y := g_y(s_y(x))$
```

and g_1 be the isotonic calibrator for

$$\big((s_1(x_1), y_1), \dots, (s_1(x_l), y_l), (s_1(x), 1)\big). \tag{4.7}$$

The multiprobabilistic prediction output by the Venn–Abers predictor is (p_0, p_1), where $p_0 := g_0(s_0(x))$ and $p_1 := g_1(s_1(x))$. (And we can expect p_0 and p_1 to be close to each other unless DIR overfits grossly.) The Venn–Abers predictor is described as Algorithm 2.

The intuition behind Algorithm 2 is that it tries to evaluate the robustness of the DIR prediction. To see how sensitive the scoring function is to the training set we extend the latter by adding the test object labelled in two different ways. And to see how sensitive the probabilistic prediction is, we again consider the training set extended in two different ways (if it is sensitive, the prediction will be fragile even if the scoring function is robust). For large data sets and inflexible scoring functions, we will have $p_0 \approx p_1$, and both numbers will be close to the DIR prediction. However, even if the data set is very large but the scoring function is very flexible, p_0 can be far from p_1 (the extreme case is where the scoring function is so flexible that it ignores all observations apart from a few that are most similar to the test object, and in this case it does not matter how big the data set is). We rarely know in advance how flexible our scoring function is relative to the size of the data set, and the difference between p_0 and p_1 gives us some indication of this.

The following proposition says that Venn–Abers predictors are Venn predictors and, therefore, inherit all properties of validity of the latter, such as Theorem 2.

Proposition 1. *Venn–Abers predictors are Venn predictors.*

Proof. Fix a Venn–Abers predictor. The corresponding Venn taxonomy is defined as follows: given a sequence

$$(z_1, \dots, z_n) = ((x_1, y_1), \dots, (x_n, y_n)) \in (\mathbf{X} \times \{0, 1\})^n$$

and $i, j \in \{1, \dots, n\}$, we set $i \sim j$ if and only if $g(s(x_i)) = g(s(x_j))$, where s is the scoring function for $(z_1, \dots, z_n)$ and g is the isotonic calibrator for

$$\big((s(x_1), y_1), \dots, (s(x_n), y_n)\big).$$

Lemma 1 below shows that the Venn predictor corresponding to this taxonomy gives predictions identical to those given by the original Venn–Abers predictor. This proves the proposition. □

Lemma 1. *Let g be the isotonic calibrator for $((t_1, y_1), \dots, (t_n, y_n))$, where $t_i \in \mathbb{R}$ and $y_i \in \{0, 1\}$, $i = 1, \dots, n$. Any $p \in \{g(t_1), \dots, g(t_n)\}$ is equal to the arithmetic mean of the labels y_i of the t_i, $i = 1, \dots, n$, satisfying $g(t_i) = p$.*

Proof. The statement of the lemma immediately follows from the definition of the PAVA [2, summary], which we will reproduce here. Arrange the numbers t_i in the strictly increasing order $t_{(1)} < \dots < t_{(k)}$, where $k \le n$ is the number of distinct elements among t_i. We would like to find the increasing function g on the set $\{t_{(1)}, \dots, t_{(k)}\} = \{t_1, \dots, t_n\}$ maximizing the likelihood (defined by (4.5) with t_i in place of $s(x_i)$ and n in place of l). The procedure is recursive. At each step the set $\{t_{(1)}, \dots, t_{(k)}\}$ is partitioned into a number of disjoint cells consisting of adjacent elements of the set; to each cell is assigned a ratio a/N (formally, a pair of integers, with $a \ge 0$ and $N > 0$); the function g defined at this step (perhaps to be redefined at the following steps) is constant on each cell. For $j = 1, \dots, k$, let $a_{(j)}$ be the number of i such that $y_i = 1$ and $t_i = t_{(j)}$, and let $N_{(j)}$ be the number of i such that $t_i = t_{(j)}$. Start from the partition of $\{t_{(1)}, \dots, t_{(k)}\}$ into one-element cells, assign the ratio $a_{(j)}/N_{(j)}$ to $\{t_{(j)}\}$, and set

$$g(t_{(j)}) := \frac{a_{(j)}}{N_{(j)}} \tag{4.8}$$

(in the notation used in this proof, a/N is a pair of integers whereas $\frac{a}{N}$ is a rational number, the result of the division). If the function g is increasing, we are done. If not, there is a pair C_1, C_2 of adjacent cells ("violators") such that C_1 is to the left of C_2 and $g(C_1) > g(C_2)$ (where $g(C)$ stands for the common value of $g(t_{(j)})$ for $t_{(j)} \in C$); in this case redefine the partition by merging C_1 and C_2 into one cell C, assigning the ratio $(a_1 + a_2)/(N_1 + N_2)$ to C, where a_1/N_1 and a_2/N_2 are the ratios assigned to C_1 and C_2, respectively, and setting

$$g(t_{(j)}) := \frac{N_1}{N_1 + N_2} g(C_1) + \frac{N_2}{N_1 + N_2} g(C_2) = \frac{a_1 + a_2}{N_1 + N_2} \tag{4.9}$$

for all $t_{(j)} \in C$. Repeat the process until g becomes increasing (the number of cells decreases by 1 at each iteration, so the process will terminate in at most k steps). The final function g is the one that maximizes the likelihood. The statement of the lemma follows from this recursive definition: it is true by definition for the initial function (4.8) and remains true when g is redefined by (4.9). □

4.4 Probabilistic predictors derived from Venn predictors

In the next section we will compare Venn–Abers predictors with known probabilistic predictors using standard loss functions. Since Venn–Abers predictors output pairs of probabilities rather than point probabilities, we will need to fit them (somewhat artificially) in the standard framework extracting one probability p from p_0 and p_1.

We will consider two loss functions, log loss and square loss. The *log loss* suffered when predicting $p \in [0, 1]$ whereas the true label is y is

$$\lambda_{\ln}(p, y) := \begin{cases} -\ln(1 - p) & \text{if } y = 0 \\ -\ln p & \text{if } y = 1. \end{cases}$$

This is the most fundamental loss function, since the cumulative loss $\sum_{i=1}^{n} \lambda_{\ln}(p_i, y_i)$ over a test sequence of size n is equal to the minus log of the probability that the predictor assigns to the sequence (this assumes either the batch mode of prediction with independent test observations or the online mode of prediction); therefore, a smaller cumulative log loss corresponds to a larger probability. The *square loss* suffered when predicting $p \in [0,1]$ for the true label y is

$$\lambda_{\mathrm{sq}}(p,y) := (y-p)^2.$$

The main advantage of this loss function is that it is *proper* (see, e.g., [26]): the function $\mathbb{E}_{y \sim B_p} \lambda_{\mathrm{sq}}(q,y)$ of $q \in [0,1]$, where B_p is the Bernoulli distribution with parameter p, attains its minimum at $q = p$. (Of course, the log loss function is also proper.)

First suppose that our loss function is $\lambda_{\ln}$ and we are given a multiprobabilistic prediction (p_0, p_1); let us find the corresponding minimax probabilistic prediction p. If the true outcome is $y = 0$, our regret for using p instead of the appropriate p_0 is $-\ln(1-p) + \ln(1-p_0)$. If $y = 1$, our regret for using p instead of the appropriate p_1 is $-\ln p + \ln p_1$. The first regret as a function of $p \in [0,1]$ strictly increases from a nonpositive value to ∞ as p changes from 0 to 1. The second regret as a function of p strictly decreases from ∞ to a nonpositive value as p changes from 0 to 1. Therefore, the minimax regret is the solution to

$$-\ln(1-p) + \ln(1-p_0) = -\ln p + \ln p_1,$$

which is

$$p = \frac{p_1}{1-p_0+p_1}. \tag{4.10}$$

The intuition behind this minimax value of p is that we can interpret the multiprobabilistic prediction (p_0, p_1) as the unnormalized probability distribution P on $\{0,1\}$ such that $P(\{0\}) = 1 - p_0$ and $P(\{1\}) = p_1$; we then normalize P to get a genuine probability distribution $P' := P/P(\{0,1\})$, and the p in (4.10) is equal to $P'(\{1\})$. Of course, it is always true that $p \in \mathrm{conv}(p_0, p_1)$.

In the case of the square loss function, the regret is

$$\begin{cases} p^2 - p_0^2 & \text{if } y = 0 \\ (1-p)^2 - (1-p_1)^2 & \text{if } y = 1 \end{cases}$$

and the two regrets are equal when

$$p := p_1 + p_0^2/2 - p_1^2/2. \tag{4.11}$$

To see how natural this expression is notice that (4.11) is equivalent to

$$p = \bar{p} + (p_1 - p_0)\left(\frac{1}{2} - \bar{p}\right),$$

where $\bar{p} := (p_0 + p_1)/2$. Therefore, p is a regularized version of $\bar{p}$: we move $\bar{p}$ towards the neutral value $1/2$ in the typical (for the Venn–Abers method) case where $p_0 < p_1$. In any case, we always have $p \in \operatorname{conv}(p_0, p_1)$.

The following lemma shows that log loss is never infinite for probabilistic predictors derived from Venn predictors.

Lemma 2. *Neither of the methods discussed in this section (see* (4.10) *and* (4.11)*) ever produces* $p \in \{0, 1\}$ *when applied to Venn–Abers predictors.*

Proof. Lemma 1 implies that $p_0 < 1$ and that $p_1 > 0$. It remains to notice that both (4.10) and (4.11) produce p in the interior of $\operatorname{conv}(p_0, p_1)$ if $p_0 \neq p_1$ and produce $p = p_0 = p_1$ if $p_0 = p_1$ (and this is true for any sensible averaging method). □

4.5 Experimental results

In this section we compare various calibration methods discussed so far by applying them to six standard scoring classifiers (we will usually omit "scoring" in this section) available within Weka [38], a machine learning tool developed at the University of Waikato, NZ. The standard classifiers are J48 decision trees (abbreviated to J48, or even to J), J48 decision trees with bagging (J48 Bagging, or JB), logistic regression (LR), naïve Bayes (NB), neural networks (NN), and support vector machines calibrated using a sigmoid function as defined by Platt [59] (SVM Platt, or simply SVM). Each of these standard classifiers produces scores in the interval $[0, 1]$, which can then be used as probabilistic predictions; however, in most previous studies these have been found to be inaccurate (see [97] and [46]). We use the scores generated by classifiers as inputs, and by applying

TABLE 4.1: Log loss (MLE) results obtained using standard Weka classifiers (W) and the three calibration methods (VA, SVA, DIR) applied to the standard classifiers' outputs for the following Weka classifiers: J48, J48 Bagging, logistic regression (upper part) and naïve Bayes, neural networks, and SVM Platt (lower part). The best results for each pair (classifier, dataset) are in bold.

	J48 (J)				J48 Bagging (JB)				logistic regression (LR)			
	W	VA	SVA	DIR	W	VA	SVA	DIR	W	VA	SVA	DIR
Australian	∞	**0.380**	0.469	∞	**0.328**	0.369	0.344	∞	0.342	**0.340**	0.340	∞
Breast	∞	**0.607**	0.642	∞	**0.581**	0.592	0.636	∞	0.584	**0.567**	0.586	∞
Diabetes	∞	**0.552**	0.635	∞	**0.504**	0.515	0.561	∞	0.492	**0.490**	0.491	∞
Echo	∞	**0.606**	0.670	∞	0.556	**0.517**	0.563	∞	∞	**0.589**	0.606	∞
Hepatitis	∞	**0.491**	0.528	∞	**0.420**	0.456	0.434	∞	∞	**0.393**	0.504	∞
Ionosphere	∞	**0.383**	0.410	∞	∞	0.387	**0.251**	∞	∞	**0.387**	0.524	∞
Labor	∞	**0.503**	0.537	∞	0.427	0.427	**0.385**	∞	1.927	0.687	**0.297**	∞
Liver	∞	**0.662**	0.866	∞	**0.609**	0.635	0.707	∞	0.619	0.622	**0.611**	∞
Vote	∞	**0.134**	0.145	∞	0.135	0.159	**0.131**	∞	1.059	0.188	**0.148**	∞

	naïve Bayes (NB)				neural networks (NN)				SVM Platt (SVM)			
	W	VA	SVA	DIR	W	VA	SVA	DIR	W	VA	SVA	DIR
Australian	0.839	**0.355**	0.367	∞	0.557	**0.427**	0.450	∞	0.391	0.356	**0.351**	∞
Breast	0.663	0.563	**0.551**	∞	0.774	**0.615**	0.738	∞	0.583	**0.568**	0.582	∞
Diabetes	0.753	**0.495**	0.508	∞	0.536	**0.500**	0.519	∞	0.491	0.497	**0.490**	∞
Echo	0.658	**0.505**	0.522	∞	0.770	**0.578**	0.605	∞	0.558	**0.495**	0.538	∞
Hepatitis	0.936	**0.365**	0.372	∞	0.753	**0.471**	0.484	∞	0.435	**0.349**	0.404	∞
Ionosphere	0.704	0.262	**0.227**	∞	0.625	0.427	**0.379**	∞	0.359	**0.250**	0.333	∞
Labor	1.854	0.410	**0.296**	∞	0.325	0.560	**0.298**	∞	3.643	0.364	**0.287**	∞
Liver	0.727	0.649	0.661	∞	0.642	**0.603**	0.615	∞	0.645	0.663	**0.639**	∞
Vote	0.594	0.218	**0.211**	∞	0.235	0.229	**0.158**	∞	0.125	0.211	**0.121**	∞

the DIR (defined in Section 4.3), Venn–Abers (VA), and simplified Venn–Abers (SVA, see below) methods we investigate how well we can calibrate the scores and improve them in their role as probabilistic predictions.

In the set of experiments described in this section we do not perform a direct comparison to the method developed by Langford and Zadrozny [46] primarily because, as far as we are aware, the algorithms described in their work are not publicly available.

For the purpose of comparison we use a total of nine datasets with binary labels (encoded as 0 or 1) obtained from the UCI machine learning repository [31]: Australian Credit (which we abbreviate to Australian), Breast Cancer (Breast), Diabetes, Echocardiogram (Echo), Hepatitis, Ionosphere, Labor Relations (Labor), Liver Disorders (Liver), and Congressional Voting (Vote). The datasets vary in size as well as the number and type of attributes in order to give a reasonable range of conditions encountered in practice.

TABLE 4.2: The analogue of Table 4.1 for square loss (RMSE).

	J48 (J)				J48 Bagging (JB)				logistic regresion (LR)			
	W	VA	SVA	DIR	W	VA	SVA	DIR	W	VA	SVA	DIR
Australian	0.366	**0.346**	0.359	0.366	**0.313**	0.338	0.318	0.323	**0.317**	0.319	0.319	0.321
Breast	0.472	**0.453**	0.463	0.473	**0.443**	0.451	0.460	0.474	0.442	**0.437**	0.444	0.450
Diabetes	0.449	**0.431**	0.443	0.449	**0.407**	0.415	0.420	0.427	**0.399**	0.401	0.401	0.402
Echo	0.478	**0.456**	0.460	0.482	0.427	**0.417**	0.423	0.444	0.457	**0.443**	0.446	0.475
Hepatitis	0.407	**0.393**	0.401	0.419	**0.362**	0.390	0.368	0.391	0.400	**0.357**	0.384	0.411
Ionosphere	0.318	0.355	**0.312**	0.318	0.267	0.356	**0.261**	0.267	0.357	0.363	**0.349**	0.361
Labor	0.407	0.403	**0.402**	0.413	0.361	0.371	**0.339**	0.341	0.294	0.498	**0.287**	0.303
Liver	0.528	**0.482**	0.518	0.528	**0.457**	0.478	0.478	0.493	0.460	0.463	**0.458**	0.461
Vote	0.187	0.186	**0.186**	0.187	0.187	0.206	**0.186**	0.188	0.198	0.233	**0.195**	0.203

	naïve Bayes (NB)				neural networks (NN)				SVM Platt (SVM)			
	W	VA	SVA	DIR	W	VA	SVA	DIR	W	VA	SVA	DIR
Australian	0.392	**0.328**	0.333	0.335	**0.360**	0.363	0.361	0.371	0.343	**0.324**	0.325	0.327
Breast	0.449	0.436	**0.427**	0.433	0.485	**0.465**	0.491	0.508	0.443	**0.431**	0.442	0.447
Diabetes	0.420	**0.406**	0.410	0.413	0.413	**0.408**	0.413	0.417	0.399	**0.393**	0.400	0.402
Echo	0.428	**0.408**	0.412	0.426	0.457	**0.436**	0.443	0.468	**0.416**	0.427	0.418	0.431
Hepatitis	0.357	0.339	**0.335**	0.342	0.396	0.402	**0.379**	0.427	0.350	**0.350**	0.353	0.364
Ionosphere	0.281	0.273	**0.250**	0.251	0.321	0.378	**0.316**	0.333	0.312	**0.309**	0.312	0.316
Labor	**0.256**	0.363	0.284	0.281	**0.279**	0.442	0.293	0.307	**0.274**	0.358	0.280	0.283
Liver	0.480	**0.476**	0.478	0.487	0.459	**0.456**	0.456	0.463	0.473	0.477	**0.472**	0.477
Vote	0.292	0.257	0.251	**0.250**	0.216	0.271	**0.206**	0.227	**0.183**	0.191	0.185	0.188

In our comparison we use the two standard loss functions discussed in the previous section. Namely, on a given test sequence of length n we will calculate the *mean log error* (MLE)

$$\frac{1}{n}\sum_{i=1}^{n}\lambda_{\ln}(p_i, y_i) \tag{4.12}$$

and the *root mean square error* (RMSE)

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\lambda_{\mathrm{sq}}(p_i, y_i)}, \tag{4.13}$$

where p_i is the probabilistic prediction for the label y_i of the ith observation in the test sequence. MLE (4.12) can be infinite, namely when predicting $p_i \in \{0, 1\}$ while being incorrect. It therefore penalises the overly confident probabilistic predictions much more significantly than RMSE. We compare the performance of the standard classifiers with their versions calibrated using the three methods (VA, SVA, and DIR) under both loss functions for each dataset. In each experiment we randomly permute the dataset and use

Algorithm 3 Simplified Venn–Abers predictor

```
Require: training sequence $(z_1, \dots, z_l)$
Require: test object $x$
Ensure: multiprobabilistic prediction $(p_0, p_1)$
  set $s$ to the scoring function for $(z_1, \dots, z_l)$
  for $y \in \{0, 1\}$ do
      set $g_y$ to the isotonic calibrator for
            $(s(x_1), y_1), \dots, (s(x_l), y_l), (s(x), y)$
      set $p_y := g_y(s(x))$
```

the first $2/3$ observations for training and the remaining $1/3$ for testing.

One of the potential drawbacks of the Venn–Abers method is its computational inefficiency: for each test object the scores have to be recalculated for the training sequence extended by including the test object first labelled as 0 and then labelled as 1. This implies that the total calculation time is at least $2n$ times that of the underlying classifier, where n is the number of test observations. Therefore, we define a simplified version of Venn–Abers predictors, for which the scores are calculated only once without recalculating them for each test object with postulated labels 0 and 1.

In detail, the *simplified Venn–Abers predictor* for a given scoring classifier is defined as follows. Let $(z_1, \dots, z_l)$ be a training sequence and x be a test object. Define s to be the scoring function for $(z_1, \dots, z_l)$, g_0 to be the isotonic calibrator for

$$\big((s(x_1), y_1), \dots, (s(x_l), y_l), (s(x), 0)\big),$$

and g_1 to be the isotonic calibrator for

$$\big((s(x_1), y_1), \dots, (s(x_l), y_l), (s(x), 1)\big)$$

(cf. (4.6) and (4.7)). The multiprobabilistic prediction output for the label of x by the simplified Venn–Abers (SVA) predictor is (p_0, p_1), where $p_0 := g_0(s(x))$ and $p_1 := g_1(s(x))$. This method, summarized as Algorithm 3, is intermediate between DIR and the Venn–Abers method.

Notice that Lemma 2 continues to hold for SVA predictors; therefore, they never suffer infinite loss even under the log loss function. On the other hand, the following proposition shows that SVA predictors can violate the property (4.2) of unbiasedness in the large; in particular, they are not Venn predictors (cf. Corollary 1).

Proposition 2. *There exists a simplified Venn–Abers predictor violating* (4.2) *for some l.*

Proof. Let the object space be the real line, $\mathbf{X} := \mathbb{R}$, and the probability distribution generating independent observations (X, Y) be such that: the marginal distribution of X is continuous; the probability that $X > 0$ (and, therefore, the probability that $X < 0$) is $1/2$; the probability that $Y = 1$ given $X < 0$ is $1/3$; the probability that $Y = 1$ given $X > 0$ is $2/3$. Therefore, $\mathbb{P}(Y = 1) = 1/2$. Let l be a large number (we are using a somewhat informal language, but formalization will be obvious). Given a training set $(z_1, \dots, z_l)$, where $z_i = (x_i, y_i)$ for all i, the scoring function s is:

$$s(x) := \begin{cases} 0 & \text{if } x \in \{x_1, \dots, x_l\} \text{ and } x < 0 \\ 1 & \text{if } x \in \{x_1, \dots, x_l\} \text{ and } x > 0 \\ 2 & \text{if } x \notin \{x_1, \dots, x_l\}. \end{cases}$$

It is easy to see that, with high probability,

$$\underline{V} \to 2/3, \quad \overline{V} = 1.$$

Therefore, (4.2) is violated. □

Proposition 2 shows that SVA predictors are not always valid; however, the construction in its proof is artificial, and our hope is that they are "nearly valid" in practice, since they are a modification of provably valid predictors.

For each dataset/classifier combination, we repeat the same experiment a total of 100 times for standard classifiers (denoted W in the tables), SVA, and DIR and 16 times for VA (because of the computational inefficiency of the latter) and average the results. The same 100 random splits into training and test sets are used for W, SVA, and DIR, but for VA the 16 splits are different.

Tables 4.1–4.2 compare the overall losses computed according to (4.12) (MLE, used in Table 4.1) and (4.13) (RMSE, used in Table 4.2) for probabilities generated by the standard classifiers as implemented in Weka (W) and the corresponding Venn–Abers (VA), simplified Venn–Abers (SVA), and direct isotonic-regression (DIR) predictors. The values in bold indicate the lowest of the four losses for each dataset/classifier combination. The column titles mention both fuller and shorter names for the six standard classifiers; the short name "SVM" is especially appropriate when using VA, SVA, and DIR, in which case the application of the sigmoid function in Platt's method is redundant. The three entries of ∞ in the column W for logistic regression of Table 4.1 come out as infinities in our experiments only because of the limited machine accuracy: logistic regression sometimes outputs probabilistic predictions that are so close to 0 or 1 that they are rounded to 0 or 1, respectively, by hardware.

In the case of MLE, the VA and SVA methods improve the predictive performance of the majority of the standard classifiers on most datasets. A major exception is J48 Bagging. The application of bagging to J48 decision trees improves the calibration significantly as bagging involves averaging over different training sets in order to reduce the underlying classifier's instability. The application of VA and SVA to J48 Bagging is not found to improve the log or square loss significantly. What makes VA and SVA useful is that for many data sets other classifiers, less well calibrated than J48 Bagging, provide more useful scores.

In the case of RMSE, the application of VA and SVA also often improves probabilistic predictions.

Whereas in the case of square loss the DIR method often produces values comparable to VA and SVA, under log loss this method fares less well (which is not obvious from [97], which only uses square loss). In all our experiments DIR suffers infinite log loss for at least one test observation, which makes the overall MLE infinite. There are modifications of the DIR method preventing probabilistic predictions in $\{0, 1\}$ (such as those mentioned in [55], Section 3.3), but they are somewhat arbitrary.

Table 4.3 ranks, for each loss function and dataset, the four calibration methods: W

TABLE 4.3: The ranking of the best three methods (among W, VA, SVA, and DIR) for each dataset according to the two loss functions (see the text for details).

	log loss	square loss
Australian	W (JB), VA (LR), SVA (LR)	W (JB), SVA (JB), VA (LR)
Breast	SVA (NB), VA (NB), W (JB)	SVA (NB), VA (SVM), DIR (NB)
Diabetes	VA (LR), SVA (SVM), W (SVM)	VA (SVM), W (LR), SVA (SVM)
Echo	VA (SVM), SVA (NB), W (JB)	VA (NB), SVA (NB), W (SVM)
Hepatitis	VA (SVM), SVA (NB), W (JB)	SVA (NB), VA (NB), DIR (NB)
Ionosphere	SVA (NB), VA (SVM), W (SVM)	SVA (NB), DIR (NB), W (JB)
Labor	SVA (SVM), W (NN), VA (SVM)	W (NB), SVA (SVM), DIR (NB)
Liver	VA (NN), W (JB), SVA (LR)	VA (NN), SVA (NN), W (JB)
Vote	SVA (SVM), W (SVM), VA (J)	W (SVM), SVA (SVM), VA (J)

(none), VA (Venn–Abers), SVA (simplified Venn–Abers), and DIR (direct isotonic regression). Only the first three methods are given (the best, the second best, and the second worst), where the quality of a method is measured by the performance of the best underlying classifier (indicated in parentheses using the abbreviations given in the column titles of Tables 4.1–4.2) for the given method, data set, and loss function. Notice that we are ranking the four calibration methods rather than the 24 combinations of Weka classifiers with calibration methods (e.g., were we ranking the 24 combinations, the entry for log loss and Australian would remain the same but the next entry, for log loss and Breast, would become "SVA (NB), VA (NB), VA (LR)").

For MLE, the best algorithm is VA or SVA for 8 data sets out of 9; for RMSE this is true for 6 data sets out of 9. In all other cases the best algorithm is W rather than DIR. (And as discussed earlier, in the case of log loss the performance of DIR is especially poor.) Therefore, it appears that the most interesting comparisons are between W and VA and between W and SVA.

What is interesting is that VA and SVA perform best on equal numbers of datasets, 4 each in the case of MLE and 3 each in the case of RMSE, despite the theoretical guarantees of validity for the former method (such as Theorem 2). The similar performance of the two methods needs to be confirmed in more extensive empirical studies, but if it is, SVA will be a preferable method because of its greater computational efficiency.

Comparing W and SVA, we can see that SVA performs better than W on 7 data sets out of 9 for MLE, and on 5 data sets out of 9 for RMSE. And comparing W and VA, we can see that VA performs better than W on 6 data sets out of 9 for MLE, and on 5 data sets out of 9 for RMSE. This suggests that SVA might be an improvement of VA not only in computational but also in predictive efficiency (but the evidence for this is very slim).

4.6 Conclusion

This chapter has introduced a new class of Venn predictors thereby extending the domain of applicability of the method. Our experimental results suggest that the Venn–Abers method can potentially lead to better calibrated probabilistic predictions for a variety of datasets and standard classifiers. The method seems particularly suitable in cases where alternative probabilistic predictors produce overconfident but erroneous predictions under an unbounded loss function such as log loss. In addition, the results suggest that an alternative simplified Venn–Abers method can yield similar results while retaining computational efficiency. Unlike the previous methods for improving the calibration of probabilistic predictors, Venn–Abers predictors enjoy theoretical guarantees of validity (shared with other Venn predictors).

Chapter 5

Large-scale probabilistic predictors

This chapter studies theoretically and empirically a method of turning machine learning algorithms into probabilistic predictors that, as the Venn–Abers predictors described in the preceding chapter, automatically enjoy a property of validity (perfect calibration) but are computationally more efficient. The price to pay for perfect calibration is that these probabilistic predictors produce imprecise (in practice, almost precise for large data sets) probabilities. When these imprecise probabilities are merged into precise probabilities, the resulting predictors, while losing the theoretical property of perfect calibration, are shown to be consistently more accurate than the existing methods in empirical studies.

5.1 Introduction

Prediction algorithms studied in this chapter belong to a class of Venn–Abers predictors, introduced in the preceding chapter and reported in [85]. As described in the introductory section of this thesis, the main desiderata for Venn ([84], Chapter 2) predictors are *validity*, *predictive efficiency* and *computational efficiency*. Whereas the standard and the special Venn-Abers predictors enjoy the first two, their computational efficiency is relatively weak compared with many standard machine learning algorithms. This chapter introduces two computationally efficient versions of Venn–Abers predictors, which we refer to as inductive Venn–Abers predictors (IVAPs) and cross-Venn–Abers predictors (CVAPs). The ways in which they achieve the three desiderata above are:

- Validity (in the form of perfect calibration) is satisfied by IVAPs automatically, and the experimental results reported in this chapter suggest that it is inherited by

CVAPs.

- Predictive efficiency is determined by the predictive efficiency of the underlying learning algorithms (so that the full arsenal of methods of modern machine learning can be brought to bear on the prediction problem at hand).

- Computational efficiency is, again, determined by the computational efficiency of the underlying algorithm; the computational overhead of extracting probabilistic predictions consists of sorting (which takes time $O(n \log n)$, where n is the number of observations) and other computations taking time $O(n)$.

The rest of this chapter is organised as follows: in Sections 5.2 and 5.3 we introduce IVAPs and CVAPs, respectively. Section 5.4 is devoted to minimax ways of merging imprecise probabilities into precise probabilities and thus making IVAPs and CVAPs precise probabilistic predictors.

In order to illustrate the applicability of this method we again concentrate on binary classification problems, in which the objects to be classified are labelled as 0 or 1. Just like VAPs, as precise probabilistic predictors, IVAPs and CVAPs are ways of converting the scores for test objects output by underlying machine learning algorithms into numbers in the range $[0, 1]$ that can serve as probabilities. Section 5.5 compares two existing alternative calibration methods, Platt's [59] and the method based on isotonic regression [96], with IVAPs and CVAPs theoretically. Section 5.6 is devoted to experimental comparisons and shows that CVAPs consistently outperform the two existing methods.

5.2 Inductive Venn–Abers predictors (IVAPs)

In this chapter we consider data sequences (usually loosely referred to as sets) consisting of *observations* $z = (x, y)$, each observation consisting of an *object* x and a *label* $y \in \{0, 1\}$; we only consider binary labels. We are given a training set whose size will be denoted l.

This section introduces inductive Venn–Abers predictors. Our main concern is how to implement them efficiently, but as functions, an IVAP is defined in terms of a scoring algorithm (see the last paragraph of the previous section) as follows:

- Divide the training set of size l into two subsets, the *proper training set* of size m and the *calibration set* of size k, so that $l = m + k$.
- Train the scoring algorithm on the proper training set.
- Find the scores $s_1, \dots, s_k$ of the calibration objects $x_1, \dots, x_k$.
- When a new test object x arrives, compute its score s. Fit isotonic regression to $(s_1, y_1), \dots, (s_k, y_k), (s, 0)$ obtaining a function f_0. Fit isotonic regression to $(s_1, y_1), \dots, (s_k, y_k), (s, 1)$ obtaining a function f_1. The multiprobability prediction for the label y of x is the pair $(p_0, p_1) := (f_0(s), f_1(s))$ (intuitively, the prediction is that the probability that $y = 1$ is either $f_0(s)$ or $f_1(s)$).

Notice that the multiprobability prediction (p_0, p_1) output by an IVAP always satisfies $p_0 < p_1$, and so p_0 and p_1 can be interpreted as the lower and upper probabilities, respectively; in practice, they are close to each other for large training sets.

First we state formally the property of validity of IVAPs (adapting the approach of Theorem 2, Chapter 4 to IVAPs). A random variable P taking values in $[0, 1]$ is *perfectly calibrated* (as a predictor) for a random variable Y taking values in $\{0, 1\}$ if $\mathbb{E}(Y \mid P) = P$ a.s. A *selector* is a random variable taking values in $\{0, 1\}$. As a general rule, in this chapter random variables are denoted by capital letters (e.g., X are random objects and Y are random labels).

Proposition 3. *Let (P_0, P_1) be an IVAP's prediction for X based on a training sequence $(X_1, Y_1), \dots, (X_l, Y_l)$. There is a selector S such that P_S is perfectly calibrated for Y provided the random observations $(X_1, Y_1), \dots, (X_l, Y_l), (X, Y)$ are i.i.d.*

Our next proposition concerns the computational efficiency of IVAPs; both propositions will be proved later in the section.

Proposition 4. *Given the scores $s_1, \dots, s_k$ of the calibration objects, the prediction rule for computing the IVAP's predictions can be computed in time $O(k \log k)$ and space $O(k)$. Its application to each test object takes time $O(\log k)$. Given the sorted scores of the calibration objects, the prediction rule can be computed in time and space $O(k)$.*

Proofs of both statements rely on the geometric representation of isotonic regression as the slope of the GCM (greatest convex minorant) of the CSD (cumulative sum diagram): see [13], pages 9–13 (especially Theorem 1.1). To make our exposition more self-contained, we define both GCM and CSD below.

First we explain how to fit isotonic regression to $(s_1, y_1), \dots, (s_k, y_k)$ (without necessarily assuming that s_i are the calibration scores and y_i are the calibration labels, which will be needed to cover the use of isotonic regression in IVAPs). We start from sorting all scores $s_1, \dots, s_k$ in the increasing order and removing the duplicates. (This is the most computationally expensive step in our calibration procedure, $O(k \log k)$ in the worst case.) Let $k' \leq k$ be the number of distinct elements among $s_1, \dots, s_k$, i.e., the cardinality of the set $\{s_1, \dots, s_k\}$. Define s'_j, $j = 1, \dots, k'$, to be the jth smallest element of $\{s_1, \dots, s_k\}$, so that $s'_1 < s'_2 < \dots < s'_{k'}$. Define $w_j := \left|\left\{i = 1, \dots, k : s_i = s'_j\right\}\right|$ to be the number of times s'_j occurs among $s_1, \dots, s_k$. Finally, define

$$y'_j := \frac{1}{w_j} \sum_{i=1,\dots,k: s_i = s'_j} y_i$$

to be the average label corresponding to $s_i = s'_j$.

The *CSD* of $(s_1, y_1), \dots, (s_k, y_k)$ is the set of points

$$P_i := \left(\sum_{j=1}^{i} w_j, \sum_{j=1}^{i} y'_j w_j \right), \quad i = 0, 1, \dots, k'; \tag{5.1}$$

in particular, $P_0 = (0, 0)$. The *GCM* is the greatest convex minorant of the CSD. The value at s'_i, $i = 1, \dots, k'$, of the *isotonic regression* fitted to $(s_1, y_1), \dots, (s_k, y_k)$ is defined to be the slope of the GCM between $\sum_{j=1}^{i-1} w_j$ and $\sum_{j=1}^{i} w_j$; the values at other s are somewhat arbitrary (namely, the value at $s \in (s'_i, s'_{i+1})$ can be set to anything between the left and right slopes of the GCM at $\sum_{j=1}^{i} w_j$) but are not needed for our purposes (unlike in the standard use of isotonic regression in machine learning, [96]): e.g., $f_1(s)$ is the value of the isotonic regression fitted to a sequence that already contains $(s, 1)$.

Proof of Proposition 3. Set $S := Y$. The statement of the proposition even holds conditionally on knowing the values of $(X_1, Y_1), \dots, (X_m, Y_m)$ and the multiset

$\wr(X_{m+1}, Y_{m+1}), \dots, (X_l, Y_l),\ (X, Y)\int$; this knowledge allows us to compute the scores $\wr s_1, \dots, s_k, s\int$ of the calibration objects $X_{m+1}, \dots, X_l$ and the test object X. The only remaining randomness is over the equiprobable permutations of

$(X_{m+1}, Y_{m+1}), \dots, (X_l, Y_l), (X, Y)$; in particular, (s, Y) is drawn randomly from the multiset $\wr(s_1, Y_{m+1}), \dots, (s_k, Y_l), (s, Y)\int$. It remains to notice that, according to the GCM construction, the average label of the calibration and test observations corresponding to a given value of P_S is equal to P_S. $\square$

The idea behind computing the pair $(f_0(s), f_1(s))$ efficiently is to pre-compute two vectors F^0 and F^1 storing $f_0(s)$ and $f_1(s)$, respectively, for all possible values of s. Let k' and s'_i be as defined above in the case where $s_1, \dots, s_k$ are the calibration scores and $y_1, \dots, y_k$ are the corresponding labels. The vectors F^0 and F^1 are of length k', and for all $i = 1, \dots, k'$ and both $\epsilon \in \{0, 1\}$, F_i^ϵ is the value of $f_\epsilon(s)$ when $s = s'_i$. Therefore, for all $i = 1, \dots, k'$:

- F_i^1 is also the value of $f_1(s)$ when s is just to the left of s'_i;
- F_i^0 is also the value of $f_0(s)$ when s is just to the right of s'_i.

Since f_0 and f_1 can change their values only at the points s'_i, the vectors F^0 and F^1 uniquely determine the functions f_0 and f_1, respectively.

Computational details of IVAPs

Let k', s'_i, and w_i be as defined above in the case where $s_1, \dots, s_k$ and $y_1, \dots, y_k$ are the calibration scores and labels. The *corners* of a GCM are the points on the GCM where the slope of the GCM changes. It is clear that the corners belong to the CSD, and we also add the extreme points (P_0 and $P_{k'}$ in the case of (5.1)) of the CSD to the list of corners.

We will only explain in detail how to compute F^1; the computation of F^0 is analogous and will be explained only briefly. First we explain how to compute F_1^1.

Extend the CSD as defined above (in the case where $s_1, \dots, s_k$ and $y_1, \dots, y_k$ are the calibration scores and labels) by adding the point $P_{-1} := (-1, -1)$. The corresponding GCM will be referred to as the *initial GCM*; it has at most $k' + 2$ corners. Algorithm 4, which operates with a stack S (initially empty), computes the corners; it is a trivial modification of Graham's scan ([35]; [20], Section 33.3). The corners are returned on the stack S, and they are ordered from left to right (P_{-1} being at the bottom of S and $P_{k'}$ at the top). The operator "and" in line 4 is, as usual, short circuiting. The expression "the angle formed by points a, b, and c makes a nonleft (resp. nonright) turn" may be taken to mean that $(b-a) \times (c-b) \leq 0$ (resp. ≥ 0), where $\times$ stands for cross product of planar vectors; this avoids computing angles and divisions (see, e.g., [20], Section 33.1).

Algorithm 4 allows us to compute F_1^1 as the slope of the line between the two bottom corners in S, but this will be done by the next algorithm. Notice that we only need to compute F_i^1 for even i, as

$$F_1^1 = F_2^1 \leq F_3^1 = F_4^1 \leq \cdots \leq F_{2k'-1}^1 = F_{2k'}^1 \leq F_{2k'+1}^1 = 1.$$

The possibilities are: s is to the left of the scores in B_1, s coincides with the scores in B_1, s is between the scores in B_1 and the scores in B_2, s coincides with the scores in B_2,..., s coincides with the scores in B_n, s is to the right of the scores in B_n. We will move over these possibilities from the left to the right.

First we extend the CSD by adding the line connecting $(-1,-1)$ and $(0,0)$ on the left. (At each point the CSD will remain a continuous piece-wise constant function.) We extend the GCM by connecting the point $(-1,-1)$ (which becomes a new corner) to the right-most corner C such that the slope of the line connecting $(-1,-1)$ and that corner is minimal; all corners between $(-1,-1)$ and that right-most corner C cease to be corners. The detailed procedure is: we move through the old corners from left to right computing

Algorithm 4 Initializing the corners for computing F^1

```
1: PUSH($P_{-1}, S$)
2: PUSH($P_0, S$)
3: for $i \in \{1, 2, \ldots, k'\}$ do
4:     while $S$.size $> 1$ and the angle formed by points
               NEXT-TO-TOP($S$), TOP($S$), and $P_i$
               makes a nonleft turn do
5:         POP($S$)
6:     PUSH($P_i, S$)
7: return $S$
```

Algorithm 5 Computing F^1

```
1:  while $\neg$STACK-EMPTY($S$) do
2:      PUSH(POP($S$), $S'$)
3:  for $i \in \{1, 2, \dots, k'\}$ do
4:      set $F^1_i$ to the slope of
            $\overrightarrow{\text{TOP}(S'), \text{NEXT-TO-TOP}(S')}$
5:      $P_{i-1} = P_{i-2} + P_i - P_{i-1}$
6:      if $P_{i-1}$ is at or above $\overrightarrow{\text{TOP}(S'), \text{NEXT-TO-TOP}(S')}$ then
7:          continue
8:      POP($S'$)
9:      while $S'$.size $> 1$ and the angle formed by points
            $P_{i-1}$, TOP($S'$), and NEXT-TO-TOP($S'$)
            makes a nonleft turn do
10:         POP($S'$)
11:     PUSH($P_{i-1}$, $S'$)
12: return $F^1$
```

the slope of the line between $(-1,-1)$ and the current corner; the slope first decreases and then starts increasing; we define C to be the last corner for which the slope takes its minimal value (and there will often be only one corner at which the minimum is attained).

The rest of the procedure for computing the vector F^1 is shown as Algorithm 5. The main data structure in Algorithm 5 is a stack S', which is initialized (in lines 1–2) by putting in it all corners of the initial GCM in reverse order as compared with S (so that $P_{-1} = (-1,-1)$ is initially at the top of S').

At each point in the execution of Algorithm 5 we will have a length-1 *active interval* and the *active corner*, which will nearly always be at the top of the stack S'. The initial CSD can be visualized by connecting each pair of adjacent points: P_{-1} and P_0, P_0 and P_1, etc. It stretches over the interval $[-1, k']$ of the horizontal axis; the subinterval $[-1, 0]$ corresponds to the test score s (assumed to be to the left of all s'_i) and each subinterval $\left[\sum_{j=1}^{i-1} w_j, \sum_{j=1}^{i} w_j\right]$ corresponds to the calibration score s'_i, $i = 1, \dots, k'$. The active corner is initially at $P_{-1} = (-1,-1)$; the corners to the left of the active corner are irrelevant and ignored (not remembered in S'). The active interval is always between the first coordinate of TOP(S') and the first coordinate of NEXT-TO-TOP(S'). At each iteration $i = 1, \dots, k'$ of the main loop 3–11 we are computing F^1_i, i.e., $f_1(s)$ for the situation where s is between s'_{i-1} and s'_i (meaning to the left of s'_1 if $i = 1$), and after that we swap the active interval (corresponding to s) and the interval corresponding to s'_i; of course, after swapping pieces of CSD are adjusted vertically in order to make the CSD as a whole continuous.

At the beginning of each iteration i of the loop 3–11 we have the CSD

$$P_{-1}, P_0, P_1, \ldots, P_{k'} \tag{5.2}$$

corresponding to

$$\text{the points } s'_1, \ldots, s'_{i-1}, s, s'_i, s'_{i+1}, \ldots, s'_{k'}$$
$$\text{with the weights } w_1, \ldots, w_{i-1}, 1, w_i, w_{i+1}, \ldots, w_{k'}$$

(respectively); the active interval is the projection of $\overrightarrow{P_{i-2}, P_{i-1}}$ (onto the horizontal axis, here and later). At the end of that iteration we have the CSD which looks identical to (5.2) but in fact contains a different point P_{i-1} (cf. line 5 of the algorithm) and corresponds to

$$\text{the points } s'_1, \ldots, s'_{i-1}, s'_i, s, s'_{i+1}, \ldots, s'_{k'}$$
$$\text{with the weights } w_1, \ldots, w_{i-1}, w_i, 1, w_{i+1}, \ldots, w_{k'}$$

(respectively); the active interval becomes the projection of $\overrightarrow{P_{i-1}, P_i}$. To achieve this, in line 5 we redefine P_{i-1} to be the reflection of the old P_{i-1} across the mid-point $(P_{i-2} + P_i)/2$. The stack S' always consists of corners of the GCM of the current CSD, and it contains all the corners to the right of the active interval (plus one more corner, which is the active corner).

At each iteration i of the loop 3–11:

- We report the slope of the GCM over the active interval as F_i^1 (line 4).
- We then swap the fragments of the CSD corresponding to the active interval and to s'_i leaving the rest of the CSD intact. This way the active interval moves to the right (from the projection of $\overrightarrow{P_{i-2}, P_{i-1}}$ to the projection of $\overrightarrow{P_{i-1}, P_i}$).

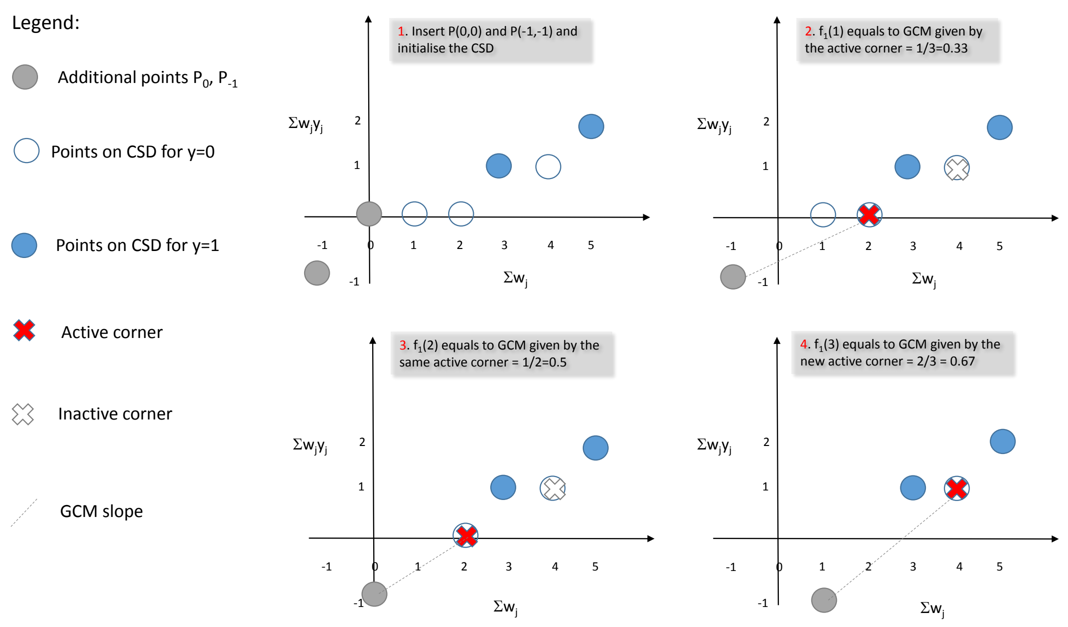
Legend:
Additional points P_0, P_{-1}
Points on CSD for y=0
Points on CSD for y=1
Active corner
Inactive corner
GCM slope
1. Insert P(0,0) and P(-1,-1) and initialise the CSD
2. $f_1(1)$ equals to GCM given by the active corner = 1/3=0.33
3. $f_1(2)$ equals to GCM given by the same active corner = 1/2=0.5
4. $f_1(3)$ equals to GCM given by the new active corner = 2/3 = 0.67
$\Sigma w_j y_j$
Σw_j

FIGURE 5.1: A graphical representation of Algorithms 4 and 5.

- If the point P_{i-1} above the left end-point of the active interval is above (or at) the GCM, move to the next iteration of the loop. (The active corner does not change.) The rest of this description assumes that P_{i-1} is strictly below.
- Make P_{i-1} the active corner. Redefine the GCM to the right of the active corner by connecting the active corner to the right-most corner C such that the slope of the line connecting the active corner and that corner is minimal; all the corners between the active corner and that right-most corner C are then forgotten.

Lemma 3. *The worst-case computation time of Algorithms 4 and 5 is $O(k')$.*

Proof. In the case of Algorithm 4, see [20], Section 33.3. In the case of Algorithm 5, it suffices to notice that the total number of iterations for the **while** loop does not exceed the total number of elements pushed onto S' (since at each iteration we pop an element off S'); and the total number of elements pushed onto S' is at most k' (in the first **for** loop) plus k' (in the second **for** loop). □

A graphical illustration of Algorithms 4 and 5 is shown in Figure 5.1. Algorithms 6 and 7 respectively below are counterparts of Algorithms 4 and 5 for computing F^0. In

Algorithm 6 Initializing the corners for computing F^0

```
1: Push($P_{k'+1}, S$)
2: Push($P_{k'}, S$)
3: for $i \in \{k'-1, k'-2, \dots, 0\}$ do
4:     while $S$.size $> 1$ and the angle formed by points Next-To-Top($S$),
            Top($S$), and $P_i$ makes a nonright turn do
5:         Pop($S$)
6:     Push($P_i, S$)
7: return $S$
```

Algorithm 7 Computing F^0

```
1: while ¬Stack-Empty($S$) do
2:     Push(Pop($S$), $S'$)
3: for $i \in \{k', k'-1, \dots, 1\}$ do
4:     set $F^0_i$ to the slope of $\overrightarrow{\text{Top}(S'), \text{Next-To-Top}(S')}$
5:     $P_i = P_{i-1} + P_{i+1} - P_i$
6:     if $P_i$ is at or above $\overrightarrow{\text{Top}(S'), \text{Next-To-Top}(S')}$ then
7:         continue
8:     Pop($S'$)
9:     while $S'$.size $> 1$ and the angle formed by points $P_i$,
            Top($S'$), and Next-To-Top($S'$) makes a nonright turn do
10:        Pop($S'$)
11:    Push($P_i, S'$)
12: return $F^0$
```

those algorithms, we do not need the point P_{-1} anymore; however, we need a new point $P_{k'+1} := P_{k'} + (1,1)$. The stacks S and S' that they use are initially empty.

Alternatively, we could use the algorithm for computing F^1 in order to compute F^0, since, for all $i \in \{1, \dots, k'\}$,

$$F^0_i\left(s'_1, \dots, s'_{k'}, w_1, \dots, w_{k'}, y'_1, \dots, y'_{k'}\right) \\ = 1 - F^1_i\left(-s'_1, \dots, -s'_{k'}, w_1, \dots, w_{k'}, 1-y'_1, \dots, 1-y'_{k'}\right),$$

where the dependence on various parameters is made explicit.

After computing F^0 and F^1 we can arrange the calibration scores $s'_1, \dots, s'_{k'}$ into a binary search tree: see Algorithm 8, where F^0_0 is defined to be 0 and $F^1_{k'+1}$ is defined to be 1; we will refer to s'_i as the *keys* of the corresponding nodes (only internal nodes will have keys). Algorithm 8 is in fact more general than what we need: it computes

Algorithm 8 $\mathrm{BST}(a, b)$ (to create the binary search tree, run $\mathrm{BST}(1, k')$)

```
1: if $b = a$ then
2:     construct the binary tree
       whose root has key $s'_a$ and payload $\{F^0_a, F^1_a\}$,
       left child is a leaf with payload $\{F^0_{a-1}, F^1_a\}$,
       and right child is a leaf with payload $\{F^0_a, F^1_{a+1}\}$ return its root
3: else if $b = a + 1$ then
4:     construct the binary tree
       whose root has key $s'_a$ and payload $\{F^0_a, F^1_a\}$,
       left child is a leaf with payload $\{F^0_{a-1}, F^1_a\}$,
       and right child is $\mathrm{BST}(b, b)$ return its root
5: else
6:     $c = \lfloor (a + b)/2 \rfloor$
7:     construct the binary tree
       whose root has key $s'_c$ and payload $\{F^0_c, F^1_c\}$,
       left child is $\mathrm{BST}(a, c - 1)$,
       and right child is $\mathrm{BST}(c + 1, b)$ return its root
```

Algorithm 9 $\mathrm{IVAP}(T', T'', x)$

```
1: set $N$ to the root of the binary search tree and compute the score $s$ of $x$
2: while $N$ is not a leaf do
3:     if $s < \mathrm{key}(N)$ then
4:         set $N$ to $N'$s left child
5:     else if $s > \mathrm{key}(N)$ then
6:         set $N$ to $N'$s right child
7:     else
8:         return $\mathrm{payload}(N)$
9: return $\mathrm{payload}(N)$
```

the binary search tree for the scores $s'_a, s'_{a+1}, \dots, s'_b$ for $a \le b$; therefore, we need to run $\mathrm{BST}(1, k')$. The size of the binary search tree is $2k' + 1$; k' of its nodes are internal nodes corresponding to different values of s'_i, $i = 1, \dots, k'$, and the other $k' + 1$ of its nodes are leaves corresponding to the $k' + 1$ intervals formed by the points $s'_1, \dots, s'_{k'}$.

Once we have the binary search tree it is easy to compute the prediction for a test object x in time logarithmic in k': see Algorithm 9, which passes x through the tree and uses N to denote the current node. Formally, we give the test object x, the proper training set T', and the calibration set T'' as the inputs of Algorithm 9; however, the algorithm uses for prediction the binary search tree built from T' and T'', and the bulk of work is done in Algorithms 4–8.

The worst-case computational complexity of the overall procedure involves the following components:

- Training the algorithm on the proper training set, computing the scores of the calibration objects, and computing the scores of the test objects; at this stage the computation time is determined by the underlying algorithm.
- Sorting the scores of the calibration objects takes time $O(k \log k)$.
- Running our procedure for pre-computing f_0 and f_1 takes time $O(k)$ (by Lemma 3).
- Processing each test object takes an additional time of $O(\log k)$ (using binary search).

In principle, using binary search does not require an explicit construction of a binary search tree (cf. [20], Exercise 2.3-5), but once we have a binary search tree we can easily transform it into a red-black tree, which allows us to add new observations to (and remove old observations from) the calibration set in time $O(\log k)$ ([20], Chapter 13).

5.3 Cross Venn–Abers predictors (CVAPs)

A CVAP is just a combination of K IVAPs, where K is the parameter of the algorithm. It is described as Algorithm 10, where $\text{IVAP}(A, B, x)$ stands for the output of IVAP applied to A as proper training set, B as calibration set, and x as test object, and GM stands for geometric mean (so that $\text{GM}(p_1)$ is the geometric mean of $p_1^1, \dots, p_1^K$ and $\text{GM}(1 - p_0)$ is the geometric mean of $1 - p_0^1, \dots, 1 - p_0^K$). The folds should be of approximately equal size, and usually the training set is split into folds at random (although we choose contiguous folds in Section 5.6 to facilitate reproducibility). One way to obtain a random assignment of the training observations to folds (see line 1) is to start from a regular array in which the first l_1 observations are assigned to fold 1, the following l_2 observations are assigned to fold 2, up to the last l_K observations which are assigned to fold K, where $|l_k - l/K| < 1$ for all k, and then to apply a random permutation. Remember that the procedure RANDOMIZE-IN-PLACE ([20], Section 5.3) can do the last step in time $O(l)$. See

Algorithm 10 $\mathrm{CVAP}(T, x)$

```
1: split the training set $T$ into $K$ folds $T_1, \dots, T_K$
2: for $k \in \{1, \dots, K\}$ do
3:     $(p_0^k, p_1^k) := \mathrm{IVAP}(T \setminus T_k, T_k, x)$
4: return $\mathrm{GM}(p_1)/(\mathrm{GM}(1-p_0) + \mathrm{GM}(p_1))$
```

the next section for a justification of the expression $\mathrm{GM}(p_1)/(\mathrm{GM}(1-p_0)+\mathrm{GM}(p_1))$ used for merging the IVAPs′ outputs.

5.4 Making probability predictions out of multiprobability ones

In CVAP (Algorithm 10) we merge the K multiprobability predictions output by K IVAPs. In this section we design a minimax way for merging them, essentially following [85]. For the log-loss function the result is especially simple, $\mathrm{GM}(p_1)/(\mathrm{GM}(1-p_0) + \mathrm{GM}(p_1))$.

The deficiency of guaranteed calibration: $\log(\mathrm{GM}(1-p_0)+\mathrm{GM}(p_1)) \in [0, 1]$ (for binary log). We need to check how small it is.

Remark 4. Notice that the probability interval $(1-\mathrm{GM}(1-p_0), \mathrm{GM}(p_1))$ (formally, a pair of numbers) is narrower than the corresponding interval for the arithmetic means; this follows from the fact that a geometric mean never exceeds the corresponding arithmetic mean and that we always have $p_0 < p_1$.

Let us check that $\mathrm{GM}(p_1)/(\mathrm{GM}(1-p_0) + \mathrm{GM}(p_1))$ is indeed the minimax expression under log loss. Suppose the pairs of lower and upper probabilities to be merged are $(p_0^1, p_1^1), \dots, (p_0^K, p_1^K)$ and the merged probability is p. The extra cumulative loss suffered by p over the correct members $p_1^1, \dots, p_1^K$ of the pairs when the true label is 1 is

$$\log \frac{p_1^1}{p} + \dots + \log \frac{p_1^K}{p}, \tag{5.3}$$

and the extra cumulative loss of p over the correct members of the pairs when the true label is 0 is

$$\log \frac{1-p_0^1}{1-p} + \dots + \log \frac{1-p_0^K}{1-p}. \tag{5.4}$$

Equalizing the two expressions we obtain

$$\frac{p_1^1 \cdots p_1^K}{p^K} = \frac{(1-p_0^1)\cdots(1-p_0^K)}{(1-p)^K},$$

which gives the required minimax expression for the merged probability (since (5.3) is decreasing and (5.4) is increasing in p).

In the case of the Brier loss function, we solve the linear equation

$$(1-p)^2 - (1-p_1^1)^2 + \cdots + (1-p)^2 - (1-p_1^K)^2 = p^2 - (p_0^1)^2 + \cdots + p^2 - (p_0^K)^2$$

in p; the result is

$$p = \frac{1}{K}\sum_{k=1}^{K}\left(p_1^k + \frac{1}{2}(p_0^k)^2 - \frac{1}{2}(p_1^k)^2\right).$$

This expression is more natural than it looks: see [85], the discussion after (11); notice that it reduces to arithmetic mean when $p_0 = p_1$.

The argument above ("conditioned" on the proper training set) is also applicable to IVAP, in which case we need to set $K := 1$; the probability predictor obtained from an IVAP by replacing (p_0, p_1) with $p := p_1/(1-p_0+p_1)$ will be referred to as the *log-minimax IVAP*. (And CVAP is log-minimax by definition.)

5.5 Comparison with other calibration methods

The two alternative calibration methods that we consider in this chapter are Platt's [59] and isotonic regression [96].

5.5.1 Platt's method

As described in Chapter 2, Platt's [59] method uses sigmoids

$$g(s) := \frac{1}{1+\exp(As+B)},$$

where $A < 0$ and B are parameters, to calibrate the scores. Platt discusses two approaches:

- run the scoring algorithm and fit the parameters A and B on the full training set,
- or run the scoring algorithm on a subset (called the proper training set in this chapter) and fit A and B on the rest (the calibration set).

Platt recommends the second approach, especially that he is interested in SVM, and for SVM the scores for the training set tend to cluster around ± 1. (In fact, this is also true for the calibration scores, as discussed below.)

Platt's recommended method of fitting A and B is

$$-\sum_{i=1}^{k} (t_i \log p_i + (1 - t_i) \log(1 - p_i)) \to \min, \tag{5.5}$$

where, in the simplest case, $t_i := y_i$ are the labels of the calibration observations (so that (5.5) minimizes the log loss on the calibration set). To obtain even better results, Platt recommends regularization:

$$t_i = t_+ := \frac{k_+ + 1}{k_+ + 2} \tag{5.6}$$

for the calibration observations labelled 1 (if there are k_+ of them) and

$$t_i = t_- := \frac{1}{k_- + 2} \tag{5.7}$$

for the calibration observations labelled 0 (if there are k_- of them). We can see from (5.6) and (5.7) that the predictions of Platt's predictor are always in the range

$$\left(\frac{1}{k_- + 2}, \frac{k_+ + 1}{k_+ + 2} \right). \tag{5.8}$$

where k_- is the number of calibration observations labelled 0 and k_+ is the number of calibration observations labelled 1. It is interesting that the predictions output by the log-minimax IVAP are in the same range (except that the end-points are now allowed): see [90].

Let us check that the predictions output by the log-minimax IVAP are in the same range as those for Platt's method (except that the end-points are now allowed):

Lemma 4. *In the case of IVAP, $p_1 \geq 1/(k_- + 1)$ and $p_0 \leq 1 - 1/(k_+ + 1)$, where k_- and k_+ are the numbers of positive and negative observations in the calibration set, respectively. In the case of log-minimax IVAP, $p \in [1/(k_- + 2), 1 - 1/(k_+ + 2)]$ (i.e., p is in the closure of* (5.8)*). In the case of CVAP, $p \in [1/(k + 2), 1 - 1/(k + 2)]$, where k is the size of the largest fold.*

Proof. The statement about IVAP is obvious, and we will only check that it implies the two other statements. For concreteness, we will consider the lower bounds. The lower bound $1/(k_- + 2)$ for log-minimax IVAP can be deduced from $p_1 \geq 1/(k_- + 1)$ using the isotonicity of $t/(c + t)$ in $t > 0$ for $c > 0$:

$$\frac{p_1}{(1 - p_0) + p_1} \geq \frac{1/(k_- + 1)}{(1 - p_0) + 1/(k_- + 1)} \geq \frac{1/(k_- + 1)}{1 + 1/(k_- + 1)} = \frac{1}{k_- + 2}.$$

In the same way the lower bound $1/(k + 2)$ for CVAP follows from $\mathrm{GM}(p_1) \geq 1/(k + 1)$:

$$\frac{\mathrm{GM}(p_1)}{\mathrm{GM}(1 - p_0) + \mathrm{GM}(p_1)} \geq \frac{1/(k + 1)}{\mathrm{GM}(1 - p_0) + 1/(k + 1)} \geq \frac{1/(k + 1)}{1 + 1/(k + 1)} = \frac{1}{k + 2}. \qquad \square$$

It is clear that the end-points of the interval (5.8) can be approached arbitrarily closely in the case of Platt's predictor and attained in the case of IVAPs.

The main disadvantage of Platt's method is that the optimal calibration curve g is quite often far from being a sigmoid; and if the training set is very big, we will suffer, since in this case we can learn the best shape of the calibrator g. This is particularly serious in asymptotics as the amount of data tends to infinity.

Zhang [99] (Section 3.3) observes that in the case of SVM and universal [70] kernels the scores tend to cluster around ± 1 at "non-trivial" objects, i.e., objects that are labelled 1 with non-trivial (not close to 0 or 1) probability. This means that any sigmoid will be a poor calibrator unless the prediction problem is very easy. Formally, we have the following statement (a trivial corollary of known results), which uses the notation $\eta(x)$ for the conditional probability that the label of an object $x \in \mathbf{X}$ is 1 and assumes that the labels take values in $\{-1, 1\}$, $y_i \in \{-1, 1\}$ (rather than $y_i \in \{0, 1\}$, as in the rest of this chapter).

Proposition 5. *Suppose that the probability of each of the events $\eta(X) = 0$, $\eta(X) = 1/2$, and $\eta(X) = 1$ is 0. Let f_m be the SVM for a training set of size m, i.e., the solution to the optimization problem*

$$C_m \|f\|_H^2 + \sum_{i=1}^{m} \phi(f(x_i)y_i) \to \min, \tag{5.9}$$

where $\phi(v) := (1-v)^+$ and H is a universal RKHS ([71], Definition 4.52).

H is separable automatically: see Lemma 4.33 in [71]. H consists of bounded functions automatically (since the elements of H are continuous). The assumptions in [71] (Theorem 8.1) are weaker: e.g., it is enough to assume that H is dense in $L_1(Q_{\mathbf{X}})$, Q being the data-generating probability measure on $\mathbf{X} \times \{0,1\}$ and $Q_{\mathbf{X}}$) being its marginal on $\mathbf{X}$.

As $m \to \infty$,

$$f_m(X) \to f(X) := \begin{cases} -1 & \text{if } \eta(X) \in [0, 1/2] \\ 1 & \text{if } \eta(X) \in (1/2, 1] \end{cases}$$

in probability provided $C_m \to \infty$ and $C_m = o(m)$.

Proof. This follows immediately from Theorem 4.4 in [99] for a natural class of universal kernels related to neural networks. In general, see the proof of Theorem 8.1 in [71]. □

The intuition behind the SVM decision values clustering around ± 1 is very simple. SVM solves the optimization problem (5.9); asymptotically as $m \to \infty$ and under natural assumptions (such as $C_m \to \infty$ and $C_m = o(m)$), this solves

$$\mathbb{E}\,\phi(f(X)Y) \to \min.$$

We can optimize separately for different values of $\eta(x)$. Given $\eta(x) = \eta^*$, we have the optimization problem

$$\eta^*\phi(f) + (1-\eta^*)\phi(-f) \to \min,$$

whose solutions are

$$f(x) \in \begin{cases} (-\infty, -1] & \text{if } \eta(x) = 0 \\ \{-1\} & \text{if } \eta(x) \in (0, 1/2) \\ [-1, 1] & \text{if } \eta(x) = 1/2 \\ \{1\} & \text{if } \eta(x) \in (1/2, 1) \\ [1, \infty) & \text{if } \eta(x) = 1. \end{cases}$$

As a function of f,

$$\eta^* \phi(f) + (1 - \eta^*) \phi(-f)$$

is a continuous function which is equal to $\eta^*(1-v)$ for $v \in (-\infty, -1]$, equal to $(1-\eta^*)(1+v)$ for $v \in [1, \infty)$, and linear for $v \in [-1, 1]$ (and these conditions determine the function).

Assuming that the probability of each of the events $\eta(X) = 0$, $\eta(X) = 1/2$, and $\eta(X) = 1$ is 0, it is easy to check that asymptotically the best achievable excess log loss of a sigmoid over the Bayes algorithm is

$$\mathbb{E}\Big(\mathrm{KL}\left(\eta \mid\mid \mathbb{E}(\eta \mid \eta > 1/2)\right) \mathbf{1}_{\eta>1/2} + \mathrm{KL}\left(\eta \mid\mid \mathbb{E}(\eta \mid \eta < 1/2)\right) \mathbf{1}_{\eta<1/2}\Big), \tag{5.10}$$

where KL is Kullback–Leibler divergence defined in terms of base 2 logarithm $\log_2$, and the conditional expectation $\mathbb{E}(\eta \mid E)$ is defined to be $\mathbb{E}(\eta \, \mathbf{1}_E) / \mathbb{P}(E)$.

Indeed, the optimal prediction for $\eta > 1/2$ is $\mathbb{E}(\eta \mid \eta > 1/2)$ and the optimal prediction for $\eta < 1/2$ is $\mathbb{E}(\eta \mid \eta < 1/2)$. Let us check, e.g., the first statement. We are to minimize over $p \in (0, 1)$ the integral of

$$-\eta \log p + (1 - \eta) \log(1 - p)$$

over the set $\eta > 1/2$. Set $C := \int_{\eta>1/2} \eta \, \mathrm{d}P$ and $D := P(\eta > 1/2)$. So we are to maximize

$$C \log p + (D - C) \log(1 - p).$$

Differentiating and solving the equation

$$\frac{C}{p} - \frac{D-C}{1-p} = 0$$

we obtain

$$p = C/D = \mathbb{E}(\eta \mid \eta > 1/2).$$

On the other hand, there are no apparent obstacles to it approaching 0 in the case of isotonic regression, considered in the next subsection.

5.5.2 Isotonic regression

There are two standard uses of isotonic regression: we can train the scoring algorithm using what we call a proper training set, and then use the scores of the observations in a disjoint calibration (also called validation) set for calibrating the scores of test objects (as in [15]); alternatively, we can train the scoring algorithm on the full training set and also use the full training set for calibration (it appears that this was done in [96]). Alternatively, we could use a cross-validation scheme similar to CVAPs. In both cases, however, we can expect to get an infinite log loss when the test set becomes large enough (see [90]). Indeed, suppose that we have fixed proper training and calibration sets (not necessarily disjoint, so that both cases mentioned above are covered) such that the score $s(X)$ of a random object X is below the smallest score of the calibration objects with a positive probability; suppose also that the distribution of the label of a random observation is concentrated at 0 with probability zero. Under these realistic assumptions the probability that the average log loss on the test set is ∞ can be made arbitrarily close to one by making the size of the test set large enough: indeed, with a high probability there will be an observation (x, y) in the test set such that the score $s(x)$ is below the smallest score of the calibration objects but $y = 1$; the log loss on such an observation will be infinite.

The presence of regularization is an advantage of Platt's method: e.g., it never suffers an infinite loss when using the log loss function. There is no standard method of regularization for isotonic regression, and we do not apply one.

5.6 Experimental results

The main loss function (cf., e.g., [83]) that we use in our empirical studies is the *log loss*

$$\lambda_{\log}(p, y) := \begin{cases} -\log p & \text{if } y = 1 \\ -\log(1-p) & \text{if } y = 0, \end{cases} \tag{5.11}$$

where $\log$ is binary logarithm, $p \in [0, 1]$ is a probability prediction, and $y \in \{0, 1\}$ is the true label. Another popular loss function is the *Brier loss*

$$\lambda_{\text{Br}}(p, y) := 4(y - p)^2. \tag{5.12}$$

We choose the coefficient 4 in front of $(y - p)^2$ in (5.12) and the base 2 of the logarithm in (5.11) in order for the minimax no-information predictor that always predicts $p := 1/2$ to suffer loss 1. An advantage of the Brier loss function is that it still makes it possible to compare the quality of prediction in cases when prediction algorithms (such as isotonic regression) give a categorical but wrong prediction (and so are simply regarded as infinitely bad when using log loss).

In the multi-class case we assume that the label space $\mathbf{Y}$ is finite and consider probability predictions that are probability measures on $\mathbf{Y}$; for example, a prediction $p \in [0, 1]$ output by a CVAP is re-interpreted as the probability measure P on $\{0, 1\}$ such that $P(\{1\}) = p$. As in the previous chapter the main loss function that we use is the *log loss*

$$\lambda_{\log}(P, y) := -\log_{|\mathbf{Y}|} P(\{y\}), \tag{5.13}$$

where we take the size $|\mathbf{Y}|$ of the label space as the base of the logarithm and the *Brier loss*

$$\lambda_{\text{Br}}(P, y) := \frac{|\mathbf{Y}|}{|\mathbf{Y}| - 1} \sum_{y' \in \mathbf{Y}} \left(1_{y'=y} - P(\{y'\})\right)^2,$$

where the coefficient in front of the sum is chosen in such a way that the minimax no-information predictor that always predicts $1/|\mathbf{Y}|$ suffers loss 1 (this is also the reason for our choice of the base of the logarithm in (5.13)).

The loss of a probability predictor on a test set will be measured by the arithmetic average of the losses it suffers on the test set, namely, by the *mean log loss* (MLL) and the *mean Brier loss* (MBL)

$$\text{MLL} := \frac{1}{n}\sum_{i=1}^{n} \lambda_{\text{log}}(p_i, y_i), \quad \text{MBL} := \frac{1}{n}\sum_{i=1}^{n} \lambda_{\text{Br}}(p_i, y_i), \tag{5.14}$$

where y_i are the test labels and p_i are the probability predictions for them. We will not be checking directly whether various calibration methods produce well-calibrated predictions, since it is well known that lack of calibration increases the loss as measured by loss functions such as log loss and Brier loss (see, e.g., [52] for the most standard decomposition of the latter into the sum of the calibration error and refinement error).

In this section we compare log-minimax IVAPs (i.e., IVAPs whose outputs are replaced by probability predictions, as explained in Section 5.4) and CVAPs with Platt's method [59] and the standard method [96] based on isotonic regression; the latter two will be referred to as "Platt" and "Isotonic" in our tables and figures. (Even though for both IVAPs and CVAPs we use the log-minimax procedure for merging multiprobability predictions, the Brier-minimax procedure leads to virtually identical empirical results.) We use the same underlying algorithms as in [85], namely J48 decision trees (abbreviated to "J48"), J48 decision trees with bagging ("J48 bagging"), logistic regression (sometimes abbreviated to "logistic"), naïve Bayes, neural networks, and support vector machines (SVM), as implemented in Weka [38] (University of Waikato, New Zealand). The underlying algorithms (except for SVM) produce scores in the interval $[0, 1]$, which can be used directly as probability predictions (referred to as "Underlying" in our tables and figures) or can be calibrated using the methods of [59, 96] or the methods proposed in this chapter ("IVAP" or "CVAP" in the tables and figures).

We start our empirical studies with the `adult` data set available from the UCI repository [31] (this is the main data set used in [59] and one of the data sets used in [96]); however, as we will see later, the picture that we observe is typical for other data sets as well. We use the original split of the data set into a training set of $N_{\text{train}} = 32,561$ observations and a test set of $N_{\text{test}} = 16,281$ observations. The results of applying the four calibration methods (plus the vacuous one, corresponding to just using the underlying algorithm) to the six underlying algorithms for this data set are shown in Figure 5.2 which reports results for the log loss (namely, MLL, as defined in (5.14)) and Figure 5.3 for the Brier loss (namely, MBL).

The underlying algorithms are given in the titles of the plots and the calibration methods are represented by different line styles, as explained in the legends. The marks on the horizontal axis are the ratios of the size of the proper training set to the size of the calibration set (except for the label `all`, which will be explained later); in the case of CVAPs, the number K of folds can be expressed as the sum of the two numbers forming the ratio (therefore, column 4:1 corresponds to the standard choice of 5 folds in the method of cross-validation). Missing curves or points on curves mean that the corresponding values either are too big and would squeeze unacceptably the interesting parts of the plot if shown or are infinite (such as many results for isotonic regression and neural networks under log loss). In the case of CVAPs, the training set is split into K equal (or as close to being equal as possible) contiguous folds: the first $\lceil N_{\text{train}}/K \rceil$ training observations are included in the first fold, the next $\lceil N_{\text{train}}/K \rceil$ (or $\lfloor N_{\text{train}}/K \rfloor$) in the second fold, etc. (first $\lceil \cdot \rceil$ and then $\lfloor \cdot \rfloor$ is used unless N_{train} is divisible by K). In the case of the other calibration methods, we used the first $\lceil \frac{K-1}{K} N_{\text{train}} \rceil$ training observation as the proper training set (used for training the scoring algorithm) and the rest of the training observations are used as the calibration set.

In the case of log loss, isotonic regression often suffers infinite losses, which is indicated by the absence of the round marker for isotonic regression; e.g., only one of the log losses for SVM is finite. We are not trying to use ad hoc solutions, such as clipping predictions to the interval $[\epsilon, 1-\epsilon]$ for a small $\epsilon > 0$, since we are also using the bounded Brier loss function. The CVAP lines tend to be at the bottom in all plots; experiments with other data sets also confirm this.

The column `all` in the plots of Figures 5.2 and 5.3 refers to using the full training set as both the proper training set and calibration set. (In our official definition of IVAP we require that the last two sets be disjoint, but in this section we continue to refer to IVAPs modified in this way simply as IVAPs; in [85], such prediction algorithms were

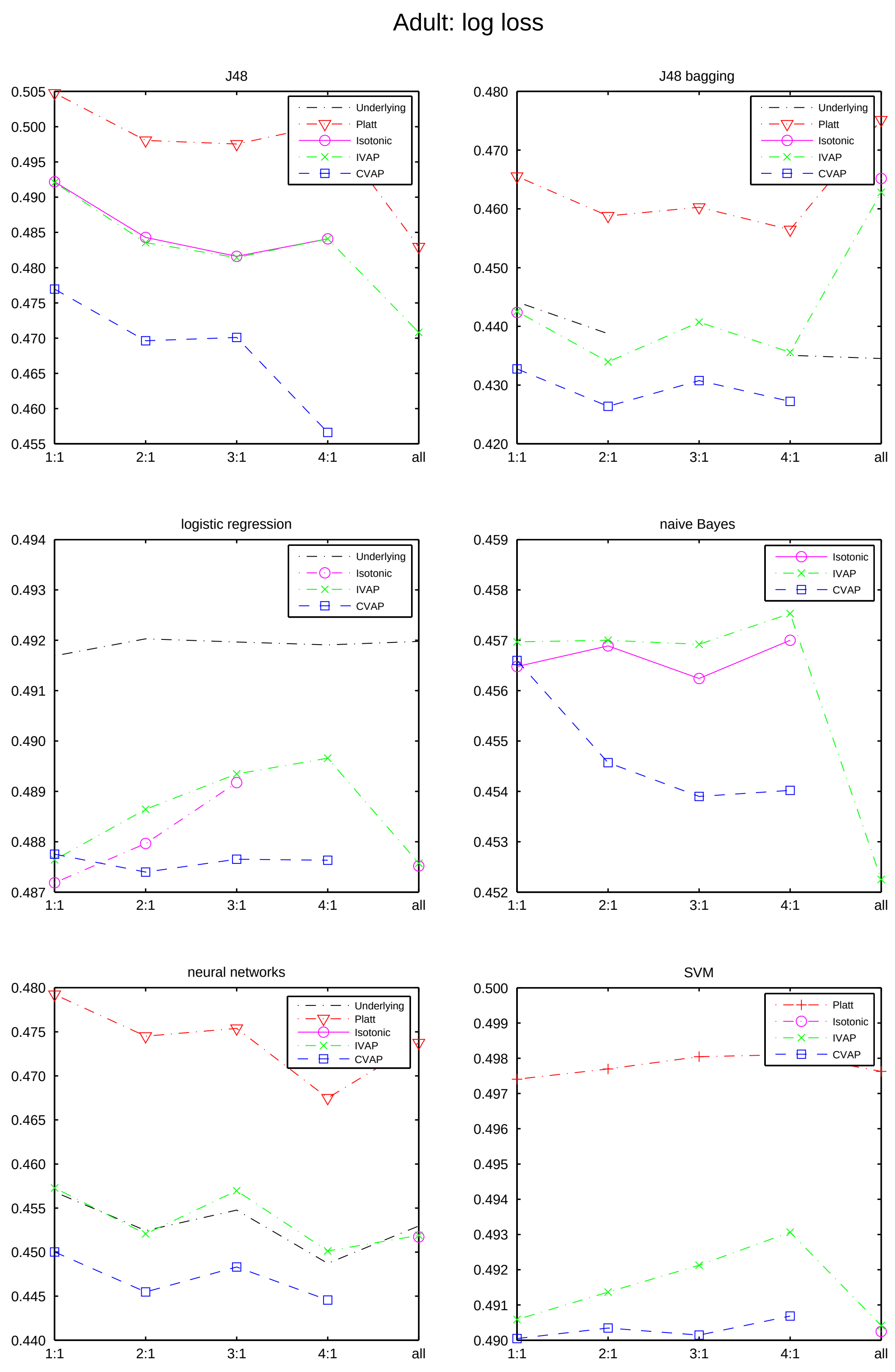

FIGURE 5.2: The log losses of the four calibration methods applied to the six prediction algorithms on the `adult` data set.

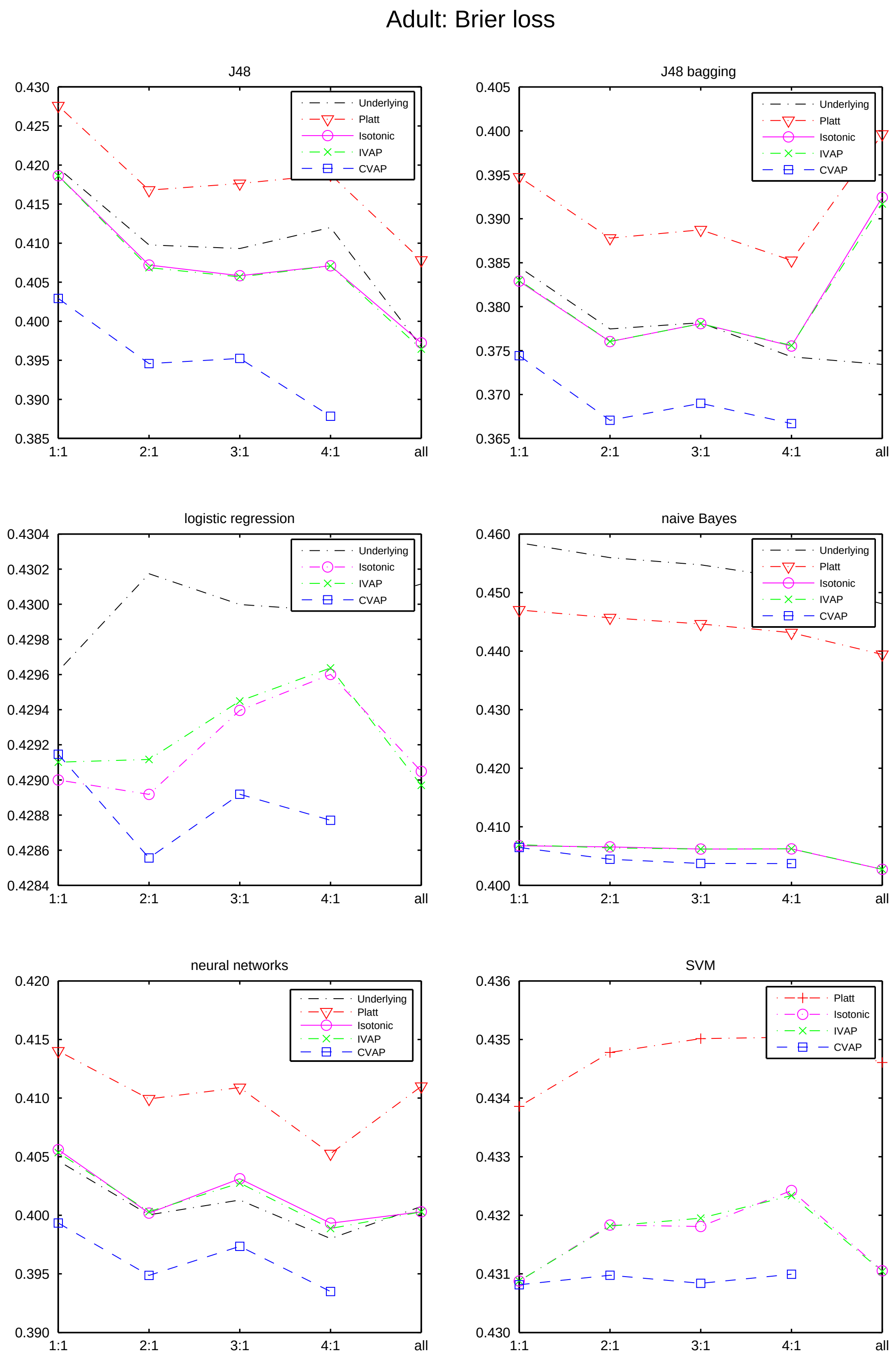

FIGURE 5.3: The analogue of Figure 5.2 for Brier loss.

referred to as SVAPs, simplified Venn–Abers predictors.) Using the full training set as both the proper training set and calibration set might appear naive (and is never used in the extensive empirical study [15]), but it often leads to good empirical results on larger data sets. However, it can also lead to very poor results, as in the case of "J48 bagging" (for IVAP, Platt, and Isotonic), the underlying algorithm that achieves the best performance in Figures 5.2 and 5.3.

A natural question is whether CVAPs perform better than the alternative calibration methods in Figures 5.2 and 5.3 (and our other experiments) because of applying cross-over (in moving from IVAP to CVAP) or because of the extra regularization used in IVAPs. The first reason is undoubtedly important for both loss functions and the second for the log loss function. The second reason plays a smaller role for Brier loss for relatively large data sets (in Figure 5.3 the curves for `Isotonic` and `IVAP` are very close to each other), but IVAPs are consistently better for smaller data sets even when using Brier loss. In Tables 5.1 and 5.2 we apply the four calibration methods and six underlying algorithms to a much smaller training set, namely to the first $5,000$ observations of the `adult` data set as the new training set, following [15]; the first $4,000$ training observations are used as the proper training set, the following $1,000$ training observations as the calibration set, and all other observations (the remaining training and all test observations) are used as the new test set. The results are shown in Tables 5.1 for log loss and 5.2 for Brier loss. They are consistently better for IVAP than for IR (isotonic regression). Results for nine very small data sets are given in Tables 1 and 2 of [85], where the results for IVAP (with the full training set used as both proper training and calibration sets, labelled "SVA" in the tables in [85]) are consistently (in 52 cases out of the 54 using Brier loss) better, usually significantly better, than for isotonic regression (referred to as DIR in the tables in [85]).

The following information might help the reader in reproducing the results (in addition to the code being publicly available [90]). For each of the standard prediction algorithms within Weka that we use, we optimise the parameters by minimising the Brier loss on the calibration set, apart from the column labelled `all`. (We cannot use the log loss since it is often infinite in the case of isotonic regression.) We then use the trained

TABLE 5.1: The log loss for the four calibration methods and six underlying algorithms for a small subset of the `adult` data set

algorithm	Platt	IR	IVAP	CVAP
J48	0.5226	∞	0.5117	0.5102
J48 bagging	0.4949	∞	0.4733	0.4602
logistic	0.5111	∞	0.4981	0.4948
naïve Bayes	0.5534	∞	0.4839	0.4747
neural networks	0.5175	∞	0.5023	0.4805
SVM	0.5221	∞	0.5015	0.4997

TABLE 5.2: The analogue of Table 5.1 for the Brier loss

algorithm	Platt	IR	IVAP	CVAP
J48	0.4463	0.4378	0.4370	0.4368
J48 bagging	0.4225	0.4153	0.4123	0.3990
logistic	0.4470	0.4417	0.4377	0.4342
naïve Bayes	0.4670	0.4329	0.4311	0.4227
neural networks	0.4525	0.4574	0.4440	0.4234
SVM	0.4550	0.4450	0.4408	0.4375

algorithm to generate the scores for the calibration and test sets, which allows us to compute probability predictions using Platt's method, isotonic regression, IVAP, and CVAP. All the scores apart from SVM are already in the $[0, 1]$ range and can be used as probability predictions. In the case of SVM, we use the Weka sequential minimal optimization algorithm with the option "build logistic models", which calibrates the SVM scores into probabilities. We then apply the other calibration methods on top of those probability predictions, which is equivalent to applying them to the original SVM scores.

Most of the parameters are set to their default values, and the only parameters that are optimised are `C` (pruning confidence) for J48 and J48 bagging, `R` (ridge) for logistic regression, `L` (learning rate) and `M` (momentum) for neural networks (`MultilayerPerceptron`), and `C` (complexity constant) for SVM (`SMO`, with the linear kernel); naïve Bayes does not involve any parameters. Notice that none of these parameters are "hyperparameters", in that they do not control the flexibility of the fitted prediction rule directly; this allows us to optimize the parameters on the training set for the `all` column. In the case of CVAPs, we optimise the parameters by minimising the cumulative Brier loss over all folds (so that the same parameters are used for all folds). To apply Platt's method to calibrate the scores generated by the underlying algorithms we use logistic regression, namely the function `mnrfit` within MATLAB's Statistics toolbox. For isotonic regression calibration we use the implementation of the PAVA in the R package `fdrtool` (namely, the function `monoreg`). Missing values are handled using the Weka filter `ReplaceMissingValues`, which replaces all missing values for nominal and numeric attributes with the modes and means from the training set.

Figures 5.4 and 5.5 show our results for the `covertype` data set available from the UCI repository [31] and also known as `forest`). In converting this multiclass classification problem to binary we follow [15]: treat the largest class as 1 and the rest as 0, and only consider a random and randomly permuted subset consisting of $30,000$ observations; the first 5000 of those observations are used as the training set and the remaining $25,000$ as the test set. The CVAP results are still at the bottom of the plots and very stable; and the values at the `all` column are still particularly unstable.

Similar results for the `insurance`, `Bank Marketing`, `Spambase`, and `Statlog German Credit Data` data sets are shown in Figures 5.6–5.13. The data sets are split into training and test sets in proportion 2:1, without randomization. In particular, for the `insurance` data set we ignore the original split into the training (5822 observations) and test (4000 observations) sets.

Since the values for the `all` column are so unstable, the reader might prefer to disregard them in the case of IVAP, Platt, and Isotonic. In Figures 5.6–5.11 the CVAP results tend to be at the bottom of the plots. The `Statlog German Credit Data` data set is much more difficult, and all results in Figures 5.12–5.13 are poor and somewhat mixed; however, they still demonstrate that CVAPs and IVAPs produce stable results and avoid the occasional bad failures characteristic of the alternative calibration methods.

And finally, Figures 5.14 and 5.15 show the results for log loss and Brier loss, respectively, for the `adult` data set and for a wide range of the ratios of the size of the proper training set to the calibration set. The left-most column of each plot is $1:9$, which means, in the case of Platt's method, isotonic regression, and IVAPs, that 10% of the training set was allocated to the proper training set and the rest to the calibration set. In the case of

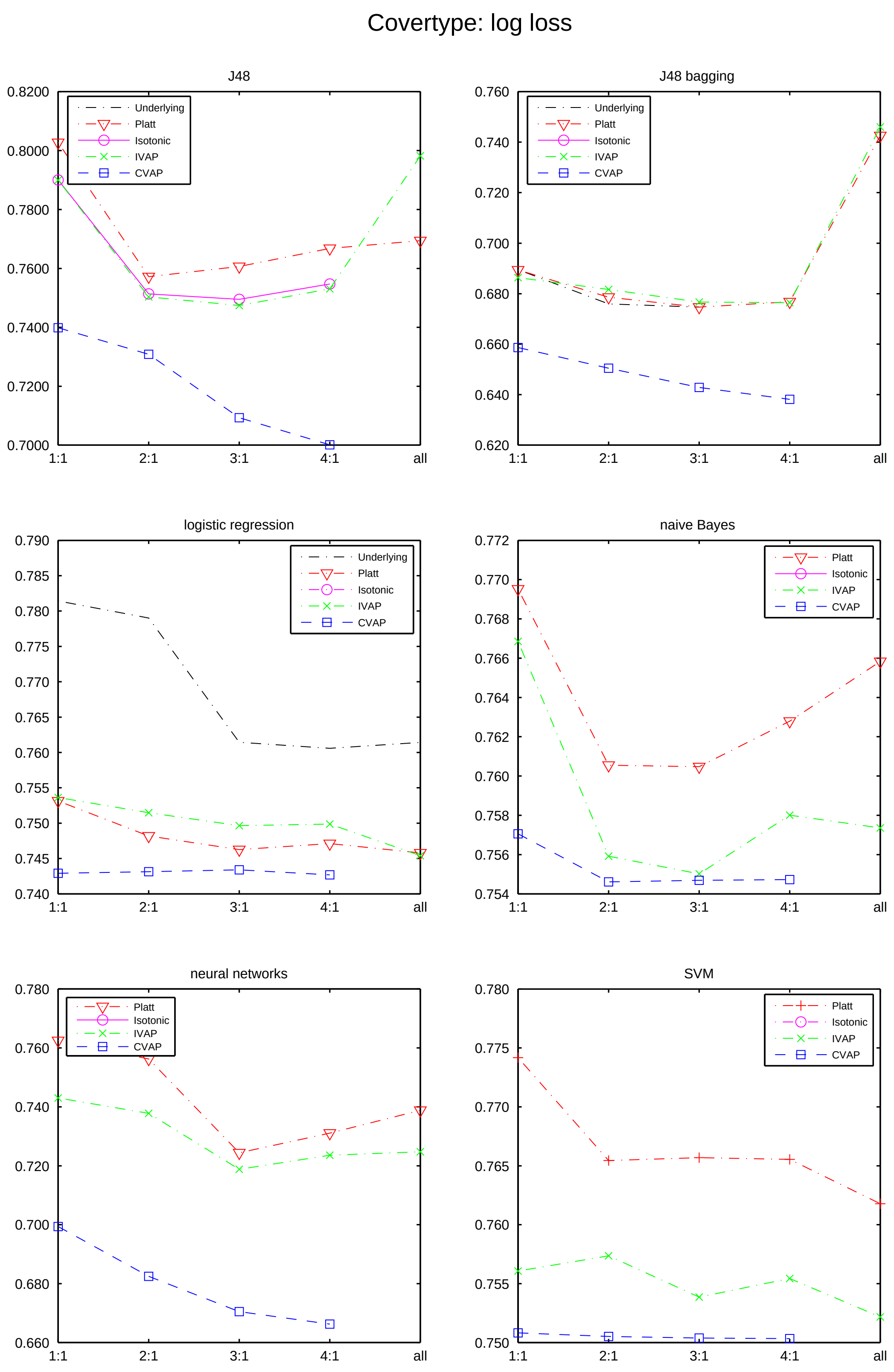

FIGURE 5.4: The analogue of Figure 5.2 for the `covertype` data set.

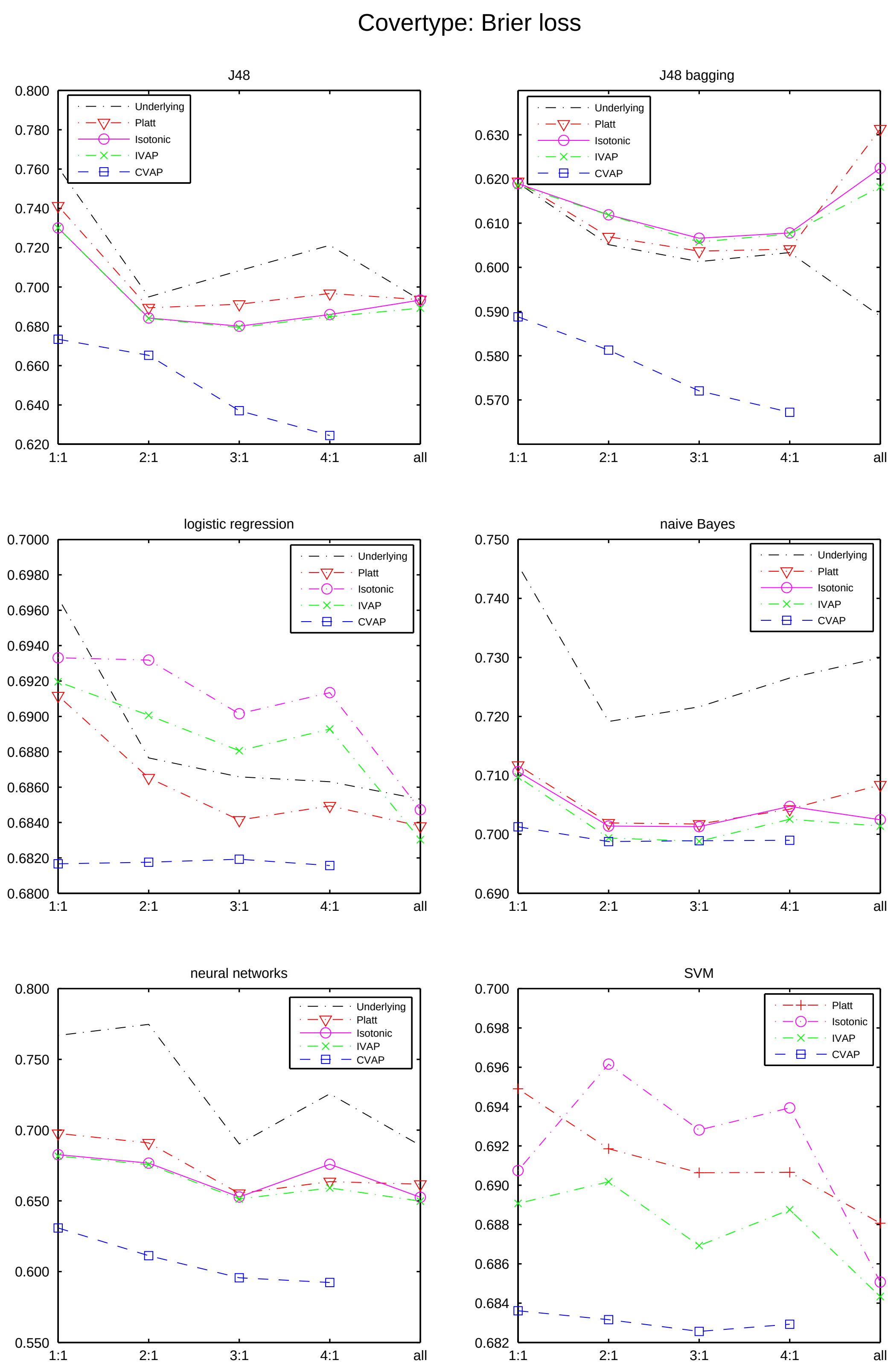

FIGURE 5.5: The analogue of Figure 5.3 for the `covertype` data set.

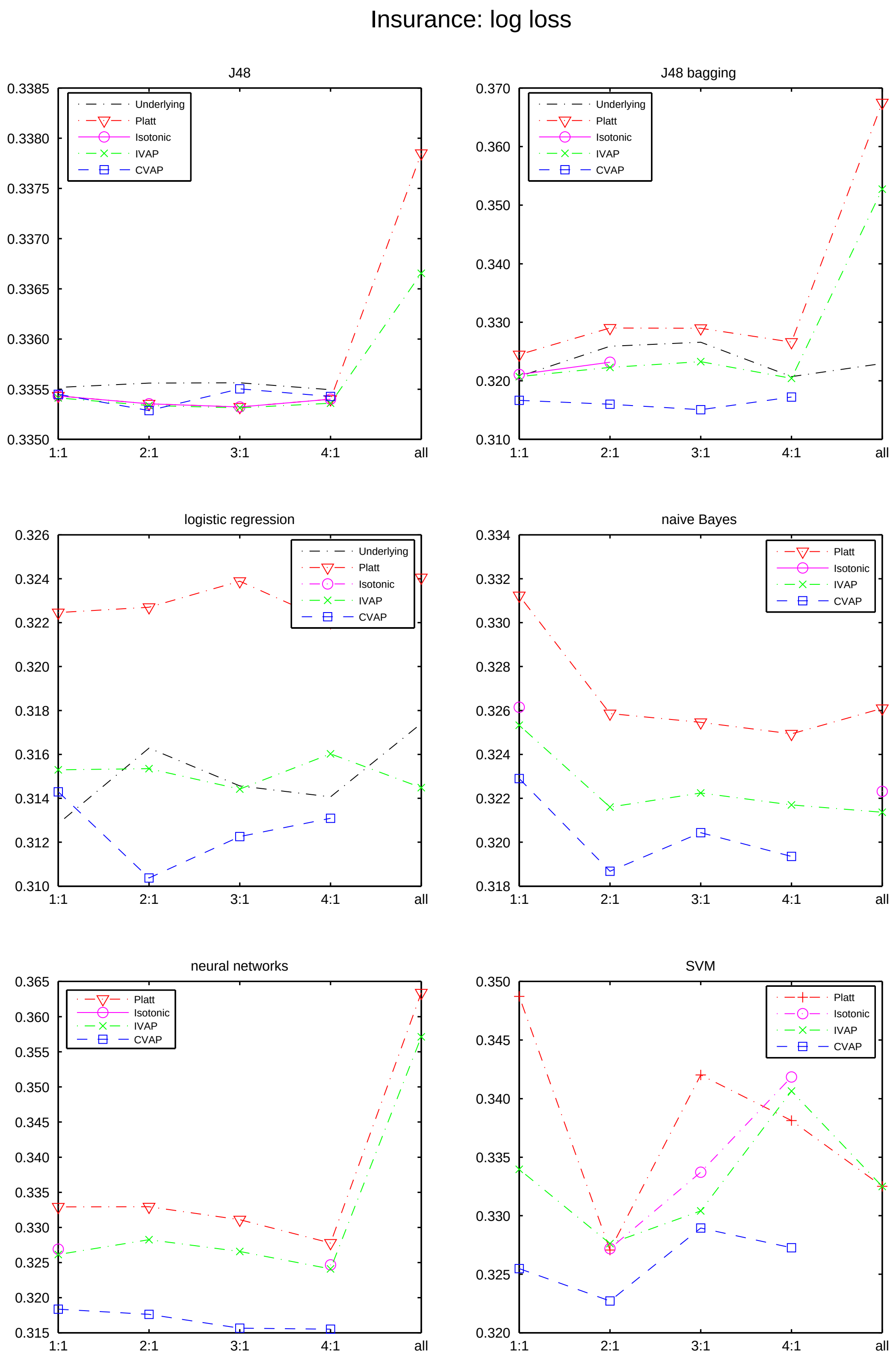

FIGURE 5.6: The analogue of Figure 5.2 for the `insurance` data set.

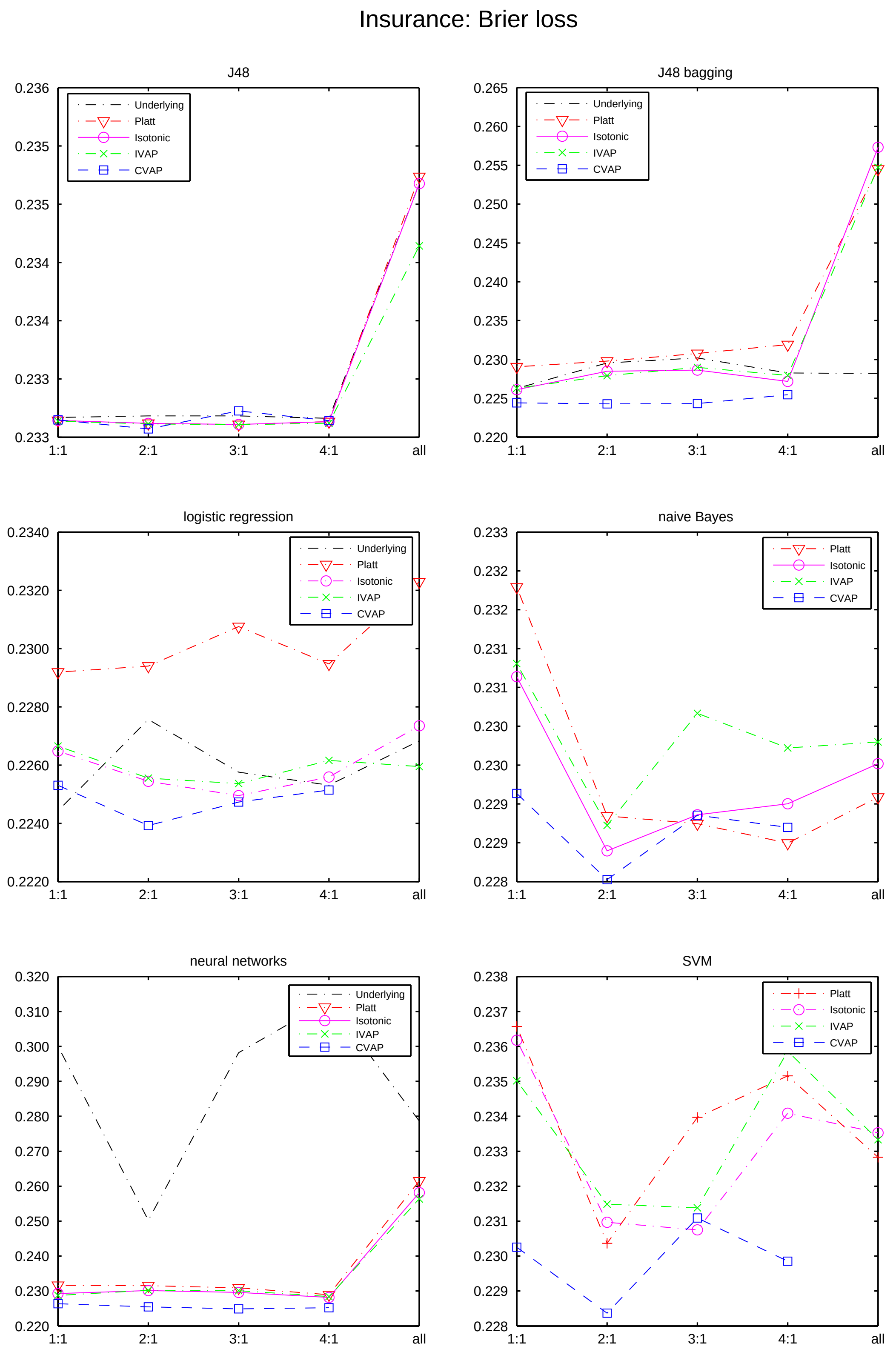

FIGURE 5.7: The analogue of Figure 5.3 for the `insurance` data set.

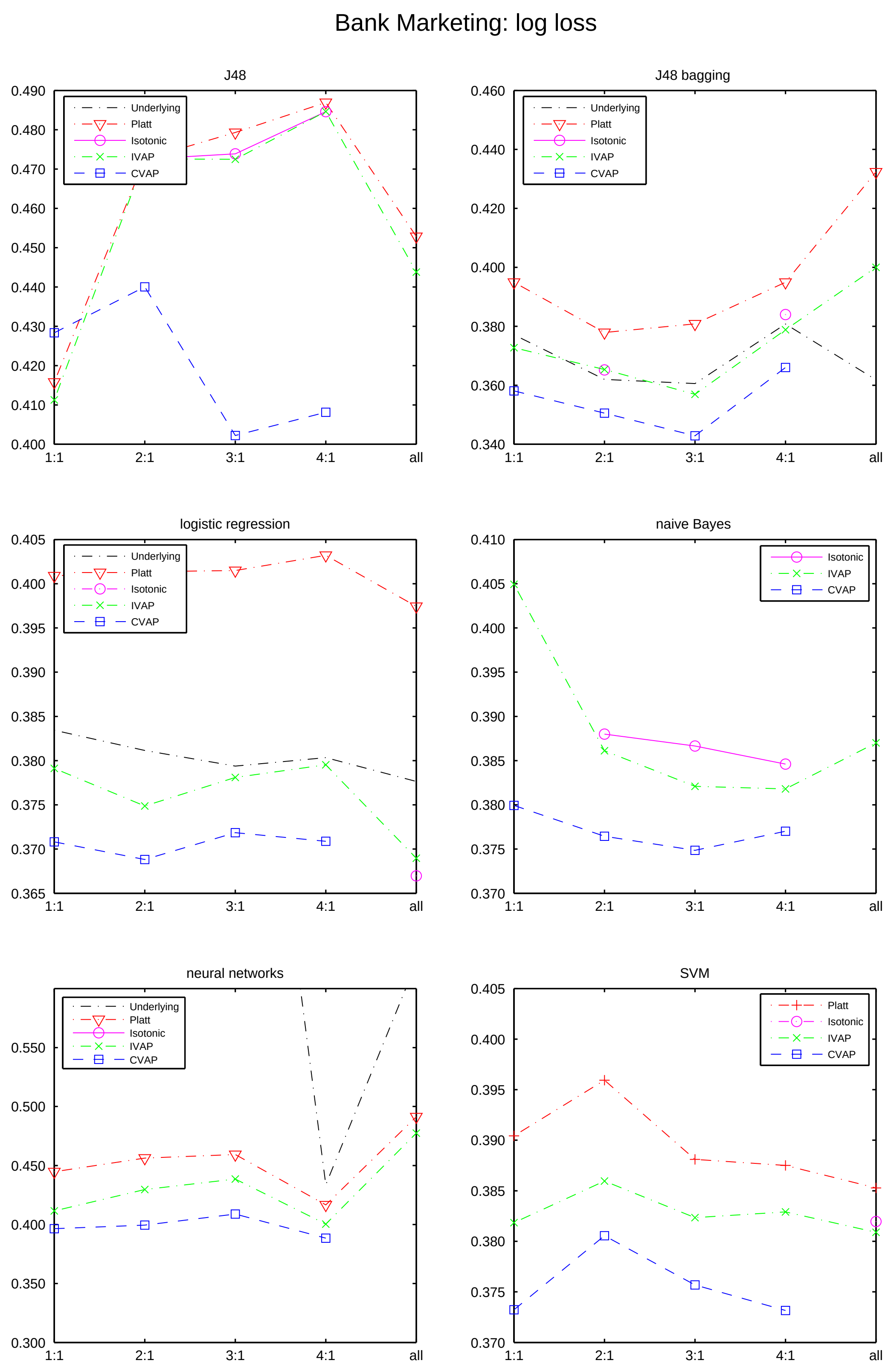

FIGURE 5.8: The analogue of Figure 5.2 for the `Bank Marketing` data set.

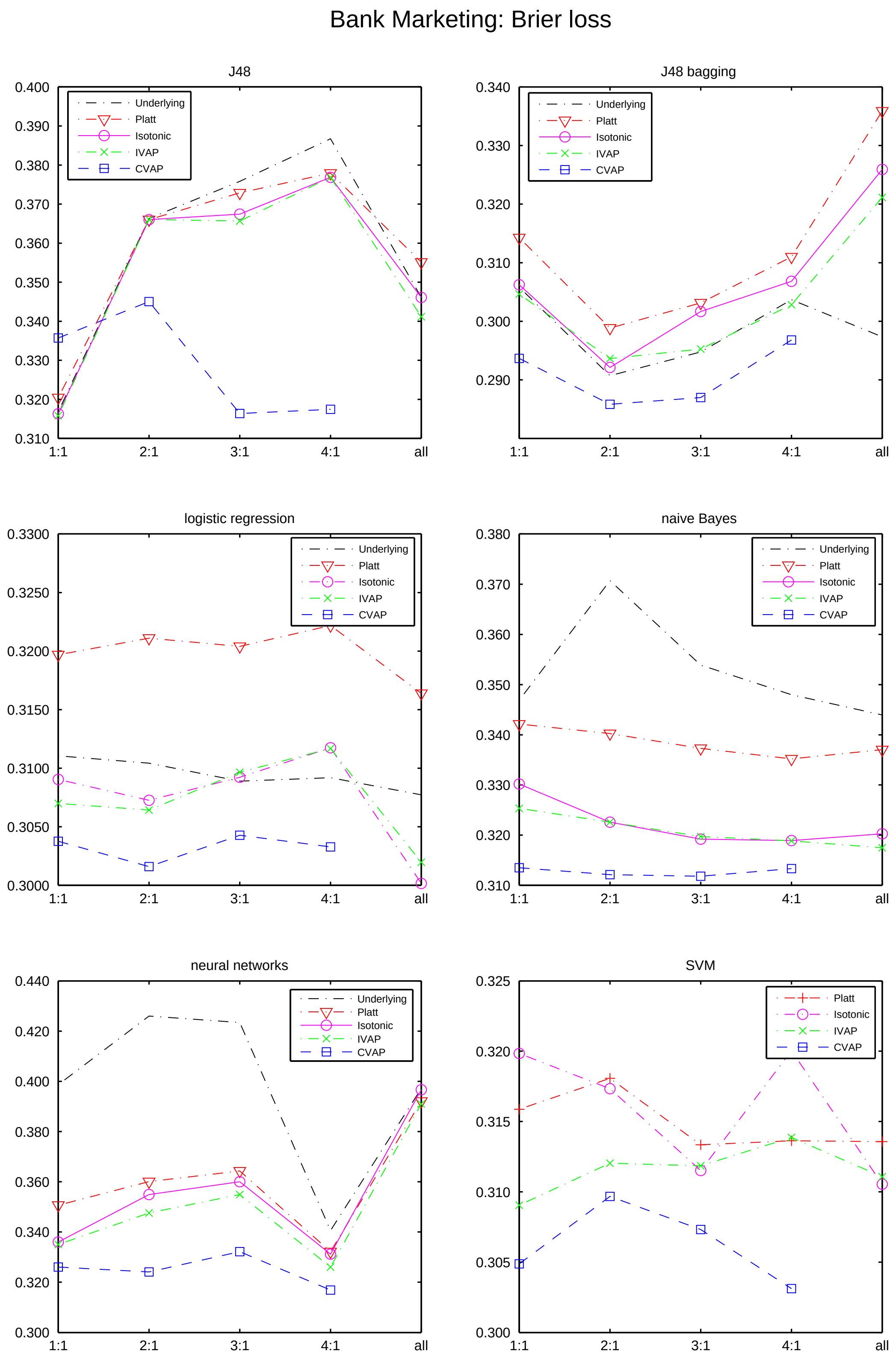

FIGURE 5.9: The analogue of Figure 5.3 for the `Bank Marketing` data set.

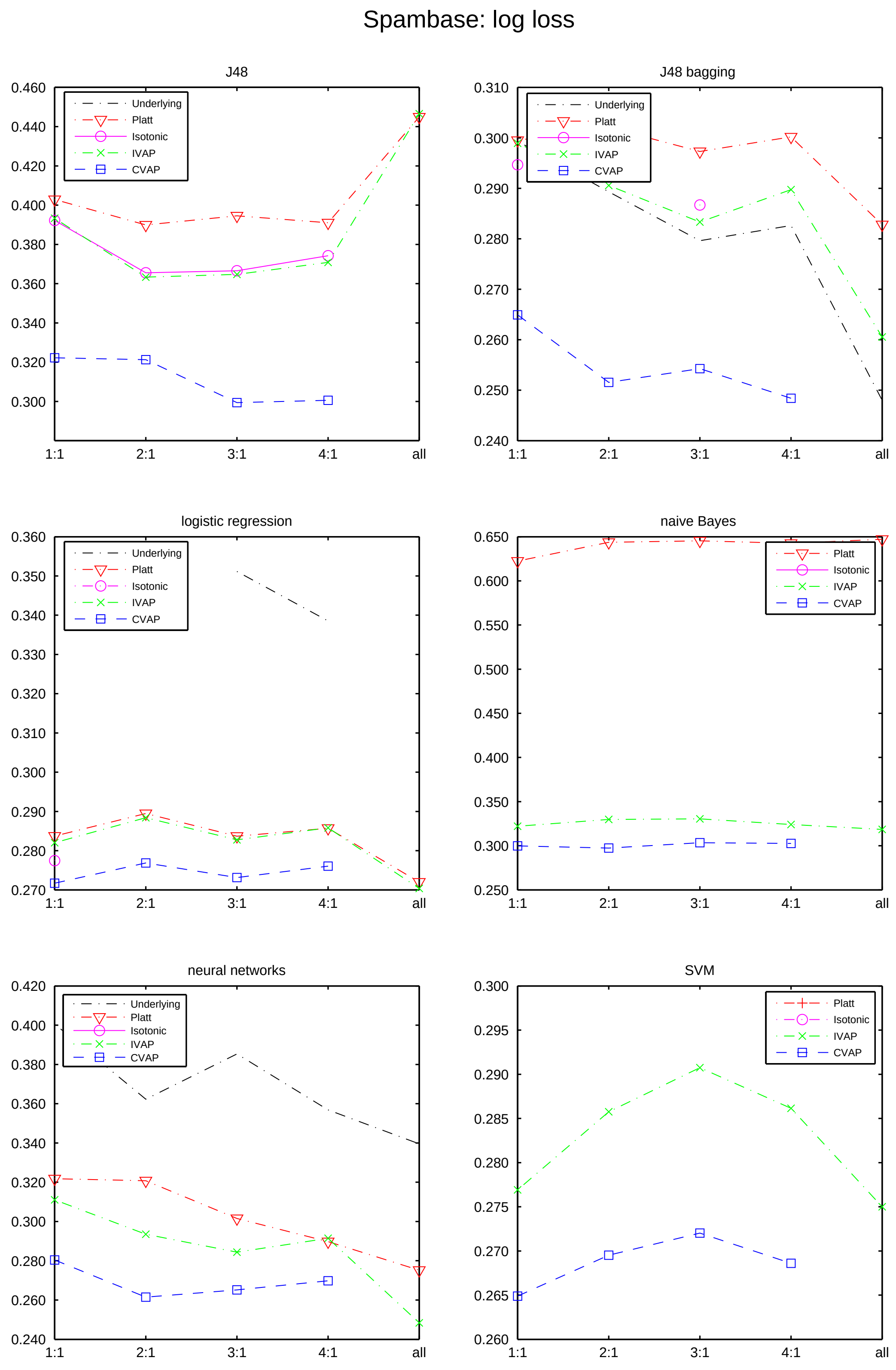

FIGURE 5.10: The analogue of Figure 5.2 for the `Spambase` data set.

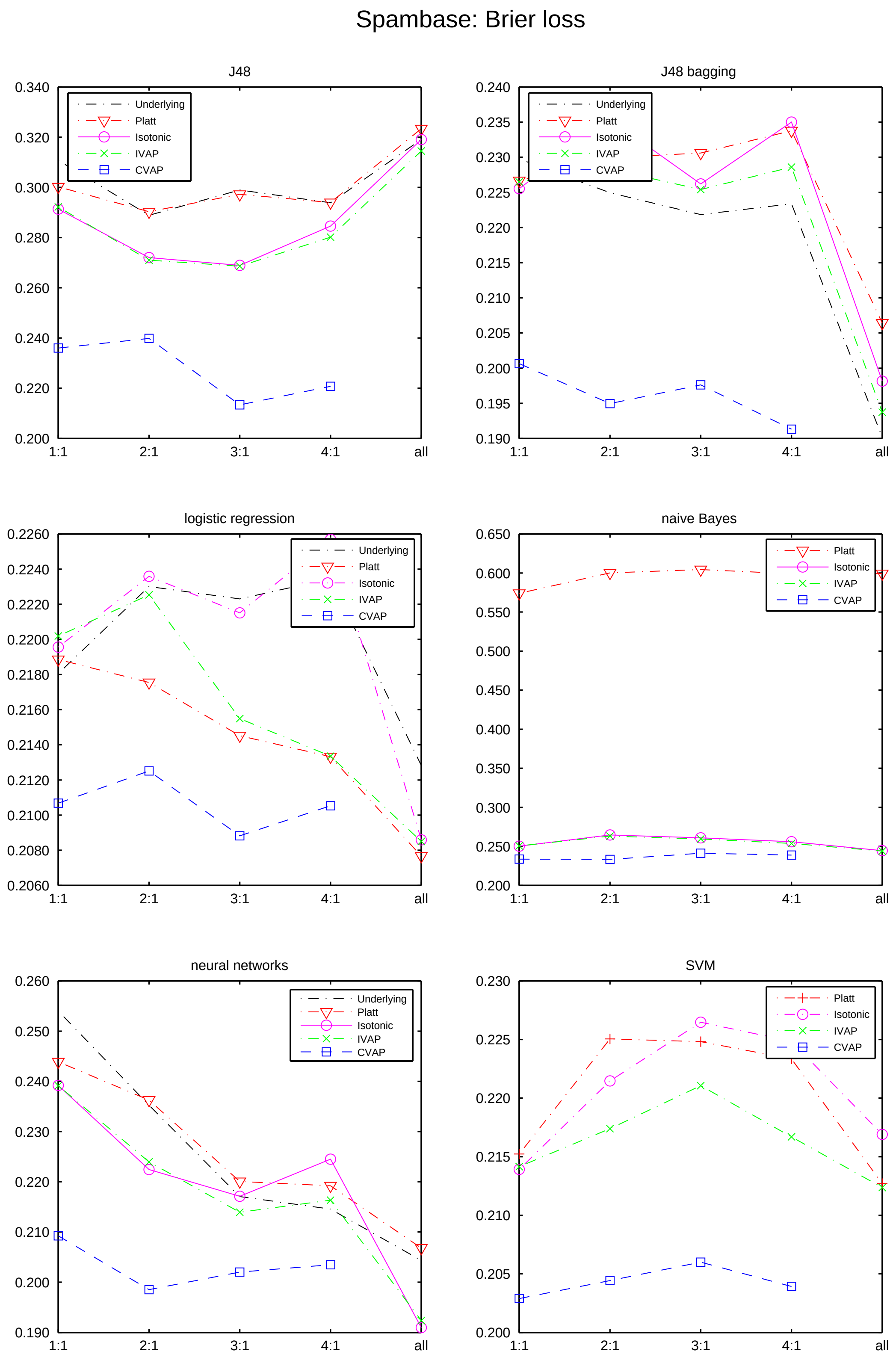

FIGURE 5.11: The analogue of Figure 5.3 for the `Spambase` data set.

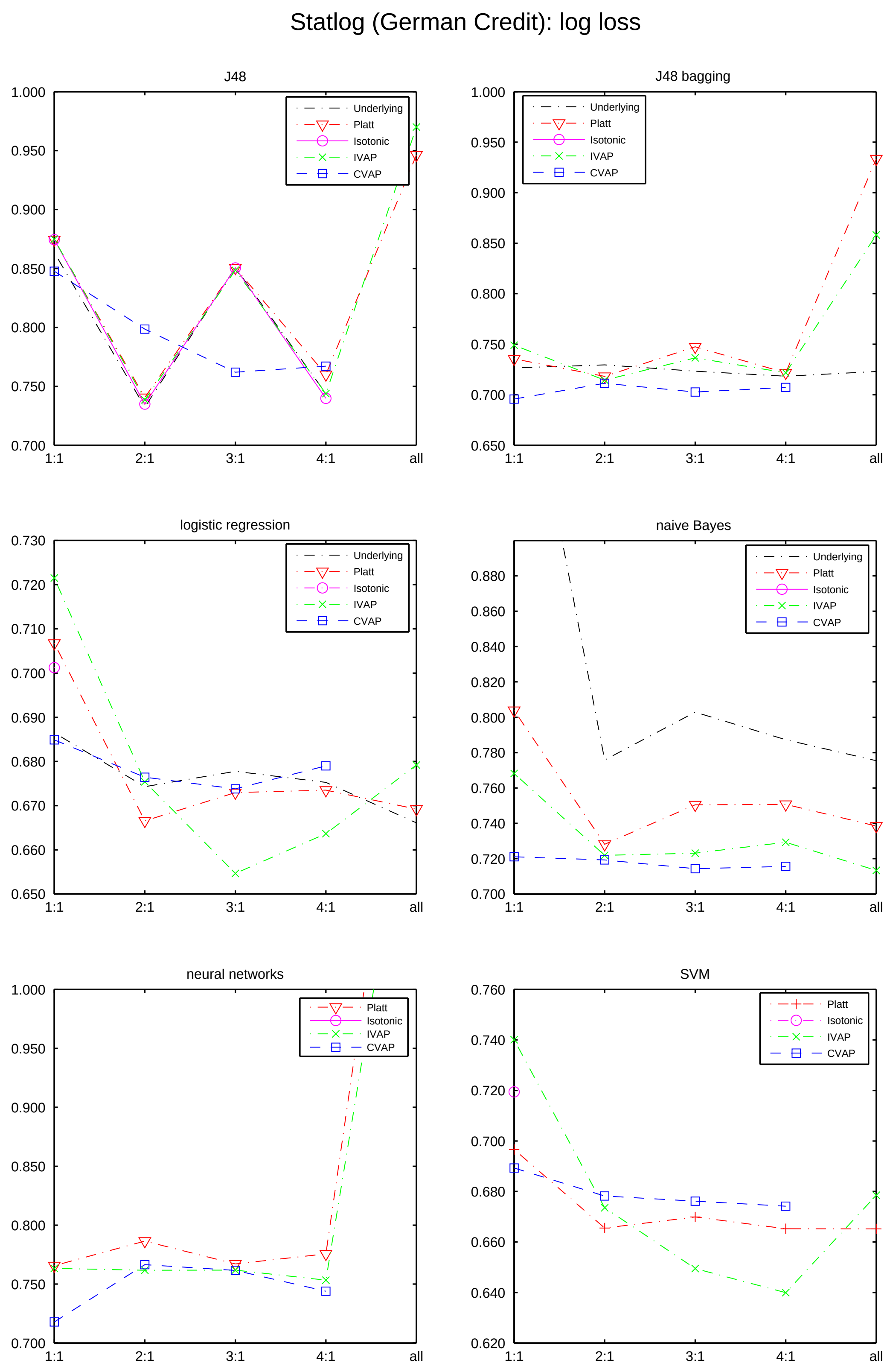

FIGURE 5.12: The analogue of Figure 5.2 for the data set `Statlog German Credit Data`.

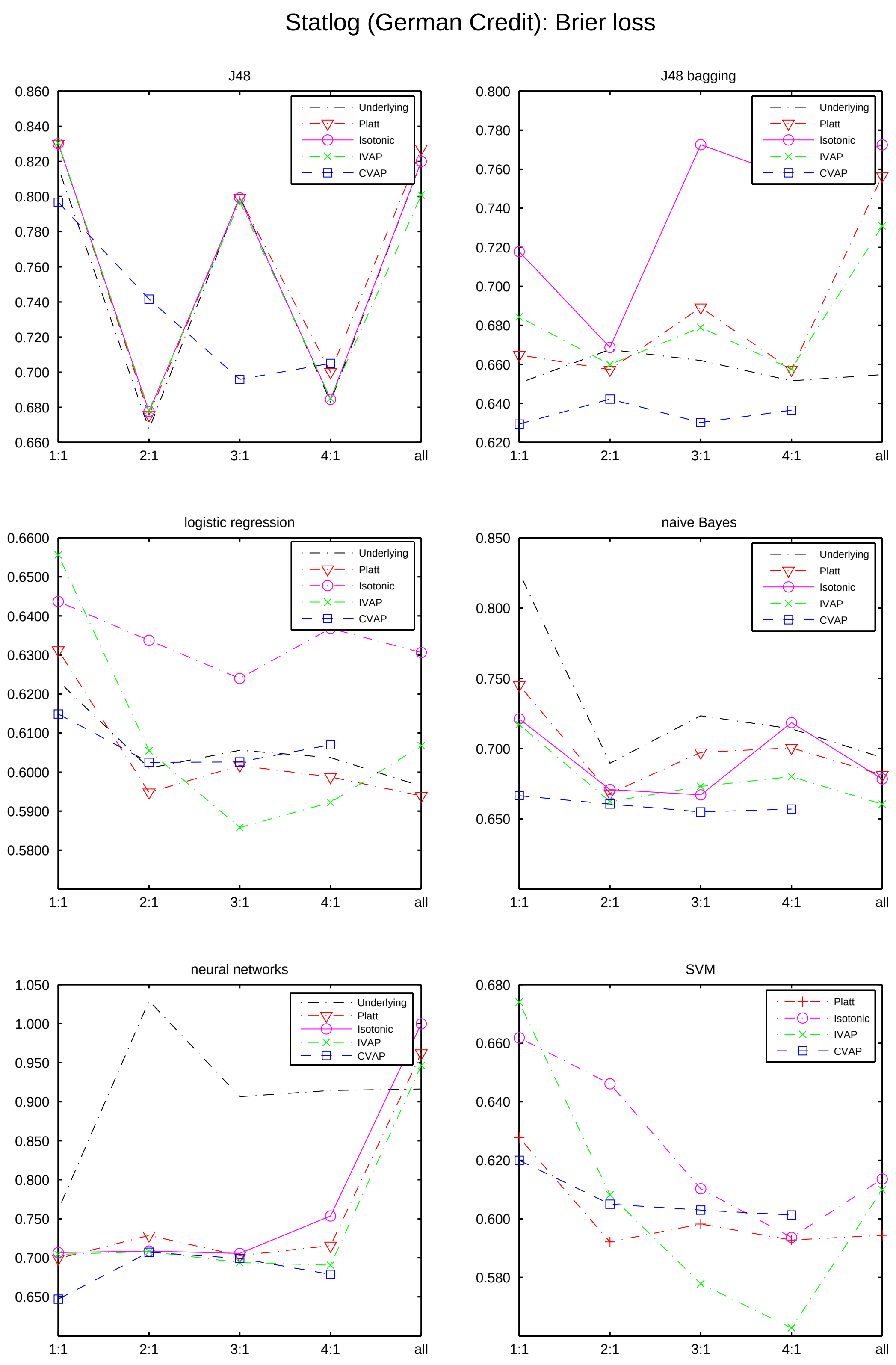

FIGURE 5.13: The analogue of Figure 5.3 for the data set `Statlog German Credit Data`.

CVAPs, $1 : 9$ means that the training set was split into 10 folds, each of them in turn was used as the proper training set, and the rest were used as the calibration set; the results were merged using the minimax procedure as described in Section 5.4. In the case of the underlying algorithm, $1 : 9$ means that only 10% of the training set was in fact used for training (the same 10% as for the first three calibration methods). The other columns are $1 : 8$, $1 : 7$,..., $1 : 2$, $1 : 1$ (which corresponds to $1 : 1$ in Figures 5.2 and 5.3),..., $4 : 1$ (which corresponds to $4 : 1$ in Figures 5.2 and 5.3, i.e., to the standard procedure of 5-fold cross-validation), $5 : 1$,..., $9 : 1$ (the latter corresponds to the other standard cross-validation procedure, that of 10-fold cross-validation); the results in those columns are analogous to those in the column $1 : 9$. In order not to duplicate the information we gave earlier for the `adult` data set, we give the results for a randomly permuted `adult` data set. There is not much difference between 5 and 10 folds for most underlying algorithms (logistic regression behaves unusually in that its performance deteriorates as the size of the proper training set increases, perhaps because less data are available for calibration).

5.7 Conclusion

This chapter introduces two new computationally efficient algorithms for probabilistic prediction, IVAP, which can be regarded as a regularised form of the calibration method based on isotonic regression, and CVAP, which is built on top of IVAP using the idea of cross-validation. Whereas IVAPs are automatically perfectly calibrated, the advantage of CVAPs is in their good empirical performance.

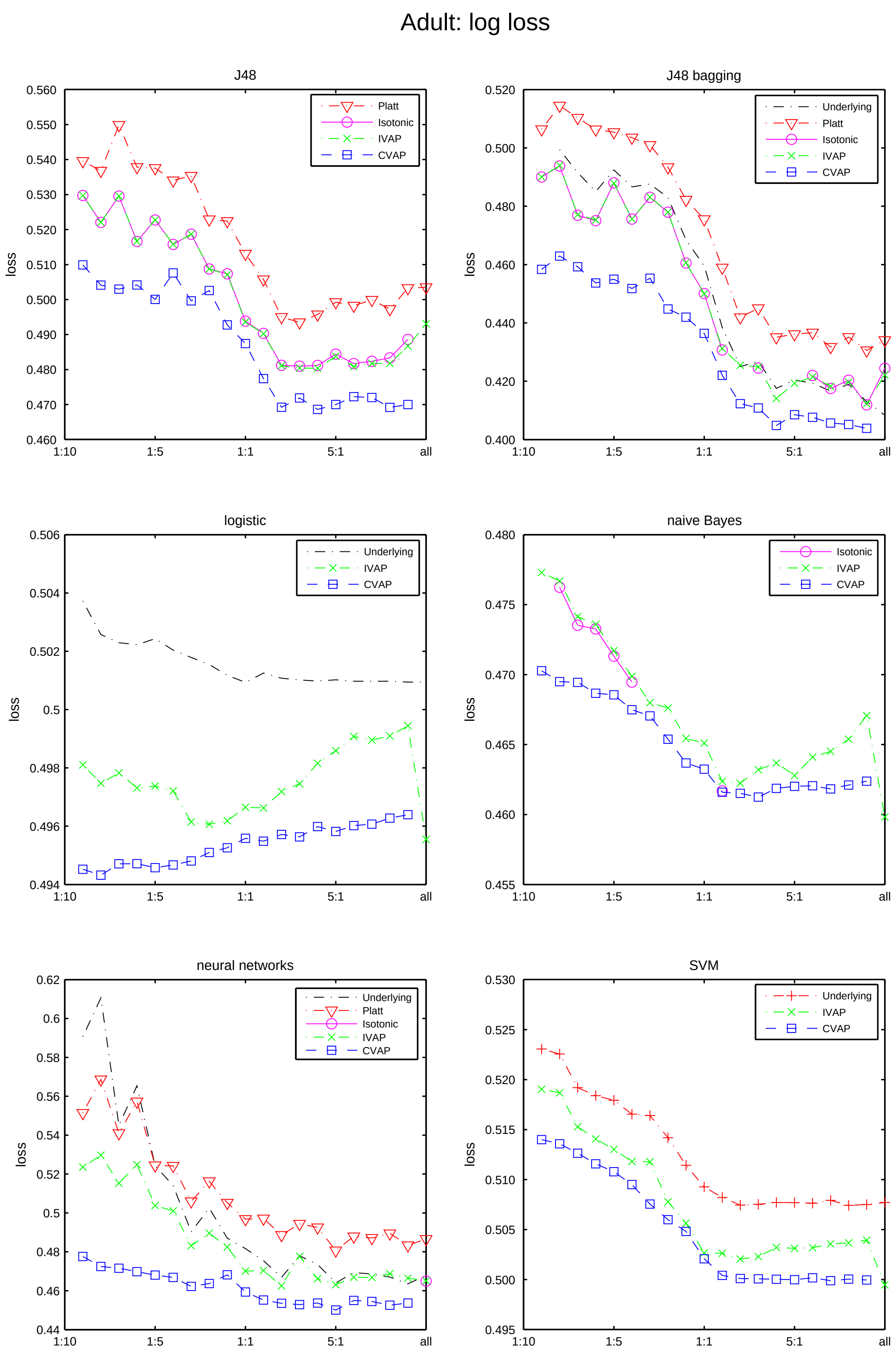

FIGURE 5.14: The log loss on the `adult` data set of the six prediction algorithms and four calibration methods

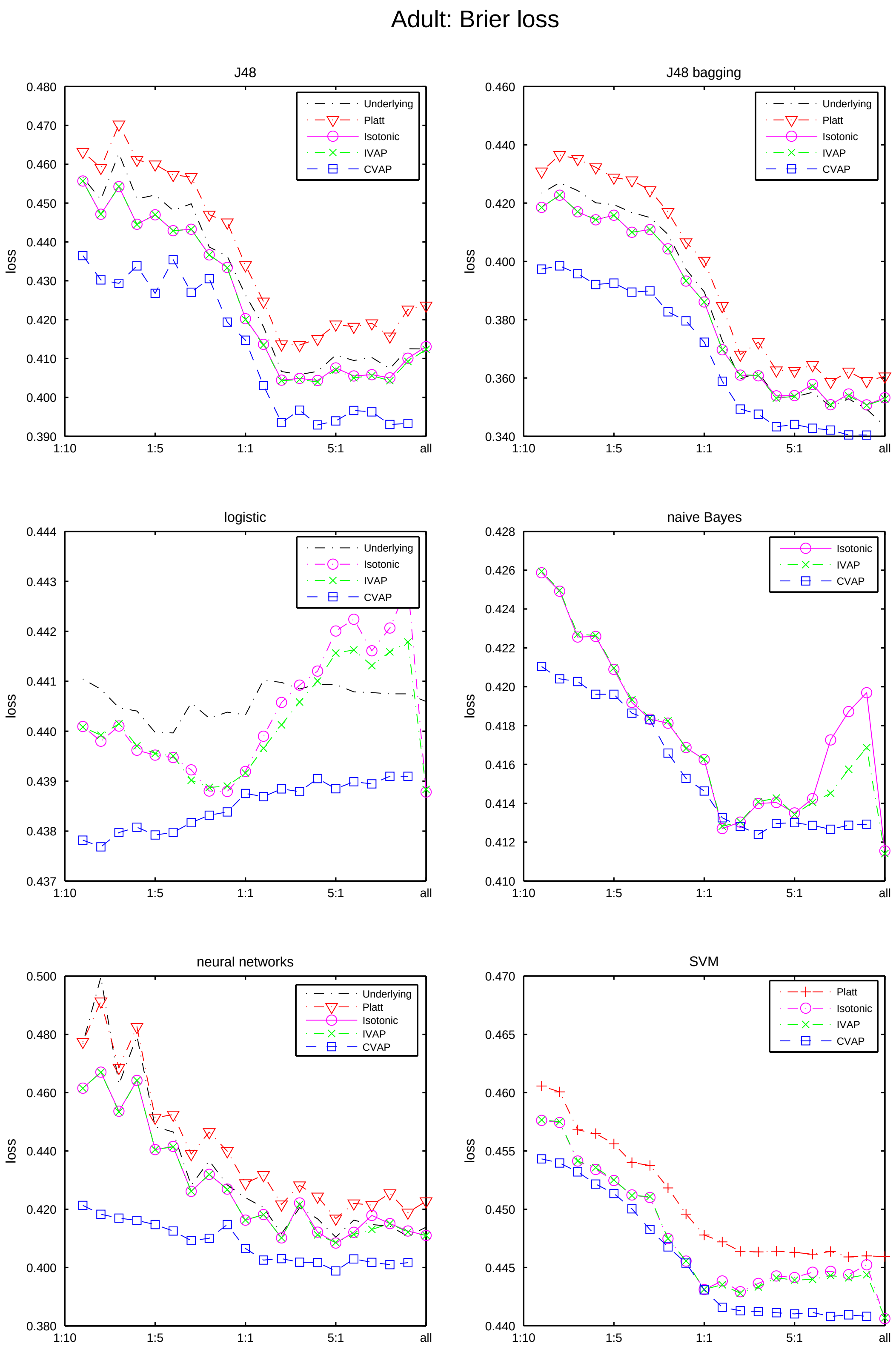

FIGURE 5.15: The analogue of Figure 5.14 for the Brier loss function

Chapter 6

Conclusion

This thesis studies novel methods of producing well calibrated probabilistic predictions using standard machine learning algorithms which can complement or improve on existing methods to date.

Chapter 1 introduces the field of probabilistic machine learning and Chapter 2 describes some of the existing methods to date. Despite the wide array of algorithms which can either produce probabilistic predictions directly or in turn produce scores which can be calibrated into probabilities, the resulting outputs are often miscalibrated. Where calibration is used to improve on the resulting accuracy, the methods often rely on parametric assumptions. A clear need is therefore identified for research into non-parametric probabilistic classification methods which form the bulk of the work carried out in this thesis.

Chapter 3 applies the idea of transforming p-values produced by conformal predictors into probabilities. The method relies on the previously reported observation that some criteria of efficiency for conformal prediction (called "probabilistic criteria") encourage using the conditional probability $Q(y \mid x)$ as the conformity score for an observation (x, y), with Q being the data-generating distribution. The chapter extends this observation to label-conditional predictors. Theorem 1 in Section 3.3 shows that given a conformal predictor Γ that is nearly optimal with respect to a probabilistic criterion, in the limit of a very large training set the p-value that Γ outputs for an observation (x, y) is a monotonic transformation of the conditional probability $Q(y \mid x)$. The results suggest that the p-values can therefore be converted back into conditional probabilities using their distribution in the test set, i.e. they can be calibrated into probabilities. Section 3.6

shows an example of a realistic situation where use of the above technique improves on the standard approach. The experiments use the USPS data set of hand-written digits and compare the results of the 1-Nearest Neighbour (1-NN) algorithm using tangent distance calibrated using the method derived in this chapter with the standard Platt's algorithm for extracting probabilities from support vector machines combined with pairwise coupling. The resulting log and Brier losses are significantly better for the former. Despite promising empirical results, one of the main identified drawbacks of the proposed method lies in its relative inefficiency caused by separate treatment of different classes at the stage of calibrating p-values.

Chapter 4 discusses Venn predictors which are an important class of multiprobability predictors. The chapter also states an important property of *validity* of Venn predictors: they are automatically well calibrated. Section 4.3 builds on the work Venn prediction to define a natural class of Venn predictors, called Venn-Abers predictors (VAP) (with the "Abers" part formed by the initial letters of the authors' surnames of the paper by Ayer *et.al.* [2] introducing the underlying technique). Venn-Abers predictors are defined on top of a wide class of classification algorithms, referred to as "scoring classifiers" in this thesis; each scoring classifier can be automatically transformed into a Venn-Abers predictor and this transformation is referred to as the "Venn–Abers method". Because of its theoretical guarantees, this method can be used for improving the calibration of probabilistic predictions.

The Venn–Abers method is a simple modification of Zadrozny and Elkan's method [97]; being a special case of Venn prediction, it overcomes the problem of potentially poor calibration. Theorem 2 in Section 4.2 shows that Venn predictors are perfectly calibrated. The price to pay, however, is that Venn predictors are multiprobabilistic predictors, in the sense of issuing a set of probabilistic predictions instead of a single probabilistic prediction. Section 4.5 explores the efficiency of Venn–Abers predictors empirically using the fundamental log loss function and another popular loss function, square loss. To apply these loss functions we need, however, probabilistic predictions rather than multiprobabilistic predictions, therefore Section 4.4 defines natural minimax ways of replacing the latter with the former. Section 4.5 explores the empirical predictive performance of the

Venn–Abers method, and the latter's simplified version, which is not only simpler but also more efficient computationally. Each methods is applied to nine benchmark data sets from the UCI repository [31] and six standard scoring classifiers, with the predictive performance of each method evaluated for each combination of a data set and classifier. The results show that the Venn–Abers and simplified Venn–Abers methods usually improve the performance of the underlying classifiers, and in these sets of experiments they work better than the original Zadrozny–Elkan method.

Despite promising results, one of the main drawbacks of the Venn–Abers and simplified Venn–Abers methods is that their computational efficiency is relatively weak compared with many standard machine learning algorithms. Chapter 5 introduces two computationally efficient versions of Venn–Abers predictors, referred to as inductive Venn–Abers predictors (IVAPs) which can be regarded as a regularised form of the calibration method based on isotonic regression and cross-Venn–Abers predictors (CVAPs) which are built on top of IVAPs using the idea of cross-validation. The results in Sections 5.2 and 5.3 show that the main desiderata of Venn predictors, namely validity, predictive efficiency and computational efficiency are satisfied by IVAPs and CVAPs. In particular the chapter shows that:

- Validity (in the form of perfect calibration) is satisfied by IVAPs automatically, and experimental results in section 5.6 suggest that it is inherited by CVAPs.
- Predictive efficiency of IVAPs and CVAPs is determined by the predictive efficiency of the underlying learning algorithms.
- Their computational efficiency is, again, determined by the computational efficiency of the underlying algorithm; the computational overhead of extracting probabilistic predictions consists of sorting (which takes time $O(n \log n)$, where n is the number of observations) and other computations taking time $O(n)$.

Just like VAPs, as precise probabilistic predictors, IVAPs and CVAPs are ways of converting the scores for test objects output by underlying machine learning algorithms into numbers in the range $[0, 1]$ that can serve as probabilities. Section 5.5 compares two existing alternative calibration methods with IVAPs and CVAPs theoretically. Section 5.6

is devoted to experimental comparisons and shows that CVAPs consistently outperform the two existing methods.

In summary, this thesis outlined three novel methods for probabilistic classification, one based on conformal prediction and two based on Venn prediction. The former, described in Chapter 3 offers promising initial results however its practical use is determined by the choice of particular conformity measure and its efficiency with respect to the probabilistic criteria for a given problem at hand. Further empirical tests are therefore necessary, perhaps most optimally on synthetic datasets, in order to determine the extent to which the method is best applied in practice. In contrast the Venn–Abers method described in Chapter 4 and its computationally more efficient version described in Chapter 5 are more universal in a sense that they can be applied on top of a wide range of classification algorithms without any parametric assumptions, aside from the monotonicity of the output of the underlying algorithm with respect to its probabilistic prediction. Experimental results reported in this thesis suggest that the Venn–Abers method can be useful for probabilistic calibration of either small datasets with a large number of features (as in Chapter 4) or for larger datasets for which existing underlying algorithms produce miscalibrated probability outputs (as in Chapter 5). The most dramatic impact of the Venn–Abers method is in calibrating algorithms which suffer a potentially infinite log loss in incorrectly predicting a class with 100% probability. Empirically, the method also seems to offer improvement for relatively simple classifiers (such as naïve Bayes and logistic regression) thereby possibly correcting for bias and for classifiers which assume a parametric relationship between the underlying algorithm outputs and class probabilities (such as Platt calibration for SVM). The lowest impact appears to be for algorithms which contain a degree of randomisation (such as J48 with bagging). It would be interesting therefore to consider whether the Venn–Abers method could also be applied in the context of ensemble learning methods. The following section outlines some other possible avenues of extending the results reported in this thesis further.

6.1 Further work

Following on from the work in this thesis, enclosed are some directions for future research which would be interesting to pursue:

- There are several algorithms for performing isotonic regression on a partially, rather than linearly, ordered set: see, e.g., [13], Section 2.3. IVAPs and CVAPs can be defined in the situation where scores take values only in a partially ordered set; moreover, Proposition 3 will continue to hold. The importance of partially ordered scores stems from the fact that they enable us to benefit from a possible "synergy" between two or more prediction algorithms [81]. Preliminary results reported in [81] in a related context suggest that the resulting predictor can outperform predictors based on the individual scalar scores. It would be interesting to test whether a similar result holds for partially ordered IVAP and CVAP equivalents.
- This thesis did not study empirically upper and lower probabilities produced by IVAPs and CVAPs, whereas the distance between them provides information about the reliability of the merged probability prediction. Finding interesting ways of using this extra information could be one of the directions of further research.
- Finally, the work in this thesis primarily focused on binary prediction problems. Some work has already begun on applying the methods derived in this thesis to multi-class problems [49] and it would be interesting to extend the results further.

Appendix A

Datasets

The experimental results of this thesis relied on use of standard datasets briefly summarised below.

A.1 USPS

The US Postal Service (USPS) dataset, first reported in [47] is a widely used dataset within the machine learning community. It consists of a collection of handwritten digits from real-life postal codes. The sizes of training and test sequences are 7291 and 2007, respectively. Each example is described by 256 features representing the brightness of pixels on the 16×16 grey-scaled image displaying a digit and its corresponding label (the digit). The brightness takes values in the interval $(-1, 1)$ and the label is a decimal digit from 0 to 9. For experiments in Chapter 3 each example is normalised as described in Appendix B.3 in [84] so that the mean brightness of pixels in each picture is 0 and its standard deviation is 1.

A.2 Australian Credit Approval

The Australian Credit Approval dataset[1] [31], reported in [62], concerns credit card applications. It contains 690 instances and 14 attributes. All attribute names and values have been changed to symbols to protect confidentiality of the data. This dataset is interesting because there is a good mix of attributes – continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also a few missing values.

[1] `http://archive.ics.uci.edu/ml/datasets/statlog+(australian+credit+approval)`

There are 6 numerical and 8 categorical attributes. The labels have been changed for the convenience of the statistical algorithms.

A.3 Breast Cancer

The Breast Cancer dataset[2], first used in [73], contains features computed from a digitized image of a fine needle aspirate of a breast mass. There are a total of 569 instances. They describe characteristics of the cell nuclei present in the image. Each image is summarised by 10 real valued attributes and a label containing the diagnosis, malignant (M) or benign (B).

A.4 Diabetes

The Pima Indians Diabetes Data Set[3] supplied by the US National Institute of Diabetes and Digestive and Kidney Diseases first used in [69], contains data on patients who are females at least 21 years old of Pima Indian heritage. There are a total of 768 instances, 8 attributes and a class label specifying whether the patient is diabetic.

A.5 Echocardiogram

The Echocardiogram dataset[4] contains 132 instances and 12 attributes for classifying if patients will survive for at least one year after a heart attack. The survival and still-alive variables, when taken together, indicate whether a patient survived for at least one year following the heart attack. The problem addressed by past researchers [43] was to predict from the other variables whether or not the patient will survive at least one year. The most difficult part of this problem is correctly predicting that the patient will not survive.

[2]`https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)`
[3]`https://archive.ics.uci.edu/ml/datasets/Pima%20Indians%20Diabetes`
[4]`https://archive.ics.uci.edu/ml/datasets/echocardiogram`

A.6 Hepatitis

The Hepatitis dataset[5] used in [28] contains 19 categorical, integer and real attributes and a class label specifying histology (YES or NO). There are a total of 155 instances.

A.7 Ionosphere

The Ionosphere dataset concerns classification of radar returns from the ionosphere. The targets were free electrons in the ionosphere. Classification labels consist of "Good" radar returns showing evidence of some type of structure in the ionosphere and "Bad" returns which do not; their signals pass through the ionosphere. There are a total of 34 continuous attributes which aid classification. The dataset was first reported in [67].

A.8 Labor Relations

The Labor Relations dataset[6], reported in [6] includes all collective agreements reached in the business and personal services sector for Canadian local authorities with at least 500 members (teachers, nurses, university staff, police, etc.) in Canada in 1987 and first quarter of 1988. It contains 57 instances and 16 attributes.

A.9 Liver Disorders

The Liver Disorders dataset[7], provided by BUPA Medical Research,contains 7 attributes and 345 instances. The variables are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. Each instance constitutes the record of a single male individual.

[5]`https://archive.ics.uci.edu/ml/datasets/hepatitis`
[6]`https://archive.ics.uci.edu/ml/datasets/Labor+Relations`
[7]`https://archive.ics.uci.edu/ml/datasets/liver+disorders`

A.10 Congressional Voting

The Congressional Voting dataset[8] includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes (serving as attributes) identified by the Congressional Quarterly Almanac (CQA) [78]. The CQA lists nine different types of votes which are summarised as "yes" or "no". The goal is to predict whether the congressmen is a democrat or a republican (serving as a class label). The dataset contains 435 instances.

A.11 Adult

The Adult dataset[9] was extracted from the 1994 US Census database and reported in [44]. There are a total 14 attributes and the prediction task is to determine whether a person earns over 50,000 USD a year. There are a total of 48842 instances.

A.12 Covertype

This is a large dataset[10] consisting of 581,012 instances and 54 categorical and real valued attributes. The goal is to predict forest cover type from cartographic variables only. The data was provided by the Department of Forest Sciences, Colorado State University, US and the study area includes four wilderness areas located in the Roosevelt National Forest of northern Colorado.

A.13 Insurance

The Insurance dataset[11] was used in the CoIL Challenge 2000 [79]. It contains information about customers in the form of 86 variables and includes product usage data and socio-demographic data derived from zip area codes. The data was supplied by the Dutch data mining company Sentient Machine Research and is based on a real world business problem. The goal is to predict whether customers have a caravan insurance policy.

[8] `https://archive.ics.uci.edu/ml/datasets/congressional+voting+records`

[9] `https://archive.ics.uci.edu/ml/datasets/adult`

[10] `https://archive.ics.uci.edu/ml/datasets/covertype`

[11] `https://archive.ics.uci.edu/ml/datasets/Insurance+Company+Benchmark+(COIL+2000)`

A.14 Bank Marketing

The dataset[12] concerns direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ("yes") or not ("no") subscribed. There are a total of 45211 instances and 17 attributes.

A.15 Spambase

This Spambase dataset[13] originated from Hewlett-Packard Labs. It contains 4610 instances and 57 attributes, consisting of frequencies of certain words and characters and several other character specific features. The goal is to predict whether an email is classified as "spam" or not.

A.16 Statlog German Credit Data

This dataset [14] provided by University of Hamburg, classifies people described by a set of attributes as good or bad credit risks. It contains a total of 1000 instances with 20 categorical and integer attributes.

[12]`https://archive.ics.uci.edu/ml/datasets/bank+marketing`
[13]`https://archive.ics.uci.edu/ml/datasets/spambase`
[14]`https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)`

TABLE A.1: The summary of the datasets

Dataset	**Attributes**	**Number of instances**	**Instances in training set**
Australian	14	690	66%
Breast	10	699	66%
Diabetes	8	768	66%
Echo	12	132	66%
Hepatitis	19	155	66%
Ionosphere	34	351	66%
Labor	16	57	66%
Liver	7	345	66%
Vote	16	435	66%
Adult	14	48842	32561
Covertype	58	30000	5000
Insurance	86	9000	66%
Bank Marketing	17	45211	66%
Spambase	57	4610	66%
Statlog	20	1000	66%

Appendix B

Implementation

The Matlab functions for calculating Brier and log losses for different underlying algorithms and calibration methods (Platt, Isotonic Regression, IVAP and CVAP) reported in this thesis are publicly available[1].

The main function *genLosses* has the following form:

```
loss = genLosses(data,wekaAlgorithm,properInt,calibrationInt,tuneType);
```

The `wekaAlgorithm` parameter is based on the Weka data mining tool Weka 3.7.12[2]. The functions also require the Matlab-R interface[3] and the Matlab-Weka interface[4].

The choice for the `wekaAlgorithm` parameter consists of:

- `'tree'` - J48 decision tree
- `'SMO'` - Platt SVM method
- `'naive'` - naïve Bayes
- `'logistic'` - Logistic regression
- `'bagging'` - J48 with bagging
- `'nn'` - neural networks

[1]`https://arxiv.org/abs/1511.00213`
[2]`http://www.cs.waikato.ac.nz/ml/weka/`
[3]`http://www.mathworks.co.uk/matlabcentral/fileexchange/5051-matlab-r-link/`
[4]`http://www.mathworks.co.uk/matlabcentral/fileexchange/21204-matlab-weka-interface`

The `data` paramater corresponds to choice of datasets available from the UCI repository described in Appendix A (formatted to Weka arff file type).

The other input parameters correspond to

- `properInt,calibrationInt` - integer determining the proportion of the proper training set to the calibration set, (in Chapter 5 referred to `proper:calibration`)
- `tuneType` - either `'coarse'` or `'fine'` tuning

The generated output `loss` is a 10 member array of the following format:

- `loss(1)` - Brier loss for the underlying algorithm
- `loss(2)` - Brier loss for Platt calibration applied to the underlying algorithm
- `loss(3)` - Brier loss for Isotonic Regression calibration applied to the underlying algorithm
- `loss(4)` - Brier loss for IVAP calibration applied to the underlying algorithm
- `loss(5)` - Brier loss for CVAP calibration applied to the underlying algorithm
- `loss(6)` - log loss for the underlying algorithm
- `loss(7)` - log loss for Platt calibration applied to the underlying algorithm
- `loss(8)` - log loss for Isotonic Regression calibration applied to the underlying algorithm
- `loss(9)` - log loss for IVAP calibration applied to the underlying algorithm
- `loss(10)` - log loss for CVAP calibration applied to the underlying algorithm

For example the following code will generate the corresponding losses for the `insurance` dataset using the `J48 tree` algorithm with a 3:1 ratio of the proper training set to calibration set:

```
loss = genLosses('insurance','tree',3,1,'fine');
```

A summary ot the Weka parameters used is shown in Table B.1.

TABLE B.1: The summary of the experimental parameters

Algorithm	**Varied parameters**	**Range**
J48	-C pruning confidence	0.25E-04 to 0.25
J48 Bagging	-C pruning confidence	0.25E-04 to 0.25
naïve Bayes		
logistic regression	-R ridge in log-likelihood	1E-03 to 1E+03
ANN	-L learning rate, -M momentum	L: 0.1 to 1E+05 M: 0.3 to 0.6
SVM Platt	- C complexity	1E-03 -to 1E+06

Bibliography

[1] Isabelle Alvarez, Stephan Bernard, and Guillaume Deffuant. “Keep the Decision Tree and Estimate the Class Probabilities using its Decision Boundary”. In: *The 20th International Joint Conference on Artificial Intelligence*. 2007, pp. 654–659.

[2] Miriam Ayer, H Daniel Brunk, George M Ewing, William T Reid, and Edward Silverman. “An empirical distribution function for sampling with incomplete information”. In: *The Annals of Mathematical Statistics* 26.4 (1955), pp. 641–647.

[3] Thomas Bayes, Richard Price, and John Canton. *An essay towards solving a problem in the doctrine of chances*. C. Davis, Printer to the Royal Society of London, 1763.

[4] Tony Bellotti, Zhiyuan Luo, Alex Gammerman, Frederick W Van Delft, and Vaskar Saha. “Qualified predictions for microarray and proteomics pattern diagnostics with confidence machines”. In: *International Journal of Neural Systems* 15.04 (2005), pp. 247–258.

[5] Paul N Bennett. “Using asymmetric distributions to improve text classifier probability estimates”. In: *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. 2003, pp. 111–118.

[6] Francesco Bergadano, Stan Matwin, Jianping Zhang, and Ryszard S Michalski. “Measuring quality of concept descriptions”. In: *Proceedings of the Third European Session on Learning*. 1988, pp. 1–14.

[7] José M Bernardo and Adrian FM Smith. *Bayesian theory*. IOP Publishing, 2001.

[8] Henrik Böstrom. “Calibrating random forests”. In: *Seventh International Conference on Machine Learning and Applications*. IEEE. 2008, pp. 121–126.

[9] Léon Bottou. *Online Learning and Neural Networks*. Cambridge University Press, 1998.

[10] Leo Breiman. “Bagging predictors”. In: *Machine learning* 24.2 (1996), pp. 123–140.

[11] Leo Breiman and Jerome H Friedman. “Predicting multivariate responses in multiple linear regression”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 59.1 (1997), pp. 3–54.

[12] Leo Breiman, Jerome H Friedman, Richard Olshen, and Charles J Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[13] Daniel H Brunk, Richard E Barlow, Daniel J Bartholomew, and James M Bremner. *Statistical inference under order restrictions (the theory and application of isotonic regression)*. Tech. rep. DTIC Document, 1972.

[14] Andres Cano, Andres R Masegosa, and Serafin Moral. “A Bayesian approach to estimate probabilities in classification trees”. In: *4th European Workshop on Probabilistic Graphical Models*. Citeseer. 2008, pp. 49–56.

[15] Rich Caruana and Alexandru Niculescu-Mizil. “An empirical comparison of supervised learning algorithms”. In: *Proceedings of the 23rd International Conference on Machine Learning*. ACM. 2006, pp. 161–168.

[16] Nitesh V Chawla and David A Cieslak. “Evaluating probability estimates from decision trees”. In: *American Association for Artificial Intelligence*. 2006.

[17] Alexey Chervonenkis and Vladimir Vapnik. “Theory of uniform convergence of frequencies of events to their probabilities and problems of search for an optimal solution from empirical data”. In: *Automation and Remote Control* 32 (1971), pp. 207–217.

[18] Ira Cohen and Moises Goldszmidt. “Properties and benefits of calibrated classifiers”. In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer. 2004, pp. 125–136.

[19] Danny Coomans and Désiré Luc Massart. “Alternative k-nearest neighbour rules in supervised pattern recognition: Part 1. k-Nearest neighbour classification by using alternative voting rules”. In: *Analytica Chimica Acta* 136 (1982), pp. 15–27.

[20] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[21] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: *Machine Learning* 20.3 (1995), pp. 273–297.

[22] David R Cox. “The regression analysis of binary sequences”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1958), pp. 215–242.

[23] David R Cox and David V Hinkley. *Theoretical Statistics*. Chapman and Hall, London, 1974.

[24] Koby Crammer and Amir Globerson. “Discriminative learning via semidefinite probabilistic models”. In: *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*. AUAI Press. 2006, pp. 98–105.

[25] Mikhail Dashevskiy and Zhiyuan Luo. “Reliable probabilistic classification and its application to internet traffic”. In: *International Conference on Intelligent Computing*. Springer. 2008, pp. 380–388.

[26] Alexander P Dawid. *Probability forecasting*. Wiley Online Library, 1986.

[27] Luc Devroye. *A course in density estimation*. Birkhauser Boston Inc., 1987.

[28] Persi Diaconis and Bradley Efron. *Computer intensive methods in statistics*. Stanford University. Division of Biostatistics, 1983.

[29] Thomas G Dietterich. “An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization”. In: *Machine learning* 40.2 (2000), pp. 139–157.

[30] Joseph Drish. “Obtaining calibrated probability estimates from support vector machines”. In: *Technique Report, Department of Computer Science and Engineering, University of California, San Diego, CA* (2001).

[31] Andrew Frank and Arthur Asuncion. *UCI machine learning repository*. 2010.

[32] Alex Gammerman, Volodya Vovk, Brian Burford, Ilia Nouretdinov, Zhiyuan Luo, Alexey Chervonenkis, Mike Waterfield, Rainer Cramer, Paul Tempst, Josep Villanueva, et al. “Serum proteomic abnormality predating screen detection of ovarian cancer”. In: *The Computer Journal* 52.3 (2009), pp. 326–333.

[33] Ursula Garczarek. “Classification rules in standardized partition spaces”. PhD thesis. Universitat Dortmund, 2002.

[34] Martin Gebel and Claus Weihs. “Calibrating Margin-Based Classifier Scores into Polychotomous Probabilities”. In: *Data Analysis, Machine Learning and Applications*. Springer, 2008, pp. 29–36.

[35] Ronald L. Graham. “An efficient algorith for determining the convex hull of a finite planar set”. In: *Information Processing Letters* 1.4 (1972), pp. 132–133.

[36] Ulf Grenander. “On the theory of mortality measurement”. In: *Scandinavian Actuarial Journal* 1956.1 (1956), pp. 70–96.

[37] Yann Guermeur. “Combining multi-class SVMs with linear ensemble methods that estimate the class posterior probabilities”. In: *Communications in Statistics-Theory and Methods* 42.16 (2013), pp. 3011–3030.

[38] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. “The WEKA data mining software: an update”. In: *ACM SIGKDD Explorations Newsletter* 11.1 (2009), pp. 10–18.

[39] Godfrey Harold Hardy, John Edensor Littlewood, and George Pólya. *Inequalities*. Cambridge University Press, 1952.

[40] Shen-Shyang Ho and Harry Wechsler. “A martingale framework for detecting changes in data streams by testing exchangeability”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.12 (2010), pp. 2113–2127.

[41] Shen-Shyang Ho and Harry Wechsler. “Transductive confidence machine for active learning”. In: *Proceedings of the International Joint Conference on Neural Networks*. Vol. 2. IEEE. 2003, pp. 1435–1440.

[42] Xiaoqian Jiang, Melanie Osl, Jihoon Kim, and Lucila Ohno-Machado. “Smooth isotonic regression: A new method to calibrate predictive models”. In: *AMIA Summits on Translational Science Proceedings* (2011), p. 16.

[43] Gerard Kan, Cees A Visser, Jacques J Koolen, and Arend J Dunning. "Short and long term predictive value of admission wall motion score in acute myocardial infarction. A cross sectional echocardiographic study of 345 patients." In: *Heart* 56.5 (1986), pp. 422–427.

[44] Ron Kohavi. "Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press. 1996, pp. 202–207.

[45] Antonis Lambrou, Harris Papadopoulos, Ilia Nouretdinov, and Alexander Gammerman. "Reliable probability estimates based on support vector machines for large multiclass datasets". In: *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer. 2012, pp. 182–191.

[46] John Langford and Bianca Zadrozny. "Estimating Class Membership Probabilities using Classifier Learners". In: *Tenth International Conference on Artificial Intelligence and Statistics* (2005), p. 198.

[47] Yann Le Cun, Lionel D Jackel, Brian Boser, John S Denker, Henry P Graf, Ian Guyon, David Henderson, Richard E Howard, and William Hubbard. "Handwritten digit recognition: Applications of neural network chips and automatic learning". In: *IEEE Communications Magazine* 27.11 (1989), pp. 41–46.

[48] Erich L Lehmann and Joseph P Romano. *Testing statistical hypotheses*. Springer Science & Business Media, 2006.

[49] Valery Manokhin. "Multi-class probabilistic classification using inductive and cross Venn–Abers predictors". In: *Conformal and Probabilistic Prediction and Applications*. 2017, pp. 228–240.

[50] Thomas Melluish, Craig Saunders, Ilia Nouretdinov, and Volodya Vovk. "Comparing the Bayes and typicalness frameworks". In: *European Conference on Machine Learning*. Springer. 2001, pp. 360–371.

[51] Marvin Minsky and Seymour A Papert. *Perceptrons: An Introduction to Computation Geometry*. 1969, pp. 355–368.

[52] Allan H Murphy. "A new vector partition of the probability score". In: *Journal of Applied Meteorology* 12.4 (1973), pp. 595–600.

[53] Allan H Murphy and Edward S Epstein. "Verification of probabilistic predictions: A brief review". In: *Journal of Applied Meteorology* 6.5 (1967), pp. 748–755.

[54] John Ashworth Nelder and R Jacob Baker. *Generalized linear models*. Wiley Online Library.

[55] Alexandru Niculescu-Mizil and Richard A Caruana. "Obtaining Calibrated Probabilities from Boosting". In: *arXiv preprint arXiv:1207.1403* (2012).

[56] Genevieve B Orr and Klaus-Robert Muller. *Neural networks: tricks of the trade*. Springer, 2003.

[57] Harris Papadopoulos. "Reliable probabilistic classification with neural networks". In: *Neurocomputing* 107 (2013), pp. 59–68.

[58] Harris Papadopoulos, Alex Gammerman, and Volodya Vovk. "Reliable diagnosis of acute abdominal pain with conformal prediction". In: *Engineering Intelligent Systems* 17.2 (2009), p. 127.

[59] John C Platt. *Advances in Large-Margin Classifiers: Probabilities for SV Machines*. MIT Press, 2000.

[60] Foster Provost and Pedro Domingos. "Well-trained PETs: Improving probability estimation trees". In: *CDER WorkingPaper, Stern School of Business. New York, NY: New York University* (2000).

[61] J Ross Quinlan. *C4. 5: programs for machine learning*. Morgan Kaufmann, 1992.

[62] J Ross Quinlan. "Simplifying decision trees". In: *International Journal of Man-Machine Studies* 27.3 (1987), pp. 221–234.

[63] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[64] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536.

[65] Arthur L Samuel. “Some studies in machine learning using the game of checkers”. In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229.

[66] Thomas Sellke, MJ Bayarri, and James O Berger. “Calibration of ρ values for testing precise null hypotheses”. In: *The American Statistician* 55.1 (2001), pp. 62–71.

[67] Vincent G Sigillito, Simon P Wing, Larrie V Hutton, and Kile B Baker. “Classification of radar returns from the ionosphere using neural networks”. In: *Johns Hopkins APL Technical Digest* 10.3 (1989), pp. 262–266.

[68] Patrice Simard, Yann LeCun, and John S Denker. “Efficient pattern recognition using a new transformation distance”. In: *Advances in Neural Information Processing Systems*. 1993, pp. 50–58.

[69] Jack W Smith, JE Everhart, WC Dickson, WC Knowler, and RS Johannes. “Using the ADAP learning algorithm to forecast the onset of diabetes mellitus”. In: *Proceedings of the Annual Symposium on Computer Application in Medical Care*. American Medical Informatics Association. 1988, p. 261.

[70] Ingo Steinwart. “On the influence of the kernel on the consistency of support vector machines”. In: *Journal of Machine Learning Research* 2.Nov (2001), pp. 67–93.

[71] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[72] Charles J Stone. “Consistent nonparametric regression”. In: *The Annals of Statistics* (1977), pp. 595–620.

[73] W Nick Street, William H Wolberg, and Olvi L Mangasarian. “Nuclear feature extraction for breast tumor diagnosis”. In: *Biomedical Image Processing and Biomedical Visualization*. Vol. 1905. International Society for Optics and Photonics. 1993, pp. 861–871.

[74] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[75] Kazuko Takahashi, Hiroya Takamura, and Manabu Okumura. “Direct estimation of class membership probabilities for multiclass classification using multiple scores”. In: *Knowledge and Information Systems* 19.2 (2009), pp. 185–210.

[76] Robert Tibshirani, Jerome Friedman, et al. *The elements of statistical learning: data mining, inference, and prediction*. Springer Heidelberg, 2001.

[77] Alan M Turing. "Computing machinery and intelligence". In: *Mind* 59.236 (1950), pp. 433–460.

[78] US and Joint Economic Committee Congress. "98th Congress, 2d Session". In: *An Act for the Budget Authorization of the Department of Defense* 40 (1984).

[79] Peter Van Der Putten and Maarten van Someren. "CoIL challenge 2000: The insurance company case". In: *Leiden Institute of Advanced Computer Science Technical Report* 9 (2000), pp. 1–43.

[80] Vladimir Vapnik. *Statistical learning theory*. Vol. 1. Wiley New York, 1998.

[81] Vladimir Vapnik and Rauf Izmailov. "Synergy of monotonic rules". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 4722–4754.

[82] Vladimir Vovk. "A logic of probability, with application to the foundations of statistics". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1993), pp. 317–351.

[83] Vladimir Vovk. "The fundamental nature of the log loss function". In: *Fields of Logic and Computation II*. Springer, 2015, pp. 307–318.

[84] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer Science and Business Media, 2005.

[85] Vladimir Vovk and Ivan Petej. "Venn-abers predictors". In: *arXiv preprint arXiv:1211.0025* (2012).

[86] Vladimir Vovk and Ivan Petej. "Venn-Abers predictors". In: *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*. AUAI Press. 2014, pp. 829–838.

[87] Vladimir Vovk, Ivan Petej, and Valentina Fedorova. "From conformal to probabilistic prediction". In: *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer. 2014, pp. 221–230.

[88] Vladimir Vovk, Ivan Petej, and Valentina Fedorova. "From conformal to probabilistic prediction". In: *arXiv preprint arXiv:1406.5600* (2014).

[89] Vladimir Vovk, Ivan Petej, and Valentina Fedorova. "Large-scale probabilistic predictors with and without guarantees of validity". In: *Advances in Neural Information Processing Systems*. 2015, pp. 892–900.

[90] Vladimir Vovk, Ivan Petej, and Valentina Fedorova. "Large-scale probabilistic predictors with and without guarantees of validity". In: *arXiv preprint arXiv:1511.00213* (2015).

[91] Vladimir Vovk, Glenn Shafer, and Ilia Nouretdinov. "Self-calibrating Probability Forecasting". In: *Proceedings of the 16th International Conference on Neural Information Processing Systems*. MIT Press. 2003, pp. 1133–1140.

[92] Vladimir Vovk, Valentina Fedorova, Ilia Nouretdinov, and Alexander Gammerman. "Criteria of efficiency for conformal prediction". In: *Symposium on Conformal and Probabilistic Prediction with Applications*. Springer. 2016, pp. 23–39.

[93] Vladimir Vovk, Ilia Nouretdinov, Valentina Fedorova, Ivan Petej, and Alex Gammerman. "Criteria of efficiency for set-valued classification". In: *Annals of Mathematics and Artificial Intelligence* (2017), pp. 1–26.

[94] Larry Wasserman. "Bayesian model selection and model averaging". In: *Journal of Mathematical Psychology* 44.1 (2000), pp. 92–107.

[95] Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng. "Probability estimates for multi-class classification by pairwise coupling". In: *Journal of Machine Learning Research* 5.Aug (2004), pp. 975–1005.

[96] Bianca Zadrozny and Charles Elkan. "Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers". In: *In Proceedings of the Eighteenth International Conference on Machine Learning*. Citeseer. 2001.

[97] Bianca Zadrozny and Charles Elkan. "Transforming classifier scores into accurate multiclass probability estimates". In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2002, pp. 694–699.

[98] Jian Zhang and Yiming Yang. “Probabilistic score estimation with piecewise logistic regression”. In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ACM. 2004, p. 115.

[99] Tong Zhang. “Statistical behavior and consistency of classification methods based on convex risk minimization”. In: *Annals of Statistics* 32 (2004), pp. 56–85.

[100] Chenzhe Zhou, Ilia Nouretdinov, Zhiyuan Luo, Dmitry Adamskiy, Luke Randell, Nick Coldham, and Alex Gammerman. “A comparison of Venn Machine with Platt’s method in probabilistic outputs”. In: *Artificial Intelligence Applications and Innovations* (2011), pp. 483–490.