Large-scale probabilistic prediction with and without validity guarantees

Vladimir Vovk, Ivan Petej, and Valentina Fedorova

практические выводы
теории вероятностей
могут быть обоснованы
в качестве следствий
гипотез о *предельной*
при данных ограничениях
сложности изучаемых явлений

On-line Compression Modelling Project (New Series)

Working Paper #13

First posted November 2, 2015. Last revised November 3, 2019.

Project web site:
http://alrw.net

Abstract

This paper studies theoretically and empirically a method of turning machine-learning algorithms into probabilistic predictors that automatically enjoys a property of validity (perfect calibration) and is computationally 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 consistently more accurate than the existing methods in empirical studies.

The conference version of this paper published in *Advances in Neural Information Processing Systems 28*, 2015.

Contents

1 Introduction **1**

2 Inductive Venn–Abers predictors (IVAPs) **2**

3 Cross Venn–Abers predictors (CVAPs) **10**

4 Making probability predictions out of multiprobability ones **11**

5 Comparison with other calibration methods **12**
5.1 Platt's method 12
5.2 Isotonic regression 15

6 Empirical studies **16**

7 Conclusion **26**

References **28**

1 Introduction

Prediction algorithms studied in this paper belong to the class of Venn–Abers predictors, introduced in [19]. They are based on the method of isotonic regression [1] and prompted by the observation that when applied in machine learning the method of isotonic regression often produces miscalibrated probability predictions (see, e.g., [8, 9]); it has also been reported ([3], Section 1) that isotonic regression is more prone to overfitting than Platt's scaling [13] when data is scarce. The advantage of Venn–Abers predictors is that they are a special case of Venn predictors ([18], Chapter 6), and so ([18], Theorem 6.6) are always well-calibrated (cf. Proposition 1 below). They can be considered to be a regularized version of the procedure used by [20], which helps them resist overfitting.

The main desiderata for Venn (and related conformal, [18], Chapter 2) predictors are validity, predictive efficiency, and computational efficiency. This paper 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 are:

- Validity (in the form of perfect calibration) is satisfied by IVAPs automatically, and the experimental results reported in this paper 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 $O(n)$.

An advantage of Venn prediction over conformal prediction, which also enjoys validity guarantees, is that Venn predictors output probabilities rather than p-values, and probabilities, in the spirit of Bayesian decision theory, can be easily combined with utilities to produce optimal decisions.

In Sections 2 and 3 we discuss IVAPs and CVAPs, respectively. Section 4 is devoted to minimax ways of merging imprecise probabilities into precise probabilities and thus making IVAPs and CVAPs precise probabilistic predictors.

In this paper we concentrate on binary classification problems, in which the objects to be classified are labelled as 0 or 1. Most of machine learning algorithms are *scoring algorithms*, in that they output a real-valued score for each test object, which is then compared with a threshold to arrive at a categorical prediction, 0 or 1. As precise probabilistic predictors, IVAPs and CVAPs are ways of converting the scores for test objects into numbers in the range $[0, 1]$ that

can serve as probabilities, or *calibrating* the scores. In Section 5 we discuss two existing calibration methods, Platt's [13] and the method [20] based on isotonic regression, and compare them with IVAPs and CVAPs theoretically. Section 6 is devoted to experimental comparisons and shows that CVAPs consistently outperform the two existing methods.

2 Inductive Venn–Abers predictors (IVAPs)

In this paper 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 [19] 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 paper random variables are denoted by capital letters (e.g., X are random objects and Y are random labels).

Proposition 1. *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 2. *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 [2], 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 $k' \le k$ 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

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

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 never needed in this paper (unlike in the standard use of isotonic regression in machine learning, [20]): e.g., $f_1(s)$ is the value of the isotonic regression fitted to a sequence that already contains $(s, 1)$.

Proof of Proposition 1. 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 precompute 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^ϵ_i is the value of $f_\epsilon(s)$ when $s = s'_i$. Therefore, for all $i = 1, \dots, k'$:

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

Remark. There are several algorithms for performing isotonic regression on a partially, rather than linearly, ordered set: see, e.g., [2], Section 2.3 (although one of the algorithms described in that section, the Minimax Order Algorithm, was later shown to be defective [10, 12]). Therefore, IVAPs (and CVAPs below) can be defined in the situation where scores take values only in a partially ordered set; moreover, Proposition 1 will continue to hold. (For the reader familiar with the notion of Venn predictors we could also add that Venn–Abers predictors will continue to be Venn predictors, which follows from the isotonic regression being the average of the original function over certain equivalence classes.) 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 [16]. Suppose, e.g., that one prediction algorithm outputs (scalar) scores $s^1_1, \dots, s^1_k$ for the calibration objects $x_1, \dots, x_k$ and another outputs $s^2_1, \dots, s^2_k$ for the same calibration objects; we would like to use both sets of scores. We could merge the two sets of scores into composite vector scores, $s_i := (s^1_i, s^2_i)$, $i = 1, \dots, k$, and then classify a new object x as described earlier using its composite score $s := (s^1, s^2)$, where s^1 and s^2 are the scalar scores computed by the two algorithms and the partial order between composite scores is defined as usual,

$$(s^1, s^2) \preceq (t^1, t^2) \iff (s^1 \le t^1) \;\&\; (s^2 \le t^2).$$

Preliminary results reported in [16] in a related context suggest that the resulting predictor can outperform predictors based on the individual scalar scores. However, we will not pursue this idea further in this paper.

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 (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 1, which operates with a stack S (initially empty), computes the corners; it is a trivial modification of Graham's scan ([6]; [4], 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 $(b-a) \times (c-b) \le 0$ (resp. ≥ 0), where $\times$ stands for cross product of planar vectors; this avoids computing angles and divisions (see, e.g., [4], Section 33.1).

Algorithm 1 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.

The rest of the procedure for computing the vector F^1 is shown as Algorithm 2. The main data structure in Algorithm 2 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 2 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 $\left[\sum_{j=1}^{i-1} w_j, \sum_{j=1}^{i} w_j\right]$

Algorithm 1 Initializing the corners for computing F^1

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

Algorithm 2 Computing F^1

1: **while** $\neg$STACK-EMPTY(S)
2: PUSH(POP$(S), S'$)
3: **for** $i \in \{1, 2, \ldots, k'\}$
4: set F_i^1 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 $\overrightarrow{\text{TOP}(S'), \text{NEXT-TO-TOP}(S')}$
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
10: POP(S')
11: PUSH(P_{i-1}, S')
12: **return** F^1

corresponds to the calibration score s'_i, $i = 1, \ldots, k'$. 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 $i = 1, \ldots, k'$ of the main loop 3–11 we are computing F_i^1, 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

$$P_{-1}, P_0, P_1, \ldots, P_{k'} \tag{2}$$

corresponding to

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

(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 (2) but in fact contains a different point P_{i-1} (cf. line 5 of the algorithm) and corresponds to

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

(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

Algorithm 3 Initializing the corners for computing F^0

```
PUSH(P_{k'+1}, S)
PUSH(P_{k'}, S)
for i ∈ {k'−1, k'−2, ..., 0}
    while S.size > 1 and the angle formed by points NEXT-TO-TOP(S),
            TOP(S), and P_i makes a nonright turn
        POP(S)
    PUSH(P_i, S)
return S
```

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 $\overrightarrow{P_{i-1}, P_i}$).
- 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 1. *The worst-case computation time of Algorithms 1 and 2 is $O(k')$.*

Proof. In the case of Algorithm 1, see [4], Section 33.3. In the case of Algorithm 2, 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). □

For convenience of the reader wishing to program IVAPs and CVAPs, we also give the counterparts of Algorithms 1 and 2 for computing F^0: see Algorithms 3 and 4 below. In those algorithms, we do not need the point P_{-1} anymore; however, we need a new point $P_{k'+1} := P_{k'} + (1, 0)$. The stacks S and S' that they use are initially empty.

Algorithm 4 Computing F^0

```
while ¬Stack-Empty(S)
    Push(Pop(S), S')
for i ∈ {k', k'−1, ..., 1}
    set F_i^0 to the slope of overrightarrow(Top(S'), Next-To-Top(S'))
    P_i = P_{i−1} + P_{i+1} − P_i
    if P_i is at or above overrightarrow(Top(S'), Next-To-Top(S'))
        continue
    Pop(S')
    while S'.size > 1 and the angle formed by points P_i,
            Top(S'), and Next-To-Top(S') makes a nonright turn
        Pop(S')
    Push(P_i, S')
return F^0
```

Alternatively, we could use the algorithm for computing F^1 in order to compute F^0 by the following lemma.

Lemma 2. *For all $i \in \{1, \dots, k'\}$,*

$$F_i^0\left(s'_1, \dots, s'_{k'}, w_1, \dots, w_{k'}, y'_1, \dots, y'_{k'}\right) = 1 - F_i^1\left(-s'_1, \dots, -s'_{k'}, w_1, \dots, w_{k'}, 1 - y'_1, \dots, 1 - y'_{k'}\right) \quad (3)$$

(where the dependence on various parameters is made explicit).

Proof. The lemma follows from the equality

$$\mathrm{IR}\left[s_1, \dots, s_k, w_1, \dots, w_k, y_1, \dots, y_k\right](s_i) = 1 - \mathrm{IR}\left[-s_1, \dots, -s_k, w_1, \dots, w_k, 1 - y_1, \dots, 1 - y_k\right](-s_i), \quad (4)$$

where all variables are generic and IR stands for "isotonic regression" (it is the operator mapping a data set to its isotonic regression function). Indeed, (3) can be rewritten as

$$\begin{aligned}
&\mathrm{IR}\Big[s'_1, \dots, s'_{k'}, w_1, \dots, w_{i-1}, w_i + 1, w_{i+1}, \dots, w_{k'}, \\
&\qquad y'_1, \dots, y'_{i-1}, \frac{w_i y'_i}{w_i + 1}, y'_{i+1}, \dots, y'_{k'}\Big](s'_i) \\
&= 1 - \mathrm{IR}\Big[-s'_1, \dots, -s'_{k'}, w_1, \dots, w_{i-1}, w_i + 1, w_{i+1}, \dots, w_{k'}, \\
&\qquad 1 - y'_1, \dots, 1 - y'_{i-1}, \frac{w_i(1 - y'_i) + 1}{w_i + 1}, 1 - y'_{i+1}, \dots, 1 - y'_{k'}\Big](-s'_i),
\end{aligned}$$

which is a special case of (4) since

$$\frac{w_i(1 - y'_i) + 1}{w_i + 1} = 1 - \frac{w_i y'_i}{w_i + 1}. \qquad \square$$

Algorithm 5 BST(a, b) (to create the binary search tree, run BST$(1, k')$)

```
if b = a
    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
else if b = a + 1
    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 BST(b, b)
    return its root
else if
    c = ⌊(a + b)/2⌋
    construct the binary tree
    whose root has key s'_c and payload {F^0_c, F^1_c},
    left child is BST(a, c − 1),
    and right child is BST(c + 1, b)
    return its root
```

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 5, 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 5 is in fact more general than what we need: it computes the binary search tree for the scores $s'_a, s'_{a+1}, \dots, s'_b$ for $a \le b$; therefore, we need to run 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 6, 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 6; however, the algorithm uses for prediction the binary search tree built from T' and T'', and the bulk of work is done in Algorithms 1–5.

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 $O(k \log k)$.

Algorithm 6 IVAP(T', T'', x) // inductive Venn–Abers predictor

```
set N to the root of the binary search tree and compute the score s of x
while N is not a leaf
    if s < key(N)
        set N to N's left child
    else if s > key(N)
        set N to N's right child
    else   //  if s = key(N)
        return payload(N)
return payload(N)
```

- Running our procedure for pre-computing f_0 and f_1 takes time $O(k)$ (by Lemma 1).
- 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. [4], 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 $O(\log k)$ ([4], Chapter 13).

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 7, where 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 GM(p_1) is the geometric mean of $p_1^1, \dots, p_1^K$ and 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 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 ([4], Section 5.3) can do the last step in time $O(l)$. See 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.

Algorithm 7 CVAP(T, x) // cross-Venn–Abers predictor for training set T

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

4 Making probability predictions out of multi-probability ones

In CVAP (Algorithm 7) we merge the K multiprobability predictions output by K IVAPs. In this section we design a minimax way for merging them, essentially following [19]. For the log-loss function the result is especially simple, $\mathrm{GM}(p_1)/(\mathrm{GM}(1-p_0) + \mathrm{GM}(p_1))$.

Remark. 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

$$\log \frac{p_1^1}{p} + \dots + \log \frac{p_1^K}{p}, \tag{5}$$

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

$$\log \frac{1-p_0^1}{1-p} + \dots + \log \frac{1-p_0^K}{1-p}. \tag{6}$$

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) is decreasing and (6) is increasing in p).

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

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

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 [19], 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 Comparison with other calibration methods

The two alternative calibration methods that we consider in this paper are Platt's [13] and isotonic regression [20].

5.1 Platt's method

Platt's [13] 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 paper) 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

$$-\sum_{i=1}^{k} (t_i \log p_i + (1 - t_i) \log(1 - p_i)) \to \min, \tag{7}$$

where, in the simplest case, $t_i := y_i$ are the labels of the calibration observations (so that (7) minimizes the log loss on the calibration set). To obtain even better results, Platt recommends regularization:

$$t_i = t_+ := \frac{k_+ + 1}{k_+ + 2} \tag{8}$$

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

$$t_i = t_- := \frac{1}{k_- + 2} \tag{9}$$

for the calibration observations labelled 0 (if there are k_- of them). We can see from (8) and (9) that the predictions of Platt's predictor are always in the range

$$\left(\frac{1}{k_- + 2}, \frac{k_+ + 1}{k_+ + 2}\right). \tag{10}$$

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 3. *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 (10)). 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 $\mathrm{GM}(p_1) \geq 1/(k + 1)$:

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

□

It is clear that the end-points of the interval (10) 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 [21] (Section 3.3) observes that in the case of SVM and universal [14] 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 paper).

Proposition 3. *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*

$$C_m \|f\|_H^2 + \sum_{i=1}^{m} \phi(f(x_i)y_i) \to \min, \tag{11}$$

where $\phi(v) := (1-v)^+$ and H is a universal RKHS ([15], Definition 4.52). 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 [21] for a natural class of universal kernels related to neural networks. In general, see the proof of Theorem 8.1 in [15]. □

The intuition behind the SVM decision values clustering around ± 1 is very simple. SVM solves the optimization problem (11); 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

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

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

$$\mathbb{E}\Big(\mathrm{KL}\,(\eta \,||\, \mathbb{E}(\eta \mid \eta > 1/2))\, \mathbf{1}_{\eta > 1/2} + \mathrm{KL}\,(\eta \,||\, \mathbb{E}(\eta \mid \eta < 1/2))\, \mathbf{1}_{\eta < 1/2}\Big), \tag{12}$$

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 $\mathbb{E}(\eta\, \mathbf{1}_E)/\,\mathbb{P}(E)$.

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

For illustration, suppose $\eta := \eta(X)$ is distributed uniformly in $[0,1]$. It is easy to see that

$$\mathbb{E}(\eta \mid \eta > 1/2) = 3/4$$
$$\mathbb{E}(\eta \mid \eta < 1/2) = 1/4;$$

therefore, the excess loss (12) is

$$\mathbb{E}\Big(\mathrm{KL}\,(\eta \,||\, 3/4)\,\mathbf{1}_{\eta>1/2} + \mathrm{KL}\,(\eta \,||\, 1/4)\,\mathbf{1}_{\eta<1/2}\Big) = \mathbb{E}\Big(\eta \log_2 \eta + (1-\eta)\log_2(1-\eta)\Big)$$
$$+\, 2\,\mathbb{E}\left(\eta 1_{\eta>1/2} \log_2 \frac{4}{3} + \eta 1_{\eta<1/2} \log_2 4\right) \approx -0.7213 + 0.8113 = 0.09.$$

We can see that the Bayes log loss is 72.13%, whereas the best loss achievable by a sigmoid is 81.13%, 9 percentage points worse.

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 [3]); 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 [20]). In both cases, however, we can expect to get an infinite log loss when the test set becomes large enough. 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[1].

[1] One of the reviewers of the conference version of this paper proposed complementing the calibration set used in isotonic regression by two dummy observations: one with score $+\infty$ and labelled by 0 and the other with score $-\infty$ and labelled by 1.

6 Empirical studies

The main loss function (cf., e.g., [17]) that we use in our empirical studies is the *log loss*

$$\lambda_{\log}(p,y) := \begin{cases} -\log p & \text{if } y=1 \\ -\log(1-p) & \text{if } y=0, \end{cases} \tag{13}$$

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*

$$\lambda_{\mathrm{Br}}(p,y) := 4(y-p)^2. \tag{14}$$

We choose the coefficient 4 in front of $(y-p)^2$ in (14) and the base 2 of the logarithm in (13) 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).

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)

$$\mathrm{MLL} := \frac{1}{n}\sum_{i=1}^{n} \lambda_{\log}(p_i,y_i), \quad \mathrm{MBL} := \frac{1}{n}\sum_{i=1}^{n} \lambda_{\mathrm{Br}}(p_i,y_i), \tag{15}$$

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., [11] 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 4) and CVAPs with Platt's method [13] and the standard method [20] 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 [19], namely J48 decision trees (abbreviated to "J48"), J48 decision trees with bagging ("J48 bagging"), logistic regression (sometimes abbreviated to "logistic"), naive Bayes, neural networks, and support vector machines (SVM), as implemented in Weka [7] (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 [13, 20] or the methods proposed in this paper ("IVAP" or "CVAP" in the tables and figures).

We start our empirical studies with the `adult` data set available from the UCI repository [5] (this is the main data set used in [13] and one of the data sets used in [20]); 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 1. The six top plots report results for the log loss (namely, MLL, as defined in (15)) and the six bottom plots 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 Figure 1 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 [19], such prediction algorithms were 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 [3]), 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 Figure 1.

A natural question is whether CVAPs perform better than the alternative

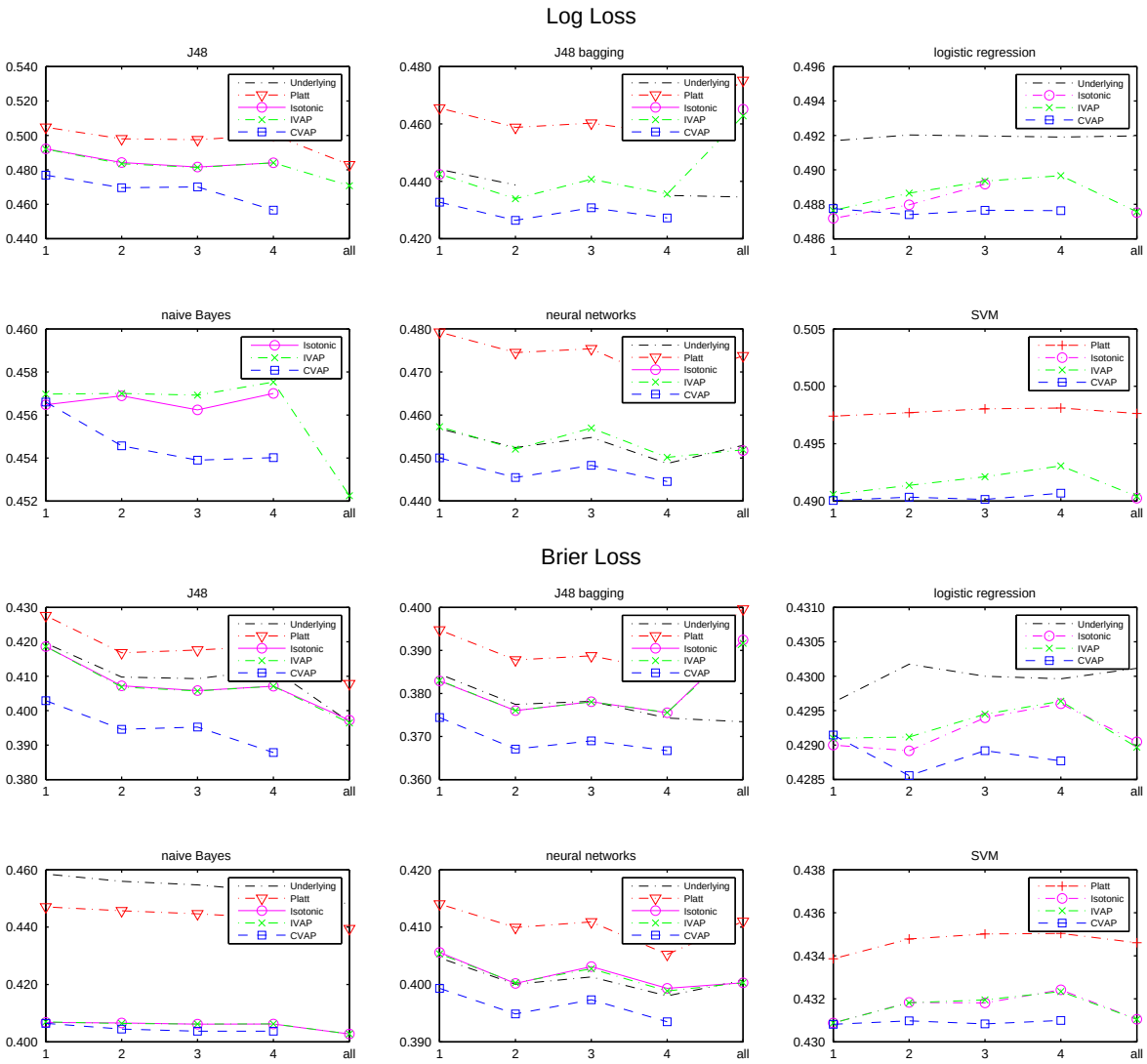

Figure 1: The log and Brier losses of the four calibration methods applied to the six prediction algorithms on the `adult` data set.

Table 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
naive Bayes	0.5534	∞	0.4839	0.4747
neural networks	0.5175	∞	0.5023	0.4805
SVM	0.5221	∞	0.5015	0.4997

Table 2: The analogue of Table 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
naive 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

calibration methods in Figure 1 (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 the lower half of Figure 1 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 1 and 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 [3]; 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 1 for log loss and 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 [19], where the results for IVAP (with the full training set used as both proper training and calibration sets, labelled "SVA" in the tables in [19]) 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 [19]).

The following information might help the reader in reproducing our results

(in addition to our code being posted on arXiv together with this paper). 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 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. 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); naive 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.

Additional experimental results

Figure 2 shows our results for the `covertype` data set (available from the UCI repository [5] and also known as `forest`). In converting this multiclass classification problem to binary we follow [3]: 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 3, 4, 5, and 6, respectively. The data sets are split into training and test sets in proportion 2:1, without randomization. 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 3, 4, and 5 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 Figure 6 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.

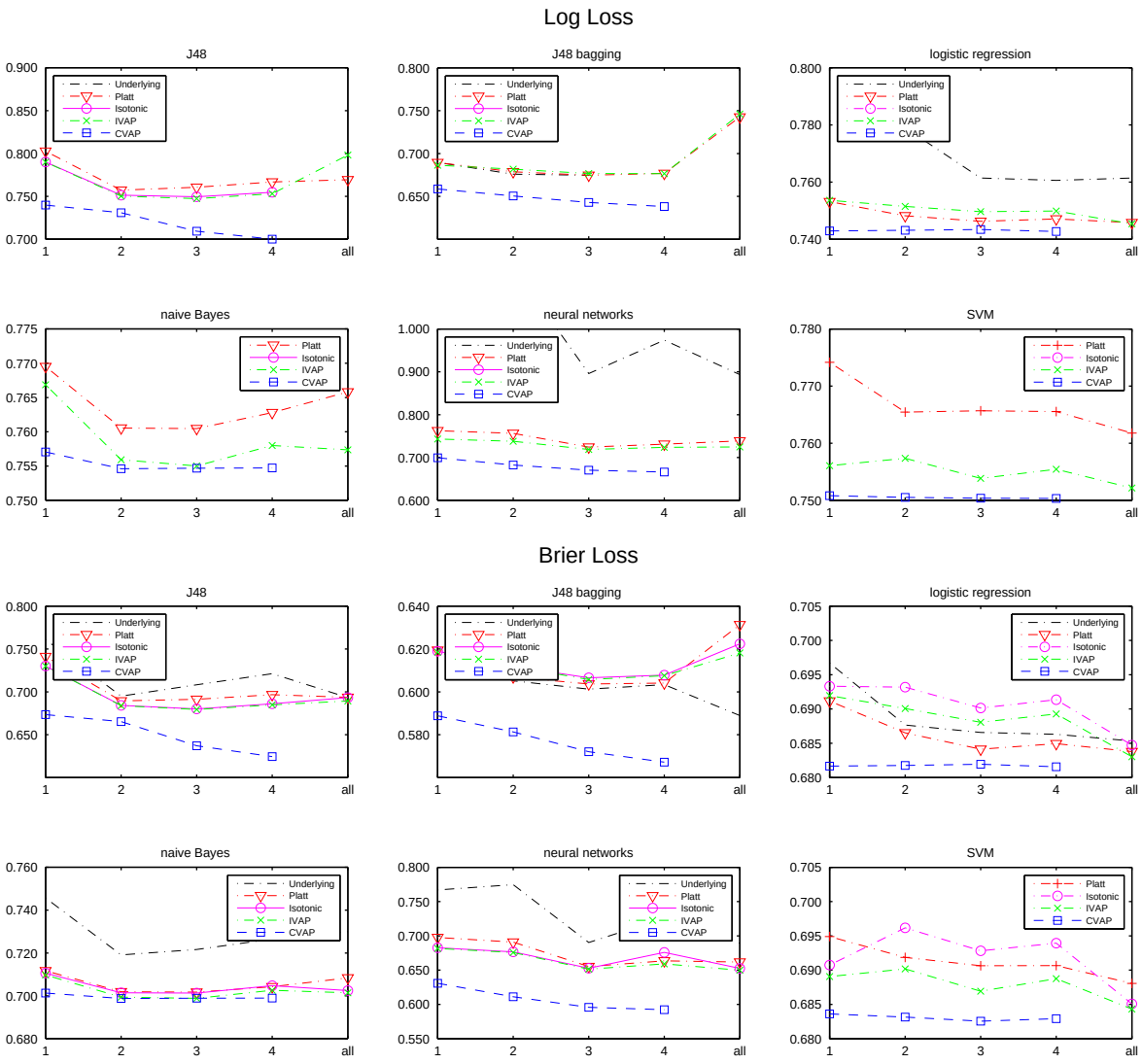

Figure 2: The analogue of Figure 1 for the `covertype` data set.

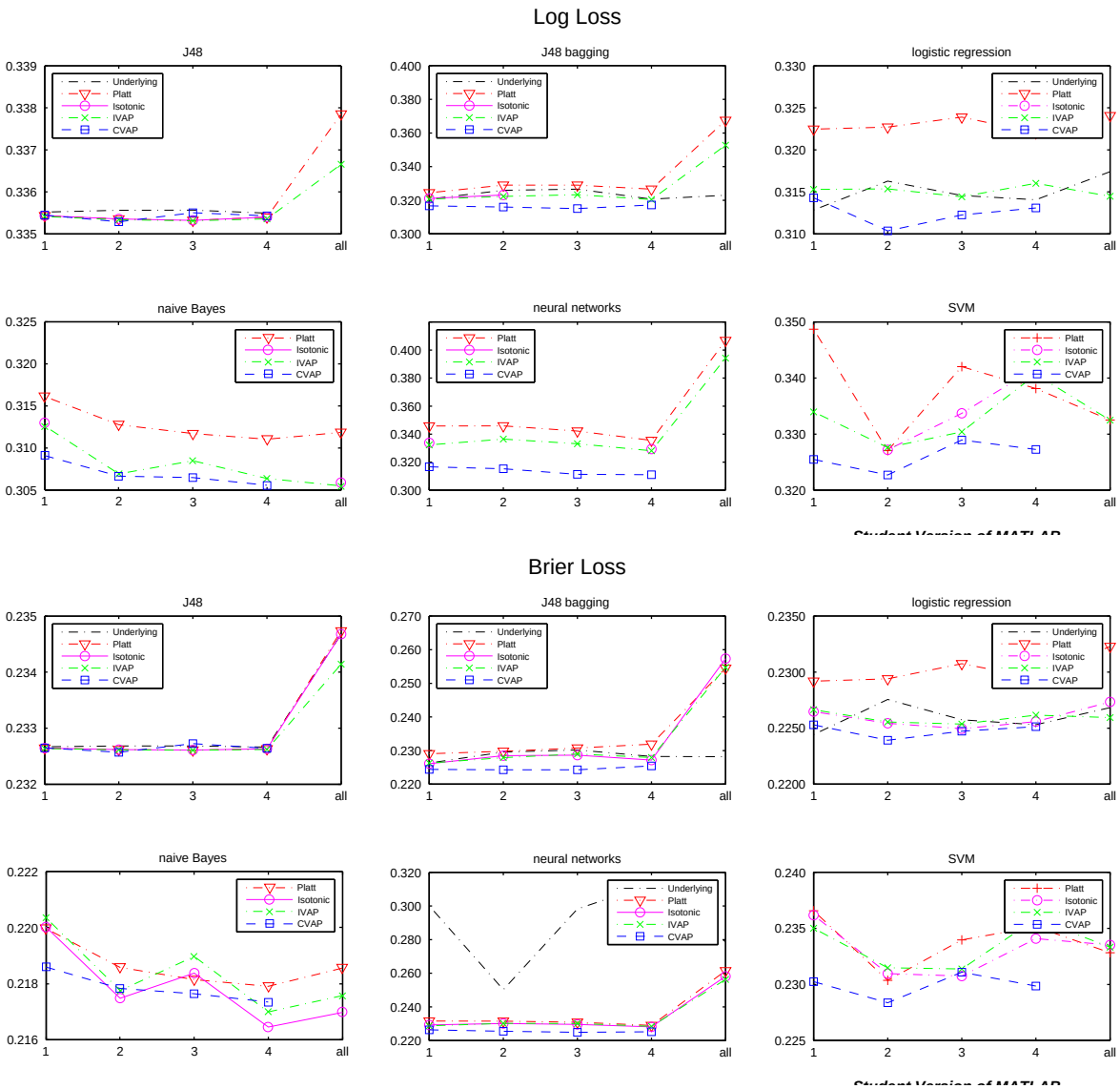

Figure 3: The analogue of Figure 1 for the `insurance` data set.

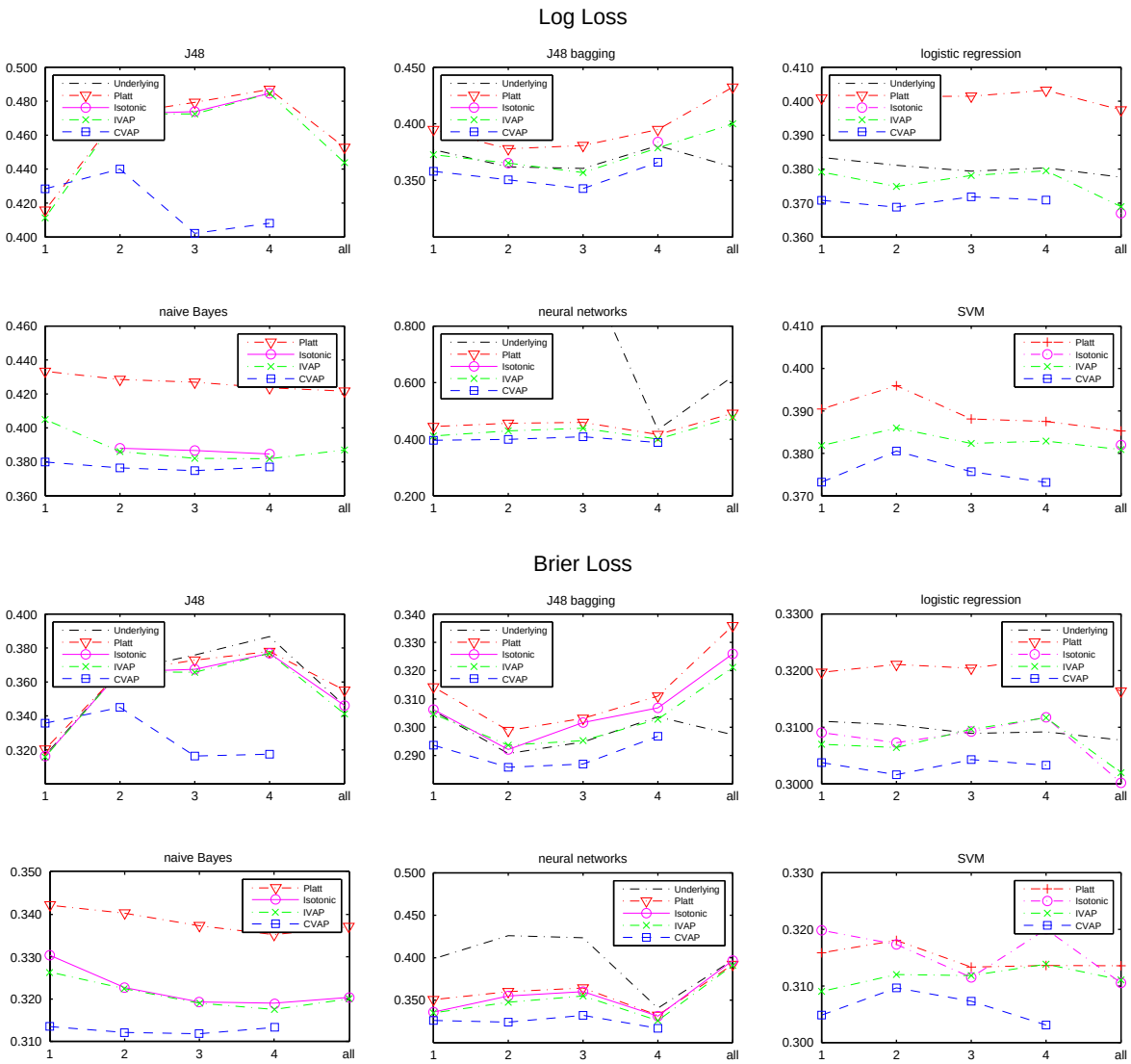

Figure 4: The analogue of Figure 1 for the `Bank Marketing` data set.

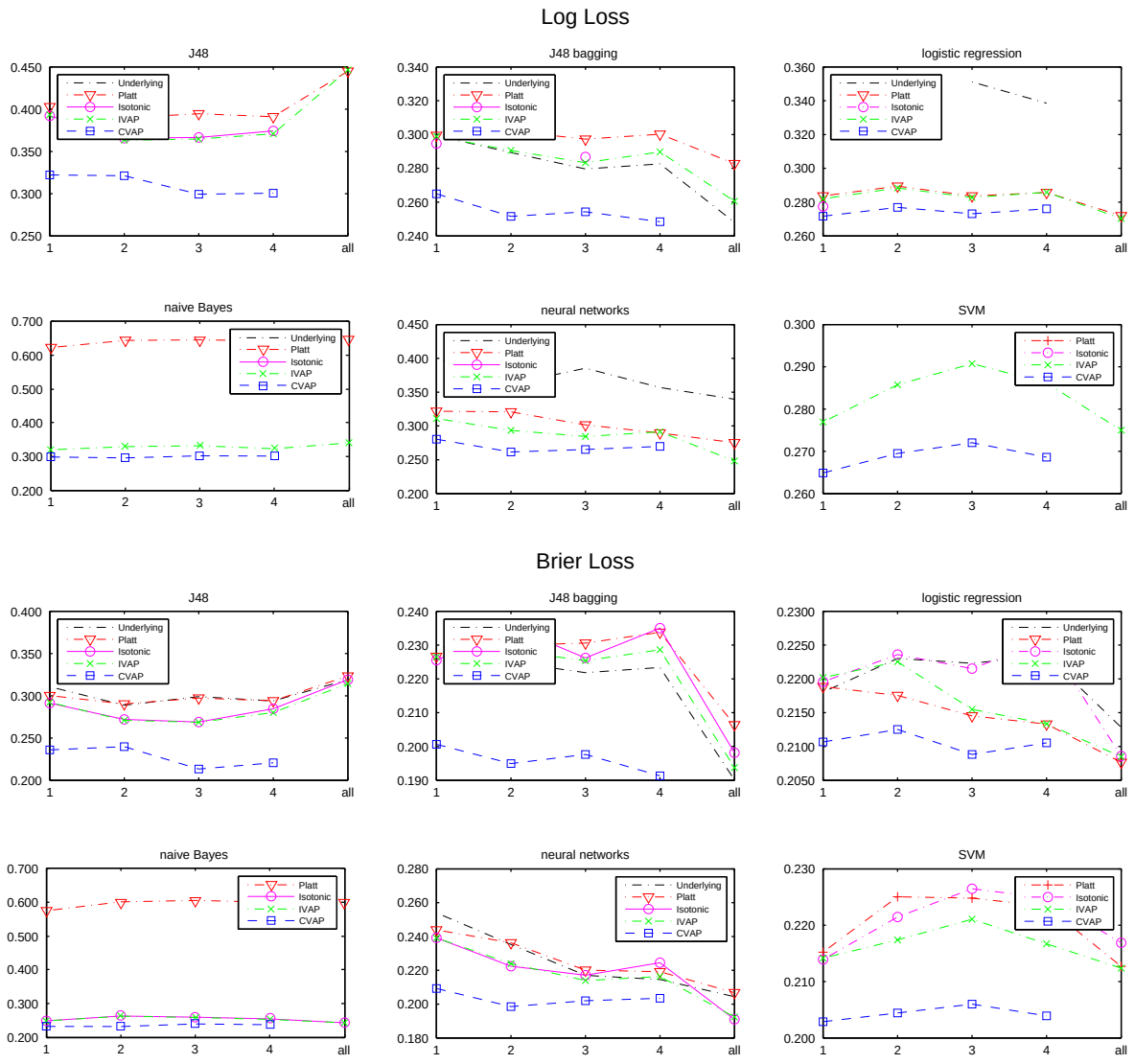

Figure 5: The analogue of Figure 1 for the `Spambase` data set.

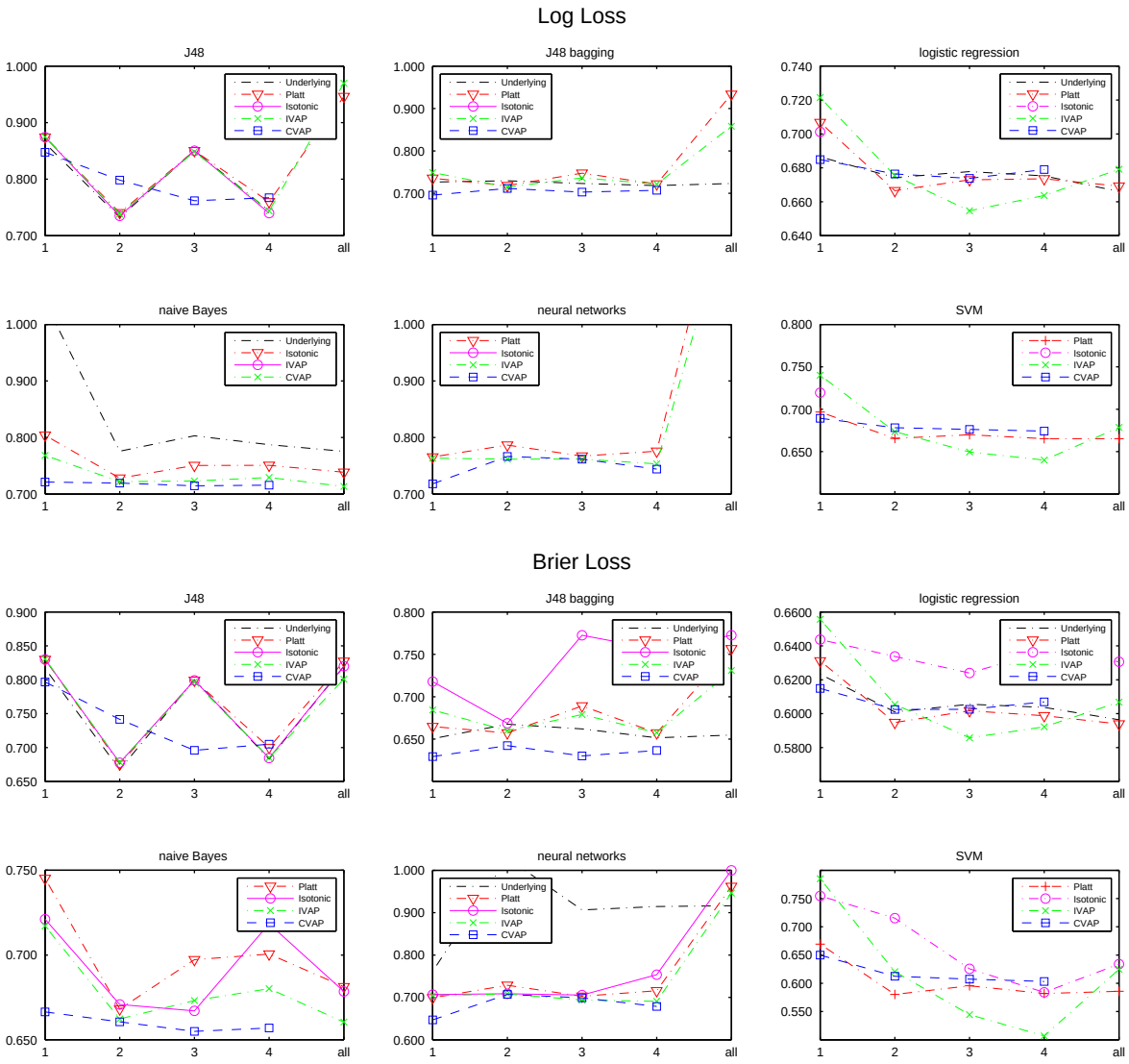

Figure 6: The analogue of Figure 1 for the data set `Statlog German Credit Data`.

And finally, Figure 7 shows the results for log loss and Brier loss 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 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 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 Figure 1),..., $4:1$ (which corresponds to $4:1$ in Figure 1, 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).

7 Conclusion

This paper 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.

This paper does 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 is one of the directions of further research.

Acknowledgments

We are grateful to the conference reviewers for numerous helpful comments and observations, to Vladimir Vapnik for sharing his ideas about exploiting synergy between different learning algorithms, and to participants in the conference *Machine Learning: Prospects and Applications* (October 2015, Berlin) for their questions and comments. The first author has been partially supported by EPSRC (grant EP/K033344/1) and AFOSR (grant "Semantic Completions"). The second and third authors are grateful to their home institutions for funding

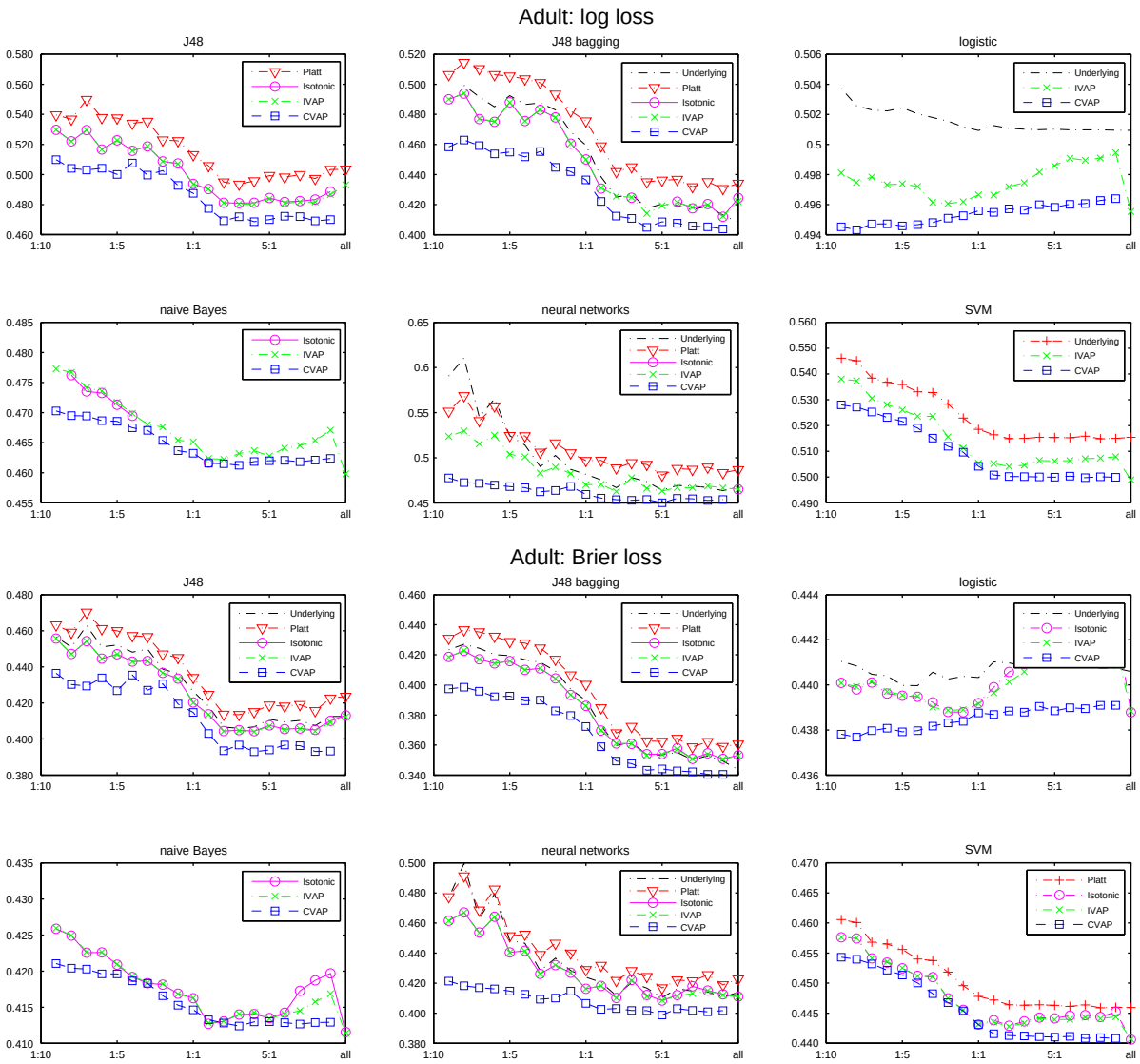

Figure 7: The log and Brier losses on the `adult` data set of the six prediction algorithms and four calibration methods

their trips to Montréal to attend NIPS 2015.

References

[1] Miriam Ayer, H. Daniel Brunk, George M. Ewing, W. T. Reid, and Edward Silverman. An empirical distribution function for sampling with incomplete information. *Annals of Mathematical Statistics*, 26:641–647, 1955.

[2] Richard E. Barlow, D. J. Bartholomew, J. M. Bremner, and H. Daniel Brunk. *Statistical Inference under Order Restrictions: The Theory and Application of Isotonic Regression.* Wiley, London, 1972.

[3] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the Twenty Third International Conference on Machine Learning*, pages 161–168, New York, 2006. ACM.

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* MIT Press, Cambridge, MA, third edition, 2009.

[5] A. Frank and A. Asuncion. UCI machine learning repository, 2015.

[6] Ronald L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972.

[7] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11:10–18, 2011.

[8] Xiaoqian Jiang, Melanie Osl, Jihoon Kim, and Lucila Ohno-Machado. Smooth isotonic regression: a new method to calibrate predictive models. *AMIA Summits on Translational Science Proceedings*, 2011:16–20, 2011.

[9] Antonis Lambrou, Harris Papadopoulos, Ilia Nouretdinov, and Alex Gammerman. Reliable probability estimates based on support vector machines for large multiclass datasets. In Lazaros Iliadis, Ilias Maglogiannis, Harris Papadopoulos, Kostas Karatzas, and Spyros Sioutas, editors, *Proceedings of the AIAI 2012 Workshop on Conformal Prediction and its Applications*, volume 382 of *IFIP Advances in Information and Communication Technology*, pages 182–191, Berlin, 2012. Springer.

[10] Chu-In Charles Lee. The Min-Max algorithm and isotonic regression. *Annals of Statistics*, 11:467–477, 1983.

[11] Allan H. Murphy. A new vector partition of the probability score. *Journal of Applied Meteorology*, 12:595–600, 1973.

[12] Gordon D. Murray. Nonconvergence of the minimax order algorithm. *Biometrika*, 70:490–491, 1983.

[13] John C. Platt. Probabilities for SV machines. In Alexander J. Smola, Peter L. Bartlett, Bernhard Schölkopf, and Dale Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 2000.

[14] Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001.

[15] Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer, New York, 2008.

[16] Vladimir N. Vapnik. Intelligent learning: Similarity control and knowledge transfer. Talk at the 2015 Yandex School of Data Analysis Conference *Machine Learning: Prospects and Applications*, 6 October 2015, Berlin.

[17] Vladimir Vovk. The fundamental nature of the log loss function. In Lev D. Beklemishev, Andreas Blass, Nachum Dershowitz, Berndt Finkbeiner, and Wolfram Schulte, editors, *Fields of Logic and Computation II: Essays Dedicated to Yuri Gurevich on the Occasion of His 75th Birthday*, volume 9300 of *Lecture Notes in Computer Science*, pages 307–318, Cham, 2015. Springer.

[18] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, New York, 2005.

[19] Vladimir Vovk and Ivan Petej. Venn–Abers predictors, On-line Compression Modelling project (New Series), `http://alrw.net`, Working Paper 7, June 2014 (first posted October 2012).

[20] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In Carla E. Brodley and Andrea P. Danyluk, editors, *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 609–616, San Francisco, CA, 2001. Morgan Kaufmann.

[21] Tong Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32:56–85, 2004.