Hierarchical Design of Fast Minimum Disagreement Algorithms

Malte Darnstädt[a], Christoph Ries[a], Hans Ulrich Simon[a]

[a]*Fakultät für Mathematik, Ruhr-Universität Bochum, D-44780 Bochum*

Abstract

We compose a toolbox for the design of Minimum Disagreement algorithms. This box contains general procedures which transform (without much loss of efficiency) algorithms that are successful for some d-dimensional (geometric) concept class $\mathcal{C}$ into algorithms which are successful for a $(d+1)$-dimensional extension of $\mathcal{C}$. An iterative application of these transformations has the potential of starting with a base algorithm for a trivial problem and ending up at a smart algorithm for a non-trivial problem. In order to make this working, it is essential that the algorithms are not proper, i.e., they return a hypothesis that is not necessarily a member of $\mathcal{C}$. However, the "price" for using a super-class $\mathcal{H}$ of $\mathcal{C}$ is so low that the resulting time bound for achieving accuracy ε in the model of agnostic learning is significantly smaller than the time bounds achieved by the up-to-date best (proper) algorithms.

We evaluate the transformation technique for $d = 2$ on both artificial and real-life data sets and demonstrate that it provides a fast algorithm, which can successfully solve practical problems on large data sets.

1. Introduction

In this paper, we are concerned with the Minimum Disagreement problem (sometimes also called Maximum Weight problem) associated with a family $\mathcal{C}$ of sets over some ground set $\mathcal{X}$: given a sequence $S = [(x_1, w_1), \ldots, (x_n, w_n)] \in (\mathcal{X} \times \mathbb{R})^n$ of points in $\mathcal{X}$ along with their weights, find a set $C \in \mathcal{C}$ whose total weight $W_S(C) := \sum_{i:x_i \in C} w_i$ is as large as possible. Note that $W_S(C)$ is maximized iff

$$E_S(C) := \sum_{i:w_i>0, x_i \notin C} w_i - \sum_{i:w_i<0, x_i \in C} w_i \quad (1)$$

is minimized. In statistical learning theory, $E_S(C)$ is called the *empirical error of C on S*, and this term plays a central role, especially in the model of agnostic learning [1].

Email addresses: `malte.darnstaedt@rub.de` (Malte Darnstädt), `christoph.ries@rub.de` (Christoph Ries), `hans.simon@rub.de` (Hans Ulrich Simon)

Although the Minimum Disagreement problem is intractable for a wide variety of classes [2, 1], it has been noticed by several researchers in an early stage of learning theory already that relatively simple and low-dimensional classification rules (e.g. axis-parallel rectangles [3, 4, 5], unions of intervals [6], or 2-level decision trees [7]) can be quite successful on benchmark data provided that these rules are given in terms of the (few) most relevant attributes. For this reason a couple of algorithms have been developed which solve the Minimum Disagreement problem w.r.t. some simple classes and run in polynomial time [8, 7, 9, 10].

It seems that efficient algorithms for the Minimum Disagreement problem have been found in the past mainly for geometric classes of a relatively low dimension d. The run-time of these algorithms usually exhibits an exponential dependence on d. Moreover, improving on the currently best time bounds does not appear to be an easy job. For instance, the algorithm from [8] solves the Minimum Disagreement problem for axis-parallel rectangles in time[1] $O(n^2 \log(n))$. It was not until recently [10] that a faster algorithm has been found (time $O(n^2)$ in case of axis-parallel rectangles or, more generally, time $O(n^d)$ in case of d-dimensional axis-parallel hyper-rectangles). Thus, one may easily get the impression that the early attempts of designing efficient Minimum Disagreement algorithms got stuck, and even modest improvements on the existing time bounds are not easy to obtain.

One means of escape from the marshy grounds of intractability is opened up by the usage of convex surrogate loss functions at the place of the discrete loss function underlying the Minimum Disagreement problem. This option is taken, for instance, by the Support Vector Machine [11, 12]. In this paper, we investigate another relaxation of the original problem: instead of searching for a set $C \in \mathcal{C}$ with the smallest possible value of $E_S(C)$, we bring suitably chosen classes $\mathcal{H}$ into play and search for a set $H \in \mathcal{H}$ such that $E_S(H) \leq \min_{C \in \mathcal{C}} E_S(C)$. While this approach is well known in the context of Boolean classes [2] and standard in the theory of agnostic learning [1], it is apparently not exploited to full extent in the context of geometric classes. Here is a short summary of our approach:

- We make use of the clever data structures that have been invented in the past in order to solve the Minimum Disagreement problem for low-dimensional geometric classes. We observe that these data structures naturally lead to the concept of "flexible" algorithms. Here, "flexibility" means that the underlying data structure can easily be updated in reaction to a modified weight parameter.

- We show that a flexible algorithm which solves the Minimum Disagreement problem for two d-dimensional classes, say $\mathcal{C}$ and $\mathcal{H}$, can be transformed (without much loss of efficiency) into a new flexible algorithm which solves

[1]The machine model used throughout the paper is a random-access machine with unit costs (even on real arithmetic).

the Minimum Disagreement problem for two (more expressive) $(d+1)$-dimensional classes. An iterative application of these transformations has the potential of starting with a base algorithm for a trivial problem and ending up at a smart algorithm for a non-trivial problem.

- By a suitable choice of the class $\mathcal{H}$, we obtain algorithms which achieve an accuracy of ε in the model of agnostic learning considerably faster than the best currently known algorithms do. For instance, we obtain a (non-proper) algorithm that agnostically learns axis-parallel rectangles in time $\tilde{O}(1/\varepsilon^2)$ while the learning procedure based on the up to date fastest proper algorithm from [10] needs time $\tilde{O}(1/\varepsilon^4)$. In this paper, $\tilde{O}$ is defined as Landau's O but additionally hides factors logarithmic in its argument and the dependency on confidence parameter δ.

It should be mentioned that fragments of our approach heavily build on existing work [8, 9]; in particular, the employed data structures are a variant of Segment Trees[2] [13]. But it seems to be the combination of three factors—data structures that provide flexibility, iteratively applicable transformations, clever choice of the class $\mathcal{H}$—which generates a surprising amount of additional horse power.

The paper is structured as follows. In Section 2, we formally introduce the notions that will play a central role in this paper and we call into mind some basic facts. Section 3 describes our transformation which maps a d-dimensional geometric concept class $\mathcal{C}$ to a $(d+1)$-dimensional extension of $\mathcal{C}$. Section 4 is the central section in this paper. Here we describe how an algorithm that (inproperly) solves the Minimum Disagreement problem for concept class $\mathcal{C}$ and hypothesis class $\mathcal{H}$ can be transformed (without much loss of efficiency) into an algorithm that solves the Minimum Disagreement problem for two higher-dimensional extensions of these classes. In Section 5, we discuss the implications of these results to inproper agnostic learning. In the final Section 6, we evaluate our transformation technique for $d = 2$ on both artificial and real-life data sets and demonstrate that it provides a fast algorithm, which can successfully solve practical problems on large data sets.

2. Definitions, Notations and Facts

In Section 2.1, we introduce the Minimum Disagreement problem. In Section 2.2, we describe its relation to agnostic PAC learning. In Section 2.3 we introduce our variant of Segment Trees, which is used later for storing partitions of $\mathbb{R}$ or of discrete subsets of $\mathbb{R}$.

[2] A Segment Tree is a binary tree storing a set of intervals with endpoints from a finite set of (sorted) real valued points. Each leaf of the tree corresponds to an elementary interval (either a point itself or an open interval between two points) and each internal node corresponds to the union of the intervals given by the children. Any interval over the points can easily be represented by an antichain of vertices.

2.1. The Minimum Disagreement Problem

Let $\mathcal{C}$ and $\mathcal{H}$ be two classes of sets over some ground set $\mathcal{X}$. A pair $(x, b) \in \mathcal{X} \times \{0, 1\}$ is called a *labeled example*. A sequence of labeled examples is called a *labeled sample* over $\mathcal{X}$. A sequence of the form $S = [(x_1, w_1), \ldots, (x_n, w_n)] \in (\mathcal{X} \times \mathbb{R})^n$ is called a *weighted sample* over $\mathcal{X}$. Intuitively, we should think of a labeled sample S as a special case of a weighted sample S': if $(x, 1)$ (resp. $(x, 0)$) occurs r-times in S, it will be represented as (x, r) (resp. as $(x, -r)$) in S'.

Let $\mathcal{C}$ and $\mathcal{H}$ be two classes over the same ground set $\mathcal{X}$. The *Minimum Disagreement problem* for $\mathcal{C}$ and $\mathcal{H}$ is denoted by $\mathrm{MinDis}(\mathcal{C}, \mathcal{H})$ in the sequel. Recall from Section 1 that it is the following problem: given a weighted sample S, find a hypothesis $H \in \mathcal{H}$ such that H does not perform worse on S than the best concept in $\mathcal{C}$ does, i.e., $E_S(H) \leq \min_{C \in \mathcal{C}} E_S(C)$ with E_S as given in (1).

Suppose that $\mathcal{H} = \cup_{\ell \geq 1} \mathcal{H}_\ell$ for a hierarchy $\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \mathcal{H}_3 \subseteq \ldots$ of classes. If an algorithm A solves $\mathrm{MinDis}(\mathcal{C}, \mathcal{H})$ and if A, when applied to a weighted sample of size n, returns a hypothesis from $\mathcal{H}_{\ell(n)}$, then we will say that A solves $\mathrm{MinDis}(\mathcal{C}, \mathcal{H}_{\ell(n)})$.

2.2. Agnostic PAC Learning

Let $\mathcal{X}$ be a ground set, let D be a distribution on $\mathcal{X} \times \{0, 1\}$, let $H \subseteq \mathcal{X}$, and let $\chi_H : \mathcal{X} \to \{0, 1\}$ denote the characteristic function of H, i.e., $\chi_H(x) = 1$ if $x \in H$ and $\chi_H(x) = 0$ otherwise. Then we define the *expected prediction error of H w.r.t. D* as follows:

$$\mathrm{er}_D(H) = \Pr_{(x,y) \sim D}[\chi_H(x) \neq y] \ .$$

Kearns, Schapire and Sellie [1] have presented a very general learning model. The following definition is obtained when their definition is restricted to the special case where all the functions to be learned are binary, where we deal with the zero-one loss function and where we allow arbitrary distributions on the set of labeled examples:

Definition 1. *Let $\mathcal{C}$ and $\mathcal{H}$ be classes of sets over some ground set $\mathcal{X}$. We say that $\mathcal{C}$ is* agnostically learnable by $\mathcal{H}$ *if there is an algorithm L and a function $n(\varepsilon, \delta)$ that is bounded by a fixed polynomial in $1/\varepsilon$ and $1/\delta$ such that, for any distribution D on $\mathcal{X} \times \{0, 1\}$ and any inputs $0 < \varepsilon, \delta \leq 1$, the following holds. Given a sequence S consisting of $n \geq n(\varepsilon, \delta)$ labeled examples drawn independently at random according to D, the algorithm L halts after a finite number of steps and returns a hypothesis $A(S) \in \mathcal{H}$ such that, with a probability*[3] *of at least $1 - \delta$, we have that*

$$\mathrm{er}_D(A(S)) \leq \inf_{C \in \mathcal{C}} \mathrm{er}_D(C) + \varepsilon \ . \tag{2}$$

If an algorithm agnostically learns $\mathcal{C}$ by $\mathcal{C}$, then it is called a proper *agnostic learning algorithm. If $\mathcal{H} = \cup_{\ell \geq 1} \mathcal{H}_\ell$ for an infinite hierarchy $\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \mathcal{H}_3 \subseteq$*

[3]taken over the random sequence S

... of classes and if L, when applied to a labeled sample of size n, returns a hypothesis from $\mathcal{H}_{\ell(n)}$, then we say that L agnostically learns *$\mathcal{C}$ by $\mathcal{H}_{\ell(n)}$.*

In the parlance of learning theory, the class $\mathcal{C}$ is referred to as the *concept class* over $\mathcal{X}$, and the class $\mathcal{H}$ is referred to as the *hypothesis class* over $\mathcal{X}$.

Recall from Section 2.1 that an algorithm A solving $\mathrm{MinDis}(\mathcal{C}, \mathcal{H})$ returns a hypothesis $H = A(S) \in \mathcal{H}$ such that $E_S(A(S)) \leq \min_{C \in \mathcal{C}} E_S(C)$ where E_S denotes the empirical error on S. If S is drawn independently at random according to D, then the empirical error E_S with $S \sim D^n$ may be a good approximation of the true error term er_D. Indeed, if

$$\forall C \in \mathcal{C}, \forall H \in \mathcal{H}: \ |E_S(C) - \mathrm{er}_D(C)| \leq \frac{\varepsilon}{2} \wedge |E_S(H) - \mathrm{er}_D(H)| \leq \frac{\varepsilon}{2} \qquad (3)$$

were valid[4], then the success condition (2) of agnostic PAC learning would follow immediately from $E_S(A(S)) \leq \min_{C \in \mathcal{C}} E_S(C)$. This raises the following question: how large must $n(\varepsilon, \delta)$ be chosen so that the uniform-convergence condition (3) holds with a probability of at least $1 - \delta$? The answer to this question involves the following combinatorial parameter associated with the hypothesis class $\mathcal{H}$.

Definition 2 ([14]). *The* VC-dimension *of a class $\mathcal{H}$ of sets over a ground set $\mathcal{X}$, denoted $VCD(\mathcal{H})$, is defined as the cardinality of the largest subset $M \subseteq \mathcal{X}$ such that every subset of M can be written in the form $M \cap H$ for some $H \in \mathcal{H}$. If there is no bound on the size of such sets M, then $VCD(\mathcal{H}) = \infty$.*

The following result leads to an upper bound on the required sample size $n(\varepsilon, \delta)$ in terms of ε, δ and $\mathrm{VCD}(\mathcal{H})$.

Lemma 3 ([15]). *There exist universal constants $c_1, c_2, c_3 > 0$ such that the following holds for any ground set $\mathcal{X}$, for any hypothesis class $\mathcal{H}$ over $\mathcal{X}$ of finite VC-dimension, say $VCD(\mathcal{H}) = d$, for any distribution D on $\mathcal{X} \times \{0, 1\}$, and for any $n \geq 1$:*

$$\Pr_{S \sim D^n}\left[\exists H \in \mathcal{H} : |\,\mathrm{er}_D(H) - E_S(H)| \geq \varepsilon\right] \leq c_1 c_2^d e^{-c_3 \varepsilon^2 n} \ . \qquad (4)$$

Setting the right hand-side of (4) less than or equal to δ and solving for n, it follows that there is a sample size $n = n(\varepsilon, \delta, d)$ of order $O((d + \log(1/\delta))/\varepsilon^2)$ so that, with a probability of at least $1 - \delta$, the uniform convergence condition (3) is satisfied. With this in mind, the following learnability results are easily obtained:

Corollary 4. *Suppose that $\mathcal{C}$ is a concept class and that $\mathcal{H}$ is a hypothesis class over some ground set $\mathcal{X}$. Suppose further that there is an algorithm A that solves*

[4]We may even replace "$\forall H \in \mathcal{H}$" by "for all hypotheses which are possibly chosen by the Minimum Disagreement algorithm".

the problem MinDis$(\mathcal{C},\mathcal{H})$*. Let d be an upper bound on the VC-dimensions of* $\mathcal{C}$*and $\mathcal{H}$. Then A can be transformed (without much loss of efficiency) into an algorithm L which agnostically learns $\mathcal{C}$ by $\mathcal{H}$. Moreover, the sample size $n = n(\varepsilon, \delta, d)$ required by L is bounded by $O((d + \log(1/\delta))/\varepsilon^2)$.*

Proof. An agnostic learning algorithm L is obtained simply by running A on a sufficiently large random sample. Some details follow.
Let D be a distribution on $\mathcal{X} \times \{0, 1\}$. According to Lemma 3, a sample size n of order $O(d + \log(1/\delta))/\varepsilon^2)$ is sufficient for achieving condition (3) with probability at least $1-\delta$. As discussed above already, condition (3) implies that the hypothesis returned by A on input $S \sim D^n$ satisfies (2). $\square$

The following result proves useful when the choice of the hypothesis class may depend on the sample size:

Corollary 5. *Let $\mathcal{C}$ be a concept class over some ground set $\mathcal{X}$. Let $\mathcal{H} = \cup_{\ell \geq 1}\mathcal{H}_\ell$ for an infinite hierarchy $\mathcal{C} \subseteq \mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \mathcal{H}_3 \subseteq \ldots$ of classes over the same ground set $\mathcal{X}$, and let the VC-dimension of $\mathcal{H}_\ell$ be bounded by $O(\ell)$. Let $\ell(n) = O(\log n)$. Suppose that there is an algorithm A that solves the problem* MinDis$(\mathcal{C}, \mathcal{H}_{\ell(n)})$*. Then A can be transformed (without much loss of efficiency) into an algorithm L which agnostically learns $\mathcal{C}$ by $\mathcal{H}_{\ell(n)}$. Moreover, the sample size $n = n(\varepsilon, \delta)$ required by L is bounded by $O\left(\frac{1}{\varepsilon^2}\log\left(\frac{1}{\varepsilon\delta}\right)\right)$.*

Proof. Again an agnostic learning algorithm L is obtained by running A on a sufficiently large random sample. Let D be a distribution on $\mathcal{X} \times \{0, 1\}$. Note that the VC-dimension of $\mathcal{H}_{\ell(n)}$ is bounded by $\ell(n) = O(\log(n))$. According to Lemma 3, there is a sufficiently large constant $c > 0$ such that a sample size n satisfying

$$n \geq \frac{c}{\varepsilon^2} \cdot \left(\log(n) + \log\left(\frac{1}{\delta}\right)\right) \tag{5}$$

is sufficient for achieving condition (3) (with $\mathcal{H}_{\ell(n)}$ in place of $\mathcal{H}$) with probability at least $1 - \delta$. As discussed above already, condition (3) implies that the hypothesis returned by A on input $S \sim D^n$ satisfies (2). It suffices therefore to analyze how large n has to be chosen so as to satisfy (5). The following condition implies (5):

$$\left(\frac{n}{2} \geq \frac{c\log(n)}{\varepsilon^2}\right) \wedge \left(\frac{n}{2} \geq \frac{c}{\varepsilon^2}\log\left(\frac{1}{\delta}\right)\right) \quad .$$

The first (resp. the second) inequality can be satisfied by some function $n = n(\varepsilon, \delta)$ of order $O(\log(1/\varepsilon)/\varepsilon^2)$ (resp. of order $O(\log(1/\delta)/\varepsilon^2)$. This shows that the required sample size is bounded by $O\left(\frac{1}{\varepsilon^2}\log\left(\frac{1}{\varepsilon\delta}\right)\right)$, as desired. $\square$

2.3. Ordered Partitions and their Tree Representations

Let $\mathcal{P}(k)$ denote the family of all ordered partitions of the reals into k non-empty intervals, i.e., $\mathcal{P}(k)$ consists of all k-tuples $(I_1, \ldots, I_k)$ such that

$I_1, \ldots, I_k \subseteq \mathbb{R}$ are pairwise disjoint non-empty intervals whose union equals $\mathbb{R}$, and the right endpoint of I_j coincides with the left endpoint of I_{j+1} for $j = 1, \ldots, k-1$. For instance $((-\infty, 0), [0, 10), [10, \infty))$ is a member of the family $\mathcal{P}(3)$. Analogously, let $\mathcal{P}'(k)$ denote the family of all ordered partitions of some bounded non-empty interval over the reals into k consecutive non-empty sub-intervals. For instance, $([-10, 0), [0, 10), [10, 20))$ is a member of the family $\mathcal{P}'(3)$.

A sub-interval $[c, d]$ of $[a, b]$ is said to be *left-aligned* (resp. *right-aligned*) in $[a, b]$ if $c = a$ (resp. $d = b$). It is called a *proper* sub-interval of $[a, b]$ if it does not coincide with $[a, b]$. If $[c, d] \subseteq (a, b)$ it is said to be *located in the interior* of $[a, b]$. Clearly, a proper sub-interval of $[a, b]$ is either left-aligned, right-aligned, or located in the interior of $[a, b]$. In the first (resp. second or third) case, we say that it is *of type "L"* (resp. *of type "R"* or *of type "I"*).

Let $Z \subset \mathbb{R}$ be finite. A partition of Z is said to be *ordered* if it is induced by an ordered partition of $\mathbb{R}$. Suppose that $(Z_1, \ldots, Z_k)$ is an ordered partition of Z into k classes $Z_1, \ldots, Z_k$ (so that $\max Z_i < \min Z_{i+1}$). Its *canonical extension* to a partition of the reals is defined as $(I_1, \ldots, I_k) \in \mathcal{P}'(k)$ where $I_1, \ldots, I_k$ are given by

$$\begin{aligned} I_1 = (-\infty, \max Z_1] \quad &, \quad I_2 = (\max Z_1, \max Z_2], \ \ldots \\ I_{k-1} = (\max Z_{k-2}, \max Z_{k-1}] \quad &, \quad I_k = (\max Z_{k-1}, \infty) \ . \end{aligned}$$

Suppose that $a, b \in \mathbb{R}$ so that $a \le b$. We define $[a : b] = [a, b] \cap \mathbb{Z}$. Moreover, for any finite set $Z \subset \mathbb{R}$, we define $[a : b]_Z = [a, b] \cap Z$. Sets of the form $[a : b]_Z$ will be called *intervals in Z* in what follows (or simply *intervals* if Z is clear from the context). Again we classify proper sub -intervals $[c : d]_Z \subset [a : b]_Z$ as being of type either "L" (if $\min[c : d]_Z = \min[a : b]_Z$) , "R" (if $\max[c : d]_Z = \max[a : b]_Z$) or "I" (otherwise). Let $(Z_1, \ldots, Z_k)$ be an ordered partition of $[a : b]_Z$. Its *canonical extension* to a partition of $[a, b]$ is then defined as $(I_1, \ldots, I_k) \in \mathcal{P}(k)$ where $I_1, \ldots, I_k$ are given by

$$\begin{aligned} I_1 = [a, \max Z_1] \quad &, \quad I_2 = (\max Z_1, \max Z_2], \ \ldots \\ I_{k-1} = (\max Z_{k-2}, \max Z_{k-1}] \quad &, \quad I_k = (\max Z_{k-1}, b] \ . \end{aligned}$$

Let $B = B_n$ be a complete binary tree with root r_B and with n leaves that are numbered $1, \ldots, n$ from left to right. For a node $u \in B$, let $B(u)$ be the subtree of B rooted at u, and let $l(u)$ (resp. $r(u)$) be the smallest (resp. largest) number of a leaf in $B(u)$. Then $I(u) = [l(u) : r(u)]$ is called the *$\mathbb{Z}$-interval represented by u.* Suppose that the i-th leaf of B stores a real number z_i such that $z_1 < \ldots < z_n$. Let $Z = \{z_1, \ldots, z_n\}$. Then $I_Z(u) = [z_{l(u)} : z_{r(u)}]_Z$ is called the *Z-interval represented by u.* As usual, two distinct nodes in B are said to be *independent* in none of them is a descendant of the other one. A set of pairwise independent nodes in B is called an *antichain in B*. An antichain A in B is called a *Z-representation of an interval* $[a, b] \subset \mathbb{R}$ *in B* if $[a : b]_Z = \cup_{u \in A} I_Z(u)$. An example of an antichain and the corresponding interval is given in Fig. 1. An antichain A is called *maximal* if it cannot be

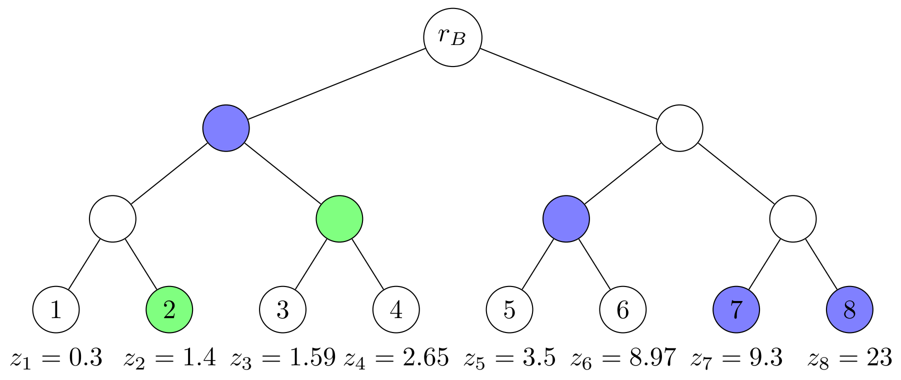


Figure 1: The complete binary tree B_8 with $Z = \{0.3, 1.4, 1.59, 2.65, 3.5, 8.97, 9.3, 23\}$. The green nodes form an anti-chain, which is a Z-representation of the interval $[1.4, 2.65]$. The blue nodes form a maximal anti-chain , which represents the partition $\{1, 2, 3, 4\}, \{5, 6\}, \{7\}, \{8\}$ of $[1 : 8]$ resp. $\{0.3, 1.4, 1.59, 2.65\}, \{3.5, 8.97\}, \{9.3\}, \{23\}$ of Z.

properly extended within B, i.e., if for every node $v \in B \setminus A$ there is a node $u \in A$ such that v is a descendant of u or vice versa. Every maximal antichain A in B represents a partition $P(A)$ of $[1 : n]$, respectively a partition $P_Z(A)$ of $Z = \{z_1, \ldots, z_n\}$, in the obvious manner. For instance $V = \{r_B\}$ represents the trivial partition with the single equivalence class $[1 : n]$ (resp. the trivial partition with the single equivalence class Z). The set of leaves in B represents the partition of $[1 : n]$ into n singletons $\{1\}, \ldots, \{n\}$ (resp. the partition of Z into the singletons $\{z_1\}, \ldots, \{z_n\}$). The other maximal antichains induce partitions which are in between these two extremes, one such example is given in Fig. 1. The following result is "folklore" (but we will sketch the proof for the sake of completeness):

Lemma 6. *For every $n \geq 2$, the following holds:*

1. *Let $s(n)$ denote the smallest integer such that every interval $[a, b] \subset \mathbb{R}$ has a Z-representation of size $s(n)$ in B_n. Then $s(n) \leq 2\lfloor \log n \rfloor$.*
2. *Let $k \geq 1$. Let $\ell_1(n) = 1$, $\ell_2(n) = \lceil \log n \rceil + 1$ and $\ell_k(n) = (k-1)\log(n)$. Furthermore, let $s(n, k)$ be the smallest integer such that, for every partition $(I_1, \ldots, I_k) \in \mathcal{P}(k)$, there exists a maximal antichain A of size at most $s(n, k)$ in B_n such that $P_Z(A)$ is a refinement of the partition induced by $(I_1, \ldots, I_k)$ on Z. Then $s(n, k) \leq \ell_k(n)$.*

Proof. For the sake of simplicity, we assume throughout the proof that n is a power of 2 (but introducing rounding operations would yield the general statement). Note that the left and the right subtree of the root r_B of $B = B_n$ are isomorphic to $B_{n/2}$, respectively. The recursive formulas that will be used within this proof are based on this observation.

1. Let $s_L(n)$ (resp. $s_R(n)$) denote the smallest integer such that every interval $[a, b] \subset \mathbb{R}$ with $a = z_1$ (resp. $b = z_n$) has a Z-representation of size $s_L(n)$

(resp. $s_R(n)$) in B_n. The following (in-)equalities for $X = L, R$ and every $n \geq 4$ are rather obvious:

$$\begin{array}{rcl} s_X(2) = 1 & \text{and} & s_X(n) \leq 1 + s_X(n/2)\ . \\ s(2) = 1 & \text{and} & s(n) \leq \max\{s(n/2), s_R(n/2) + s_L(n/2)\}\ . \end{array}$$

For instance, the inequality with $s(n)$ on the left hand-side follows from the observation that a Z-interval $[a : b]_Z$ either is fully contained in one of the sets $[z_1 : z_{n/2}]_Z$, $[z_{n/2+1} : z_n]_Z$ or it is the composition of a sub-interval of $[z_1 : z_{n/2}]_Z$ of type "R" (resp. the full interval $[z_1 : z_{n/2}]_Z$) with a sub-interval of $[z_{n/2+1} : z_n]_Z$ of type "L" (resp. the full interval $[z_{n/2+1} : z_n]_Z$). The above (in-)equalities imply that $s_X(n) \leq \log(n)$, and this implies that $s(n) \leq 2(\log(n) - 1)$.

2. It is easy to see that $s(n, 1) = 1$ and $s(n, 2) = \log(n) + 1$. Let k_0 (resp. k_1) be the number of intervals I_j among $I_1, \ldots, I_k$ such that $I_j \cap [z_1 : z_{n/2}]_Z \neq \emptyset$ (resp. $I_j \cap [z_{n/2+1} : z_n]_Z \neq \emptyset$). There is at most one interval among $I_1, \ldots, I_k$ which satisfies both conditions. It follows that $k_0 + k_1 \in \{k, k + 1\}$. Hence $s(n, k)$ is upper-bounded by

$$\max\{s(n/2, k_0) + s(n/2, k_1) : 1 \leq k_0, k_1 \leq n/2 \wedge k_0 + k_1 = k + 1\}\ . \quad (6)$$

Let $k \geq 3$. We assume inductively that $s(n, k') \leq (k' - 1)\log(n)$ holds for every $3 \leq k' < k$. In order to upper-bound (6), we proceed by case analysis:

Case 1: $k_0, k_1 \geq 3$. It follows inductively that

$$\begin{array}{rcl} s(n/2, k_0) + s(n/2, k_1) & \leq & (k_0 - 1)\log(n/2) + (k_1 - 1)\log(n/2) \\ & < & (k - 1)\log(n)\ . \end{array}$$

Case 2: $k_0 = 2$ and $k_1 = k - 1 \geq 3$ (or vice versa). It follows inductively that

$$\begin{array}{rcl} s(n/2, k_0) + s(n/2, k_1) & \leq & \log(n/2) + 1 + (k_1 - 1)\log(n/2) \\ & < & (k - 1)\log(n)\ . \end{array}$$

Case 3: $k = 3$ and $k_0 = k_1 = 2$. It follows inductively that

$$s(n/2, k_0) + s(n/2, k_1) \leq 2(\log(n/2) + 1) = 2\log(n) = (k - 1)\log(n)\ .$$

Case 4: $k_0 = 1$ and $k_1 = k$ (or vice versa). This case can occur only if $n \geq 2k$ (since, otherwise, the largest antichain in the left or the right subtree of r_B would be of size $n/2 < k$). We may therefore assume inductively that $s(n', k) \leq \ell_k(n)$ for every $n' < n$. Hence

$$s(n/2, k_0) + s(n/2, k_1) \leq 1 + (k_1 - 1)\log(n/2) < (k - 1)\log(n)\ .$$

In each case $\ell_k(n)$ upper-bounds $s(n, k)$. □

3. From Simple to More Complex Concept Classes

With each concept class $\mathcal{C}$ over some ground set $\mathcal{X}$ and with each $k \geq 1$, we associate the following concept classes over $\mathbb{R} \times \mathcal{X}$:

$$\mathcal{C}[k] = \left\{ \bigcup_{j=1}^{k'} (I_j \times C_j) : \ 0 \leq k' \leq k \wedge (I_1, \ldots, I_{k'}) \in \mathcal{P}(k') \wedge C_1, \ldots, C_{k'} \in \mathcal{C} \right\}$$

Analogously, let $\mathcal{C}'[k]$ be defined as $\mathcal{C}[k]$ with $\mathcal{P}$ replaced by $\mathcal{P}'$. Note that the empty set is a member of $\mathcal{C}[k]$ and $\mathcal{C}'[k]$.

In the sequel, $\mathcal{I}$ denotes the class of bounded intervals over the ground set $\mathbb{R}$, $\mathcal{R}$ denotes the class of bounded axis-parallel rectangles over the ground set $\mathbb{R}^2$, $\mathcal{I}_k$ denotes the class of unions of at most k bounded intervals, and $\mathcal{R}_k$ denotes the class of unions of at most k bounded axis-parallel rectangles.

Example 7. *Let $\mathcal{X} = \{x\}$ and $\mathcal{C}_1 = \{\mathcal{X}\}$ and $\mathcal{C}_2 = \{\emptyset, \mathcal{X}\}$. We identify the ground set $\mathbb{R} \times \{x\}$ with $\mathbb{R}$ in the obvious manner. Then, for each $k \geq 1$, $\mathcal{C}_1'[k]$ coincides with $\mathcal{I}$ and $\mathcal{C}_2'[2k-1]$ coincides with $\mathcal{I}_k$. Moreover, $\mathcal{I}_k$ is a subclass of $\mathcal{C}_2[2k+1]$.*

Example 8. *Obviously, $\mathcal{I}'[1] = \mathcal{R}$. The class $\mathcal{I}'[k]$ with $k \geq 2$ contains horizontally connected sequences of at most k bounded axis-parallel rectangles, i.e., it contains concepts of the form $\cup_{l=1}^{k'} (I_l \times J_l)$ with $k' \leq k$, $(I_1, \ldots, I_{k'}) \in \mathcal{P}'(k')$ and $J_1, \ldots, J_{k'} \in \mathcal{I}$.*

Example 9. *Obviously, $\mathcal{I}_s'[k]$ is the class over $\mathbb{R}^2$ whose concepts are of the form $\cup_{l=1}^{k'} (I_l \times U_l)$ with $k' \leq k$, $(I_1, \ldots, I_{k'}) \in \mathcal{P}'(k')$ and $U_1, \ldots, U_{k'} \in \mathcal{I}_s$. It is easy to see that $\mathcal{R}_k$ is a subclass of $\mathcal{I}_k'[2k-1]$ and $\mathcal{I}_k[2k+1]$, respectively. See Fig. 2 for an illustration.*

4. From Trivial to Smart Algorithms

We will assume from now on that a ground set $\mathcal{X}$ is equipped with a linear order and that a weighted sample of the form $S = [(x_1, w_1), \ldots, (x_n, w_n)] \in (\mathcal{X} \times \mathbb{R})^n$ is ordered so that $x_1 \leq \ldots \leq x_n$.

An algorithm that solves $\text{MinDis}(\mathcal{C}, \mathcal{C})$ is called a *proper* Minimum Disagreement algorithm for $\mathcal{C}$. An algorithm A that solves $\text{MinDis}(\mathcal{C}, \mathcal{H})$ is called a *flexible* Minimum Disagreement algorithm with time bounds T_1, T_2, T_3 if the following holds:

1. Given a sorted weighted sample $S = [(x_1, w_1), \ldots, (x_n, w_n)] \in (\mathcal{X} \times \mathbb{R})^n$, A builds a data structure $\text{DS}(S)$ in time $T_1(n)$.
2. After a modification of one of the weights in S, the data structure $\text{DS}(S)$ can be updated accordingly in time $T_2(n)$.

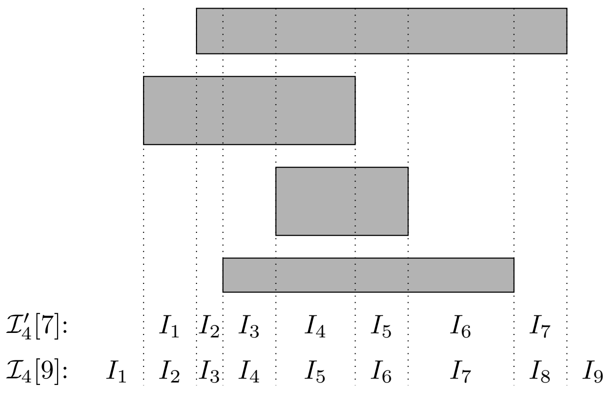

$\mathcal{I}_4'[7]$: I_1 I_2 I_3 I_4 I_5 I_6 I_7
$\mathcal{I}_4[9]$: I_1 I_2 I_3 I_4 I_5 I_6 I_7 I_8 I_9

Figure 2: An example showing that a union of 4 rectangles can be viewed as a concept from $\mathcal{I}_4'[7]$ or as a concept from $\mathcal{I}_4[9]$, respectively.

3. DS(S) implicitly represents a hypothesis $H(S) \in \mathcal{H}$ which satisfies

$$E_S(H(S)) \le \min_{C \in \mathcal{C}} E_S(C) \ . \tag{7}$$

Given DS(S) and $x \in \mathcal{X}$, it can be decided in time $T_3(n)$ whether $x \in \mathcal{H}(S)$.
4. Given DS(S), the quantity $E_S(H(S))$ can be computed in constant time.

Moreover we say that the data structure DS *can be merged efficiently* if, for every pair S_1, S_2 of sorted weighted samples (with the x-components in S_1 being not greater than the x-components in S_2), the data structure for the composition of S_1 and S_2 can be built in constant time from DS(S_1) and DS(S_2).

Here is a trivial example for a proper and flexible Minimum Disagreement algorithm, that we will use as a building block for the design of clever and highly non-trivial algorithms:

Example 10. *Let $\mathcal{C}_1 = \{\mathcal{X}\}$ for $\mathcal{X} = \{x\}$ be the trivial class that we had considered in Example 7 already. We claim that the (trivial) problem* MinDis($\mathcal{C}_1, \mathcal{C}_1$) *can be solved by a flexible algorithm with time bounds $T_1(n) = O(n)$, $T_2(n) = O(1)$ and $T_3(n) = O(1)$:*

- *For $S = [(x, w_1), \ldots, (x, w_n)]$, set* DS($S$) $:= W_S^- := \sum_{i:w_i<0} w_i$. *Thus,* DS($S$) *is simply a real number that can be determined in time $O(n)$.*
- *If a weight w_k is replaced by a new weight w_k', then* DS(S) *is updated in constant time by setting $W_S^- := W_S^- + \min\{w_k', 0\} - \min\{w_k, 0\}$.*
- DS(S) *represents $H(S) := \{x\}$, the only hypothesis in $\mathcal{H}$. The evaluation problem for $H(S)$ is trivial.*

- *Note that $E_S(\{x\}) = |W_S^-|$. Thus, given* $\mathrm{DS}(S) = W_S^-$, $E_S(H(S))$ *is computed in constant time.*

If the sample S is the composition of the samples S_1 and S_2, then $W_S^- = W_{S_1}^- + W_{S_2}^-$. Thus, the data structure DS *can be merged efficiently.*

Let $\mathcal{C}_2$ be the other trivial class that we had considered in Example 7. We briefly note that there is a flexible algorithm for $\mathrm{MinDis}(\mathcal{C}_2, \mathcal{C}_2)$ which has the same time bounds as the algorithm for $\mathrm{MinDis}(\mathcal{C}_1, \mathcal{C}_1)$.

In the sequel, we assume that $T_i(n) = o(n)$ for $i = 2, 3$ and $T_1(n)$ is of the form $nh(n)$ for some monotonically non-decreasing function $h(n) \geq 1$. From the latter assumption, it follows that

$$\sum_{i=1}^{s} n_i = n \implies \left(\sum_{i=1}^{s} T_1(n_i) \leq \sum_{i=1}^{s} (n_i h(n)) = nh(n) = T_1(n) \right) \ . \qquad (8)$$

Here comes the first main result of this section:

Theorem 11. *A flexible algorithm A solving* $\mathrm{MinDis}(\mathcal{C}, \mathcal{H})$ *with time bounds T_1, T_2, T_3 can be transformed into a flexible algorithm A' that solves* $\mathrm{MinDis}(\mathcal{C}'[1], \mathcal{H}'[2\lfloor \log n \rfloor])$ *with time bounds $T_i'(n) = O(\log(n) T_i(n))$ for $i = 1, 2$ and $T_3'(n) = O(\log(n) + T_3(n))$. Moreover, if the data structure used by A can be merged efficiently, then the first two time bounds for A' are even better, namely $T_1'(n) = O(T_1(n))$ and $T_2'(n) = O(\log(n) + T_2(n))$.*

Proof. We write vectors from $\mathbb{R} \times \mathcal{X}$ in the form $x' = (z, x)$ with $z \in \mathbb{R}$ and $x \in \mathcal{X}$, and we equip $\mathbb{R} \times \mathcal{X}$ with the lexicographic order. Let

$$S' = [(x_1', w_1), \ldots, (x_n', w_n)] \in (\mathbb{R} \times \mathcal{X} \times \mathbb{R})^n \qquad (9)$$

be a lexicographically sorted weighted sample. Let $S = [(x_1, w_1), \ldots, (x_n, w_n)] \in (\mathcal{X} \times \mathbb{R})^n$ be the sequence obtained by stripping off the z-coordinates of the items in S'. Note that segments of S with the same z-coordinate are sorted according to the linear order of $\mathcal{X}$. Let $z_1' < \ldots < z_{n'}'$ with $n' \leq n$ be the sorted sequence of distinct z-coordinates of the items in S' and let $Z = \{z_1', \ldots, z_{n'}'\}$. For each interval $[l : r] \subseteq [1 : n']$, we define $S'[l : r]$ as the coherent sub-sequence of S' consisting of all items in S' whose z-coordinate lies in the interval $[z_l' : z_r']$, i.e.,

$$S'[l : r] = \{(x_k', w_k) : \ z_l' \leq z_k \leq z_r'\} \ .$$

Let $S[l : r]$ be the corresponding list with z-coordinates omitted. Let B be a complete binary tree with n' leaves which are numbered $1, \ldots, n'$ from left to right. With each node u in B, we associate the following pieces of information:

1. $l(u) \in [1 : n']$ (resp. $r(u) \in [1 : n']$) is the number of the leftmost (resp. rightmost) leaf in the sub-tree of B induced by u.
2. $S(u)$ is defined as the "sorted version" of $S[l(u) : r(u)]$, i.e., it contains the same items as $S[l(u) : r(u)]$ but in $S(u)$ they are ordered according to non-decreasing x-values.

3. DS(u) = DS$(S(u))$, i.e., DS(u) is the data structure returned by the algorithm A on input $S(u)$.
4. $d_0(u) = E_{S(u)}(\emptyset)$. Note that $d_0(u)$ equals the sum of all positive weights that are found in $S(u)$.
5. Let $H(u) = H(S(u))$, i.e., $H(u)$ is the hypothesis which is represented by the data structure DS(u). Let $d_1(u) = E_{S(u)}(H(u))$. We may conclude from (7) that, for all nodes u in B,
$$d_1(u) = E_{S(u)}(H(u)) \le \min_{C \in \mathcal{C}} E_{S(u)}(C) \ . \tag{10}$$
6. Recall from Section 2.3 that $I(u) = [l(u) : r(u)]$ and $I_Z(u) = [z'_{l(u)} : z'_{r(u)}]_Z$. For $a, b \in I(u)$, define $J(a,b) = [z'_a : z'_b]_Z$. Let $V(a,b)$ be the smallest Z-representation of $J(a,b)$ in $B(u)$ s.t. $J(a,b) = \cup_{v \in V(a,b)} I_Z(v)$. We say that
$$H_{a,b}(u) = \bigcup_{v \in V(a,b)} I_Z(v) \times H(v) \tag{11}$$
is the *hypothesis induced by* $J(a,b)$ *at node* u. Note that $|V(a,b)| \le 2\lfloor \log n \rfloor$ according to Lemma 6. Given the convention $\min \emptyset = \infty$, we set
$$\begin{aligned} d_I(u) &= \min_{a,b \in I(u): a \le b} E_{S'(u)}(H_{a,b}(u)) \ , \\ d_L(u) &= \min_{b \in I(u)} E_{S'(u)}(H_{l(u),b}(u)) \ , \\ d_R(u) &= \min_{a \in I(u)} E_{S'(u)}(H_{a,r(u)}(u)) \ , \end{aligned}$$
i.e., $d_I(u)$ is the error on $S'(u)$ of the best hypothesis among the ones which are induced by some sub-interval of $I(u)$ of type "I". The analogous remark applies to $d_L(u)$ and $d_R(u)$, respectively. The sub-interval $[a : b]$ of $I(u)$ of type "I" that satisfies $d_I(u) = E_{S'(u)}(H_{a,b}(u))$ is denoted $J_I(u) = [a_I(u) : b_I(u)]$ in what follows. The notations $J_L(u) = [l(u) : b_L(u)]$ and $J_R(u) = [a_R(u) : r(u)]$ are understood analogously.

The tree B augmented by $K = (K(u))_{u \in B}$ for
$$\begin{aligned} K(u) \ = \ & [l(u), r(u), \mathrm{DS}(u) \ , \ d_0(u), d_1(u), d_L(u), d_R(u), d_I(u) \ , \\ & J_L(u), J_R(u), J_I(u)] \end{aligned}$$
constitutes the data structure DS(S'). Leaving out of account the computation of K, B can be built in time $O(n)$. The additional piece of information, K, can be computed as follows:

1. The quantities $(l(u), r(u), d_0(u))_{u \in B}$ are easy to compute within $O(n)$ steps in a bottom-up fashion. The sorted sequences $(S(u))_{u \in B}$ can be computed bottom-up in time $O(n \log(n))$ (in the same way as it is done by "Mergesort").

2. Making use of (8), it is easy to see that, within $T_1(n)$ steps, we can compute $\mathrm{DS}(u)$ for all nodes at a fixed level. Thus, it takes time $O(T_1(n)\log(n))$ to compute $(\mathrm{DS}(u))_{u\in B}$. Moreover, if DS can be merged efficiently, then it is easy to see that the sequences $(S(u))_{u\in B}$ are not needed because $(\mathrm{DS}(u))_{u\in B}$ can be computed directly in time $O(T_1(n)+n)=O(T_1(n))$.
3. For each $u \in B$, the parameter $d_1(u)$ can be computed in constant time given $\mathrm{DS}(u)$ (due to the fourth property from the definition of a flexible Minimum Disagreement algorithm).
4. Given $(d_1(u))_{u\in B}$, we can compute the quantities $(d_L(u), d_R(u), d_I(u), J_L(u), J_R(u), J_I(u))_{u\in B}$ in a bottom-up fashion in time $O(n)$. Let u be a node with left child u_0 and right child u_1, then $d_R(u)$ is computed according to

$$d_R(u) = \min\{d_0(u_0)+d_R(u_1), d_0(u_0)+d_1(u_1), d_R(u_0)+d_1(u_1)\}\ .$$

   This equation mimics the following fact: a sub-interval of $I(u)$ of type "R" is either a sub-interval of $I(u_1)$ of type "R" or equals $I(u_1)$ or it is composed of a sub-interval of $I(u_0)$ of type "R" and $I(u_1)$. A similar reasoning shows that

$$\begin{aligned} d_L(u) &= \min\{d_L(u_0)+d_0(u_1), d_1(u_0)+d_0(u_1), d_1(u_0)+d_L(u_1)\}\ , \\ d_I(u) &= \min\{d_0(u_0)+d_L(u_1), d_0(u_0)+d_I(u_1), d_R(u_0)+d_0(u_1), \\ &\qquad d_I(u_0)+d_0(u_1), d_R(u_0)+d_L(u_1)\}\ . \end{aligned}$$

   Moreover, for each $X \in \{L, R, I\}$, the computation of $J_X(u)$ is just as easy as the computation of $d_X(u)$.

It follows from the previous discussion that $(K(u))_{u\in B}$ can be computed in time $O(\log(n)T_1(n))$. Moreover, if DS can be merged efficiently, then time $O(T_1(n))$ is sufficient.

Suppose that for one item in S', say the item (z_k, x_k, w_k), the weight parameter is modified. Let j be the unique index with $z'_j = z_k$ and let v be the leaf in B numbered j. Since $K(u)$ need not be changed for all nodes in B but only for those which are located on the path P from v to the root r_B of B, it follows that $(K(u))_{u\in B}$ can be updated in time $O(\log(n)T_2(n))$, or even in time $O(\log(n)+T_2(n))$ if DS can be merged efficiently.

Let $d_{min} = \min\{d_0(r_B), d_1(r_B), d_L(r_B), d_R(r_B), d_I(r_B)\}$. We now claim that $\mathrm{DS}(S')$ represents an easy-to-evaluate hypothesis $H(S') \in \mathcal{H}'[2\lfloor\log n\rfloor]$ that satisfies $d_{min} = E_{S'}(H(S'))$. This can be seen as follows. If $d_{min} = d_0(r_B)$, we set $H(S') = \emptyset$. If $d_{min} = d_1(r_B)$, we set $H(S') = H_{1,n'}(r_B) \in \mathcal{H}$. Finally, if $d_{min} = d_X(r_B)$ for $X \in \{I, L, R\}$, then we set $H(S')$ to a properly defined extension of $H_{a_X(r_B),b_X(r_B)}(r_B)$ (as explained below). The evaluation problem for $H(S') = \emptyset$ is trivial. As for the remaining cases, note first that $(z,x) \in H_{1,n'}(r_B)$ iff $z \in [z'_1, z'_{n'}]$ and $x \in H(r_B)$. Suppose now that $d_{min} = d_X(r_B)$. For the sake of brevity, let $a = a_X(r_B)$ and $b = b_X(r_B)$. If $z \notin [z'_a, z'_b]$, then clearly $(z,x) \notin H_{a,b}(r_B)$. We may therefore assume that $z \in [z'_a, z'_b]$. The intervals $I_Z(v)$ with $v \in V(a,b)$ form a partition of $[z'_a : z'_b]_Z$. Let $I^E_Z(v)$ with $v \in V(a,b)$

be the canonical extension to a partition of the continuous interval $[z'_a, z'_b]$. This induces the following extension of $H_{a,b}$:

$$H^E_{a,b} = \bigcup_{v \in V(a,b)} I^E_Z(v) \times H(v) \ .$$

Set $H(S') = H^E_{a,b}$. In order to test whether $(z,x) \in H(S')$ for some $z \in [z'_a, z'_b]$, we proceed as follows:

Step 1: Find the node $v \in V(a,b)$ such that $z \in I^E_Z(v)$.
The search for v is started at r_B and proceeds top-down. Assume that the search has reached a node u. Note that $u = v$ iff

$$d_1(v) = \min\{d_0(v), d_1(v), d_L(v), d_R(v), d_I(v)\} \ .$$

Suppose that we have not yet reached v. Then u must be an inner node. The search proceeds to the left child if $z \le z_{r(u)}$, and it proceeds to the right child of u otherwise. It follows that Step 1 takes time $O(\log(n)$.

Step 2: Check whether $x \in H(v)$.
Here, we make use of the fact that $(z,x) \in H(S')$ iff $x \in H(v)$. Checking whether $x \in H(v)$ takes time $T_3(n)$.

Thus time $O(\log(n) + T_3(n))$ is sufficient for the execution of both steps, as desired.
Clearly, d_{min} is retrieved from DS(S') in constant time. The following reasoning shows that $d_{min} \le \min_{C \in \mathcal{C}'[1]} E_{S'}(C)$:

- The claim is obvious if the concept from $\mathcal{C}'[1]$ with the best performance on S' assigns label 0 to all instances in S'. We may therefore assume that this is not the case. It follows that, without loss of generality, the concept from $\mathcal{C}'[1]$ with the best performance on S' is of the form $[z'_{a^*}, z'_{b^*}] \times C^*$ for some $a^*, b^* \in [1:n']$ with $a^* \le b^*$ and some $C^* \in \mathcal{C}$.
- The update scheme for $d_0(u), d_1(u), d_L(u), d_R(u), d_I(u)$ implies that $d_{min} \le \min_{a,b \in [1:n']: a \le b} E_{S'}(H_{a,b}(r_B))$. Specifically, $d_{min} \le E_{S'}(H_{a^*,b^*}(r_B))$.
- Finally, Condition (10) implies that $E_{S'}(H_{a^*,b^*}(r_B)) \le E_{S'}([a^*, b^*] \times C^*)$.

The fact that $d_{min} \le \min_{C \in \mathcal{C}'[1]} E_{S'}(C)$ concludes the proof of the theorem. □

Note that the proof of Theorem 11 is completely constructive. The minimum disagreement algorithms given in the following arise from an iterative application of Theorem 11 and the trivial Example 10.

Recall that $\mathcal{I}$ denotes the class of bounded real intervals. As discussed in Example 7, $\mathcal{I} = \mathcal{C}'_1[1] = \mathcal{C}'_1[2\lfloor \log n \rfloor]$. We immediately obtain the following result:

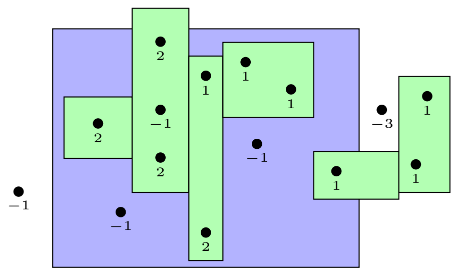


Figure 3: The qualitative difference between sets of maximal weight from $\mathcal{R}$ (in blue) and from $\mathcal{I}'[2\lfloor \log n \rfloor]$ (in green). The $n = 15$ sample points are shown together with their real-valued weights.

Theorem 12. *The transformation from Theorem 11 applied to the flexible algorithm for* $\mathrm{MinDis}(\mathcal{C}_1, \mathcal{C}_1)$ *from Example 10 yields a flexible algorithm that solves* $\mathrm{MinDis}(\mathcal{I}, \mathcal{I})$ *with time bounds* $T_1(n) = O(n)$ *and* $T_i(n) = O(\log(n))$ *for* $i = 2, 3$.

The flexible algorithm for $\mathrm{MinDis}(\mathcal{I}, \mathcal{I})$ resulting from Theorem 12 basically coincides with the algorithm for $\mathrm{MinDis}(\mathcal{I}, \mathcal{I})$ from [8]. However, since our transformation is general and can be iterated, we can now go one step further and obtain the following result:

Theorem 13. *The problem* $\mathrm{MinDis}(\mathcal{R}, \mathcal{I}'[2\lfloor \log n \rfloor])$ *can be solved by a flexible algorithm with time bounds* $T_1(n) = O(n \log(n))$, *and* $T_2(n) = O(\log^2(n))$ *and* $T_3(n) = O(\log(n))$.

Proof. Recall from Example 8 that $\mathcal{R} = \mathcal{I}'[1]$. The theorem now follows immediately from Theorems 11 and 12. □

An example showing the difference between a proper rectangle from $\mathcal{R}$ and a set from $\mathcal{I}'[2\lfloor \log n \rfloor]$ can be found in Fig. 3.
The proof of the next result bears some similarity to the proof of Theorem 11:

Theorem 14. *Let the function* $\ell_k(n)$ *be defined as in Lemma 6. Suppose that there is a flexible algorithm* A *for* $\mathrm{MinDis}(\mathcal{C}, \mathcal{H})$ *with time bounds* T_1, T_2, T_3. *Then the problem* $\mathrm{MinDis}(\mathcal{C}[k], \mathcal{H}[\ell_k(n)])$ *can be solved by a flexible algorithm with time bounds* $T'_1(n) = O(\log(n)T_1(n) + k^2 n \log^2(n))$, $T'_2(n) = O(\log(n)T_2(n) + k^2 \log^3(n))$ *and* $T'_3(n) = O(\log(n) + T_3(n))$. *Moreover, if the data structure used by* A *can be merged efficiently, then the first two time bounds are even better, namely* $T'_1(n) = O(T_1(n) + k^2 n \log^2(n))$ *and* $T'_2(n) = O(T_2(n) + k^2 \log^3(n))$.

Proof. We reuse the notation from the proof of Theorem 11. For instance, S' as given by (9) denotes a lexicographically sorted weighted sample and $\mathrm{DS}(S')$ denotes the corresponding data structure (which we still have to describe). As in the proof of Theorem 11, the tree B (with exactly one leaf for each distinct

z-coordinate in S') is part of $\mathrm{DS}(S')$. We augment B by storing for every node $u \in B$ the following piece of information:

$$K(u) = [l(u), r(u), \mathrm{DS}(u), (d_\ell(u))_{\ell=1,\dots,\ell_k(n)}, \ell_0(u), \ell_1(u)] \ .$$

The parameters $l(u), r(u), \mathrm{DS}(u)$ and $d_1(u)$ have precisely the same meaning as in the proof of Theorem 11. The remaining parameters have the following meaning:

1. Think of $d_\ell(u)$ as the smallest empirical error on $S'(u)$ that can be achieved by a hypothesis which is induced by a maximal antichain of size ℓ in $B(u)$. More formally, let $H(u) = H(S(u))$ be the hypothesis from $\mathcal{H}$ which is represented by the data structure $\mathrm{DS}(u)$. Then, for every set V of nodes in B, we define
$$H_V = \bigcup_{v \in V} I_Z(v) \times H(v) \ . \tag{12}$$
The parameter $d_1(u)$ is given by $d_1(u) = E_{S(u)}(H(u))$ as in the proof of Theorem 11. The inequality (10) is valid again. For every $\ell = 2, \dots, \ell_k(n)$, we set
$$d_\ell(u) = \min\{E_{S'(u)}(H_V) :\ V \text{ is a maximal antichain of size } \ell \text{ in } B(u)\} \ .$$
We denote by $V(u)$ be the maximal antichain in $B(u)$ that minimizes $E_{S'(u)}(H_V)$.
2. Recall our convention that the minimum of the empty set equals ∞. Since the size of the largest antichain in $B(u)$ equals $r(u) - l(u) + 1$ (which is the number of leaves in $B(u)$), it follows that $d_\ell(u) = \infty$ for all $\ell > r(u) - l(u) + 1$. Specifically, if u is a leaf, then $d_\ell(u) = \infty$ for all $\ell \geq 2$.
3. Let u be an inner node in B with left child u_0 and right child u_1. A maximal antichain V in $B(u)$ either consists of u only or decomposes into a maximal antichain V_0 of size ℓ_0 in $B(u_0)$ and a maximal antichain V_1 of size ℓ_1 in $B(u_1)$ so that V is of size $\ell = \ell_0 + \ell_1$. Hence the following recursive formula is valid for $\ell = 2, \dots, \ell_k(n)$:
$$d_\ell(u) = \min\{d_{\ell_0}(u_0) + d_{\ell_1}(u_1) :\ (\ell_0, \ell_1 \geq 1) \wedge (\ell_0 + \ell_1 = \ell)\} \ . \tag{13}$$
4. If $d_1(u) = \min\{d_\ell(u) : \ell = 1, \dots, \ell_k(u)\}$, then we set $\ell_0(u) = \ell_1(u) = 0$. This definition applies, for instance, to all leaves in $B(u)$. Suppose now that $d_1(u) > \min\{d_\ell(u) : \ell = 1, \dots, \ell_k(n)\}$, which implies that u is an inner node, say a node with left child u_0 and right child u_1. In this case, we set $\ell_0(u)$ (resp $\ell_1(u)$) equal to the number of nodes of $V(u)$ which belong to $B(u_0)$ (resp. to $B(u_1)$).

It is easy to see that building this data structure takes time $O(\log(n)T_1(n) + k^2 n \log^2(n))$. The term $\log(n)T_1(n)$ accounts for the time needed to compute $l(u), r(u), \mathrm{DS}(u), d_1(u)$ for every node u in B (which can be done in the same way as described in the proof of Theorem 11). The term $k^2 n \log^2(n)$ accounts

for the time needed to compute $d_\ell(u)$ recursively according to (13) which has to be done for every node $u \in B$ and for $\ell = 2, \ldots, \ell_k(n)$. The fact that the first term in the time bound, $\log(n)T_1(n)$, can be replaced by $T_1(n)$ if the data structure used by A can be merged efficiently follows in the same way as in the proof of Theorem 11.
Suppose that for one item in S' the weight parameter is modified. Let v be the leaf that stores the z-coordinate of this item. Then $K(u)$ must be changed only for the nodes u located on the path from v to the root r_B of B. It is easy to see that this takes time $O(\log(n)T_2(n) + k^2 \log^3(n))$ (resp. $O(T_2(n) + k^2 \log^3(n))$ if the data structure used by A can be merged efficiently). The first term in this bound accounts for the time needed to update $l(u), r(u), \mathrm{DS}(u), d_1(u)$. The second term accounts for the time needed to update $d_\ell(u)$ for $\ell = 2, \ldots, \ell_k(n)$ according to (13). This takes time $O((\ell_k(n)^2) = O(k^2 \log^2(n))$ for each node on the path from v to r_B and hence it takes a total time of order $O(k^2 \log^3(n))$.
Recall that $V(r_B)$ denotes the maximal antichain V in B that minimizes $E_{S'}(H_V)$. The hypothesis $H_{V(r_B)}$ does not necessarily belong to $\mathcal{H}[\ell_k(n)]$ because—compare with (12)—the intervals $I_Z(v) = [z'_{l(v)} : z'_{r(v)}]_Z$ with v ranging over the nodes in $V(r_B)$ form a partition of Z but not yet a partition of $\mathbb{R}$. However, we obtain a hypothesis from $\mathcal{H}[\ell_k(n)]$ when we replace the partition $(I_Z(v))_{v \in V(r_B)}$ by its canonical extension. The latter will be denoted by $(I_Z^E(v))_{v \in V(r_B)}$. With these notations, the hypothesis $H_{V(r_B)}$ and its extension $H^E_{V(r_B)}$ are given as follows:

$$H_{V(r_B)} = \bigcup_{v \in V(r_B)} I_Z(v) \times H(v) \text{ and } H^E_{V(r_B)} = \bigcup_{v \in V(r_B)} I^E_Z(v) \times H(v) \ .$$

For the sake of brevity, we set $H^* = H^E_{V(r_B)}$.
We claim that it can be tested in time $O(\log(n) + T_3(n))$ whether a given point $(z, x) \in \mathbb{R} \times \mathcal{X}$ belongs to H^*. We proceed as follows:

Step 1: Find the node v in $V(r_B)$ such that $z \in I^E_Z(v)$.
The search for this v is started at r_B and proceeds top-down. Assume that the search has reached a node u. Note that $u = v$ if $\ell_0(v) = \ell_1(v) = 0$ (which indicates that the best antichain at u is the singleton $\{u\}$). Suppose that we have not yet reached v. Then u must be an inner node. The search proceeds to the left child if $z \le z_{r(u)}$, and it proceeds to the right child of u otherwise. It follows that Step 1 takes time $O(\log(n))$.

Step 2: Check whether $x \in H(v)$.
Here, we make use of the fact that $(z, x) \in H^*$ iff $x \in H(v)$. Checking whether $x \in H(v)$ takes time $T_3(n)$.

Thus time $O(\log(n) + T_3(n))$ is sufficient for the execution of both steps, as desired.
Let $d_{min} = \min\{d_\ell(r_B) : \ell = 1, 2, \ldots, \ell_k(n)\}$. Note that $d_{min} = d_1(r_B)$ iff $\ell_0(r_B) = \ell_1(r_B) = 0$ (as exploited earlier in the proof already). Otherwise, $d_{min} = d_\ell$ for $\ell = \ell_0(r_B) + \ell_1(r_B)$. It follows that d_{min} is retrieved from $\mathrm{DS}(S')$ in constant time. Making use of (10), it easily follows (by a reasoning

similar to the corresponding reasoning in the proof of Theorem 11) that $d_{min} \leq \min_{C \in \mathcal{C}'[k]} E_{S'}(C)$. This concludes the proof. $\square$

Recall that $\mathcal{I}_k$ denotes the class of unions of at most k bounded intervals. As mentioned in Example 7, $\mathcal{I}_k$ is a subclass of $\mathcal{C}_2[2k+1]$. A flexible algorithm that successfully competes with the best concept from $\mathcal{I}_k$ is obtained when we apply the transformation from Theorem 14 to the (trivial) flexible algorithm for $\mathrm{MinDis}(\mathcal{C}_2, \mathcal{C}_2)$. The resulting time bounds are $T_1(n) = O(k^2 n \log^2(n))$, $T_2(n) = O(k^2 \log^3(n))$ and $T_3(n) = O(k + \log(n))$. However, the algorithm resulting from this general transformation is inferior to the algorithm from [8] (which is specialized to the class $\mathcal{I}_k$):[5]

Theorem 15 ([8]). *The problem* $\mathrm{MinDis}(\mathcal{I}_k, \mathcal{I}_k)$ *can be solved by a flexible algorithm with* $T_1(n,k) = O(k^2 n)$, $T_2(n,k) = O(k^2 \log(n))$ *and* $T_3(n,k) = O(k)$.

As a final application, we consider unions of axis-parallel rectangles:

Theorem 16. *Let the function* $\ell_k(n)$ *be defined as in Lemma 6. Then the problem* $\mathrm{MinDis}(\mathcal{R}_k, \mathcal{I}_k[\ell_{2k+1}(n)])$ *can be solved by a flexible algorithm with* $T_1(n,k) = O(k^2 n \log^2(n))$, $T_2(n,k) = O(k^2 \log^3(n))$ *and* $T_3(n,k) = O(k + \log(n))$.

PROOF. Recall from Example 9 that $\mathcal{R}_k$ is a subclass of $\mathcal{I}_k[2k+1]$. Combining Theorems 15 and 14, we may conclude that the problem $\mathrm{MinDis}(\mathcal{I}_k[2k+1], \mathcal{I}_k[\ell_{2k+1}(n)])$ can be solved by a flexible algorithm with time bounds as given in the assertion of the theorem. $\square$

The transformations described in Theorems 11 and 14 preserve flexibility but destroy properness. As for the transformation described in the following theorem, we have the reverse situation:

Theorem 17. *A flexible algorithm A for* $\mathrm{MinDis}(\mathcal{C}, \mathcal{H})$ *with time bounds* T_1, T_2, T_3 *can be transformed into an algorithm A' that solves the problem* $\mathrm{MinDis}(\mathcal{C}[2], \mathcal{H}[2])$ *in time* $O(n \log(n) + T_1(n) + nT_2(n))$.

PROOF. Let $S' = [(x'_1, w_1), \ldots, (x'_n, w_n)] = [(z_1, x_1, w_1), \ldots, (z_n, x_n, w_n)] \in (\mathbb{R} \times \mathcal{X} \times \mathbb{R})^n$ be a given instance of $\mathrm{MinDis}(\mathcal{C}[2], \mathcal{H}[2])$. Let n' be the number of distinct z-coordinates in S', and let $z'_1 < z'_2 < \ldots < z'_{n'}$ be the corresponding sorted sequence. For sake of convenience, let $z'_{n'+1} = z'_{n'} + 1$. For $k = 1, \ldots, n'+1$, let $S'_1(k) = \{(x'_i, w_i) : z_i < z'_k\}$ and $S'_2(k) = \{(x'_i, w_i) : z_i \geq z'_k\}$. Similarly, let $S_1(k) = \{(x_i, w_i) : z_i < z'_k\}$ and $S_2(k) = \{(x_i, w_i) : z_i \geq z'_k\}$.

Without loss of generality let $C^* = ((-\infty, z'_{k'}) \times C_1) \cup ([z'_{k'}, +\infty) \times C_2) \in \mathcal{C}[2]$ be the concept with the smallest empirical error on S' among all concepts

[5]In [8], flexibility of algorithms is not an issue. An inspection of the algorithm for $\mathrm{MinDis}(\mathcal{I}_k, \mathcal{I}_k)$ reveals, however, that the underlying data structure provides flexibility.

from $\mathcal{C}[2]$. For each $k \in \{1, \ldots, n'+1\}$, let H_1^k (resp. H_2^k) be the hypothesis represented by $\mathrm{DS}(S_1(k))$ (resp. by $\mathrm{DS}(S_2(k))$. Let

$$H^k = ((-\infty, z'_k) \times H_1^k) \cup ([z'_k, \infty) \times H_2^k) \ .$$

Furthermore, let k'' be a minimizer of

$$W(k) := E_{S_1(k)}(H_1^k) + E_{S_2(k)}(H_2^k) \tag{14}$$

and let $H^* = H^{k''}$. With these notations, we get

$$\begin{aligned} E_{S'}(H^*) &= E_{S_1(k'')}(H_1^{k''}) + E_{S_2(k'')}(H_2^{k''}) \le E_{S_1(k')}(H_1^{k'}) + E_{S_2(k')}(H_2^{k'}) \\ &\le E_{S_1(k')}(C_1) + E_{S_2(k')}(C_2) = E_{S'}(C^*) \ . \end{aligned}$$

Thus the empirical error of $H^* \in \mathcal{H}[2]$ on S' is not larger than the empirical error of $C^* \in \mathcal{C}[2]$ on S'. Suppose that we know the values $W(k)$ for all $k = 1, \ldots, n'+1$. Then we can determine (a representation of) H^* as follows:

1. Set $k'' := \mathrm{argmin}\{W(k) : k \in \{1, \ldots, n'+1\}\}$. This takes $O(n)$ steps.
2. Extract $S_1(k'')$ and $S_2(k'')$ from S' and sort each of these two sequences according to the x-coordinates of its items. This takes $O(n\log(n))$ steps.
3. Feed $S_1(k'')$ (resp. $S_2(k'')$) into A and obtain the data structure $\mathrm{DS}(S_1(k''))$ (resp. $\mathrm{DS}(S_2(k''))$). This takes $O(T_1(n))$ steps.
4. Recall that $\mathrm{DS}(S_i(k''))$ represents $H_i^{k''}$ for $i = 1, 2$. These data structures augmented by $z'_{k''}$ form a suitable and easy-to-evaluate representation of H^*.

It remains to answer the question how the values $W(k)$ for $k = 1, \ldots, n'+1$ can be computed efficiently. We observe first that the operation of deleting an item (x_k, w_k) from a set S of (at most n) items can be simulated by setting $w_k = 0$. According to (14), $W(k)$ is easy to compute from $E_{S_1(k)}(H_1^k)$ and $E_{S_2(k)}(H_2^k)$. For reasons of symmetry, it suffices to describe how the values $E_{S_1(k)}(H_1^k)$ for $k = 1, \ldots, n'+1$ can be computed efficiently. This is done (similarly to a procedure used in [7] for learning 2-level decision trees) as follows:

1. Given S', let $k := n'+1$, $S := S_1(k)$ and sort this sequence according to non-decreasing x-coordinates. This takes $O(n\log(n))$ steps.
2. Feed S into A and obtain $\mathrm{DS}(S)$. This takes $O(T_1(n))$ steps.
3. Given $\mathrm{DS}(S)$, compute $E_S(H_1^k)$ and store it in $W(k)$. This takes $O(1)$ steps.
4. If $k = 1$, then stop. Otherwise, set $w_k := 0$, update the data structure $\mathrm{DS}(S)$ accordingly, set $k := k-1$ and go back to Step 3. This takes $T_2(n)$ steps.

The time complexity of the whole procedure for computing H^* is dominated by the amount of time needed for computing the quantities $W(k)$ for $k = 1, \ldots, n'+1$, and this takes $O(n\log(n) + T_1(n) + nT_2(n))$ steps. □

5. The Advantages of Inproper Agnostic Learning

Recall from Corollary 4 that an algorithm A that solves $\mathrm{MinDis}(\mathcal{C},\mathcal{H})$ can be transformed into an algorithm that agnostically learns $\mathcal{C}$ by $\mathcal{H}$. In the previous section, we have seen some smart algorithms for Minimum Disagreement whose design is based on choosing a hypothesis class $\mathcal{H}$ that is more powerful than the concept class $\mathcal{C}$. When these algorithms are used for the purpose of learning, they are *inproper* agnostic learning algorithms. The advantages of using these algorithms are as follows:

- The algorithms for $\mathrm{MinDis}(\mathcal{C},\mathcal{H})$ are sometimes much more efficient than the best up-to-date known algorithms for $\mathrm{MinDis}(\mathcal{C},\mathcal{C})$.
- The algorithm that solves $\mathrm{MinDis}(\mathcal{C},\mathcal{H})$ is guaranteed to return a hypothesis H whose empirical error on the input sequence S is provably not greater (and sometimes considerably smaller!) than the smallest possible empirical error that can be achieved by some hypothesis from $\mathcal{C}$.

A close inspection of Corollary 4 reveals, however, that this comes at a price: as the required sample size is proportional to the VC-dimension of the more powerful class $\mathcal{H}$, we need to feed a comparably large sample into the algorithm that solves $\mathrm{MinDis}(\mathcal{C},\mathcal{H})$. The hope is that the higher efficiency and the (potentially) smaller empirical risk overcompensate the larger sample size that is actually required. In this section, we give theoretical support for this hope. In Section 5.1, we show that the price that we have to pay (in terms of an increased VC-dimension) is relatively low. In Section 5.2, we present an algorithm for inproperly agnostically learning axis-parallel rectangles whose time bound for achieving accuracy ε in the model of agnostic learning is significantly smaller than the time bounds achieved by the up-to-date best (proper) algorithms.

5.1. VC-Dimension Considerations

The first result in this section clarifies how much the VC-dimension can grow when we expand a concept class $\mathcal{C}$ to $\mathcal{C}[k]$ or to $\mathcal{C}'[k]$:

Theorem 18. *Let $\mathcal{C}$ be a concept class of VC-dimension d over some ground set $\mathcal{X}$ such that $\emptyset \in \mathcal{C}$. Then, for all $k \geq 1$, we have that:*

1. $VCD(\mathcal{C}[k]) \leq VCD(\mathcal{C}'[k]) \leq VCD(\mathcal{C}[k]) + 2$.
2. $kd \leq VCD(\mathcal{C}[k]) \leq k(d+1) - 1$.
3. $kd \leq VCD(\mathcal{C}'[k]) \leq k(d+1) + 1$.

Proof. It suffices to prove the following inequalities:

$$kd \leq \mathrm{VCD}(C[k]) \leq \mathrm{VCD}(C'[k]) \leq \mathrm{VCD}(C[k]) + 2 \ , \ \mathrm{VCD}(C[k]) \leq (d+1)k - 1 \ .$$

The inequality $kd \leq \mathrm{VCD}(C[k])$ is obvious from the following implication:

$$x_1, \ldots, x_d \text{ is shattered by } C \Rightarrow \{1, \ldots, k\} \times \{x_1, \ldots, x_d\} \text{ is shattered by } C[k] \ .$$

Next observe that any finite sequence $(z_1, x_1), \dots, (z_d, x_d)$ that is shattered by $C[k]$ can be shattered by $C'[k]$ too (because there is a bounded interval that contains $z_1, \dots, z_d$). This shows that $\mathrm{VCD}(C[k]) \le \mathrm{VCD}(C'[k])$.
We move on and verify the inequality $\mathrm{VCD}(C'[k] \le \mathrm{VCD}(C[k])+2$. To this end, let $S = [(z_1, x_1), \dots, (z_s, x_s)]$ be a sequence of instances from $\mathbb{R} \times \mathcal{X}$ ordered according to non-decreasing z-coordinates. Suppose that S is shattered by $\mathcal{C}'[k]$. Then each label combination $(b_1, \dots, b_s) \in \{0,1\}^s$ can be realized by some concept in $\mathcal{C}'[k]$. Let $\cup_{j=1}^{k'}(I'_j \times C_j) \in \mathcal{C}'[k]$ be a concept realizing the bit pattern $(1, b_2, \dots, b_{s-1}, 1)$ on S. Then the same bit pattern can be realized by $\cup_{j=1}^{k'}(I_j \times C_j) \in \mathcal{C}[k]$ where I_1 is the interval from $-\infty$ to the right endpoint of I'_1, I_s is the interval from the left endpoint of I'_s to ∞, and $I_j = I'_j$ for $j \notin \{1, s\}$. It follows that $\mathrm{VCD}(\mathcal{C}'[k]) \le \mathrm{VCD}(\mathcal{C}[k]) + 2$, as desired.
As for the only remaining inequality, $\mathrm{VCD}(C[k]) \le (d+1)k - 1$, we have to show that a sequence $S = [(z_1, x_1), \dots, (z_s, x_s)]$ of length $s = (d+1)k$ cannot be shattered by $\mathcal{C}[k]$. This can be seen as follows. Split S into k segments of length $d+1$. For each segment, choose a label combination that cannot be realized by any concept from $\mathcal{C}$. It is then easy to see that the resulting label combination for the full sequence S cannot be realized by any concept from $\mathcal{C}[k]$.[6] From this discussion, it follows that $\mathrm{VCD}(\mathcal{C}[k]) \le (d+1)k - 1$, as desired. □

The following two lemmas show that the bounds given in Theorem 18 are tight.

Lemma 19. *Let $\mathcal{X}_d = \{x_1, \dots, x_d\}$ be a ground set of size d. Let $\mathcal{C}_d$ be the powerset of $\mathcal{X}_d$ (so that $VCD(\mathcal{C}_d) = d$). Then $VCD(\mathcal{C}_d[k]) = kd$. Moreover, if k is even, then $VCD(\mathcal{C}'_d[k]) = kd$.*

Proof. We know from Theorem 18 that $\mathrm{VCD}(\mathcal{C}'_d[k]) \ge \mathrm{VCD}(\mathcal{C}_d[k]) \ge kd$. It suffices therefore to show that kd upper-bounds the VC-dimension of these classes. Let $s = kd + 1$ and let $S = [(z_1, x'_1), \dots, (z_s, x'_s)] \in (\mathbb{R} \times \mathcal{X}_d)^s$ where $z_1 \le \ldots \le z_s$. It suffices to show that the members of the sequence S are not shattered by $\mathcal{C}_d[k]$ and, if k is even, not even shattered by $\mathcal{C}'_d[k]$. According to the pigeonhole principle, there must exist an element $x \in \mathcal{X}_d$ that occurs at least $k+1$ times within the sequence $x'_1, \dots, x'_s$, say $x = x'_{i(0)} = \ldots = x'_{i(k)}$ for $1 \le i(0) < \ldots < i(k) \le s$. For $\ell = 0, \dots, k$, assign label 1 to $(z_{i(\ell)}, x'_{i(\ell)})$ if ℓ is even, and assign label 0 to $(z_{i(\ell)}, x'_{i(\ell)})$ otherwise. A concept of the form $\cup_{j=1}^{k'}(I_j \times C_j) \in \mathcal{C}_d[k]$ cannot be consistent with this labeling because there must exist two subsequent occurrences of x, say the occurrences at coordinates $z'_{i(\ell)}$ and $z'_{i(\ell+1)}$, that fall into the same interval of the ordered partition $I_1, \dots, I_{k'}$, say they fall into $I_{j'}$. But, since the examples $(z_{i(\ell)}, x'_{i(\ell)})$ and $(z_{i(\ell+1)}, x'_{i(\ell)+1})$ have opposite labels, the concept $C_{j'}$ cannot label both of them correctly. This shows that $\mathrm{VCD}(\mathcal{C}_d[k]) \le kd$. It is easy to see that basically the same reasoning

[6]The same argument was used in [1] in connection with a class of piecewise defined functions over the ground set $\mathbb{R}$.

applies to the class $\mathcal{C}'_d[k]$ in place of $\mathcal{C}_d[k]$ provided that both of $(z_{i(0)}, x'_{i(0)})$ and $(z_{i(k)}, x'_{i(k)})$ are labeled 1. This is precisely the case when k is even, which concludes the proof. □

Lemma 20. *For every $a \in \mathbb{R}^d$, let $C_a = \{x \in \mathbb{R}^d : x_i \le a_i \text{ for } i = 1, \ldots, d\}$. Let $\mathcal{B}_d = \{C_a : a \in \mathbb{R}^d\}$ (so that VCD($\mathcal{B}_d$) $= d$). Then VCD($\mathcal{B}_d[k]$) $= k(d+1)-1$ and VCD($\mathcal{B}'_d[k]$) $= k(d+1)+1$.*

PROOF. Within this proof, the all-ones vector is denoted by $\vec{e}$ and the vector with 1 in component i and 0 in the remaining components is denoted by $\vec{e_i}$.
We know from Theorem 18 that $\mathrm{VCD}(\mathcal{B}_d[k]) \le k(d+1)-1$, $\mathrm{VCD}(\mathcal{B}'_d[k]) \le k(d+1)+1$ and that $\mathrm{VCD}(\mathcal{B}_d[k]) \ge \mathrm{VCD}(\mathcal{B}'_d[k] - 2$. It suffices therefore to show that $\mathrm{VCD}(\mathcal{B}'_d[k]) \ge k(d+1)+1$. To this end, consider the following sequence S of length $s = k(d+1)+1$ consisting of k segments of length $d+1$ and an extra-example:

- For $\ell = 1, \ldots, k$, the ℓ-th segment S_ℓ of S is given by

$$((\ell-1)(d+1)+1, (\ell-1/2)\vec{e})\ ,\ ((\ell-1)(d+1)+2, \ell\vec{e_1})\ ,\ \ldots,\ (\ell(d+1), \ell\vec{e_d})\ .$$

- The extra-example is $(k(d+1)+1, (k-1/2)\vec{e})$.

We call $(\ell-1/2)\vec{e}$ the *sum vector* in S_ℓ. The vectors $\ell\vec{e_1}, \ldots, \ell\vec{e_d}$ are called the *spanning vectors* in S_ℓ. We refer to the numbers $(\ell-1)(d+1)+1, (\ell-1)(d+1)+2, \ldots, \ell(d+1)$ as the *z-coordinates* of these vectors. Likewise $k(d+1)+1$ is referred to as the z-coordinate of the extra-example. It is easy to check that the sequence S has the following properties:

1. For every $\ell \in \{1, \ldots, k\}$ and every $b \in \{0,1\}^d$, the label combination $(1, b_1, \ldots, b_d)$ for the vectors $[(\ell-1/2)\vec{e}, \ell\vec{e_1}, \ldots, \ell\vec{e_d}]$ is realizable by a suitable concept[7] in $\mathcal{B}_d$.
2. For every $\ell \in \{2, \ldots, k\}$ and every $b \in \{0,1\}^d$, the label combination $(b_1, \ldots, b_d, 0)$ for the vectors $[(\ell-1)\vec{e_1}, \ldots, (\ell-1)\vec{e_d}, (\ell-1/2)\vec{e}]$ is realizable by some concept in $\mathcal{B}_d$.
3. For every $b \in \{0,1\}^d$, the label combination $(b_1, \ldots, b_d, 1)$ for the vectors $[k\vec{e_1}, \ldots, k\vec{e_d}, (k-1/2)\vec{e}]$ is realizable by some concept in $\mathcal{B}_d$.

It suffices to show that the members of the sequence S are shattered by $\mathcal{B}'_d[k]$. Consider an arbitrary label combination for the members of S. We choose the intervals $(I_1, \ldots, I_k) \in \mathcal{P}'(k)$ according to the following rules:

1. For every $\ell \in \{1, \ldots, k\}$, the z-coordinates of the spanning vectors in S_ℓ fall into I_ℓ. If the sum vector in S_ℓ has label 1, then its z-coordinate falls into I_ℓ too.

[7]In this and in the following properties, the smallest concept from $\mathcal{B}_d$ that covers all vectors with label 1 is suitable.

2. For every $\ell \in \{2, \ldots, k\}$, the z-coordinate of the sum vector in S_ℓ falls into $I_{\ell-1}$ if it has label 0.
3. I_k includes (resp. excludes) the z-coordinate of the extra-example if the latter has label 1 (resp. label 0).
4. I_1 includes (resp. excludes) the z-coordinate of the sum vector of S_1 if the latter has label 1 (resp. label 0).

It follows from the aforestated properties of S that the intervals $I_1, \ldots, I_k$ can be provided with concepts $C_1, \ldots, C_k \in \mathcal{B}_d$ such that $\cup_{\ell=1}^{k}(I_\ell \times C_\ell)$ is consistent with the given labeling of the members of S. □

5.2. Agnostic Learning of Bounded Axis-parallel Rectangles

Recall that $\mathcal{R}$ denotes the class of axis-parallel rectangles and $\mathcal{I}$ denotes the class of bounded real intervals. The class $\mathcal{I}$ has VC-dimension 2. According to Theorem 18, the VC-dimension of $\mathcal{I}'[k]$ is at most $3k+1$. Let now $\ell(n) = 2\lfloor \log n \rfloor$ and $d(n) = 3\ell(n) + 1$. According to Theorem 13, there is a flexible algorithm that solves $\mathrm{MinDis}(\mathcal{R}, \mathcal{I}'[\ell(n)])$ with time bounds $T_1(n) = O(n \log(n))$, $T_2(n) = O(\log^2(n))$ and $T_3(n) = O(\log n)$. Recall that T_1 is the time required to build up the data structure $\mathrm{DS}(S)$ for a given sequence S, and $\mathrm{DS}(S)$ implicitly represents the hypothesis from $\mathcal{I}'[\ell(n)]$. Moreover $T_3(n)$ is the time required to evaluate this hypothesis on a new data point. According to Corollary 5, the algorithm solving $\mathrm{MinDis}(\mathcal{R}, \mathcal{I}'[\ell(n)])$ can be transformed into a learning algorithm L that agnostically learns $\mathcal{R}$ by $\mathcal{I}'[\ell(n)]$ provided that the sample size n satisfies

$$n \geq c \cdot \frac{1}{\varepsilon^2} \log\left(\frac{1}{\varepsilon\delta}\right)$$

for a sufficiently large constant $c > 0$. The time needed by L for achieving accuracy ε with probability at least $1 - \delta$, is therefore bounded by

$$O\left(T_1\left(\frac{1}{\varepsilon^2} \log\left(\frac{1}{\varepsilon\delta}\right)\right)\right) = \tilde{O}\left(\frac{1}{\varepsilon^2}\right) \quad .$$

This favorably compares to the time $\tilde{O}(1/\varepsilon^4)$ needed by the up-to-date fastest algorithm for properly learning $\mathcal{R}$ in the agnostic model.

6. Experimental Results

We conducted several experiments to compare the performance of our algorithm with established methods from the literature on both synthetic and real-word data sets. We chose to investigate Minimum Disagreement algorithms for $\mathcal{R}$, the class of axis-aligned rectangles in two dimensions, because we expect to observe a considerable improvement in the running time in light of the much lower asymptotic worst-case time bound.

The goal of this section is to demonstrate that our algorithm

1. is able to solve real world machine learning problems with a level of quality comparable to established methods,

2. is fast enough to be feasible even on large data sets,
3. is especially significantly faster than the up-to-date best proper algorithms solving the Minimum Disagreement Problem for rectangles. In particular, we would like to confirm our theoretical results from Section 5.2.

This section is organized as follows. We present the investigated algorithms and data sets in Section 6.1 resp. Section 6.2. We report and discuss the results of the experiments in Section 6.3.

6.1. Algorithms

We implemented the following three algorithms as single-threaded Java programs:

- The algorithm solving the Minimum Disagreement Rectangle problem from [8], which we denote by `RECT`, with an asymptotic worst-case time bound of $O(n^2 \log(n))$, where n denotes the sample size.
- The algorithm solving the Minimum Disagreement Rectangle problem from [10], which we denote by `RECT2`, with an asymptotic worst-case time bound of $O(n^2)$.
- Our method as described by Theorem 13, which will be denoted by `TRANS` in this section. Its asymptotic worst-case time bound is $O(n \log(n))$.

6.2. Data sets

We are solving binary classification problems where the weight of any instance is either -1 or $+1$. We use three different data sets:

- One artificially generated data set, which is given by a mixture of two two-dimensional Gaussian distributions—one distribution for each weight—with identical covariance matrices.
- The "Glass" data set from [16], which consists of nine-dimensional instances from forensic examinations of glass samples. While the original data set contains seven classes and 214 instances, we obtained a binary classification problem with 163 instances by following the procedure from [6] (which consists of merging classes one and three and removing all instances from class four to seven).
- The "MAGIC" data set from [17, 18], which is composed of 19020 ten-dimensional instances of (simulated) observations of a "Cherenkov gamma telescope". The task is to discern gamma ray events from background noise.

The latter two data sets are freely available on the UC Irvine Machine Learning Repository.

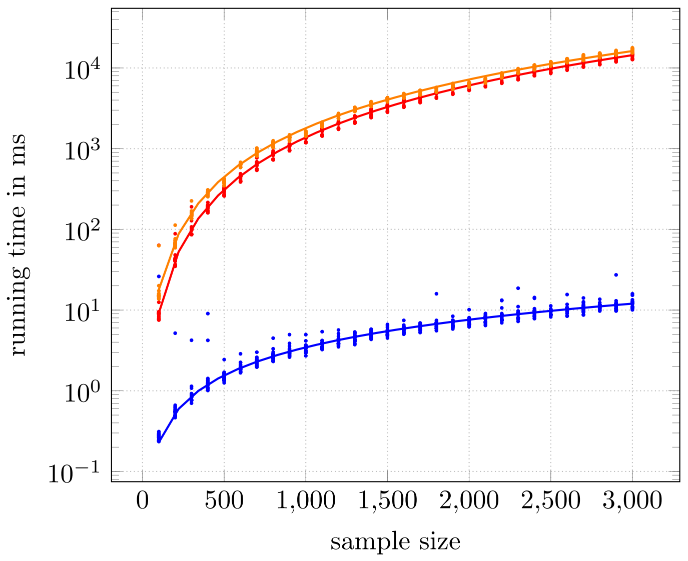


Figure 4: Scatter plot of the running time of `RECT` (in red), `RECT2` (in orange) and `TRANS` (in blue) versus the sample size on the artificial Gaussian distribution. The solid lines depict the asymptotic running times according to theoretical analysis. Leading constants, hidden by Landau's O, were determined empirically.

6.3. Experimental results and discussion

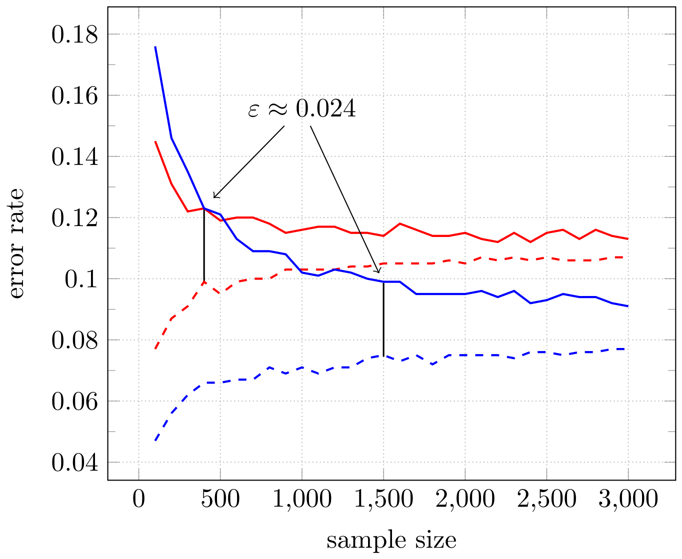
Note that—as in the theoretical analysis—we measured all running times without taking the time for pre-sorting the training data into account. This is justified as all considered algorithms rely on pre-sorted data.

All experiments were performed on a AMD Opteron 6234 processor, running at 2400 MHz, with Oracle Java 1.8.0_31 under CentOS 6.6.

Results on the Gaussian mixture. Fig. 4 depicts the measured running times of all three algorithms on the Gaussian mixture data set. We also added plots of the theoretically derived asymptotic worst-case time bounds, which nicely match the empirical results. As expected, `TRANS` is orders of magnitude faster than both `RECT` and `RECT2`.

Despite the fact that $O(n^2)$ is clearly better than $O(n^2 \log(n))$, observe that in the range of sample sizes used in the experiment the algorithm `RECT` is actually slightly faster than `RECT2` (this effect could also depend on details of our implementation). Since both algorithms solve the same problem—finding a rectangle with Minimal Disagreement to the sample set—we will ignore `RECT2` in the following and focus on comparing `TRANS` with `RECT`[8].

[8]There is another complication concerning `RECT2`, which makes its application rather cumbersome: every x- and y-coordinate may only contain one sample point. This property is violated by both the Glass and MAGIC data set. We chose not to implement possible solu-



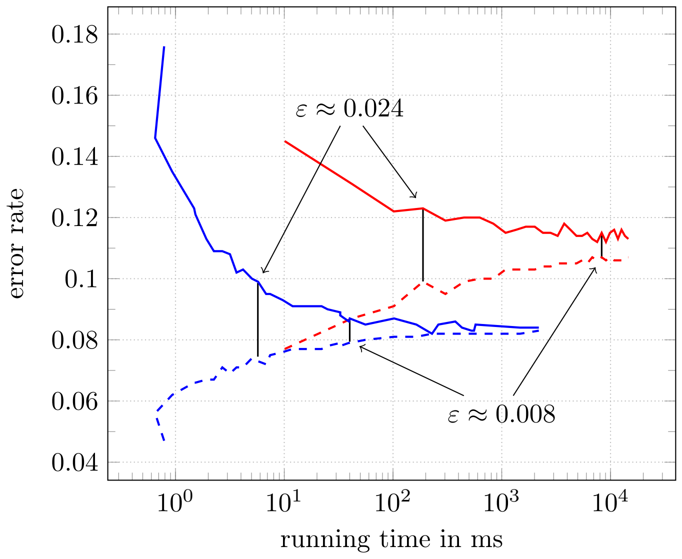


Figure 5: Error rates of `RECT` (in red) and `TRANS` (in blue) as a function of the sample size (upper figure) and of the running time (lower figure) on the artificial Gaussian distribution. The x-axis on the lower figure is logarithmic to accommodate the vast range of running times. The solid lines depict the error rates on the test set, while the dashed lines show the error rate on the training set. Note that the accuracy parameter ε in the model of agnostic learning is proportional to the difference between these two error rates. All values were measured on an independent test set of size 1000 and averaged over 50 runs.

Table 1: Experimental results for the best instance pair using TRANS and RECT on the Glass and MAGIC data sets. Data sets were randomly split 2:1 into a training and test set. All values are averages over 50 runs, except for RECT on the MAGIC data set, which was run only 10 times.

data set	algorithm	error rate on training set	error rate on test set	time
Glass	TRANS	0.081	0.233	14 ms
	RECT	0.160	0.252	393 ms
MAGIC	TRANS	0.178	0.189	6 s
	RECT	0.214	0.219	17256 s

Let us now examine empirical error rates: as a function of the sample size, shown in the upper part of Fig. 5, the error rates behave as expected. RECT, whose hypothesis class has the smaller VC-dimension, achieves a smaller error on the test set for small sample sizes and its error rates converge faster (as a function of the sample size). Note that TRANS' error rate outperforms RECT already at $n \approx 500$ because its higher estimation error is getting more than compensated by its lower approximation error.

When computation time is the resource of consideration, as depicted in the lower part of Fig. 5, the contrast between the two algorithms becomes more drastic: TRANS consistently outperforms the much slower RECT-algorithm. Furthermore, as predicted by our discussion in Section 5.2, TRANS' error rates indeed converge much faster.

Results on the Glass and MAGIC data sets. The dimensions of the instances in both the Glass and MAGIC data sets are larger than two, so we cannot directly apply RECT and TRANS. We followed [6] and trained one hypothesis on every pair of coordinates[9], choosing the hypothesis with the smallest error on the training set.[10] The results are given in Table 1 and show that—for both the smaller Glass data set with 108 training instances and the larger MAGIC data set with 12680 training instances—TRANS' error rates are smaller than the ones from RECT. As expected, TRANS is considerably faster than RECT: notice the giant gap between six seconds and almost six hours on the MAGIC data set.

The mean error rate of 0.233 on the Glass data set is in fact smaller than the rates reported in [6], which were 0.271 for a simple one-dimensional hypothesis and 0.257 for a (more complex) decision tree.

Our mean error rate of 0.189 on the MAGIC data set is considerably larger than the ones reported in [18], which were in the range of 0.16 to 0.138 (on the

tions to this problem after observing the superior performance of RECT.

[9]We also tried all orders of coordinates (i.e., (x_i, x_j) and (x_j, x_i)) for TRANS, since the resulting hypotheses are not equivalent.

[10]Another approach would be to iteratively transform the MinDis algorithm until we arrive at the desired dimension. However, this method introduces too much overhead using our implementation for nine resp. ten dimensions.

Table 2: Experimental results for AdaBoost using `TRANS` as the base learner on the MAGIC data set with different numbers of iterations T. Data sets were randomly split 2:1 into a training and test set. All values are averages over 50 runs.

algorithm	T	error rate on training set	error rate on test set	time
AdaBoost on `TRANS`	1	0.178	0.189	6 s
	5	0.153	0.167	29 s
	10	0.138	0.156	59 s
	20	0.124	0.151	117 s
	40	0.107	0.149	235 s
	60	0.094	0.147	350 s

test set) for different variants of decision trees.

`TRANS` as a base learner for AdaBoost. We try to close the gap between our approach and the algorithms from [18] in the following by using the well-known AdaBoost [19] scheme with `TRANS` as the base learner. We would like to remind the reader that AdaBoost iteratively runs the base learner while adjusting the weights of the sample points in each iteration. The resulting hypothesis of AdaBoost is a weighted sum of the base learner's hypotheses from all iterations.

The results for AdaBoost on the MAGIC data set are given in Table 2. Obviously, one round of boosting is equivalent to the previous approach of choosing the hypothesis with the smallest empirical error. Note that already twenty iterations provide an error rate on the test set that is comparable with the rates from [18], and that `TRANS` is obviously fast enough for boosting to be practical on 12680 instances.

Furthermore, our boosted classifier surpasses all methods considered in [17] in five of the six measures of merit considered in that paper. To give an example, we will now determine two of these measures, called *loacc* and *quality factor* $Q_{0.5}$. Both are based on the ROC curve [20], which plots the true positive rate (i.e. the probability to give a positive label to a positive test point) versus the false positive rate (i.e. the probability to give a positive label to a negative test point) for varying threshold values used in the combined hypothesis of AdaBoost[11]. The remaining four measures of merit are defined in a similar way. The ROC curve of our classifier after $T = 60$ iterations of boosting is depicted in Fig. 6.

The first measure of merit, loacc, is determined by calculating the average of the true positive rates for false positive rates of 0.01, 0.02 and 0.05. The corresponding points are marked red in Fig. 6. Our boosted algorithm achieves a loacc value of 0.565, which is better than 0.472, the best value reported in [17].

For the second measure of merit, the quality factor $Q_{0.5}$, we have to find

[11]The user of AdaBoost may choose their favorite threshold after inspecting the ROC curve. Plain AdaBoost uses a threshold value of zero, which was also used to determine error rates in Table 2.

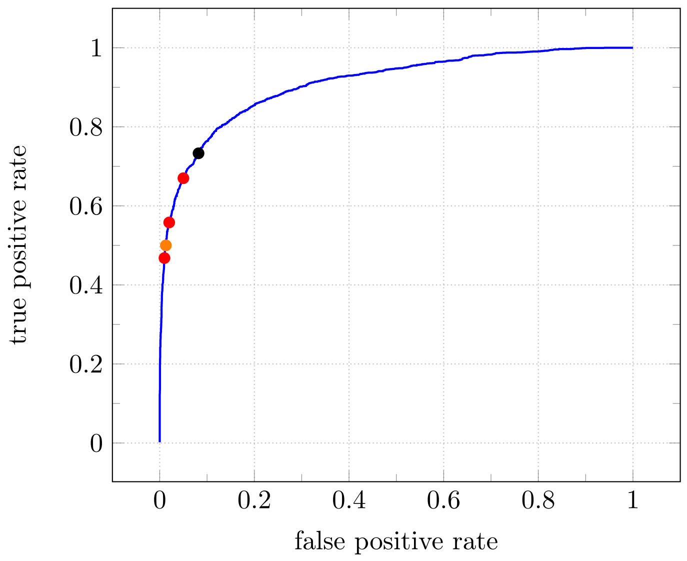


Figure 6: The ROC curve for one run of AdaBoost based on `TRANS` with $T = 60$ iterations on the MAGIC data set. The black point marks the performance of AdaBoost's own threshold value of zero, while the red and orange points are used to calculate the measures 'loacc' and '$Q_{0.5}$' from [17]. The data set was randomly split 2:1 into a training and test set.

the point with a true positive rate of 0.5 on the ROC curve and measure the corresponding false positive rate ε_F. The quality factor is now given by $Q_{0.5} = 0.5/\sqrt{\varepsilon_F}$. The relevant point on the ROC curve is marked orange in Fig. 6. Our boosted algorithm achieves a quality factor of 4.45, which is better than 3.5, the best value reported in [17].

Admittedly, we suspect that the good performance of our boosted algorithm is mostly due to boosting itself and independent from the choice of the base learner, as experiments with AdaBoost using decision stumps, i.e. simple one-dimensional threshold functions, yield similar error rates (while AdaBoost based on decision stumps needs roughly ten times the number of iterations to reach similar test error rates as boosted `RECT`, the over-all running time is almost 300 times smaller). Nevertheless, we have successfully demonstrated that `TRANS` is indeed fast enough to be used as a base learner for AdaBoost, while boosting `RECT` and `RECT2`—the best known proper algorithms solving the Minimum Disagreement Problem for rectangles—seems like an almost ridiculous idea due to their impracticable running times.

References

[1] M. J. Kearns, R. E. Schapire, L. M. Sellie, Toward efficient agnostic learning, Machine Learning 17 (2) (1994) 115–141.

[2] L. Pitt, L. G. Valiant, Computational limitations on learning from examples, Journal of the Association on Computing Machinery 35 (4) (1988) 965–984.

[3] S. M. Weiss, I. Kapouleas, An empirical comparison of pattern recognition, neural nets, and machine learning classification methods, in: IJCAI '89, 1989, pp. 781–787.

[4] S. M. Weiss, C. A. Kulikowski, Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Expert Systems, Morgan Kaufmann, 1990.

[5] S. M. Weiss, R. S. Galen, P. Tadepalli, Maximizing the predictive value of production rules, Artificial Intelligence 45 (1-2) (1990) 47–71.

[6] R. C. Holte, Very simple classification rules perform well on most commonly used datasets, Machine Learning 11 (1) (1993) 63–91.

[7] P. Auer, R. C. Holte, W. Maass, Theory and applications of agnostic PAC-learning with small decision trees, in: ICML '95, 1995, pp. 21–29.

[8] W. Maass, Efficient agnostic PAC-learning with simple hypothesis, in: COLT '94, 1994, pp. 67–75.

[9] C. Cortés, J. M. Díaz-Báñez, P. Pérez-Lantero, C. Seara, J. Urrutia, I. Ventura, Bichromatic separability with two boxes: A general approach, Journal of Algorithms 64 (2-3) (2009) 79–88.

[10] J. Barbay, T. M. Chan, G. Navarro, P. Pérez-Lantero, Maximum-weight planar boxes in $O(n^2)$ time (and better), Information Processing Letters 114 (8) (2014) 437–445.

[11] V. Vapnik, Statistical learning theory, Wiley & Sons, 1998.

[12] S. Shalev-Shwartz, S. Ben-David, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, 2014.

[13] M. d. Berg, O. Cheong, M. v. Kreveld, M. Overmars, Computational Geometry: Algorithms and Applications, Springer-Verlag, Santa Clara, CA, USA, 2008.

[14] V. N. Vapnik, A. Y. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities, Theory of Probability and its Applications XVI (2) (1971) 264–280.

[15] M. Talagrand, Sharper bounds for gaussian and empirical processes, The Annals of Probability 22 (1) (1994) 28–76.

[16] I. W. Evett, E. J. Spiehler, Rule induction in forensic science, in: P. H. Duffin (Ed.), Knowledge Based Systems, Halsted Press, 1988, pp. 152–160.

[17] R. Bock, A. Chilingarian, M. Gaug, F. Hakl, T. Hengstebeck, M. Jiřina, J. Klaschka, E. Kotrč, P. Savický, S. Towers, A. Vaiciulis, W. Wittek, Methods for multidimensional event classification: a case study using images from a cherenkov gamma-ray telescope, Nuclear Instruments and Methods in Physics Research A 516 (2–3) (2004) 511 – 528.

[18] J. Dvorák, P. Savický, Softening splits in decision trees using simulated annealing, in: ICANNGA '07, Part I, 2007, pp. 721–729.

[19] Y. Freund, R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1) (1997) 119 – 139.

[20] J. P. Egan, Signal detection theory and ROC analysis, Academic Press, 1975