Hierarchical Design of Fast Minimum Disagreement Algorithms

Malte Darnstädt, Christoph Ries and Hans Ulrich Simon

Fakultät für Mathematik, Ruhr-Universität Bochum, D-44780 Bochum
`{malte.darnstaedt,christoph.ries,hans.simon}@rub.de`

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 domain $\mathcal{X}$: 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

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

is minimized. In learning theory, $E_S(C)$ is called the *empirical error of* C *on* S, and this term plays a central role in statistical learning theory, especially in the model of agnostic learning [10].

Although the Minimum Disagreement problem is intractable for a wide variety of classes [12, 10], 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 [17, 18, 16], unions of intervals [9], or 2-level decision trees [1]) 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 [11, 1, 5, 2].

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 [11] solves the Minimum Disagreement problem for axis-parallel rectangles in time[1] $O(n^2 \log(n))$. It was not until recently [2] 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 [14, 13]. 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 $E_S(H) \le \min_{C \in \mathcal{C}} E_S(C)$. While this approach is well known in the context of Boolean classes [12] and standard in the theory of agnostic learning [10], 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 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 $\tilde{O}(1/\varepsilon^2)$

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

while the learning procedure based on the up to date fastest proper algorithm from [2] 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 builds on existing work [11, 5]; in particular, the employed data structures are a variant of Segment Trees[2] [3]. 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.

2 Definitions, Notations and Facts

Let $\mathcal{X}$ be a set. In the parlance of learning theory, any subset of $\mathcal{X}$ is called a *concept* over the domain $\mathcal{X}$ or, alternatively, a *hypothesis* over $\mathcal{X}$. A family of concepts (resp. hypotheses) over $\mathcal{X}$ is called a *concept class* (resp. *hypothesis class*) 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 $\mathcal{X}$. We will assume throughout the paper that the domain $\mathcal{X}$ is equipped with a linear ordering and that S is ordered so that $x_1 \leq \ldots \leq x_n$.

Intuitively, a concept C "performs well" on a weighted sample S if it includes the points of S with a positive weight and excludes the points in S with a negative weight. The empirical error of C on S, denoted $E_S(C)$ and already defined in Section 1, measures to which extent the concept C does not perform well.

Let $\mathcal{C}$ and $\mathcal{H}$ be two classes over the same domain $\mathcal{X}$. 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 sorted 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., $E_S(H) \leq \min_{C \in \mathcal{C}} E_S(C)$.

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 of the reals into k consecutive non-empty subintervals. 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

[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.

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"*). For $a, b \in \mathbb{R}$, we define $[a : b] := [a, b] \cap \mathbb{Z}$.

Let B 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 sub-tree 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 $[l(u) : r(u)]$ is called the *interval represented by* u. Every maximal antichain V of nodes in B represents a partition of $\{1, \ldots, n\}$ in the obvious manner. For instance $V = \{r_B\}$ represents the trivial partition with the single equivalence class $\{1, \ldots, n\}$. The set of leaves in B represents the partition of $\{1, \ldots, n\}$ into n singletons $\{1\}, \ldots, \{n\}$. The other maximal antichains induce partitions which are in between these two extremes. The following result is not hard to show:

Lemma 1. *1. For all $1 \leq a \leq b \leq n$, there exists an antichain V of size at most $2\lfloor \log n \rfloor$ in B such that $[a : b] = \cup_{v \in V}[l(v) : r(v)]$. Moreover, given B and a, b, the smallest antichain with this property can be found in time $O(\log(n))$.*
2. Let $k \geq 2$, $\ell_2(n) = \lceil \log n \rceil + 1$ and $\ell_k(n) = (k-1)\log(n)$ for $k \geq 3$. Then, for every partition $(I_1, \ldots, I_k) \in \mathcal{P}(k)$, there exists a maximal antichain V of size at most $\ell_k(n)$ in B such that the partition represented by V is a refinement of the partition induced by $(I_1, \ldots, I_k)$ on $\{1, \ldots, n\}$.

3 From Simple to More Complex Concept Classes

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

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

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 domain $\mathbb{R}$, $\mathcal{R}$ denotes the class of bounded axis-parallel rectangles over the domain $\mathbb{R}^2$, $\mathcal{I}_k$ 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 1. Let $\mathcal{X} = \{x\}$ and $\mathcal{C}_1 = \{\mathcal{X}\}$ and $\mathcal{C}_2 = \{\emptyset, \mathcal{X}\}$. We identify the domain $\mathbb{R} \times \{x\}$ 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 $\mathcal{C}_2[2k+1]$.

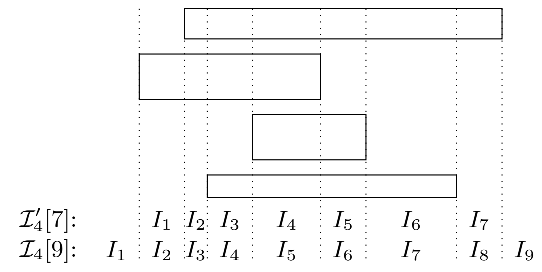
$\mathcal{I}_4'[7]$: I_1 I_2 I_3 I_4 I_5 I_6 I_7
$\mathcal{I}_4[9]$: I_1 I_2 I_3 I_4 I_5 I_6 I_7 I_8 I_9

Fig. 1. 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.

Example 2. 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 3. 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. 1 for an illustration.

The *VC-dimension* of class $\mathcal{H}$ over domain $\mathcal{X}$, denoted VCdim$(\mathcal{H})$, 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 VCdim$(\mathcal{H}) = \infty$. Let $d =$ VCdim$(\mathcal{H})$. It is well known that the number of random examples, required for achieving ε-accuracy in the model of agnostic learning, is of order $\tilde{O}(d/\varepsilon^2)$ [15], and a suitable hypothesis is found by running a Minimum Disagreement algorithm A with hypothesis class $\mathcal{H}$ on a random sample of this size. Thus, a time bound $T(n)$ for A reads as $\tilde{O}(T(d/\varepsilon^2))$ when written in terms of d and ε. We now analyze how much we have to pay in terms of the VC-dimension for moving to more complex concept classes.

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

- *VCdim$(\mathcal{C}[k]) \leq$ VCdim$(\mathcal{C}'[k]) \leq$ VCdim$(\mathcal{C}[k]) + 2$*
- *$dk \leq$ VCdim$(\mathcal{C}[k]) \leq (d+1)k - 1$*
- *$dk \leq$ VCdim$(\mathcal{C}'[k]) \leq (d+1)k + 1$*

(All these bounds can be shown to be tight.)

Proof. The inequalities $dk \leq$ VCdim$(\mathcal{C}[k]) \leq$ VCdim$(\mathcal{C}'[k])$ are rather obvious. Let $S = [(z_1, x_1), \ldots, (z_s, x_s)]$ be a sequence of instances from $\mathbb{R} \times \mathcal{X}$ ordered according to non-decreasing z-coordinates. Suppose that S is shattered by $\mathcal{C}'[k]$. Thus each label combination $(b_1, \ldots, b_s) \in \{0,1\}^s$ 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 $(1, b_2, \ldots, b_{s-1}, 1)$ 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 $\mathrm{VCdim}(\mathcal{C}'[k]) \leq \mathrm{VCdim}(\mathcal{C}[k]) + 2$. We still have to show that the above sequence S of length s cannot be shattered by $\mathcal{C}[k]$ if $s = (d+1)k$. 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 $\mathcal{C}[k]$.[3] From this discussion, it follows that $\mathrm{VCdim}(\mathcal{C}[k]) \leq (d+1)k - 1$ and therefore $\mathrm{VCdim}(\mathcal{C}'[k]) \leq \mathrm{VCdim}(\mathcal{C}[k]) + 2 \leq (d+1)k + 1$. □

4 From Trivial to Smart Algorithms

An algorithm that solves $\mathrm{MinDis}(\mathcal{C}, \mathcal{C})$ is called a *proper* Minimum Disagreement algorithm for $\mathcal{C}$. An algorithm A that solves $\mathrm{MinDis}(\mathcal{C}, \mathcal{H})$ 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 $\mathrm{DS}(S)$ in time $T_1(n)$.
2. After a modification of one of the weights in S, the data structure $\mathrm{DS}(S)$ can be updated accordingly in time $T_2(n)$.
3. $\mathrm{DS}(S)$ implicitly represents a hypothesis $H(S) \in \mathcal{H}$ which satisfies

$$E_S(H(S)) \leq \min_{C \in \mathcal{C}} E_S(C) \ . \tag{1}$$

   Given $\mathrm{DS}(S)$ and $x \in \mathcal{X}$, it can be decided in time $T_3(n)$ whether $x \in \mathcal{H}(S)$.
4. Given $\mathrm{DS}(S)$, 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 samples, the data structure for the composition of S_1 and S_2 can be built in constant time from $\mathrm{DS}(S_1)$ and $\mathrm{DS}(S_2)$.

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 4. Let $\mathcal{C}_1 = \{\mathcal{X}\}$ for $\mathcal{X} = \{x\}$ be the trivial class that we had considered in Example 1 already. We claim that the (trivial) problem $\mathrm{MinDis}(\mathcal{C}_1, \mathcal{C}_1)$ 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 $\mathrm{DS}(S) := W_S^- := \sum_{i: w_i < 0} w_i$. Thus, $\mathrm{DS}(S)$ is simply a real number that can be determined in time $O(n)$.

[3] The same argument was used in [10] in connection with a class of piecewise defined functions over the real domain.

– 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 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 1. We briefly note that there is a flexible algorithm for MinDis($\mathcal{C}_2, \mathcal{C}_2$) which has the same time bounds as the algorithm for 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

$$\sum_{i=1}^{s} n_i = n \implies \left(\sum_{i=1}^{s} T_1(n_i) \leq \sum_{i=1}^{s} (n_i h(n)) = nh(n) = T_1(n) \right) . \tag{2}$$

Here comes the first main result of this section:

Theorem 2. *A flexible algorithm A solving* MinDis($\mathcal{C}, \mathcal{H}$) *with time bounds T_1, T_2, T_3 can be transformed into a flexible algorithm A' that solves* 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 ordering. Let

$$S' = [(x'_1, w_1), \ldots, (x'_n, w_n)] \in (\mathbb{R} \times \mathcal{X} \times \mathbb{R})^n$$

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 ordering 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'. 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. $\mathrm{DS}(u) = \mathrm{DS}(S(u))$, i.e., $\mathrm{DS}(u)$ 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 $\mathrm{DS}(u)$. Let $d_1(u) = E_{S(u)}(H(u))$. We may conclude from (1) that, for all nodes u in B,
$$d_1(u) = E_{S(u)}(H(u)) \le \min_{C \in \mathcal{C}} E_{S(u)}(C) \ . \tag{3}$$
6. Let J be a sub-interval of $[l(u) : r(u)]$. Let $V[J]$ be the smallest antichain V in B that satisfies $J = \cup_{v \in V}[l(v) : r(v)]$. We say that
$$H_J^u = \bigcup_{v \in V[J]} [z'_{l(v)}, z'_{r(v)}] \times H(v) \tag{4}$$
is the *hypothesis induced by J at node u*. Note that $|V[J]| \le 2\lfloor \log n \rfloor$ according to Lemma 1 so that $H_J^u \in \mathcal{H}'[2\lfloor \log n \rfloor]$. Given the convention $\min \emptyset = \infty$, we set
$$\begin{aligned} d_I(u) &= \min_{(a,b): l(u) < a \le b < r(u)} E_{S(u)}(H^u_{[a:b]}) \ , \\ d_L(u) &= \min_{b: l(u) \le b < r(u)} E_{S(u)}(H^u_{[l(u):b]}) \ , \\ d_R(u) &= \min_{a: l(u) < a \le r(u)} E_{S(u)}(H^u_{[a:r(u)]}) \ , \end{aligned}$$
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 $[l(u) : r(u)]$ of type "I". The analogous remark applies to $d_L(u)$ and $d_R(u)$, respectively. The sub-interval J of $[l(u) : r(u)]$ of type "I" that satisfies $d_I(u) = E_{S(u)}(H_J^u)$ is denoted $J_I(u)$ in what follows. The notations $J_L(u)$ and $J_R(u)$ 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 $\mathrm{DS}(S')$. The remaining part of the proof is sketched only. Leaving out of account the computation of K, B can be built in time $O(n)$. The additional pieces 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 (2), it is easy to show 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. Given $(\mathrm{DS}(u))_{u \in B}$, it is easy to compute $(d_1(u))_{u \in B}$ in time $O(n)$.
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 $O(n)$. For instance, if u is a node with left child u_0 and right child u_1, then $d_R(u)$ is computed according to $d_R(u) = \min\{d_0(u_0) + d_1(u_1), d_R(u_0) + d_1(u_1), d_0(u_0) + d_R(u_1)\}$. Similar equations can be set up for $d_L(u)$ and $d_I(u)$. 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 easily 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 $H(S') = H^{r_B}_{[1:n']}$. Finally, if $d_{min} = d_X(r_B)$ for $X \in \{I, L, R\}$, then we set $H(S') = H^{r_B}_{J_X(r_B)}$. The evaluation problem for $H(S') = \emptyset$ is trivial. As for the remaining cases, note first that $(z, x) \in H^{r_B}_{[1:n']}$ iff $z \in [z'_1, z'_{n'}]$ and $x \in H(r_B)$. Suppose now that $d_{min} = d_X(r_B)$. Let $J_X = [a : b]$. If $z \notin [z'_a, z'_b]$, then clearly $(z, x) \notin H^{r_B}_{a,b}$. Otherwise, we use B as a search tree and follow the search path for z until we reach a node v satisfying $d_1(v) = \min\{d_0(v), d_1(v), d_L(v), d_R(v), d_I(v)\}$. This must be a node from the antichain $V[a : b]$. An inspection of (4) shows that $(z, x) \in H^{r_B}_{a,b}$ iff $x \in H(v)$. It follows from this discussion that, in any case, the evaluation problem for $H(S')$ can be solved in time $O(\log(n) + T_3(n))$.

Clearly, d_{min} is retrieved from $\mathrm{DS}(S')$ in constant time. Making use of (3), it is not hard to show that $d_{min} \leq \min_{C \in \mathcal{C}'[1]} E_{S'}(C)$, which concludes the proof. □

Note that the proof of Theorem 2 is completely constructive. The minimum disagreement and learning algorithms given in the following arise from an iterative application of Theorem 2 and the trivial Example 4.

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

Theorem 3. *The transformation from Theorem 2 applied to the flexible algorithm for* $\mathrm{MinDis}(\mathcal{C}_1, \mathcal{C}_1)$ *from Example 4 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 3 basically coincides with the algorithm for $\mathrm{MinDis}(\mathcal{I}, \mathcal{I})$ from [11]. However, since our transformation is general and can be iterated, we can now go one step further and obtain the following result:

Theorem 4. *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 2 that $\mathcal{R} = \mathcal{I}'[1]$. The theorem now follows immediately from Theorems 2 and 3. □

By following the construction in the paragraph before Theorem 1 and applying the bounds on the VC-dimension from Theorem 1 and on the running time T_1 from Theorem 4, one immediately obtains the fast (non-proper) agnostic learner for the class of axis-parallel rectangles promised in the introduction:

Theorem 5. *Our algorithm for the problem* $\mathrm{MinDis}(\mathcal{R}, \mathcal{I}'[2\lfloor \log n \rfloor])$ *agnostically learns concepts from class* $\mathcal{R}$ *with accuracy* ε *in time* $\tilde{O}(1/\varepsilon^2)$ *if a random sample of size* $n = \tilde{O}(1/\varepsilon^2)$ *is provided.*

The proof of the following result (omitted here because of space constraints) bears some similarity to the proof of Theorem 2:

Theorem 6. *Let the function* $\ell_k(n)$ *be defined as in Lemma 1. 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* $T_3'(n) = O(k + \log(n) + T_3(n))$*. 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* $T_2'(n) = O(T_2(n) + k^2 \log^3(n))$.

Recall that $\mathcal{I}_k$ denotes the class of unions of at most k bounded intervals. As mentioned in Example 1, $\mathcal{I}_k$ 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 6 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 [11] (which is specialized to the class $\mathcal{I}_k$):[4]

[4] In [11], 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.

Theorem 7 ([11]). *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 8. *Let the function* $\ell_k(n)$ *be defined as in Lemma 1. 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 3 that $\mathcal{R}_k$ is a subclass of $\mathcal{I}_k[2k+1]$. Combining Theorems 7 and 6, 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. □

The transformations described in Theorems 2 and 6 preserve flexibility but destroy properness. As for the transformation described in the following theorem, we have the reverse situation:

Theorem 9. *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 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

$$H^k = ((-\infty, z_k') \times H_1^k) \cup ([z_k', \infty) \times H_2^k) \ . \tag{5}$$

Furthermore, let k'' be a minimizer of

$$W(k) := E_{S_1(k)}(H_1^k) + E_{S_2(k)}(H_2^k) \tag{6}$$

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

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

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 $k = 1, \dots, n' + 1$ 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 (6), $W(k)$ 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 $k = 1, \dots, n' + 1$ can be computed efficiently. This is done (similarly to a procedure used in [1] 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 $k = 1, \dots, n'+1$, and this takes $O(n\log(n) + T_1(n) + nT_2(n))$ steps. □

5 Experimental Results

We chose to investigate Minimum Disagreement algorithms for the class of axis-aligned rectangles $\mathcal{R}$ in the following experiments, because we expect to observe a considerable improvement in light of Theorem 4: the algorithm from Theorem 4 has an asymptotic worst-case time bound of only $O(n\log(n))$, where n is the number of examples, compared to the proper algorithms from [11, 2] with a running time of $O(n^2\log(n))$ and $O(n^2)$, respectively. While $O(n^2)$ is clearly better than $O(n^2\log(n))$, we noticed that in the range of sample sizes used in the following experiments (and for our implementation) the learner from [11] is actually slightly faster than the one from [2]. Therefore, we compare our method from Theorem 4, which we denote by `TRANS`, with the algorithm from [11], which we denote by `RECT`. 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. The experiments were performed on a AMD Opteron 6234 processor, running at 2400 MHz, with Oracle Java 1.8.0_31 under CentOS 6.6.

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

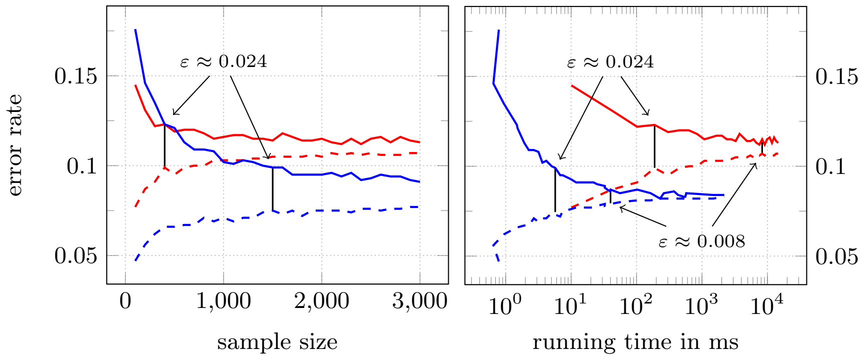

Fig. 2. Error rates of `RECT` (in red) and `TRANS` (in blue) as a function of the sample size (left-hand side) and of the running time (right-hand side) on the artificial Gaussian distribution. The x-axis on the right-hand side is logarithmic to accommodate the vast range of running times. The solid 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 ε from Theorem 5 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.

Gaussian distributions—one distribution for each weight—with identical covariance matrices. The "Glass" data set from [7], 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 [9] (merging classes one and three and removing all instances from class four to seven). The "MAGIC" data set from [4, 6], which consists 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 available on the UC Irvine Machine Learning Repository.

Experimental results and discussion. The results for the Gaussian mixture are given in Fig. 2. The error rates as a function of the sample size, shown in the left-hand side of Fig. 2, 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 right-hand side of Fig. 2, the differences become more drastic: `TRANS` consistently outperforms the much slower `RECT`-algorithm. Furthermore, as predicted by Theorem 5, `TRANS`' error rates indeed converge much faster. We would like to add that the measured running times nicely match the theoretical analysis.

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 [9]

Table 1. Experimental results for the best instance pair using `TRANS` and `RECT` on the Glass and MAGIC data sets (upper table), and for AdaBoost using `TRANS` as the base learner on the MAGIC data set with different numbers of iterations T (lower table). 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

data set	algorithm	T	error rate on training set	error rate on test set	time
MAGIC	AdaBoost	1	0.178	0.189	6 s
	(on `TRANS`)	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

and trained one hypothesis on every pair of coordinates choosing the hypothesis with the smallest error on the training set. (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 9 resp. 10 dimensions.) The results are given in the upper part of 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 [9], 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 [6], which were in the range of 0.16 to 0.138 (on the test set) for different variants of decision trees. We try to close this gap in the following by using the well-known AdaBoost [8] scheme with `TRANS` as the base learner. The results for AdaBoost on the MAGIC data set are given in the lower part of Table 1. 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 [6], and that `TRANS` is obviously fast enough for boosting to be practical on 12680 instances. Furthermore, our boosted classifier surpasses all methods considered in [4] in all but one measure of merit (we omit details due to space constrains). Admittedly, we suspect that this is mostly due to boosting and independent from the choice of base learners, as experiments using decision stumps yield similar error rates with a larger number of iterations but a smaller over-all running time.

References

1. Auer, P., Holte, R.C., Maass, W.: Theory and applications of agnostic PAC-learning with small decision trees. In: ICML '95. pp. 21–29 (1995)
2. Barbay, J., Chan, T.M., Navarro, G., Pérez-Lantero, P.: Maximum-weight planar boxes in $O(n^2)$ time (and better). Information Processing Letters 114(8), 437–445 (2014)
3. Berg, M.d., Cheong, O., Kreveld, M.v., Overmars, M.: Computational Geometry: Algorithms and Applications. Springer-Verlag, Santa Clara, CA, USA (2008)
4. Bock, R., Chilingarian, A., Gaug, M., Hakl, F., Hengstebeck, T., Jiřina, M., Klaschka, J., Kotrč, E., Savický, P., Towers, S., Vaiciulis, A., Wittek, W.: 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), 511 – 528 (2004)
5. Cortés, C., Díaz-Báñez, J.M., Pérez-Lantero, P., Seara, C., Urrutia, J., Ventura, I.: Bichromatic separability with two boxes: A general approach. Journal of Algorithms 64(2-3), 79–88 (2009)
6. Dvorák, J., Savický, P.: Softening splits in decision trees using simulated annealing. In: ICANNGA '07, Part I. pp. 721–729 (2007)
7. Evett, I.W., Spiehler, E.J.: Rule induction in forensic science. Tech. rep., Central Research Establishment, Home Office Forensic Science Service (1987)
8. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119 – 139 (1997)
9. Holte, R.C.: Very simple classification rules perform well on most commonly used datasets. Machine Learning 11(1), 63–91 (1993)
10. Kearns, M.J., Schapire, R.E., Sellie, L.M.: Toward efficient agnostic learning. Machine Learning 17(2), 115–141 (1994)
11. Maass, W.: Efficient agnostic PAC-learning with simple hypothesis. In: COLT '94. pp. 67–75 (1994)
12. Pitt, L., Valiant, L.G.: Computational limitations on learning from examples. Journal of the Association on Computing Machinery 35(4), 965–984 (1988)
13. Shalev-Shwartz, S., Ben-David, S.: Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press (2014)
14. Vapnik, V.: Statistical learning theory. Wiley & Sons (1998)
15. Vapnik, V.N., Chervonenkis, A.Y.: On the uniform convergence of relative frequencies of events to their probabilities. Theory of Probability and its Applications XVI(2), 264–280 (1971)
16. Weiss, S.M., Galen, R.S., Tadepalli, P.: Maximizing the predictive value of production rules. Artificial Intelligence 45(1-2), 47–71 (1990)
17. Weiss, S.M., Kapouleas, I.: An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In: IJCAI '89. pp. 781–787 (1989)
18. Weiss, S.M., Kulikowski, C.A.: Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Expert Systems. Morgan Kaufmann (1990)