Theory and Applications of Agnostic PAC-Learning with Small Decision Trees

Peter Auer[*]
Institute for Theoretical Computer Science
Technische Universitaet Graz
A-8010 Graz, Austria
pauer@igi.tu-graz.ac.at

Robert C. Holte
Computer Science Dept.
University of Ottawa
Ottawa, Canada K1N 6N5
holte@csi.uottawa.ca

Wolfgang Maass
Institute for Theoretical Computer Science
Technische Universitaet Graz
A-8010 Graz, Austria
maass@igi.tu-graz.ac.at

Abstract

We exhibit a theoretically founded algorithm T2 for agnostic PAC-learning of decision trees of at most 2 levels, whose computation time is almost linear in the size of the training set. We evaluate the performance of this learning algorithm T2 on 15 common "real-world" datasets, and show that for most of these datasets T2 provides simple decision trees with little or no loss in predictive power (compared with C4.5). In fact, for datasets with continuous attributes its error rate tends to be lower than that of C4.5. To the best of our knowledge this is the first time that a PAC-learning algorithm is shown to be applicable to "real-world" classification problems.

Since one can *prove* that T2 is an agnostic PAC-learning algorithm, T2 is *guaranteed* to produce close to optimal 2-level decision trees from sufficiently large training sets for *any* (!) distribution of data. In this regard T2 differs strongly from all other learning algorithms that are considered in applied machine learning, for which no *guarantee* can be given about their performance on *new* datasets.

We also demonstrate that this algorithm T2 can be used as a diagnostic tool for the investigation of the expressive limits of 2-level decision trees. Finally, T2, in combination with new bounds on the VC-dimension of decision trees of bounded depth that we derive, provides us now for the first time with the tools necessary for comparing learning curves of decision trees for "real-world" datasets with the theoretical estimates of PAC-learning theory.

1 INTRODUCTION

Numerous articles have been written about the design and analysis of algorithms for PAC-learning, since Valiant [Val84] had introduced the model for <u>p</u>robably <u>a</u>pproximately <u>c</u>orrect learning in 1984. In applied machine learning an even larger literature exists about the performance of various other learning algorithms on "real-world" classification tasks. However, curiously enough, this article apparently marks the first time that the performance of a PAC-learning algorithm for a model powerful enough to cover "real-world" datasets as a special case, is evaluated on "real-world" classification tasks. The PAC-learning algorithm T2 that we have developed for this purpose is described in section 2 of this article, and results about its performance on "real-world" classification problems are discussed in the subsequent sections. In this introduction we will define some basic notions from theoretical and applied machine learning, and also address some obstacles which one has to overcome in order to combine both approaches. It should be mentioned in this context, that although T2 is apparently the first PAC-learning algorithm that is tested on "real-world" classification problems, there has previously been already a fruitful migration of various *ideas* from PAC-learning theory into applications (see e.g. [DSS93]).

In applied machine learning concrete *datasets* from quite diverse application domains are viewed as prototypes for "real-world" classification problems. The performance of many practical learning algorithms on these datasets is described in a number of interesting comparative studies (see e.g. [Min89], [WGT90], [WK90], [WK91], [BN92], [Hol93]). Each dataset is a list of *items* (typically between a few dozen and several thousand), each item consisting of n *attribute values* (typically $n < 40$) and an associated *classification*. The attributes might be *continuous*, ranging over **R**, or *categorical*, ranging over some finite set. For some items some of the attribute values might be missing (i.e. unknown). We will denote items by $x = \langle x_1, \ldots, x_n, x_0 \rangle$ where $x_0 \in \{1, \ldots, p\}$ is the classification of x out of p possible classes (typically $p = 2$), and x_α, $\alpha = 1, \ldots, n$, is the value of attribute α. If α is continuous then $x_\alpha \in \mathbf{R} \cup \{\texttt{missing}\}$, if α is categorical then $x_\alpha \in \{1, \ldots, b, \texttt{missing}\}$ for some fixed b, usually $b \leq 6$. X_n denotes the space of attribute vectors $\langle x_1, \ldots, x_n \rangle$.

[*]Currently with the University of California at Santa Cruz.

A *learning algorithm* A computes for any list S_{train} of items from $X_n \times \{1, \ldots, p\}$ a *hypothesis* $A(S_{\text{train}})$, which represents a function from X_n into $\{1, \ldots, p\}$. The goal of a learning algorithm is to provide correct classifications for *new* items that are produced by the same *distribution* D as the training set S_{train}. From this point of view one can consider a concrete dataset S with m items as a result of m random drawings according to this distribution D. The *true error* (also called generalization error) of a hypothesis $H : X_n \to \{1, \ldots, p\}$ (with respect to the underlying distribution D) is given by $Err_D[H] := Pr_{\langle x_1, \ldots, x_n, x_0 \rangle \in D}[H(x_1 \ldots, x_n) \neq x_0]$. Of course $Err_D[H]$ is not available in practice, but it can be estimated by the error-rate $Err_{S_{\text{test}}}[H]$ of H ([1]) on a test set S_{test} of randomly drawn items from the dataset S. Of course the expected value of the true error $Err_D[A(S_{\text{train}})]$ will in general *depend* not only on A, but also on the size m of S_{train} and on the underlying distribution D.

"Agnostic" PAC-learning (due to [Hau92] and [KSS92]) is the variant of PAC-learning (due to [Val84]) most relevant to practical machine learning. It differs from normal PAC-learning in two important ways. In normal PAC-learning, training data is generated by sampling examples according to a probability distribution over the space X_n of attribute vectors, and then classifying them according to a target concept from a known concept class. In agnostic PAC-learning, the probability distribution is over the product of X_n and the set $\{1, \ldots, p\}$ of possible classifications; there is no notion of target concept or class, and no restriction on the distribution. The term agnostic emphasizes that the learning algorithm has no *a priori* knowledge whatsoever about the process that classifies examples. The second distinctive characteristic of agnostic PAC-learning is its definition of "successful" learning. In normal PAC-learning, an algorithm is required to find an arbitrarily close approximation to the target concept. In agnostic PAC-learning, the target is not a concept from a known class but an arbitrary probability distribution. For many targets there will be no good approximation in the learning algorithm's hypothesis class, $\mathcal{H}$. To "succeed" in this setting a learning algorithm is required to find a hypothesis in $\mathcal{H}$, that approximates the target distribution nearly as close as possible. An algorithm is an efficient agnostic learning algorithm (for hypothesis class $\mathcal{H}$) if, for any target distribution, it can find a hypothesis arbitrarily close to the best approximation in $\mathcal{H}$ of the target distribution, in polynomial time with a polynomially-sized sample. More precisely, in PAC-learning theory one says that A is an *agnostic PAC-learning algorithm* if there exists a function $m : \mathbf{R}^2 \times \mathbf{N} \to \mathbf{N}$ that is bounded by a polynomial, such that for any given $\varepsilon, \delta > 0$, any $n \in \mathbf{N}$, for any distribution D over $X_n \times \{1, \ldots, p\}$, and any sequence S_{train} of $\geq m(1/\varepsilon, 1/\delta, n)$ items drawn according to D, $|Err_D[A(S_{\text{train}})] - \inf_{H \in \mathcal{H}_n} Err_D[H]| \leq \varepsilon$ with probability $\geq 1 - \delta$ (with regard to the random drawing of S_{train}). One says that A is an *efficient* agnostic PAC-learning algorithm if $A(S_{\text{train}})$ can be computed with a number of computation steps that is bounded by a polynomial in the (bit-length) size of the representation of S_{train}. [Hau92] and [KSS92] have shown that there is an efficient PAC-learning algorithm for a family of hypothesis classes $\mathcal{H}_n$ if and only if the VC-dimension of $\mathcal{H}_n$ grows polynomially in n and there is a polynomial time algorithm that computes for any set S_{train} of items a hypothesis $H \in \mathcal{H}_n$ that minimizes $Err_{S_{\text{train}}}[H]$.

In other attempts to make the original version of PAC-learning (where one focuses on hypothesis classes $\mathcal{H}_n$ and distributions D such that there is some "target concept" $C_t \in \mathcal{H}_n$ with $Err_D[C_t] = 0$) more realistic, it has been extended to include certain "noise models" (see e.g. [AL88], [EK91], [Elo92], [Kea93], [KL93], [Dec93]): the target concept $C_t \in \mathcal{H}_n$ is either disguised by a large amount of "white" noise, or by a small (in comparison with the desired error rate of the learner) fraction of arbitrary (even "malicious") noise. Unfortunately the version with "white" noise does not model the situation that one encounters in the here considered "real-world" classification problems S (e.g. the systematic noise reported in [DP93]). On the other hand, in the model with malicious noise the PAC-learner can only achieve error rates that are intolerably large from the point of view of applied machine learning.

Although it is rather obvious that the model for *agnostic* PAC-learning is the most adequate one for the investigation of "real-world" classification tasks, relatively few results are known for this model. One reason is perhaps that there do exist two remarkable *negative* results. It has been shown that neither for $\mathcal{H}_n = \{\text{halfspaces over } \mathbf{R}^n\}$ [HSV93] nor for $\mathcal{H}_n = \{\text{monomials over } n \text{ boolean attributes}\}$ [KSS92] does there exist an efficient agnostic PAC-learning algorithm (unless $\mathcal{RP} = \mathcal{NP}$). These negative results are quite disappointing, since in learning theory one usually views these as the "simplest" nontrivial hypothesis classes for continuous and boolean attributes respectively. On the other hand one *did* succeed in designing agnostic PAC-learning algorithms for a few hypothesis classes $\mathcal{H}_n$ ([KSS92], [Maa93], [Maa94], [DGM95]). However, for these classes $\mathcal{H}_n$, either the polynomial time bound of the algorithm is too large, or $\mathcal{H}_n$ is less interesting for most applications, since one expects that the least possible error $\inf_{H \in \mathcal{H}_n} Err_S[H]$ that one can achieve with hypotheses from this class is, for "real-world" datasets S, substantially larger than the error actually achieved by existing heuristic algorithms.

In this article we present not only a polynomial but a nearly linear time agnostic PAC-learning algorithm T2 for the hypothesis class $\mathcal{H}_n$ of 2-level decision trees over n attributes. The results of our experiments demonstrate that this hypothesis class is rich enough to contain good classifiers for most of the common "real-world" datasets.

A straightforward exhaustive search algorithm which looks for the 2-level decision tree with the least error-rate on the training set runs in polynomial time and satisfies the requirements for efficient agnostic PAC-learning. But notice that for training sets of size m with continuous attributes in

[1]Formally we have for $S_{\text{test}} = (x^i)_{1 \leq i \leq t}$ $Err_{S_{\text{test}}}[H] := \left|\{i \in \{1, \ldots, t\} : H(x_1^i, \ldots, x_n^i) \neq x_0^i\}\right|/t$.

general m^3 decision trees have to be considered. The main advantage of our algorithm T2 is that it finds the 2-level decision tree with the least error-rate on the training set in nearly linear time $O(m \log m)$. Thus T2 is applicable even to very big datasets.

A main difference between an agnostic PAC-learning algorithm and those learning algorithms that are usually considered in applied machine learning is that an agnostic PAC-learning algorithm A "performs well" for *any* (!) distribution D over $X_n \times \{1, \ldots, p\}$, in the sense that it computes, given an S_{train} (sufficiently large relative to the VC-dimension of $\mathcal{H}_n$) drawn according to D, a hypothesis $A(S_{\text{train}})$ whose true error (with high probability) is arbitrarily close to the *least true error* of *any* hypothesis from the associated hypothesis class $\mathcal{H}_n$. On the other hand, in applied machine learning one finds out at best that a particular learning algorithm "performs well" for certain commonly considered datasets ("distributions"), and usually it is quite hard to *predict* whether such a learning algorithm will perform well for an entirely new dataset. In fact, even for extremely successful practical learning algorithms such as C4.5 [Qui92] it is relatively easy to construct distributions (respectively datasets) for which they do *not* "perform well" in the abovementioned sense, and hence these are *not* agnostic PAC-learning algorithms.

In practice, however, training sets are virtually always smaller than the size needed for these theoretical performance guarantees. An important practical question is, how do agnostic PAC-learning algorithms perform on "real-world" training sets? This question is investigated in section 3 and 4, where T2 is experimentally evaluated on a wide variety of "real-world" datasets, and its performance is compared to C4.5, a state of the art decision tree learning algorithm.

2 THE AGNOSTIC PAC-LEARNING ALGORITHM T2

We will describe in this section the new learning algorithm T2, we prove in Theorem 1 that T2 is a computationally efficient agnostic PAC-learning algorithm, and we exhibit some extensions of our approach in Theorems 2 and 3. The algorithm T2 computes for any given list L of m examples x from $X_n \times \{1, \ldots, p\}$ and any given $K \in \mathbf{N}$ in $O(K^2 \cdot n^2 \cdot m \cdot \log m)$ computation steps (on a RAM with uniform cost criterion) a 2-level tree T that makes a *minimal* number of incorrect classifications for points in L (compared with all other trees in TREE$(2, n, p, K)$)[2].

The hypothesis class TREE$(2, n, p, K)$ consists of all functions $f : X_n \rightarrow \{1, \ldots, p\}$ that can be represented by a 2-level decision tree T in the usual fashion. At the root of T (= first level of T) one queries either a categorical attribute β with $b(\beta) \leq b$ possible values (in which case $b(\beta) + 1$ edges leave the root of T, labeled by $1, \ldots, b(\beta)$, `missing`), or one queries a continuous attribute β (in which case 3 edges leave the root, labeled by $I_1, I_2,$ `missing`, for some partition I_1, I_2 of $\mathbf{R}$ into two intervals)[3]. On the second level of T each node ν is either labeled by some classification $c \in \{1, \ldots, p\}$, or it queries another attribute α ($\alpha = \beta$ is also allowed). If α is a categorical attribute, the $b(\alpha) + 1$ edges with labels $1 \ldots, b(\alpha)$, `missing`, leave the node ν. If α is a continuous attribute, then $k(\nu) + 1 \leq K + 1$ edges leave ν with labels $I_1, \ldots, I_{k(\nu)}$, `missing` , where $I_1, \ldots, I_{k(\nu)}$ is some partition of $\mathbf{R}$ into $k(\nu) \leq K$ intervals. Notice that at the root a continuous attribute is only split into 2 intervals, whereas on the second level it can be split into up to K intervals. All leaves of T are labelled by classifications $c \in \{1, \ldots, p\}$. It should be noted that up to $2 + b$ attributes may be queried altogether in such a 2-level tree T, and T can have up to $(b + 1) \cdot (1 + \max(b, K))$ leaves.

We will also discuss on the side the hypothesis class TREE$(1, n, p, K)$ of functions $f : X_n \rightarrow \{1, \ldots, p\}$ that are defined by 1-level trees. In a 1-level tree only a single attribute α is queried (at the root of the tree), which has similarly as the nodes on level 2 of the 2-level trees either $b(\alpha) + 1$ outgoing edges (if α is a categorical attribute), or up to $K + 1$ edges (if α is a continuous attribute). Note that TREE$(1, n, p, K)$ is the hypothesis class that is used by Holte's learning algorithm 1R [Hol93]. In our experiments we have always chosen $K := p + 1$. Furthermore we always identify a decision tree T with the function $f : X_n \rightarrow \{1, \ldots, p\}$ that is computed by T, and occasionally we write TREE(d) instead of TREE(d, n, p, K).

The algorithm T2 essentially tries out all possible assignments of attributes to the up to $b + 2$ query-nodes in a 2-level tree T. This is done in a careful manner so that it only gives rise to a factor n^2 in the time-bound (instead of n^{b+2}). For each assignment of attributes to query nodes the algorithm T2 computes in $O(m \log m)$ steps (i.e. up to constant factors *as fast as sorting* the list L according to one of its continuous attributes) an optimal assignment of labels to the edges and leaves in T. More precisely, T2 computes endpoints for the up to $\max(2 + 3K, (b + 1)K)$ intervals for continuous attributes in T, and it assigns classifications $c \in \{1, \ldots, p\}$ to all leaves of T, so that the resulting number of misclassifications of items in L is minimal among all 2-level trees with the same assignment of attributes to query nodes. Of course in case that continuous attributes are queried both on level 1 and level 2, the associated intervals *cannot* be optimized *independently* from each other, and the most delicate part of the algorithm T2 is the reduction of this 2-dimensional optimization problem to a 1-dimensional problem that is more complicated, but which can be solved in $O(m \log m)$ computation steps (see [Maa94], [DGM95] for other applications of such a method).

We create "from below" more complex datastructures, which not only tell us for an interval I of the range of

[2]For a special case of depth 2 decision trees ("corners") a $O(m \log m)$ algorithm was already given in [Lub94]

[3]Observe that we treat `missing` as another attribute value.

a continuous attribute α an optimal split of I into $\leq K$ intervals with associated classifications (where "optimal" refers to minimizing the number of incorrect classifications of items in L with $x_\alpha \in I$). In addition, we also compute for any $k \leq K$ the optimal split into k intervals with associated classifications, and we do this separately for all possible choices of the classification c_{left} of the leftmost interval, and all possible choices of the classification c_{right} of the rightmost interval. The advantage is, that if we have all these data available for two adjacent intervals I and I', we can compute rather easily via the procedure MERGE the corresponding data for the *union* $I \cup I'$ of both intervals. In order to illustrate this, we consider a scenario where I' lies to the right of I, and for any optimal split of $I \cup I'$ into K intervals one of its K intervals has a nonempty intersection with both I and I'. One can detect (and exhibit) this optimal split if one examines for all $c \in \{1, \ldots, p\}$ and for all $k, k' \in \{1, \ldots, K\}$ with $k + k' - 1 = K$ the total number of misclassifications that result from combining an optimal split of I into k intervals with $c_{\text{right}} = c$, and an optimal split of I' into k' intervals with $c_{\text{left}} = c$.

However the procedure for computing an optimal split for an attribute α that is queried by a node on level 2 has to be intertwined with the search for an optimal decision boundary for another continuous attribute β that is queried on level 1 (i.e. at the root) of the same decision tree, since otherwise we would just get an $O(m^2 \log m)$ algorithm (instead of the desired $O(m \log m)$ algorithm). This combination of 2 simultaneous search procedures makes the algorithm T2 conceptually a bit more complicated. We have to assemble for each interval I the previously described data separately for each sublist $\tilde{L}$ that may result from L by a split of the range $\mathbf{R}$ of the other attribute β into two intervals $(-\infty, y)$ and $[y, \infty)$, where y is the value x_β of attribute β for some $x \in L$ with $x_\alpha \in I$. This strategy causes another small technical complication, since the set of values y that arises in this way, will in general be *different* for different intervals I, I'. However it turns out to suffice if one combines in the procedure MERGE the data for some y in the datastructure for I with the *next larger* value y' that occurs in the datastructure for I' (since we may in this case conclude that there does not exist any point $x \in L$ with $x_\alpha \in I'$ and $x_\beta \in [y, y')$, hence no additional misclassification of points in L can arise in this way).

More precisely, the algorithm T2 proceeds as follows. Assume that two continuous attributes $\alpha, \beta \in \{1, \ldots, n\}$ have been fixed. T2 computes for various lists $\tilde{L}$ of items from $X_n \times \{1, \ldots, p\}$, for any $k \in \{1, \ldots, K\}$, and any $c_{\text{left}}, c_{\text{right}} \in \{1, \ldots, p\}$, a partition OPTSPLIT$^\alpha_{k, c_{\text{left}}, c_{\text{right}}}(\tilde{L})$ of the range $\mathbf{R}$ of the attribute α into k intervals $I_1, \ldots, I_k$ (numbered from left to right), together with a classification $C(I_i) \in \{1, \ldots, p\}$ for each $i \in \{2, \ldots, k-1\}$, so that in combination with the classifications $C(I_1) := c_{\text{left}}$ and $C(I_k) := c_{\text{right}}$ this split minimizes the number of items x in $\tilde{L}$ for which x_0 differs from the classification $C(I_i)$ of the interval I_i with $x_\alpha \in I_i$. Formally OPTSPLIT$^\alpha_{k, c_{\text{left}}, c_{\text{right}}}(\tilde{L})$ is a vector whose components are the endpoints of those intervals $I_1, \ldots, I_k$ together with their chosen classifications $C(I_2), \ldots, C(I_{k-1})$. In addition its last component specifies the number of incorrect classifications of items $x \in \tilde{L}$ that result from this split. The second considered attribute β determines for which sublists $\tilde{L}$ of L the preceding data are assembled. Ignoring missing values, all sublists $\tilde{L}$ are of the form $\tilde{L} := \langle x \in L : x_\alpha \in I \text{ and } x_\beta < y \rangle$ or $\tilde{L} := \langle x \in L : x_\alpha \in I \text{ and } x_\beta \geq y \rangle$ for some interval I and some $y \in \mathbf{R}$. We will focus on the handling of lists $\tilde{L}$ of the first type, since the handling of lists $\tilde{L}$ of the second type is analogous. Thus we consider arrays of the form $V_I^{\alpha,\beta}(y) := \langle \text{OPTSPLIT}^\alpha_{k, c_{\text{left}}, c_{\text{right}}}(\langle x \in L : x_\alpha \in I \text{ and } x_\beta < y \rangle) \rangle^{c_{\text{left}}, c_{\text{right}} \in \{1,\ldots,p\}}_{k \in \{1,\ldots,K\}}$, and we write $V_I^{\alpha,\beta}$ for the list of all $V_I^{\alpha,\beta}(y)$ such that $y = \tilde{x}_\beta$ for some $\tilde{x} \in L$ with $\tilde{x}_\alpha \in I$, sorted according to y.

We employ a procedure MERGE which computes $V_{I \cup I'}^{\alpha,\beta}$ from $V_I^{\alpha,\beta}$ and $V_{I'}^{\alpha,\beta}$ for certain pairs I, I' of adjacent intervals (with I' to the right of I). Consider some $y \in \mathbf{R}$ such that $y = \tilde{x}_\beta$ for some $\tilde{x} \in L$ with $\tilde{x}_\alpha \in I \cup I'$. In the case where $\tilde{x}_\alpha \in I$, the procedure MERGE combines data from $V_I^{\alpha,\beta}(y)$ and $V_{I'}^{\alpha,\beta}(y')$, where y' is the least value $\hat{x}_\beta$ for any $\hat{x} \in L$ with $\hat{x}_\alpha \in I'$ and $\hat{x}_\beta \geq y$ (we had motivated this choice in our informal remarks). If $\tilde{x}_\alpha \in I'$, we go to some analogously chosen $y' = \hat{x}_\beta \geq y$ for some $\hat{x} \in L$ with $\hat{x}_\alpha \in I$, and combine data from $V_I^{\alpha,\beta}(y')$ and $V_{I'}^{\alpha,\beta}(y)$. Since the arrays in the lists $V_I^{\alpha,\beta}$ and $V_{I'}^{\alpha,\beta}$ are assumed to be sorted according to y, the total number of computation steps of MERGE is proportional to the product of K^2 and the number of items $x \in L$ with $x_\alpha \in I \cup I'$ (we assume that p is a constant).

The algorithm T2 initializes these operations with a partition of $\mathbf{R}$ into at most m intervals I, such that each I contains exactly one value x_α of the attribute α of the items x in L, and it computes $V_I^{\alpha,\beta}$ for each of these intervals. In the next phase it merges pairs of adjacent intervals I and I' (so that each I and I' occurs in exactly one pair), and it computes $V_{I \cup I'}^{\alpha,\beta}$ with the help of the procedure MERGE for each of the resulting larger intervals $I \cup I'$. After repeating this phase $\lceil \log m \rceil - 1$ times, we have in this way computed $V_{\mathbf{R}}^{\alpha,\beta}$. Obviously the total number of computation steps in each phase can be bounded by $O(K^2 \cdot m)$. Hence we have computed $V_{\mathbf{R}}^{\alpha,\beta}$ in altogether $O(K^2 \cdot m \log m)$ steps.

From $V_{\mathbf{R}}^{\alpha,\beta}$ one can read off in $O(K \cdot m)$ steps for each value y of the attribute β of some item in L an optimal assignments of labels to edges and leaves for a subtree of a decision tree from TREE$(2, n, p, K)$ that queries the attribute α on level 2, and which is connected by an edge with label $(-\infty, y)$ to the node on level 1 where attribute β is queried.

Analogously one can also compute in altogether $O(K^2 \cdot m \log m)$ steps such optimal assignment of labels for a similar subtree which is connected by an edge with label

$[y, \infty)$ or label "`missing`" to the node on level 1 where attribute β is queried.

The algorithm T2 carries out the preceding computations successively for all attributes α, β, which gives rise to a factor n^2 in its time bound. In the case where α or β are categorical attributes, one can replace in the previously described subroutines the rather sophisticated computation of optimal intervals $I \subseteq \mathbf{R}$ as labels for edges by an exhaustive search over all of the up to $b + 1$ values of a categorical attribute. Thus we have proven the following result.

Theorem 1 *The algorithm T2 computes for any $n \in \mathbf{N}$, any $K \in \mathbf{N}$, and any list L of m items from $X_n \times \{1, \ldots, p\}$ in $O(K^2 \cdot n^2 \cdot m \log m)$ computation steps a decision tree from TREE$(2, n, p, K)$ that misclassifies a minimal number of items in L.*

Theorem 2 *T2 is an algorithm for efficient agnostic PAC-learning with hypothesis class TREE$(2, n, p, K)$.*

Proof: It is easy to show (see section 4) that for fixed p the VC-dimension of TREE$(2, n, p, K)$ is bounded by a polynomial in n and K. This fact, in combination with Theorem 1, implies according to [Hau92] that T2 is an efficient agnostic PAC-learning algorithm. ∎

According to Theorem 1 the algorithm T2 outputs a tree T that minimizes the "disagreement between T and L", i.e. $|\{x \in L : T(x_1, \ldots, x_n) \neq x_0\}|$. However T2 can easily be adapted to optimize instead of the "disagreement" any other "additive split criterion" in the sense of Lubinsky [Lub94]. In particular it can be used to minimize the total cost of all misclassifications for any given "confusion matrix" [WK91]. As a special case T2 yields a computational tool for choosing *optimal multi-variate* splits with regard to the split criterion *weighted inaccuracy* ("wacc"). Lubinsky [Lub94] has shown that this split criterion "wacc" performs for many datasets as well as "Gini" [BFOS84], if used as a criterion for greedy algorithms that build decision trees of unbounded depth. Another extension of algorithm T2 is considered in the following result, which may be of some practical interest for small datasets with few attributes and $d = 3$, or even $d = 4$.

Theorem 3 *One can design for any $d \geq 2$ an algorithm Td that computes in $O(K^2 \cdot n^d \cdot m^{d-1} \cdot \log m)$ computation steps for any $n \in \mathbf{N}$, any $K \in \mathbf{N}$, and any given list L of m items from $X_n \times \{1, \ldots, p\}$ a tree $T \in$ TREE(d, n, p, K) that makes a minimal number of misclassifications of items in L. Analogously T1 computes in $O(K^2 \cdot n \cdot m \cdot \log m)$ steps an optimal tree from TREE$(1, n, p, K)$. Hence Td, for $d \geq 1$, are algorithms for efficient agnostic PAC-learning with hypothesis classes TREE(d, n, p, K).*

3 EVALUATION OF T2 ON "REAL-WORLD" CLASSIFICATION PROBLEMS

In this section, T2's performance on a diverse set of "real-world" datasets is experimentally compared with that of C4.5 ([Qui92]), a state-of-the-art heuristic machine learning algorithm. C4.5's hypothesis class includes decision trees of arbitrary depth, but with only binary splits on continuous attributes. Of the fifteen datasets used in the experiments nine (BC,CH,G2,HD,HE,IR,LA,SE,SO) have already been used in [Hol93], and six are new. AP, the appendicitis dataset in [WGT90], was kindly supplied by S. Weiss of Rutgers University. The other five new datasets were obtained from the UCI repository.[4] Table 1 gives the datasets' main characteristics. "Size" is the total number of examples in the dataset. "Classes" is the number of classes in the dataset. "Baseline Accuracy" is the percentage of examples in the most frequently occurring class in the dataset. "Missing values" indicates whether there are any examples in the dataset for which the value of some attribute is unknown. The remaining columns give the number of continuous ("cont.") attributes and categorical attributes with various numbers of values. Attributes that have the same value on all examples are not counted.

The accuracy (= percentage of correct classifications on the testing set) for T2 and C4.5 reported in Table 2 is the average of 9 independent 25-fold cross-validation[5] estimates (a total of 225 runs were made of each system on each dataset). The tiny numbers in these lines give the standard deviation of the 9 crossvalidation estimates[6]. The complexity of the trees produced by C4.5 was measured on each run in 3 different ways: "dynamic complexity" ($d.c.$) is the number of attributes queried in order to classify the average example; "$\% > 2$" is the percentage of examples classified at depth greater than 2; and "depth" is the depth of the tree. Table 2 gives the average of each complexity measure over all the runs. Sky2 is the optimal accuracy of any 2-level tree on the entire dataset; it was obtained by running T2 on the whole dataset.

Comparing the accuracy results, we observe several effects. On HE and LA the difference between T2's and C4.5's ac-

[4]FTP site ics.uci.edu, directory pub/machine-learning-databases. The subdirectories in the repository corresponding to these datasets are: credit-screening (CR), ionosphere (IO), pima-indians-diabetes (PI), molecular-biology/promoter-gene-sequences (PR), and molecular-biology/splice-junction-gene-sequences (SP).

[5]To evaluate a learning algorithm A on a dataset S by N-fold cross-validation one partitions S randomly into N pieces of about equal size. For each of the resulting N pieces S' one computes the hypothesis $A(S_{\text{train}})$ for $S_{\text{train}} := S - S'$ and records its error-rate $Err_{S_{\text{test}}}[A(S_{\text{train}})]$ for $S_{\text{test}} := S'$. One then takes the *average* of the resulting N error-rates $Err_{S_{\text{test}}}[A(S_{\text{train}})]$ as a measure for the performance of learning algorithm A on dataset S.

[6]The standard deviation gives a very rough estimate on the significance of the difference between the averages for T2 and C4.5.

Dataset	Size	Classes	Baseline accuracy	Missing values	Attributes ... number of distinct values							
					cont.	2	3	4	5	6	>6	Total
AP	106	2	80.2	yes	8							8
BC	286	2	70.3	yes		3	2		1	1	2	9
CH	3196	2	52.2	no		35	1					36
CR	690	2	55.5	yes	6	4	2	1			2	15
G2	163	2	53.4	no	9							9
HD	303	2	54.5	yes	5	3	3	2				13
HE	155	2	79.4	yes	6	13						19
IO	351	2	64.1	no	32	1						33
IR	150	3	33.3	no	4							4
LA	57	2	64.9	yes	8	3	5					16
PI	768	2	65.1	no	8							8
PR	106	2	50.0	no				57				57
SE	3163	2	90.7	yes	7	18						25
SO	47	4	36.2	no		13	3	4			1	21
SP	3190	3	51.9	yes				60				60

Table 1: Datasets used in the experiments

	Datasets														
	HE	LA	AP	G2	IR	PI	CH	SP	BC	PR	SE	SO	HD	IO	CR
Sky2	89.0	98.2	96.2	87.7	98.7	78.0	86.9	79.6	79.4	92.5	97.6	100.	82.5	92.9	87.7
T2	$78.6_{3.1}$	$86.6_{2.0}$	$88.6_{0.8}$	$79.7_{1.4}$	$95.7_{0.6}$	$74.8_{0.6}$	$86.9_{0.0}$	$79.4_{0.2}$	$66.3_{1.2}$	$69.3_{1.8}$	$97.3_{0.1}$	$91.1_{1.4}$	$67.1_{1.2}$	$86.1_{0.6}$	$84.2_{0.1}$
C4.5	$80.2_{1.1}$	$86.1_{2.3}$	$85.1_{0.8}$	$76.5_{1.3}$	$94.1_{0.9}$	$73.4_{1.1}$	$99.2_{0.0}$	$94.4_{0.1}$	$75.2_{0.4}$	$76.6_{1.7}$	$98.0_{0.1}$	$97.8_{0.6}$	$74.5_{0.8}$	$94.3_{0.5}$	$84.9_{0.4}$
d.c.	1.91	1.86	1.93	4.37	2.00	6.77	4.56	3.55	1.22	1.65	1.56	1.81	3.22	4.66	2.85
$\% > 2$	10	17	1	90	34	78	58	68	1	10	13	2	83	76	33
depth	4.82	3.00	3.00	9.82	4.00	33.3	13.0	8.00	3.86	3.00	7.56	2.97	14.0	11.2	18.3

Table 2: Experimental comparison of T2 and C4.5

curacy is virtually zero. On AP, G2, IR, and PI (whose attributes are all continuous) T2's trees are slightly more accurate than C4.5's, which means that for these datasets T2's optimal choice of the decision tree (in respect to the training set) is superior to C4.5's heuristic choice. On CH and SP the performance of T2 is limited by the expressiveness of the hypothesis class TREE(2) (see the values of Sky2). On these datasets, and only on these, the restriction to 2-level trees definitely reduces accuracy.

For the datasets BC, PR, SE, and SO C4.5 found trees of small depth (compare the complexity measures) with a higher accuracy than T2's trees. This suggests that T2 was overfitting the training data in these cases. This provides a useful reminder of the fact that the theoretical guarantee of T2's performance is not significant when the training set is small. C4.5 outperformed T2 on HD and IO, too. Note that for these datasets most examples were classified by C4.5 beyond level 2 of the trees, but that the Sky2 values of HD and IO were considerable higher than the accuracy of T2. Dataset CR seems to be a borderline case where C4.5 took advantage of using a more complex hypothesis than T2, but also it is likely that T2 was overfitting the training data (observe that this dataset has two multi-valued categorical attributes).

Overall we would like to point to the somewhat curious fact that T2, which is the straightforward implementation of a simple theoretical approach, produces results that are not altogether incomparable to those which are produced by a state-of-the-art learning algorithm such as $C4.5$, which is the result of many years of experimenting and fine-tuning. This suggests that there exists, especially for domains that have mostly continuous attributes, substantial room for designing learning algorithms that output simpler hypotheses *and* achieve higher accuracy.

Turning to the complexity results, C4.5's trees had an average dynamic complexity of 2.93, and an average depth of 9.32. T2's trees, of course, are guaranteed to have a dynamic complexity and depth of at most 2. Depth (or some other measure of static complexity) is a suitable complexity measure if the tree is to be interpreted by humans. In this case, "a simple, although only approximately accurate concept definition may be more useful than a completely accurate definition which involves a lot of detail" (p.223 [BB94]). From this perspective, C4.5's trees for G2, CR, PI, and SE are clearly too complex. For HD and IO, C4.5's trees are considerably more complex than T2's but only moderately more accurate. For the purposes described in [BB94], T2's trees are the more desirable. CH and SP are special cases since they absolutely require a complex tree to attain high accuracy.

In the framework defined in [BB94], the hypotheses produced by "learning algorithms" are not used to predict unseen instances, but to summarize the contents of a dataset. An intriguing question is whether common "real-world" classification problems S can in fact be characterized accurately by a simple hypothesis of a given type, see e.g. [Elo94]. Unfortunately very little information of this type is available at this point, since it is usually too time consuming to compute $\min_{H\in\mathcal{H}} Err_S(H)$ for interesting classes $\mathcal{H}$ of simple hypotheses and common datasets S (for an exception see [WGT90] for the case of production rules of length ≤ 3 on the dataset AP). It should be noted here that, because OPT[BB94] and 1Rw[Hol93] are heuristic measures, these values underestimate the ability of simple rules to summarize a dataset. However with the help of the algorithms Td from section 2, it is feasible now to calculate the true capacity for summarization for the hypothesis spaces TREE(d, n, p, k) for small values of d. The average of the Sky2 values in Table 2 is 89.8%, and it is below 85% on only 4 of the datasets. Clearly, T2 is an excellent algorithm for the task of summarizing datasets defined in [BB94].

4 LEARNING CURVES FOR DECISION TREES OF SMALL DEPTH

At the heart of learning theory are certain statistical results (due to [Vap82], [BEHW89], [Hau92], [Tal94], [DL95] and others) that provide for *any* distribution D, and any size of the training set S_{train} drawn according to D, an upper bound for the difference between the true error $Err_D[\hat{H}]$ of a hypothesis $\hat{H} \in \mathcal{H}$ that minimizes $Err_{S_{\text{train}}}[H]$ and the least true error $\inf_{H\in\mathcal{H}} Err_D[H]$ of *any* hypothesis in $\mathcal{H}$. However, because of the previous lack of algorithms that are sufficiently efficient so that one can *actually compute* such hypotheses $\hat{H}$ for interesting hypothesis classes $\mathcal{H}$ and "real-world" data, these theories have so far been tested only an artificial data (see e.g. [SSSD90], [CT92]).

Our new algorithm T2 now permits us, for the first time, to actually *compute* a hypothesis $\hat{H} \in$ TREE(2) that minimizes $Err_{S_{\text{train}}}$, hence we can evaluate the predictions of this essential piece of PAC-learning theory on "real-world" classification problems. This is of quite some interest, because the abovementioned results hold for the *worst case* distribution D of data and hence also for those distributions that "generate" the common benchmark datasets S. But so far it is unknown whether the distributions D that generate these "real-world" datasets S behave enough like worst-case distributions to make these theoretical estimates *significant*.

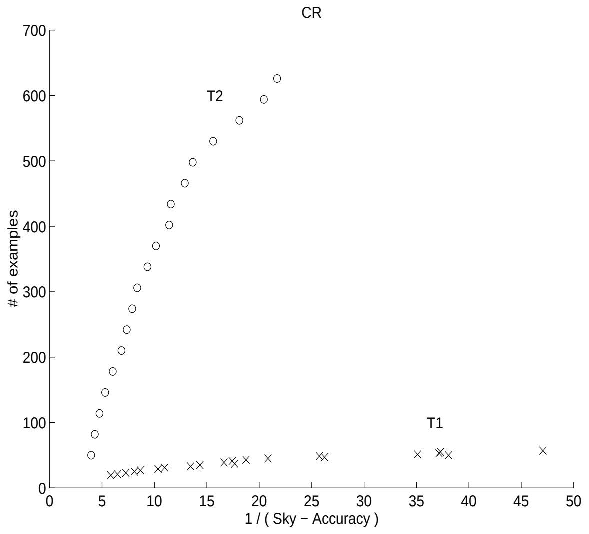
There is an important difference between decision trees of a fixed depth d for classification problems with categorical attributes, and decision trees of depth d that also have continuous attributes. In the latter case the number of trees that cause different classifications for some points in S_{train} grows with the number of examples in S_{train}, whereas it is fixed in the first case. This might suggest that choosing a 2-level decision tree that *optimally* fits the training set is especially harmful (because of "overfitting") for those learning problems that possess some *continuous* attributes. However the abovementioned theoretical results entail that not really the *size* of the hypothesis class $\mathcal{H}_n$ (respectively of $\{H \cap S_{\text{train}} : H \in \mathcal{H}_n\}$) is relevant in this context, but only the VC-dimension of $\mathcal{H}_n$. We will show in Theorem 4 that continuous attributes do *not* increase the VC-dimension of decision trees by very much. To the best of our knowledge, Theorem 4 provides the first significant bounds for the VC-dimension of decision trees of bounded depth. Its proof requires nontrivial combinatorial arguments.

Theorem 4 *The VC-dimension of TREE(d, n, p, K) can be bounded by $\Theta(\log n)$ for any fixed d, p, K. Furthermore the contribution of a continuous attribute to this VC-dimension is not larger than that of a categorical attribute with $K(1 + \log\log n)$ different values.*

More precise bounds depend on the specific structure of the trees (respectively on the relative sizes of d, b, n, p, K, where b is the maximal number of values of a categorical attribute). For example the VC-dimension of the class of decision trees from TREE$(d, n, 2, K)$ that query on levels $1, \ldots, d-1$ categorical attributes with b values, and which query continuous attributes on level d, lies between $(b+1)^{d-1}(\log(n+1-d)+\max(K, b))$ and $((b+1)^d-1)\log n + ((b+1)^{d-1}(K-1))+1)\log\log n$, where we have to assume for the upper bound that n is sufficiently large. ∎

Because of space limitations we can report here our experimentally determined learning curves for T1 and T2 only for the datasets CR and SE. In order to compare these learning curves with the abovementioned upper bounds on the difference ε between $Err_D[\hat{H}]$ and $\inf_{H\in\mathcal{H}} Err_D[H]$, we have plotted in Figure 1 the size m of S_{train} as a function of the inverse $1/\varepsilon$ of this difference ε for the hypothesis classes TREE(1) and TREE(2). We have measured this ε by subtracting from the true errors of the hypotheses T1(S_{train}) respectively T2(S_{train}) our best "guesses" Sky1 and Sky2 for the true errors of the best hypotheses for the considered datasets S. These guesses are somewhat unsatisfactory, but arguably the best approximation that we can achieve (without having access to *further* examples from the distribution D that generated S).

[Hau92], [Tal94], [DL95] have achieved the best known theoretical bounds for the minimal size m of a training set that is needed to guarantee a certain value (respectively upper bound) for the abovementioned difference ε. Ignoring



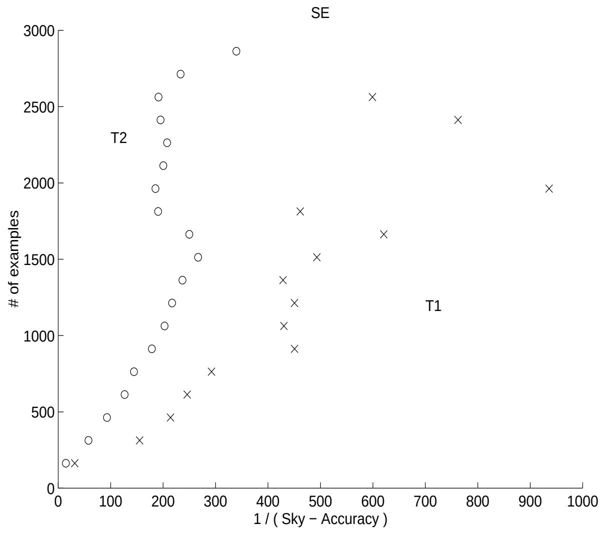


Figure 1: Learning curves for CR and SE: functional dependency of m on $1/\varepsilon$

log-factors and the dependency on the confidence parameter δ, these yield an upper bound $m = O\left(\frac{\text{VC-dim}(\mathcal{H})}{\varepsilon^2}\right)$ which holds for *any* distribution D [Tal94], and a lower bound $m = \Omega\left(\inf_{H\in\mathcal{H}} Err_D[H] \cdot \frac{\text{VC-dim}(\mathcal{H})}{\varepsilon^2}\right)$ which holds for *some* distribution D [DL95].

Unfortunately these upper and lower bounds differ by large constant factors, and hence one cannot predict from these bounds for concrete distributions D the actual functional dependency of m on ε. However these bounds *do predict* that m depends on $\mathcal{H}$ *only* through the VC-dimension of $\mathcal{H}$, and through $\inf_{H\in\mathcal{H}} Err_D[H]$ in the case of the lower bound. One easy way of testing this prediction (at least heuristically) is to estimate the size of the terms $\frac{\text{VC-dim(TREE(2))}}{\text{VC-dim(TREE(1))}}$, respectively $\frac{\inf_{H\in\text{ TREE(2)}} Err_D[H]\cdot\text{VC-dim(TREE(2))}}{\inf_{H\in\text{ TREE(1)}} Err_D[H]\cdot\text{VC-dim(TREE(1))}}$, by which these upper and lower bounds differ for the two considered hypothesis classes TREE(1) and TREE(2). Our bounds for the VC-dimension of decision trees indicate that for CR these terms have values between 17 and 25, and that for SE the first term has a value around 4 and the second term a value around 2. These theoretically predicted values match quite well the "factors" by which the curves for T1 and T2 differ in Figure 1.

With regard to the functional dependencies of m on ε the diagram for SE in Figure 1 suggests a growth that is consistent with the bounds with $m \approx 1/\varepsilon^2$. The diagram for CR rather suggests a functional dependence of the form $m \approx \log 1/\varepsilon$, that has previously already been theoretically predicted and experimentally verified for some *artificial* datasets ([TLS89],[SSSD90],[CT92],[HKST94]).

Altogether we believe that the possibility to compute with the help of our new algorithms T1 and T2 optimal hypothesis $H \in \mathcal{H}$ for arbitrary (even larger) datasets opens a new chapter in the experimental investigation of learning curves for "real-world" data.

5 CONCLUSION

In this paper we presented a novel algorithm, T2, which computes a decision tree of depth 2 or less in time $O(m \log m)$ which optimally fits a training set of size m. We derived bounds for the VC-dimension of decision trees of bounded depth, and proved that T2 is an agnostic PAC-learning algorithm. The latter guarantees that T2 will produce close to optimal (in respect to the distribution generating the examples) 2-level decision trees from sufficiently large training sets, for *any* distribution. These theoretical results were directly applied to several matters of genuine practical interest. First, T2 was experimentally evaluated as a learning algorithm by comparing its performance with that of C4.5 on 15 "real-world" datasets. The accuracy of T2's trees rivalled or surpassed C4.5's on 8 of the datasets, including all but one of the datasets having only continuous attributes. In many cases, C4.5's trees were considerably more complex than T2's. T2 was also evaluated as an algorithm for summarizing a dataset. On this task it performed extremely well. On average, T2's decision tree summarizes a dataset with 90% accuracy. Finally, T2, in combination with VC-dimension bounds we derived, provides us now for the first time with the tools necessary for comparing learning curves of decision trees for "real-world" datasets with the theoretical estimates of PAC-learning theory.

Acknowledgement

The first author was partially supported by grand J01028-MAT from the Fonds zur Förderung der wissenschaftlichen Forschung, Austria.

References

[AL88] D. Angluin, P. Laird, *Learning from noisy examples*, Machine Learning, vol. 2, 1988, 343 - 370.

[BB94] M. Bohanec, I. Bratko, *Trading Accuracy for Simplicity in Decision Trees*, Machine Learning, vol. 15, no. 3, 1994, 223 - 250.

[BEHW89] A. Blumer, A. Ehrenfeucht, D. Haussler, M. K. Warmuth, *Learnability and the Vapnik-Chervonenkis dimension*, JACM 36(4), 1989, 929 - 965.

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, *Classification and Regression Trees*, Wadsworth (Belmont, 1984).

[BN92] W. Buntine, T. Niblett, *A further comparison of splitting rules for decision-tree induction*, Machine Learning, vol. 8, 1992, 75 - 82.

[CT92] D. Cohn, G. Tesauro, *How tight are the Vapnik-Chervonenkis bounds*, Neural Computation 4, 1992, 249 - 269.

[Dec93] S. E. Decatur, *Statistical queries and faulty PAC oracles*, Proc. of the 6th ACM Conference on Computational Learning Theory, 1993, 262 - 268.

[DGM95] D. P. Dobkin, D. Gunopulos, W. Maass, *Computing the maximum bichromatic discrepancy, with applications in computer graphics and machine learning*, invited paper for a special issue of the Journal of Computer and System Sciences.

[DL95] L. Devroye, G. Lugosi, *Lower bounds in pattern recognition and learning*, appears in Pattern recognition.

[DP93] A. Danyluk, F. Provost, *Small Disjuncts in Action: Learning to Diagnose Errors in the Local Loop of the Telephone Network*, Proc. 10th International Conf. Machine Learning (ML'93), Morgan Kaufmann, 1993, 81 - 88.

[DSS93] H. Druker, R. Schapire, P. Simard, *Improving performance in neural networks using a boosting algorithm*, Advances in Neural Information Processing Systems, vol. 5, Morgan Kaufmann, 1993, 42-49.

[EK91] T. Elomaa, J. Kivinen, *Learning decision trees from noisy examples*, Report A-1991-3, Dept. of Computer Science, University of Helsinki (1991).

[Elo92] T. Elomaa, *A hybrid approach to decision tree learning*, Report C-1992-61, Dept. of Computer Science, University of Helsinki (1992).

[Elo94] T. Elomaa, *In defense of C4.5: Notes on learning one-level decision trees*, Proc. of the 11th Int. Conf. on Machine Learning, Morgan Kaufmann, 1994, 62 - 69.

[Hau92] D. Haussler, *Decision theoretic generalizations of the PAC-model for neural nets and other learning applications*, Inf. and Comp., vol. 100, 1992, 78 - 150.

[HKST94] D. Haussler, M. Kearns, H. S. Seung, N. Tishby, *Rigorous learning curve bounds from statistical mechanics*, Proc. of the 7th Annual ACM Conference on Computational Learning Theory 1994, ACM-Press (1994), 76 - 87.

[Hol93] R. C. Holte, *Very simple classification rules perform well on most commonly used datasets*, Machine Learning, vol. 11, 1993, 63 - 91.

[HSV93] K. U. Hoeffgen, H. U. Simon, K. S. Van Horn, *Robust trainability of single neurons*, preprint (1993)

[Kea93] M. Kearns, *Efficient noise-tolerant learning from statistical queries*, Proc. of the 25th ACM Symp. on the Theory of Computing, 1993, 392 - 401.

[KL93] M. Kearns, M. Li, *Learning in the presence of malicious errors*, SIAM J. Comput., vol. 22, 1993, 807 - 837.

[KSS92] M. J. Kearns, R. E. Schapire, L. M. Sellie, *Toward efficient agnostic learning*, Proc. of the 5th ACM Workshop on Computational Learning Theory, 1992, 341 - 352.

[Lub94] D. J. Lubinsky, *Bivariate Splits and Consistent Split Criteria in Dichotomous Classification Trees*, HD-Dissertation in Computer Science, Rutgers University (1994).

[Maa93] W. Maass, *Agnostic PAC-learning of functions on analog neural nets,* Advances in Neural Information Processing Systems, vol. 6, Morgan Kaufmann, 1994, 311-318; journal version to appear in Neural Computation.

[Maa94] W. Maass, *Efficient Agnostic PAC-Learning with Simple Hypotheses*, Proc. of the 7th Annual ACM Conference on Computational Learning Theory, 1994, 67-75.

[Min89] J. Mingers, *An empirical comparison of pruning methods for decision tree induction*, Machine Learning, vol. 4, 1989, 227 - 243.

[Qui92] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1992.

[SSSD90] D. B. Schwartz, V. K. Samalan, S. A. Solla, J. S. Denker, *Exhaustive Learning*, Neural Computation 2, 1990, 374 - 385.

[Tal94] M. Talagrand, *Sharper bounds for Gaussian and empirical processes*, Annals of Probability, vol. 22, 1994, 28–76.

[TLS89] N. Tishby, E. Lavin, S. A. Solla, *Consistent inference of probabilities in layered networks: Predictions and generalizations*, Proc. of IJCNN 1989, Vol. II, 403 - 409.

[Val84] L. G. Valiant, *A theory of the learnable*, Comm. of the ACM, vol. 27, 1984, 1134 - 1142.

[Vap82] V. N. Vapnik, *Estimations of Dependencies Based on Empirical Data*, Springer (New York, 1982).

[WGT90] S. M. Weiss, R. Galen, P. V. Tadepalli, *Maximizing the predictive value of production rules*, Art. Int., vol. 45, 1990, 47 - 71.

[WK90] S. M. Weiss, I. Kapouleas, *An empirical comparison of pattern recognition, neural nets, and machine learning classification methods*, Proc. of the 11th Int. Joint Conf. on Art. Int. 1990, Morgan Kaufmann, 781 - 787.

[WK91] S. M. Weiss, C. A. Kulikowski, *Computer Systems that Learn*, 1991, Morgan Kaufmann.