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

Peter Auer

Institute for Theoretical Computer Science
Technische Universitaet Graz
Klosterwiesgasse 32/2
A-8010 Graz, Austria
e-mail: pauer@igi.tu-graz.ac.at

Robert C. Holte

Computer Science Dept.
University of Ottawa
Ottawa, Canada K1N 6N5
e-mail: holte@csi.uottawa.ca

Wolfgang Maass

Institute for Theoretical Computer Science
Technische Universitaet Graz
Klosterwiesgasse 32/2
A-8010 Graz, Austria
e-mail: maass@igi.tu-graz.ac.at

Abstract

We exhibit a new algorithm $T2$ for agnostic PAC-learning with 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 datasets, and show that for most of these datasets $T2$ provides substantially simpler decision trees with little or no loss in predictive power. In fact, for datasets with continuous attributes its error rate tends to be lower than that of $C4.5$. Furthermore, since one can *prove* that $T2$ is an angostic 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. To the best of our knowledge this is in fact the first time that a PAC-learning algorithm is shown to be applicable to "real-world" classification problems.

We also demonstrate that this algorithm $T2$ can be used as a diagnostic tool for the computation of learning curves for common "real-world" datasets, and for the investigation of the expressive limits of 2-level decision trees. In combination with the new bounds for the VC-dimension of decision trees of bounded depth that we prove, these new algorithmic tools allow us to "place PAC-learning theory on a test stand" and compare its theoretical estimates for learning curves with our experimental results for "real-world" classification problems.

1 Introduction and Basic Definitions

Numerous articles have been written about the design and analysis of algorithms for PAC-learning, since Valiant [V] had introduced the model for probably approximately correct 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 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. [DSS]).

In applied machine learning one views various concrete datasets from quite diverse application domains as prototypes for "real-world" classification problems. Most of these datasets have been made publicly available through the University of California at Irvine (?? ftp - address ?). The performance of many practical learning algorithms on these datasets is described in a number of interesting comparative studies (see e.g. [Mi], [WGT], [WK 90], [WK 91], [BN], [Ho]). Each of these datasets S is a list of a certain number (typically between a few dozen and several thousand) of items $x \in X_n \times \{1, \ldots, p\}$. Each item x in such dataset S consists of a fixed number n of values $x_1, \ldots, x_n$ for n "attributes" of x (typically $n < 40$), and an associated "classification" $x_0 \in \{1, \ldots, p\}$ for some fixed number p of "classes" (usually $p = 2$). The individual attributes $\alpha \in \{1, \ldots, n\}$ may be either "*continuous*" (i.e. $x_\alpha \in \mathbf{R} \cup \{missing\}$), or "*categorical*" (i.e. $x_\alpha \in \{1, \ldots, b\} \cup \{missing\}$ for some fixed $b \in \mathbf{N}$; usually $b \leq 6$). Thus formally the attribute vectors $\langle x_1, \ldots, x_n \rangle$ for items x in S are elements of some domain $X_n \subseteq (\mathbf{R} \cup \{missing\})^{n_1} \times (\{1, \ldots, b\} \cup \{missing\})^{n_2}$ for certain natural numbers n_1, n_2 with $n_1 + n_2 = n$. The situation that items have missing attribute values has to be taken into account, since about half of the datasets in the UC Irvine collection contain items with missing attribute values.

A *learning algorithm* A computes for any given list S_{train} of items from $X_n \times \{1, \ldots, p\}$ a *hypothesis* $A(S_{\text{train}})$, which is formally a function from X_n into $\{1, \ldots, p\}$. We refer to the class $\mathcal{H}_n$ of all hypotheses that may potentially occur as $A(S_{\text{train}})$ for a training sequence S_{train} as the *hypothesis space* of the learning algorithm A.

The error-rate of any hypothesis $H : X_n \to \{1, \ldots, c\}$ on a list $S_{\text{test}} = (\langle x_1^i, \ldots, x_n^i, x_0^i \rangle)_{1 \leq i \leq t}$ of t items from $X_n \times \{1, \ldots, c\}$ is defined as

$$Err_{S_{\text{test}}}[H] := \frac{|\{i \in \{1, \ldots, t\} : H(x_1^i, \ldots, x_n^i) \neq x_0^i\}|}{t} \quad .$$

In order to evaluate the performance of a concrete learning algorithm A on a concrete dataset S one usually applies crossvalidation. For that purpose one partitions S randomly into N pieces of about equal size (in our experiments $N = 25$). 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.

Of course the real goal of a learning algorithm A is to provide correct classifications (i.e. $A(S_{\text{train}})(x_1, \ldots, x_n) = x_0$) for *new* items $x = \langle x_1, \ldots, x_n, x_0 \rangle$ that are produced by the same *distribution* D as the training set S_{train}. From this point of view one can consider a concrete finite dataset S with m items as a result of m random drawings according to this distribution D. In this interpretation one may view $Err_{S_{\text{test}}}[H]$ for any sequence S_{test} that consists of randomly drawn items from the dataset S as an *estimate* for the *true error* $Err_D[H] := Pr_{\langle x_1, \ldots, x_n, x_0 \rangle \in D}[H(x_1 \ldots, x_n) \neq x_0]$ of a hypothesis $H : X_n \to \{1, \ldots, c\}$ (with regard to the underlying distribution D). The goal of the previously described experimental protocol for evaluating a learning algorithm A is to get an estimate for the true error $Err_D[A(S_{\text{train}})]$ of A for a sequence S_{train} of m items that are drawn independently according to the underlying distribution D. 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.

Both in practical and in theoretical approaches one evaluates the performance of a learning algorithm A by measuring (respectively by estimating) the difference between $Err_D[A(S_{\text{train}})]$ and the true error $\inf_{H \in \mathcal{H}_n} Err_D[H]$ of the *best possible* hypothesis H from the hypothesis class $\mathcal{H}_n$ from which A chooses its hypotheses. 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 $X_n \times \{1, \ldots, c\}$, and any sequence S_{train} of $\geq m(1/\varepsilon, 1/\delta, n)$ items drawn according to D

$$\left| Err_D[A(S_{\text{train}})] - \inf_{H \in \mathcal{H}_n} Err_D[H] \right| \leq \varepsilon$$

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 of the representation of S_{train}.

The here defined version of PAC-learning (due to [H] and [KSS]) is usually referred to as *agnostic PAC-learning*, in contrast to the original PAC-learning model (due to Valiant [V]) where one focuses on hypothesis classes $\mathcal{H}_n$ and distributions D such that $\inf_{H \in \mathcal{H}_n} Err_D[H] = 0$. More precisely, in this original model one assumes that the distribution D over $X_n \times \{1, \ldots, p\}$ is generated by some "target concept" $C_t \in \mathcal{H}_n$ together with an arbitrary distribution over X_n. In order to make this original version better applicable to "real-world" learning problems, one has also

designed PAC-learning algorithms for a more general scenario (see e.g. [AL], [EK], [E 92], [K], [KL], [D]) where 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" (i.e. not systematic) noise does not model the situation that one encounters in the here considered "real-world" classification problems S. On the other hand, in the model with malicious noise the PAC-learner can only achieve error rates that are intolerable 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 exist two remarkable *negative* results about agnostic PAC-learning. It has been shown that neither for $\mathcal{H}_n = \{\text{halfspaces over } \mathbf{R}^n\}$ [HSV] nor for $\mathcal{H}_n = \{\text{monomials over } n \text{ boolean attributes}\}$ [KSS] there exists an efficient agnostic PAC-learning algorithm (unless $R = NP$). These negative results are quite disappointing, since in learning theory one usually views these two hypothesis classes as the "simplest" nontrivial hypothesis classes for continuous respectively boolean attributes. On the other hand one *did* suceed in designing efficient agnostic PAC-learning algorithms for a few relatively *small* hypothesis classes $\mathcal{H}_n$ ([KSS], [M 93], [M 94], [DGM]). However these classes $\mathcal{H}_n$ appear to be 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 these classes is for common datasets S substantially larger than for those hypothesis classes $\mathcal{H}_n$ that are usually employed in applied machine learning.

In this article we present an *efficient* agnostic PAC-learning algorithm $T2$ for the more expressive hypothesis class $\mathcal{H}_n = TREE(2, n, p, K)$, where K is some arbitrary fixed parameter (with default value $K := p + 1$). The results of our experiments demonstrate that this learning algorithm $T2$ is of some interest from the point of view of applications, since $\inf_{H \in TREE(2,n,p,K)} Err_S[H]$ is relatively small for most commonly considered datasets S.

The hypothesis class $TREE(2, n, p, K)$ consists of all functions $f : X_n \to \{1, \ldots, p\}$ 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 α with $b(\alpha) \leq b$ possible values (in which case $b(\alpha) + 1$ edges leave the root of T, labeled by $1, \ldots, b(\alpha), missing$), or it queries a continuous attribute α (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). 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 β ($\beta = \alpha$ is also allowed). If β is a categorical attribute, the $b(\beta)$ edges with labels $1 \ldots, b(\beta), missing$, leave the node ν. If β 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. 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 $f : X_n \to \{1, \ldots, p\}$ 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 $1R$ [Ho]. In our experiments we have always chosen $K := p+1$. Note that we always identify a decision tree T with the function $f : X_n \to \{1, \ldots, p\}$ that is computed by T.

An essential 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 (with high probability) from any list S_{train} of examples that are drawn according to D a hypothesis $A(S_{\text{train}})$ whose true error is arbitrarily close to the *least true error* of *any* hypothesis from the associated hypothesis space $\mathcal{H}_n$ (if S_{train} is sufficiently large relative to the VC-dimension of $\mathcal{H}_n$). 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 learning algorithm will perform well for a particular dataset. In fact, even for extremely successful practical learning algorithms such as C4.5 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.

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 $TREE(2, p, n, K)$). 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 assignment of labels to the edges in T. In

particular $T2$ 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 of 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 [M 94], [DGM] 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 a numerical 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 $\leq k$ 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 numerical 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 $O(m \log n)$ algorithm). This combination of 2 simultaneous search precedures 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 continu-

ous attributes $\alpha, \beta \in \{1, \ldots, n\}$ have been fixed. $T2$ computes for various lists $\check{L}$ 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 $\text{OPTSPLIT}^{\alpha}_{k, c_{\text{left}}, c_{\text{right}}}(\check{L})$ 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 $\check{L}$ for which x_0 differs from the classification $C(I_i)$ of the interval I_i with $x_\alpha \in I_i$. Formally $\text{OPTSPLIT}^{\alpha}_{k, c_{\text{left}}, c_{\text{right}}}(\check{L})$ 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 $x \in \check{L}$ that result from this split. The second considered attribute β determines for which sublists $\check{L}$ of L the preceding data are assembled. All sublists $\check{L}$ are of the form $\check{L} := \langle x \in L : x_\alpha \in I \text{ and } x_\beta < y \rangle$ or $\check{L} := \langle x \in L : x_\alpha \in I \text{ and } x_\beta \geq y \rangle$ for some interval I and some $y \in \mathbf{R}$. We will focus on the handling of lists $\check{L}$ of the first type, since the handling of lists $\check{L}$ of the second type is analogous. Thus we consider arrays of the form

$$V_I^{\alpha,\beta}(y) := \Big\langle \text{OPTSPLIT}^{\alpha}_{k, c_{\text{left}}, c_{\text{right}}}(\langle x \in L : x_\alpha \in I \text{ and } x_\beta < y \rangle) \Big\rangle_{\substack{k \in \{1, \ldots, K\} \\ c_{\text{left}}, c_{\text{right}} \in \{1, \ldots, p\}}},$$

and we write $V_I^{\alpha,\beta}$ for the list of all $V_I^{\alpha,\beta}(y)$ such that $y = \check{x}_\beta$ for some $\check{x} \in L$ with $\check{x}_\alpha \in I$, 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 $y = \check{x}_\beta$ for some $\check{x} \in L$ with $\check{x}_\alpha \in I \cup I'$. In the case where $\check{x}_\alpha \in I$, 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 $\check{x}_\alpha \in I'$, 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 m intervals I, such that each I contains the value x_α of the attribute α for at most one 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 assignment of labels to edges and leaves for a subtree

of a decision tree from $TREE(2,p,n,K)$ 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 $\min(m, b+1)$ values of a categorical attribute. Thus we have proven the following result.

Theorem 1: *The algorithm $T2$ computes for any $n_1, n_2 \in \boldsymbol{N}$ with $n_1 + n_2 = n$, any $K \in \boldsymbol{N}$, and any list L of m items from $(\boldsymbol{R} \cup \{missing\})^{n_1} \times \{1, \ldots, b, missing\}^{n_2} \times \{1, \ldots, p\}$ in $O(K^2 \cdot n^2 \cdot m \log m)$ computation steps a decision tree from $TREE(2,p,n,K)$ that misclassifies a minimal number of items in L.*

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

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 [H] 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 [L]. In particular it can be used to minimize the total cost of all misclassifications for any given "confusion matrix" [WK 91]. 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 [L] has shown that this split criterion "wacc" performs for many datasets as well as "Gini" [BFOS], 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 $n_1, n_2 \in \boldsymbol{N}$ with $n_1 + n_2 \in n$, any $K \in \boldsymbol{N}$, and any given list L of m items from $\boldsymbol{R} \cup \{missing\})^{n_1} \times \{1, \ldots, b, missing\}^{n_2} \times \{1, \ldots, p\}$ a tree $T \in TREE(d,n,p,K)$ that makes a min-*

imal number of misclassifications of items in L. Hence Td is an algorithm for efficient agnostic PAC-learning with hypothesis class $TREE(d, n, p, K)$.

3 Evaluation of $T2$ on "Real-World" Classification Problems

We compare the performance of $T2$ on various common "real-world" classification problems with that of $C4.5$ ([Q]), a state-of-the-art heuristic machine learning algorithm for building (and pruning) decision trees of unbounded depth. Furthermore we compare the performance of $T2$ with that of $1R$ ([H]), another algorithm that like $T2$ (and unlike $C4.5$) always outputs *simple* hypotheses. The datasets for which we have carried out these tests are listed in table 1.

				Attributes ... number of distinct values							
Dataset	Size	Baseline accuracy	Missing values	cont	2	3	4	5	6	>6	Total
AP	106	80.2	yes	8							8
BC	286	70.3	yes		3	2		1	1	2	9
CH	3196	52.2	no		35	1					36
CR	690	55.5	yes	6	4	2	1			2	15
G2	163	53.4	no	9							9
HD	303	54.5	yes	5	3	3	2				13
HE	155	79.4	yes	6	13						19
IO	351	64.1	no	34							34
IR (3)	150	33.3	no	4							4
LA	57	64.9	yes	8	3	5					16
PI	768	65.1	no	8							8
PR	106	50.0	no				58				58
SE	3163	90.7	yes	7	18						25
SO (4)	47	36.2	no		13	3	4			1	35
SP (3)	3190	51.9	yes				60				60

Table 1: *Datasets used in the experiments*

The error-rates for $T2$ and $C4.5$ in this table resulted from 25-fold cross validation. For a few smaller datasets (??) we have found that this error rate varied by several percentage points for repetitions of this experiment, hence we give for these the *average* of the error rates from (??) repetitions of 25-fold cross validation. For $1R$ we quote in this table the error rates that were already reported in [H], which resulted from 25 random partitions of the dataset into 1/3 for testing and 2/3 for training.

In order to get an estimate for the complexity of the decision trees that were produced by $C4.5$ in these experiments, we have also recorded the maximal depth mx

(= maximal number of attributes queried on a branch) of the resulting decision tree, the average number of attributes queried in that tree (dc = "dynamic complixity"), and the percentage of items for which more than 2 attributes were queried in that tree (these data are also averages that resulted from 25-fold crossevalidation, like the error-rates).

	Datasets														
	AP	BC	CH	CR	G2	HD	HE	IO	IR	LA	PI	PR	SE	SO	SP
1R	84.2	65.6	66.5	85.5	72.5	72.9	81.9	79.9	92.9	70.6	71.3	65.1	95.0	86.7	63.3
T2	88.6	66.3	86.9	84.2	79.7	67.1	78.6	86.1	95.7	86.6	74.8	69.3	97.3	91.1	79.4
C4.5	85.1	75.2	99.2	84.9	76.5	74.5	80.2	94.3	94.1	86.1	73.4	76.6	98.0	97.8	94.4
d.c.	1.93	1.22	4.56	2.85	4.37	3.22	1.91	4.66	2.00	1.86	6.77	1.65	1.56	1.81	3.55
% > 2	1	1	58	33	90	83	10	76	34	17	78	10	13	2	68
mx	3	4	13	18	10	14	5	11	4	3	33	3	8	3	8

Table 2: *Experimental comparison of* $1R$, $T2$, *and* $C4.5$.

Table 2 shows that the error rate of $T2$ is consistently higher than that of $C4.5$ for those datasets that have only categorical attributes (BL, CH, PR, SO, SP), that the error rate of $T2$ is close (but in 4 cases a bit higher) than that of $C4.5$ on the 5 datasets that have both continuous and categorical attributes (CR, HD, HE, LA, SE), and that $T2$ has a lower error rate than $C4.5$ for 4 of the 5 datasets that only have continuous attributes (AP, $G2$, IO, IR , PI). This suggests that $T2$ may be preferrable over $C4.5$ for datasets that have only continuous attributes.

If one also takes the interpretability of the learning output for the human user into account (for example if the user wants to get heuristic insight into the *structure* of a real-world classification problem), then $T2$ has an edge over $C4.5$ in 8 of the 15 datasets, since the output trees of $C4.5$ (*after* pruning) are consistantly more complex than those of $T2$ (see the last 3 rows in Table 2). Even for datasets such as CR and HD (which we have counted as "losses" for $T2$ in the preceding account) the human user might want to see besides the decision trees with branches of length respectively ?? that ?? $C4.5$ outputs, also the *substantially* simpler decision trees that are produced by $T2$.

Finally 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 uncomparable 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.

4 PAC-learning Theory on the Teststand

At the heart of learning theory are certain statistical results (due to [Vap], [BEHW], [De], [H], [DL] and others) that provide for *any* distribution D of the data an up-

per bound for the difference between the true error $Err_D[\hat{H}]$ of the hypothesis $\hat{H} \in \mathcal{H}$ with the least empirical error on the training set S_{train} (i.e. $\hat{H}$ minimizes $Err_{S_{\text{train}}}[H]$), and the least true error $\inf_{H \in \mathcal{H}} Err_D[H]$ on *any* hypothesis in $\mathcal{H}$. These theoretical results are essential for the design of algorithms in PAC-learning theory, since they show that it suffices to minimize the error on the training set. With the help of our new algorithms T_1 and T_2 we can now *compute* for nontrivial hypothesis classes $\mathcal{H}$ the hypothesis $\hat{H} \in \mathcal{H}$ with the least empirical error on a training set S_{train}, and 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, since the abovementioned estimates 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*.

The abovementioned estimates are given in terms the Vapnik-Chervonenkis dimension (VC-dimension) of the considered hypothesis class $\mathcal{H}$. Since so far no significant bounds for the VC-dimension of decision trees have been available, we porvide them in the following Theorem 4. Its proof requires nontrivial combinatorial arguments.

Theorem 4: *The VC-dimension of* $TREE(d, n, 2, K)$ *is* $\Theta(\log n)$ *for any fixed* d, b, K. *More precise bounds depend on the specific structure of the trees (respectively on the relative sizes of* $d, b, n, K,$*). For all 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* $((b+1^d - 1)\log n + K \log\log n + O(1)$.■

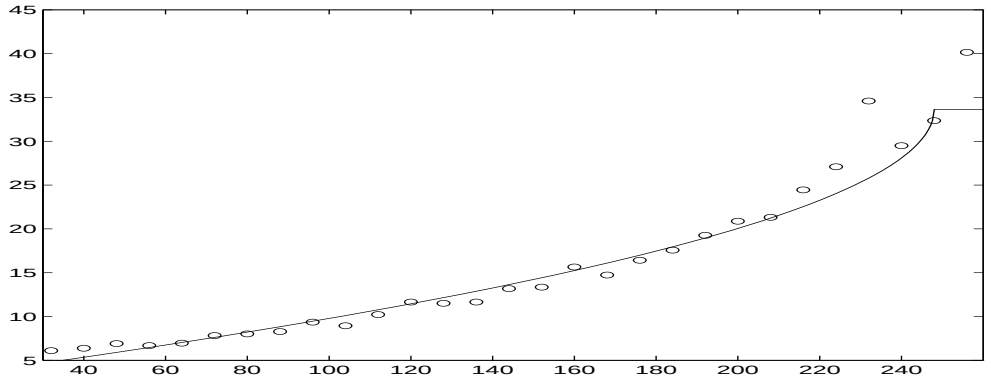

Figure 1:

5 Application of T_1 and T_2 as Diagnostic Tools

An intriguing question regarding learning with algorithms that focus on finding *simple* hypotheses with good (or at least significant) predictive power, is whether common "real-world" classification problems S can in fact be characterized sufficiently well by *any* simple hypothesis of the considered type. 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 [WGT] for the case of production rules of length ≤ 3 on the dataset AP). It should be noted here that (according to their definitions) the measures $1R^*$ and $1Rw$ in [Ho] do *not* give the performance of the best 1-rule on the there considered datasets. However with the help of the algorithms Td from section 2 this is now feasible for the hypothesis spaces $TREE(d, n, p, k)$ for small values of d. The results for $d = 1$ and $d = 2$ (denoted by "sky1" and "sky2") for the here considered 15 datasets S are given in Table 3. One can read off various interesting facts from this table.

(1) Although $1R$ is a *greedy* algorithm (which may perform arbitrarily bad on a synthetic dataset), it manages to find for *all (?)* of the here considered datasets *almost optimal* hypotheses from the hypothesis space $TREE(1, n, p, p+1)$. We find this empirical fact quite surprising (a similar phenomenon is known in combinatorial optimization, where the greedy simplex algorithm is known to work very well in practise, although its worst case performace is very bad). This fact suggests that many (most ?) common datasets S have special properties, that distinguish them from "worst case" datasets.

(2) For all except two (CH and SP) of the considered 15 datasets there exists a rather small set of attributes (those queried by the optimal tree from $TREE(2, n, p, p+1)$) which contains almost all information needed for correct classification. Furthermore for 11 of the 15 datasets S there *exist* 2-level decision trees that have a substantially lower error rate on S than the more complex decision trees that $C4.5$ has found (in 25 fold crossvalidation). For two other datasets, SE and IO, the error rates of the more complex output trees of $C4.5$ are insignificant lower, and only for CH and SP (2 datasets that have only categorical attributes) they are substantially lower. This suggests that there exists, especially for domains that have only continuous attributes, substantial room for designing learning algorithms that output simpler hypotheses *and* achieve lower error rates.

	Datasets														
	AP	BC	CH	CR	G2	HD	HE	IO	IR	LA	PI	PR	SE	SO	SP
1R	84.2	65.6	66.5	85.5	72.5	72.9	81.9	79.9	92.9	70.6	71.3	65.1	95.0	86.7	63.3
T1	80.0	67.2	66.1	85.5	76.2	70.9	79.2	78.3	91.9	71.6	73.6	66.3	95.0	85.3	63.2
Sky 1	89.6	72.7	68.3	85.5	78.5	76.6	85.2	83.8	96.0	84.2	75.0	80.2	95.0	87.2	63.4
T2	88.6	66.3	86.9	84.2	79.7	67.1	78.6	86.1	95.7	86.6	74.8	69.3	97.3	91.1	79.4
C4.5	85.1	75.2	99.2	84.9	76.5	74.6	80.2	94.3	94.1	86.1	73.4	76.6	98.0	97.8	94.4
Sky 2	96.2	79.4	86.9	87.7	87.7	82.5	91.0	92.9	98.7	98.2	78.0	92.5	97.6	100.	79.6

Table 3:

References

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

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

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

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

[DGM] 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.

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

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

[E 92] T. Elomaa, *A hybrid approach to decision tree learning*, Report C-1992-61, Dept. of Computer Science, University of Helsinki (1992).

[E 94] 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 (San Mateo, 1994), 62 - 69.

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

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

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

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

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

[KSS] 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.

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

[M 93] W. Maass, *Agnostic PAC-learning of functions on analog neural nets,* Advances in Neural Information Processing Systems, vol. 6, Morgan Kaufmann (San Mateo, 1994), 311-318; journal version to appear in Neural Computation.

[M 94] W. Maass, *Efficient Agnostic PAC-Learning with Simple Hypotheses*, Proc. of the 7th Annual ACM Conference on Computational Learning Theory, 1994, 67-75.

[Mi] J. Mingers, *An empirical comparision of pruning methodes for decision tree induction*, Machine Learning, vol. 4, 1989, 227 - 243.

[Q2] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1992.

[T] M. Talagrand, *Sharper bounds for empirical processes*, to appear in Annals of Probability and its Applications.

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

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

[WK 90] 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.

[WK 91] S. M. Weiss, C. A. Kulikowski, *Computer Systems that Learn*, 1991, Morgan Kaufmann.