Decision Trees: More Theoretical Justification for Practical Algorithms

Amos Fiat and Dmitry Pechyony*

School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel
{fiat,pechyony}@tau.ac.il

Abstract. We study impurity-based decision tree algorithms such as CART, C4.5, *etc.*, so as to better understand their theoretical underpinnings. We consider such algorithms on special forms of functions and distributions. We deal with the uniform distribution and functions that can be described as a boolean linear threshold functions or a read-once DNF.

We show that for boolean linear threshold functions and read-once DNF, maximal purity gain and maximal influence are logically equivalent. This leads us to the exact identification of these classes of functions by impurity-based algorithms given sufficiently many noise-free examples. We show that the decision tree resulting from these algorithms has minimal size and height amongst all decision trees representing the function.

Based on the statistical query learning model, we introduce the noise-tolerant version of practical decision tree algorithms. We show that if the input examples have small classification noise and are uniformly distributed, then all our results for practical noise-free impurity-based algorithms also hold for their noise-tolerant version.

* Dmitry Pechyony is a full-time student and thus this paper is eligible for the "Best Student Paper" award according to conference regulations.

1 Introduction

Introduced in 1983 by Breiman *et al.* [4], decision trees are one of the few knowledge representation schemes which are easily interpreted and may be inferred by very simple learning algorithms. The practical usage of decision trees is enormous (see [24] for a detailed survey). The most popular practical decision tree algorithms are CART ([4]), C4.5 ([25]) and their various modifications. The heart of these algorithms is the choice of splitting variables according to maximal purity gain value. To compute this value these algorithms use various impurity functions. For example, CART employs the Gini index impurity function and C4.5 uses an impurity function based on entropy. We refer to this family of algorithms as "impurity-based".

Despite practical success, most commonly used algorithms and systems for building decision trees lack strong theoretical basis. It would be interesting to obtain the bounds on the generalization errors and on the size of decision trees resulting from these algorithms given some predefined number of examples.

There have been several results justifying theoretically practical decision tree building algorithms. Kearns and Mansour showed in [18] that if the function, used for labelling nodes of tree, is a weak approximator of the target function then the impurity-based algorithms for building decision tree using Gini index, entropy or the new index are boosting algorithms. This property ensures distribution-free PAC learning and arbitrary small generalization error given sufficiently input examples. This work was recently extended by Takimoto and Maruoka [26] for functions having more than two values and by Kalai and Servedio [16] for noisy examples.

We restrict ourselves to the input of uniformly distributed examples. We provide new insight into practical impurity-based decision tree algorithms by showing that for unate boolean functions, the choice of splitting variable according to maximal exact purity gain is equivalent to the choice of variable according to the maximal influence. Then we introduce the algorithm **DTExactPG**, which is a modification of impurity-based algorithms that uses exact probabilities and purity gain rather that estimates. Let $f(x)$ be a read-once DNF or a boolean linear threshold function and let h be the minimal depth of decision tree representing $f(x)$. The main results of our work are:

Theorem 1 *The algorithm* ***DTExactPG*** *builds a decision tree representing $f(x)$ and having minimal size amongst all decision trees representing $f(x)$. The resulting tree has also minimal height amongst all decision trees representing $f(x)$.*

Theorem 2 *For any $\delta > 0$, given $O\big(2^{9h} \ln^2 \frac{1}{\delta}\big) = poly(2^h, \ln \frac{1}{\delta})$ uniformly distributed noise-free random examples of $f(x)$, with probability at least $1-\delta$, CART and C4.5 build a decision tree computing $f(x)$ exactly. The resulting tree has minimal size and minimal height amongst all decision trees representing $f(x)$.*

Theorem 3 *For any $\delta > 0$, given $O\big(2^{9h} \ln^2 \frac{1}{\delta}\big) = poly(2^h, \ln \frac{1}{\delta})$ uniformly distributed random examples of $f(x)$ corrupted by classification noise with constant*

Function	Exact Influence	Exact Purity Gain	CART, C4.5, *etc.* $poly(2^h)$ uniform noise-free examples	Modification of CART, C4.5, *etc.*, $poly(2^h)$ uniform examples with small classification noise
Boolean LTF	min size min height	min size min height	min size min height	min size min height
Read-once DNF	min size min height	min size min height	min size min height	min size min height

Fig. 1. Summary of bounds on decision trees, obtained in our work.

Algorithm	Model, Distribution	Running Time	Hypothesis	Bounds on the Size of DT	Function Learned
Jackson and Servedio [15]	PAC, uniform	$poly(2^h)$	Decision Tree	none	almost any DNF
Impurity-Based Algorithms (Kearns and Mansour [18])	PAC, any	$poly((\frac{1}{\epsilon})^{\frac{c}{\gamma^2}})$	Decision Tree	none	any function satisfying Weak Hypothesis Assumption
Bshouty and Burroughs [5]	PAC, any	$poly(2^n)$	Decision Tree	at most min-sized DT representing the function	any
Kushilevitz and Mansour [21], Bshouty and Feldman [6], Bshouty *et al.* [7]	PAC, examples from uniform random walk	$poly(2^h)$	Fourier Series	N/A	any
Impurity-Based Algorithms (our work)	PC (exact, identification), uniform	$poly(2^h)$	Decision Tree	minimal size, minimal height	read-once DNF, boolean LTF

Fig. 2. Summary of decision tree noise-free learning algorithms.

rate $\eta < 0.5$, with probability at least $1 - \delta$, a noise-tolerant version of impurity-based algorithms builds a decision tree representing $f(x)$. The resulting tree has minimal size and minimal height amongst all decision trees representing $f(x)$.

Fig. 1 summarizes the bounds on the size and height of decision trees, obtained in our work.

1.1 Previous Work

Building in polynomial time a decision tree of minimal height or with a minimal number of nodes, consistent with all examples given, is NP-hard ([14]). The single polynomial-time deterministic approximation algorithm, known today, for approximating the height of decision trees is the simple greedy algorithm ([23]), achieving the factor $O(\ln(m))$ (m is the number of input examples). Combining the results of [12] and [10], it can be shown that the depth of decision tree cannot be approximated within a factor $(1-\epsilon)\ln(m)$ unless $NP \subseteq DTIME(n^{O(\log\log(n))})$. Hancock *et al.* showed in [11] that the problem of building a decision tree with a minimal number of nodes cannot be approximated within a factor $2^{\log^{\delta} \mathrm{OPT}}$ for any $\delta < 1$, unless $\mathrm{NP} \subset \mathrm{RTIME}[2^{poly \log n}]$. Blum *et al.* showed at [3] that decision trees cannot even be weakly learned in polynomial time from statistical queries dealing with uniformly distributed examples. This result is an evidence for the difficulty of PAC learning of decision trees of arbitrary functions in the noise-free and noisy settings.

Fig. 2 summarizes the best results obtained by theoretical algorithms for learning decision trees from noise-free examples. Many of them may be modified, to obtain corresponding noise-tolerant versions.

Kearns and Valiant ([19]) proved that distribution-free weak learning of read-once DNF using any representation is equivalent to several cryptographic problems widely believed to be hard. Mansour and Schain give in [22] an algorithm for proper PAC-learning of read-once DNF in polynomial time from random examples taken from any maximum entropy distribution. This algorithm may be easily modified to obtain polynomial-time probably correct learning in case the underlying function has a decision tree of logarithmic depth and input examples are uniformly distributed, matching the performance of our algorithm in this case. Using both membership and equivalence queries Angluin *et al.* showed in [1] the polynomial-time algorithm for exact identification of read-once DNF by read-once DNF using examples taken from any distribution.

Boolean linear threshold functions are polynomially properly PAC learnable from both noise-free examples (folk result) and examples with small classification noise ([9]). In both cases the examples may be taken from any distribution.

1.2 Structure of the Paper

In Section 2 we give relevant definitions. In Section 3 we introduce a new algorithm **DTInfluence** for building decision trees using an oracle for influence and prove several properties of the resulting decision trees. In Section 4 we prove Theorem 1. In Section 5 we prove Theorem 2. In Section 6 we introduce the noise-tolerant version of impurity-based algorithms and prove Theorem 3. In Section 7 we outline directions for further research.

2 Background

In this paper we use standard definitions of PAC ([27]) and statistical query ([17]) learning models. All our results are in the PAC model with zero generalization error. We denote this model by PC (Probably Correct).

A boolean function (*concept*) is defined as $f : \{0,1\}^n \rightarrow \{0,1\}$ (for boolean formulas, *e.g.* read-once DNF) or as $f : \{-1,1\}^n \rightarrow \{0,1\}$ (for arithmetic formulas, *e.g.* boolean linear threshold functions). Let x_i be the i-th variable or *attribute*. Let $x = (x_1,\ldots,x_n)$, and $f(x)$ be the *target* or *classification* of x. The vector $(x_1, x_2,\ldots,x_n, f(x))$, is called an *example*. Let $f_{x_i=a}(x)$, $a \in \{0,1\}$ be

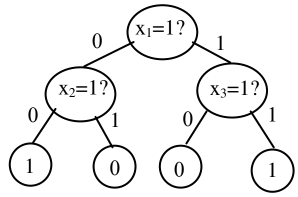

Fig. 3. Example of the decision tree representing $f(x) = x_1x_3 \vee \overline{x_1x_2}$

the function $f(x)$ restricted to $x_i = a$. We refer to the assignment $x_i = a$ as a *restriction*. Given the set of restrictions $R = \{x_{i_1} = a_1, \ldots, x_{i_k} = a_k\}$, the restricted function $f_R(x)$ is defined similarly. $x_i \in R$ iff there exists a restriction $x_i = a \in R$, where a is any value.

A *literal* $\tilde{x}_i$ is a boolean variable x_i itself or its negation $\bar{x}_i$. A *term* is a conjunction of literals and a *DNF (Disjunctive Normal Form) formula* is a disjunction of terms. Let $|F|$ be the number of terms in the DNF formula F and $|t_i|$ be the number of literals in the term t_i. Essentially F is a set of terms $F = \{t_1, \ldots, t_{|F|}\}$ and t_i is a set of literals, $t_i = \{\tilde{x}_{i_1}, \ldots, \tilde{x}_{i_{|t_i|}}\}$. The term t_i is *satisfied* iff $\tilde{x}_{i_1} = \ldots = \tilde{x}_{i_{|t_i|}} = 1$.

Given literal $\tilde{x}_i$ contained in term t_j we say that the value v of x_i *agrees with* t_j if setting x_i to v does not set t_j to be false. If setting x_i to v implies t_j is false then we say that the value v of x_i *disagrees with* t_j.

If for all $1 \leq i \leq n$, $f(x)$ is monotone w.r.t. x_i or $\overline{x}_i$ then $f(x)$ is a *unate function*. A DNF is *read-once* if each variable appears at most once. Given a weight vector $\overrightarrow{a} = (a_1, \ldots, a_n)$, such that for all $1 \leq i \leq n$, $a_i \in \Re$, and a threshold $t \in \Re$, the boolean linear threshold function (LTF) $f_{a,t}$ is $f_{a,t}(x) = \sum_{i=1}^{n} a_i x_i > t$. Clearly, both read-once DNF and boolean linear threshold function are unate functions.

Let e_i be the vector of n components, containing 1 in the i-th component and 0 in all other components. The *influence* of x_i on $f(x)$ under distribution $\mathcal{D}$ is $I_f(i) = \Pr_{x \sim \mathcal{D}}[f(x) \neq f(x \oplus e_i)]$. We use the notion of *influence oracle* as an auxiliary tool. The influence oracle runs in time $O(1)$ and returns the exact value of $I_f(i)$ for any f and i.

In our work we restrict ourselves to binary univariate decision trees for boolean functions. We use the standard definitions of *inner nodes*, *leaves* and *splitting variables* (see [25]). The left (right) son of inner node s is also called the 0-son (1-son) and is referred to as s_0 (s_1). Let $c(l)$ be the label of the leaf l. Upon arriving to the node s, we pass the input x to the ($x_i = 1$?)-son of s. The classification given to the input x by the T is denoted by $c_T(x)$. The path from the root to the node s corresponds to the set of restrictions of values of variables leading to s. Similarly, the node s corresponds to the restricted function $f_R(x)$. In the sequel we use the identifier s of the node and its corresponding restricted function interchangeably.

The *height* of T, $h(T)$, is the maximal length of path from the root to any node. The *size* of T, $|T|$, is the number of nodes in T. A decision tree T *represents* $f(x)$ iff $f(x) = c_T(x)$ for all x. An example of a decision tree is shown in Fig. 3.

The function $\phi(x) : [0,1] \rightarrow \Re$ is an *impurity function* if it is concave, $\phi(x) = \phi(1-x)$ for any $x \in [0,1]$ and $\phi(0) = \phi(1) = 0$. Examples of impurity functions

DTApproxPG(s, X, R, ϕ)
1: **if** all examples arriving at s have the same classification **then**
2: Set s as a leaf with that value.
3: **else**
4: Choose $x_i = \arg\max_{x_i \in X}\{\widehat{PG}(f_R, x_i, \phi)\}$ to be a splitting variable.
5: Run **DTApproxPG**(s_1, $X - \{x_i\}$, $R \cup \{x_i = 1\}$, ϕ).
6: Run **DTApproxPG**(s_0, $X - \{x_i\}$, $R \cup \{x_i = 0\}$, ϕ).
7: **end if**

Fig. 4. DTApproxPG algorithm - generic structure of all impurity-based algorithms.

are the Gini index $\phi(x) = 4x(1-x)$ ([4]), the entropy function $\phi(x) = -x\log x - (1-x)\log(1-x)$ ([25]) and the new index $\phi(x) = 2\sqrt{x(1-x)}$ ([18]). Let $s_a(i)$, $a \in \{0,1\}$, denote the a-son of s that would be created if x_i is placed at s as a splitting variable. For each node s let $\Pr[s_a(i)]$, $a \in \{0,1\}$, denote the probability that a random example from the uniform distribution arrives at $s_a(i)$ given that it has already arrived at s. Let $p(s)$ be the probability that an example arriving at s is positive. The *impurity sum* (IS) of x_i at s using impurity function $\phi(x)$ is $\mathrm{IS}(s, x_i, \phi) = \Pr[s_0(i)]\phi(p(s_0(i))) + \Pr[s_1(i)]\phi(p(s_1(i)))$. The *purity gain* (PG) of x_i at s is: $\mathrm{PG}(s, x_i, \phi) = \phi(p(s)) - \mathrm{IS}(s, x_i, \phi)$. The estimated values of all these quantities are $\widehat{PG}$, $\widehat{IS}$, *etc.* We say that the quantity A is estimated within accuracy $\alpha \geq 0$ if $A - \alpha < \hat{A} < A + \alpha$.

Since the value of $\phi(p(s))$ is attribute-independent, the choice of maximal $\mathrm{PG}(s, x_i, \phi)$ is equivalent to the choice of minimal $\mathrm{IS}(s, x_i, \phi)$. For uniformly distributed examples $\Pr[s_0(i)] = \Pr[s_1(i)] = 0.5$. Thus if impurity sum is computed exactly, then $\phi(p(s_0(i)))$ and $\phi(p(s_1(i)))$ have equal weight. We define the *balanced impurity sum* of x_i at s as $\mathrm{BIS}(s, x_i, \phi) = \phi(p(s_0(i))) + \phi(p(s_1(i)))$.

Fig. 4 gives the structure of all impurity-based algorithms. The algorithm takes four parameters: s, identifying current tree's node, X, standing for the set of attributes available for testing, R, which is a set of function's restrictions leading to s and ϕ, identifying the impurity function. Initially s is set to the root node, X contains all attribute variables and R is an empty set.

3 Building Decision Trees Using an Influence Oracle

In this section we introduce a new algorithm, **DTInfluence** (see Fig. 5), for building decision trees using an influence oracle. This algorithm greedily chooses the splitting variable with maximal influence. Clearly, the resulting tree consists of only relevant variables. The parameters of the algorithm have the same meaning as those of **DTApproxPG**.

Lemma 1 *Let $f(x)$ be any boolean function. Then the decision tree T built by the algorithm **DTInfluence** represents $f(x)$ and has no inner node such that all examples arriving at it have the same classification.*

Proof See Appendix A.1.

DTInfluence$(s,\ X,\ R)$

if $\forall x_i \in X,\ I_{f_R}(i) = 0$ **then**
  Set classification of s as a classification of any example arriving to it.
else
  Choose $x_i = \arg\max_{x_i \in X}\{I_{f_R}(i)\}$ to be a splitting variable.
  Run **DTInfluence**$(s_1, X - \{x_i\}, R \cup \{x_i = 1\})$.
  Run **DTInfluence**$(s_0, X = \{x_i\}, R \cup \{x_i = 0\})$.
end if

Fig. 5. DTInfluence algorithm.

DTMinTerm$(s,\ F)$

if $\exists\ t_i \in F$ such that $t_i = \emptyset$ **then**
  Set s as a positive leaf.
else
  if $F = \emptyset$ **then**
    Set s as a negative leaf.
  else
    Let $t_{min} = \arg\min_{t_i \in F}\{|t_i|\}$. $t_{min} = \{\tilde{x}_{m_1}, \tilde{x}_{m_2}, \ldots, \tilde{x}_{m_{|t_{min}|}}\}$.
    Choose any $\tilde{x}_{m_i} \in t_{min}$. Let $t^{'}_{min} = t_{min} \backslash \{\tilde{x}_{m_i}\}$.
    if $\tilde{x}_{m_i} = x_{m_i}$ **then**
      Run **DTMinTerm**$(s_1, F\backslash\{t_{min}\} \cup \{t^{'}_{min}\})$, **DTMinTerm**$(s_0, F\backslash\{t_{min}\})$.
    else
      Run **DTMinTerm**$(s_0, F\backslash\{t_{min}\} \cup \{t^{'}_{min}\})$, **DTMinTerm**$(s_1, F\backslash\{t_{min}\})$.
    end if
  end if
end if

Fig. 6. DTMinTerm algorithm.

3.1 Read-Once DNF

Lemma 2 *For any $f(x)$ which can be represented as a read-once DNF, the decision tree, built by the algorithm **DTInfluence**, has minimal size and minimal height amongst all decision trees representing $f(x)$.*

The proof of Lemma 2 consists of two parts. In the first part of the proof we introduce the algorithm **DTMinTerm** (see Fig. 6) and prove Lemma 2 for it. In the second part of the proof we show that the trees built by **DTMinTerm** and **DTInfluence** are the same.

Assume we are given read-once DNF formula F. We change the algorithm **DTInfluence** so that the splitting rule is to choose any variable x_i in the smallest term $t_j \in F$. The algorithm stops when the restricted function becomes constant (true or false). The new algorithm, denoted by **DTMinTerm**, is shown in Fig. 6. The initial value of the first parameter of the algorithm is the same as in **DTInfluence**, and the second parameter is initially set to function's DNF formula F. The following two lemmata are proved at Appendices A.2 and A.3.

DTCoeff(s, X, t_s)

if $\sum_{x_i \in X} |a_i| \le t_s$ or $-\sum_{x_i \in X} |a_i| > t_s$ **then**
    The function is constant. s is a leaf.
else
    Choose a variable x_i from X, having the largest $|a_i|$.
    Run **DTCoeff**$(s_1, X - \{x_i\}, t_s - a_i)$ and **DTCoeff**$(s_0, X - \{x_i\}, t_s + a_i)$.
end if

Fig. 7. DTCoeff algorithm.

Lemma 3 *For any $f(x)$ which can be represented as a read-once DNF, the decision tree T, built by the algorithm **DTMinTerm**,*

1. *Represents $f(x)$.*
2. *Has no node such that all inputs arriving at it have the same classification.*
3. *Has minimal size and height amongst all decision trees representing $f(x)$.*

Lemma 4 *Let $x_l \in t_i$ and $x_m \in t_j$. Then $|t_i| > |t_j| \leftrightarrow I_f(l) < I_f(m)$.*

Proof (Lemma 2): It follows from Lemmata 1, 3 and 4 that the trees produced by the algorithms **DTMinTerm** and **DTInfluence** have the same size and height. Moreover, due to part 3 of Lemma 3, this size and height is minimal amongst all decision trees representing $f(x)$. $\square$

3.2 Boolean Linear Threshold Functions

Lemma 5 *For any linear threshold function $f_{a,t}(x)$, the decision tree built by the algorithm **DTInfluence** has minimal size and minimal height amongst all decision trees representing $f_{a,t}(x)$.*

The proof of the Lemma 5 consists of two parts. In the first part of the proof we introduce the algorithm **DTCoeff** (see Fig. 7) and prove Lemma 5 for it. In the second part of the proof we show that the trees built by **DTCoeff** and **DTInfluence** have the same size.

The difference between **DTCoeff** and **DTinfuence** is in the choice of splitting variable. **DTCoeff** chooses the variable with the largest $|a_i|$ and stops when the restricted function becomes constant (true or false). The meaning and initial values of the first two parameters of the algorithm are the same as in **DTInfluence**, and the third parameter is initially set to the function's threshold t. The following two lemmata are proved at Appendices A.4 and A.5:

Lemma 6 *For any boolean LTF $f_{a,t}(x)$ the decision tree T, built by the algorithm **DTCoeff**:*

1. *Represents $f_{a,t}(x)$.*
2. *Has no node such that all inputs arriving at it have the same classification.*
3. *Has minimal size and height amongst all decision trees representing $f_{a,t}(x)$.*

DTExactPG(s, X, R, ϕ)

if all examples arriving at s have the same classification **then**
  Set s as a leaf with that value.
else
  Choose $x_i = \arg\max_{x_i \in X}\{PG(f_R, x_i, \phi)\}$ to be a splitting variable.
  Run **DTExactPG**($s_1, X - \{x_i\}, R \cup \{x_i = 1\}, \phi$).
  Run **DTExactPG**($s_0, X - \{x_i\}, R \cup \{x_i = 0\}, \phi$).
end if

Fig. 8. **DTExactPG** algorithm.

Lemma 7 *If $I_f(i) > I_f(j)$ then $|a_i| > |a_j|$.*

Note that if $I_f(i) = I_f(j)$ then there may be any relation between $|a_i|$ and $|a_j|$. The next lemma shows that choosing the variables with the same influence in any order does not change the size of the resulting decision tree. For any node s, let X_s be the set of all variables in X which are untested on the path from the root to s. Let $\hat{X}(s) = \{x_1, \ldots x_k\}$ be the variables having the same non-zero influence, which in turn is the largest influence amongst the influences of variables in X_s.

Lemma 8 *Let T_i (T_j) be the smallest decision tree one may get when choosing any $x_i \in \hat{X}_s$ ($x_j \in \hat{X}_s$) at s. Let $|T_{opt}|$ be the size of the smallest tree rooted at s. Then $|T_i| = |T_j| = |T_{opt}|$.*

Proof See Appendix A.6.

Proof (Lemma 5) One can prove an analogous result for Lemma 8, dealing with the height rather than size. Combining this result with Lemmata 1, 6, 7 and 8 we obtain that the trees built by **DTInfluence** and **DTCoeff** have the same size and height. Moreover, due to part 3 of Lemma 6, these size and height are minimal amongst all decision trees representing $f_{a,t}(x)$. $\square$

4 Optimality of Exact Purity Gain

In this section we introduce a new algorithm for building decision trees, named **DTExactPG**, (see Fig. 8) using exact values of purity gain. The next lemma follows directly from the definition of the algorithm:

Lemma 9 *Let $f(x)$ be any boolean function. Then the decision tree T built by the algorithm **DTExactPG** represents $f(x)$ and there exists no inner node such that all inputs arriving at it have the same classification.*

Lemma 10 *For any boolean function $f(x)$, uniformly distributed x, and any node s, $p(s_0(i))$ and $p(s_1(i))$ are symmetric relative to $p(s)$: $|p(s_1(i)) - p(s)| = |p(s_0(i)) - p(s)|$ and $p(s_1(i)) \neq p(s_0(i))$.*

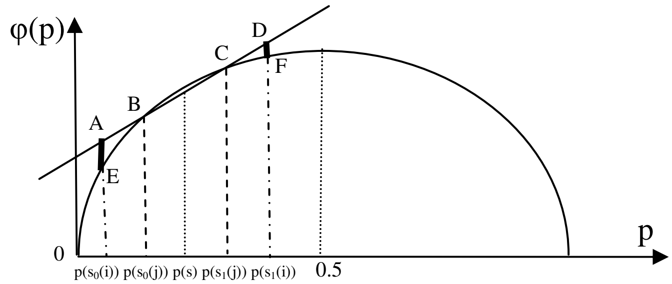

Fig. 9. Comparison of impurity sum of x_i and x_j

Proof See Appendix B.1.

Lemma 11 *For any unate boolean function $f(x)$, uniformly distributed input x, and any impurity function ϕ, $I_f(i) > I_f(j) \leftrightarrow PG(f, x_i, \phi) > PG(f, x_j, \phi)$.*

Proof Since x is distributed uniformly, it is sufficient to prove $I_f(i) > I_f(j) \leftrightarrow BIS(f, x_i, \phi) < BIS(f, x_j, \phi)$. Let d_i be number of pairs of examples differing only in x_i and having different target value. Since all examples have equal probability $I_f(i) = \frac{d_i}{2^{n-1}}$. Consider a split of node s according to x_i. All positive examples arriving at s may be divided into two categories:

1. Flipping the value of i-th attribute does not change the target value of example. Then the first half of such positive examples passes to s_1 and the second half passes to s_0. Consequently such positive examples contribute equally to the probabilities of positive examples in s_1 and s_0.
2. Flipping the value of i-th attribute changes the target value of example. Consider such pair of positive and negative examples, differing only in x_i. Since $f(x)$ is unate, either all positive example in such pairs have $x_i = 1$ and all negative examples in such pairs have $x_i = 0$, or vice versa. Consequently either all such positive examples pass either to s_1 or to s_0. Thus such examples increase the probability of positive examples in one of the nodes $\{s_1, s_0\}$ and decrease the probability of positive examples in the other.

The number of positive examples in the second category is d_i. Thus $I_f(i) > I_f(j) \leftrightarrow \max\{p(s_1(i)), p(s_0(i))\} > \max\{p(s_1(j)), p(s_0(j))\}$. By Lemma 10, if $\max\{p(s_1(i)), p(s_0(i))\} > \max\{p(s_1(j)), p(s_0(j))\}$ then the probabilities of x_i are more distant from $p(s)$ than those of x_j. Fig. 9 depicts one of the possible variants of reciprocal configuration of $p(s_1(i))$, $p(s_0(i))$, $p(s_1(j))$, $p(s_0(j))$ and $p(s)$, when all these probabilities are less than 0.5. Let $\phi(A)$ be the value of impurity function at point A. Then $BIS(f, x_j, \phi) = \phi(p(s_1(j))) + \phi(p(s_0(j))) = \phi(B) + \phi(C) = \phi(A) + \phi(D) > \phi(E) + \phi(F) = BIS(f, x_i, \phi)$. Note that the last inequality holds due to concavity of impurity function. The rest of the cases of reciprocal configurations of probabilities and 0.5 may be proved similarly. $\square$

Proof (of Theorem 1) The theorem follows from combining Lemmata 1, 9, 11, 2 and 5. $\square$

5 Optimality of Approximate Purity Gain

The purity gain computed by practical algorithms is not exact. However, under some conditions approximate purity gain suffices. The proof of this result (which is essentially Theorem 2) is based on the following lemma (proved at Appendix C.1)

Lemma 12 *Let $f(x)$ be a boolean function, which can be represented by decision tree of depth h. Suppose x is distributed uniformly. Then $Pr(f(x)=1)=\frac{r}{2^h},\ \ r\in\mathbb{Z},\ 0\le r\le 2^h$.*

Appendix C.2 contains a proof of Theorem 2.

6 Noise-Tolerant Probably Correct Learning

In this section we assume that each input example is misclassified with probability $\eta<0.5$. Since our noise-free algorithms learn probably correctly, we would like to obtain the same results of probable correctness with noisy examples. Our definition of PC learning with noise is that the examples are noisy yet, nonetheless, we insist upon zero generalization error. Previous learning algorithms with noise (*e.g.* [2] and [17]) require a non-zero generalization error.

We introduce an algorithm **DTStatQuery** (see Appendix D), which is a reformulation of the practical impurity-based algorithms in terms of statistical queries. [2] and [17] show how to simulate statistical queries from examples corrupted by small classification noise. Adapting this simulation to the case of PC learning and combining it with **DTStatQuery** algorithm we obtain the noise-tolerant version of impurity-based algorithms. See Appendix D for the proof of Theorem 3.

7 Directions for Future Research

Basing on our results, the following open problems may be attacked:

1. **Extension to other types of functions.** In particular we conjecture that for all unate functions the algorithm **DTExactPG** builds minimal depth decision trees and the size of the resulting trees is not far from minimal.
2. **Extension to other distributions**. It may be verified that all our results hold for constant product distributions ($\forall i\ \Pr[x_i=1]=c$) and does not hold for general product distributions ($\forall i\ \Pr[x_i=1]=c_i$).
3. **Small number of examples.** The really interesting case is when number of examples is less than $poly(2^h)$. In this case nothing is known about the size and depth of resulting decision tree.

Moreover we would like to compare our noise-tolerant version of impurity-based algorithms *vs.* pruning methods. Finally since influence and impurity gain are logically equivalent, it would be interesting to use the notion of purity gain in the field of analysis of boolean functions.

Acknowledgements

We thank Yishay Mansour for his great help with all aspects of this paper. We also thank Adam Smith who greatly simplified and generalized an earlier version of Theorem 1. We are grateful to Rocco Servedio for pointing us an error in the earlier version of the paper.

References

1. D. Angluin, L. Hellerstein and M. Karpinski. Learning Read-Once Formulas with Queries. *Journal of the ACM*, 40(1):185-210, 1993.
2. J.A. Aslam and S.E. Decatur. Specification and Simulation of Statistical Query Algorithms for Efficiency and Noice Tolerance. *Journal of Computer and System Sciences*, 56(2):191-208, 1998.
3. A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour and S. Rudich. Weakly Learning DNF and Characterizing Statistical Query Learning Using Fourier Analysis. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing*, pages 253-262, 1994.
4. L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.
5. N.H. Bshouty and L. Burroughs. On the Proper Learning of Axis-Parallel Concepts. *Journal of Machine Learning Research*, 4:157-176, 2003.
6. N.H. Bshouty and V. Feldman. On Using Extended Statistical Queries to Avoid Membership Queries. *Journal of Machine Learning Research*, 2:359-395, 2002.
7. N.H. Bshouty, E. Mossel, R. O'Donnel and R.A. Servedio. Learning DNF from Random Walks. In *Proceedings of the 44th Annual Symposium on Foundations of Computer Science*, 2003.
8. H. Chernoff. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations. *Annals of Mathematical Statistics*, 23:493-509, 1952.
9. E. Cohen. Learning Noisy Perceptron by a Perceptron in Polynomial Time. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 514-523, 1997.
10. U. Feige. A Threshold of $\ln n$ for Approximating Set Cover. *Journal of the ACM* 45(4):634-652, 1998.
11. T. Hancock, T. Jiang, M. Li and J. Tromp. Lower bounds on Learning Decision Trees and Lists. *Information and Computation*, 126(2):114-122, 1996.
12. D. Haussler. Quantifying Inductive Bias: AI Learning Algorithms and Valiant's Learning Framework. *Artificial Intelligence*, 36(2): 177-221, 1988.
13. W. Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58:13-30, 1963.
14. L. Hyafil and R.L. Rivest. Constructing Optimal Binary Decision Trees is NP-Complete. *Information Processing Letters*, 5:15-17, 1976.
15. J. Jackson, R.A. Servedio. Learning Random Log-Depth Decision Trees under the Uniform Distribution. In *Proceedings of the 16th Annual Conference on Computational Learning Theory*, pages 610-624, 2003.
16. A. Kalai and R.A. Servedio. Boosting in the Presence of Noise. In *Proceedings of the 35th Annual Symposium on the Theory of Computing*, pages 195-205, 2003.
17. M.J. Kearns. Efficient Noise-Tolerant Learning from Statistical Queries. *Journal of the ACM*, 45(6):983-1006, 1998.

18. M.J. Kearns and Y. Mansour. On the Boosting Ability of Top-Down Decision Tree Learning Algorithms. *Journal of Computer and Systems Sciences*, 58(1):109-128, 1999.
19. M.J. Kearns, L.G. Valiant. Cryptographic Limitations on Learning Boolean Formulae and Finite Automata. *Journal of the ACM*, 41(1):67-95, 1994.
20. M.J. Kearns, U. Vazirani. *An Introduction to Computational Learning Theory.* The MIT Press, 1994.
21. E. Kushilevitz and Y. Mansour. Learning Decision Trees using the Fourier Spectrum. *SIAM Journal on Computing*, 22(6):1331-1348, 1993.
22. Y. Mansour and M. Schain. Learning with Maximum-Entropy Distributions. *Machine Learning*, 45(2):123-145, 2001.
23. M. Moshkov. Approximate Algorithm for Minimization of Decision Tree Depth. In *Proceedings of 9th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 611-614, 2003.
24. S.K. Murthy. Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. *Data Mining and Knowledge Discovery*, 2(4): 345-389, 1998.
25. J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
26. E. Takimoto and A. Maruoka. Top-Down Decision Tree Learning as Information Based Boosting. *Theoretical Computer Science*, 292:447-464, 2003.
27. L.G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134-1142, 1984.

A Proofs of Lemmata from Section 3

A.1 Proof of Lemma 1

The stopping condition of **DTInfluence** is given at line 1 of Fig. 5. We need to prove that upon arriving to some inner node s with the set X of variables available for testing and the set R of restrictions there is a variable in X with non-zero influence in $f_R(x)$ iff there exist two examples arriving to s and having different classification.

$\Rightarrow$ case. Suppose that upon arriving to some inner node s with the set X of variables available for testing there is a variable $x_i \in X$ with non-zero influence in $f_R(x)$. Therefore, by the definition of influence, there are two examples arriving to s, differing only in the i-th attribute and having different classification.

$\Leftarrow$ case. Suppose there exist two examples r_1 and r_2 arriving at s and having different classification. Since both examples arrive at s there exists a finite-length walk from r_1 to r_2, containing only examples arriving at s, such that two successful examples in the walk differ only in some single variable $x_i \notin R$. Since r_1 and r_2 have different classification there exist two successful examples on the walk differing in only single variable $x_i \notin R$ and having differing classification. Thus $I_{f_R}(i) \neq 0$. $\square$

A.2 Proof of Lemma 3

1) We need to prove that for all x, $f(x) = c_T(x)$. Suppose at some inner node s the algorithm chose $x_i \in t_j$. Let F_s be the value of the second parameter F

upon arriving recursively to the inner node s. Let X'_l be the set of all variables tested on the path from the root to the leaf l. All inputs arriving at l have the same values in each $x_i \in X'_l$.

If some value v of x_i agrees with $t_j \in F_s$ then, upon passing to the son corresponding to the v value, the algorithm deletes the variable x_i from $t_j \in F_s$. Assume some leaf l of T has classification '1'. Since $\emptyset \in F_l$, there exists a term $t_j \in F$, such that the values of variables in X'_l satisfy it. Since all inputs arriving at l have the same values in variables from X'_l, only positive inputs arrive at l.

If some value v of x_i disagrees with $t_j \in F_s$, then, upon passing to the son corresponding to the v value, the algorithm deletes t_j from F_s. Let some leaf l of T has classification '0'. By the definition of the algorithm $F_l = \emptyset$. Then the values of variables in X'_l prevents from satisfaction of any term $t_i \in F$. Since all examples arriving at l have the same values in variables from X'_l, only negative examples arrive at l.

2) Suppose there is an inner node s such that only positive or only negative inputs arrive at it. Let F_s be the value of second parameter when the algorithm arrives recursively to s. The rest of the proof is similar to that Lemma 3. Let X'_s be the set of variables having the same values in all inputs arriving at s. Therefore the variables from X'_s are exactly the variables tested on the path from root to s.

Suppose all inputs arriving at s have classification '1'. Then the values of variables at X'_s, leading to s satisfy at least one term. Therefore, upon arriving at s, $\emptyset \in F$ and s is set as a leaf with '1' classification.

Suppose all inputs arriving at s have classification '0'. Then the values of variables at X'_s, leading to s prevent from satisfaction of any term. Therefore upon arriving at s, $F = \emptyset$ and s is set as a leaf with '0' classification.

3) We give the proof of minimal size. The proof of minimal height is exactly the same (just change the words size and height). At each node of the minimal size decision tree the splitting variable is chosen from the set of variables which are untested on the path from the root to that inner node. We only need to prove that there exists a minimal size decision tree having at its root inner node the test of any variable presenting in the smallest term of f. Due to a recursive nature of decision trees this proof implies that there exists a minimal size decision tree, such that at each its inner node s the variable chosen presents in the smallest term of the read-once DNF formula describing examples arriving at s. The part 3 of the lemma may be concluded from combining this result with parts 1 and 2 of the current lemma.

In the following proof we assume that the function is monotone. The proof for nonmonotone function is the same except s_1 and s_0 are interchanged if $\tilde{x}_i = \overline{x}_i$. The proof is by induction on the number of variables in F. The base case of single variable in F is trivial - in this case the only possible tree consists of single inner node testing this variable and this tree is the minimal one. Assume the proposition holds for $n-1$ variables.

Given n variables, suppose there exists minimal size decision tree testing $x_j \in t_l$ at its root, such that there exists $x_i \in t_k$ and $|t_l| > |t_k| \geq 1$. We prove that there exists also minimal size decision tree testing x_i at it's root. Let $t'_l = t_l \backslash \tilde{x}_j$. Since $|t_l| > 1$ the right son s_1 of the root is an inner node. The (read-once) DNF formula corresponding to examples arriving to s_1 is $F_{s_1} = F \backslash t_l \cup t'_l$. Since F contains at least two terms, the left son s_0 of the root is an inner node too. The read-once DNF formula corresponding to examples arriving to s_0 is $F_{s_0} = F \backslash t_l$. Both F_{s_1} and F_{s_0} have $n-1$ variables. Moreover t_k is still the smallest term in both F_{s_1} and F_{s_0}. Consequently, by inductive hypothesis, there exists a smallest subtree T_1, starting at s_1 and having x_i at its root. Similarly, by the inductive hypothesis, there exists a minimal size subtree T_0, starting at s_0 and having x_i at its root. Therefore there exists a minimal size decision tree T representing $f(x)$, having x_i at its root and x_j in both sons of root. As can be seen from Fig. 10(a) the decision tree T is equivalent to another decision tree T' testing x_i at its root and having the same size as T. $\square$

A.3 Proof of Lemma 4

The influence of x_l is the probability that inverting x_l would yield the change of $f(x)$. This may happen only in two cases:

1. The term t_i is satisfied, all other terms are not satisfied and the change in x_l causes t_i to be not satisfied.
2. All terms are not satisfied and the change in x_l causes t_i to be satisfied.

Let P_0 be the probability that all terms, except t_i and t_j, are not satisfied. Since $f(x)$ can be represented by a read-once formula F and x is uniformly distributed, the probability of each of two above-mentioned cases is $\frac{1}{2^{|t_i|}}\left(1-\frac{1}{2^{|t_j|}}\right) \cdot \mathrm{P}_0$. Therefore $I_f(l) = 2 \cdot \frac{1}{2^{|t_i|}}\left(1-\frac{1}{2^{|t_j|}}\right) \cdot \mathrm{P}_0$ and $I_f(m) = 2 \cdot \frac{1}{2^{|t_j|}}\left(1-\frac{1}{2^{|t_i|}}\right) \cdot \mathrm{P}_0$. Thus, if $|t_i| = |t_j|$ then $I_f(l) = I_f(m)$. Moreover, if $|t_i| > |t_j|$ then $\frac{1}{2^{|t_i|}}\left(1-\frac{1}{2^{|t_j|}}\right) < \frac{1}{2^{|t_j|}}\left(1-\frac{1}{2^{|t_i|}}\right)$ and therefore $I_f(l) < I_f(m)$. $\square$

A.4 Proof of Lemma 6

1) When passing from s to its sons, s_1 and s_0, the algorithm updates the threshold which should be passed by inputs arriving at the right subtree's (rooted by s_1) leaves and left subtree's (rooted by s_0) leaves in order to obtain the classification '1'. Intuitively, if at step 1 of the algorithm $-\sum_{x_i \in X} |a_i| > t_s$, then all inputs arriving at s pass the threshold T disregarding of values of variables which are not on the path from the root to s. Thus every input arriving at s has '1' classification. Similarly, if $\sum_{x_i \in X} |a_i| \leq t_s$ then all inputs arriving at s do not pass the threshold T, disregarding of values of variables which are not on the path from the root to s. In this case every input arriving at s has '0' classification. Next we prove formally the last two statements.

Suppose some leaf l of T has classification '1'. Let X be the set of all variables and let X' be the set of variables tested on the path from the root to l. All inputs arriving at l have the same values in each $x_i \in X'$. Since l has classification

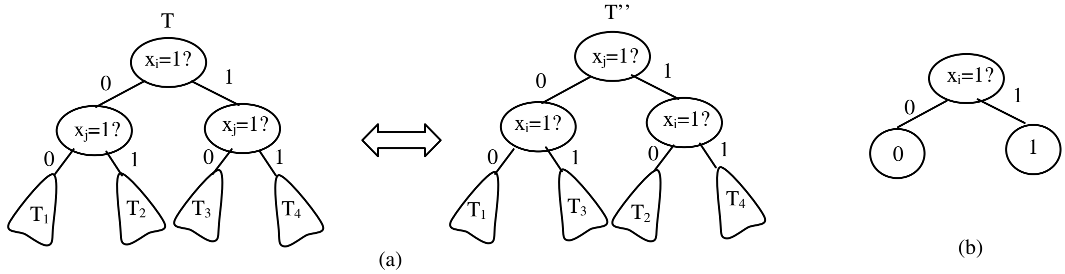

Fig. 10. (a) Case A of Lemma 6; (b) Case B of Lemma 6.

'1', for all inputs arriving at l holds $-\sum_{x_i \in X \setminus X'} |a_i| > t_l = t - \sum_{x_i \in X'} a_i x_i$. Thus for all inputs arriving at l, $\sum_{x_i \in X} a_i x_i = \sum_{x_i \in X'} a_i x_i + \sum_{x_i \in X \setminus X'} a_i x_i \geq \sum_{x_i \in X'} a_i x_i - \sum_{x_i \in X \setminus X'} |a_i| > t$. Consequently only positive inputs arrive at the leaves with classification '1'. Similarly for negative leaves $\sum_{x_i \in X \setminus X'} |a_i| \leq t_l = t - \sum_{x_i \in X'}$ and $\sum_{x_i \in X} a_i x_i = \sum_{x_i \in X'} a_i x_i + \sum_{x_i \in X \setminus X'} a_i x_i \leq \sum_{x_i \in X'} a_i x_i + \sum_{x_i \in X \setminus X'} |a_i| \leq t$. Thus only negative inputs arrive at the leaves with classification '0'. Therefore the decision tree T classifies inputs with '1' iff $\sum_{x_i \in X} a_i x_i > t$.

2) Suppose there is an inner node s such that only positive or only negative inputs arrive at it. Let X_s and t_s be the values of the second and third parameter when the algorithm arrives recursively at s. Then either $\sum_{x_i \in X_s} |a_i| \leq t_s = t - \sum_{x_i \in X \setminus X_s}$ (in case of negative inputs) or $-\sum_{x_i \in X_s} |a_i| > t_s = t - \sum_{x_i \in X \setminus X_s}$ (in case of positive inputs). Since the algorithm creates leaf from s if either $\sum_{x_i \in X} |a_i| \leq t_s$ or $-\sum_{x_i \in X} |a_i| > t_s$, there is no inner node at T such that all inputs arriving at it have the same classification.

3) We give the proof of minimal size. The proof of minimal height is exactly the same (just change the words size and height).

At each inner node of the optimal decision tree the variable is chosen from the set of variables which are untested on the path from the root to that inner node. We prove that there exists a minimal size decision tree having at its root inner node the test of any variable with largest absolute value of its coefficient. Due to a recursive nature of decision trees this implies that there exists a minimal size decision tree such that at each its inner node the variable chosen has the largest absolute coefficient amongst the coefficients which are untested on the path from the root to that inner node. Then part 3 of the lemma is concluded from combining this result with parts 1 and 2 of the current lemma.

The proof is by induction on the number of variables. In the base case there is a single variable and the lemma trivially holds. Assume the lemma holds for $n-1$ variables. Given n variables consider a tree T representing $f_{a,t}(x)$. Suppose that T tests at its root the variable x_i, such that there exists another variable x_j, and for all k, $|a_k| \leq |a_j|$. Then, to complete the proof of the existence of minimal size decision tree choosing at each node the variable with the largest absolute value of the coefficient, several cases of reciprocal configuration of the node with x_i, the node with x_j and the relevant leaves should be considered.

Case A (see Fig. 10(a)). In this case both sons of the root node are not leafs. Since x_j has the largest value $|a_j|$, by inductive hypothesis each son of the root node tests x_j. Then the tree T can be transformed to the equivalent tree T' with the same size and height, testing at the root the variable x_j with largest $|a_j|$.

Case B (see Fig. 10(b)). In this case both sons of the root node are leaves, the right leaf has '1' classification and the left leaf has '0' classification. Since the right leaf has '1' classification, by the definition of the algorithm, $-|a_j| - \sum_{k=1,\ k\neq i,j}^{n} |a_k| > t - a_i$. Similarly since the left leaf has '0' classification, $|a_j| + \sum_{k=1,\ k\neq i,j}^{n} |a_k| \leq t + a_i$ Then the following system of equations holds:

$$\begin{cases} a_i - |a_j| - \sum_{k=1,\ k\neq i,j}^{n} |a_k| > t \\ -a_i + |a_j| + \sum_{k=1,\ k\neq i,j}^{n} |a_k| \leq t \end{cases}$$

It follows from this system of equations that $0 \leq \sum_{k=1,\ k\neq i,j}^{n} |a_k| < (a_i - |a_j|)$. This is a contradiction to the assumption $|a_i| \leq |a_j|$. Thus this case can not happen.

Case C (see Fig. 11). In this case the right son of the root is a leaf with '1' classification and the left son is the inner node. By the inductive hypothesis this inner node tests x_j. Since the left son of x_i is an inner node, $-\sum_{k=1,\ k\neq i} |a_k| \leq t + a_i$. Using the fact that the right son of x_i is a leaf with '1' classification, the following system of equations holds:

$$\begin{cases} a_i - |a_j| - \sum_{k=1,\ k\neq i,j}^{n} |a_k| > t \\ -a_i - |a_j| - \sum_{k=1,\ k\neq i,j}^{n} |a_k| \leq t \end{cases}$$

From these equations it follows that $a_i > 0$. There are two subcases (see Fig. 11):

- $a_j > 0$. This implies $a_j - a_i - \sum_{k=1,\ k\neq i,j}^{n} |a_k| \geq -|a_j| + a_i - \sum_{k=1,\ k\neq i,j}^{n} |a_k| > t$. Therefore $-\sum_{k=1,\ k\neq i,j}^{n} |a_i| > t + a_i - a_j$ and the right son of x_j is a leaf with '1' classification. Consequently the tree T essentially has the structure of the tree T'. Moreover if we write the last inequality in slightly different form we obtain $-\sum_{k=1,\ k\neq j}^{n} |a_k| > t - a_j$. Thus if x_j is placed at the root then its right son will be a leaf with '1' classification. Finally we got that T' in turn is equivalent to the tree $T1$.
- $a_j < 0$. In this subcase the proof is very similar to the previous subcase and is omitted.

In both subcases the initial tree is equivalent to the tree with the same size, having x_j tested at its root.

The remaining cases of reciprocal configuration of the node with x_i, the node with x_j and the relevant leaves, differ from three cases proved above in the placement and classification of leaves, and may be proved very similarly. □

A.5 Proof of Lemma 7

We prove the equivalent proposition that if $|a_i| \geq |a_j|$ then $I_f(i) \geq I_f(j)$. Let x_i

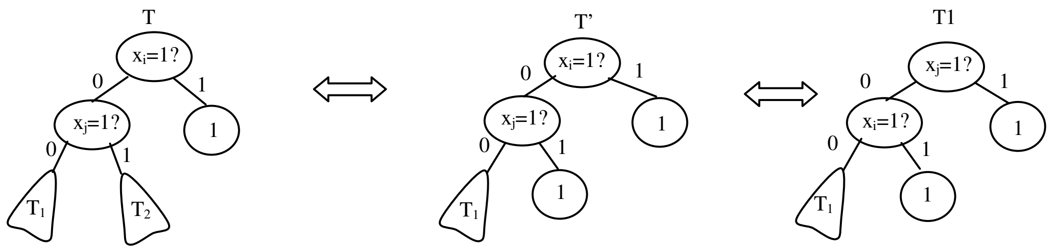

Fig. 11. Case C of Lemma 6.

x_i	x_j	other variables	function value
v	v'	$w_1, w_2, w_3, \ldots, w_{n-2}$	t_1
$-v$	v'	$w_1, w_2, w_3, \ldots, w_{n-2}$	t_2
v	$-v'$	$w_1, w_2, w_3, \ldots, w_{n-2}$	t_3
$-v$	$-v'$	$w_1, w_2, w_3, \ldots, w_{n-2}$	t_4

Fig. 12. Structure of the truth table G_w from $G(i,j)$.

and x_j be two different variables in $f(x)$. For each of the 2^{n-2} possible assignments to the remaining variables we get a 4 row truth table for different values of x_i and x_j. Let $G(i,j)$ be the multi set of 2^{n-2} truth tables, indexed by the assignment to the other variables. *I.e.*, G_w is the truth table where the other variables are assigned values $w = w_1, w_2, \ldots, w_{n-2}$. The structure of a single truth table is shown in Fig. 12. In this figure, and generally from now on, v and v' are constants in $\{-1, 1\}$. Observe that $I_f(i)$ is proportional to the sum over the 2^{n-2} G_w's in $G(i,j)$ of the number of times $t_1 \neq t_2$ plus the number of times $t_3 \neq t_4$. Similarly, $I_f(j)$ is proportional to the sum over the 2^{n-2} G_w's in $G(i,j)$ of the number of times $t_1 \neq t_3$ plus the number of times $t_2 \neq t_4$.

Consider any truth table $G_w \in G(i,j)$ (see Fig. 12). Let the *contribution* of G_w to the influence of x_j be the number of pairs of rows in G_w differing only in x_j and in the function value. By our previous observations, to prove the lemma it is sufficient to prove that the contribution of G_w to the influence of x_i is at least the contribution of G_w to the influence of x_j. According to the definition of G_w the contribution may be zero, one or two. Next we will consider these three cases of the contribution of G_w to the influence of x_j:

The first case is of zero contribution. Clearly, in this case the contribution of G_w to the influence of x_i is at least the contribution of G_w to the influence of x_j.

The next case is when the contribution of G_w to the influence of x_j is exactly one. *I.e.*, there are exactly two rows in G_w, in one of which $x_j = v$ and in the other $x_j = -v$, whereas the function values are different. Because $|a_i| \geq |a_j|$ it must be the case that there are also two rows in the table, in one of which $x_i = v$ and in the other $x_i = -v$, where the function values in these two rows

x_i	x_j	other variables	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
v	v'	$w_1, w_2, w_3, \dots, w_{n-2}$	0	0	0	1	1	0	0	1	0
$-v$	v'	$w_1, w_2, w_3, \dots, w_{n-2}$	1	1	1	1	1	0	0	1	0
v	$-v'$	$w_1, w_2, w_3, \dots, w_{n-2}$	0	1	0	1	0	0	1	0	0
$-v$	$-v'$	$w_1, w_2, w_3, \dots, w_{n-2}$	0	1	1	1	0	0	1	1	1

Fig. 13. Possible variants (functions $f_1 - f_9$) of contents of the truth table G_w from $G(i,j)$, such that $I_f(i) > 0$.

are different too. Thus, in this case the contribution of G_w to the influence of x_i is at least the contribution of G_w to the influence of x_j.

The last case is when the contribution of G_w to the influence of x_j is two. *I.e.*, there are two pairs of rows in the table, in every pair one row has $x_j = v$ and the other has $x_j = -v$, whereas the function values are different. By the definition of boolean linear threshold function, this means that x_j determines the value of the function in this G_w whereas x_i does not. This contradicts the assumption that $|a_i| \geq |a_j|$. Therefore this case is not possible. $\square$

A.6 Proof of Lemma 8

The proof is by induction on k. For $k = 1$ the lemma trivially holds. Assume the lemma holds for all $\ell < k$. Next we prove the lemma for k.

Consider two attributes x_i and x_j, $x_i \in \hat{X}(s)$, $x_j \in \hat{X}(s)$, $I_f(i) = I_f(j)$. If $|a_i| = |a_j|$ then the proof of Lemma 6 shows that the choice of both x_i and x_j in s would lead to the decision trees of the same size. Consequently, if all variables in $\hat{X}(s)$ have the same absolute values of their coefficients then the choice of any of that variables at s would lead to the smallest decision tree. Thus let us assume $|a_i| > |a_j|$.

The next part of proof is quite long and initially we give a high-level overview of it. In the proof we use the multiset $G(i,j)$, defined earlier. We consider the possible values of target value assignments in the truth table $G_w \in G(i,j)$. Then we show that if the underlying function is a boolean linear threshold function, $I_f(i) = I_f(j)$ and $|a_i| > |a_j|$ then there are only six possible types of target value assignments in the truth tables in $G(i,j)$. Moreover we find that there are several interesting relations within these six types of target value assignments. Finally, exploiting these relations we show that whenever x_i is chosen the choice of x_j would result in the decision tree of the same size.

Consider a multiset $G(i,j)$. Since $I_f(i) \neq 0$ there is at least one truth table $G_w \in G(i,j)$ such that changing the value x_i in the examples of this truth table from v to $-v$ and leaving x_j set to v' would change the value of the function from 0 to 1. Since the change of the value of x_i from v to $-v$, when x_j equals to v', causes the function to change from 0 to 1, the sum $\sum_{i=1}^{n} a_i x_i$ grows. Thus when changing the value of x_i from v to $-v$, when x_j equals to $-v'$, the sum $\sum_{i=1}^{n} a_i x_i$ grows too. In this case the function value may not decrease, *i.e.* there is no truth table $G_w \in G(i,j)$ such that the change of x_i in it from v to $-v$ causes the function to change from 1 to 0. Under such constraint each truth table $G_w \in G(i,j)$ has one of 9 function assignments, $f_1 - f_9$, depicted in Fig. 13. In the next part of the proof we further reduce the set of possible function assignment for each truth table $G_w \in G(i,j)$.

Consider the function f_5. According to this function the change of x_j from $-v'$ to v', when $x_i = v$, changes the value of function from 0 to 1. We showed previously that for any fixed value of x_j changing x_i from v to $-v$ increases the value of the sum $\sum_{i=1}^{n} a_i x_i$. Therefore, since $|a_i| > |a_j|$, the change from v to $-v$ in the value of x_i when x_j is $-v'$ should also change the value of the function from 0 to 1. But this is a contradiction to the definition of f_5, defining in this case the value of function to remain 0. By the very similar proof it can be shown that the definition of the function f_7 contradicts the condition $|a_i| > |a_j|$ too. Thus in our settings there is no truth table $G_w \in G(i,j)$ having assignment to the examples according to f_5 or f_7.

So far we have remained with the possibilities of functions f_1, f_2, f_3, f_4, f_6, f_8 and f_9. In all these functions except f_3 the number of times (over 4 assignments in G_w) the function changes due to a change in x_i equals to the number of times the function changes due to a change in x_j. In case of function f_3, the number of times the function changes due to a change in x_i is 2 and the number of times the function changes due to a change in x_j is zero. Since there is no function in $\{f_1, f_2, f_4, f_6, f_8, f_9\}$ such that the number of times the function changes due to a change in x_j is strictly greater than the number of times the function changes due to a change in x_i, $I_f(i) > I_f(j)$. This is a contradiction since we are given $I_f(i) = I_f(j)$. Consequently there is no truth table $G_w \in G(i,j)$ having function values according to f_3.

Thus the only possible function assignments to any truth table $G_w \in G(i,j)$ are those done according to f_1, f_2, f_4, f_6, f_8 and f_9. We refer to the set of this functions as F.

Consider the smallest tree T having x_i is tested at s. There are 3 cases to be considered:

1. Both sons of x_i are leaves.
2. Both sons of x_i are non-leaves.
3. Exactly one of the sons of x_i is a leaf.

First let's analyze the first case. In this case the value of x_i defines the value of the function. Amongst all functions in F there is no function satisfying this conditions. Consequently the first case is not possible.

Consider the second case. Since T is the smallest tree amongst the trees with x_i tested at their root, right and left subtrees of x_i are the smallest ones. By inductive hypothesis there exist right and left smallest subtrees of x_i, such that each of them is rooted by x_j. Thus, as shown in Fig. 10(a), x_i and x_j may be interchanged to produce equivalent decision tree T', testing x_j at s and having the same number of nodes.

The third case. Since T is the smallest tree amongst the trees with x_i tested at their root, the subtree rooted at the non-leaf son of x_i is the smallest one. By inductive hypothesis there exists smallest subtree starting from the non-leaf son of x_i and testing x_j at its root. To complete the proof of the current case we will explore various relationships between the functions in F.

Consider the functions f_1 and f_9. According to the function f_1, the largest contribution of x_i and x_j to the sum $\sum_{i=1}^{n} a_i x_i$ is when $x_i = -v$ and $x_j = v'$. But according to the function f_9 the largest contribution of x_i and x_j is when $x_i = -v$ and $x_j = -v'$. Therefore the multiset $G(i,j)$ cannot contain truth tables with function value assignments according to both f_1 and f_9. By the similar argument (considering smallest contribution of x_i and x_j) it can be showed that $G(i,j)$ cannot contain truth tables with function value assignments according to both f_2 and f_8.

Consider the functions f_1 and f_2. According to f_1 the target does not decrease when x_j is changed from $-v'$ to v' and x_i remains untouched. But according to f_2 there is a case when when x_j is changed from $-v'$ to v', x_i remains untouched (and equal to v) and the target value decreases. This contradicts to the definition of boolean linear threshold function. Therefore $G(i,j)$ cannot contain truth tables with targets assigned according to both f_1 and f_2.

We defined at the beginning of the proof that there is at least one pair of examples differing only in the attribute x_i, having $x_j = v'$ such that changing the value of x_i in this pair from v to $-v$ changes the target value of example from 0 to 1. Therefore the multiset $G(i,j)$ should contain at least one truth table G having target value assignments according to f_1 or f_2.

Combining the constraints of last two paragraphs we conclude that there are two possible variants of target value assignments in $G(i,j)$

A The target values in truth tables $G_w \in G(i,j)$ are set according to the functions from $F_1 = \{f_1, f_8, f_4, f_6\}$.
B The target values in truth tables $G_w \in G(i,j)$ are set according to the functions from $F_2 = \{f_2, f_9, f_4, f_6\}$.

If the multiset $G(i,j)$ contains two truth tables G_w and $G_{w'}$ such that one of them has function assignments according to f_1 and the other has function assignments according to f_2, then none of the values of x_i determine the value of the function. Therefore in this case none of the sons of the node with x_i is a leaf. Consequently both sons of x_i are inner nodes with x_j and this case is essentially the case 2 considered previously. The same reduction to the case 2 exists if $G(i,j)$ contains two truth tables G_w and $G_{w'}$ such that one of them has function assignments according to f_4 and the other has function assignments according to f_6, or f_1 and f_4, or f_2 and f_6, or f_1 and f_8 or f_2 and f_9.

Recall that we stated in the beginning of the proof that since $I_f(i) \neq 0$ there is at least one truth table $G_w \in G(i,j)$ such that changing the value x_i in the examples of this group from v to $-v$ and leaving x_j set to v' would change the value of the function from 0 to 1. Thus $G(i,j)$ must contain at least one group G having function assignments according to f_1 or f_2. Subject to restrictions we have described in the previous two paragraphs there are only two cases that exactly one of the sons of x_i will be a leaf:

3.1. $G(i,j)$ contains only truth tables having function value assignments according to f_1 and f_6.

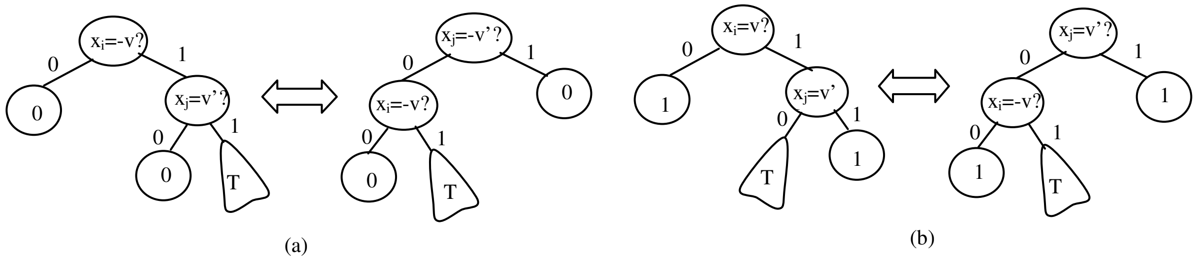

Fig. 14. Transformations needed for Lemma 8. (a) - Case 3.1; (b) - Case 3.2

3.2. $G(i, j)$ contains only truth tables having function value assignments according to f_2 and f_4.

The part of the original decision tree, starting from s, looks like the one showed in Fig. 14(a) for case 3.1, or like the one showed in Fig. 14(b) for case 3.2. As can be seen from this figure, in both cases this part of decision tree can be transformed to the equivalent part of decision tree testing x_j and then x_i, and having the same number of nodes as the original tree. This completes the proof of the case 3.

Proving the cases 1, 2, and 3 we showed that if $|a_i| > |a_j|$ then $|T_j| \leq |T_i|$. Note that if $|a_i| > |a_j|$ and $|T_j| < |T_i|$ then there is a contradiction with Lemma 6. Therefore $|T_i| = |T_j|$. By Lemmata 6 and 7 the optimal tree tests at s some variable $x_l \in \hat{X}(s)$. Therefore $|T_i| = |T_j| = |T_{opt}|$. $\square$

B Proofs of Lemmata from Section 4

B.1 Proof of Lemma 10

Let k be the number of positive examples in s and k_a be the number of positive examples in the a-son of s ($a \in \{0, 1\}$). Clearly, $k_0 + k_1 = k$. Thus if $k_0 = \frac{k}{2} + b$ then $k_1 = \frac{k}{2} - b$. Note that b is any integer example. Suppose that m examples arrive to s, half of them goes to s_0 and the other half goes to s_1. Thus $p(s) = \frac{k}{m}$, $p(s_0(i)) = \frac{k/2}{m/2} + \frac{b}{m/2} = p(s) + \frac{2b}{m}$ and $p(s_1(i)) = \frac{k/2}{m/2} - \frac{b}{m/2} = p(s) - \frac{2b}{m}$. $\square$

C Proofs of Lemma 12 and Theorem 2

C.1 Proof of Lemma 12

Since the examples are distributed uniformly the probability of arriving to the leaf of depth $l \leq h$ is $\frac{1}{2^l} = \frac{2^{h-l}}{2^h} = \frac{r_l}{2^h}$, where r_l is an integer between 0 and 2^h. Due to uniform distribution of examples, the probability of arriving to some leaf l is essentially the percentage of examples arriving to l amongst all 2^n possible examples. Each example may arrive to at most one leaf. Therefore $\Pr(f(x) = 1) = \sum_{\text{positive leafs } l} \text{probability of arriving to } l = \sum_{\text{positive leafs } l} \frac{r_l}{2^h} = \frac{r}{2^h}, \quad r \in \mathbb{Z}$. By the definition of probability, $0 \le r \le 2^h$. $\Box$

C.2 Proof of Theorem 2

The proofs presented below are independent of the type of function used to generate input examples. However they do depend on the specific form of impurity function used in the impurity sum computation.

Consider the choice of splitting variable according to the minimum approximate impurity sum. If the approximate impurity sums are computed within very small error ϵ' then exact and approximate impurity sums are equivalent, *i.e.* for all attributes x_i, x_j and all inner nodes s the following condition holds:

$$\mathrm{IS}(s, x_i, \phi) > \mathrm{IS}(s, x_j, \phi) \leftrightarrow \widehat{\mathrm{IS}}(s, x_i, \phi) > \widehat{\mathrm{IS}}(s, x_j, \phi)$$

Our goal in the proof of Theorem 2 is to find such ϵ'.

The rest of the proof has the following structure. In section C.2 we find the accuracy needed for the equivalence of exact and approximate balanced impurity sums. In section C.2 we find the accuracy needed for the equivalence of exact and approximate impurity sums. The section C.2 concludes the proof of Theorem 2.

Equivalence of Exact and Approximate Balanced Impurity Sums

Lemma C1 *Let $\phi(x)$ be Gini index or the entropy or the new index impurity function. If for all $1 \le i \le n$, and all inner nodes s, $\hat{p}(s_0(i))$ and $\hat{p}(s_1(i))$ are computed within accuracy $\epsilon = \frac{1}{2^{2h}}$, then*

$$\phi(p(s_0(i))) + \phi(p(s_1(i))) > \phi(p(s_0(j))) + \phi(p(s_1(j))) \Leftrightarrow \\ \phi(\hat{p}(s_0(i))) + \phi(\hat{p}(s_1(i))) > \phi(\hat{p}(s_0(j))) + \phi(\hat{p}(s_1(j)))$$

Proof By Lemmata 10 and 12, $p(s) = \frac{k}{2^h}$, $p(s_0(i)) = \frac{k+k_1}{2^h}$ and $p(s_1(i)) = \frac{k-k_1}{2^h}$. We assume that $k_1 \ge 0$, the proof for the case $k_1 < 0$ is almost the same.

Initially we consider the case of $0 \le p(s_1(j)) < p(s_1(i)) < p(s) < p(s_0(i)) < p(s_0(j)) \le \frac{1}{2}$. Due to Lemmata 12 and 10 this case implies that $\frac{4}{2^h} \le p(s_0(j)) < \frac{1}{2}$ and $h \ge 4$. Suppose the probabilities $p(s_0(i))$, $p(s_1(i))$, $p(s_0(j))$ and $p(s_1(j))$ are estimated within accuracy ϵ. Then $\phi(\hat{p}(s_1(i))) + \phi(\hat{p}(s_0(i))) \ge \phi(p(s_1(i)) - \epsilon) + \phi(p(s_0(i)) - \epsilon)$ and $\phi(\hat{p}(s_1(j))) + \phi(\hat{p}(s_0(j))) \le \phi(p(s_1(j)) + \epsilon) + \phi(p(s_0(j)) + \epsilon)$. Therefore to prove the case of $0 \le p(s_1(j)) < p(s_1(i)) < p(s) < p(s_0(i)) < p(s_0(j)) \le \frac{1}{2}$ it is sufficient to find ϵ satisfying

$$\phi(p(s_1(j)) + \epsilon) + \phi(p(s_0(j)) + \epsilon) < \phi(p(s_1(i)) - \epsilon) + \phi(p(s_0(i)) - \epsilon) \tag{1}$$

Combining the assumption $0 \le p(s_1(j)) < p(s_1(i)) < p(s) < p(s_0(i)) < p(s_0(j)) \le \frac{1}{2}$, definition of impurity function and Lemmata 12 and 10, we obtain that the differences $\phi(p(s_1(j)) + \epsilon) + \phi(p(s_0(j)) + \epsilon) - \phi(p(s_1(j))) + \phi(p(s_0(j)))$

and $\phi(p(s_1(i))) + \phi(p(s_0(i))) - \phi(p(s_1(i)) - \epsilon) - \phi(p(s_0(i)) - \epsilon)$ take maximal values when

$$p(s_1(j)) = 0, p(s_1(i)) = \frac{1}{2^h}, p(s) = \frac{2}{2^h}, p(s_0(i)) = \frac{3}{2^h} \text{ and } p(s_0(j)) = \frac{4}{2^h} \quad (2)$$

Thus the value of ϵ satisfying the inequality (1) for these values of probabilities would satisfy (1) for any values of $0 \leq p(s_1(j)) < p(s_1(i)) < p(s) < p(s_0(i)) < p(s_0(j)) \leq \frac{1}{2}$.

Initially we obtain ϵ satisfying (1) for Gini index. Rewriting (1) we obtain

$$\phi(p(s_1(i)) - \epsilon) - \phi(p(s_1(j)) + \epsilon) > \phi(p(s_0(j)) + \epsilon) - \phi(p(s_0(i)) - \epsilon) \quad (3)$$

But since $\phi(x)$ is concave the following two inequalities[1] hold:

$$\phi(p(s_1(i)) - \epsilon) - \phi(p(s_1(j)) + \epsilon) \geq \phi'(p(s_1(i)) - \epsilon)(p(s_1(i)) - p(s_1(j)) - 2\epsilon) \quad (4)$$

$$\phi(p(s_0(j)) + \epsilon) - \phi(p(s_0(i)) - \epsilon) \leq \phi'(p(s_0(i)) - \epsilon)(p(s_0(j)) - p(s_0(i)) + 2\epsilon) \quad (5)$$

Combining the inequalities (3), (4) and (5) we obtain further that to prove the lemma for any $0 \leq p(s_1(j)) < p(s_1(i)) < p(s) < p(s_0(i)) < p(s_0(j)) \leq \frac{1}{2}$ and Gini index it is sufficient to find ϵ satisfying

$$\frac{\phi'(p(s_1(i)) - \epsilon)}{\phi'(p(s_0(i)) - \epsilon)} > \frac{p(s_0(j)) - p(s_0(i)) + 2\epsilon}{p(s_1(i)) - p(s_1(j)) - 2\epsilon} \quad (6)$$

Substituting the values of probabilities (2) to (6) we obtain that finding ϵ satisfying

$$\frac{\phi'\left(\frac{1}{2^h} - \epsilon\right)}{\phi'\left(\frac{3}{2^h} - \epsilon\right)} > \frac{\frac{1}{2^h} + 2\epsilon}{\frac{1}{2^h} - 2\epsilon} \quad (7)$$

suffices for the proof of the case with $0 \leq p(s_1(j)) < p(s_1(i)) < p(s) < p(s_0(i)) < p(s_0(j)) \leq \frac{1}{2}$ and Gini index.

Let $a = \frac{1}{2^h}$. Writing the explicitly the derivative of Gini index and simplifying the resulting expression we obtain: $-2\epsilon^2 - \epsilon(1 - 4a) + a^2 > 0$. Solving the last inequality for ϵ and taking into account that $\epsilon > 0$ we obtain that

$$0 < \epsilon < -\frac{1}{4} + a + \frac{\sqrt{(1 - 4a)^2 + 8a^2}}{4} = B \quad (8)$$

It can be easily verified that $\sqrt{(1 - 4a)^2 + 8a^2} > (1 - 4a) + a^2$ for any $0 < a < 1$ (*i.e.* for any h). Therefore $B > a^2$ and the value $\epsilon = a^2 = \frac{1}{2^{2h}}$ satisfies (8).

Up to now we showed that if $0 \leq p(s_1(j)) < p(s_1(i)) < p(s) < p(s_0(i)) < p(s_0(j)) \leq \frac{1}{2}$ and the impurity function is Gini index then the estimation of all probabilities within accuracy $\epsilon = a^2 = \frac{1}{2^{2h}}$ leads to the equivalence of exact and approximate balanced impurity sum. Next we show that the same accuracy suffices also for the new index and the entropy impurity functions.

[1] Note that derivatives at inequalities (4) and (5) are taken correspondingly w.r.t. $p(s_1(i)) - \epsilon$ and $p(s_0(i)) - \epsilon$

We denote by ϕ_g, ϕ_n, ϕ_e correspondingly Gini index, the new index and entropy impurity functions. Writing the left part of (7) for $\phi_n(x)$ we obtain:

$$\frac{\phi_n'\left(\frac{1}{2^h}-\epsilon\right)}{\phi_n'\left(\frac{3}{2^h}-\epsilon\right)}=\frac{\phi_g'\left(\frac{1}{2^h}-\epsilon\right)}{\phi_g'\left(\frac{3}{2^h}-\epsilon\right)}\cdot\frac{\sqrt{\left(\frac{3}{2^h}-\epsilon\right)\left(1-\frac{3}{2^h}+\epsilon\right)}}{\sqrt{\left(\frac{1}{2^h}-\epsilon\right)\left(1-\frac{1}{2^h}+\epsilon\right)}}>\frac{\phi_g'\left(\frac{1}{2^h}-\epsilon\right)}{\phi_g'\left(\frac{3}{2^h}-\epsilon\right)}$$

Therefore $\epsilon=a^2=\frac{1}{2^{2h}}$ satisfies (7) when $\phi(x)=\phi_n(x)$.

Writing explicitly (1) with the values of (2) and ϵ we obtain that we need to prove the following inequality for the entropy impurity function:

$$f(h)=\phi_e\left(\frac{1}{2^{2h}}\right)+\phi_e\left(\frac{4}{2^h}+\frac{1}{2^{2h}}\right)<\phi_e\left(\frac{1}{2^h}-\frac{1}{2^{2h}}\right)+\phi_e\left(\frac{3}{2^h}-\frac{1}{2^{2h}}\right)=g(h) \quad (9)$$

It can be verified that both $f(h)$ and $g(h)$ are decreasing convex positive functions of h, $\lim_{h\to\infty} f(h)=0$ and $\lim_{h\to\infty} g(h)=0$. Since $f(h)$ and $g(h)$ are decreasing functions, $f'(h)<0$ and $g'(h)<0$. Moreover it can be verified that $g'(h)<f'(h)$ for $h\geq 4$. In addition the inequality (9) holds for $h=4$, *i.e.* $g(4)>f(4)$.

Recall that $f(h)$ and $g(h)$ are decreasing convex functions limited by zero when h goes to infinity. If $f(h)$ and $g(h)$ intersect two or more times then there must be values of h such that $g'(h)>f'(h)$. This is a contradiction an therefore $f(h)$ and $g(h)$ intersect at most one time. Since $g'(h)<f'(h)<0$, $g(h)$ approaches zero faster than $f(h)$. Thus if $f(h)$ and $g(h)$ have an intersection point at $h=h_1$ then the difference $|f(h)-g(h)|$ would increase as h goes from h_1 to infinity. This contradicts the fact that $\lim_{h\to\infty} f(h)=\lim_{h\to\infty} f(h)=0$. Therefore $f(h)$ and $g(h)$ do not intersect. Since $g(4)>f(4)$, for all $h\geq 4$ also holds $g(h)>f(h)$. Consequently the inequality (9) holds for all $h\geq 4$ and the value $\epsilon=\frac{1}{2^{2h}}$ satisfies (1) for entropy impurity function and any $0\leq p(s_1(j))<p(s_1(i))<p(s)<p(s_0(i))<p(s_0(j))\leq\frac{1}{2}$.

To this end we have completed the proof of the lemma for the case $0\leq p(s_1(j))<p(s_1(i))<p(s)<p(s_0(i))<p(s_0(j))\leq\frac{1}{2}$. Clearly, almost the same proofs may be obtained for the other cases of reciprocal configuration of $p(s_1(j))$, $p(s_1(i))$, $p(s)$, $p(s_0(i))$, $p(s_0(j))$ and 0.5. $\square$

Equivalence of Exact and Approximate Impurity Sums

Lemma C2 *Let $\phi(x)$ be Gini index or the entropy or the new index impurity function. If for all $1\leq i\leq n$ and all inner nodes s, the approximate probabilities $\hat{p}(s_0(i))$, $\hat{p}(s_1(i))$, $\widehat{Pr}[s_0(i)]$ and $\widehat{Pr}[s_1(i)]$ are computed within accuracy $\epsilon=\frac{1}{4\cdot 2^{4h}}$ then*

$$Pr[s_0(i)]\phi(p(s_0(i)))+Pr[s_1(i)]\phi(p(s_1(i)))>Pr[s_0(j)]\phi(p(s_0(j)))+Pr[s_1(j)]\phi(p(s_1(j)))\Leftrightarrow$$
$$\widehat{Pr}[s_0(i)]\phi(\hat{p}(s_0(i)))+\widehat{Pr}[s_1(i)]\phi(\hat{p}(s_1(i)))>\widehat{Pr}[s_0(j)]\phi(\hat{p}(s_0(j)))+\widehat{Pr}[s_1(j)]\phi(\hat{p}(s_1(j)))$$

Proof We consider initially the case of $p(s_a(i)) < \frac{1}{2}$, $a \in \{0,1\}$. If we approximate $p(s_a(i))$ ($a \in \{0,1\}$) within accuracy $\epsilon' = \frac{1}{2^{2h}}$, then $\frac{1}{2}\phi(p(s_a(i)) - \epsilon') < \Pr[s_a(i)]\phi(\hat{p}(s_a(i))) < \frac{1}{2}\phi(p(s_a(i)) + \epsilon')$. We showed at Lemma C1 that the existence of last two inequalities is sufficient for the equivalence of exact and balanced impurity sum. Consequently, if we approximate $\Pr[s_a(i)]$ within accuracy ϵ'' and $p(s_a(i))$ within accuracy $\frac{\epsilon'}{2}$, such that

$$\left(\frac{1}{2} + \epsilon''\right)\phi\Big(p(s_a(i)) + \frac{\epsilon'}{2}\Big) < \frac{1}{2}\phi(p(s_a(i)) + \epsilon') \tag{10}$$

and

$$\left(\frac{1}{2} - \epsilon''\right)\phi\Big(p(s_a(i)) - \frac{\epsilon'}{2}\Big) > \frac{1}{2}\phi(p(s_a(i)) - \epsilon') \tag{11}$$

then approximate and exact impurity sums would be equivalent. Next we find ϵ'' satisfying this condition. From the last two inequalities we obtain that

$$\epsilon'' < \min_{p,i}\left\{\frac{\phi(p(s_a(i)) + \epsilon')}{2\phi(p(s_a(i)) + \epsilon'/2)} - \frac{1}{2}, -\frac{\phi(p(s_a(i)) - \epsilon')}{2\phi(p(s_a(i)) - \epsilon'/2)} + \frac{1}{2}\right\} \tag{12}$$

Due to the concavity of impurity function

$$\frac{\phi(p(s_a(i)) + \epsilon')}{\phi(p(s_a(i)) + \epsilon'/2)} \geq \frac{\phi\big(\frac{1}{2}\big)}{\phi\big(\frac{1}{2} - \frac{\epsilon'}{2}\big)} \text{ and } \frac{\phi(p(s_a(i)) - \epsilon')}{\phi(p(s_a(i)) - \epsilon'/2)} \leq \frac{\phi\big(\frac{1}{2} - \frac{\epsilon'}{2}\big)}{\phi\big(\frac{1}{2}\big)} \tag{13}$$

The quantities $A = \frac{\phi\left(\frac{1}{2}\right)}{\phi\left(\frac{1}{2} - \frac{\epsilon'}{2}\right)}$ and $B = \frac{\phi\left(\frac{1}{2} - \frac{\epsilon'}{2}\right)}{\phi\left(\frac{1}{2}\right)}$ depend on the specific form of impurity function. Here we provide the proof for three commonly used impurity functions, namely Gini index, the entropy and the new index. It can be easily verified that amongst these impurity functions A has the smallest value at the new index and B has the greatest value at the new index too. Consequently the bounds on accuracy found for the new index are also valid for the other two impurity functions. Thus in the rest of the proof we deal only with the new index impurity function.

Writing down the explicit expression of the new index in the formulas of A and B, substituting A and B to (13) and finally substituting (13) to (12) we obtain:

$$\epsilon'' < \min\left\{\frac{1 - \sqrt{1 - \epsilon'^2}}{2\sqrt{1 - \epsilon'^2}}, \frac{1}{2} - \frac{1}{2}\sqrt{1 - \epsilon'^2}\right\}$$

Using the inequality $\sqrt{1 - x^2} < 1 - \frac{x^2}{2}$ for $|x| < \sqrt{2}$ we obtain $\frac{1-\sqrt{1-\epsilon'^2}}{2\sqrt{1-\epsilon'^2}} > \frac{1-\sqrt{1-\epsilon'^2}}{2} > \frac{\epsilon'^2}{4}$ and $\frac{1}{2} - \frac{1}{2}\sqrt{1 - \epsilon'^2} > \frac{\epsilon'^2}{4}$. Thus the value $\epsilon'' < \frac{\epsilon'^2}{4} = \frac{1}{4 \cdot 2^{4h}}$ satisfies the equations (12), (10) and (11).

To this point we have completed the proof for the case $p(s_a(i)) < \frac{1}{2}$. Clearly, the proof for the case of $p(s_a(i)) > \frac{1}{2}$ is almost identical. Thus the last case we have to consider is $p(s_a(i)) = \frac{1}{2}$.

In the case of $p(s_a(i)) = \frac{1}{2}$ we seek ϵ'' satisfying (11) and

$$\Big(\frac{1}{2} + \epsilon''\Big)\phi\Big(p(s_a(i)) + \frac{\epsilon'}{2}\Big) > \frac{1}{2}\phi(p(s_a(i)) + \epsilon') \tag{14}$$

Substituting $p(s_a(i)) = \frac{1}{2}$ to (11) and (14) we obtain

$$\frac{1}{2} - \frac{\phi\big(\frac{1}{2} - \epsilon'\big)}{2\phi\big(\frac{1}{2} - \frac{\epsilon'}{2}\big)} > \epsilon'' > \frac{\phi\big(\frac{1}{2} + \epsilon'\big)}{2\phi\big(\frac{1}{2} + \frac{\epsilon'}{2}\big)} - \frac{1}{2} \tag{15}$$

It can be verified that for the three commonly used impurity functions the fractions $\frac{\phi\left(\frac{1}{2}-\epsilon'\right)}{2\phi\left(\frac{1}{2}-\frac{\epsilon'}{2}\right)}$ and $\frac{\phi\left(\frac{1}{2}+\epsilon'\right)}{2\phi\left(\frac{1}{2}+\frac{\epsilon'}{2}\right)}$ take the greatest value when $\phi(x)$ is the new index. Therefore without loss of generality we may restrict ourselves to the new index impurity function. Writing down the explicit expression of the new index in 15, simplifying and using again the inequality $\sqrt{1-x^2} < 1 - \frac{x^2}{2}$ for $|x| < \sqrt{2}$ we obtain:

$$\frac{1}{4}\epsilon'^2 > \epsilon'' > -\frac{1}{4}\epsilon'^2 \tag{16}$$

Obviously, the value $0 < \epsilon'' < \frac{1}{4\cdot 2^{4h}}$, found for the case $p(s_a(i)) < \frac{1}{2}$, would satisfy (16). $\square$

Final Part of the Proof

Lemma C3 *Suppose for all $1 \leq i \leq n$ and all inner nodes s, the approximate probabilities $\hat{p}(s_0(i))$, $\hat{p}(s_1(i))$, $\widehat{Pr}[s_0(i)]$ and $\widehat{Pr}[s_1(i)]$ are computed within accuracy $\epsilon = \frac{1}{4\cdot 2^{4h}}$. Let the impurity function be Gini index, entropy or the new index. Let $f(x)$ be a read-once DNF or a boolean linear threshold function. Then the algorithm **DTApproxPG** builds decision tree representing $f(x)$ and having minimal size and minimal height amongst all decision trees representing $f(x)$.*

Proof Consider the root inner node s of the tree T built by **DTExactPG**. Let x_i be the splitting variable at s and let X be the set of all variables available for testing at s. There are two cases to be considered:

1. $\forall x_j \in X$ such that $i \neq j$, $\mathrm{IS}(s, x_j, \phi) > \mathrm{IS}(s, x_i, \phi)$. Then, by Lemma C2, $\widehat{\mathrm{IS}}(s, x_j, \phi) > \widehat{\mathrm{IS}}(s, x_i, \phi)$ and the algorithm **DTApproxPG** chooses x_i as a splitting variable at s too.
2. $\exists\, X' = \{x_{j_1}, x_{j_2}, \ldots, x_{j_k}\} \subset X$ such that $x_i \notin X'$ and $\forall x_j \in X'$ $\mathrm{IS}(s, x_j, \phi) = \mathrm{IS}(s, x_i, \phi)$. It follows from Theorem 1 that in this case the choice of any variable from $X'' = X' \cup \{x_i\}$ by **DTExactPG** would lead to the decision tree of the same size and height (which according to Theorem 1 are also minimal) representing $f(x)$. Let x_r be the variable having maximal approximate impurity sum. According to Lemma C2, $x_r \in X''$.

In both cases **DTApproxPG** chooses splitting variable x_r at the root s such that there exists minimal size and minimal height decision tree representing

$f(x)$ and splitting at s according to x_r. The same argument holds also for other inner nodes, which are descendants of s. Combining this result with the stopping condition of **DTApproxPG** we may conclude that the decision tree built by this algorithm represents $f(x)$ and has a minimal size and minimal height amongst all decision trees representing $f(x)$. $\square$

Lemma C4 *For any $\delta > 0$, given $O\big(2^{9h} \ln^2 \frac{1}{\delta}\big)$ uniformly distributed noise-free random examples of $f(x)$, with probability at least $1 - \delta$, for each inner node s the probabilities $\hat{p}(s_0(i))$, $\hat{p}(s_1(i))$, $\widehat{Pr}[s_0(i)]$ and $\widehat{Pr}[s_1(i)]$ may be computed within accuracy $\epsilon \leq \frac{1}{4 \cdot 2^{4h}}$.*

Proof Let h_s be the depth of inner node s. According to the Hoeffding bound ([13]), to compute each of the probabilities $\hat{p}(s_0(i))$, $\widehat{\Pr}[s_0(i)]$, $\hat{p}(s_1(i))$ and $\widehat{\Pr}[s_1(i)]$ within accuracy $\epsilon \leq \frac{1}{4 \cdot 2^{4h}}$ with probability $1 - \delta$, $O(2^{8h} \ln \frac{1}{\delta})$ random examples arriving correspondingly at s_0 and s_1 would suffice. Since all examples are uniformly distributed the probability that a random example would arrive at s_0 (or s_1) is no less than $\frac{1}{2^h}$. Therefore, according to Chernoff bound ([8]), to obtain $O(2^{8h} \ln \frac{1}{\delta})$ random examples arriving at s_0 (or s_1) $O(2^{9h} \ln^2 \frac{1}{\delta})$ random examples would suffice.

There are at most 2^h nodes in the resulted tree. Thus substituting $\frac{\delta}{2^h}$ instead of δ in the above computation we still obtain that $O(2^{9h} \ln^2 \frac{1}{\delta})$ random examples would suffice to compute all probabilities with the desired level of accuracy in all inner nodes. $\square$

Observe (see [4] and [25]) that the probabilities $\hat{p}(s_0(i))$, $\widehat{\Pr}[s_0(i)]$, $\hat{p}(s_1(i))$ and $\widehat{\Pr}[s_1(i)]$ are estimated by impurity-based decision tree algorithms exactly in the same way as they are estimated by the Hoeffding Lemma, which is used in the proof of Lemma C4. Combining this fact with Lemmata C3 and C4 we obtain the Theorem 2.

D Algorithm DTStatQuery and Proof of Theorem 3

Let $\widehat{\Pr}[f_R = 1](\alpha)$ be a statistical query of $\Pr[f_R = 1]$ within accuracy α. We refer to the computation of approximate purity gain, using the calls to statistical queries within accuracy α, as $\widehat{PG}(f_R, x_i, \phi, \alpha)$. The algorithm **DTStatQuery**, which is a reformulation of **DTApproxPG** in terms of statistical queries, is shown at Fig. 15.

Lemma 13 *Let $f(x)$ be a read-once DNF or a boolean linear threshold function. Then, for any impurity function, the decision tree,built by the algorithm **DT-StatQuery**, represents $f(x)$ and has minimal size and minimal height amongst all decision trees representing $f(x)$.*

Proof The stopping condition of **DTApproxPG**, states "if all examples arriving at s have the same classification then set s a leave with classification as that of all examples arriving at it". Instead of checking explicitly classification

DTStatQuery(s, X, R, ϕ, h)

1: **if** $\widehat{\Pr}[f_R = 1]\left(\frac{1}{2 \cdot 2^h}\right) > 1 - \frac{1}{2 \cdot 2^h}$ **then**
2: Set s as a positive leaf.
3: **else**
4: **if** $\widehat{\Pr}[f_R = 1]\left(\frac{1}{2 \cdot 2^h}\right) < \frac{1}{2 \cdot 2^h}$ **then**
5: Set s as a negative leaf.
6: **else**
7: Choose $x_i = \arg\max_{x_i \in X}\{\widehat{PG}(f_R, x_i, \phi, \frac{1}{4 \cdot 2^{4h}})\}$ to be a splitting variable.
8: Run **DTStatQuery**(s_1, $X - \{x_i\}$, $R \cup \{x_i = 1\}$, ϕ, h).
9: Run **DTStatQuery**(s_0, $X - \{x_i\}$, $R \cup \{x_i = 0\}$, ϕ, h).
10: **end if**
11: **end if**

Fig. 15. DTStatQuery algorithm. h is a minimal height of decision tree representing the function.

of all examples we may try to estimate the probability of positive examples. In Lemma 12 we proved that such probability can be written always in the form $\frac{k}{2^h}$, where $k \in \mathbb{Z}^+$ and h is a minimal height of decision tree representing the function. Suppose we estimate the probability of positive examples within accuracy $\frac{1}{2 \cdot 2^h}$. Then, if the estimated probability is greater than $1 - \frac{1}{2 \cdot 2^h}$ then surely all examples arriving at the inner node are positive. Similarly, if the estimated probability is smaller than $\frac{1}{2 \cdot 2^h}$ then surely all examples arriving at the inner node are negative. Consequently we may use statistical query $\widehat{\Pr}[f_R = 1]\left(\frac{1}{2 \cdot 2^h}\right)$, returning the probability of f_R to be 1 within accuracy $\frac{1}{2 \cdot 2^h}$, as a stopping condition. The statistical query stopping rule is shown at lines 1 and 3 of Fig. 15.

The splitting criterion of **DTApproxPG** is the choice of variable with maximal approximate purity gain, which in turn is equivalent to minimum approximate impurity sum. The computation of impurity sum involves the estimation of probabilities $\hat{p}(s_0(i))$, $\hat{p}(s_1(i))$, $\widehat{\Pr}[s_0(i)]$ and $\widehat{\Pr}[s_1(i)]$ (see Appendix C.2). Clearly, the estimations of probabilities $p(s)$ and $\Pr[s_a(i)]$ ($a \in \{0, 1\}$), made by the algorithm **DTApproxPG** may be changed to the calls to statistical queries returning the estimations within required accuracy. In particular we proved in Lemma C3 that if we obtain the values of these probabilities within accuracy $\frac{1}{4 \cdot 2^{4h}}$ then the approximate purity gain splitting rule chooses at each step the variable leading to the smallest decision tree representing the function. $\square$

Kearns shows in [17] how to simulate statistical queries from examples corrupted by small classification noise. [2] This simulation involves the estimation of noise rate η. [17] shows that if statistical queries need to be computed within accuracy α then η should be estimated within accuracy $\Delta/2 = \Theta(\alpha)$. Such an

[2] Aslam and Decatur show in [2] more efficient procedure for simulation of statistical queries from noisy examples. However their procedure needs the same adaptation to PC learning model as that of Kearns [17]. For the sake of simplicity we show the adaptation to PC model with the procedure of [17]. The procedure of [2] may be adapted to PC model in the similar manner.

DTNoiseTolerant

Let $\Delta = \Theta(\frac{1}{4 \cdot 2^{4h}})$.
for $i = 0$ to $\lceil \frac{1}{2\Delta} \rceil$ **do**
    Run **DTStatQueryPG**(f,X,$\emptyset$,ϕ,h). Simulate each statistical query made by algorithm using noisy examples according to the technique of [17] with noise rate $\hat{\eta}_i = i\Delta$.
    Let h_i be the hypothesis returned by the algorithm.
    Compute estimation $\hat{\gamma}_i$ of $\gamma_i = \Pr_{EX^{\eta}(\mathcal{U})}[h_i(x) \neq f(x)]$ within accuracy $\frac{\Delta}{4}$.
end for
Let $H = \{i \mid |\hat{\gamma}_i - i\Delta| < \frac{3\Delta}{4}\}$.
if $|H| = 1$ **then**
    $H = \{j\}$. Output h_j.
else
    Let i_1 and i_2 be two lowest numbers in H.
    if $|i_1 - i_2| > 1$ or $|\hat{\gamma}_{i_1} - \hat{\gamma}_{i_2}| > \frac{\Delta}{2}$ **then**
        $j = \min\{i_1, i_2\}$. Output h_j.
    else
        Let $\hat{\eta} = \frac{\hat{\gamma}_{i_1} + \hat{\gamma}_{i_2}}{2}$.
        Run **DTStatQueryPG**(f,X,$\emptyset$,ϕ,h) with noise rate $\hat{\eta}$ and output the resulted hypothesis.
    end if
end if

Fig. 16. DTNoiseTolerant algorithm. h is a minimal height of decision tree representing the function.

estimation may be obtained by taking $\lceil \frac{1}{2\Delta} \rceil$ estimations of η of the form $i\Delta$, $i = 0, 1, \ldots, \lceil \frac{1}{2\Delta} \rceil$. Running the learning algorithm using each time different estimation we obtain $\lceil \frac{1}{2\Delta} \rceil + 1$ hypotheses $h_0, h_1, \ldots, h_{\lceil \frac{1}{2\Delta} \rceil}$. By the definition of Δ, amongst these hypotheses there exists at least one hypothesis h_j having the same generalization error as the statistical query algorithm. Then [17] describes a procedure how to recognize the hypothesis having generalization error of at most ϵ. The naïve approach to recognize the minimal sized decision tree having zero generalization error amongst $h_0, \ldots, h_{\lceil \frac{1}{2\Delta} \rceil}$ is to apply the procedure of [17] with $\epsilon = \frac{1}{2 \cdot 2^n}$. However in this case this procedure requires about 2^n noisy examples. Next we show how to recognize minimal size decision tree with zero generalization error using only $poly(2^h)$ uniformly distributed noisy examples.

Amongst $\lceil \frac{1}{2\Delta} \rceil$ estimations $\hat{\eta}_i = i\Delta$ of η there exists $i = j$ such that $|\eta - j\Delta| \leq \Delta/2$. Our current goal is to recognize such j.

Let $\gamma_i = \Pr_{EX^{\eta}(\mathcal{U})}[h_i(x) \neq f(x)]$ be the generalization error of h_i over the space of uniformly distributed noisy examples. Clearly, $\gamma_i \geq \eta$ for all i, and $\gamma_j = \eta$. Let $\hat{\gamma}_i$ be the estimation of γ within accuracy $\Delta/4$. Then $|\gamma_j - j\Delta| < \frac{3\Delta}{4}$. Let $H = \{i \mid |\hat{\gamma}_i - i\Delta| < \frac{3\Delta}{4}\}$. Clearly $j \in H$. Therefore if $|H| = 1$ then H contains only j. Consider the case of $|H| > 1$. Since for all i $\gamma_i \geq \eta$, if $i \in H$ then $i \geq j - 1$. Therefore one of the two minimal values in H is j. Let i_1 and i_2 be two minimal values in H. If h_{i_1} and h_{i_2} are the same tree then clearly they are the one with smallest size representing the function. If $|i_1 - i_2| > 1$ then, using the argument $i \in H \rightarrow i \geq j-1$, we get that $j = \min\{i_1, i_2\}$. If $|i_1 - i_2| = 1$ and $|\hat{\gamma}_{i_1} - \hat{\gamma}_{i_2}| \geq \frac{\Delta}{2}$, then, since the accuracy of $\hat{\gamma}$ is $\Delta/4$, $j = \min\{i_1, i_2\}$. The final subcase to be considered is $|\hat{\gamma}_{i_1} - \hat{\gamma}_{i_2}| < \frac{\Delta}{2}$ and $|i_1 - i_2| = 1$. In this case $\hat{\eta} = (\hat{\gamma}_{i_1} + \hat{\gamma}_{i_2})/2$ estimates the true value of η within accuracy $\Delta/2$. Thus running the learning algorithm with the value $\hat{\eta}$ for noise rate produces the same tree as the one produced by statistical query algorithm.

Figure 16 summarizes our noise-tolerant version (named **DTNoiseTolerant**) of impurity-based algorithms. It can be shown that to simulate statistical queries and recognize hypothesis with zero generalization error all estimations should be done within accuracy $poly(\frac{1}{2^h})$. Thus the sample complexity of the algorithm **DTNoiseTolerant** is the same as in **DTApproxPG**. Consequently, Theorem 3 follows.