# On learning Decision Committees

**Richard Nock** **Olivier Gascuel**
Dépt. d'Informatique Fondamentale
LIRMM, 161 rue Ada
34392 Montpellier Cedex 5, France
{nock,gascuel}@lirmm.fr

## Abstract

This paper presents decision committees. A decision committee contains rules, each of these beeing a couple (monomial, vector). Each monomial is a condition that, when matched by an instance, returns its vector. When each monomial is tested, the sum of the returned vectors is used to take the classification decision. We show that for every constant $k$, the subclass of decision committees whose elements have monomial of length $\leq k$ is *pac*-learnable and that it properly contains $k$-DL. However, we also show that the problem of inducing the shortest consistent decision committee is NP-Hard. This leads to theoretical results on non-learnability, and to negative considerations for practical optimization problems on decision committees. A two-stages heuristic algorithm, IDC, is presented, that learns by a particular subclass of decision committees. It first chooses monomials by a breadth-first search inspired from branch-and-bound algorithms. Then it clusters gradually the resulting rules to form decision committees, according to the minimization of empirical risk. Finally it selects the decision committee over the final population, which is the best according to the learning sample. Experimental results on 15 artificial and real domains tend to show that IDC achieves good results, while constructing small, and interpretable decision committees.

## 1 Introduction

A decision committee contains rules, each of these beeing a couple (monomial, vector). Each monomial is a condition that, when fired, returns its vector. After each monomial has been tested, the sum of the returned vectors is used to take the classification decision. A decision committee combines rules in an additive fashion, like in MYCIN-type expert systems (Shortliffe [Sho76]). The main idea is to allow the absence of any underlying ordering in the decision procedure (unlike the ordering of literals in a decision tree, or the ordering of rules in a decision list), and to take advantage of *multiple knowledge* (Gams [Gam89], Kononenko and Kovačič [KK92]).

Decision committees may be viewed as a generalization of Threshold Functions (Bruck [Bru90]) in the multiclass case. They are also a very special case of Neural Networks, where there would be constraints over the activation functions and the network architecture. The idea of combining the decision of rules is not new (see for example Nilsson [Nil65], Bongard [Bon70], Quinqueton and Sallantin [QS83], Cestnik and Bratko [CB88], Gams [Gam89], Gascuel [Gas89], Kononenko and Kovačič [KK92]). Decision committees can be viewed as a formalism that allows to express classes of concepts using shared knowledge such as Sum of distribution, Voting method of Kononenko and Kovačič [KK92], Voting method of Gascuel [Gas89]. Our aim is to use this formalization to provide theoretical *pac*-learnability results regarding classes of decision committees having practical and theoretical relevance.

One essential problem in systems that use shared knowledge is how to combine rules to classify examples. If we restrict the monomials to single monotonous literals, this problem is subsumed by that of linear discrimination. Whithout this restriction, the problem appears to be harder and a good practical illustration of its difficulty is given by Kononenko and Kovačič [KK92]: indeed, they observe that their Voting and Sum of distribution methods give "bad or unstable" results. We propose a two-stages algorithm, IDC (Induction of Decision Committees), that constructs decision committees from a particular subclass chosen for its practical relevance. Firstly IDC isolates rules in a very general way, each rule beeing judged only on its intrinsic properties and on the whole learning sample. Secondly IDC constructs from a global population of rules some decision committees by clustering rules (or decision committees) gradually, according to a simple criterion related to the principle of minimization of empirical risk (Vapnik, [Vap82]).

Section §2 is devoted to definitions and theoretical results on learnability of decision committees. Section §3 details the components of IDC. Section §4 presents experimental results of IDC on publically available databases, which are discussed in section §5.

## 2 Definitions and theoretical results

As the reader shall see, decision committees have several good properties: we show that if we limit the size of each monomial appearing in a decision committee to a constant $k$, then the resulting class is *pac*-learnable (in the sense of Valiant [Val84]). For any constant $k$, we also show that this subclass properly contains $k$-DL, the set of decision lists where the size of each monomial is $\leq k$, Rivest [Riv87]. We also study subclasses of decision committees having higher practical interest, *e.g.* those whose elements have limited size. We show that, like for many classes of Boolean formulae (Rivest [Riv87], Kearns *et al.* [KLPV87a], Pitt and Valiant [PV88], Hancock *et al.* [HJLT95]) the practical aim which is to obtain a good compromise between complexity and goodness-of-fit is intractable for decision committees. This leads to negative results for *pac*-learnability of the corresponding subclasses. This also justifies the heuristic IDC we use in Section §3 to construct these decision committees.

### 2.1 Notations

Let $c$ be the number of classes. A decision committee contains two parts:

- A set of unordered couples (or rules) $\{(t_i, \vec{v}_i)\}$, where each $t_i$ is a monomial (a conjunction of literals) over $\{0, 1, *\}^n$ ($n$ being the number of description variables), and each $\vec{v}_i$ is a vector in $\mathbb{R}^c$ (in the two-classes case, it is sufficient to add a single number rather than a 2-component vector).
- A Default Vector $\vec{\mathrm{D}}$ in $[0,1]^c$. Again, in the two-classes case, the reader shall remark that $\vec{\mathrm{D}}$ can be replaced by a default class $\in \{+,-\}$.

For any example $e$ and any monomial $t_i$, the proposition "$e$ satisfies $t_i$" is denoted by $e \Rightarrow t_i$. The opposite proposition "$e$ does not satisfy $t_i$" is denoted by $e \not\Rightarrow t_i$.

The classification of any example $e$ is made in the following way: define $\vec{V}_e$ as follows:

$$\vec{V}_e = \sum_{(t_i,\vec{v}_i)|e\Rightarrow t_i} \vec{v}_i$$

The class assigned to $e$ is then

- $\arg\max_{1\leq j\leq c} \vec{V}_e[j]$ if $|\arg\max_{1\leq j\leq c} \vec{V}_e[j]| = 1$.
- $\arg\max_{j\in\arg\max_k \vec{V}_e[k]} \vec{\mathrm{D}}[j]$ otherwise.

In other words, if the maximal component of $\vec{V}_e$ is unique then its index gives the class assigned to $e$. Otherwise, we take the index of the maximal component of $\vec{\mathrm{D}}$ corresponding to the maximal components of $\vec{V}_e$.

We call DC the whole class of decision committees. Define $\forall k \leq n$, $k$-DC to be the subclass of DC where each element has monomials of length $\leq k$. Furthermore, $\forall k > 0$, $k$-complex-DC denotes the subset of DC where each element has a total number of literals $\leq k$ (if a literal appears $l$ times, it is counted $l$ times).

Let $\mathcal{U} \subseteq \mathbb{R}$ and $\mathcal{F} \in \{\mathrm{DC}, k\text{-DC}, k\text{-complex-DC}\}$. $\mathcal{F}_\mathcal{U}$ denotes the subset of $\mathcal{F}$ where for each decision committee and each rule, $\vec{v}_i \in \mathcal{U}^c$ (*e.g.* $\mathcal{U} = \{-1,0,1\}, \mathbb{R}^+$). For example, 3-DC$_{\mathbb{R}^+}$ contains all the decision committees whose monomials contains at most 3 literals each, and whose vectors are elements of $\mathbb{R}^{+^c}$.

The previous definition leads to the following remark: there is no point in fixing for $\mathcal{U}$ sets such as $\{-1,0,1\}$ for classes where no bound is fixed on the size of a decision committee (DC, $k$-DC), if we allow repetitions of rules. Indeed, we artificially increase the power of the decision committees, and, by this way, can get "as close as" desired to real-valued vectors. Therefore, when we write DC$_{\{-1,0,1\}}$, $k$-DC$_{\{-1,0,1\}}$, this implicitely means that no repetition of rules is allowed. When we speak of other subclasses having limited-size formulae like $k$-complex-DC$_{\{-1,0,1\}}$, we add whether or not such repetitions are allowed.

The previous definitions are general ones. Subsections §2.2 and §2.3 take place in the Boolean context, *i.e.* $c = 2$. Therefore, we assume without loss of generality that each vector is replaced by a number, and that $\vec{\mathrm{D}}$ is replaced by a default class $\in \{+,-\}$. In order to preserve homogeneous notations, in the following Subsections, we keep definitions for $\mathcal{U}$ such as $\mathcal{U} = \{-1,0,1\}$, even if "0" is not necessary in the Boolean framework. This is due to the fact that we make a large use of $\mathcal{U} = \{-1,0,1\}$, particularly in our applications.

### 2.2 $k$-DC generalizes $k$-DL, and it is learnable

It is well known that for every constant $k > 0$, $k$-DNF (disjunctive normal forms with monomials whose size is $\leq k$), $k$-CNF (conjunctive normal forms with clauses having size $\leq k$) and $k$-DL are *pac*-learnable (Valiant [Val84], Rivest [Riv87]). It is also shown in Rivest [Riv87] that $k$-DL properly contains $k$-DNF and $k$-CNF (which properly contains $k$-DT, that is, depth-$k$ decision trees). We show that for every constant $k > 0$, $k$-DC properly contains $k$-DL, and furthermore that it is learnable:

**Proposition 1** *$\forall k > 0$ constant, $k$-DL $\subset$ $k$-DC, and $k$-DC is pac-learnable.*

**Proof** Let $f = \{(t_1, c_1), (t_2, c_2), ..., (t_l, c_l)\}$ be a $k$-decision list. To see that $k$-DL $\subseteq$ $k$-DC, It is sufficient to associate to $t_i$ a sufficiently high integer greater than the sum of the integers associated to the monomials $t_j, i+1 \leq j \leq l$. This ensures that the first monomial satisfied by an example in $f$ is the one that achieves the decision in the decision committee. To see that this inclusion is strict, we show that 1-DC$_{\{-1,0,1\}} \not\subset$ $k$-DL, for every constant $k > 0$. Fix $n > 2k$ an even integer. Consider the following 1-DC$_{\{-1,0,1\}}$:

$$sign(\sum_{j=1}^{\frac{n}{2}} x_j - \sum_{j=\frac{n}{2}+1}^{n} x_j)$$

The choice of the default class replacing $\vec{\text{D}}$ is not important. Every monomial using fewer than $\frac{n}{2}$ literals classifies at least one positive and one negative example, so the first monomial of every $k$-DL is bound to make mistakes, whatever the class to which it is associated.

Now, for the *pac*-learnability property, note that when $k$ is fixed, the number of monomials of length $\leq k$ is

$$\sum_{j=1}^{k} 2^j \binom{n}{j} = \mathcal{O}(n^k)$$

Thus, instead of expressing the examples in $\{0,1\}^n$, we can express them in $\{0,1\}^{\mathcal{O}(n^k)}$. This transformation is polynomial in $n$, and now the aim is only to find a linear separator in this $\mathcal{O}(n^k)$-dimensional space. This can be done using the polynomial-time algorithm of Karmarkar [Kar84] for linear programming. □

*Remark:* the integers chosen in the preceding proof to transform the decision list into a decision committee are exponential in $l$. In our applications, we have chosen $\mathcal{U} = \{-1,0,1\}$, without allowing repetitions of rules. This leads to a subclass of decision committees which cannot be used to check the inclusion relationship of Proposition 1. Nevertheless, this restriction allows us to preserve a useful property. The most used, and practically-efficient, Boolean formulae are decision trees, and we have

$$\forall 0 < k < n, k\text{-DT} \subset k\text{-DC}_{\{-1,0,1\}}$$

To check this, replace each path from the root to a leaf by the corresponding monomial associated to $+1$ or $-1$ whether the corresponding leaf is labeled "+" or "−". This leads to a formula $\in$ $k$-DC$_{\{-1,0,1\}}$. The inclusion is strict since $k$-DNF and $k$-CNF are also properly contained in $k$-DC$_{\{-1,0,1\}}$.

## 2.3 Negative results encountered in practice

A learning problem can be stated as an optimization problem. The aim is to find a decision procedure consistent with sufficiently big parts of the training sample, but also sufficiently small (this leads to Occam's principle, Murphy and Pazzani [MP94], Turán [Tur91]). Usually, we hope to find one presenting a good compromise between these two properties. In fact, like for many other classes of Boolean formulae (Hyafil and Rivest [HR76], Rivest [Riv87], Kearns *et al.* [KLPV87a], Hancock *et al.* [HJLT95]) this aim is also intractable for decision committees:

**Proposition 2** *It is $NP$-Hard to find the smallest decision committee consistent with a set of examples, for any fixed $\mathcal{U}$, with or without allowing repetitions of rules.*

Here, "size" can either mean whole number of literals (reduction from the "Exact Cover By 3-Sets" problem, Garey and Johnson [GJ79]) or number of rules (reduction from the "2-NM-Colorability" problem, Kearns *et al.* [KLPV87a], Kearns *et al.* [KLPV87b], Pitt and Valiant [PV88]).

Furthermore, in practice, interpretability means restrictions over $\mathcal{U}$. For the expert's convenience, we have decided to retain a small number of integers allowing natural and simple interpretations of the rules such as: "this rule is in favor (disfavor) of this class", "this rule is likely to be not correlated with this class". That's why we fix in our applications $\mathcal{U} = \{-1,0,1\}$ (and so we do not allow repetitions of rules). This additional restriction intuitively makes the learning task harder. A good illustration in the Boolean framework is the following:

**Proposition 3** *Unless $RP = NP$, 1-DC$_{\{-1,0,1\}}$ is not pac-learnable.*

(Reduction from "Zero-One Integer Programming", Kearns *et al.* [KLPV87a]). Thus, this limitation over $\mathcal{U}$ makes the positive result of Proposition (1) become negative. In a way, the negative results of proposition (2) and (3) justify the use of the heuristic IDC we give in the following section.

## 3 Learning by $DC_{\{-1,0,1\}}$: the IDC algorithm

In order to preserve their interpretability, we have chosen to construct decision committees where:

1. The vectors are elements of $\{-1,0,1\}^c$.
2. The rules appear at most once.

This section is devoted to the presentation of our heuristic algorithm: IDC. The first part shows how IDC constructs rules (*i.e.* chooses the monomials and calculates the corresponding vectors). The second part presents our clustering algorithm, which leads to the hypothesis decision committee.

## 3.1 Constructing rules

### 3.1.1 Choice of monomials

The algorithm we use to choose the monomials is a version restricted to Boolean representations of an algorithm implemented in the software PLAGE (Gascuel [Gas86], [Gas89]). This algorithm is based on a breadth-first search, inspired from Branch-and-Bound algorithms. It uses the idea that a good rule must present a good compromise between simplicity and goodness-of-fit. Indeed, every rule $R$ is judged on the base of two criteria:

1. The simplicity criterion (or coverage) $\mathcal{S}(R)$ which is the number of examples satisfying the rule.
2. The $\chi^2$ criterion, which is a measure of the discriminant capacity of the rule.

We construct for $R$ a $2\times c$ table as below:

| | | | |
|---|---|---|---|
| $n_{t1}$ | $n_{t2}$ | $\cdots$ | $n_{tc}$ |
| $n_{f1}$ | $n_{f2}$ | $\cdots$ | $n_{fc}$ |

Where $n_{ti}$ is the number of training examples of class $i$ that satisfy $R$, and $n_{fi}$ is the number of training examples of class $i$ that do not satisfy $R$. Thus $n_{ti} + n_{fi} = |c_i|$, where $|c_i|$ is the cardinality of the class $c_i$ in the training sample. Then we have:

$$\mathcal{S}(R) = \sum_{j=1}^{c} n_{tj}$$

and

$$\chi^2(R) = |LS| \left( \sum_{j=1}^{c} \frac{n_{tj}^2}{|c_j|\mathcal{S}(R)} + \sum_{j=1}^{c} \frac{n_{fj}^2}{|c_j|(|LS| - \mathcal{S}(R))} \right) - |LS|$$

where $|LS|$ is the size of the learning sample. The user fixes two thresholds: $s_{\mathcal{S}}$, the simplicity threshold, and $s_{\chi^2}$, the $\chi^2$ threshold. The algorithm ensures that every selected rule $R$ satisfies (1) $\mathcal{S}(R) \geq s_{\mathcal{S}}$, (2) $\chi^2(R) \geq s_{\chi^2}$ and no generalization of this rule satisfies them. The algorithm starts with the list $L_1$ of rules whose monomials have length 1. Every selected rule of $L_1$ (satisfying (1) and (2)) is put into the final list of selected rules $L_{end}$. It is obvious that every rule $R$ satisfying $\mathcal{S}(R) < s_{\mathcal{S}}$ can be removed from $L_1$, since specializing this rule cannot increase its simplicity. Now, the remaining rules of $L_1$ satisfy $\mathcal{S}(R) \geq s_{\mathcal{S}}$ but $\chi^2(R) < s_{\chi^2}$. The algorithm estimates the best $\chi^2$ that a specialization of $R$ could have (call it "opt$_{\chi^2}$"). This calculation is facilitated by a convexity property of the $\chi^2$ which allows the search of a greatly reduced number of possibilities (see Gascuel [Gas89] for a complete presentation). Then the algorithm keeps in $L_1$ all the remaining rules satisfying opt$_{\chi^2} \geq s_{\chi^2}$. For each of these rules, all the direct specializations are constructed and put in a list $L_2$. Every rule of $L_2$ having a generalization in $L_{end}$ is removed from $L_2$, and the process is repeated iteratively by putting in $L_{end}$ all the selected rules of $L_2$, removing the rules of $L_2$ having a simplicity $< s_{\mathcal{S}}$, etc... It ends at a step $j$ iff the current list $L_j = \emptyset$, and returns $L_{end}$. This algorithm ensures that any rule of the rule space that satisfies (1) and (2) (or one of its generalization that satisfies (1) and (2), if it exists) is retained in $L_{end}$. Thus it leads to a good cover of the rule space, but can also lead to long searches if $s_{\mathcal{S}}$ and $s_{\chi^2}$ parameters are not adjusted well. Nevertheless, in practice, research time took from a few seconds to 15 minutes, using a non-optimized C++ program running on a Sun Sparc 10. Furthermore, we remarked in practice that for almost all the domains that were used to test IDC, $L_{end}$ was composed of monomials having no more than 3 literals.

### 3.1.2 Calculation of the vectors

This is a two (at most) stage process that calculates the corresponding vector of each monomial. Let $t_i$ be a monomial chosen in the list $L_{end}$. The first stage is the following: let $e \in c_j$ be an example that satisfies $t_i$ (for a given class $c_j$). The values from $\{-1, 0, 1\}$ are associated to corresponding conditions as follows:

- $\vec{v}_i[j] = +1$ is associated to the condition $\Pr(e \Rightarrow t_i) > \frac{1}{2}$ with sufficiently high probability,
- $\vec{v}_i[j] = -1$ is associated to the condition $\Pr(e \Rightarrow t_i) < \frac{1}{2}$ with sufficiently high probability,
- $\vec{v}_i[j] = 0$ otherwise.

Fix $f_j = \frac{n_{fj}}{|c_j|}$. Under suitable hypotheses, and with probability $\geq 1-\alpha$, an upperbound of $\Pr(e \not\Rightarrow t_i)$ may be approximated using $f_j$ by the following quantity:

$$\mathcal{B}(f_j) = \frac{1}{1+2q}\left(f_j + q + \sqrt{q^2 + 2qf_j - 2qf_j^2}\right)$$

Where $q = \frac{1}{|c_j|}\ln(\frac{1}{\alpha})$ (Hoeffding [Hoe63], Gascuel and Caraux [GC92]). Obviously, if we can upperbound $\Pr(e \not\Rightarrow t_i)$ with probability $1-\alpha$, then we can lowerbound $\Pr(e \Rightarrow t_i)$ with probability $1-\alpha$.

Our rule to calculate $\vec{v}_i[j]$ is the following:

- **If** $\mathcal{B}(f_j) \leq \frac{1}{2}$ **then** $\vec{v}_i[j] = +1$.
- **Else If** $\mathcal{B}(1-f_j) \leq \frac{1}{2}$ **then** $\vec{v}_i[j] = -1$
- **Else** $\vec{v}_i[j] = 0$.

The second stage of the calculation takes place in rarely encountered cases[1], typical of rules having a low $\chi^2$. Namely, it takes place when all the values of the calculated $\vec{v}_i$ satisfy $\forall j,k \leq c, \vec{v}_i[j] = \vec{v}_i[k] = v = \pm 1$. $\mapsto$ Firstly take the case $v = +1$. Let $f_{j_1} = \max_{k \leq c} f_k$ and $f_{j_2} = \max_{k \neq j_1 \leq c} f_k$. Formula (1) above allows us to bound the corresponding probabilities ($\Pr(e \in$

[1]This particularly holds for the MONKS dataset #2, where the target function is an XOR-like function.

$c_{j_1} \not\Rightarrow t_i)$ for $f_{j_1}$ and $\Pr(e \in c_{j_2} \not\Rightarrow t_i)$ for $f_{j_2}$), and in that case, if $\mathcal{B}(j_2) \leq 1-\mathcal{B}(1-j_1)$, then $\vec{v}_i[j_1]$ is flipped and becomes $\vec{v}_i[j_1] = -1$.
$\mapsto$ Secondly take the case $v = -1$, let $f_{j_1} = \min_{k \leq c} f_k$ and $f_{j_2} = \min_{k \neq j_1 \leq c} f_k$. If $\mathcal{B}(j_1) \leq 1 - \mathcal{B}(1 - j_2)$, then $\vec{v}_i[j_1]$ is flipped and becomes $+1$. The previous calculations are then repeated for every monomial of $L_{end}$.

### 3.2 Combining rules

The aim of this algorithm is to extract from the population $L_{end}$ a subset of rules which, when combined in a decision committee, have small error on the learning sample. Its general principle is inspired from the agglomerative clustering algorithms of the statistical litterature. In our case, the initial population $P_0$ is the list of rules $L_{end}$ (where each monomial $t_i \in L_{end}$ has its corresponding vector $\vec{v}_i$ calculated). The individuals are the rules, and they can also be assimilated to decision committees (with one rule). Clusters of individuals are decision committees of more than one rule, and grouping of individuals consists simply in merging decision committees (to form new ones). When constructing a decision committee, the default vector is calculated by putting into it the observed distribution of ambiguous examples. At each step, two decision committees are merged to form a new one. They are chosen because their union maximizes a gain-criterion $G$. The algorithms stops when every merging produces a gain $G \leq 0$. Let $C$ and $C'$ be two decision committees, and $C \cup C'$ be the decision committee formed by merging $C$ and $C'$. Let $f_C$, $f_{C'}$ and $f_{C \cup C'}$ be the respective error frequency on the training sample of the previous decision committees. Then the gain criterion $G_{C,C'}$ produced by merging the two decision committees is:

$$G_{C,C'} = \min\{f_C; f_{C'}\} - f_{C \cup C'}$$

The two decision committees $C$, $C'$ merged at step $k$ satisfy $G_{C,C'} = \max_{C_1,C_2 \in P_k}\{G_{C_1,C_2}\} > 0$. When this maximal gain is $\leq 0$, the algorithm ends and returns the best decision committee (according to the learning sample) of the current population. This algorithm tries to take advantage of the shared nature of the knowledge of the rules in a decision committee, by not forgetting some rules (or decision committees), that in fact could advantageously complete the description quality of one when merged to it. Moreover, this approach empirically proved to be better than the greedy one that consists of merging to a current decision committee (initalized to $\emptyset$) the rule that increases the most the accuracy of this decision committee.

Clustering algorithms are frequently used in practice in domains such as statistics. The algorithm for combining rules is an adaptation of these techniques, that gave good results. Nevertheless, it must be noted that this algorithm is an heuristic, and we have the following property:

**Proposition 4** *We cannot devise (unless $P = NP$) even in the Boolean case an algorithm that, given $L_{end}$ and the set of examples $LS$ from which $L_{end}$ was constructed, can find (by merging rules from $L_{end}$) a decision committee whose error on $LS$ is optimal. And this even holds for very restricted cases of $L_{end}$ in which each rule consists only in a single monotone literal associated to $+1$.*

This is proved by reduction from the Vertex Cover problem (Garey and Johnson [GJ79]).

## 4 Experiments

Subsection §4.1 presents in a general way how we used datasets, and how we fixed the parameters of IDC. Subsection §4.2 presents the datasets, and the results we obtained.

### 4.1 Experimental process

*Binarization of continuous values:* Some problems contain continuous attributes, hardly manageable for binary-attributes based systems. Some algorithms use sophisticated binarization procedures to overcome this problem. Using a simple procedure (in order not to bias the results of IDC), continuous values were ternarized. Ternarization was chosen so as not to lose a lot of information. Take a continuous attribute. We cut the interval of its possible values three times, so that the four remaining subintervals contain approximately the same number of examples. Call $[i_0, i_1], [i_1, i_2], [i_2, i_3], [i_3, i_4]$ these subintervals. Then the first corresponding literal of an example is "1" iff the corresponding value is $\leq i_1$, and "0" otherwise. The second corresponding literal is "1" iff the corresponding value is $\leq i_2$, and "0" otherwise, etc... Finally, using the preceeding procedure, every continuous attribute gives indeed rise to three binary descriptors.

*Choice of parameters:* The parameters of the algorithm constructing monomials are fixed as follows:

- $s_S$ is a function of the quantity $min_c = \min_{i \leq c} |c_i|$. When $c > 2$, $s_S = min_c$. When $c = 2$, we put $s_S = \frac{min_c}{2}$ for almost all the problems. This puts a fairly high constraint over selected rules. However, for some particular problems (*e.g.* the MONKS dataset #2), where sometimes rules satisfying this constraint are not found, we put $s_S = \frac{min_c}{b}$, where $b$ is determined by cross-validation. We set an interval $S$ of values that $b$ will take (2 to 10 in our experiments), and for each of these, we perform 5 times the following. Split the training set into a set $LS_l$ (2/3) and $LS_t$ (1/3); perform IDC on $LS_l$ with adequate values of $b$, and test it on $LS_t$. After having averaged

Table 1: Characteristics of Data Sets.

| Domain | #Learning | #LS | #Test | #Attrs | $c$ | Comments |
|---|---|---|---|---|---|---|
| V0 | 435 | 1 | Ø | 16 | 2 | Congress-Votes Problem |
| V1 | 435 | 1 | Ø | 15 | 2 | V0+ attribute "Physician-fee Freeze" deleted |
| LE | 200 | 11 | 5000 | 7 | 10 | Digit recognition Problem |
| L24 | 200 | 11 | 5000 | 24 | 10 | LE + 17 irrelevant attributes |
| WB | 300 | 11 | 5000 | 21 | 3 | Waveform Recognition Problem Binarized |
| WT | 300 | 11 | 5000 | 21 | 3 | Waveform Recognition Problem Ternarized |
| GL | 214 | 1 | Ø | 9 | 6 | Identification of glass samples |
| G2 | 163 | 1 | Ø | 9 | 2 | GL+ class 1, 3 grouped and class 4 to 6 deleted |
| IR | 150 | 1 | Ø | 4 | 3 | Fisher's Iris dataset |
| M1 | 124 | 1 | 432 | 6 | 2 | MONKS dataset #1 |
| M2 | 169 | 1 | 432 | 6 | 2 | MONKS dataset #2 |
| M3 | 122 | 1 | 432 | 6 | 2 | MONKS dataset #3 |
| HE | 270 | 1 | Ø | 13 | 2 | Heart dataset |
| AU | 690 | 1 | Ø | 14 | 2 | Australian dataset |
| LA | 57 | 1 | Ø | 16 | 2 | Labor Negotiations |

References are: Breiman *et al.* [BFOS84]: LE, L24, WB, WT. Thrun *et al.* [TBBB91]: M1, M2, M3. Buntine and Niblett [BN92]: V0, V1, LE, GL, IR. Holte [Hol93]: V0, V1, GL, G2, IR. Gascuel and Gallinari [GG95]: WB, WT. Kohavi [Koh95]: V0, V1, GL, G2, IR, M1, M2, M3, HE, AU, LA.

error frequency on the 5 trials for each $b$, the minimal value of error frequency gives the value of $b$ to learn with the whole learning sample.

- $s_{\chi^2}$ corresponds to a probability of overtaking $\alpha = 0.05$ for a $\chi^2$ random variable with $c-1$ degrees of freedom.

*Cross-validations:* When there is only one set of examples without any test set, we proceed by averaging over 10 iterations the result of the following cross-validation: randomly split the whole sample into a learning sample (2/3 of the examples) and a test sample (1/3 of the examples); use the learning sample to construct a decision committee with IDC, and test it on the test set. This is the same experimental process for cross-validation as the one of Holte [Hol93].

### 4.2 Experimental results

IDC was tested on the datasets summed up in Table 1. There are 15 datasets. Datasets V0, V1, GL, IR, M1, M2, M3, HE, AU, LA and Aha's programs (for LE and L24) are from the collection available at the *UCI Repository of machine learning database*, and were used exactly as they are found in the January 1995 distribution. Datasets WB, WT are available on request. Below Table 1 are additional references for further informations or results concerning the datasets.

Table 2 shows the performances of IDC compared with other algorithms (values are of the form "Mean $\pm$ Standard Deviation"). Column "Best reported" is a non-exhaustive review of results about the best known results. These results concerns algorithms different than CART, C4 and IDC. They either come from mentioned publications or are available at the *UCI repository of machine learning database* (results distinguished by "o").

Table 2 shows that IDC can perform good results not only in simulated and noisy domains (*e.g.* LE and L24, for which Bayes accuracy is 74%), but also in simulated unnoisy domains (M2). If we exclude GL (where, however, IDC's result is not significantly different from CART's), this is also the case in real domains (HE, AU, VO, IR). These results are to be compared in the light of the corresponding sizes of the decision committees. Rules are neither numerous nor complex.

## 5 Discussion

IDC's heuristic is not the first we have studied to construct decision committees. Particularly, we have already studied stochastic techniques (genetic algorithms, simulated annealing in the same way as De Carvalho Gomes and Gascuel [dCGG94]), and various greedy techniques. But we abandoned them because of their results. Indeed, the results we obtained were never as good as IDC's. Their common point is that the problem of constructing decision committees was solved in one step, and not two like for IDC (construction of rules/combination of rules). In fact, this one-step constructions implied that rules did not have individual value, since any judgment was made only on the base of the whole decision committee.

On almost every dataset, IDC gave good results. Particularly, in a way, it proved experimentally to be noise-tolerant. For simulated problems such as LE

Table 2: Performances of IDC, compared with decision trees algorithms, and others.

| | *Accuracies* | | | | *Sizes* | |
|---|---|---|---|---|---|---|
| | IDC | D. Trees | Best reported | | IDC | D. Trees |
| V0 | $95.24 \pm 1.1$ | $95.5 \pm 1.0^{\ddagger}$ | 95.3 | NN [Hol93] | $1.0 \pm 0.0$ | $9.2 \pm 5.0^{\ddagger}$ |
| V1 | $89.11 \pm 1.8$ | $87.2 \pm 1.5^{\ddagger}$ | 86.8 | 1R [Hol93] | $6.4 \pm 1.9$ | $15.8 \pm 8.0^{\ddagger}$ |
| LE | $74.27 \pm 0.6$ | $66.2 \pm 3.1^{\ddagger}$ | 73.3 | IWN$^{\circ}$ | $12.2 \pm 1.9$ | $24.6 \pm 5.4^{\ddagger}$ |
| L24 | $73.60 \pm 2.0$ | $70^{\ddagger}$ | 71.5 | NT-growth$^{\circ}$ | $17.6 \pm 3.5$ | |
| WB | $76.25 \pm 2.3$ | $71.2 \pm 1.6^{\dagger}$ | 79.1 | NN [GG95] | $15 \pm 6.4$ | |
| WT | $80.36 \pm 1.6$ | $70.5 \pm 1.8^{\dagger}$ | 81.9 | NN [GG95] | $36 \pm 12.7$ | |
| GL | $55.89 \pm 6.4$ | $60.4 \pm 6.2^{\ddagger}$ | 62.0 | BruteDL [SE94] | $20 \pm 7.2$ | $14.2 \pm 8.0^{\ddagger}$ |
| G2 | $72.96 \pm 8.0$ | $70.6 \pm 2.0$ | 72.9 | 1R [Hol93] | $12.1 \pm 5.9$ | |
| IR | $96.00 \pm 2.7$ | $95.0 \pm 3.1^{\ddagger}$ | 98.0 | Lin. Disc. [Hol93] | $4.3 \pm 2.8$ | $5 \pm 1^{\ddagger}$ |
| M1 | 83.34 | 75.7 | 100 | AQ-17 [TBBB91] | 5 | |
| M2 | 70.61 | 65.0 | 100 | AQ-17 [TBBB91] | 18 | |
| M3 | 97.33 | 97.2 | 100 | AQ-17 [TBBB91] | 2 | |
| HE | $82.11 \pm 4.6$ | $76.7 \pm 1.8$ | 80.4 | IDTM [Koh95] | $15.2 \pm 7.8$ | |
| AU | $85.43 \pm 1.4$ | $85.4 \pm 1.1$ | 84.9 | IDTM [Koh95] | $1.8 \pm 1.9$ | |
| LA | $83.69 \pm 7.0$ | $85.7 \pm 3.5$ | 90.0 | AQ-15 [Hol93] | $6.8 \pm 2.9$ | |

$\mapsto$ Results for decision trees (D. Trees) are given for C4.5 (Kohavi [Koh95]), except:

$\dagger$ Induction of decision trees based on Kolmogorov-Smirnov distance (Celeux and Lechevallier [CL82], Gascuel and Gallinari [GG95]).

$\ddagger$ CART results (Breiman *et al.* [BFOS84] for L24), and Buntine and Niblett results [BN92] for V0, V1, LE, GL.

$\mapsto$ Sizes are the number of literals for IDC, and the number of edges for decision trees.

and L24, this tolerance is optimal or nearly optimal. WB, WT, M3 also reflect this property. Cestnik and Bratko [CB88] present redundancy as ways to cope with noise. It seems that IDC confirms this remark for several datasets. IDC is to be judged not only by its performances on testing, but also in the light of the corresponding sizes of the decision committees it creates. These decision committees are comprehensible, easy-to-interpret concepts. Firstly, this is due to the choice of the subclass of decision committees used by IDC. Secondly, note that the qualities of a rule are judged on the whole learning sample, independently of the previous retained rules. Learned rules can therefore be considered also in isolation (Segal and Etzioni [SE94]). But obviously, the smaller the decision committees, the easier the interpretation. Thirdly, note that IDC effectively tends to choose decision committees that have few rules.

WB and WT proved the importance of combining rules in the additive manner of decision committees. Indeed, Gascuel and Gallinari [GG95] remark that optimal frontiers in these problems seems to be almost linear separators. The decision committees constructed allow to approximate these surfaces, hence our results. On the contrary, linear separators are hardly approximable by decision trees, which obtain lesser results (cf Table 2), unless "Oblique decision trees" are used (Breiman *et al.* [BFOS84], Murthy *et al.* [MKS94])

In a more general context, our work empirically confirmed for several problems the importance of redundant and multiple knowledge (Cestnik and Bratko [CB88], Gams [Gam89], Kononenko and Kovačič [KK92]).

## References

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

[BN92] W. Buntine and T. Niblett. A further comparison of splitting rules for Decision-Tree induction. *Machine Learning*, pages 75–85, 1992.

[Bon70] H. Bongard. *Pattern Recognition.* Mac Millan, 1970.

[Bru90] J. Bruck. Harmonic analysis of polynomial threshold functions. *SIAM Journal on Discrete Mathematics*, 3:168–177, 1990.

[CB88] B. Cestnik and Y. Bratko. Learning redundant rules in noisy domains. In *Proc. of the 8 th ECAI*, pages 348–350, 1988.

[CL82] G. Celeux and Y. Lechevallier. Methodes de segmentation non paramétriques. *Revue de statistique appliquée*, pages 39–53, 1982.

[dCGG94] F. de Carvalho Gomes and O. Gascuel. Sdl, a stochastic algorithm for learning decision lists with limited complexity. *Annals of Mathematics and AI*, 10:281–302, 1994.

[Gam89] M. Gams. New measurements highlight the importance of Redundant Knowledge. In *Proc. of the 4 th EWSL*, pages 71–79, 1989.

[Gas86] O. Gascuel. Plage, a way to give and use knowledge in learning. In *Proc. of the 1 st EWSL*, 1986. also available in "Machine and Human Learning", Y. Kodratoff (Ed.), Michael Horwood series in Artificial Intelligence, pp. 105-120, 1989.

[Gas89] O. Gascuel. Inductive learning, numerical criteria and combinatorial optimization, some results. In Nova Science Publishers, editor, *Data Analysis, Learning Symbolic and Numeric Knowledge*, pages 417–424. E. Diday, 1989.

[GC92] O. Gascuel and G. Caraux. Distribution-free performances bounds with the resubstitution error estimate. *Pattern Recognition*, 13:757–764, 1992.

[GG95] O. Gascuel and P. Gallinari. Méthodes symboliques-numériques de discrimination. In *Actes des 5èmes Journées Nationales du PRC-IA*, pages 29–76. Teknea, 1995. Published under the collective name *SYMENU*.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability, a guide to the theory of NP-Completeness.* Bell Telephone Laboratories, 1979.

[HJLT95] T. Hancock, T. Jiang, M. Li, and J. Tromp. Lower bounds on learning Decision Lists and Trees. In *STACS'95*, 1995.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. Am. Stat. Assoc.*, pages 13–30, 1963.

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

[HR76] L. Hyafil and R. Rivest. Constructing optimal binary decision tree is NP-complete. *Information Processing Letters*, pages 15–17, 1976.

[Kar84] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, pages 373–395, 1984.

[KK92] I. Kononenko and M. Kovačič. Learning as Optimization: Stochastic Generation of multiple knowledge. In *Proc. of the 9 th ICML*, 1992.

[KLPV87a] M. J. Kearns, M. Li, L. Pitt, and L. Valiant. On the learnability of boolean formulae. *Proc. of the 19 th STOC*, pages 285–295, 1987.

[KLPV87b] M. J. Kearns, M. Li, L. Pitt, and L. Valiant. Recent results on boolean concept learning. *Proc. of the 4 th ICML*, pages 337–352, 1987.

[Koh95] R. Kohavi. The power of Decision Tables, 1995. Draft accepted to ECML95.

[MKS94] S. K. Murthy, S. Kasif, and S. Salzberg. A system for Induction of Oblique Decision Trees. *JAIR*, pages 1–32, 1994.

[MP94] P. M. Murphy and M. J. Pazzani. Exploring the Decision Forest: an empirical investigation of Occam's Razor in Decision Tree induction. *JAIR*, pages 257–275, 1994.

[Nil65] N. J. Nilsson. *Learning machines.* McGraw-Hill, 1965.

[PV88] L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *J. ACM*, pages 965–984, 1988.

[QS83] J. Quinqueton and J. Sallantin. Expansion and compression of binary data to build features by learning. In *IJCPR*, pages 641–644, 1983.

[Riv87] R.L. Rivest. Learning decision lists. *Machine Learning*, pages 229–246, 1987.

[SE94] R. Segal and O. Etzioni. Learning Decision Lists using homogeneous rules. In *Proc. of AAAI-94*, pages 619–625, 1994.

[Sho76] E. Shortliffe. *Computer based medical consultations: MYCIN.* American Elsevier, 1976.

[TBBB91] S. B. Thrun, J. Bala, E. Bloedorn, and I. Bratko. The MONK's problems: a performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, 1991.

[Tur91] G. Turán. A survey of some aspects of Computational Learning Theory. In *Proc. of the 32 th FOCS*, pages 89–103, 1991.

[Val84] L. G. Valiant. A theory of the learnable. *J. ACM*, pages 1134–1142, 1984.

[Vap82] V. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer Verlag, 1982.