Knowledge-Based Systems 15 (2002) 37–43

Knowledge-Based SYSTEMS

www.elsevier.com/locate/knosys

A quality index for decision tree pruning

D. Fournier*, B. Crémilleux

GREYC, CNRS — UMR 6072, Université de Caen, F-14032 Caen cedex, France

Abstract

Decision tree is a divide and conquer classification method used in machine learning. Most pruning methods for decision trees minimize a classification error rate. In uncertain domains, some sub-trees that do not decrease the error rate can be relevant in pointing out some populations of specific interest or to give a representation of a large data file. A new pruning method (called *DI* pruning) is presented here. It takes into account the complexity of sub-trees and is able to keep sub-trees with leaves yielding to determine relevant decision rules, although they do not increase the classification efficiency. *DI* pruning allows to assess the quality of the data used for the knowledge discovery task. In practice, this method is implemented in the UnDeT software. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Decision tree; Quality; Pruning; Uncertain data

1. Introduction

Data mining and knowledge discovery in databases (KDD) are fields of increasing interest combining databases, artificial intelligence, machine learning and statistics. Broadly speaking, the purpose of KDD is to extract non-trivial 'nuggets' of information in an easily understandable form from large amount of data. Such discovered knowledge may be for instance regularities or exceptions.

In this context, with the growth of databases, methods which explore data — like for example decision trees — are required. Such methods can give a summary of the data (which is easier to analyze than the raw data) or can be used to build a tool (like for example a classifier) to help a user for many different decision-makings.

Briefly, a decision tree is built from a set of training data having attribute values and a class name. The result of the process is represented as a tree, the nodes of which specify attributes and the branches specify attribute values. Leaves of the tree correspond to sets of examples with the same class or to elements in which no more attributes are available. Construction of decision trees is described, among others, by Breiman et al. [1] who present an important and well-known monograph on classification trees. A number of standard techniques have been developed, for example like the basic algorithms ID3 [2] and CART [1].

Nevertheless, in many areas, such as medicine, data are uncertain: this means that there are always some examples which escape the rules. Translated in the context of decision trees means that these examples seem similar but in fact differ from their classes. In these situations, it is well known (see Refs. [1,3]) that decision tree algorithms tend to divide nodes having few examples and that the resulting trees tend to be very large and overspecified. Some branches, especially towards the bottom, are present due to sample variability and are statistically meaningless (one can also say that they are due to noise in the sample). Pruning methods (see Refs. [1,2,4]) try to cut such branches in order to avoid this drawback.

In uncertain domains, the understanding of the mechanism of these methods is a key point for their use in practice: in order to achieve a fruitful process of extraction of information, these methods require declarativity during treatments. In Section 3, it is seen that this point is included in the pruning strategy.

This paper is organized as follows. Section 2 outlines existing decision trees pruning methods and sets out the question of pruning in uncertain domains. The principal pruning methods are based on a classification error rate and that it may be a drawback in uncertain domains is seen. So, in Section 3, a quality index (called *DI* for depth-impurity quality index) which is a trade-off between the depth and the impurity of nodes of a tree is proposed. From this index, a new pruning method for decision trees (denoted *DI* pruning) that is appropriate in uncertain domains is inferred: unlike usual methods, this method is not bound to the possible use of a tree for classification. It is capable of giving an efficient description of a data file

* Corresponding author. Tel.: +33-2-3156-7379; fax: +33-2-3156-7330.
E-mail addresses: fournier@info.unicaen.fr (D. Fournier), cremilleux@info.unicaen.fr (B. Crémilleux).

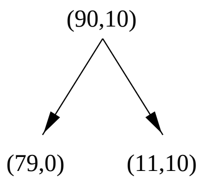


Fig. 1. A tree which could be interesting although it does not decrease the number of errors.

oriented by a priori classification of its elements and to highlight interesting sub-populations, even though the classification error rate does not decrease (see examples in Section 4). It is considered that it is a major point for the use of decision trees in uncertain domains. In Section 4, a short overview of the UnDeT software which implements this method is given. Examples in a real world domain in which *DI* pruning is used as a tool to optimize a final decision tree is presented.

2. Motivations

The principal methods for pruning decision trees are examined in Refs. [4–6]. Most of these pruning methods are based on minimizing a classification error rate where each element of the same node is classified in the most frequent class in this node. The latter is estimated with a test file or using statistical methods such as cross-validation or bootstrap.

These pruning methods are inferred from situations where the built tree will be used as a classifier and they systematically discard a sub-tree which does not improve the used classification error rate. The sub-tree depicted in Fig. 1 is considered. *D* is the class and it is here bivalued. In each node the first (resp. second) value indicates the number of examples having the first (resp. second) value of *D*. This sub-tree does not lessen the error rate, which is 10% both in its root or in its leaves; nevertheless the sub-tree is of interest since it points out a specific population with a constant value of *D* while in the remaining population it is impossible to predict a value for *D*. The authors think that, in uncertain domains, cutting such a sub-tree would introduce more uncertainty than keeping the leaves.

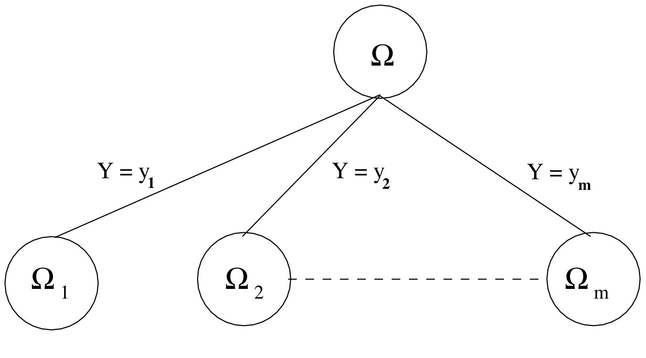


Fig. 2. Splitting of a node Ω using an attribute *Y*.

Another way to prune decision trees, based on a quality improvement, is treated in Refs. [7,8]. These methods use a quality measurement that indicates an information gain brought by the tree. Even though both methods have the ability to spare sub-trees like in Fig. 1, they have some limitations: in Ref. [7], they do not take into account the complexity of the trees and Wehenkel [8] only computes the total number of nodes without integrating the layout of the trees. In the context of inductive rule learners, the idea to manage a trade-off between the gain of a rule (i.e. the improvement of the accuracy relative to the initial situation) and its structure (i.e. the number of examples covered) is, for instance, discussed in Ref. [9].

Here a twofold aim is pursued: on one hand, to develop a pruning method which does not systematically discard a sub-tree whose classification error rate is equal to the rate of the root, and on the other hand, to handle the complexity of the trees. Section 3 describes a tree quality index and a pruning method based on this index which integrates these constraints. This method is able to highlight interesting — in our opinion — sub-populations (like the ones shown in Fig. 1). This method is not based on a classification error rate and the authors claim that it is a major point in uncertain domains.

3. Depth-impurity quality index and pruning

3.1. Framework for a quality index

The question of a quality index is formulated. The authors claim that the quality index of a tree *T* has its maximum value if and only if the two following conditions are satisfied:

(i) All the leaves of *T* are pure.
(ii) Depth of *T* is 1.

These conditions are part of the usual framework to properly define suitable attribute selection criteria to build decision trees is noticed [1,7]. This framework states that on a theoretical level, criteria derived from an impurity measure [10] perform comparably [1,7,11].[1] Such criteria are called concave-maximum criteria (CM criteria) because an impurity measure, among other characteristics, is defined by a concave function. The most commonly used criteria which are the Shannon entropy (in the family of C4.5 software [12]) and the Gini criterion (in CART algorithms [1]) are CM criteria.

The idea is then to use known properties (based on an impurity measure) of CM criteria to define a quality index. In order to better understand the genesis of the quality index that is present below, a closer look at the question of attribute selection criteria in decision trees and its usual

[1] The term impurity is used here because it is the most usual in the machine learning community [1,10].

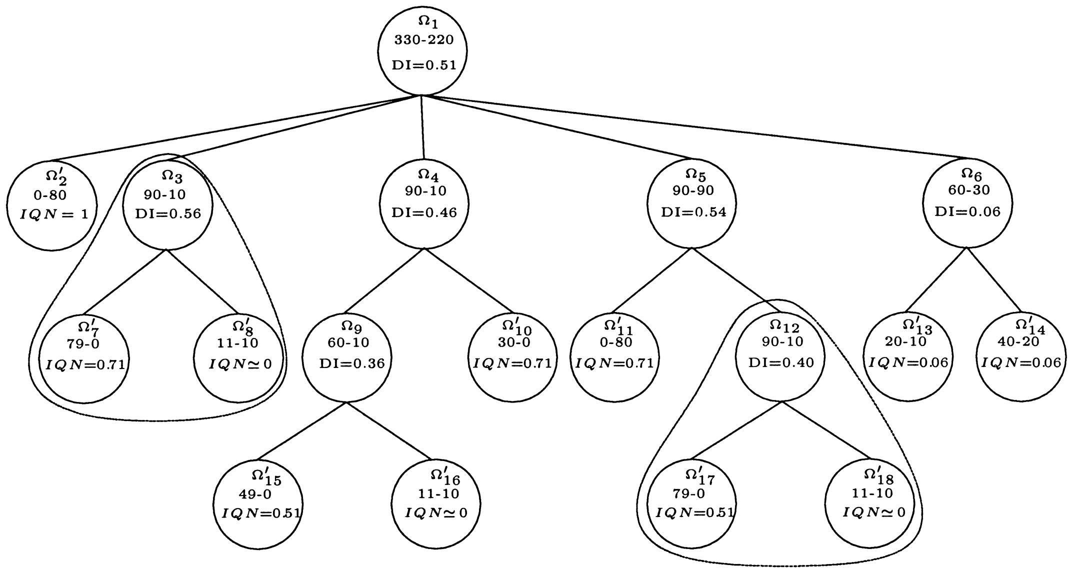


Fig. 3. Examples of *DI* and *IQN* values on a pedagogic tree.

notations is to be taken. Let D be called the class of the examples of a data set, with values $d_1,\dots,d_k$, Ω a node and Y any attribute defined on Ω, with values $y_1,\dots,y_m$. Let ψ be a CM criterion (see Fig. 2).

$\psi(Y) = \sum_i^m (\alpha(\Omega_i)/\alpha(\Omega))\varphi(\Omega_i)$ where $\Omega_1,\dots,\Omega_m$ are the sub-nodes yielded by Y, $\alpha(\Omega_i)$ is the number of examples in Ω_i, and φ is an impurity measure of D (see Refs. [1,7,10]). $\psi(Y)$ can be viewed as a combined measure of impurity of the sub-nodes induced by Y. The value of the criterion in a node reflects how appropriately the chosen attribute divides the data: it is a way to quantify the quality of a tree of depth 1. For example, theoretical results on CM criteria claim that the minimum value of $\psi(Y)$ is reached if and only if the sub-nodes induced by Y are pure with respect to D, $\psi(Y)$ has its maximum value if and only if the frequency distributions of D in Ω and in the sub-nodes induced by Y are equal.

In fact, these criteria, used to quantify the quality of a splitting (that means of a tree of depth 1) can be straightforwardly extended in order to evaluate the quality of a tree of any depth: the sub-nodes of depth 1 induced by Y are replaced by the leaves of any depth of the tree. Furthermore, that the aim is to offer a suitable pruning method in uncertain domains should not be forgotten. In this context, the authors claim that a deep tree is less relevant than a small one: the deeper a tree, the less understandable and reliable. For example, in Fig. 3, the tree which has the root denoted Ω_3 is preferred than the one with root Ω_{12} (even if the frequency distributions of D are the same on all leaves). On the other hand, both trees with root Ω_3 and Ω_4 have the same number of miss-classified examples, but the tree with root Ω_3 is preferred because it is simpler.

So, a quality index which takes into account both impurities and depths of leaves should be defined properly. Section 3.2 presents this index.

3.2. The quality index DI

First of all, the quality of a node Ω (called *IQN* for impurity quality node) is defined as a combination between its purity and its depth. $IQN(\Omega)$ is relative to the tree T where Ω is located (so T is introduced in the notation). $IQN_T(\Omega)$ is given by Eq. (1):

$$IQN_T(\Omega) = (1 - \varphi(\Omega))f(depth_T(\Omega)) \quad (1)$$

where T is the decision tree which contains the node Ω and φ is an impurity measure normalized between [0,1]. It is noted that as φ is an impurity measure, $(1 - \varphi)$ defines a purity measure. By introducing a damping function (denoted as f), *IQN* is able to take into account the depth of a node within T (denoted depth_T): the deeper a node, the lower its quality.

Eq. (2) defines the quality index *DI* (for depth-impurity) of a subtree T_s in T (either $DI(T_s$ is noted) or $DI(\Omega_s)$, the quality index of the tree T_s of root Ω_s). *DI* is directly stemmed from *IQN* as the weighted average of the quality of the tree's leaves

$$DI(T_s) = DI(\Omega_s) = \sum_{i=1}^{m} \frac{\alpha_i}{\alpha_s} IQN_T(\Omega'_i) \quad (2)$$

where $\Omega'_1,\dots,\Omega'_m$ are the leaves of T_s, α_i (resp. α_s)[2] gives the number of examples in Ω'_i (resp. Ω_s).

So, Eq. (2) can also be written as

$$DI(T_s) = DI(\Omega_s) = \sum_{i=1}^{m} \frac{\alpha_i}{\alpha_s} (1 - \varphi(\Omega'_i))f(\text{depth}_T(\Omega'_i)) \quad (3)$$

Due to the definition of *IQN*, the depth of a leaf is

[2] To simplify notations, $\alpha(\Omega_i)$ is replaced by α_i.

computed from the root of the whole tree (and not from the root of T_s). For instance, in Fig. 3, to compute the quality of the sub-tree of root Ω_3, the depth of the leaf Ω'_7 is with respect to Ω_1. Referring again to the example given at the end of Section 3.1 (comparison between trees of root Ω_3 versus the one of root Ω_{12}), the quality index of the tree of root Ω_{12} is lower because it is deeper is noticed. This choice to compute the depth allows comparing whichever nodes. As *DI* is based on *IQN* (and thus on the damping function used in *IQN*), *DI* also takes into account the depth of the leaves.

The authors are led to address now the question of defining the damping function *f*. The argument of *f* is a depth of a node (denoted *d*). A minimal set of constraints is:

(1) *f* is decreasing.
(2) $f(1) = 1$.

Constraint (1) corresponds to the damping process and (2) is necessary to satisfy condition (ii) of our framework. If an impurity measure normalized between [0,1] is chosen, as $\sum_i^m (\alpha_i/\alpha_s) = 1$, it is will be seen immediately with both constraints that the value of *DI* is between [0,1]. The higher the value of *DI*, the better is the quality of T_s.

Furthermore, the three following constraints are added:

(3) $f(d) \simeq 0$ when *d* tends towards the total number of attributes.
(4) $0 \le f(d) \le 1$.
(5) $f(d+1) = \beta f(d)$ where $\beta \in \mathbb{R}^+$ (in practice, $\beta \in [0,1[$ to respect constraint (1)).

Constraint (3) means that a leaf which has a depth similar to the total number of attributes is likely to be unreliable (particularly in uncertain domains), so it seems sensible that the value of its quality is close to the minimum value of *DI*. Constraint (4) allows the values of the damping function and of the purity (or impurity) of a leaf to have same rough estimates. Over the achievement of a linear damping, constraint (5) is suggested by algorithmic considerations: it will be seen that it allows to have a linear computation of *DI* as indicated by Remark 1. So, as the number of elementary steps in the whole process is limited, the computational cost of *DI* pruning (see Section 3.3) is particularly low and it is tractable even with large databases.

Translated into a mathematical form, this set of constraints leads to choose an exponential function for *f*.

Proof. From $f(d+1) = \beta f(d)$, it is deduced that $f(d) = \beta^{(d-1)} f(1)$. Thus from (2): $f(d) = \beta^{(d-1)}$ is obtained. This equality implies that *f* is an exponential function.

Thus the following function *f* is chosen:

$$f(d) = \mathrm{e}^{-(d-1/N)} \tag{4}$$

where *N* is the total number of attributes (let us note that any exponential function can be chosen).

□

Coming back to the tree given in Fig. 3, it is noticed that the tree which has the root denoted by Ω_3 has a greater *DI* value than the one with root Ω_4: impurities of leaves of these trees are equal, but the latter is more complex than the former. This has been taken into account by *DI*.

Remark 1. With regard to the algorithmic point of view, computation of *DI* is not expensive. *DI*(T_s) can be easily written according to the *DI* values of sons of T_s: in other words, the computation of *DI*(T_s) is the weighted average of the *DI* values of the sons of T_s.

Proof. As used previously, $\Omega'_1, \ldots, \Omega'_m$ are the leaves of a tree T_s of root Ω_s. Let Ω_k be called as a son of Ω_s and Ω'_j as the leaves stemmed from Ω_k. Then:

$$DI(\Omega_k) = \sum_j \frac{\alpha_j}{\alpha_k} IQN_T(\Omega'_j). \tag{5}$$

To consider all the sons of Ω_s:

$$\sum_k \frac{\alpha_k}{\alpha_s} DI(\Omega_k) = \sum_k \frac{\alpha_k}{\alpha_s}\left(\sum_j \frac{\alpha_j}{\alpha_k} IQN_T(\Omega'_j)\right) \tag{6}$$

As the sets of leaves of each Ω_k make a partition of the leaves of Ω_s, then

$$\sum_k \frac{\alpha_k}{\alpha_s}\left(\sum_j \frac{\alpha_j}{\alpha_k}\right) = \sum_{i=1}^m \frac{\alpha_i}{\alpha_s} \tag{7}$$

With Eq. (7), Eq. (6) becomes

$$\sum_k \frac{\alpha_k}{\alpha_s} DI(\Omega_k) = \sum_{i=1}^m \frac{\alpha_i}{\alpha_s} IQN_T(\Omega'_i) \tag{8}$$

It follows easily that the definition of *DI* (see Eq. (2)) can be rewritten

$$DI(\Omega_s) = \sum_k \frac{\alpha_k}{\alpha_s} DI(\Omega_k) \tag{9}$$

□

By ensuring that *DI* is computed for each node in one step, this point will allow a low computational cost for the *DI* pruning method. This subject is discussed now.

3.3. DI pruning

DI leads to a straightforward way to prune sub-trees T_s: the main idea is to compare *DI*(T_s) with the relative quality

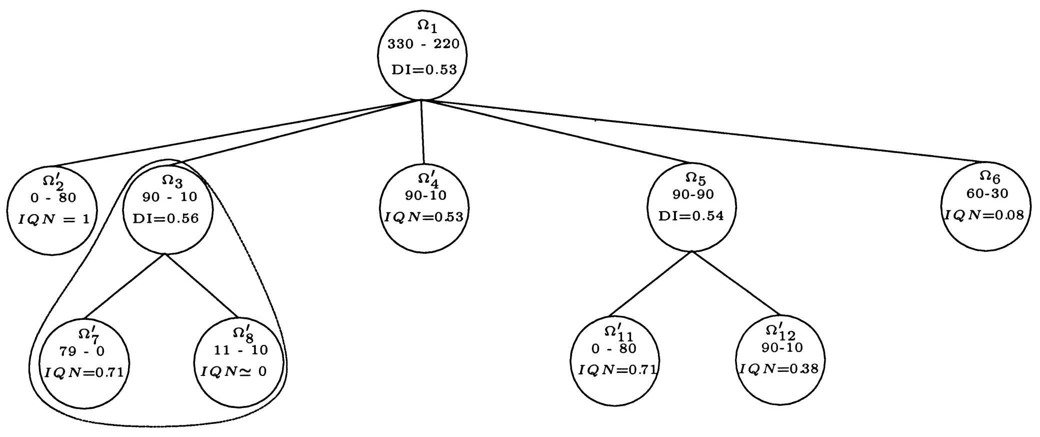
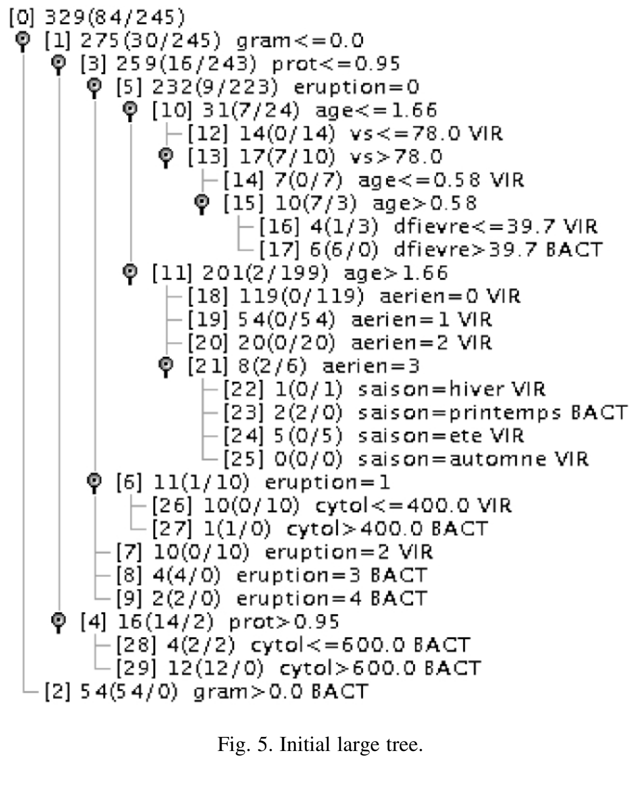


Fig. 4. Examples of *DI* values on the pedagogic pruned tree.

of its root. When $DI(T_s)$ is lower than the relative quality of its root, the 'cost' of T_s (due to its complexity) is more 'expensive' than the benefit of the explanation it has generated. So, T_s reduces the quality of the whole tree T. Then cutting and replacing T_s by its root, which becomes a leaf, improve the quality of T. Within our framework, this means that if a subtree T_s of root Ω_s is cut satisfying Eq. (10), the quality of the whole tree T can only increase

$$DI(\Omega_s) \leq \frac{\alpha_s}{\alpha} IQN_T(\Omega_s) \tag{10}$$

where α is the number of examples of T. So, a straightforward pruning method that maximizes the quality of T, consists in pruning recursively from the bottom of T, all sub-trees T_s of T satisfying Eq. (10). This method is called *DI* pruning because it goes with *DI*.



```
[0] 329(84/245)
 [1] 275(30/245) gram<=0.0
   [3] 259(16/243) prot<=0.95
     [5] 232(9/223) eruption=0
       [10] 31(7/24) age<=1.66
         ├[12] 14(0/14) vs<=78.0 VIR
         [13] 17(7/10) vs>78.0
           ├[14] 7(0/7) age<=0.58 VIR
           [15] 10(7/3) age>0.58
             ├[16] 4(1/3) dfievre<=39.7 VIR
             └[17] 6(6/0) dfievre>39.7 BACT
       [11] 201(2/199) age>1.66
         ├[18] 119(0/119) aerien=0 VIR
         ├[19] 54(0/54) aerien=1 VIR
         ├[20] 20(0/20) aerien=2 VIR
         [21] 8(2/6) aerien=3
           ├[22] 1(0/1) saison=hiver VIR
           ├[23] 2(2/0) saison=printemps BACT
           ├[24] 5(0/5) saison=ete VIR
           └[25] 0(0/0) saison=automne VIR
     [6] 11(1/10) eruption=1
       ├[26] 10(0/10) cytol<=400.0 VIR
       └[27] 1(1/0) cytol>400.0 BACT
     ├[7] 10(0/10) eruption=2 VIR
     ├[8] 4(4/0) eruption=3 BACT
     └[9] 2(2/0) eruption=4 BACT
   [4] 16(14/2) prot>0.95
     ├[28] 4(2/2) cytol<=600.0 BACT
     └[29] 12(12/0) cytol>600.0 BACT
 └[2] 54(54/0) gram>0.0 BACT
```

Fig. 5. Initial large tree.

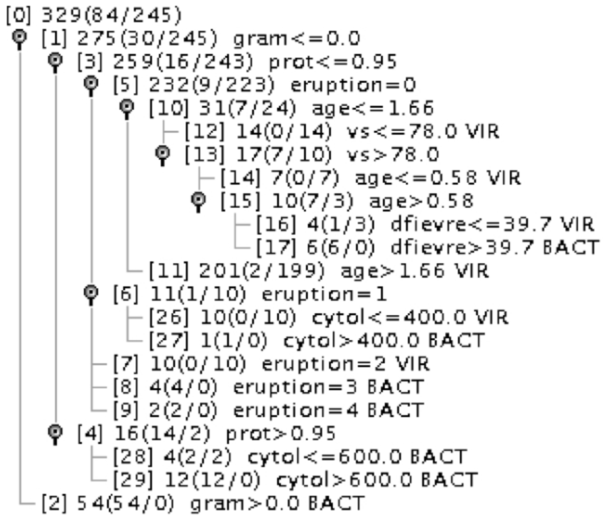
It is noted that the cost-complexity method of Breiman [1] uses a similar process (that means a trade-off between a 'cost' and a 'benefit' to cut sub-trees), but the cost is based on a classification error rate.

From experiments, it is noticed that the degree of pruning is quite bound to the uncertainty embedded in data. In practice, this means the damping process has to be adjusted according to the data in order to obtain, in all situations, a relevant number of pruned trees. For that, a parameter is introduced (denoted k) to control the damping process: the higher k, the more extensive the pruning stage (i.e. more sub-trees are cut). Eq. (11) gives the damping function updated by this parameter (as usual, N is the total number of attributes)

$$f_k(d) = \mathrm{e}^{-(k(d-1)/N)} \text{ with } k \in \mathbb{R}^+ \tag{11}$$

With $k = 1$, again Eq. (4) is obtained. By varying k, *DI* pruning produces a family of nested pruned trees spreading from the initial large tree to the tree restricted to its root. In practice, it should be not easy to select automatically the 'best' pruned tree (but it is not the main aim of this stage of this work). Nevertheless, curves of *DI* as a function of k and as a function of the number of pruned nodes give a pragmatic method to stop the pruning process. Furthermore, if we are eager to obtain automatically a single 'best' pruned tree, a procedure requiring a test file can be used [1,6].

Fig. 4 shows the pruned tree obtained (with $k = 1$) from the whole tree indicated in Fig. 3. Two sub-trees (of root Ω_4 and Ω_{12}) have been cut. Although the sub-trees of root Ω_3 and Ω_{12} are identical, only the latter is cut because it is deeper than the former. Moreover, even though the frequency distributions of D in Ω_3 and in Ω_4 are equal, the sub-tree of root Ω_4 is removed (and *not* the sub-tree of root Ω_3), because Ω_4 is more complex than Ω_3. The authors remark that $DI(\Omega_1)$ has increased, as expected.



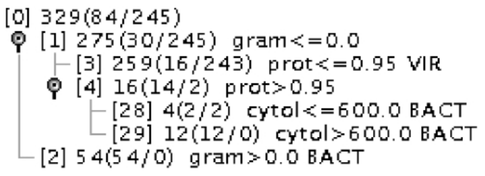
```
[0] 329(84/245)
  [1] 275(30/245) gram<=0.0
    [3] 259(16/243) prot<=0.95
      [5] 232(9/223) eruption=0
        [10] 31(7/24) age<=1.66
          ├[12] 14(0/14) vs<=78.0 VIR
          [13] 17(7/10) vs>78.0
            ├[14] 7(0/7) age<=0.58 VIR
            [15] 10(7/3) age>0.58
              ├[16] 4(1/3) dfievre<=39.7 VIR
              └[17] 6(6/0) dfievre>39.7 BACT
        └[11] 201(2/199) age>1.66 VIR
      [6] 11(1/10) eruption=1
        ├[26] 10(0/10) cytol<=400.0 VIR
        └[27] 1(1/0) cytol>400.0 BACT
      ├[7] 10(0/10) eruption=2 VIR
      ├[8] 4(4/0) eruption=3 BACT
      └[9] 2(2/0) eruption=4 BACT
    [4] 16(14/2) prot>0.95
      ├[28] 4(2/2) cytol<=600.0 BACT
      └[29] 12(12/0) cytol>600.0 BACT
  └[2] 54(54/0) gram>0.0 BACT
```

Fig. 6. Pruned tree ($k = 2$).

4. Experiments

An induction software called UnDeT (for uncertain decision trees) which produces decision trees has been designed. Three paradigms of attribute selection criteria are available in UnDeT: gain, gain ratio and ORT criterion (the choice of one of these depends on the kind of data and the aim wished by the user: see Ref. [13]). UnDeT computes *IQN* and *DI* indexes for each node and pruned trees with *DI* pruning. The authors' tool also offers all the functionalities which one can expect from an effective data mining tool: management of the training and test sets, automatic cross-validation, confusions matrix, back-ups and re-use of the classification built model (more information about all the functionalities of UnDeT and its use to study difficult problems issued from uncertain domains are available in Ref. [14]). UnDeT is included in a more general data mining tool-box in which another major part is dedicated to the treatment of missing values [15].

In this section the results obtained by running UnDeT on a real world database is discussed. UnDeT with the gain criterion (Shannon entropy) is performed [12] because it is the most commonly used one. The data set is a medical database coming from the University Hospital at Grenoble (France) and runs on child's meningitis (trees in Fig. 5–7 are snapshots produced by UnDeT).



```
[0] 329(84/245)
  [1] 275(30/245) gram<=0.0
    ├[3] 259(16/243) prot<=0.95 VIR
    [4] 16(14/2) prot>0.95
      ├[28] 4(2/2) cytol<=600.0 BACT
      └[29] 12(12/0) cytol>600.0 BACT
  └[2] 54(54/0) gram>0.0 BACT
```

Fig. 7. Pruned tree ($k = 4$).

4.1. Child's meningitis

Faced with a child with a case of acute meningitis, the clinician must quickly decide which medical course should be taken. Briefly, the majority of these cases are viral infections for which a simple medical supervision is sufficient, whereas about one quarter cases are caused by bacteria and need treatment with suitable antibiotics.

In typical cases, the diagnosis could be considered as obvious and a few simple rules enable a quasi certain decision. However, nearly one-third of these cases are presented with non-typical clinical and biological data: the difficulty of the diagnosis lies in the fact that the observed attributes, considered separately, have little diagnostic signification. The aim of this section is to study the relevance of *DI* pruning in such domains.

The used data set is composed of 329 instances, described by 22 (quantitative or qualitative) attributes. The class is bivalued (viral versus bacterial).

4.2. Results

The initial large tree (see Fig. 5) has 29 nodes with pure leaves. Its quality (0.739) is high, especially for a medical domain. This result is due to the relevance of the used attributes.

In the tree depicted in Fig. 5 and further, let T_i be called the sub-tree of root labelled `[i]`. T_4 has an impurity in its root equal to 0.54 (two miss-classified instances among 16) and T_{10} has an impurity in its root equal to 0.77 (seven miss-classified instances among 31). Although T_{10} increases significantly the classification result (it finally leads to a single miss-classified instance), $DI(T_{10})$ remains lower than $DI(T_4)$ ($DI(T_{10}) = 0.584$ and $DI(T_4) = 0.625$). This behavior was expected: T_{10} is complex and some leaves are reached only at the seventh level of the tree. So, the explanation given by T_{10} is not very reliable.

Fig. 6 represents the first pruned tree obtained with $k = 2$, its quality is slightly improved (0.743). The sub-tree of root labelled `[11]` becomes a leaf. This pruning introduces two miss-classified instances. This number is not high regarding the 201 instances of the node. Furthermore, in order to properly classify these two instances, the initial large tree had to build a complex sub-tree with deep leaves. Such leaves appear not to be very reliable in uncertain domains.

Finally, Fig. 7 indicates the next pruned tree obtained with $k = 4$. With this pruning stage, a new step of simplification is reached. The sub-tree of root labelled [3] becomes a leaf (in this case, the sub-tree T_{10} is cut: its complexity to classify a single instance is not reliable).

It is important to notice that the sub-tree of root labelled [4] is *not* destroyed: even if it does not decrease the number of miss-classified examples (which is two on both root and leaves), this sub-tree highlights a reliable sub-population when the attribute 'cytol' is higher than 600 (this result is checked by the medical expert). It is typically the situation that the authors presented in their motivations (see Section 2).

5. Conclusions

The authors have presented a quality index *DI* for decision trees for uncertain domains which realizes a trade-off between the impurities and the depth of the leaves of a tree. Stemming from *DI*, the authors present a pruning method which is able to keep sub-trees which do not decrease an error rate but can point out some populations of specific interest; yet usual methods are based on a classification error rate. The authors claim that it is a major point in uncertain domains.

Further work is to optimize the choice of the damping parameter so that it is not linked to the degree of uncertainty embedded in the data and it gives the 'best' pruned tree. A way for this is to use a test file, then move to another stage of the authors' work, which will be to select a pruned tree which reflects — *in general* — the sound knowledge of the studied data. A further stage is to compare *DI* pruning with others known methods [4,5,16].

Finally, it is noted that when the quality index has a low value on a subtree T_s, it suggests that T_s contains poor data for the KDD tasks. Another direction is to use this index to manage a feedback to the experts of the domain in order to improve such data.

Acknowledgements

The authors wish to thank Dr P. François (University Hospital of Grenoble, France) for providing data on child's meningitis and many valuable comments.

References

[1] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Clasification and Regression Trees, Statistics Probability Series, Wadsworth, Belmont, CA, 1984.

[2] J.R. Quilan, Induction of decision trees, Machine Learning 1 (1986) 81–106.

[3] J. Catlett, Overpruning large decision trees, Proceedings of the Twelfth International Joint Conference on Artificial Intelligence IJCAI 91 (1991) 764–769.

[4] J. Mingers, An empirical comparison of pruning methods for decision-tree induction, Machine Learning 4 (1989) 227–243.

[5] F. Esposito, D. Malerba, G. Semeraro, Decision tree pruning as search in the state space, in: P.B. Brazdil (Ed.), Proceedings of European Conference on Machine Learning ECML 93, Lecture Notes in Artificial Intelligence, vol. 667, Springer, Berlin, 1993, pp. 165–184.

[6] J.R. Quinlan, Simplifying decision trees, International Journal of Man–Machine Studies 27 (1987) 221–234.

[7] B. Crémilleux, C. Robert, A theoretical framework for decision trees in uncertain domains: application to medical data sets, in: E. Keravnou, C. Garbay, R. Baud, J. Wyatt (Eds.), Proceedings of the Sixth Conference on Artificial Intelligence in Medicine Europe (AIME 97), Lecture Notes in Artificial Intelligence, vol. 1211, Springer, Berlin, 1997, pp. 145–156.

[8] L. Wehenkel, Decision tree pruning using an additive information quality measure, in: B. Bouchnon-Meunier, L. Valverde, R.R. Yager (Eds.), Uncertainty in Intelligent Systems, Elsevier/North Holland, Amsterdam, 1993, pp. 397–411.

[9] L. Todorovski, P. Flach, N. Lavrač, Predictive performance of weighted relative accuracy, Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases, vol. 1910, Springer, Berlin, 2000, pp. 255–264.

[10] U.M. Fayyad, K.B. Irani, The attribute selection problem in decision tree generation, Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI Press/MIT Press, Cambridge, MA, 1992, pp. 104–110.

[11] W. Bountine, T. Niblett, A further comparison of splitting rules for decision-tree induction, Machine Learning 8 (1992) 75–85.

[12] J.R. Quinlan, C4.5 Programs for Machine Learning, Morgan Kaufmann, San Mateo, California, 1993.

[13] B. Crémilleux, C. Robert, Use of selection criteria in decision trees in uncertain domains, in: B. Bouchon Meunier, R.R. Yager, Z.A. Zadeh (Eds.), Uncertainty in Intelligent and Information Systems, World Scientific, Singapore, 2000.

[14] D. Fournier, Undet: a tool for building uncertain trees, Computing and Information Systems Journal 7 (2000) 73–78.

[15] A. Ragel, B. Crémilleux, Mvc — a preprocessing method to deal with missing values, Knowledge-Based Systems (1999) 285–291.

[16] M. Bohanec, I. Bratko, Trading accuracy for simplicity in decision trees, Machine Learning 15 (1994) 223–250.