A Quality Index for Decision Tree Pruning

D. Fournier and B. Crémilleux

GREYC, CNRS - UMR 6072
Université de Caen
F-14032 Caen Cédex France
Tel : +33 (0)2.31.56.73.79 Fax : +33 (0)2.31.56.73.30
{fournier,cremilleux}@info.unicaen.fr

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 which do not decrease the error rate can be relevant to point out some populations of specific interest or to give a representation of a large data file. We present here a new pruning method (called DI pruning). 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.

Key words: 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 from large amounts of data non trivial "nuggets" of information in an easily understandable form. 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

Preprint submitted to Elsevier Science

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 which nodes specify attributes and 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. (1984) [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, this means that these examples seem similar but in fact differ from their classes. In these situations, it is well-known (see [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 [1], [4], [2]) 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 requires declarativity during treatments. We will see in Section 3 that we include this point in our 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. We will see that the principal pruning methods are based on a classification error rate and that it may be a drawback in uncertain domains. So, in Section 3, we propose 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. From this index, we infer a new pruning method for decision trees (denoted DI pruning) which is appropriate in uncertain domains: 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 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). We think that it is a major point for the use of decision trees in uncertain domains. In Section 4, we give a short overview of the UnDeT software which implements this method. We present examples in a real world domain in which we use DI

pruning as a tool to optimize a final decision tree.

2 Motivations

The principal methods for pruning decision trees are examined in [5], [4] and [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.

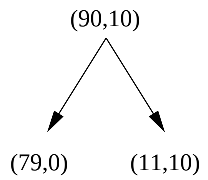

Fig. 1. A tree which could be interesting although it doesn't decrease the number of errors.

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 doesn't improve the used classification error rate. Let us consider the sub-tree depicted in Fig. 1. 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 doesn't 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's impossible to predict a value for D. We think that, in uncertain domains, cutting such a sub-tree would introduce more uncertainty than keeping the leaves.

Another way to prune decision trees, based on a quality improvement, is treated in [7] and [8]. These methods use a quality measurement which indicates an information gain brought by the tree. Even though both methods have the ability to spare subtrees like in Fig. 1, they have some limitations: in [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 [9].

We pursue here a twofold aim: on one hand, we want to make a pruning method which does not systematically discard a sub-tree which classification error rate is equal to the rate of the root, and on the other hand, we would like to handle the complexity of the trees. The following section 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 showed in Fig. 1). This method is not based on a classification error rate and we claim that it is a major point in uncertain domains.

3 Depth-Impurity quality index and pruning

3.1 Framework for a quality index

Let us formulate the question of a quality index. We 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.

We notice that these conditions are part of the usual framework to properly define suitable attribute selection criteria to build decision trees ([1], [7]). This framework states that on a theoretical level, criteria derived from an impurity measure ([10]) perform comparably ([1], [11] and [7]) [1]. Such criteria are called C.M. criteria (concave-maximum 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 C.M. criteria.

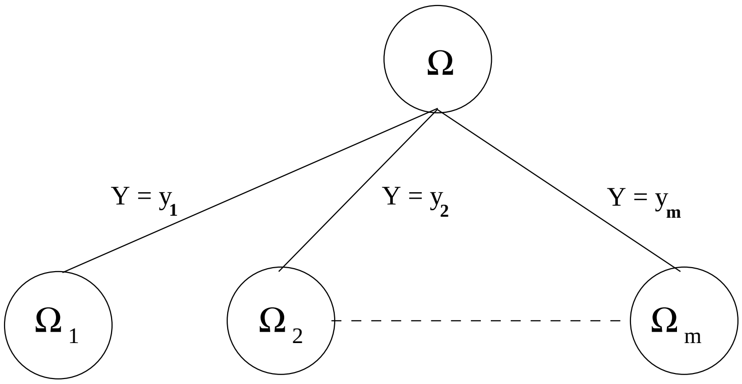

Fig. 2. Splitting of a node Ω using an attribute Y.

[1] We used here the term of impurity because it is the most usual in the machine learning community ([1] [10]).

The idea is then to use known properties (based on an impurity measure) of C.M. criteria to define a quality index. In order to better understand the genesis of the quality index that we present below, we have to take a closer look at the question of attribute selection criteria in decision trees and its usual notations. Let us call D the class of the examples of a data set, with values $d_1, \ldots, d_k$, Ω a node and Y any attribute defined on Ω, with values $y_1, \ldots, y_m$. Let us note Ψ a C.M. criterion (see Fig. 2).

We know (see [1], [10] and [7]) that $\Psi(Y) = \sum_i^m \frac{\alpha(\Omega_i)}{\alpha(\Omega)} \varphi(\Omega_i)$ where $\Omega_1, \ldots, \Omega_m$ 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. $\Psi(Y)$ 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 C.M. 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, that $\Psi(Y)$ 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.

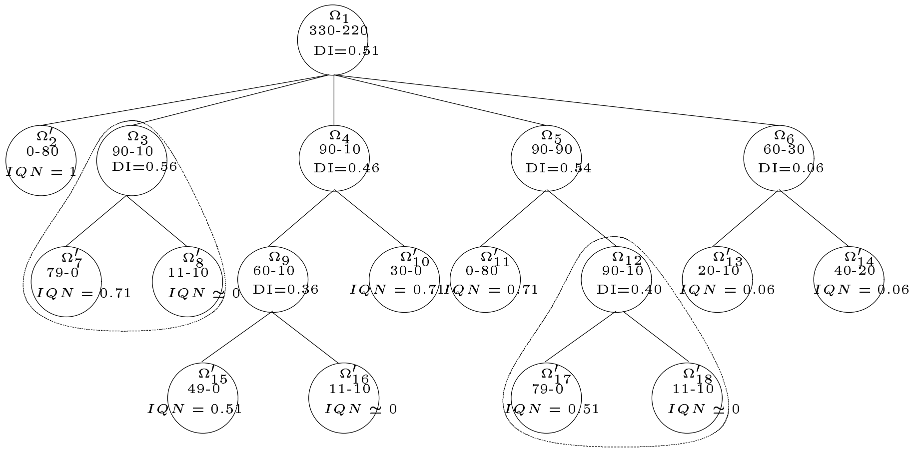

Fig. 3. Examples of DI and IQN values on a pedagogic tree.

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, we shall not forget that our aim is to offer a suitable pruning method in uncertain domains. In this context, we claim that a deep tree is less relevant than a small: the deeper a tree, the less understandable and reliable. For example, in Fig. 3, we prefer the tree which has the root denoted Ω_3 to 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 we prefer the tree with root Ω_3 because it is simpler.

So, we have to properly define a quality index which takes into account both impurities and depths of leaves. The next section presents this index.

3.2 The quality index DI

First of all, we define the quality of a node Ω (called IQN for Impurity Quality Node) as a combination between its purity and its depth. $IQN(\Omega)$ is relative to the tree T where Ω is located (so we introduce T in the notation). $IQN_T(\Omega)$ is given by Equation 1:

$$IQN_T(\Omega) = (1 - \varphi(\Omega))f(depth_T(\Omega)) \tag{1}$$

where T is the decision tree which contains the node Ω and φ is an impurity measure normalized between $[0, 1]$. Let us note that as φ is an impurity measure, $(1 - \varphi)$ defines a purity measure. By introducing a damping function (denoted 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.

Equation 2 defines the quality index DI (for Depth-Impurity) of a subtree T_s in T (we note either $DI(T_s)$ 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:

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

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

So, Equation 2 can also be written:

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

Due to the definition of IQN, the depth of a leaf is 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. If we come back to the example given at the end of the previous section (comparison between trees of root Ω_3 versus the one of root Ω_{12}), we notice that the quality index of the tree of root Ω_{12} is lower because

[2] to simplify notations, we replace $\alpha(\Omega_i)$ by α_i

it is deeper. This choice to compute the depth allows to compare 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.

We 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 we choose an impurity measure normalized between $[0, 1]$, as $\sum_i^m \frac{\alpha_i}{\alpha_s} = 1$, we will see immediately with both constraints that the value of DI is between $[0, 1]$. The higher the value of DI, the better quality of T_s.

Furthermore, we add the three following constraints:

3 $f(d) \simeq 0$ when d tends towards the total number of attributes.
4 $0 \leq f(d) \leq 1$
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: we will see that it allows to have a linear computation of DI as indicated by the remark below. 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 $f(d+1) = \beta.f(d)$, we deduce: $f(d) = \beta^{(d-1)}.f(1)$. We thus get from 2: $f(d) = \beta^{(d-1)}$. This equality implies that f is an exponential function.

We thus choose the following function f:

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

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

If we come back to the tree given in Fig. 3, we notice that the tree which has the root denoted 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: with regard to the algorithmic point of view, computation of DI is not expensive. We can easily write $DI(T_s)$ 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 us call Ω_k a son of Ω_s and Ω'_j the leaves stemmed from Ω_k. We have:

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

To consider all the sons of Ω_s:

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

As the sets of leaves of each Ω_k make a partition of the leaves of Ω_s, we have:

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

With (7), (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 Equation 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. We will now move to this subject.

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 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 we cut a subtree T_s of root Ω_s satisfying Equation (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 Equation 10. We call this method DI pruning because it goes with DI.

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

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

With $k = 1$, we find again Equation 4. 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, we can use a procedure requiring a test file [1] [6].

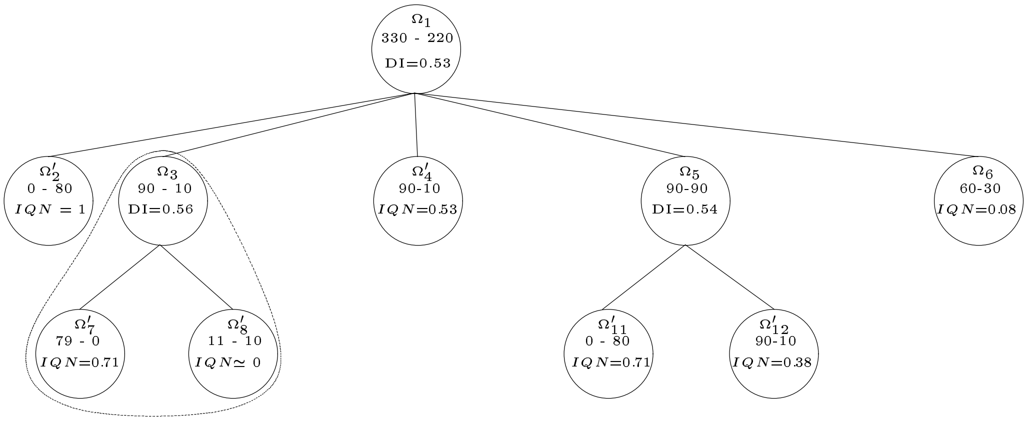

Fig. 4. Examples of DI values on the pedagogic pruned tree.

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. Let us remark that $DI(\Omega_1)$ has increased, as expected.

4 Experiments

We have designed an induction software called UnDeT (for Uncertain Decision Trees) which produces decision trees. 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 [13]). UnDeT computes IQN and DI indexes for each node and pruned trees with DI pruning. Our 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 [14]). UnDeT is included in a more general data mining toolbox in which another major part is dedicated to the treatment of missing values [15].

In this section we describe the results obtained by running UnDeT on a real world database. We perform UnDeT with the gain criterion (Shannon entropy) [12] because it is the most commonly used. 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 to 7 are snapshots produced by UnDeT).

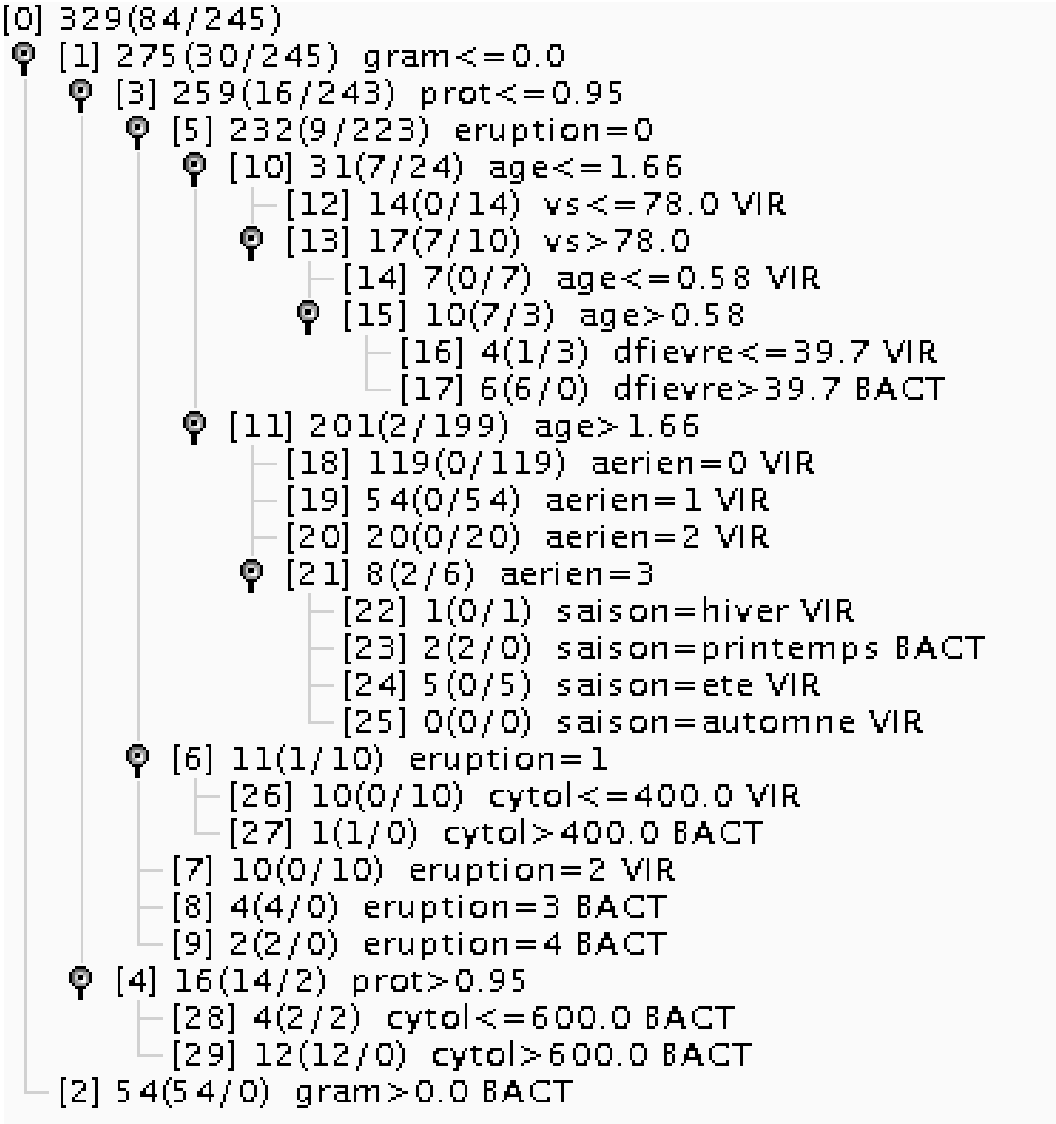

Fig. 5. Initial large tree.

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.

On the tree depicted in Fig. 5 and further, let us call T_i the sub-tree of root labelled `[i]`. T_4 has an impurity in its root equal to 0.54 (2 miss-classified instances among 16) and T_{10} has an impurity in its root equal to 0.77 (7 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 2 miss-classified instances. This number is not high regarding the 201 instances of the node. Furthermore, in order to properly classify these 2 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

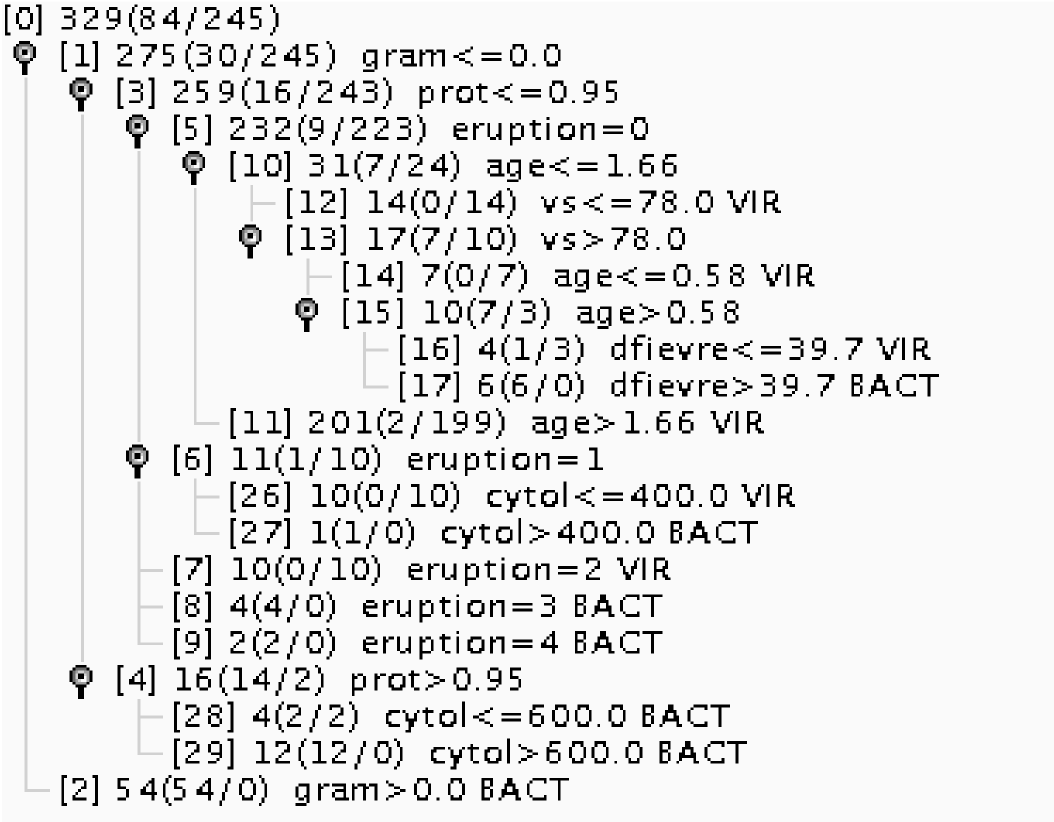

Fig. 6. Pruned tree ($k = 2$).

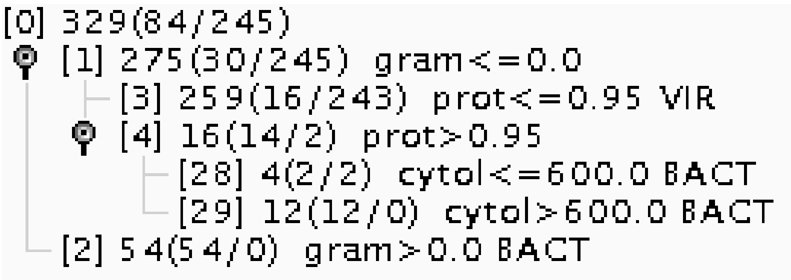

Fig. 7. Pruned tree ($k = 4$).

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 2 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 we presented in our motivations (see Sect. 2).

5 Conclusion

We 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, we 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. We 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. We will move then to an other stage of our 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, let us note that when the quality index has a low value on a subtree T_s, it suggests that T_s contains poor data for the Knowledge Discovery in Databases 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, and C. J. Stone. *Classification and regression trees.* Statistics probability series. Wadsworth, Belmont, 1984.

[2] J. R. Quinlan. Induction of decision trees. *Machine Learning 1*, pages 81–106, 1986.

[3] J. Catlett. Overpruning large decision trees. In *proceedings of the Twelfth International Joint Conference on Artificial Intelligence IJCAI 91*, pages 764–769, Sydney, Australia, 1991.

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

[5] F. Esposito, D. Malerba, and G. Semeraro. Decision tree pruning as search in the state space. In P. B. Brazdil, editor, *proceedings of European Conference*

on Machine Learning ECML 93, Lecture notes in artificial intelligence. N^o 667, pages 165–184, Vienna (Austria), 1993. Springer-Verlag.

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

[7] B. Crémilleux and C. Robert. A theoretical framework for decision trees in uncertain domains: Application to medical data sets. In E. Keravnou, C. Garbay, R. Baud, and J. Wyatt, editors, *6th Conference on Artificial Intelligence In Medicine Europe (AIME 97)*, Lecture notes in artificial intelligence. N^o 1211, pages 145–156, Grenoble (France), 1997. Springer-Verlag.

[8] L. Wehenkel. Decision tree pruning using an additive information quality measure. In B. Bouchon-Meunier, L. Valverde, and R.R. Yager, editors, *Uncertainty in Intelligent Systems*, pages 397–411. Elsevier - North Holland, 1993.

[9] L. Todorovski, P. Flach, and N. Lavrac. Predictive performance of weighted relative accuracy. In *proceedings of the fourth European Conference on Principles and Practice of Knowledge Discovery in Databases*, volume 1910, pages 255–264, Lyon (France), 2000. Springer Verlag.

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

[11] W. Buntine and T. Niblett. A further comparison of splitting rules for decision-tree induction. *Machine Learning 8*, pages 75–85, 1992.

[12] J.R Quinlan. *C4.5 Programs for machine learning.* Morgan Kaufmann, San Mateo, Californie, 1993.

[13] B. Crémilleux and C. Robert. Use of selection criteria in decision trees in uncertain domains. In B. Bouchon Meunier, R. R. Yager, and Z. A. Zadeh, editors, *Uncertainty in Intelligent and Information Systems*, pages 150–161. World Scientific, 2000.

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

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

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