On the quest for easy-to-understand splitting rules

Fernando Berzal *

Dept. Computer Science and AI, University of Granada (Spain)

Juan-Carlos Cubero

Dept. Computer Science and AI, University of Granada (Spain)

Fernando Cuenca

Xfera, Madrid (Spain)

María J. Martín-Bautista

Dept. Computer Science and AI, University of Granada (Spain)

Abstract

Decision trees are probably the most popular and commonly-used classification model. They are built recursively following a top-down approach (from general concepts to particular examples) by repeated splits of the training dataset. The chosen splitting criterion may affect the accuracy of the classifier, but not significantly. In fact, none of the proposed splitting criteria in the literature has proved to be universally better than the rest. Although they all yield similar results, their complexity varies significantly, and they are not always suitable for multi-way decision trees. Here we propose two new splitting rules which obtain similar results to other well-known criteria when used to build multi-way decision trees, while their simplicity makes them ideal for non-expert users.

Key words: supervised learning, classification, decision trees, splitting rules

* Fernando Berzal Galiano (fberzal@decsai.ugr.es), Department of Computer Science and Artificial Intelligence, E.T.S. Ingeniería Informática, University of Granada, 18071, Spain. Telephone: +34 958 242376. Fax: +34 958 243317

Email addresses: fberzal@decsai.ugr.es (Fernando Berzal), JC.Cubero@decsai.ugr.es (Juan-Carlos Cubero), fernando.cuenca@xfera.com (Fernando Cuenca), mbautis@decsai.ugr.es (María J. Martín-Bautista).

1 Introduction

Decision trees are probably the most popular and commonly-used classification model; e.g. see [16] and [5]. Decision trees are built recursively following a top-down approach (from general concepts to particular examples). That is the reason why the acronym TDIDT, which stands for Top-Down Induction on Decision Trees, is used to refer to this kind of algorithms.

TDIDT algorithms do not usually include any mechanisms to handle noise in the input data themselves; that is, they just try to obtain a perfect description of the training dataset. This is counterproductive in real problems, where management of noisy data and uncertainty is a must. Post-pruning techniques (such as those used in ASSISTANT and C4.5) have proved to be really useful in order to avoid this problem, also referred to as overfitting. Those branches with lower predictive power are usually pruned once the whole decision tree has been built.

The TDIDT algorithm family includes classical algorithms such as CLS (Concept Learning System), ID3 [15], C4.5 [17] and CART (Classification And Regression Trees) [1] and also more recent ones such as SLIQ [12], SPRINT [19], QUEST [9], PUBLIC [18], RainForest [6] and BOAT [4].

Most TDIDT algorithms use some kind of greedy heuristics to build the decision tree. Such heuristics, also known as splitting criteria, are usually hard to understand for non-expert users. In this paper we present two new splitting rules which are simpler than previous proposals in the literature and preserve the overall classifier accuracy. Similarity-based impurity measures are introduced in the following section. Classical proposals are described in Section 3. Section 4 details the process he have followed to obtain two alternative splitting rules. These splitting rules are compared with previous proposals in Section 5 using some datasets from the UCI Machine Learning Repository.

2 Impurity-based splitting criteria

Most TDIDT algorithms decide how to branch the tree using some measure of node impurity, as the information gain criterion found in ID3 [15], C4.5 gain ratio [17], and Gini index in CART [1]. That kind of heuristics tries to benefit splits which are useful to differentiate among problem classes. They attempt to build small decision trees following Occam's Economy Principle.

The node impurity makes reference to the mixture of classes in the training examples covered by the node. When all the examples in a given node belong

to the same problem class, then the node is said to be pure. The more equal proportions of classes there are in a node, the more impure the node is.

A decision tree impurity measure can be recursively obtained from the impurity measures of its leaves $\tilde{T}$ (also called terminal nodes) in the following way:

$$\phi(T) = \sum_{t \in \tilde{T}} p(t)\phi(t)$$

where $p(t)$ is the probability that a given instance is covered by the leaf t, and $\phi(t)$ is the impurity of this terminal node (t).

The goodness of a given split can be measured as the impurity decrease which is achieved in the tree when it is branched. The split goodness maximization, therefore, is equivalent to the minimization of the split-generated tree impurity, since the starting point is the same for every possible split.

An impurity function ϕ measures the impurity of a given tree node. Given a classification problem with J different classes, this function is non-negative and it is defined over the set of J-tuples $(p_1, p_2, .., p_J)$, where each p_j is the probability that a case belongs to class j in the current subtree. It is obvious that $\sum p_j = 1$. Any function ϕ must have the following properties [1]:

- The function ϕ has a unique maximum at $(1/J, 1/J, .., 1/J)$. The impurity measure is largest when there is the same number of instances beloging to each one of the problem classes (that is, the classes are distributed evenly in the node).
- The function ϕ has J minima at $\phi(1, 0, .., 0)$, $\phi(0, 1, .., 0)$ ... $\phi(0, 0, .., 1)$. Moreover, ϕ equals 0 at those points. In other words, a tree node is pure when it contains only examples of a given class.
- The function ϕ is symmetric with respect to p_1, p_2, ... p_J.

3 Classical splitting rules

Every possible test which splits the training dataset into several subsets will eventually lead to the construction of a complete decision tree, provided that at least two of the generated subsets are not empty. The final aim of the decision tree learning process is, however, to build a decision tree which conveys interesting information in order to make predictions and classify previously unseen data.

Each possible test must be evaluated using heuristics and, as most TDIDT

algorithms perform a one-ply lookahead heuristic search without backtracking (i.e. they are greedy), the selected heuristic plays an essential role during the learning process. Several splitting rules have been proposed in the literature. Here, we review some of the most relevant ones.

3.1 Information gain: Entropy

ID3 [15] attempts to maximize the information gain achieved through the use of an attribute A_i to branch the tree by minimizing function I:

$$I(A_i) = \sum_{j=1}^{M_i} p(A_{ij}) H(C|A_{ij})$$

where A_i is the attribute used to branch the tree, M_i is the number of different values for A_i, $p(A_{ij})$ is the probability of attribute A_i taking its jth value, and $H(C|A_{ij})$ is the classification entropy of the set of instances where A_i takes its jth value. This classification entropy is defined as

$$H(C|A_{ij}) = -\sum_{k=1}^{J} p(C_k|A_{ij}) \log_2 p(C_k|A_{ij})$$

where J is the number of classes in our classification problem and $p(C_k|A_{ij})$ is the probability of an example belonging to class C_k given that its attribute A_i takes its jth value.

The information conveyed by a message depends on the probability of the message p and can be expressed in bits as $-\log_2 p$, which corresponds to the number of yes-no questions needed to pick out a given message among $1/p$ different messages. For example, if we had 256 different messages (as the number of ASCII characters), each message would convey 8 bits. The probability of a randomly chosen example belonging to class C_k is $p(C_k)$ and the information obtained is $-\log_2 p$. The information we expect to obtain when classifying any example in the training set is $-\sum p(C_k) \log_2 p(C_k)$. This quantity is known as the set entropy.

The information needed to transmit the splitting of the training set T into M_i subsets T_j equals $\sum p(T_j) H(T_j)$, where $p(T_j)$ is the probability of an example belonging to subset T_j and $H(T_j)$ is the classification entropy of the set T_j.

The information gain we obtain by splitting T into the T_j subsets equals $H(T) - \sum p(T_j) H(T_j)$, where $H(T)$ is the entropy of T. The information gain achieved by alternative splits is compared to select the best partition of T. We

just have to check $-\sum p(T_j)H(T_j)$ since $H(T)$ is the same for every possible split.

This heuristic favors the construction of decision trees with a high branching factor: "it has a strong bias in favor of tests with many outcomes" [17, p. 23]. This fact gave origin to the gain ratio criterion, which is the main topic of the following subsection.

3.2 The gain ratio criterion

Using the information gain as partition criterion, we would tend to build decision trees using key or nearly-key attributes. We would obtain the optimum tree regarding to its information gain, but this tree could be useless to classify unseen data.

We can still resort to Information Theory in order to attain a normalization criterion. Let us consider the information content of the message which indicates not the class of a given instance, but the value of a given predictive attribute (i.e. an attribute which can be used to branch the tree). The information content of such a message is $-\sum p(A_{ij}) \log_2 p(A_{ij})$. Using this value, it is possible to redefine the previous splitting criterion as follows:

$$R(A_i) = \frac{H(C) - \sum_{j=1}^{M_i} p(A_{ij}) H(C|A_{ij})}{-\sum_{j=1}^{M_i} p(A_{ij}) \log_2 p(A_{ij})}$$

This ratio is the splitting criterion used in C4.5 [17]. When the training set partition is nearly trivial, the value of $R(A_i)$ approaches zero. We have to choose the attribute which maximizes $R(A_i)$ and whose information gain is at least equal to the average gain obtained with the analyzed partitions of the training set.

Since in practice we have access to many more training examples than different problem classes, the gain ratio criterion avoids the construction of decision trees which classify input instances attending to their keys (i.e. their identifiers).

It has been observed that the gain criterion tends to build quite unbalanced decision trees, a property which inherits from its predecessor, the information gain. Both heuristics are based on entropy measures which favor uneven training set partitions when one of them is of great purity (when all its training

instances belong to the same problem class), even when it does not have a high support (that is, when it covers only a few training examples).

3.3 Gini index of diversity

Gini index tries to minimize the impurity contained in the training subsets generated after branching the decision tree. It employs the following function:

$$G(A_i) = \sum_{j=1}^{M_i} p(A_{ij}) G(C|A_{ij})$$

$$G(C|A_{ij}) = -\sum_{k=1}^{J} p(C_k|A_{ij}) p(\neg C_k|A_{ij}) = 1 - \sum_{k=1}^{J} p^2(C_k|A_{ij})$$

where A_i is the attribute used to branch the tree, J is the number of classes in our classification problem, M_i is the number of different values for A_i, $p(A_{ij})$ is the probability of attribute A_i taking its jth value, $p(C_k|A_{ij})$ is the probability of an example belonging to class C_k given that its attribute A_i takes its jth value, and $p(\neg C_k|A_{ij})$ is $1 - p(C_k|A_{ij})$.

Gini index measures the class diversity in the nodes of a decision tree. This splitting criterion is used, for example, in CART [1].

As we can see from the above expressions, these formulae are quite similar to the ones we employed to define the information gain: we have just substituted $p(\neg C_k|A_{ij})$ for $-\log_2 p(C_k|A_{ij})$. In fact, both are alternative impurity measures and share their properties (remember Section 2). However, in binary decision trees, Gini index prefers splits that put the largest class into one pure node, and all others into the other, while the entropy criterion, for instance, puts its emphasis on balancing the sizes at the two children nodes [2].

3.4 Other criteria

Lopez de Mantaras [10] proposed an alternative to the gain ratio normalization. He prevented the information gain bias towards selecting variables which fragment the training set using a distance metrics:

$$LM(A_i) = \frac{H(C) - \sum_{k=1}^{M_i} p(A_{ij}) H(C|A_{ij})}{-\sum_{j=1}^{M_i} \sum_{k=1}^{J} \frac{n(C_k|A_{ij})}{N} \log_2 \frac{n(C_k|A_{ij})}{N}}$$

Taylor and Silverman [21] proposed the mean posterior improvement (MPI) criterion as an alternative to the Gini rule. Their MPI criterion was defined for binary trees and, although it can be extended to handle multi-way decision trees, it does not yield noteworthy results in latter situation. Maintaining the notation used in this paper, their criterion would be formulated as

$$MPI(A_i) = \prod_{j=1}^{M_i} p(A_{ij}) * \left(1 - \sum_{k=1}^{J} \frac{\prod_{j=1}^{M_i} p(C_k|A_{ij})}{P(C_k)} \right)$$

Other splitting criteria have been proposed in the literature during the last decade. Most of them are impurity functions as the ones presented in this section, which emphasize the purity of the split subsets, although other measures fall into different categories [11]:

- Some of them measure the difference among the split subsets using distances or angles, emphasizing the disparity of the subsets (on binary trees, typically).
- Others are statistical measures of independence (a χ^2 test, for example) between the class proportions and the split subsets, emphasizing the reliability of class predictions.

[11] includes an good survey of splitting criteria, a study on the correlation among them, and empirical results for binary decision trees. Splitting criteria for binary decision trees are also studied in [20].

Other research papers also include comparisons of several splitting rules [13] [3]. For example, [13] compares six of them and points out that understandability could be another evaluation criterion for evaluating alternative splitting rules (apart from the commonly used tree size and classification accuracy). Unfortunately, no standard measure exists for the understandability criterion. More theoretical studies on splitting rules can be found in [2] and [8]. The interested reader can find additional information and related references in Murthy's extensive survey [14].

Most of the proposed splitting rules improve the decision trees accuracy only a little in some cases (if any), but they are always more complicated than their

predecessors, i.e. Gini index and the information gain criterion. In the following section, we propose two impurity-based splitting criteria which are simpler than other proposals found in the literature without sacrificing classification accuracy.

4 Alternative splitting rules

> "Everything should be made as simple as possible, but not simpler." Albert Einstein.

Our aim is to obtain alternative splitting criteria whose formulation may be easier to understand for end-users, since the company executives and analysts who work with decision trees may feel reluctant to accept classification models just because they do not understand how they are obtained. In this section we describe the process which led us to a pair of alternative splitting criteria which can be described in terms of simple language that involves example counts rather than probability or Information Theory.

4.1 MaxDif

Impurity-based splitting criteria (such as information gain maximization) minimize entropy. The probability corresponding to the most common class is the term which increases the overall entropy the least. This probability is thus a candidate maximization criterion to measure the goodness of a given partition. We are tempted, therefore, to use the following function:

$$K(A_i) = \sum_{j=1}^{M_i} p(A_{ij}) K(C|A_{ij})$$

$$K(C|A_{ij}) = \max_k p(C_k|A_{ij})$$

where A_i is the attribute used to branch the tree, M_i is the number of different values for A_i, $p(A_{ij})$ is the probability of attribute A_i taking its jth value, and $P(C|A_{ij})$ corresponds to the evaluation of each child node. This evaluation function is just the maximum class probability $p(C_k|A_{ij})$ in the child node resulting from the jth value of attribute A_i (i.e. the maximum class probability in the training set given that attribute A_i takes its jth value).

In some sense, K is a purity-based measure. This function meets the three properties of any impurity function (see Section 2) if we consider the transformation $\sum_{j=1}^{M_i} p(A_{ij})/J - K(A_i)$. Since the function $K(A_i)$ is symmetric,

Attribute	# Values	Class distribution
A_1	2	(90, 10, 0)
		(0, 85, 15)
A_2	3	(90, 9, 0)
		(0, 84, 13)
		(0, 2, 2)

Table 1
A classification problem with three classes.

its minimum is obtained at (1/J,1/J,..,1/J), and its maxima are achieved at (1,0,..,0), (0,1,..,0), ..., and (0,0,..,1); then the above transformation demonstrates it is a purity measure.

This simple criterion, however, tends to build decision trees with a high branching factor, as happens with the entropy measure used by ID3.

For example, let us consider the example shown in table 1. The first column in this table indicates the attribute used to branch the tree. The second one shows the number of different values for that attribute, that is, the number of branches resulting from branching the tree using that attribute. Finally, the third one collects the class distribution of the training examples for each attribute value (i.e. the number of examples belonging to each class in every child node of the tree resulting from using the specified attribute to branch the tree).

Let us assume that the probability $p(X)$ is estimated as $\frac{n(X)}{N}$, where $n(X)$ is the number of examples verifying a property X and N is the total number of examples. If we used the above criterion, we would obtain $K(A_1) = \frac{90+85}{200} = \frac{175}{200}$ and $K(A_2) = \frac{90+84+2}{200} = \frac{176}{200}$. This would lead us to employ attribute A_2 to branch the tree, which is not the best decision in this case because the training dataset is unnecessarily fragmented and one of the resulting branches, the one corresponding to A_2 third value, holds an insufficient number of training examples to build a robust classification model. The presence of the slightest amount of noise in those training examples would render useless the inferred classification model.

The situation could be even worse if we had a primary key in our dataset with 200 different values. In that case, we would have an attribute which partitions the training set in 200 subsets, each one containing just one training instance. Using the above function K, we would evaluate this attribute A_{key} and obtain $K(A_{key}) = \frac{1+1+..+1}{200} = \frac{200}{200}$, which is the maximum value of our purity function K since all the child nodes are pure (i.e. containing examples of just one class).

We could reduce this side effect just by redefining the previous function as

$$K'(A_i) = \sum_{j \in U} p(A_{ij}) K(C|A_{ij})$$

$$U = \{j | \max\{n(C_k|A_{ij})\} \geq S\}$$

$$K(C|A_{ij}) = \max_k p(C_k|A_{ij})$$

where S is a user-established support threshold and $n(C_k|A_{ij})$ is the number of examples in the training set which belong to class C_k and take the jth value for attribute A_i.

The support threshold S indicates the minimum number of examples we are interested in to branch the decision tree further. In other words, we consider that training sets with less than S examples are useless to build a good classifier since they do not constitute a representative sample of the population we try to model. Support thresholds have been traditionally used in association rule mining [7] and they can be considered as the simplest pre-pruning technique for decision trees.

It should be noted, however, that our use of a support threshold differs from the conventional stopping rules used by many TDIDT software packages. We use this threshold just to cancel the contribution of small subsets such as the one which appears when using the A_2 attribute to branch the tree in the example shown in Table 1. Those small contributions to the node purity do not convey any meaningful information for the decision tree construction process because the resulting training subsets cannot be used to build reliable classification models.

We could set S to 1 by default, but in this case our classification model would be affected by any outlier in the training dataset (as most TDIDT classifiers are), and we would need further post-processing to improve our classifier accuracy. Typical decision tree pruning techniques would be useful here. Setting a value higher than 1 for the support threshold would automatically discard any small subsets. In the above example, if we had set $S = 3$, then $K'(A_1) = \frac{90+85}{200} = \frac{175}{200}$, $K'(A_2) = \frac{90+84}{200} = \frac{174}{200}$, and $K'(A_{key}) = \frac{0}{200}$. Thus, we would use A_1 to branch the tree, which seems to be the wiser decision at this point.

Taking into account that $P(C_k|A_{ij}) = n(C_k|A_{ij})/n(A_{ij})$ and also that $P(A_{ij}) = n(A_{ij})/N_i$, being N_i the total number of training examples for which attribute A_i is not null, $n(A_{ij})$ the number of instances which take the jth value of its ith attribute, and $n(C_k|A_{ij})$ the number of examples belonging to class C_k

which take A_i jth value, we can reformulate function $K'(A_i)$ in the following way:

$$K'(A_i) = \frac{1}{N_i} \sum_{j \in U} K'(C|A_{ij})$$

$$U = \{j | \max\{n(C_k|A_{ij})\} \geq S\}$$

$$K'(C|A_{ij}) = \max_k n(C_k|A_{ij})$$

If there are no missing values, the factor $\frac{1}{N_i}$ will be the same for every attribute and can be eliminated from the above expression. In this way, we obtain a splitting rule which is just a sum of support counts in accordance with our quest for simple partition rules.

We have observed, however, that the K' heuristics favors the construction of decision trees with the lowest possible branching factor (that is, binary trees), even when the attribute used to branch the tree is irrelevant to the classification task at hand. For example, in problems where one class accounts for most of the training examples, it would not be difficult to obtain decision trees which use binary tests over numeric attributes even when those attributes are not predictive. These tests send most of the training examples to one subtree and leave only a few for the other subtree. The heuristics is mixed up just because the majority class is very frequent in the large training subset and the small subset contributes a little to make the selected test attractive.

The unwanted effect which we have described in the previous paragraph could be compensated somewhat by adding an additional factor $\#U$ (i.e. the cardinality of U) to the K' function:

$$K''(A_i) = \frac{\#U}{N_i} \sum_{j \in U} K'(C|A_{ij})$$

$$U = \{j | \max\{n(C_k|A_{ij})\} \geq S\}$$

$$K'(C|A_{ij}) = \max_k n(C_k|A_{ij})$$

Intuitively, the summation in the above expression counts the number of examples which will be correctly classified within the training dataset (without branching the tree further). Dividing by N_i, we normalize that count to be able to compare different attributes which may have missing values. The final factor $\#U$ is used to bias the decision tree learning process towards flat

decision trees (that is, trees with more interesting branches).

The use of the factor $\#U$ combined with the minimum support threshold S provides an artificial mechanism to avoid using primary keys to branch the tree, although it lacks a theoretical basis. It does not guarantee the construction of good decision trees, although it sometimes improves the results obtained with more complex heuristics such as C4.5 gain ratio criterion.

We have obtained a well-behaved splitting criterion, although its artificial complexity is not desirable. The apparent dead end we have reached in our journey from $K(A_i)$ to $K''(A_i)$ can be avoided if we reformulate the original maximization problem:

Let us consider using the difference between the number of examples belonging to the most common class and the rest of examples in a particular node. This criterion substracts the number of cases which would be misclassified in a leaf to the number of correctly-classified instances. Mathematically,

$$D(A_i) = \sum_{j=1}^{M_i} p(A_{ij}) D(C|A_{ij})$$

$$D(C|A_{ij}) = \max_k \{p(C_k|A_{ij}) - p(\neg C_k|A_{ij})\}$$

This heuristics, which we call MaxDif, behaves properly without making it more complex and is equivalent to the original function K when there are no missing values in the training dataset, since $p(C_k|A_{ij}) - p(\neg C_k|A_{ij}) = 2p(C_k|A_{ij}) - 1$. As its predecessor, it satisfies the properties of any impurity-based splitting criterion if we reverse its sign.

We can also express MaxDif using support counts in order to obtain the following expressions:

$$D(A_i) = \frac{1}{N_i} \sum_{j=1}^{M_i} D(C|A_{ij})$$

$$D(C|A_{ij}) = \max_k \{n(C_k|A_{ij}) - n(\neg C_k|A_{ij})\}$$

Using again the example shown in Table 1, we would obtain the following figures:

$$D(A_1) = \frac{(90-10)+(85-15)}{200} = \frac{80+70}{200} = \frac{150}{200},$$

$D(A_2) = \frac{(90-9)+(84-13)+(2-2)}{200} = \frac{81+71+0}{200} = \frac{152}{200}$, and, even worse,

$D(A_{key}) = \frac{(1-0)+(1-0)+..+(1-0)}{200} = \frac{200}{200}$.

The above results show that this criterion still lacks the necessary properties needed to avoid the effects caused by noise and isolated outliers in the training datset. Those outliers make the decision trees overfit the training dataset and the additional pruning step a necessary evil (evil in the sense that it requires additional computing resources).

We should also face the problem of primary keys and small training subsets. As we did before, we can introduce a minimum support threshold constraint in the following way:

$$D'(A_i) = \frac{1}{N_i} \sum_{j \in U} D(C|A_{ij})$$

$$U = \{j | \max\{n(C_k|A_{ij})\} \geq S\}$$

$$D(C|A_{ij}) = \max_k \{n(C_k|A_{ij}) - n(\neg C_k|A_{ij})\}$$

After some unsuccessful attempts, we have finally obtained a relatively simple splitting criterion which performs well in most situations (see our experimental results in Section 5): we decide how to branch a decision tree just by counting how many examples are sent to each subtree and the number of instances corresponding to the most common class in each subtree. We subtract the number of misclassified examples in a node from the number of correctly-classified instances (supposing that the tree will not be branched further) and that is our heuristic evaluation of that node. We aggregate our evaluations of every child node using a weighted average as usual, and obtain the figure which will be used to decide how to branch the tree (that is, which attribute will be used to branch the tree at the current node).

An additional minimum support constraint is included to prevent the effects of keys and outliers in the training set. The minimum support constraint can actually be used with any other impurity-based splitting criterion in order to improve the performance of any TDIDT algorithm. The effect of primary keys and isolated outliers on TDIDT classifiers can be removed just by using a minimum support threshold above 1 tuple. A higher support threshold, between 1% and 5% of the training data set size, would also allow us to prevent the undesirable contribution of scarcely populated branches to the overall node impurity measure and, thus, it can be used to address the tendency of entropy-based impurity measures to build quite unbalanced decision trees, as discussed

in Section 3.2.

4.2 Generalized Gini

Some similarities exist between Gini index and our MaxDif criterion when no minimum support threshold is used: the summation is substituted by a maximum and the product by a substraction. This suggests us another approach in our quest for easy-to-understand splitting rules.

We could try to generalize Gini index criterion using $\odot$ and $\oplus$ operators instead of the typical arithmetic operations:

$$GG_1(A_i) = \sum_{j=1}^{M_i} p(A_{ij}) GG_1(C|A_{ij})$$

$$GG_1(C|A_{ij}) = \bigoplus_{k=1..J} (p(C_k|A_{ij}) \odot p(\neg C_k|A_{ij}))$$

It should be noted that the first equation is not altered because it just performs a weighted sum of the child nodes evaluations.

The formula above is still too complex to be amenable to further simplification, so we will use an alternative formulation of Gini index:

$$G(A_i) = \sum_{j=1}^{M_i} p(A_{ij}) G(C|A_{ij})$$

$$G(C|A_{ij}) = 1 - \sum_{k=1}^{J} p^2(C_k|A_{ij})$$

Loosening the above formulae, we obtain the following expressions:

$$GG_2(A_i) = \sum_{j=1}^{M_i} p(A_{ij}) GG_2(C|A_{ij})$$

$$GG_2(C|A_{ij}) = 1 - \bigoplus_{k=1..J} (p(C_k|A_{ij}) \odot p(C_k|A_{ij}))$$

We can simplify the above equation by removing the $\odot$ operator, since $p^2 \leq p$ between 0 and 1, and underestimating the node impurity in $GG_2(C|A_{ij})$ does not significantly affect the overall heuristics performance. In this way, we obtain the following expressions for our division rule:

$$GG_3(A_i) = \sum_{j=1}^{M_i} p(A_{ij}) GG_3(C|A_{ij})$$

$$GG_3(C|A_{ij}) = 1 - \bigoplus_{k=1..J} p(C_k|A_{ij})$$

When we are trying to split the training set, we assume that a local decision will yield good results whatever happens hereafter. If we have to choose between a node whose class distribution is (0.6,0.4,0.0) and another node (0.6,0.3,0.1), we usually opt for the former one, although we have no evidence that it will lead to a better decision subtree. For example, there could exist a 3-valued attribute which differentiates among the three classes perfectly in the apparently worst node (0.6,0.3,0.1). Just by branching the subtree at that point we would attain a 100% classification accuracy, something which might be impossible if we had chosen the supposedly more promising first alternative (0.6,0.4,0.0).

When we arrive at a node in the decision tree during TDIDT learning, we cannot assume that a given node will lead to a better decision tree than another one if the probability of the most common class is the same in both nodes. That would be the training classification accuracy for both nodes if we were unable to branch the tree further. Thus, we could simplify our splitting rule taking that fact into account:

$$GG(A_i) = \sum_{j=1}^{M_i} p(A_{ij}) GG(C|A_{ij})$$

$$GG(C|A_{ij}) = 1 - \max_{k=1..J} p(C_k|A_{ij})$$

This function, in fact, is quite similar to the original function $K(A_i)$ we used to derive MaxDif and, unlike that initial attempt, it allows the construction of good decision trees keeping the splitting criterion complexity to a minimum.

Using support counts instead of probabilities and a minimum support threshold as we did in MaxDif, we obtain the following equations:

$$GG'(A_i) = \frac{1}{N_i} \sum_{j \in U} GG'_n(C|A_{ij})$$

$$U = \{j | \max\{n(C_k|A_{ij})\} \geq S\}$$

$$GG'_n(C|A_{ij}) = n(A_{ij}) - \max_{k=1..J} n(C_k|A_{ij})$$

We just have to count the number of instances belonging to the most common class in each node to obtain the number of examples which would be misclassified if the tree were not branched further (i.e. $GG'_n(C|A_{ij})$) , and aggregate those counts into a unique value in the usual way.

4.3 Summary

In this section, we have obtained two functions which can be used as splitting criteria to build multi-way decision trees:

$$D(A_i) = \sum_{j=1}^{M_i} p(A_{ij}) D(C|A_{ij})$$

$$D(C|A_{ij}) = \max_k \{p(C_k|A_{ij}) - p(\neg C_k|A_{ij})\}$$

and

$$GG(A_i) = \sum_{j=1}^{M_i} p(A_{ij}) GG(C|A_{ij})$$

$$GG(C|A_{ij}) = 1 - \max_k p(C_k|A_{ij})$$

As previous splitting rules, they perform a weighted sum of impurity measurements over the resulting subtrees. It is remarkable that both of them depend only on the probability of the most common class in each subtree and still yield excellent results. In fact, that probability is the only true evidence we have to compare alternative splits when building decision trees.

Moreover, a minimum support threshold can be used to improve TDIDT performance in the presence of keys and noise in the training dataset. This threshold provides a simple and powerful mechanism to avoid common problems in decision tree learning, as the tendency to build quite unbalanced trees when some branches are pure even if they hold only a few training examples. It

should be noted that this threshold could also be used with other splitting criteria.

5 Empirical results

We have chosen multi-way splits because knowledge workers (the executives, analysts, and managers who employ decision support systems) feel more comfortable with them than with binary trees. Binary trees have less leaves, but are deeper and their training time is longer, as indicated by [11]. Shallow trees are preferred because they are easier to understand and multi-way decision trees tend to be more shallow.

Tables 2 and 3 summarize the results we have obtained building multi-way decision trees for some datasets obtained from the UCI Machine Learning Repository (*http://www.ics.uci.edu/∼mlearn/MLRepository.html*). Table 2 shows TDIDT classification accuracy before pruning and Table 3 shows the results when pessimistic pruning is used (with CF=0.25).

Primary keys were removed from those datasets in order to avoid the problems they cause when the information gain criterion is used: algorithms such as ID3 would directly choose them to branch the tree and the resulting classifiers would be useless.

All the classification accuracy results reported in this paper were obtained using 10-CV (ten-fold cross-validation) without any kind of a minimum support constraint because primary keys were removed beforehand.

The accompanying tables include the results obtained using three splitting rules based on Information Theory concepts (the entropy-based information gain criterion used by ID, the gain ratio criterion proposed in C4.5, and Lopez de Mantaras' alternative to the gain ratio normalization [LM]), as well as Gini index and our two proposals. The intermediate criteria we have described in the preceding section are not included in these tables because they were explained just to illustrate the process which led us to MaxDif and GG.

All the studied splitting criteria exhibit a similar behaviour. No rule is always better than the others. Even ID3 information gain criterion can be better than its alleged improvements (C4.5 gain ratio and Lopez de Mantaras' rule) when primary keys are removed from the training dataset. In fact, in problems with a small number of classes, all criteria should produce similar results [2]. The accuracy differences among the alternative splitting criteria are not significant (below 2% on average when the deviation among the experiments in a given cross-validation is above 4% on average).

Even without primary keys in the input datasets, the use of a minimum support threshold can improve the overall classifier accuracy in specific situations. As a mechanism to avoid overfitting, such a threshold cancels the contributions of small data subsets to the overall purity measure for a given node in the decision tree. For example, using the Bupa data set and MaxDif as impurity measure, if we set the minimum support threshold to 5% of the tuples in that data set, we improve classifier accuracy up to 67% before pruning and outperform the results obtained without any support constraint. Different support constraints could be suitable for other datasets and there are datasets for which there is no suitable support constraint. It should be taken into account that the suitable value for the support threshold depends on the nature of the training dataset and on the preprocessing steps performed to clean this dataset. It should also be noted that a minimum support threshold in the splitting criterion can be applied to any of the splitting rules discussed in this article, since its use is orthogonal to the splitting rule used to branch the tree. The above considerations explain why we have not included a separate set of experiments using different support thresholds.

Our simplified splitting criteria proposals seem to obtain worse results in problems where there are many different classes and a small number of training examples for some of them (as in the SOYBEAN dataset, which includes 19 different classes and only 683 examples), just because focusing their attention only on the most common class provides less information to build an accurate decision tree in that kind of problems. In fact, when the number of classes is high, the original Gini index may also produce splits that are too unbalanced [2] and degrade its performance. This problem would disappear if we could define a taxonomy for the problem classes and build a decision tree of decision trees (in some sense, a second order decision tree classifier). This is an open research problem which could eventually lead to interesting results.

During our experiments we have also found that MaxDif tends to build smaller decision trees than its counterparts before and after pruning, as Tables 4 and 5 illustrate. In fact, it seems that MaxDif trades a little accuracy for a noteworthy decrease in tree complexity. Unfortunately, no theoretical result supports this fact.

Despite the problems encountered with some particular datasets, our experiments show that our alternative splitting rules obtain results which are always comparable to those obtained by more complex criteria. It should be noted that our proposals are not random. They verify some properties which are required for any impurity-based measure without unnecessary complexities. Since they are easier to formulate (and, thus, easier to understand by non-expert users) than previous proposals, we believe that their use can be justified.

6 Conclusion

No particular splitting rule does significantly improve classification accuracy for decision tree classifiers in TDIDT algorithms using standard splitting criteria such as ID3 information gain.

However, some of them are notably more complex to formulate than others. This fact makes TDIDT algorithms somewhat mysterious and harder to understand for lay users. In turn, this poor understanding of the process which leads to a given decision tree can provoke some reticences from executives, analysts and managers. These knowledge workers often refrain from using classification models because they do not understand how they are obtained.

In this paper we have presented two alternative splitting criteria whose formulation is easier to understand for people without previous exposure to Information Theory concepts and advanced statistics topics. Both rules depend on the probability of the most common class in each subtree exclusively, and both of them obtain results which, even though they are not significantly better than previous criteria, are comparable to the outcome of more complex rules.

References

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

[2] Breiman, L. (1996). *Techical note: Some properties of splitting criteria*. Machine Learning, vol. 24, no. 1, pp. 41-47.

[3] Buntine, W.L., and Niblett, T. (1992). *A Further Comparison of Splitting Rules for Decision-Tree Induction*. Machine Learning, vol. 8, no. 1, pp. 75-85.

[4] Gehrke, J., Ganti, V., Ramakrishnan, R., and Loh, W.-Y. (1999a). *BOAT - Optimistic Decision Tree Construction*. Proceedings of the 1999 ACM SIGMOD international conference on Management of Data, May 31 - June 3, 1999, Philadelphia, PA USA, pp. 169-180

[5] Gehrke, J., Loh, W.-Y., and Ramakrishnan, R. (1999b). *Classification and regression: money can grow on trees*. Tutorial notes for ACM SIGKDD 1999 international conference on Knowledge Discovery and Data Mining, August 15-18, 1999, San Diego, California, USA, pp. 1-73

[6] Gehrke, J, Ramakrishnan, R., and Ganti, V. (2000). *RainForest - A Framework for Fast Decision Tree Construction of Large Datasets*. Data Mining and Knowledge Discovery, Volume 4, Numbers 2/3, July 2000, pp. 127-162

[7] Hipp, J., Güntzer, U., and Nakhaeizadeh, G. (2000). *Algorithms for Association Rule Mining - A General Survey and Comparison*. SIGKDD Explorations, Volume 2, Issue 1, June 2000, pp. 58-64

[8] Kononenko, I. (1995). *On biases in estimating multi-valued attributes*. In IJCAI-95, pp. 1034-1040.

[9] Loh, W.-Y., and Shih, Y.-S. (1997). *Split Selection Methods for Classification Trees*. Statistica Sinica, Vol.7, 1997, pp. 815-840

[10] Lopez de Mantaras, R. (1991). *A Distance-Based Attribute Selection Measure for Decision Tree Induction*. Machine Learning, 6, pp. 81-92.

[11] Martin, J. K. (1997). *An Exact Probability Metric for Decision Tree Splitting and Stopping*. Machine Learning, 28, pp. 257-291.

[12] Mehta, M., Agrawal, R., and Rissanen, J. (1996). *SLIQ: A Fast Scalable Classifier for Data Mining*. Advances in Database Technology - Proceedings of the Fifth International Conference on Extending Database Technology (EDBT'96), Avignon, France, March 25-29, 1996, pp. 18-32

[13] Mingers, J. (1989). *An empirical comparison of selection measures for decision-tree induction*. Machine Learning, vol. 3, no. 4, pp. 319-342.

[14] Murthy, S.K. (1998). *Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey*. Data Mining and Knowledge Discovery, vol. 2, no. 4, pp. 345-389.

[15] Quinlan, J.R. (1986a). *Induction on Decision Trees*. Machine Learning, 1, 1986, pp. 81-106

[16] Quinlan, J.R. (1986b). *Learning Decision Tree Classifiers*. ACM Computing Surveys, 28:1, March 1986, pp. 71-72

[17] Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[18] Rastogi, R., and Shim, K. (1998). *PUBLIC: A Decision Tree Classifier that integrates building and pruning*. VLDB'98, Proceedings of 24th International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA, pp. 404-415

[19] Shafer, J.C., Agrawal, R., and Mehta, M. (1996). *SPRINT: A Scalable Parallel Classifier for Data Mining*. VLDB'96, Proceedings of 22nd International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India, pp. 544-555

[20] Shih, Y.-S. (1999). *Families of splitting criteria for classification trees*. Statistics and Computing, vol.9, no.4; Oct. 1999; pp. 309-315.

[21] Taylor, P. C., and Silverman, B. W. (1993). *Block diagrams and splitting criteria for classification trees*. Statistics and Computing, vol. 3, no. 4, pp. 147-161.

Dataset	Entropy	GainRatio	LM	Gini	GG	MaxDif
Audiology	76.00%	**81.81%**	78.70%	75.45%	80.45%	74.68%
Australian	80.72%	78.70%	82.46%	80.87%	80.14%	**84.35%**
Bupa	62.40%	62.03%	63.26%	**66.45%**	60.62%	63.01%
Car	**94.73%**	94.50%	94.56%	94.50%	89.07%	93.46%
Glass	72.40%	71.10%	72.40%	69.13%	**72.79%**	71.95%
Hayes-Roth	73.75%	**75.00%**	74.38%	73.13%	74.38%	74.38%
Heart	**74.07%**	71.85%	70.00%	73.33%	68.52%	71.85%
Ionosphere	89.45%	91.74%	**93.18%**	90.32%	87.18%	90.89%
Iris	**95.33%**	**95.33%**	94.00%	**95.33%**	94.67%	94.67%
Mushroom	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**
Nursery	**98.89%**	98.80%	98.80%	**98.89%**	97.39%	97.77%
Pima	71.22%	70.17%	67.68%	70.83%	66.91%	**72.50%**
Soybean	89.60%	92.97%	**93.41%**	91.65%	88.00%	87.56%
Splice	91.24%	**91.72%**	91.34%	90.17%	85.35%	82.80%
TicTacToe	85.29%	**86.96%**	86.44%	85.49%	83.72%	83.93%
Titanic	**79.05%**	**79.05%**	**79.05%**	**79.05%**	**79.05%**	**79.05%**
Votes	94.93%	95.16%	94.93%	94.93%	95.62%	**95.85%**
Waveform	75.92%	76.50%	71.72%	75.26%	72.00%	**76.86%**
Wine	94.41%	**95.52%**	94.93%	89.95%	92.19%	89.93%
Average	84.12%	84.64%	84.25%	83.85%	82.48%	83.28%

Table 2
TDIDT classification accuracy obtained with different splitting criteria using 10-CV (before pruning)

Dataset	Entropy	GainRatio	LM	Gini	GG	MaxDif
Audiology	72.51%	**81.36%**	79.57%	75.18%	77.81%	71.58%
Australian	85.07%	84.06%	85.51%	**86.52%**	83.62%	85.07%
Bupa	64.71%	64.36%	61.53%	**65.87%**	59.18%	62.43%
Car	**93.46%**	92.88%	93.00%	93.00%	85.60%	92.07%
Glass	70.50%	68.70%	71.47%	71.49%	**72.90%**	71.49%
Hayes-Roth	72.50%	73.75%	73.13%	73.75%	72.50%	**74.38%**
Heart	**80.00%**	75.93%	74.07%	77.04%	73.33%	72.96%
Ionosphere	88.60%	91.45%	**92.62%**	89.75%	89.17%	90.32%
Iris	**95.33%**	**95.33%**	94.67%	**95.33%**	94.67%	93.33%
Mushroom	**100.00%**	**100.00%**	**100.00%**	99.99%	**100.00%**	**100.00%**
Nursery	96.17%	96.17%	96.17%	**96.52%**	95.08%	94.35%
Pima	**72.65%**	72.38%	70.16%	71.60%	68.73%	71.46%
Soybean	90.62%	93.70%	93.71%	91.95%	88.00%	88.15%
Splice	93.51%	**94.08%**	93.80%	93.54%	89.48%	89.83%
TicTacToe	**84.35%**	83.82%	83.62%	**84.35%**	83.82%	83.93%
Titanic	**79.05%**	**79.05%**	**79.05%**	**79.05%**	**79.05%**	**79.05%**
Votes	94.71%	**95.86%**	**95.86%**	94.95%	94.03%	94.48%
Waveform	76.90%	76.94%	72.76%	76.16%	74.66%	**76.98%**
Wine	**94.97%**	94.41%	94.48%	90.45%	91.60%	89.35%
Average	84.51%	84.96%	84.48%	84.55%	82.80%	83.22%

Table 3
TDIDT classification accuracy obtained with different splitting criteria using 10-CV and pessimistic pruning (CF=0.25)

Dataset	Entropy	GainRatio	LM	Gini	GG	MaxDif
AUDIOLOGY	113.6	100.2	97.0	108.6	115.8	119.9
AUSTRALIAN	197.3	230.1	193.7	210.3	297.4	99.7
BUPA	140.8	139.0	180.0	144.4	220.0	35.4
CAR	373.3	369.1	369.4	373.0	567.1	382.3
GLASS	76.0	74.6	82.8	90.0	111.0	58.8
HAYESROTH	53.1	53.5	53.4	53.3	53.7	53.7
HEART	80.2	87.5	98.7	83.8	136.2	35.6
IONOSPHERE	37.0	37.0	43.2	44.0	87.0	23.2
IRIS	16.4	16.4	18.0	16.4	26.2	7.2
MUSHROOM	29.0	25.0	25.0	29.6	29.9	31.3
NURSERY	1126.4	1125.4	1124.8	1070.1	1848.1	1993.4
PIMA	232.8	241.2	309.2	244.8	401.0	55.2
SOYBEAN	157.5	124.6	127.7	157.8	203.9	190.9
SPLICE	491.7	466.8	465.2	513.2	824.6	883.5
TICTACTOE	300.3	291.6	294.6	295.2	363.1	351.3
TITANIC	15.0	15.0	12.0	15.0	14.1	15.0
VOTES	54.4	58.1	56.5	55.9	96.9	78.9
WAVEFORM	862.0	862.8	1146.0	985.2	1721.2	479.0
WINE	15.8	14.8	12.4	18.6	29.4	18.4
Relative tree complexity	150%	151%	168%	158%	238%	100%

Table 4
Average number of tree nodes using different splitting criteria and 10-CV (before pruning)

Dataset	Entropy	GainRatio	LM	Gini	GG	MaxDif
AUDIOLOGY	38.6	67.0	55.4	42.1	51.7	62.2
AUSTRALIAN	43.6	30.7	64.3	42.7	48.9	37.6
BUPA	62.8	63.4	107.2	60.2	72.4	27.2
CAR	164.8	162.0	163.0	161.6	193.2	153.3
GLASS	41.8	44.4	54.6	46.6	50.0	41.0
HAYESROTH	24.8	24.8	24.8	24.8	24.9	24.9
HEART	35.6	36.6	44.4	38.2	33.3	28.2
IONOSPHERE	25.4	28.4	19.8	26.6	24.6	21.8
IRIS	7.0	7.0	8.2	7.0	7.0	6.2
MUSHROOM	29.0	25.0	25.0	28.5	28.8	29.1
NURSERY	406.4	406.4	406.4	400.4	498.0	494.8
PIMA	110.6	116.8	163.0	122.0	127.6	41.6
SOYBEAN	87.3	73.9	77.1	91.0	110.6	105.2
SPLICE	134.6	140.0	144.8	136.3	171.4	121.3
TICTACTOE	123.9	120.6	120.8	124.8	120.6	118.1
TITANIC	15.0	15.0	12.0	15.0	14.1	15.0
VOTES	13.3	12.9	13.0	13.5	8.4	5.3
WAVEFORM	549.8	582.8	807.8	581.0	786.6	373.8
WINE	9.6	9.6	8.6	11.6	15.2	14.6
Relative tree complexity	126%	127%	149%	129%	137%	100%

Table 5
Average number of nodes using different splitting criteria, 10-CV and pessimistic pruning (CF=0.25)