

On the Well-Behavedness of Important Attribute Evaluation Functions

Tapio Elomaa

Institute for Systems, Informatics and Safety
Joint Research Centre, European Commission
TP 270, I-21020 Ispra (Va), Italy
tapio.elomaa@jrc.it

Juho Rousu

VTT Biotechnology and Food Research
Tietotie 2, P. O. Box 1501
FIN-02044 VTT, Finland
Juho.Rousu@vtt.fi

Abstract. The class of *well-behaved* evaluation functions simplifies and makes efficient the handling of numerical attributes; for them it suffices to concentrate on the *boundary points* in searching for the optimal partition. This holds always for binary partitions and also for multisplits if only the function is *cumulative* in addition to being well-behaved. The class of well-behaved evaluation functions is a proper superclass of convex evaluation functions. Thus, a large proportion of the most important attribute evaluation functions are well-behaved. This paper explores the extent and boundaries of well-behaved functions. In particular, we examine C4.5's default attribute evaluation function *gain ratio*, which has been known to have problems with numerical attributes. We show that gain ratio is not convex, but is still well-behaved with respect to binary partitioning. However, it cannot handle higher arity partitioning well. Our empirical experiments show that a very simple cumulative rectification to the poor bias of *information gain* significantly outperforms gain ratio.

1 Introduction

In decision tree learning the sample is recursively divided into successively smaller subsets in an attempt to discover a final partition in which the class distribution of subsets reflects the classification of (future) instances. In top-down induction of decision trees (TDIDT) the correlation of an instance's class and its attribute values is heuristically approximated by an *evaluation function*, or a *goodness criterion*. In the most basic scheme a single attribute's value is the basis of sample partitioning: the data is greedily divided into subsets according to the value of that attribute, which evaluates as the best.

If the attribute in question has a (small) nominal domain, it is simple to continue tree construction by growing a subtree corresponding to each separate value in the domain. Handling a large nominal domain can turn out more problematic, since the evaluation functions tend to have biases for (or against) multivalued attributes [15, 11]. Many new evaluation functions have been developed to address this deficiency [8, 12, 14].

The real challenge for decision tree learning, however, is issued by *numerical attributes* with a very large, even infinite domain, which can be either discrete (integer) or continuous (real). Numerical attributes cannot be handled by the TDIDT scheme quite as naturally as nominal attributes. In real-world induction tasks numerical attribute ranges, nevertheless, regularly appear, and therefore, they also need to be taken into account in decision tree construction [4, 7, 8, 9, 16, 17, 18]. Independent of which evaluation function is used to determine the goodness of an attribute, a numerical attribute's value range needs to be categorized into two or more intervals for evaluation.

When discretizing a numerical value range we inherently meet the problem of choosing the right arity for the partitioning; i.e., into how many intervals should we split the range. This is often evaded in practical decision tree learners by using *binarization* [3, 16], in which the value range is split into only two intervals at a time (in one node). Further partitioning of the domain is generated by continuing the binarization of previously induced intervals further down the tree.

An alternative approach uses *greedy multisplitting* [4, 8], where the value range is similarly partitioned by recursive binarization, but at once; the resulting multisplit is assigned to a single node of the evolving tree. Neither of the above-mentioned methods can guarantee the quality (as measured by the evaluation function) of the resulting partition. Algorithms for finding *optimal multisplits* have been devised by Fulton, Kasif, and Salzberg [9] and by Elomaa and Rousu [6]. These methods finally make it possible to discretize a numerical value range into intervals that are determined best by the evaluation function.

Independent of the algorithmic strategy that is used for partitioning the numerical value ranges it is important to ensure that obviously “bad” partitions are not selected by the function, i.e., that the function is well-behaved. A manifestation of poor behavior of an evaluation function is a partition that breaks up a sequence of successive examples of the same class, thus needlessly separating examples of a single class into different subsets.

This paper recapitulates the basic definitions and results that are known about numerical value range discretization. They come mainly from Fayyad and Irani [7] and Elomaa and Rousu [6]. We explore the extent of the class of well-behaved evaluation functions. As a special case, we examine the gain ratio function [14] which, along with information gain, is one of the most widely used evaluation functions due to its status as the default evaluation criterion in C4.5 [16] decision tree learner. We show that gain ratio is not a convex evaluation function [3, 2, 7]. Nevertheless, it turns out to be well-behaved with respect to binary partitioning, but is shown to be unable to guarantee good behavior with respect to higher arity partitioning. Finally, we investigate empirically the benefits of well-behavedness of an evaluation function for the predictive accuracy of a decision tree.

2 Partitioning numerical value ranges

The basic setting that we consider is the following. At our disposal we have preclassified data acquired from the application domain. The intent is to find a good predictor for classifying, on the basis of the given attributes, further instances from the same application domain. For that end the greedy TDIDT construction algorithm needs to approximate how well an attribute’s values correlate with the classification of examples. The attributes respective correlations are then compared and the one that appears best is chosen to the evolving tree. For correlation comparison a fixed evaluation function is applied.

In numerical attribute discretization the underlying assumption is that we have sorted our n examples into (ascending) order according to the value of a numerical attribute A . With this sorted sequence of examples in hand we try to elicit the best possible (in class prediction) binary or multiway partition along the dimension determined by attribute A .

Let $\text{val}_A(s)$ denote the value of the attribute A in the example s . A *partition* $\bigcup_{i=1}^k S_i$ of sample S into k intervals consists of non-empty, disjoint subsets that cover the whole domain. If partition $\bigcup_{i=1}^k S_i$ has been induced on the basis of attribute A , then for all $i < j$, if $s_i \in S_i$ and $s_j \in S_j$, then $\text{val}_A(s_i) < \text{val}_A(s_j)$. When splitting a set S of examples on the basis of the value of an attribute A , then there is a set of thresholds $\{T_1, \dots, T_{k-1}\} \subseteq \text{Dom}(A)$ that

defines a partition $\bigcup_{i=1}^k S_i$ for the sample in an obvious manner:

$$S_i = \begin{cases} \{s \in S \mid \text{val}_A(s) \leq T_1\} & \text{if } i = 1, \\ \{s \in S \mid T_{i-1} < \text{val}_A(s) \leq T_i\} & \text{if } 1 < i < k, \\ \{s \in S \mid T_{k-1} < \text{val}_A(s)\} & \text{if } i = k. \end{cases}$$

A partition is a function from the domain of the numerical attribute into the intervals. Hence, a partition cannot be realized so that two examples with an equal value for that attribute would belong to different intervals. Therefore, a standard technique in numerical value range discretization—used e.g., in C4.5 [16]—is to consider a categorized version of the data. That is, to throw all examples that have the same value for the attribute in question into a common *bin* and consider only thresholds in between bins as potential *cut points*. This is a technique that can, and should, be applied regardless of the particular evaluation function that is being used. Note that typically $V \ll n$, where V is the number of values in a range of a numerical attribute.

The goodness criteria that are used to evaluate candidate partitions are many (see e.g., [11]). A significant portion of them belong to the class of *impurity measures*. Perhaps the most important of them is the *average class entropy*, which is applied, e.g., in the ID3 [14] decision tree learner. Let $\bigcup_{i=1}^k S_i$ be a partition of S , then by $ACE(\bigcup_{i=1}^k S_i)$ we denote the average class entropy of the partition:

$$ACE(\bigcup_{i=1}^k S_i) = \sum_{i=1}^k \frac{|S_i|}{|S|} H(S_i) = \frac{1}{n} \sum_{i=1}^k |S_i| H(S_i),$$

where H is the entropy function,

$$H(S) = - \sum_{i=1}^m P(C_i, S) \log_2 P(C_i, S),$$

in which m denotes the number of classes and $P(C, S)$ stands for the proportion of examples in S that have class C . Impurity measures tend to be *cumulative*; i.e., the impurity of a partition is obtained by (weighted) summation over the impurities of its intervals.

Definition 1 Let $\bigcup_{i=1}^k S_i$ be a partition of the example set S . An evaluation function F is cumulative if there exists a function f such that $F(\bigcup_{i=1}^k S_i) = c_S \cdot \sum_{i=1}^k f(S_i)$, where c_S is an arbitrary constant coefficient (whose value may depend on S , but not on its partitions).

Cumulative evaluation functions have the benefit that finding the partition of optimal goodness is efficient. This is particularly valuable when multisplitting is considered as the number of candidate partitions increases exponentially in the arity of the partition. For cumulative functions, dynamic programming can be used to elicit the best partition in time quadratic in the number of bins [6].

3 The well-behavedness of evaluation functions

The aim in numerical value discretization is to obtain as pure partition as possible, i.e., such a partition that its subsets are as uniform as possible in what regards the class membership of their elements. Therefore, it is important for an evaluation function to favor partitions that keep examples of a same class together, whenever possible. In particular, a sequence of examples that belong to a single class should not be broken apart. With this behavior in mind, Fayyad and Irani [7] defined the profound concept of a *boundary point*:

Definition 2 (Fayyad and Irani [7]) A value T in the range of the attribute A is a boundary point iff in the sequence of examples sorted by the value of A , there exist two examples $s_1, s_2 \in S$, having different classes, such that $\text{val}_A(s_1) < T < \text{val}_A(s_2)$; and there exists no other example $s \in S$ such that $\text{val}_A(s_1) < \text{val}_A(s) < \text{val}_A(s_2)$.

This definition can be interpreted as follows: a boundary point is a point between two successive bins such that there exist a pair of examples, one from each bin, which belong to different classes. In other words, cut point that separates two successive bins of examples that all belong to a single class is not a boundary point.

We call the intervals separated by boundary points as *blocks*. Let B be the number of blocks in the domain. For any relevant attribute $B \ll n$, since otherwise there clearly is no correlation between the value of the attribute and the examples' classification [7]. Since boundary points are taken from among the potential cut points, it is clear that $B \leq V$ always holds.

Fayyad and Irani [7] proved that average class entropy is a well-behaved evaluation function in the sense that it obtains its optimal (minimal) value on a boundary point. Their proof is based on the fact that that average class entropy is *convex* (downwards) in between any two boundary points. Thus, average class entropy's minimum values for binary partitions can only occur at boundary points.

Convexity of a given function is a property that can be easily detected by observing its (first and second) derivatives. However, it can be a needlessly restrictive requirement to pose to an evaluation function. Independent of the shape of the function curve, for teleologically good behavior, as regards binary partitions, it suffices that the function receives its minimum value at a boundary point. Convex functions, of course, also possess this property and, hence, constitute a subclass of well-behaved evaluation functions.

In the following, by shorthand notation $F(T)$ we denote the value of the evaluation function F for the (binary) partition that is defined by the cut point T . Also, we consider a set of boundary points that is *augmented* by the low and high extremes of the value range as to ensure that each example set always has at least two boundary points.

Definition 3 Let S be an arbitrary example set and let $\mathcal{T} = \{T_0, \dots, T_{B+1}\}$ be the augmented set of boundary points in S in the dimension determined by an arbitrary numerical attribute A . An evaluation function F is well-behaved if there exists a value T in \mathcal{T} such that $F(T) \leq F(W)$ for all $W \in \text{Dom}(A)$.

Through the use of the augmented set of boundary points this definition also covers the degenerate case, where there are no natural boundary points in the example set. According to this definition also *ACE* is well-behaved. In fact, the requirements of this definition are so weak that well-behaved evaluation functions is, clearly, the largest class of teleologically useful functions that can be defined.

Definition 3 constraints search of the optimal binary partition to boundary points. Moreover, if a well-behaved evaluation function is also cumulative, then it has the same desirable property with respect to multiway partitions. In order to minimize the goodness score of a partition, with respect to any well-behaved and cumulative evaluation function, it suffices to examine only boundary points.

Theorem 1 (Elomaa and Rousu [6]) For any cumulative and well-behaved evaluation function F there exists a partition of an arbitrary example set such that it minimizes the value of F and the corresponding cut points are boundary points. \square

Theorem 1 tells us that an evaluation function that is both cumulative and well-behaved in binary splitting also behaves well in multisplitting. In other words, sequences of examples consisting of instances of a single class are not broken apart by the multisplit. Using such an evaluation function allows efficient attribute selection, since we can focus on the boundary points in searching for the optimal partition. For cumulative functions the best k -partition of a subsample $S' \subset S$ does not depend on the splitting of its complement set $S \setminus S'$, thus the required search strategy can be implemented easily by using dynamic programming [9, 6]. The time requirement of finding the optimal partitioning into at most k intervals is $O(kB^2)$, where B is the number of blocks in the range. Hence, in practice the method is efficient enough as to recover the best partition from among all partitions (arities $2, \dots, B$).

4 Examples of well-behaved evaluation functions

According to Definition 3 an evaluation function must satisfy only very weak requirements for it to be well-behaved. It suffices that (one of) the lowest-impurity-partitions into two intervals falls on a boundary point. Clearly, convex evaluation functions fulfill this requirement. Thus, the well-behavedness of some important attribute evaluation functions is already known.

Breiman *et al.* [3, 2] have studied learning regression trees using two evaluation functions: the *gini index* of diversity and the *twoing criterion*. Gini index, or the quadratic entropy, is defined as

$$GI\left(\bigcup_{i=1}^k S_i\right) = \sum_{i=1}^k \frac{|S_i|}{|S|} gini(S_i),$$

where *gini* is the function

$$gini(S) = - \sum_{i=1}^m P(C_i, S)(1 - P(C_i, S)).$$

Gini criterion was shown to be convex by Breiman [2] and, therefore, it is also well-behaved. It is also cumulative, and therefore would suit multisplitting if its bias of favoring high-arity partitions was balanced somehow.

Twoing criterion is based on the idea of grouping the classes into two superclasses and then finding the optimal partition for this two class problem. As twoing criterion relies on another evaluation function to actually find the best partition its well-behavedness depends on the evaluation function that was chosen. Note that grouping of classes can only remove boundary points from the example sequence. Thus, using a well-behaved evaluation function in conjunction with the twoing criterion ensures well-behaved functioning.

Theorem 2 *The impurity measure gini index and twoing criterion are well-behaved.* □

Fayyad and Irani [7] defined the profound concept of a boundary point in examining how to speed up ID3's [14] numerical attribute handling. They proved that the evaluation function *ACE* is convex (downwards) in between any two boundary points. From that result it follows that also ID3's attribute evaluation function, information gain, is convex (upwards). Using our earlier notation it can be expressed as

$$IG(S, A) = H(S) - ACE\left(\bigcup_{i=1}^k S_i\right),$$

where $H(S)$ is the class entropy of the sample S prior to partitioning. The value of $H(S)$ is the same for all attributes and partitions. Hence, *IG* obtains its maximum value when *ACE* receives its minimum value. Thus, the well-behavedness of information gain is also clear.

Theorem 3 *The evaluation functions ACE and IG are well-behaved.* □

As to examine the extent and generality of well-behaved attribute evaluation functions, Elomaa and Rousu [6] took two non-standard functions and proved their well-behavedness: the incremental MDL exception coding scheme of Wallace and Patrick [19], denoted by *WP*, and the straightforward evaluation scheme of minimizing the training set error of a decision tree, *TSE*, which is applied, e.g., in the T2 algorithm [1]. Both of these are cumulative and were shown to be well-behaved as well. As gini index and information gain, also *TSE* suffers from the bias of favoring high-arity partitions. The MDL measure, however, is more balanced in this respect.

Moreover, Elomaa and Rousu [6] examined a variant of information gain function, *balanced gain*, which can be expressed as $BG(S, A) = IG(S, A)/k$, where k is the arity of the partition. The motivation behind this simple modification to information gain was to balance the bias of information gain so that partitions of high arity would not be favored, while keeping the function cumulative, thus ensuring its well-behavedness.

Theorem 4 *The evaluation functions WP, TSE, and BG are well-behaved.* □

Clearly, many of the commonly used, important evaluation functions are well-behaved. However, all the functions above are also convex. Are there any evaluation functions that are not convex but still fulfill the weak requirements of a well-behaved evaluation function?

As we can see from Theorem 1, for cumulative evaluation functions the definition of a well-behaved function lives up to its name: if the function behaves well on binary partitioning, then it will do the same on multisplitting. How about the ones that are not cumulative? Does the definition then discriminate between in practice well-behaving and poorly-doing evaluation functions? As a step towards answering these questions, in the following we examine the properties and behavior of gain ratio function [14].

5 Gain ratio does not behave well on multisplits

Information gain function does not place any penalty to the increase of the arity of a partition. Therefore, it favors excessively multi-valued nominal attributes and multisplitting numerical attribute value ranges. As a rectification to this deficiency Quinlan [14] suggested dividing the *IG* score of a partition by the term $\kappa = -\sum_{i=1}^k (|S_i|/|S|) \log_2(|S_i|/|S|)$. The resulting evaluation function is known as the *gain ratio*

$$GR(S, A) = IG(S, A)/\kappa.$$

The renowned decision tree learner C4.5 [16] incorporates gain ratio as the default choice for attribute evaluation. Hence, it is probably one of the most frequently used evaluation functions, considering the status of C4.5 among decision tree learners. Gain ratio has been observed to have some difficulties in particular in connection with numerical attributes. As to overcome those problems López de Mántaras [12] has proposed to use another, but closely related evaluation function, and Quinlan [17] has recently been compelled to change the evaluation of numerical attributes in C4.5. Still, analysis of this function has been surprisingly scarce (c.f. [5, 10, 12]). In the following we take few steps towards understanding where the pitfalls lie in using this function.

Firstly, let us review how gain ratio is actually used in C4.5. In the first phase, the gain ratio maximizing partition is computed for each attribute. Then, instead of picking the gain ratio maximizing partition among the candidates, Quinlan [16] places an additional constraint

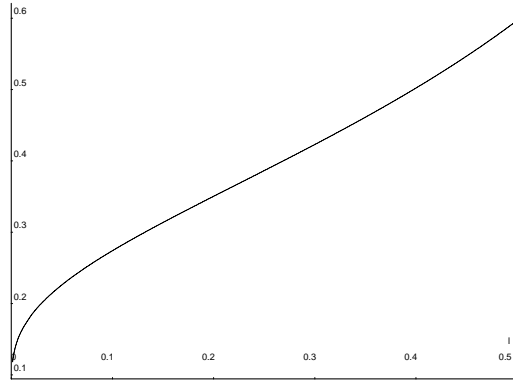


Figure 1: The gain ratio curve of the sample data.

to the partition that is determined as the best one: its IG score has to be at least as good as the average score among candidate partitions. This strategy tries to ensure that splits with low information gain are filtered out.

The guilty part for the risk of choosing low information gain splits is the denominator κ : it is the smaller the more uneven the partition is. The κ value can hinder a split of good information gain to be selected, if the partition induced by the split is too even. This is a bias that cannot easily be justified. In fact, one is tempted to think that the opposite strategy—that of favoring even partitions—would be just as good, if not better.

In the following we examine how the bias induced by the denominator κ manifests itself. Let us first demonstrate that gain ratio is not convex. Consider a sample which has the following class distribution: three fourths of the data belong to class A and one fourth to class B . Let one block contain all B instances and equally many members of class A . The rest of A instances constitute their own block (on either side of the mixed block), but may be in different bins. If gain ratio was convex, it should have a convex curve over the block consisting solely of members of class A .

Figure 1 plots the gain ratio curve of this data over the block consisting solely of instances of class A . Even from this picture it is easy to see that the curve is in first part convex and in the second part concave downwards. The existence of an inflection point was checked by inspecting the second derivative GR'' of the gain ratio function on the same situation. We could verify that there is an inflection point $x \approx 0.22308$ such that

$$GR''(y) = \begin{cases} < 0 & \text{when } y < x, \\ 0 & \text{when } y = x, \text{ and} \\ > 0 & \text{when } x < y. \end{cases}$$

Hence, this example shows that gain ratio function is not convex.

Theorem 5 *The evaluation function gain ratio is not convex.* □

Although convexity is a desirable property, the lack thereof does not imply that the function is not well-behaved (the opposite implication holds, however). In the following we show that gain ratio is, in fact, well-behaved with respect to binary splitting.

Theorem 6 *The evaluation function gain ratio is well-behaved.*

Proof It suffices to show that gain ratio of a binary split has no local maximas in an interval whose end points are two successive boundary points (c.f. [7, 6]).

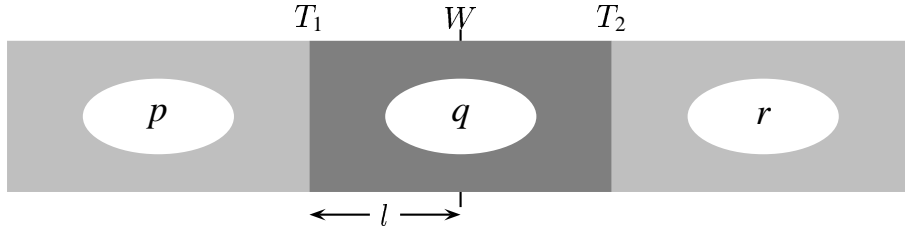


Figure 2: Illustration of the situation in the proof of Theorem 6.

Let two successive boundary points T_1 and T_2 define an interval of q examples (of a single class), p examples precede the interval and q examples follow it (see Fig. 2). No assumptions are made about the class distributions of the examples outside the interval. For each $0 \leq l \leq q$ we denote by $GR(l)$ the gain ratio that results when the cutpoint is situated after the l th example of the interval so that $q - l$ examples of the interval are left to the other subset. $IG(l)$ and $\kappa(l)$ are the information gain and κ values corresponding to the same situation. In other words $GR(l) = IG(l)/\kappa(l)$.

The first derivative¹ of $GR(l)$ is given by

$$GR'(l) = \frac{d}{dl}GR(l) = \frac{d}{dl} \frac{IG(l)}{\kappa(l)} = \frac{IG'(l)\kappa(l) - \kappa'(l)IG(l)}{\kappa^2(l)}.$$

Let us define $n(l) = IG'(l)\kappa(l) - \kappa'(l)IG(l)$, and note that

$$\begin{aligned} n'(l) &= IG''(l)\kappa(l) + \kappa'(l)IG'(l) - \kappa''(l)IG(l) - \kappa'(l)IG'(l) \\ &= IG''(l)\kappa(l) - \kappa''(l)IG(l) > 0, \end{aligned}$$

since for each $0 < l < q$ it holds that $IG''(l) > 0$ by the results of Fayyad and Irani [7], $\kappa(l) > 0$ by definition, $\kappa''(l) = -\frac{1}{(p+l)(r+q-l)} < 0$, and again by definition, $IG(l) \geq 0$.

Using the shorthand notation the second derivative of $GR(l)$ is expressed by

$$GR''(l) = \frac{d}{dl}GR'(l) = \frac{d}{dl} \frac{n(l)}{\kappa^2(l)} = \frac{n'(l)\kappa^2(l) - 2\kappa(l)\kappa'(l)n(l)}{\kappa^4(l)}.$$

Let now $\iota \in (0, q)$ be a potential location for a local maxima, i.e., such a point that $GR'(\iota) = 0$. From this assumption it follows that $n(\iota) = 0$. Hence the expression for $GR''(\iota)$ is further simplified into

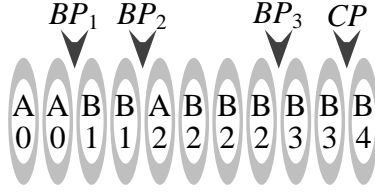
$$GR''(\iota) = \frac{n'(\iota)}{\kappa^2(\iota)} > 0,$$

because $n'(\iota) > 0$. In other words, $GR(l)$ is not a local maxima. Since ι was chosen arbitrarily, we have shown that $GR(l)$ obtains its maximum value at either of the boundary points T_1 and T_2 , where $l = 0$ and $l = q$, respectively. \square

The above analysis lets us conclude that the somewhat suspect bias induced by the denominator κ does not ruin the usefulness of the gain ratio function as far as binary splitting is considered. But what about multisplitting? As gain ratio is not cumulative, its well-behavedness on binary splitting does not necessarily imply the same on multisplitting.

Consider the set of eleven examples shown on the next page. For each example its class label (A or B) and the value of a numerical (integer in this case) attribute is shown. There are three boundary points in this data, BP_1, \dots, BP_3 , and one further possible cut point, CP . The gain ratio scores of the three-way partitions defined by these four cut point candidates are:

¹No harm is caused in taking $GR(l)$ to be continuous and twice differentiable in the open interval $l \in (0, q)$.



$$\begin{aligned}
 GR(\{BP_1, BP_2\}) &= 0.358 & GR(\{BP_1, CP\}) &= 0.411 & GR(\{BP_2, CP\}) &= 0.096 \\
 GR(\{BP_1, BP_3\}) &= 0.342 & GR(\{BP_2, BP_3\}) &= 0.119 & GR(\{BP_3, CP\}) &= 0.138
 \end{aligned}$$

In other words, the optimal three-way partition is not defined by two boundary points. Thus, the gain ratio function, which is not convex, but is well-behaved with respect to binary partitioning, cannot handle higher arity splitting well. We have given a counterexample for arity 3 and since, due to the increasing weight of κ , the relative frequency of “bad” splits increases by increasing arity. On the other hand, if the number of the classes is high, the information gain can outweigh κ and, therefore, the number of inappropriate splits remains lower. Nevertheless, even if “bad” partitions are not selected, the improper bias of κ can cause a partition of lower information gain to be selected over one with a higher gain value even if both partitions were cut at boundary points.

Moreover, the non-cumulativity of gain ratio means in practice that there is no efficient evaluation scheme for this function. In addition to gain ratio many other evaluation function proposals, e.g., [12], have been of the form $f(\cup_{i=1}^k S_i)/g(\cup_{i=1}^k S_i)$, which means that they too are non-cumulative and cannot be evaluated efficiently using the dynamic programming scheme.

Incidentally, the above counterexample also applies to the *normalized distance measure* d_N proposed by López de Mántaras [12] as an alternative to information gain and gain ratio functions. It too fails to select a three-way partition defined solely by boundary points. The measure can be expressed with the help of information gain as $1 - d_N(S, A) = IG(S, A)/\lambda$, where

$$\lambda = - \sum_{i=1}^m \sum_{j=1}^k \frac{N(C_i, S_j)}{|S|} \log_2 \frac{N(C_i, S_j)}{|S|},$$

in which m is the number of classes and k is the number of intervals, as usual, and $N(C, S)$ stands for the number of instances of class C in the set S . The intent is to minimize the value of distance $d_N \in [0, 1]$ or, equally, maximize the value of $1 - d_N$. The $1 - d_N$ values of the potential three-way partitions in the above example set are:

$$\begin{aligned}
 1 - d_N(\{BP_1, BP_2\}) &= 0.278 & 1 - d_N(\{BP_1, CP\}) &= 0.302 & 1 - d_N(\{BP_2, CP\}) &= 0.062 \\
 1 - d_N(\{BP_1, BP_3\}) &= 0.274 & 1 - d_N(\{BP_2, BP_3\}) &= 0.084 & 1 - d_N(\{BP_3, CP\}) &= 0.084
 \end{aligned}$$

Once again the denominator of this evaluation function not only manages to hinder efficient evaluability, but also loses the well-behavedness that comes along with the information gain function.

6 Empirical experiments

The following well-behaved variant of the information gain function is contrasted to gain ratio in these empirical experiments.

$$BG_{\log}(S, A) = IG(S, A) / \log_2 k.$$

Table 1: Average prediction accuracies for C4.5 using GR and IG functions and for the multisplitting strategies using GR and BG_{\log} . The symbols ++ (--) denote statistically significantly better (worse) performance w.r.t. BG_{\log} ; + and - denote almost significant differences.

DATA SET	C4.5 (GR)	C4.5 (IG)	GR (multi)	BG_{\log} (multi)
Annealing	90.5 \pm 0.5 ⁻	91.6 \pm 0.6	89.4 \pm 0.5 ⁻⁻	91.3 \pm 0.4
Australian	84.7 \pm 0.8	85.8 \pm 0.8	83.7 \pm 0.4 ⁻⁻	85.3 \pm 0.6
Auto insurance	76.8 \pm 1.9 ⁺	70.7 \pm 0.9 ⁻⁻	78.5 \pm 1.3 ⁺⁺	75.2 \pm 1.6
Breast W	94.7 \pm 0.5	94.5 \pm 0.3	94.0 \pm 0.4 ⁻	94.7 \pm 0.5
Colic	84.7 \pm 0.8	84.5 \pm 0.6	85.8 \pm 0.7 ⁺⁺	84.2 \pm 0.7
Diabetes	72.9 \pm 1.5 ⁻	74.9 \pm 1.0	73.2 \pm 1.0 ⁻	74.3 \pm 1.0
Euthyroid	98.6 \pm 0.1	98.8 \pm 0.1	98.6 \pm 0.1	98.6 \pm 0.1
German	71.6 \pm 0.9	73.3 \pm 0.9 ⁺	71.1 \pm 2.1 ⁻	72.3 \pm 0.7
Glass	68.6 \pm 2.3 ⁻⁻	69.8 \pm 1.4 ⁻⁻	68.9 \pm 1.0 ⁻⁻	72.3 \pm 1.4
Heart C	53.4 \pm 1.6	53.4 \pm 1.0	53.2 \pm 0.7	53.7 \pm 1.7
Heart H	81.8 \pm 1.3 ⁺	75.6 \pm 1.3 ⁻⁻	82.3 \pm 1.2 ⁺	79.7 \pm 1.5
Hepatitis	79.9 \pm 1.4 ⁻⁻	81.0 \pm 2.4	79.8 \pm 1.3 ⁻⁻	81.9 \pm 1.7
Hypothyroid	99.1 \pm 0.0	99.3 \pm 0.0	99.2 \pm 0.1	99.2 \pm 0.1
Iris	93.5 \pm 1.2	94.3 \pm 1.3	92.5 \pm 0.9	93.4 \pm 0.9
Mole	81.4 \pm 1.5 ⁻	81.1 \pm 1.1 ⁻⁻	81.8 \pm 1.2	82.5 \pm 0.9
Robot	99.3 \pm 0.1	99.6 \pm 0.1	99.6 \pm 0.1	99.5 \pm 0.1
Satellite	86.3 \pm 0.5	86.8 \pm 0.3 ⁺⁺	86.6 \pm 0.3 ⁺	86.0 \pm 0.3
Segmentation	86.6 \pm 1.1 ⁺	85.7 \pm 1.2	86.5 \pm 1.3 ⁺	85.4 \pm 1.1
Sonar	66.8 \pm 2.9 ⁻⁻	75.5 \pm 2.5	65.0 \pm 3.1 ⁻⁻	74.9 \pm 2.2
Vehicle	71.8 \pm 0.7	73.0 \pm 1.3 ⁺	71.4 \pm 1.2	72.0 \pm 0.6
Wine	92.5 \pm 1.1 ⁻⁻	93.9 \pm 0.9	94.2 \pm 1.0	94.4 \pm 1.0
AVERAGE	82.6	82.9	82.6	83.4

To see that it is closely related to the gain ratio function², observe that the denominator κ in the formula of GR is the entropy function H applied to the sizes of the intervals, not to the class distribution. Clearly, it holds that $0 < \kappa \leq \log_2 k$. Thus, BG_{\log} penalizes all equal arity partitions uniformly and always maximally as regards GR . An information theoretic interpretation exists for $\log_2 k$ also: it is the entropy of the intervals when assigned equal probability regardless of their size. Note also that BG_{\log} coincides with IG when binary splitting is considered.

As for relation between BG_{\log} and BG , note that the former is less heavily biased against the increase of arity of the multisplit. Our earlier experiments with the function BG [6] convinced us that the function favored binary splits excessively, since the number of multisplits in the induced trees remained very low. Indeed, results with BG_{\log} compare favorably against BG .

Our tests contrast BG_{\log} to GR in evaluating numerical attributes, both in binarization and multisplitting context. We run the same set of tests for four configurations, namely binarization tests with C4.5 using GR and IG (i.e. BG_{\log}) functions and multisplitting tests using GR and BG_{\log} .

We implemented a dynamic programming method for finding multisplits [6] into C4.5 as well as the evaluation function BG_{\log} . As no efficient optimization strategy is known for GR in multisplitting context, we used the following evaluation scheme: for each attribute and for each arity, we searched for the optimal information gain partition by dynamic programming. From a set of candidate splits generated this way the selection was done by gain ratio. This

²As a historical note, Quinlan [15] motivated gain ratio with experimental results reported by Kononenko, Bratko and Roskar; in their experiments BG_{\log} was used in evaluating multi-valued nominal attributes, but—for some reason—the results were not satisfactory.

scheme favors *GR*, since no “bad” splits are left to choose from. The differences in predictive accuracies are then explained only by the fact that gain ratio selects splits that are a little more uneven and have lower information gain than partitions selected by BG_{\log} .

We tested the functions on 21 real-world data sets that come mainly from the UCI repository [13]. The test strategy was 10-fold cross-validation repeated 10 times. For significance testing two-tailed Student’s *t*-test was used. Table 1 lists the average prediction accuracies and standard deviations obtained. In the table, symbols $^{++}$ and $^{--}$ denote statistically significantly better and worse performance with respect to BG_{\log} , respectively; $^{+}$ and $^{-}$ denote corresponding almost significant differences.

From Table 1 we see that in multisplitting context the statistically significant differences are more often in favor of BG_{\log} (5+3) than in favor of *GR* (3+2). Thus, it is reasonable to assume the difference of 0.8 percentage points in the average prediction accuracies over all domains to be significant as well. Even clearer edge is obtained by BG_{\log} over C4.5 using *GR*: it is statistically significantly better than C4.5 on 7 (4+3) data sets, while C4.5 only manages to obtain 3 (1+2) “wins.” As for the question whether multisplitting has anything to do with the good performance of BG_{\log} , observe that the multisplitting variant of BG_{\log} is statistically significantly better than its binarizing counterpart (C4.5 (IG)) on 4 domains. On the other hand, it loses statistically significantly only once and twice almost significantly—a result speaking in favor of multisplitting.

All in all, it appears that even an almost trivial penalization added to information gain function against favoring multisplitting excessively can significantly outperform needlessly complicated penalization attempt of gain ratio, which has led it to have a poorly balanced bias as demonstrated above and admitted by Quinlan [17]. Moreover, these experiments would seem to indicate that multisplitting (on boundary points) possesses advantages over the binarization approach. However, the difference is very small when using the poorly balanced *GR* function.

7 Conclusion and future work

Well-behavedness of an evaluation function is a necessary, but not a sufficient condition for teleologically good behavior as regards numerical attributes. For cumulative evaluation functions, such as information gain, well-behavedness on binary splitting extends to multisplitting situations. This is not true for evaluation functions in general; in the absence of general theory of discretization, the suitability of an evaluation function for multisplitting has to be examined per function basis.

In this paper we have shown that the evaluation function gain ratio behaves well in binary splitting, but does not do the same in multisplitting. This is an indication of the fact that multisplitting is more demanding task for an evaluation function than previously believed: in addition to the well-known problem of improper biases for (or against) the increase in the arity of the partition, also the well-behavedness is hard to ensure.

As for binary splitting, gain ratio is an example of a non-convex evaluation function that is well-behaved. Its performance in the binarization task has not been satisfactory, as its recent replacement from numerical attribute evaluation in C4.5 demonstrates [17]. As far as the authors know, all evaluation functions that have been shown to behave well in multisplitting are convex and cumulative, although, at least in principle, convexity is not required for good behavior. In order to shed more light on this matter, one should examine non-cumulative and/or non-convex evaluation functions as to find out whether there exists functions that are able to guarantee for all arities that their best partition is defined by boundary points, or at

least that the optimal partition over all arities is defined solely by boundary points.

References

- [1] P. Auer, R. Holte and W. Maass, Theory and application of agnostic PAC-learning with small decision trees. In: A. Prieditis and S. Russell (eds.), *Proc. Twelfth International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1995, 21–29.
- [2] L. Breiman, Some properties of splitting criteria. *Machine Learning* **24** (1996) 41–47.
- [3] L. Breiman, J. Friedman, R. Olshen and C. Stone, *Classification and Regression Trees*. Wadsworth, Pacific Grove, CA, 1984.
- [4] J. Catlett, On changing continuous attributes into ordered discrete attributes. In: Y. Kodratoff (ed.), *Proc. Fifth European Working Session on Learning*, Lecture Notes in Computer Science **482**. Springer-Verlag, Berlin, 1991, 164–178.
- [5] T. Dietterich, M. Kearns and Y. Mansour, Applying the weak learning framework to understand and improve C4.5. In: L. Saitta (ed.), *Proc. Thirteenth International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1996, 96–104.
- [6] T. Elomaa and J. Rousu, General and efficient multisplitting of numerical attributes. Report C-1996-82. Department of Computer Science, University of Helsinki, 1996.
- [7] U. Fayyad and K. Irani, On the handling of continuous-valued attributes in decision tree generation. *Machine Learning* **8** (1992) 87–102.
- [8] U. Fayyad and K. Irani, Multi-interval discretization of continuous-valued attributes for classification learning. In: *Proc. Thirteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA, 1993, 1022–1027.
- [9] T. Fulton, S. Kasif and S. Salzberg, Efficient algorithms for finding multi-way splits for decision trees. In: A. Prieditis and S. Russell (eds.), *Proc. Twelfth International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1995, 244–251.
- [10] M. Kearns and Y. Mansour, On the boosting ability of top-down decision tree learning algorithms. In: *Proc. Twenty-Eighth Annual ACM Symposium on Theory of Computing*. ACM Press, New York, NY, 1996, 459–468.
- [11] I. Kononenko, On biases in estimating multi-valued attributes. In: *Proc. Fourteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, San Francisco, CA, 1995, 1034–1040.
- [12] R. López de Mántaras, A distance-based attribute selection measure for decision tree induction. *Machine Learning* **6** (1991) 81–92.
- [13] C. Merz and P. Murphy, UCI repository of machine learning databases (<http://www.ics.uci.edu/~mllearn/MLRepository.html>). Department of Information and Computer Science, University of California at Irvine.
- [14] R. Quinlan, Induction of decision trees. *Machine Learning* **1** (1986) 81–106.
- [15] R. Quinlan, Decision trees and multi-valued attributes. In: J. Hayes, D. Michie and J. Richards (eds.), *Machine Intelligence 11: Logic and the Acquisition of Knowledge*. Oxford University Press, 1988, 305–318.
- [16] R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.
- [17] R. Quinlan, Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research* **4** (1996) 77–90.
- [18] T. Van de Merckt, Decision trees in numerical attribute spaces. In: *Proc. Thirteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA, 1993, 1016–1021.
- [19] C. Wallace and J. Patrick, Coding decision trees. *Machine Learning* **11** (1993) 7–22.