Efficient Multisplitting on Numerical Data

Tapio Elomaa[1] and Juho Rousu[2]

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

[2] VTT Biotechnology and Food Research
Tietotie 2, P. O. Box 1501, FIN-02044 VTT, Finland
juho.rousu@vtt.fi

Abstract. Numerical data poses a problem to symbolic learning methods, since numerical value ranges inherently need to be partitioned into intervals for representation and handling. An evaluation function is used to approximate the goodness of different partition candidates. Most existing methods for multisplitting on numerical attributes are based on heuristics, because of the apparent efficiency advantages. We characterize a class of well-behaved cumulative evaluation functions for which efficient discovery of the *optimal* multisplit is possible by dynamic programming. A single pass through the data suffices to evaluate multisplits of all arities. This class contains many important attribute evaluation functions familiar from symbolic machine learning research. Our empirical experiments convey that there is no significant differences in efficiency between the method that produces optimal partitions and those that are based on heuristics. Moreover, we demonstrate that optimal multisplitting can be beneficial in decision tree learning in contrast to using the much applied binarization of numerical attributes or heuristical multisplitting.

1 Introduction

When presenting symbolic information about numerical data, the underlying value range needs to be partitioned into two or more intervals. Numerical domain is either discrete (integer) or continuous (real) and typically very large, even infinite. Therefore, it has been the general misconception that formally founded multiway partitioning of numerical data is inherently inefficient. Hence, many existing machine learning and data mining systems simply use *ad hoc* heuristics in handling numerical value ranges. Efficient, well-founded methods for induction are particularly valuable in data mining where massive collections of data are processed.

In this paper we show that efficient, well-founded multisplitting of numerical data is feasible. We characterize the class of well-behaved evaluation functions for which efficient discovery for the *optimal* multisplit is possible by dynamic programming. The work reported in this paper builds upon the tradition of supervised machine learning, decision tree learning in particular. We consider the most basic setting where 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. The correlation of an instance's class and the values of its attributes is approximated by an *evaluation function*, or a *goodness criterion.*

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 learners by using *binarization* (Breiman et al. 1984, Quinlan 1993), in which the value range is split into only two intervals at a time. Further partitioning of the domain is generated by subsequently continuing the binarization of previously induced intervals. This approach easily produces decision trees that are unnecessarily large and complicated, thus, hard to interpret and understand.

An alternative approach uses *greedy multisplitting* (Catlett 1991, Fayyad and Irani 1993), where the value range is similarly partitioned by recursive binarization, but at once; the resulting multisplit is assigned as a single proposition to the evolving rule or decision tree.

Neither of the above-mentioned methods can guarantee the quality (as measured by the evaluation function) of the resulting partition. Elomaa and Rousu (1996) devised a general and efficient algorithm for finding *optimal multisplits.* The evaluation functions that can be handled by this general scheme need to be *well-behaved* and *cumulative.* An earlier attempt to develop a general algorithm for optimal multisplits was made by Fulton, Kasif, and Salzberg (1995), but their method fails to guarantee that the optimality of the resulting partition with respect to the evaluation function (Elomaa and Rousu 1996).

This paper recapitulates the basic definitions and results that are known about numerical value range discretization. We explore the extent of the class of well-behaved evaluation functions. Finally, we experiment empirically with different multisplitting strategies and evaluation functions. These experiments underline that the optimal multisplitting algorithm can handle numerical data without compromising on the efficiency of processing or the intelligibility and the quality of the resulting description. Thus, the method is particularly well suited for data mining applications.

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 a machine learning 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 concept description. 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 $\mathrm{val}_A(s)$ 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 $\mathrm{val}_A(s_i) < \mathrm{val}_A(s_j)$. When splitting a set S of examples on the basis of the value of an attribute A, then there is a set of thresholds $\{T_1, \ldots, T_{k-1}\} \subseteq \mathrm{Dom}(A)$ that defines a partition $\bigcup_{i=1}^k S_i$ for the sample in an obvious manner: $S_1 = \{s \in S \mid \mathrm{val}_A(s) \leq T_1\}$, $S_i = \{s \in S \mid T_{i-1} < \mathrm{val}_A(s) \leq T_i\}$, when $1 < i < k$, and $S_k = \{s \in S \mid T_{k-1} < \mathrm{val}_A(s)\}$.

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, we can, as well, consider a categorized version of the data: we can 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 the standard technique of numerical value range discretization, used e.g., in the C4.5 decision tree learner (Quinlan 1993). In practice, usually $V \ll n$, where V denotes the number of different values (bins) in an attribute's domain.

Fayyad and Irani's (1992) analysis of the binarization technique proved that substantial reductions in time consumption can be obtained for the *information gain* function (Quinlan 1986), since only *boundary points* need to be considered as potential cut points, because optimal splits always fall on boundary points. Let us recapitulate the exact definition of a boundary point.

Definition 1 (Fayyad and Irani 1992). 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 $\mathrm{val}_A(s_1) < T < \mathrm{val}_A(s_2)$; and there exists no other example $s \in S$ such that $\mathrm{val}_A(s_1) < \mathrm{val}_A(s) < \mathrm{val}_A(s_2)$.

In between any two values there are infinitely many real numbers that would all qualify as boundary points according to the above definition. For all that follows, it is immaterial which of them is chosen as long as one is fixed. A partition with the same intervals would result even if one or the other of the inequalities in the last equation of Definition 1 was allowed to be non-strict. For instance, C4.5 takes a threshold defining a partition to be the largest appearing value from the domain of the attribute in question such that it is less or equal to the boundary point value conforming to Definition 1.

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 (Fayyad and Irani 1992). Since boundary points are taken from among the potential cut points, it is clear that $B \leq V$ always holds.

As to avoid unnecessary complications in the exceptional example sets that have only one or do not have any boundary points, we take the low and high extremes of the value range always to be additional boundary points. We refer to this extended set of boundary points as the *augmented* set of boundary points. Thus all example sets have at least two boundary points with respect to every numerical attribute.

The goodness criteria that are used to evaluate candidate partitions are many (see e.g., Kononenko 1995). Fayyad and Irani (1992) focused on a particular impurity measure, the *average class entropy*. 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:

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

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 2. 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).

For most evaluation functions the aim is to find their least value. In the sequel we normally assume that evaluation is a task involving minimization (of the impurity). However, there are also goodness functions for which the maximum is to be sought for. The results in this paper have their natural counterparts that apply for maximization.

Fayyad and Irani (1992) proved that average class entropy is a teleologically well-behaved evaluation function in the sense that it will never favor an obviously bad cut, one that needlessly disperses examples of one class into different sides of a cut. They proved that when searching for the best binary split by choosing a single cut point, we can restrict our attention to boundary points. Their main result can be restated as follows:

Theorem 3 (Fayyad and Irani 1992). *If value T defines a partition $S_0 \cup S_1$ of S such that it minimizes the impurity $ACE(S_0 \cup S_1)$, then T is a boundary point.* □

There is one exception to this result, viz., in the degenerate case that the sample S contains only members of one class, then $ACE(S_0 \cup S_1) = 0$, independent of whether the value defining the partition is a boundary point or not. Recall that in this case the only boundary points are at the far ends of the value range. We can surmount this complication easily by focusing on such minimum impurity partitions that do not contain superfluous thresholds.

3 The well-behavedness of functions

Fayyad and Irani (1992) base the proof of Theorem 3 on the fact 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 is 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.

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.

Definition 4 (Elomaa and Rousu 1996). 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 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 4 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 5 (Elomaa and Rousu 1996). *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.* □

Theorem 5 tells us that we are can focus on the boundary points in searching for the optimal partition if using a well-behaved and cumulative evaluation function. Since the best k-partition of a subsample $S' \subset S$ does not depend on the splitting of its complement set $S \setminus S'$, the required search strategy can be implemented easily by using dynamic programming (Fulton, Kasif and Salzberg 1995). The impurities of cut point candidates can very conveniently be calculated from the impurities of shorter intervals and smaller-arity partitions (Elomaa and Rousu 1996). The computation entails obtaining impurities for lower arity partitions during the process. Hence, 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. Since in practice $B \leq V \ll n$, the method is efficient enough as to recover (in a single pass) the best partition from among all partitions (arities $2,\ldots,B$).

In summary, the well-behavedness of an evaluation function suffices to guarantee efficient optimal binarization of the numerical value range and, thus, also efficient greedy multisplitting of the data using, e.g., the method of Fayyad and Irani (1993). Moreover, Theorem 5 shows that we can efficiently find good-quality multisplits, if only the evaluation function at hand is also cumulative. Hence, a well-behaved and cumulative evaluation function combined with the evaluation scheme outlined above constitutes a good candidate data mining applications.

4 The class of well-behaved evaluation functions

According to Definition 4 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. (1984) and Breiman (1996) have studied learning regression trees using two evaluation functions: the *gini index* of diversity, or the quadratic entropy, and the *twoing criterion.* They proved that both these impurity measures are convex. Thus, they are well-behaved as well.

Theorem 6. *The impurity measures gini index and twoing criterion are well-behaved.* □

Fayyad and Irani (1992) defined the profound concept of a boundary point in examining how to speed up ID3's (Quinlan 1986) 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 $IG(S,A) = H(S) - ACE(\bigcup_{i=1}^{k} S_i)$, 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 7. *The evaluation functions average class entropy and information gain are well-behaved.* □

As to examine the extent and generality of well-behaved attribute evaluation functions, Elomaa and Rousu (1996) took two non-standard functions and proved their well-behavedness: The incremental MDL exception coding scheme of Wallace and Patrick (1993), denoted as *WP*, and the straightforward evaluation scheme of minimizing the training set error, *TSE*, of a decision tree, which is applied, e.g., in the T2 algorithm (Auer, Holte and Maass 1995). Both of these were shown to be well-behaved.

Moreover, in their empirical experiments Elomaa and Rousu (1996, 1997) have examined two variants 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 second variant is an enhanced, logarithmic version of balanced gain: $BG_{\log}(S, A) = IG(S, A)/\log_2 k$. Both of these are clearly well-behaved by the well-behavedness of IG. The logarithmic version of balanced gain never exhibits performance that is significantly worse than that of BG.

Theorem 8. *The evaluation functions WP, TSE, BG, and $BG_{\log}$ are well-behaved.* □

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 attempt to this deficiency Quinlan (1986) 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*: $GR(S, A) = IG(S, A)/\kappa$.

However, Quinlan (1986, 1993) places an additional constraint 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. Notice that gain ratio is not cumulative, since the coefficient $1/\kappa$ to its cumulative component IG cannot be evaluated incrementally as it refers to the partition $\bigcup_{i=1}^{k} S_i$ of the sample S. Therefore, gain ratio evaluation cannot be implemented in the efficient dynamic programming fashion and it really only suits binarization tasks.

To see that the logarithmic version of balanced gain is closely related to the gain ratio function, observe that the denominator κ in the formula of GR is the entropy function H applied to the intervals, not to the class distribution. Hence, its value is $0 < \kappa \leq \log_2 k$. Thus, $BG_{\log}$ penalizes all equal arity partitions uniformly and always maximally as regards GR.

Gain ratio has been observed to have some difficulties in particular in connection with numerical attributes. As to overcome those problems López de Mántaras (1991) has proposed to use another, but closely related evaluation function, and Quinlan (1996) has been compelled to change the evaluation of numerical attributes in C4.5.

Elomaa and Rousu (1997) demonstrated that gain ratio is not convex. But, what about the well-behavedness of gain ratio? It appears that gain ratio is well-behaved by Definition 4, which only requires good behavior as regards binary partition. The proof would involve some numerical manipulation, but it should not be too difficult. However, for present purposes it suffices to conjecture, based on empirical evidence, that the claim holds: when tested on several real-world data sets, we could not find a contradicting case.

Elomaa and Rousu (1997), further, present a set of examples such that the optimal, w.r.t. gain ratio, three-way partition is not defined by two boundary points. Thus, the gain ratio function, which is not convex, but appears to be well-behaved with respect to binary partitioning, cannot handle higher arity splitting well. Moreover, its non-cumulativity means in practice that there is no efficient (single-pass) evaluation scheme for this function.

The class of well-behaved evaluation functions manages to capture many important attribute evaluation functions, even such that are not convex. 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. Unfortunately, for the ones that are not cumulative the definition cannot discriminate between in practice well-behaving and poorly-doing evaluation functions as regards multisplitting.

5 Empirical experiments

We report on experiments carried out using standard machine learning test data (mainly from the UCI repository). These data sets cannot be described as massive (the largest one contains 4,435 examples), but they serve to indicate the relative performance of different splitting strategies and evaluation functions. We describe two sets of experiments: In the first one the focus is on the running times of different multisplitting strategies. The second set of experiments observes the practical effects of splitting strategies and evaluation functions on decision tree learning. The test strategy is 10-fold cross-validation repeated 10 times.

Judging by the number of greedy approaches to multisplitting it is a public sentiment that finding the optimum is inefficient. The results in Table 1 speak for the opposite: in practice, optimal splitting is very fast, provided that we use a cumulative, well-behaved evaluation function. The table lists average running times required by three partitioning strategies to produce a multisplit. This figure also includes preprocessing time, which is also given separately in its own column. The partitioning strategies are the multisplitting method proposed by Fulton, Kasif, and Salzberg (1995), greedy multisplitting as suggested by Fayyad and Irani (1993) and optimal multisplitting as presented by Elomaa and Rousu (1996). The evaluation function for all three strategies is the average class entropy.

From Table 1 we can observe that the FKS approach clearly is not a feasible data mining tool. It is orders of magnitude slower than the two other approaches. As for the remaining strategies, we can see that the Greedy approach is somewhat faster than finding optimal multisplits. However, the difference is minimal, and totally dominated by the preprocessing time. Moreover, what is not shown here, is the fact that the quality of partitions produced by the Greedy approach is inferior to optimal partitions (Elomaa and Rousu 1996). Since Greedy multisplitting on numerical data obviously is efficient enough even for massive collections of data, we can conclude from this experiment that it is feasible to go mining for optimal partitions and not be content with heuristically good multisplits on numerical data.

For the second set of tests we implemented $BG_{\log}$ into C4.5 as an alternative numerical attribute evaluation function. Nominal attributes are always handled

Table 1. Average running times of the multisplitting strategies (in seconds/split).

Data set	Preproc.	FKS	Greedy	Optimal
Annealing	0.31	27.22	0.32	0.33
Australian	0.23	14.78	0.25	0.89
Auto insurance	0.02	1.50	0.04	0.16
Breast W	0.24	15.05	0.24	0.24
Colic	0.04	1.67	0.05	0.11
Diabetes	0.28	18.66	0.30	0.87
Euthyroid	4.33	197.64	4.41	4.63
German	0.48	32.04	0.50	1.47
Glass	0.03	1.97	0.05	0.30
Heart C	0.05	3.67	0.06	0.13
Heart H	0.04	2.53	0.05	0.13
Hepatitis	0.01	0.57	0.02	0.04
Hypothyroid	7.39	233.72	7.41	8.28
Iris	0.01	0.73	0.02	0.02
Mole	0.09	3.06	0.10	0.25
Robot	2.12	205.58	2.16	2.16
Satellite	11.27	876.51	11.29	11.43
Segmentation	0.03	1.94	0.07	0.48
Sonar	0.02	1.28	0.05	0.31
Vehicle	0.34	26.80	0.36	0.69
Wine	0.02	10.47	0.03	0.12

by C4.5's default evaluation scheme. For significance testing two-tailed Student's t-test was used. Table 2 lists the average prediction accuracies and standard deviations obtained using the standard C4.5 binarization combined with the gain ratio evaluation function and two multisplitting version of C4.5 that use $BG_{\log}$ as the numerical attribute evaluation function – optimal and greedy multisplitting strategies are employed. In the table statistically significant difference in favor of evaluation scheme A over B in the average values is denoted as $B \ll A$ (or equivalently $A \gg B$) and, respectively, a single $<$ sign stands for an almost statistically significant difference.

From Table 2 we see that the statistically significant differences are more often in favor of the optimal multisplitting scheme (4+3) than in favor of C4.5 (1+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. Very similar results are obtained when greedy and optimal multisplitting strategies are contrasted with each other. This time 4+2 statistically significant differences are in favor of the optimal strategy and 1+1 in favor of the greedy strategy.

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

Table 2. Optimal multisplitting vs. standard C4.5 binarization and greedy multisplitting.

Data set	C4.5	Optimal	Greedy
Annealing	90.5 $\pm$0.5	$<$ 91.3 $\pm$0.4 $\gg$	88.6 $\pm$0.5
Australian	84.7 $\pm$0.8	85.3 $\pm$0.6	84.9 $\pm$0.6
Auto insurance	76.8 $\pm$1.9 $>$	75.2 $\pm$1.6	$<$ 76.4 $\pm$1.7
Breast W	94.7 $\pm$0.5	94.7 $\pm$0.5 $>$	94.0 $\pm$0.5
Colic	84.7 $\pm$0.8	84.2 $\pm$0.7 $\gg$	82.7 $\pm$0.9
Diabetes	72.9 $\pm$1.5	$<$ 74.3 $\pm$1.0	73.6 $\pm$1.1
Euthyroid	98.6 $\pm$0.1	98.6 $\pm$0.1	98.3 $\pm$0.2
German	71.6 $\pm$0.9	72.3 $\pm$0.7	72.0 $\pm$1.0
Glass	68.6 $\pm$2.3	$\ll$ 72.3 $\pm$1.4 $>$	71.0 $\pm$1.9
Heart C	53.4 $\pm$1.6	53.7 $\pm$1.7	53.8 $\pm$1.3
Heart H	81.8 $\pm$1.3 $\gg$	79.7 $\pm$1.5	79.0 $\pm$0.4
Hepatitis	79.9 $\pm$1.4	$\ll$ 81.9 $\pm$1.7 $\gg$	80.1 $\pm$1.6
Hypothyroid	99.1 $\pm$0.0	99.2 $\pm$0.1	99.2 $\pm$0.0
Iris	93.5 $\pm$1.2	93.4 $\pm$0.9	93.5 $\pm$1.1
Mole	81.4 $\pm$1.5	$<$ 82.5 $\pm$0.9	81.7 $\pm$1.1
Robot	99.3 $\pm$0.1	99.5 $\pm$0.1	99.6 $\pm$0.1
Satellite	86.3 $\pm$0.5	86.0 $\pm$0.3	$\ll$ 86.9 $\pm$0.3
Segmentation	86.6 $\pm$1.1 $>$	85.4 $\pm$1.1	86.2 $\pm$1.4
Sonar	66.8 $\pm$2.9	$\ll$ 74.9 $\pm$2.2 $\gg$	71.3 $\pm$1.8
Vehicle	71.8 $\pm$0.7	72.0 $\pm$0.6	71.4 $\pm$1.2
Wine	92.5 $\pm$1.1	$\ll$ 94.4 $\pm$1.0	93.9 $\pm$1.2
Average	82.6	83.4	82.8

led it to have a poorly balanced bias as demonstrated above and admitted by Quinlan (1996). Moreover, these experiments clearly indicate that optimal multisplitting possesses significant advantages over both the binarization approach and the greedy multisplitting strategy.

6 Conclusion

The class of well-behaved evaluation functions can capture a larger set of the most important attribute evaluation functions than convex functions can. Unfortunately, for non-cumulative functions, like gain ratio, the well-behavedness on binary partitioning does not necessarily carry over to higher arity partitioning.

Well-behavedness of an evaluation function is a necessary, but not a sufficient condition for teleologically good behavior as regards numerical attributes. In addition the function has to be balanced. For cumulative evaluation functions well-behavedness alone suffices to guarantee good behavior. Cumulativity also makes a function evaluable in a single pass through the data.

Our empirical experiments have given a clear indication that cumulative, well-behaved evaluation functions are a tool that suits the handling of even massive collections of data. They are as efficient to evaluate as one can hope for, and can enhance the quality of the resulting concept classifier as compared to those produced using *ad hoc* heuristics.

Acknowledgements The work of T. Elomaa is in part supported by European Commission's Marie Curie Fellowship. He is on leave from the Department of Computer Science, University of Helsinki, Finland.

References

Auer, P., Holte, R., Maass, W.: Theory and application of agnostic PAC-learning with small decision trees. In A. Prieditis, S. Russell (eds.), *Proc. Twelfth International Conference on Machine Learning* (21–29). Morgan Kaufmann, San Francisco, CA, 1995.

Breiman, L.: Some properties of splitting criteria. Mach. Learn. **24** (1996) 41–47.

Breiman, L., Friedman, J., Olshen, R., Stone, C.: *Classification and Regression Trees.* Wadsworth, Pacific Grove, CA, 1984.

Catlett, J.: On changing continuous attributes into ordered discrete attributes. In Y. Kodratoff (ed.), *Proc. Fifth European Working Session on Learning* (164–178), Lecture Notes in Computer Science **482**. Springer-Verlag, Berlin, 1991.

Elomaa, T., Rousu, J.: General and efficient multisplitting of numerical attributes. Report C-1996-82. Department of Computer Science, University of Helsinki. Oct. 1996, 25 pp.

Elomaa, T., Rousu, J.: On the well-behavedness of important attribute evaluation functions. NeuroCOLT Technical Report NC-TR-97-006. Department of Computer Science, Royal Holloway, University of London. Feb. 1997, 14 pp.

Fayyad, U., Irani, K.: On the handling of continuous-valued attributes in decision tree generation. Mach. Learn. **8** (1992) 87–102.

Fayyad, U., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. Thirteenth International Joint Conference on Artificial Intelligence* (1022–1027). Morgan Kaufmann, San Mateo, CA, 1993.

Fulton, T., Kasif, S., Salzberg, S.: Efficient algorithms for finding multi-way splits for decision trees. In A. Prieditis, S. Russell (eds.), *Proc. Twelfth International Conference on Machine Learning* (244–251). Morgan Kaufmann, San Francisco, CA, 1995.

Kononenko, I.: On biases in estimating multi-valued attributes. In *Proc. Fourteenth International Joint Conference on Artificial Intelligence* (1034–1040). Morgan Kaufmann, San Francisco, CA, 1995.

López de Mántaras, R.: A distance-based attribute selection measure for decision tree induction. Mach. Learn. **6** (1991) 81–92.

Quinlan, R.: Induction of decision trees. Mach. Learn. **1** (1986) 81–106.

Quinlan, R.: *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

Quinlan, R.: Improved use of continuous attributes in C4.5. J. Artif. Intell. Res. **4** (1996) 77–90.

Wallace, C., Patrick, J.: Coding decision trees. Mach. Learn. **11** (1993) 7–22.