Conference on Machine Learning, ICML 2001 (Sydney, Australia) pp. 131-138.

Fast Minimum Training Error Discretization

Tapio Elomaa ELOMAA@CS.HELSINKI.FI
Juho Rousu ROUSU@CS.HELSINKI.FI
Department of Computer Science, P. O. Box 26, FIN-00014 University of Helsinki, Finland

Abstract

The need to discretize a numerical range into class coherent intervals is a problem frequently encountered. Training Set Error (*TSE*) is one of the commonly used impurity functions in this task.

We show that in order to find *TSE*-optimal discretization one only needs to examine a small subset of all cut points, called *alternation points*. On the other hand, we prove that failing to check an alternation point may lead to a suboptimal discretization. Alternation points can be identified efficiently once the data has been ordered.

Our empirical evaluation demonstrates that the number of alternations in numerical ranges typically is much lower than the total number of cut points. In our experiments the discretization algorithm running on top of alternation points was significantly faster than the same algorithm on all cut points. Thus, the sublinear number of evaluated threshold candidates further reduces the practical time requirement of *TSE* optimization.

1. Introduction

Consider having a set of labeled data points (with duplicate values allowed) from some range in the real axis. In order to distill some general information on the relation between the data values and their labels, the numerical range is discretized into labeled intervals. An upper bound on the number of allowed intervals is usually given as an external parameter.

As goodness criterion — an evaluation function — to discriminate between the candidate discretizations we use the *Training Set Error* (subsequently *TSE* for short): choose the discretization in which the interval labeling differs least often from that of the data points. In this paper we examine efficient ways to obtain discretizations that are optimal with respect to *TSE*.

TSE is arguably the most commonly used attribute evaluation function in machine learning algorithms (Auer, Holte & Maass, 1995; Brodley, 1995; Lubinsky, 1995; Kearns et al., 1997). Using *TSE* in growing the hypothesis is prone to overfit it to the training examples. On the other hand, the principle of *empirical risk minimization* is based on the choosing the hypothesis that minimizes training error (Vapnik & Chervonenkis, 1971). Under certain conditions one can guarantee that the minimum-error hypothesis will also have a small generalization error.

An union of intervals of a numerical range has sometimes been used as hypotheses in theoretical frameworks (Maass, 1994; Kearns et al., 1997; Lozano, 2000), but more often the bounded-arity discretization problem is encountered as a subproblem in learning more complex hypotheses. For example, in learning decision trees one needs to partition the domains of numerical attributes into a modest number of intervals (Breiman et al., 1984; Quinlan, 1986). In classifier induction discretization of numerical domains is a potential time-consumption bottleneck, since in the general case the number of possible discretizations is exponential in the number of interval threshold candidates within the domain.

With respect to many commonly used evaluation functions numerical ranges can be optimally discretized in quadratic time in the number of interval threshold candidates using a dynamic programming algorithm (Fulton, Kasif & Salzberg, 1995; Zighed, Rakotomalala & Feschet, 1997; Elomaa & Rousu, 1999), but only *TSE* is known to optimize in linear time (Fulton et al., 1995; Auer, 1997; Birkendorf, 1997). For quadratic-time algorithms pruning cut point candidates in preprocessing has turned out to be a useful way to speed up the search (Fayyad & Irani, 1992; Elomaa & Rousu, 1999; 2000). In this paper we explore the possibilities and limitations of further speed-up of minimum *TSE* discretization via preprocessing the data.

The remainder of this paper is organized as follows. In the next section we introduce the minimum training error discretization problem more exactly and review subquadratic-time search algorithms that have been put forward for finding *TSE*-optimal discretizations. Pruning threshold candidates without losing the possibility to recover an optimal discretization is considered in Section 3. In Section 4 we prove that when using *TSE* it is possible to prune more thresholds than when using other common evaluation functions. In Section 5 we report empirical experiments on the pruned set of threshold candidates. The final section presents the concluding remarks of this paper.

2. Minimum *TSE* Discretization

Let us first define the discretization problem more formally. A sample $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ consists of n labeled real values (given in the increasing order of x). For each (x, y), $x \in \mathbb{R}$ and y is the label of x from the set of classes $C = \{c_1, \ldots, c_m\}$. A k-interval discretization of the sample is generated by picking $k-1$ *interval thresholds* or *cut points* $T_1 < T_2 < \cdots < T_{k-1}, T_j \in (x_{\min}, x_{\max})$, where $x_{\min} = \min\{x \mid (x, y) \in S\}$ and $x_{\max} = \max\{x \mid (x, y) \in S\}$. Moreover, empty intervals are not allowed. The set of $k-1$ thresholds defines a partition $\biguplus_{i=1}^{k} S_i$ of the set S as follows:

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

In this paper we use Training Set Error to determine the goodness of a partition. Let $\delta_j(S) = |\{(x, y) \in S \mid y \neq c_j\}|$ denote the *error*, or the *number of disagreements*, with respect to class c_j in the set S. That is, if all instances in S were predicted to belong to class c_j, we would make $\delta_j(S)$ errors on S. Furthermore, let $\delta(S) = \min_{c_j \in C} \delta_j(S)$ denote the minimum error on S. A class $c_j \in C$ is called a *majority class* of S, if predicting class c_j leads to minimum number of errors on S, that is, $\delta_j(S) = \delta(S)$. Note that more than one class can qualify as a majority class. The majority classes of a set S are denoted by $\mathrm{maj}_C(S) = \{c_j \in C \mid \delta_j(S) = \delta(S)\}$.

Given a k-interval partition $\biguplus_{i=1}^{k} S_i$ of S, where each interval is labeled by (one of) its majority class, its Training Set Error is given by

$$TSE\left(\biguplus_{i=1}^{k} S_i\right) = \sum_{i=1}^{k} \delta(S_i).$$

Intuitively, *TSE* is the number of training instances falsely classified in the partition when each interval is labeled by one of its majority classes.

The *global* minimum error discretization problem is to find a partition $\biguplus_{i=1}^{k} S_i$ of S that has the minimum *TSE* value over all partitions of S. The maximum number of intervals k may be given as a parameter. Then the problem is to find the *TSE*-optimal partition among those that have at most k intervals. This is called *bounded-arity* discretization.

In the following we review efficient (subquadratic) algorithms that have been proposed for minimum training error discretization of numerical ranges. All algorithms require an $O(n \log n)$ time sorting step prior to discretization. Note that it is not possible to discretize an *unordered* sample of $(x, y) \in \mathbb{R} \times C$ pairs faster than sorting; given such a hypothetical discretization algorithm one could sort arbitrary real numbers by assigning each data point a unique class label and asking the discretization algorithm to create *TSE*-optimal discretization. The output would have to define the sequence of bins and the sorted data could be recovered in linear time from it.

Maass (1994) was the first to devise a subquadratic algorithm for minimizing *TSE* in bounded arity discretization. In his algorithm a balanced binary tree, to leaves of which the data points are assigned, is constructed in $O(n \log n + nk^2)$ time. Optimal interval assignment and labeling are then found in $O(k^2)$ time using this data structure. Kearns et al. (1997) presented an algorithm requiring $O(n \log n)$ time. Their algorithm is based on the observation that given an optimal partition of arity k, in the two-class case, the optimal partition of arity $k-2$ either has the label of both its first and last interval flipped, or one of the other (internal) intervals j has its label flipped, in which case one interval is composed of out of the three intervals $j-1, j, j+1$ in the original k-interval partition.

Using dynamic programming to compose the partition from the best lower-arity partitions of subsets yields a linear-time algorithm in the two-class case. Such an algorithm was first presented by Fulton, Kasif, and Salzberg (1995). In processing the sorted data from left to right, we only have to decide, at suitable threshold points, whether the uncommitted data points are combined to the last interval of the already constructed discretization or do they start a new interval in the discretization. The decision, of course, is based on which of the alternatives yields a smaller training error. Using this approach and keeping track of the optimal discretization of the processed data for all arities $1, \ldots, k$ lets us, in the end, choose the optimal discretization

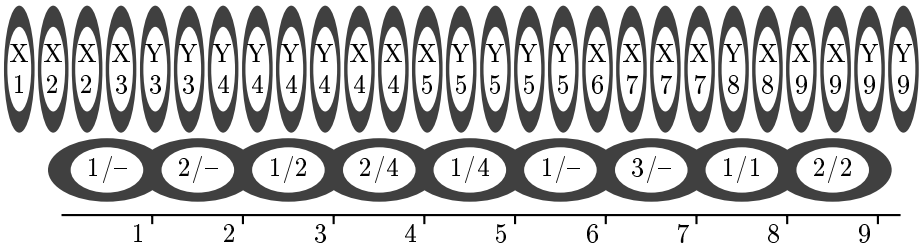

Figure 1. A sequence of data points sorted according to their numerical values (above). The class labels (X and Y) of the data points are also shown. The sequence of data bins with the respective class distributions (below). Interval thresholds can be set at the bin borders.

of the numerical range with at most k intervals. The time complexity of this approach clearly is $O(kn)$.

How to generalize the dynamic programming computation to the case when there are $m \geq 2$ classes was shown by Auer (1997). The sufficient modification is to maintain separately for each class c_j $(1 \leq j \leq m)$ at each threshold candidate the optimal discretizations of arity $1, \ldots, k$ such that the last interval is labeled with c_j. Taking multiple classes into account raises the time requirement of the algorithm to $O(kmn)$. Auer (1997) also showed how the space complexity of the algorithm can be kept low. Birkendorf (1997) has given a similar algorithm as a special case of a more general optimization problem. Below, we experiment with this Auer-Birkendorf algorithm.

The linear-time search algorithms reviewed above are asymptotically optimal in the number of examples. One cannot guarantee finding the optimal discretization without examining all of the cut point candidates, and they cannot be found without going through all of the data. For the dynamic programming scheme, a worst-case lower bound of $\Omega(kmn)$ can be derived from the general literature concerning dynamic programming optimization (Elomaa & Rousu, 2001).

However, in practice there are many potential thresholds that can be overlooked without risking to lose the optimal discretization. Hence, the sublinear number of threshold candidates gives a possibility to reduce the time requirement of the search. In the following we review approaches to optima-preserving pruning of cut point candidates.

3. Pruning Threshold Candidates in Preprocessing

As noted before, the processing of a numerical value range starts with sorting of the data points. If one could make its own partition interval out of each data point in the sorted sequence, this discretization would have zero training error. However, one cannot — nor wants to — discern between all data points. Only those that differ in their value can be separated from each other. Consider, for example, the data set shown in Figure 1. There are 27 integer-valued data points. They are instances of two classes; X and Y. Interval thresholds can only be set in between those points where the data point value changes. Therefore, we can preprocess the data into *bins*. There is one bin for each existing data point value. Within each bin we record the class distribution of the instances that belong to it. The class distribution information suffices to evaluate the goodness of the partition; the actual data set does not need to be maintained.

The sequence of bins has the minimal *attainable* misclassification rate. However, the same rate can usually be obtained with a smaller number of intervals. The analysis of the entropy function by Fayyad and Irani (1992) has shown that cut points embedded into class-uniform intervals need not be taken into account, only the end points of such intervals — the *boundary points* — need to be considered to find the optimal discretization. Elomaa and Rousu (1999) showed that the same is true for a large class of commonly used evaluation functions including also *TSE*. The analysis can be used in preprocessing in a straightforward manner: we merge together adjacent class uniform bins with the same class label to obtain example *blocks* (see Figure 2). The boundary points of the value range are the borders of its blocks. Block construction still leaves all bins with a mixed class distribution as their own blocks.

Subsequently, a more general property was also proved for *TSE* and some other evaluation functions (Elomaa & Rousu, 2000): *segment borders* — points that lie in between two adjacent bins with different relative class distributions — are the only points that need to be taken into account. It is easy to see that segment borders are a subset of boundary points. Example *segments* are easily obtained from bins by comparing the relative class distributions of adjacent bins (see Figure 2).

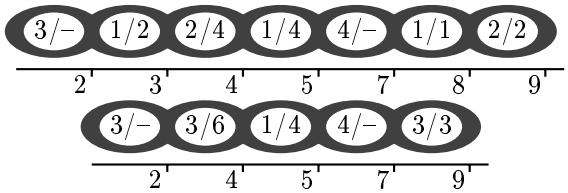

Figure 2. The blocks (above) and segments (below) in the sample of Figure 1. Block borders are the boundary points of the numerical range and segment borders are a subset of them.

4. Further Prepruning Opportunities for Training Set Error

An earlier proof shows that only segment borders need to be considered when trying to find *TSE*-optimal partitions (Elomaa & Rousu, 2000). In this section we prove that even some segment borders can safely be disregarded.

Let a *majority alternation point*[1] be the border in between two consecutive bins (or, as well, blocks or segments) that have different majority classes. More exactly, let $S_1 = \{ (x, y) \in S \mid x = v_1 \}$ and $S_2 = \{ (x, y) \in S \mid x = v_2 \}$, $v_1, v_2 \in \mathbb{R}$, be two adjacent bins. There is a majority alternation point in between S_1 and S_2, if and only if

$$\mathrm{maj}_C(S_1) \cap \mathrm{maj}_C(S_2) = \emptyset,$$

that is, the sets of majority classes in S_1 and S_2 are disjoint. For example in the dataset of Figure 2 there are only two majority alternations. Figure 3 shows the data organized in to sequences between majority alternations, which help to find *TSE*-optimal partitions.

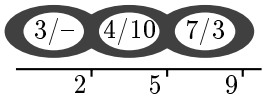

Figure 3. The majority alternations in the sample of Figure 1.

Theorem 1 *The partition defined by all majority alternation points in the sample S has the minimum TSE value and has minimal number of intervals.*

Proof Let us first show that all majority alternation points must be cut points in the minimal *TSE*-optimal partition. Assume that $\pi = \biguplus_{i=1}^{k} S_i$ is a partition of the sample S such that it does not contain all majority alternation points. Then in π there must exist an interval $S_i = \{ (x, y) \in S \mid T_{i-1} < x \leq T_i \}$ that contains the majority alternation points $a_1, \ldots, a_r, r \geq 1$, which satisfy $T_{i-1} < a_1 < \cdots < a_r < T_i$. Let $\biguplus_{h=1}^{r+1} Q_h$ denote the partition of S_i induced by the r alternation points.

From the definition of a majority alternation point it follows that for any class c_j there is a subset Q_{h_j} within $\biguplus_{h=1}^{r+1} Q_h$ for which $\delta_j(Q_{h_j}) > \delta(Q_{h_j})$, that is, c_j is not a majority class in Q_{h_j}. Thus, $\delta_j(S_i) > \sum_{i=1}^{r+1} \delta(Q_i)$, and since c_j is arbitrary also $\delta(S_i) > \sum_{h=1}^{r+1} \delta(Q_h)$.

Therefore, partitioning S_i into $r + 1$ subintervals at these majority alternations reduces the number of misclassifications in S_i and, thus, gives a better partition than π. Hence, π cannot be *TSE*-optimal. This shows that any *TSE*-optimal partition has a cut point in each majority alternation point.

Let us now assume that a *TSE*-optimal partition $\pi = \biguplus_{i=1}^{k} S_i$ has also other cut points than majority alternations. Let the cut points points in the partition be $T_1, \ldots, T_{k-1}$ and let us, further, define $T_0 = -\infty$ and $T_k = \infty$. Let T_i be one cut point in π that is not a majority alternation point. Now T_i separates two intervals in the partition, $S_i = \{ (x, y) \in S \mid T_{i-1} < x \leq T_i \}$ and $S_{i+1} = \{ (x, y) \in S \mid T_i < x \leq T_{i+1} \}$. Since T_i is not a majority alternation, there must exist a class c_j for which $\delta_j(S_i) = \delta(S_i)$ and $\delta_j(S_{i+1}) = \delta(S_{i+1})$. Thus, labeling both intervals S_i and S_{i+1} with the same label c_j results in a minimum error partition. Therefore, removing T_i will not affect the number of misclassifications. The same holds for all cut points that are not majority alternations.

Hence, we have shown that for globally minimal *TSE* value one needs to cut on each majority alternation point and to keep the number of intervals at minimum, no other thresholds except majority alternation points can be used. □

The above theorem shows that, when the number of intervals in the partition is not bounded, the best partition is the one that cuts on each majority alternation point. However, when an optimal partition of bounded-arity is required, examining majority alternation points alone does not suffice; one has to consider the frequency of the minority classes as well. In the following, we generalize the concept of a majority alternation point so that handling the bounded-arity case becomes possible.

Let P and Q be two adjacent bins with relative class frequency distributions $D_P = \{ p_1, \ldots, p_m \}$ and $D_Q = \{ q_1, \ldots, q_m \}$, respectively. There is an *alternation*

[1] Note that the term *alternation* has a different meaning here from, e.g., that in (Fulton et al., 1995; Kearns et al., 1997).

point in between P and Q if there is no index set $I = \{i_1, \ldots, i_m\}$ such that $p_{i_1} \geq p_{i_2} \geq \cdots \geq p_{i_m}$ and $q_{i_1} \geq q_{i_2} \geq \cdots \geq q_{i_m}$. In other words, an alternation occurs in between two bins, if ordering of the classes in descending order of frequency is different in them; that is, there exists a pair of classes h and j such that $p_h > p_j$ and $q_h < q_j$. Let us call the numerical range intervals induced by the set of alternation points as *alternating segments*. Clearly, a majority alternation is a special case of an alternation. In the two-class setting the two concepts coincide. Thus the number of alternations is always at least as high as that of majority alternations.

Next we show that only alternations need to be considered when searching for the *TSE*-optimizing partition.

Theorem 2 *For any numerical range and each $k \geq 2$ there is an TSE-optimal partition with at most k intervals that is defined on alternation points.*

Proof Let L, P, Q, and R form a sequence of adjacent subsets along the numerical range. Let the relative class distributions of sets L and R, D_L and D_R, be arbitrary and let $D_P = \{p_1, \ldots, p_m\}$ and $D_Q = \{q_1, \ldots, q_m\}$ be such that there is no alternation point in between the sets P and Q.

Let us now consider splitting the set into two intervals and labeling the left-hand side with class h and the right-hand side with class j. In such a situation, let $\delta_{h,j}(S \uplus T)$ denote the error of a binary partition with S as the left-hand side and T as the right-hand side. The errors of the partitions are

$$\delta_{h,j}(L \uplus (P \cup Q \cup R)) = \delta_h(L) + \sum_{S=P,Q,R} \delta_j(S)$$

$$\delta_{h,j}((L \cup P) \uplus (Q \cup R)) = \sum_{S=L,P} \delta_h(S) + \sum_{S=Q,R} \delta_j(S)$$

$$\delta_{h,j}((L \cup P \cup Q) \uplus R) \sum_{S=L,P,Q} \delta_h(S) + \delta_j(R).$$

By assumption, the point in between P and Q is not an alternation. Therefore, from the definition of an alternation it follows that for any pair of classes h and j, $1 \leq h, j \leq m$, either 1) $p_h \leq p_j$ and $q_h \leq q_j$ or 2) $p_h \geq p_j$ and $q_h \geq q_j$. In the first case $\delta_j(P) \leq \delta_h(P)$ and, consequently,

$$\delta_{h,j}(L \uplus (P \cup Q \cup R)) \leq \delta_{h,j}((L \cup P) \uplus (Q \cup R)).$$

In the second case $\delta_j(Q) \geq \delta_h(Q)$, and

$$\delta_{h,j}((L \cup P) \uplus (Q \cup R)) \geq \delta_{h,j}((L \cup P \cup Q) \uplus R).$$

Hence, the partition $(L \cup P) \uplus (Q \cup R)$ is always at most as good as the two other partitions. Since the classes h and j and the sets L and R are arbitrary, we have shown that in any partition a cut point that is not an alternation point, can be replaced with another cut point without increasing the error. Thus, one can slide cut points to the left or to the right until an alternation point or the end of interval is encountered. Hence, a *TSE*-optimal binary partition of an interval is defined on alternation points.

Since in the *TSE*-optimal k-interval discretization the embedded binary partitions are *TSE*-optimal, the claim follows. □

The consequence of the above theorem is that only those cut points that are alternations need to be considered when looking for the bounded arity optimal *TSE* partition. Hence the sample can be processed into alternating segments.

We show next that no alternations can be proven suboptimal without considering the context in which the cut point is; that is, which other cut points are present in the k-interval discretization of the range.

Theorem 3 *For each alternation point there is a context in which it is the TSE-optimal cut point.*

Proof Let L, P, Q, and R be adjacent subsets along a numerical range. Let there be a pair of classes h and j for which it holds that $p_h > p_j$ and $q_h < q_j$. That is, there is an alternation point in between P and Q. Now, let us choose the class distribution for L so that h is the majority class of $L \cup P \cup Q$. Such a distribution is easily generated by choosing L to be large enough and consist of instances of a single class h. Now, $\delta_j(P) > \delta_h(P)$ and, consequently,

$$\delta_{h,j}(L \uplus (P \cup Q \cup R)) > \delta_{h,j}((L \cup P) \uplus (Q \cup R)).$$

Similarly, the class distribution of R can be set so that j is the majority class of $P \cup Q \cup R$. Since $\delta_j(Q) < \delta_h(Q)$, we get

$$\delta_{h,j}((L \cup P) \uplus (Q \cup R)) < \delta_{h,j}((L \cup P \cup Q) \uplus R).$$

Thus, the optimal binary split with the left side labeled with class h and the right side labeled with class j has the cut point on the alternation point in between P and Q. Since h and j are majority classes of the left and the right sides of all binary splits of the subsets L, P, Q, and R, the split has the minimum *TSE*. Since h and j are arbitrary, the claim follows. □

The above theorem shows that the usefulness of an alternation point cannot be judged by examining the two

adjacent bins alone. This shows that while a fast left-to-right scan over the data — examining two bins at a time — is sufficient to find alternation points, discovering an equally simple and fast algorithm for pruning out some of the alternation points is unlikely.

5. Empirical Evaluation

In this section we examine alternation points with real-world data. We test first for 29 well-known data sets from the UCI data repository (Blake & Merz, 1998) what are the relations of average numbers of bin borders, boundary points, segment borders, alternation points, and majority alternations per numerical attribute. Then the practical speed-up gained by examining alternations rather than bin borders, boundary points, or segment borders is inspected.

5.1 On Finding Alternation Points Efficiently

Let us first make a note about the preprocessing algorithm used in these tests. All preprocessing algorithms first extracted bins from the sorted example sequence. Then, boundary points, segment borders, or alternation points were extracted. The preprocessing algorithms for boundary points and segment borders both run in time $O(n+mV)$, where V is the number of bins.

The trivial algorithm for alternation points is one which checks frequencies of up to $(m^2 - m)/2$ pairs of classes in two adjacent bins until a disagreement in the class ordering is found. The processing of the entire range takes $O(n+m^2V)$ time using this approach.

A faster algorithm for finding alternation points is obtained by sorting (in a left-to-right pass) the class distribution histograms of the bins using bucket sort in amortized $O(n)$ time and checking a bucket and the class frequency distribution of the adjacent interval to the right for a disagreement in class ordering. The bucket is merged to the interval if no alternation point is found. This $O(n + mV)$ time alternation point extraction meets the asymptotic bound for extracting bins from the data. However, the coefficients in this approach are quite large. The UCI data sets mostly have few classes, which prohibits time savings for the linear-time approach. Therefore, we have used the trivial alternation point detection in our experiments.

5.2 Empirical Results

Figure 4 depicts the results of the first experiment. Over all 29 test domains the average reduction in cut point candidates when moving from segment borders to alternation points is approximately 40% and close to 60% when compared to all cut points. The number of segment borders is only slightly smaller than that of boundary points. The number of alternations and majority alternations is the same on two-class domains (e.g., Adult, Australian, Breast Wisconsin, etc.), which is clear from their definition. For some two-class domains (e.g., Breast Wisconsin, Euthyroid, Heart H, and Hypothyroid) this number is significantly (ca. 75%) lower than that of segment borders. On other multiclass domains (e.g., Heart C, Letter recog-

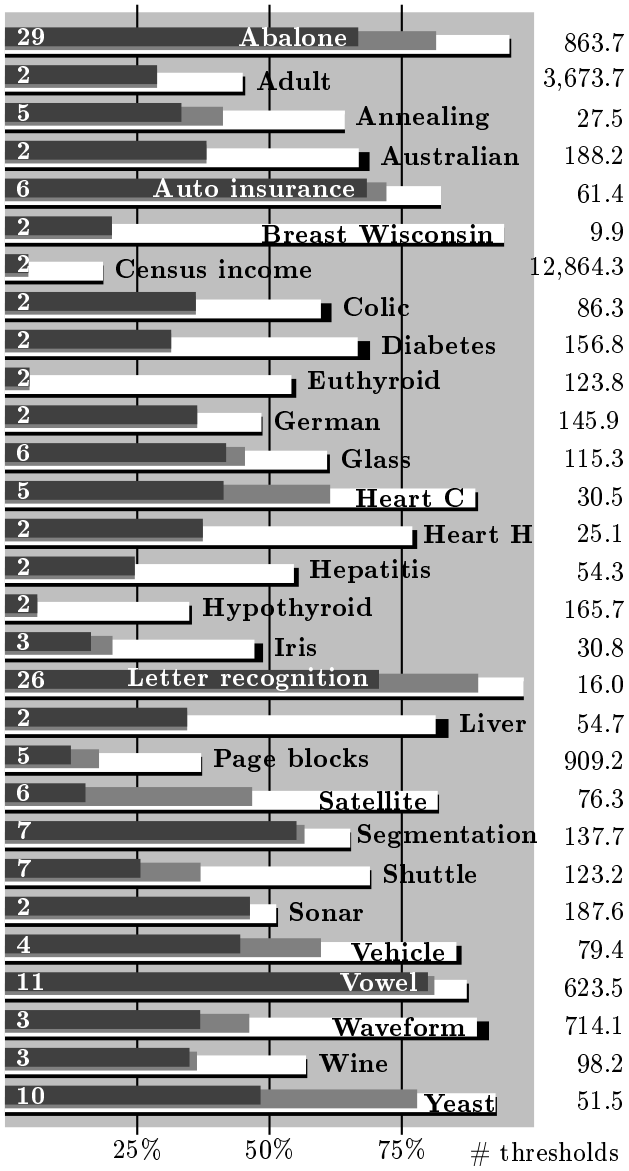

Figure 4. The average number of bin borders (the figures on the right) and the relative numbers of boundary points (black bars below), segment borders (white bars), alternation points (gray bars), and majority alternations (dark gray bars on top) per numerical attribute of the domain. The white figure on top of the dark gray bar is the number of classes.

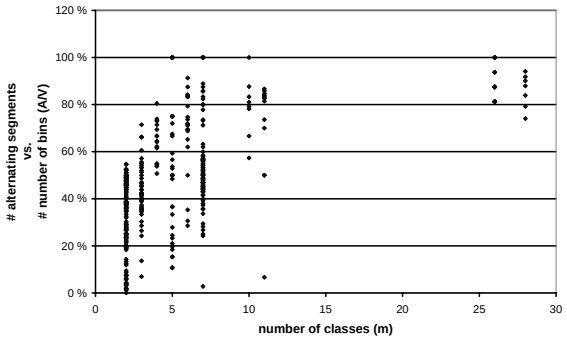

Figure 5. The effects of the number of classes on the efficiency gain obtained by using alternations rather than bin borders.

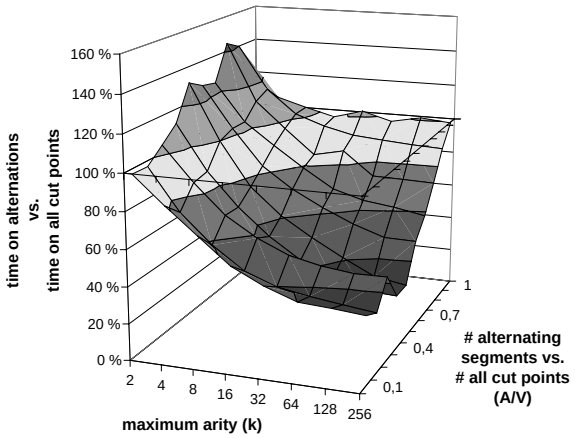

Figure 6. The time required in searching for *TSE*-optimal discretization when operating on alternations contrasted to that on all cut points. The effects of the allowed maximum arity and the relation A/V are also taken into account.

nition, Satellite, and Yeast) there is a significant difference in the numbers of alternations and majority alternations.

A striking result is that in many of those domains that have many classes (e.g., Abalone, Auto insurance, Letter recognition, Vowel, and Yeast) the number of alternations is not much smaller than that of segment borders. The number of majority alternations, though, is usually still somewhat smaller even for these domains. An explanation for this is that the more there are classes the less common it is that two adjacent segments have the same frequency order for all of them. The majority class may still be the same in two adjacent segments even though the number of classes is high.

Figure 5 plots the relation between the number of classes and the amount of reduction obtained in the number of cut point candidates by using alternation points. Each numerical attribute within our test domains is represented by one dot. Largest reductions are obtained in domains with few classes. In the two data sets with over 25 classes the reduction stays below 25%. The correlation between the ratio A/V, where A is the number of alternating segments, and the number of classes is quite clear. Hence, one may expect that in domains with many classes the time savings are not as great as in domains with fewer classes.

Now that we know that the number of alternation points is often even substantially lower than that of boundary points and segment borders, the question remains how much, if at all, can we benefit from using alternations rather than their alternatives.

Figure 6 plots the total running time (preprocessing and search) of the Auer-Birkendorf algorithm on alternation points against the same algorithm operating on bin borders. It can be clearly observed that as the relative number of alternations reduces or the number of intervals allowed in the discretization grows, the advantage gained by using alternation points increases significantly. The overhead associated with finding the alternation points dictates that if either of these parameters is not favorable, then discretization based on alternations is less efficient than that on easily-found bin borders. For $k \geq 8$ an A/V ratio below 0.9 promises gains, while in binary discretization no gains are obtained irrespective of the A/V ratio.

In the experiments we also observed that the discretizations defined on majority alternations were always optimal even for domains with more than two classes. This indicates that the condition, which sets a cut point on an alternation point that is not a majority alternation, is very rare in practice. So, if the optimal score is not required, one can safely choose the cut points from among majority alternations.

6. Conclusion

Examining segment borders — a subset of boundary points — is necessary and sufficient in searching for the optimal partition of a value range with respect to a strictly convex evaluation function (Elomaa & Rousu, 2002). We showed that with *TSE* — which is not strictly convex — only a well-defined subset of segment borders need to be examined: only majority alternations and alternation points need to be consid-

ered when searching for the global and bounded-arity optimum, respectively.

On the other hand, we were able to show that in bounded-arity discretization no alternations can be ignored with *TSE* without considering the placement of adjacent cut points. The Auer-Birkendorf algorithm, incidentally, does exactly this kind of bookkeeping: it keeps track of the best context to the left for each class and each arity. Hence, improving the dynamic programming scheme on the conceptual level seems difficult.

On some real-world domains the number of alternation points was discovered to be significantly smaller than that of segment borders. Practical gains were obtained by using alternation points rather than segment borders.

References

Auer, P. (1997). *Optimal splits of single attributes* (Unpublished manuscript). Institute for Theoretical Computer Science, Graz University of Technology.

Auer, P., Holte, R. C., & Maass, W. (1995). Theory and application of agnostic PAC-learning with small decision trees. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 21–29). San Francisco: Morgan Kaufmann.

Birkendorf, A. (1997). On fast and simple algorithms for finding maximal subarrays and applications in learning theory. *Computational Learning Theory, Proceedings of the Third European Conference* (pp. 198–209). Berlin, Heidelberg: Springer-Verlag.

Blake, C. L., & Merz, C. J. (1998). *UCI repository of machine learning databases.* Department of Information and Computer Science, University of California at Irvine.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees.* Pacific Grove: Wadsworth.

Brodley, C. (1995). Automatic selection of split criterion during tree growing based on node location. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 73–80). San Francisco: Morgan Kaufmann.

Elomaa, T., & Rousu, J. (1999). General and efficient multisplitting of numerical attributes. *Machine Learning*, *36*, 201–244.

Elomaa, T., & Rousu, J. (2000). Generalizing boundary points. *Proceedings of the Seventeenth National Conference on Artificial Intelligence* (pp. 570–576). Menlo Park: AAAI Press.

Elomaa, T., & Rousu, J. (2001). On the computational complexity of optimal multisplitting. *Fundamenta Informaticae*, *47*, 35–52.

Elomaa, T., & Rousu, J. (2002). Linear-time preprocessing in optimal numerical range partitioning. *Journal of Intelligent Information Systems*, *18*, 55–70.

Fayyad, U. M., & Irani, K. B. (1992). On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, *8*, 87–102.

Fulton, T., Kasif, S., & Salzberg, S. (1995). Efficient algorithms for finding multi-way splits for decision trees. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 244–251). San Francisco: Morgan Kaufmann.

Kearns, M., Mansour, Y., Ng, A. Y., & Ron, D. (1997). An experimental and theoretical comparison of model selection methods. *Machine Learning*, *27*, 1–50.

Lozano, F. (2000). Model selection using rademacher penalization. *Proceedings of the Second ICSC Symposium on Neural Networks.* Berlin: ICSC Academic.

Lubinsky, D. J. (1995). Increasing the performance and consistency of classification trees by using the accuracy criterion at the leaves. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 371–377). San Francisco: Morgan Kaufmann.

Maass, W. (1994). Efficient agnostic PAC-learning with simple hypotheses. *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory* (pp. 67–75). New York: ACM Press.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*, 81–106.

Vapnik, V., & Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, *16*, 264–280.

Zighed, D., Rakotomalala, R., & Feschet, F. (1997). Optimal multiple intervals discretization of continuous attributes for supervised learning. *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* (pp. 295–298). Menlo Park: AAAI Press.