Smoothed Discretization for Simplified Cutpoints

Georg Dietrich, Florian Lemmerich, and Frank Puppe
University of Würzburg
{dietrich, lemmerich, puppe}@informatik.uni-wuerzburg.de

Abstract

This paper describes work in progress. Discretization is one of the most common pre-processing steps in data mining and machine learning. We propose a novel approach to obtain simpler discretization cutpoints, which are easier to capture for human users, e.g., as they require less non-zero digits. For that purpose, a post-processing step is performed after applying an arbitrary conventional discretization method. It trades-off the necessary modifications in comparison to the original discretization scheme with the reduction in complexity of the cutpoints. Experiments with classification tasks show, that this leads to considerably simpler cutpoints with only marginal influence on the algorithmic performance, i.e., the prediction accuracy.

1 Introduction

The following paper reports preliminary results of ongoing research. Many machine learning and data mining algorithms, e.g., rule learners or decision-tree algorithms, can be applied automatically, but aim at models, which allow for introspection by the user. Other approaches, such as subgroup discovery, are not intended for automatic application at all, but provide patterns, which are directly interpreted by human experts. Both categories of algorithms require simple input data to build understandable models.

Discretization is a key pre-processing technique. It transforms numeric attributes into nominal ones in order to apply algorithms, which allow only for nominal attributes as inputs. Over the last decades a large number of sophisticated discretization methods have been proposed [7; 4]. Until now, research on these methods has focussed almost exclusively on the predictive power of the thresholds, but mostly ignored the resulting complexity of the discretization thresholds. This leads to discretization intervals that are inconvenient for humans, e.g., $income = [38952.4; 60427.2[$. Findings for such boundaries are not only unintuitive, but also potentially less useful in the application domain, as they are difficult to compare with previous knowledge. Additionally, such discretization bounds are potentially subject to *over-fitting* on the training data.

In this paper, we present a novel meta-method for discretization that aims at obtaining discretization thresholds, which are more intuitive for human users. For example, a very similar, but much simpler discretization interval for the above interval could be $income = [40000; 60000[$. Our approach obtains such simpler intervals by post-processing the resulting cutpoints of an arbitrary discretization method. In doing so, we combine the advantages of sophisticated discretization algorithms with intuitive discretization thresholds. The extent of the modification is traded-off against the complexity reduction of the results. Although our approach is applicable as a pre-processing method for arbitrary data mining tasks, the evaluation focuses in this work-in-progress on the classification tasks, since their results can be easily compared.

The rest of this paper is structured as follows: Section 2 introduces notations and discusses some related work. Next, Section 3 presents our novel approach of smoothed discretization bounds. First experimental results are provided in Section 4. The paper concludes with pointers to future work in Section 5.

2 Background and Related Work

In this paper, a dataset $\mathcal{D} = (\mathcal{I}, \mathcal{A})$ is formally defined as an ordered pair of a set of *instances* $\mathcal{I} = i_1, i_2, \ldots, i_y$ and a set of attributes $\mathcal{A} = A_1, A_2, \ldots, A_z, C$. Each attribute $A \in \mathcal{A} : I \rightarrow dom(A)$ is a function that indicates a characteristic of an instance by mapping it to a value in its range. Consequently, $A_m(i)$ denotes the value of the attribute A_m for the instance i. In our setting, there is one class attribute in each dataset A_C, which is to be predicted by a classification algorithm. We assume the class attribute to be *nominal* and all other attributes $A_1, \ldots, A_i$ to be *numeric*, i.e., $dom(A_i) = \mathbb{R}$.

Many data mining algorithms are not directly suited for numeric attributes, but require nominal attributes as input data. Therefore, *discretization algorithms* are used in a pre-processing step to transform a numeric attribute A into a new nominal attribute A'. These methods split the range of a numeric attribute into $n + 1$ disjunct intervals defined by a set of cutpoints $cp_1, \ldots, cp_n$: $\mathbb{R} =]-\infty; cp_1],]cp_1; cp_2], \ldots,]cp_{n-1}; cp_n],]cp_n; +\infty[$. The new attribute has one value for each of these intervals. Instance values are mapped accordingly:

$$A'(i) = \begin{cases} 0, & \text{if } A(i) \leq cp_1 \\ k & \text{if } cp_k < A(i) \leq cp_{k+1}, k = 1, \ldots n \\ n & \text{if } A(i) > cp_{k+1} \end{cases}$$

The cutpoints for different attributes are determined independent from each other by using a discretization method. For this task, a large amount of discretization methods have been proposed in literature, see [7; 4] for two recent overviews. The most popular methods include Equal-frequency discretization, top-down entropy-based discretization [3] and bottom-up discretization based on chi-values [6; 8]. Discretization methods, which lead to easy-to-read intervals, have received only little attention so far. An exception to this is the *intuitive partitioning* proposed by Han and Kamber [5] that discretizes an attribute into "natural" segments: In a top-down approach, the range of the attribute A is split into three, four, or five subintervals depending on the difference in the most significant digit in the attribute range. In contrast to this technique, our novel method joins the power of supervised discretization

algorithms with the goal of easy-to-read cutpoints. It can be combined with arbitrary discretization methods.

3 Smoothed Discretization

In the next section, we present our novel approach for discretization. The main idea is as following: First, any traditional discretization algorithm is run. The resulting set of cutpoints $cp_1, \ldots, cp_n$ is used as the input for our technique. A new discretization scheme is obtained by modifying the cutpoints cp_i one-by-one. For each cutpoint, an alternative new cutpoint is determined. The selection of the new cutpoints follows two criteria: 1. The replacement cutpoint should be "natural", i.e., less complex and easier-to-read. 2. The replacement cutpoint should be as close to the original cutpoint as possible. In the following sections, we present novel measures to quantify these criteria as well as a simple scoring function, which allows to trade-off between them. Furthermore, we outline a simple algorithm that allows to identify the best alternative cutpoint. It generates a number of candidate cutpoints, which are scored by the presented measure.

3.1 Complexity

The perceived complexity of a number differs from user to user. Due to this inherent subjectiveness quantifying its complexity is a difficult issue. One can consider several different intuitions to measure the complexity of a number, which are plausible for most users: First, short numbers are easier to comprehend than longer numbers: As a consequence the number 624 should receive a lower complexity score than a number like 7245. Second, one benchmark could be, how difficult it is to remember a number. Therefore, 1.000 would have a lower complexity score than 8103. A potential third intuition is, that numbers should receive a lower complexity score, if they are used more often by humans.

Next, we present one simple method to capture the complexity of a number. We are fully aware that this is definitely not the only solution for this problem and one can think of several variations of this measure. Our measure is based on the decimal representation of a number x. The scoring is based on the number of digits $k(x)$, which are required to write x, excluding trailing zeros. The count $k(x)$ is increased by one, if it contains a decimal point. Then, the complexity for x is defined as:

$$complexity(x) = \begin{cases} 0, & \text{if } x = 0 \\ 1 & \text{if } x = 10^n, n \in \mathbb{N} \\ 1 + k(x) & \text{else} \end{cases}$$

The following table shows some examples for this complexity measure.

x	$complexity(x)$	x	$complexity(x)$
0	0	400	2
1	1	725	4
100	1	-725	4
4	2	7.25	5

This basic measure could be improved in a variety of directions: One may argue that a number ending with the digit 5 is simpler than other numbers. E.g., 95 can be considered as a simpler, more intuitive bound than 93. Another issue is, if a decimal really increases the complexity, i.e., if 0.4 is a more complex number than 4. Although these considerations could be incorporated in more sophisticated variations in future approaches, we focus in this paper on the complexity measure presented above for the sake of simplicity and transparency.

3.2 Modification measures

Additionally, our approach requires a measure that compares, how strongly the original discretization scheme is modified, if a candidate cutpoint cp'_k is used instead of the respective original cutpoint cp_k. To quantify this amount of modification we propose two measures.

Distance-based deviation measures

The distance-based measure describes the difference in the range of the discretized attribute. It is computed as the percentage of the interval between the original cutpoint cp_k and the candidate cutpoint cp'_k in relation to the distance between the current cutpoint cp_k and the neighboring cutpoint in the original discretization scheme. For candidates smaller than the original cutpoint, the neighboring cutpoint is given by the next lowest cutpoint cp_{k-1}, otherwise it is the next highest cutpoint cp_{k+1}. For the special cases, that the current cutpoint is the first one ($k = 1$) or the last one ($k = n$), the instances in the dataset with the lowest, respectively highest, attribute values are used as neighboring cutpoints. Formally it is computed as (ignoring the special cases):

$$mod^{dist}(cp'_k, cp_k) = \begin{cases} 0 & \text{if } cp'_k = cp_k \\ \frac{cp_k - cp'_k}{cp_k - cp_{k-1}} & \text{if } cp'_k < cp_k \\ \frac{cp'_k - cp_k}{cp_{k+1} - cp_k} & \text{if } cp'_n > cp_k \end{cases}$$

Instance-based deviation measures

The distance-based deviation measure just considers the difference between the original cutpoint and the candidate cutpoint, independent of additional information contained in the dataset. The second approach, the instance-based deviation, additionally takes the values of each instance i for the attribute A, which is discretized, into account. It measures the percentage of the instances in the interval, which are relocated to another interval, if the original cutpoint cp_k is exchanged with the candidate cp'_k:

$$mod^{inst}(cp'_k, cp_k) = \begin{cases} 0 & \text{if } cp'_k = cp_k \\ \frac{|\{i | cp'_k \leq A(i) < cp_k\}|}{|\{i | cp_{k-1} \leq A(i) < cp_k\}|} & \text{if } cp'_k < cp_k \\ \frac{|\{i | cp_k \leq A(i) < cp'_k\}|}{|\{i | cp_k \leq A(i) < cp_{k+1}\}|} & \text{if } cp'_k > cp_k \end{cases}$$

3.3 Smoothed cutpoint selection

Cutpoint smoothing is a trade-off between reducing the complexity of a cutpoint and modifying the intervals generated by the original discretization method. For that purpose, we propose the following family of functions that balances between these two goals using the complexity and modification measures presented above. The candidate with the lowest score according to this measures is considered the best cutpoint.

$$score(cp'_k) = complexity(cp'_k) + \frac{1}{\alpha} \cdot mod(cp'_k, cp_k)$$

Here, α is a user chosen parameter. For high values of α less complex cutpoints are preferred, even if they strongly modify the original solution. In contrast, lower values of α emphasize the similarity to the original discretization scheme, even if the resulting cutpoints are only slightly less complex than the original ones. α can be interpreted, which ratio of an interval the cutpoint can be moved to reduce the complexity by one point. E.g., if $\alpha = 0.05$ the algorithm will shift the cutpoint by up to 5% of the adjacent interval (based on the pure difference or the number of contained instances), if this decreases the complexity by one.

3.4 Computation of smoothed cutpoints

The computation of the best smoothed cutpoint is straight forward: First, candidates are generated in two directions. For that purpose the cutpoint is iteratively rounded up with decreasing precision. This is repeated, until either zero or

the middle of the adjacent interval is reached. This prevents, that two different original cutpoints are smoothed to identical values. Candidate cutpoints smaller than the original cutpoint are obtained analogously by rounding down. Additionally, the original cutpoint is also considered as a candidate. Then, every candidate is evaluated by the score-function with user-chosen parametrization. The best (lowest) scoring candidate then replaces the original cutpoint in the smoothed discretization scheme.

3.5 Example

We demonstrate our approach in a small example: Initially the user chooses a parameter α for the scoring function and one of the two proposed modification measures. We assume an α value of 0.01 and the distance-based deviation measure in this example. To discretize an attribute A with our approach, first a traditional discretization algorithm, e.g., frequency-based discretization, is executed. We assume, this method resulted in the 3 cutpoints $cp_1 = -724$, $cp_2 = 692$, and $cp_3 = 1525$. For each of these cutpoints, our approach determines a smoothed cutpoint cp_i^*, which should be easier-to-read. In this example we focus on the cutpoint cp_2. For this cutpoint, first candidates for alternative cutpoints are determined by rounding up and down. This results in the candidates 700 and 1000 for rounding up and 690, 600, and 0 for rounding down. Additionally, the original cutpoint 692 is considered as a candidate. For each of these six cutpoint candidates the score is determined as described in Section 3.3. For example, the score for the candidate 700 is determined as follows: The complexity of the candidate is $complexity(700) = 2$. Its distance is computed as $\frac{700-692}{1525-692} \approx 0.0096$. The score for this candidate is $score(700) = 2 + \frac{1}{0.01} \cdot 0.0096 = 2.96$. Analogously, the score for the cutpoint 1000, which has a complexity of 1 is determined as $1 + \frac{1}{0.01} \cdot \frac{1000-692}{1525-692} \approx 37.97$. As another example, the original cutpoint has a complexity of 4 and a distance of 0 and thus a score of $score(692) = 4$. As it turns out after computing the scores for all six cutpoint candidates, 700 has the lowest (best) score and thus is used as a replacement for the original cutpoint in the novel discretization scheme.

4 Evaluation

To evaluate the effectiveness of our novel approach, we performed an experimental study on a classification task, using the well-known decision tree algorithm *C4.5* [9]. We used 12 data sets from the UCI Machine Learning Database Repository [2] and from the KEEL data set repository [1]. Except for the class attributes, these data sets consists of numerical attributes only.

We applied a standard 10-fold cross-validation procedure. For each training data set the discretization cutpoints were determined for the three popular discretization methods *equal-frequency discretization*, *entropy-based discretization*, and *Chi2 discretization*. Afterwards, the introduced smoothing techniques were performed with distance-based and instance-based modification measures and with different settings for the parameter α in the scoring function. A very low α value (here $\alpha = 10^{-7}$) in combination with an instance-based distance measure means, that the cutpoint is replaced with the alternative cutpoint with the lowest complexity, which implies no reallocation of any instance to another discretization interval. For the basic discretization and for the classification algorithm implementations from the KEEL software suite were used with default parametrization. In particular, the equal-frequency discretization performed a split into 10 intervals. For the resulting discretized data a classifier was learned on the training data and the accuracy was measured in the test data. Summarized results, which are averaged over all datasets, are shown in Tables 1a, 1b and 1c. Exemplary detailed result for all datasets using entropy-based discretization are denoted in Tables 2a and 2b. These tables show the predictive accuracy of the classifier with the discretized attributes as input as well as the averaged complexity score of the smoothed discretized cutpoints. All base discretizers lead to a high complexity of the used cutpoints: the cutpoints are overall hard-to-read for humans. For all discretizers, smoothing these cutpoints even with only low values of α leads to a drastic decrease of the complexity. The complexity of cutpoints is further reduced for increased parameter values of α. These adaptations influence the accuracy of the classifiers only marginally, i.e., the improved classification accuracy of entropy-based discretization is maintained even for substantially simplified cutpoints. This may hint at possible overfitting of the discretization algorithms. Only for the highest settings of α ($\alpha \geq 0.1$) a slight decrease of the accuracy can be observed for the supervised discretization algorithms. The reduction of the accuracy is smaller for instance-based smoothing methods, while similar complexity reductions are achieved. Therefore, this variation is to be preferred based on the current results. These experiments overall demonstrate the effectiveness of our novel approach, as it succeeds in decreasing the complexity of the used cutpoints with only marginal influence on the main algorithm, which uses the discretization intervals.

	distance		instance	
α	comp	acc	comp	acc
0.0	9.293	0.812		
10^{-7}			4.766	0.811
0.01	4.668	0.810	4.350	0.807
0.05	3.810	0.812	3.697	0.813
0.1	3.240	0.811	3.337	0.817
0.3	2.672	0.800	2.665	0.816
0.5	2.509	0.799	2.619	0.813

(a) Entropy-based discretization

	distance		instance	
α	comp	acc	comp	acc
0.0	9.102	0.781		
10^{-7}			4.879	0.780
0.01	4.818	0.784	4.520	0.780
0.05	4.002	0.784	4.017	0.785
0.1	3.488	0.774	3.645	0.788
0.3	2.942	0.768	2.887	0.780
0.5	2.746	0.771	2.841	0.779

(b) Chi2 discretization

	distance		instance	
α	comp	acc	comp	acc
0.0	9.042	0.784		
10^{-7}			4.981	0.784
0.01	5.121	0.784	4.730	0.785
0.05	4.301	0.780	4.266	0.782
0.1	3.912	0.782	3.931	0.780
0.3	3.277	0.780	3.372	0.774
0.5	2.992	0.785	3.298	0.780

(c) Equal-frequency discretization

Table 1: Results for different values for the parameter α in the scoring function and both modification measures. Each table refers to a different discretization technique. For each setting, the prediction accuracy of the classification algorithm and the cutpoint complexity are denoted averaged over all data sets.

α	0.0		0.01		0.05		0.1		0.3		0.5	
	comp	acc	comp	acc	comp	acc	comp	acc	comp	acc	comp	acc
appendicitis	8.729	0.834	5.184	0.852	4.936	0.852	4.436	0.832	3.607	0.868	3.547	0.878
banana	9.331	0.748	4.423	0.752	3.335	0.745	3.207	0.745	2.296	0.745	2.296	0.745
glass	11.973	0.758	4.658	0.735	3.789	0.731	3.173	0.747	2.879	0.700	2.844	0.700
movement	10.978	0.606	5.356	0.567	4.847	0.589	4.534	0.589	4.035	0.586	3.425	0.569
pageblocks	6.359	0.968	4.205	0.966	3.483	0.965	3.071	0.965	2.452	0.963	2.298	0.965
phoneme	9.030	0.812	4.909	0.814	3.771	0.817	3.203	0.813	2.539	0.806	2.353	0.804
segment	11.255	0.939	4.454	0.937	3.424	0.944	3.116	0.938	2.679	0.936	2.556	0.941
sonar	7.794	0.764	5.407	0.759	4.882	0.744	4.533	0.759	3.926	0.700	3.753	0.701
spambase	8.121	0.927	4.202	0.924	2.806	0.920	2.410	0.918	2.139	0.918	2.044	0.920
titanic	11.333	0.771	4.333	0.771	4.000	0.771	1.667	0.771	1.667	0.771	1.667	0.771
vowel	8.814	0.719	4.601	0.736	3.047	0.720	2.600	0.731	1.549	0.687	1.305	0.669
wine	7.801	0.898	4.286	0.904	3.401	0.944	2.935	0.922	2.302	0.915	2.014	0.921

(a) Distance-based modification

α	10^{-7}		0.01		0.05		0.1		0.3		0.5	
	comp	acc	comp	acc	comp	acc	comp	acc	comp	acc	comp	acc
appendicitis	5.250	0.834	5.250	0.834	4.749	0.842	4.447	0.860	3.933	0.859	3.933	0.859
banana	5.186	0.749	4.215	0.751	3.363	0.747	2.662	0.746	2.315	0.746	2.216	0.741
glass	4.814	0.753	4.755	0.753	4.147	0.765	3.840	0.754	2.775	0.756	2.618	0.721
movement	5.572	0.603	5.384	0.564	4.896	0.589	4.606	0.569	4.047	0.614	4.003	0.631
pageblocks	4.243	0.968	3.891	0.968	3.550	0.966	3.259	0.966	2.821	0.965	2.750	0.965
phoneme	5.773	0.813	4.955	0.815	4.041	0.823	3.611	0.829	2.769	0.825	2.723	0.825
segment	5.152	0.939	4.194	0.938	3.372	0.944	3.086	0.942	2.605	0.936	2.558	0.936
sonar	5.718	0.759	5.396	0.735	4.943	0.750	4.633	0.783	4.285	0.778	4.267	0.768
spambase	4.851	0.927	4.450	0.929	3.930	0.928	3.549	0.925	1.238	0.896	1.165	0.890
titanic	0.667	0.771	0.667	0.771	0.667	0.771	0.667	0.771	0.667	0.771	0.667	0.771
vowel	5.461	0.720	4.778	0.725	3.091	0.718	2.549	0.738	2.028	0.718	2.028	0.718
wine	4.507	0.898	4.267	0.898	3.614	0.915	3.137	0.921	2.494	0.933	2.494	0.933

(b) Instance-based modification

Table 2: Detailed results for different values of the parameter α in the scoring function. Each table refers to a different modification measure. For each setting and each data set, the average prediction accuracy of the classification algorithm and the average cutpoint complexity are denoted. As basis entropy-based discretization was used.

5 Conclusions

In this paper we proposed a novel approach on discretization, which aims at cutpoints, which are easy-to-read for human users, e.g., as they require less non-zero digits. For that purpose, a post-processing step is performed after applying an arbitrary conventional discretization method. It trades-off the necessary modifications in comparison to the original discretization scheme with the reduction in complexity. In that direction novel functions for measuring the complexity of a number and for measuring the difference between the original cutpoints and candidates for alternatives have been discussed. Experiments with classification tasks showed, that our approach leads to considerably simpler cutpoints, while the algorithmic performance, i.e., the prediction accuracy, is only marginally influenced.

Since this paper presents work in progress, we plan to extend it in several areas: The proposed function to measure the complexity of a number is currently very simple and could be replaced by a more sophisticated one. This, however, will require an extensive user study to evaluate human perception. Furthermore, we will extend the performed experiments to descriptive data mining tasks such as subgroup discovery. Since the results of these tasks are directly interpreted by domain experts, natural, easy-to-read intervals are especially useful in these areas.

References

[1] Alcala-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., Garcia, S.: Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Multiple-Valued Logic and Soft Computing 17(2-3), 255–287 (2011)

[2] Blake, C., Merz, C.J.: {UCI} Repository of machine learning databases (1998)

[3] Fayyad, U.M., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning (1993)

[4] García, S., Luengo, J., Saez, J., Lopez, V., Herrera, F.: A survey of discretization techniques: taxonomy and empirical analysis in supervised learning. IEEE Transactions on Knowledge and Data Engineering 25(4), 734–750 (2013)

[5] Han, J., Kamber, M., Pei, J.: Data mining: concepts and techniques. Morgan kaufmann (2006)

[6] Kerber, R.: Chimerge: Discretization of numeric attributes. In: Proceedings of the tenth national conference on Artificial intelligence. pp. 123–128. AAAI Press (1992)

[7] Kotsiantis, S., Kanellopoulos, D.: Discretization techniques: A recent survey. GESTS International Transactions on Computer Science and Engineering 32(1), 47–58 (2006)

[8] Liu, H., Setiono, R.: Chi2: feature selection and discretization of numeric attributes. Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence pp. 388–391 (1995)

[9] Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)