A Simple Approach to Ordinal Classification

Eibe Frank and Mark Hall

Department of Computer Science
University of Waikato
Hamilton, New Zealand
{eibe, mhall}@cs.waikato.ac.nz

Abstract. Machine learning methods for classification problems commonly assume that the class values are unordered. However, in many practical applications the class values do exhibit a natural order—for example, when learning how to grade. The standard approach to ordinal classification converts the class value into a numeric quantity and applies a regression learner to the transformed data, translating the output back into a discrete class value in a post-processing step. A disadvantage of this method is that it can only be applied in conjunction with a regression scheme.
In this paper we present a simple method that enables standard classification algorithms to make use of ordering information in class attributes. By applying it in conjunction with a decision tree learner we show that it outperforms the naive approach, which treats the class values as an unordered set. Compared to special-purpose algorithms for ordinal classification our method has the advantage that it can be applied without any modification to the underlying learning scheme.

1 Introduction

Classification algorithms map a set of attribute values to a categorical target value, represented by a class attribute. Practical applications of machine learning frequently involve situations exhibiting an order among the different categories represented by the class attribute. However, standard classification algorithms cannot make use of this ordering information: they treat the class attribute as a nominal quantity—a set of unordered values.

Statisticians differentiate between four basic quantities that can be represented in an attribute, often referred to as levels of measurement [9]. There are four types of measurements: nominal, ordinal, interval, and ratio quantities. The difference between nominal and ordinal quantities is that the latter exhibit an order among the different values that they can assume. An ordinal attribute could, for example, represent a coarse classification of the outside temperature represented by the values Hot, $Mild$, and $Cool$. It is clear that there is an order among those values and that we can write $Hot > Mild > Cool$.

Interval quantities are similar to ordinal quantities in that they exhibit an order. They differ because their values are measured in fixed and equal units. This implies that the difference between two values can be determined by subtracting

L. De Raedt and P. Flach (Eds.): ECML 2001, LNAI 2167, pp. 145–156, 2001.

them. This does not make sense if the quantity is ordinal. Temperature measured in degrees Fahrenheit is an interval quantity. Ratio quantities additionally exhibit a zero point. This means that it is possible to multiply their values.

Standard classification algorithms for nominal classes can be applied to ordinal prediction problems by discarding the ordering information in the class attribute. However, some information is lost when this is done, information that can potentially improve the predictive performance of a classifier.

This paper presents a simple method that enables standard classification algorithms to exploit the ordering information in ordinal prediction problems. Empirical results—obtained using the decision tree learner C4.5 [7]—show that it indeed improves classification accuracy on unseen data. A key feature of our method is that it does not require any modification of the underlying learning algorithm—it is applicable as long as the classifier produces class probability estimates.

The method is explicitly designed for ordinal problems—in other words, for classification tasks with ordered categories. Standard regression techniques for numeric prediction problems can be applied when the target value represents an interval or ratio quantity. However, their application to truly ordinal problems is necessarily *ad hoc*.

This paper is structured as follows. In Section 2 we present our approach to ordinal classification. Section 3 contains experimental results on a collection of benchmark datasets, demonstrating that the predictive accuracy of decision trees can be improved by applying this method to exploit ordering information in the class. Section 4 discusses related work on custom-made learning algorithms for ordinal problems and approaches that use regression techniques for ordinal classification. Section 5 summarizes the contributions made in this paper.

2 Transforming the Ordinal Classification Problem

Figure 1 shows in diagrammatic form how our method allows a standard classification learner to be applied to an ordinal prediction task. The data is from a fictional temperature prediction problem and has an ordinal class attribute with three values ($Cool$, $Mild$ and Hot). The upper part depicts the training process and the lower part the testing process.

A simple trick allows the underlying learning algorithms to take advantage of ordered class values. First, the data is transformed from a k-class ordinal problem to $k-1$ binary class problems. Figure 2 shows the process of converting an ordinal attribute A^* with ordered values $V_1, V_2, ..., V_k$ into $k-1$ binary attributes, one for each of the original attribute's first $k-1$ values. The ith binary attribute represents the test $A^* > V_i$.

Training starts by deriving new datasets from the original dataset, one for each of the $k-1$ new binary class attributes. In Figure 1 there are two derived datasets, the first has a class attribute that represents $Target > Cool$ and the second has a class attribute that represents $Target > Mild$. Each derived dataset contains the same number of attributes as the original, with the same attribute values for each instance—apart from the class attribute. In the next

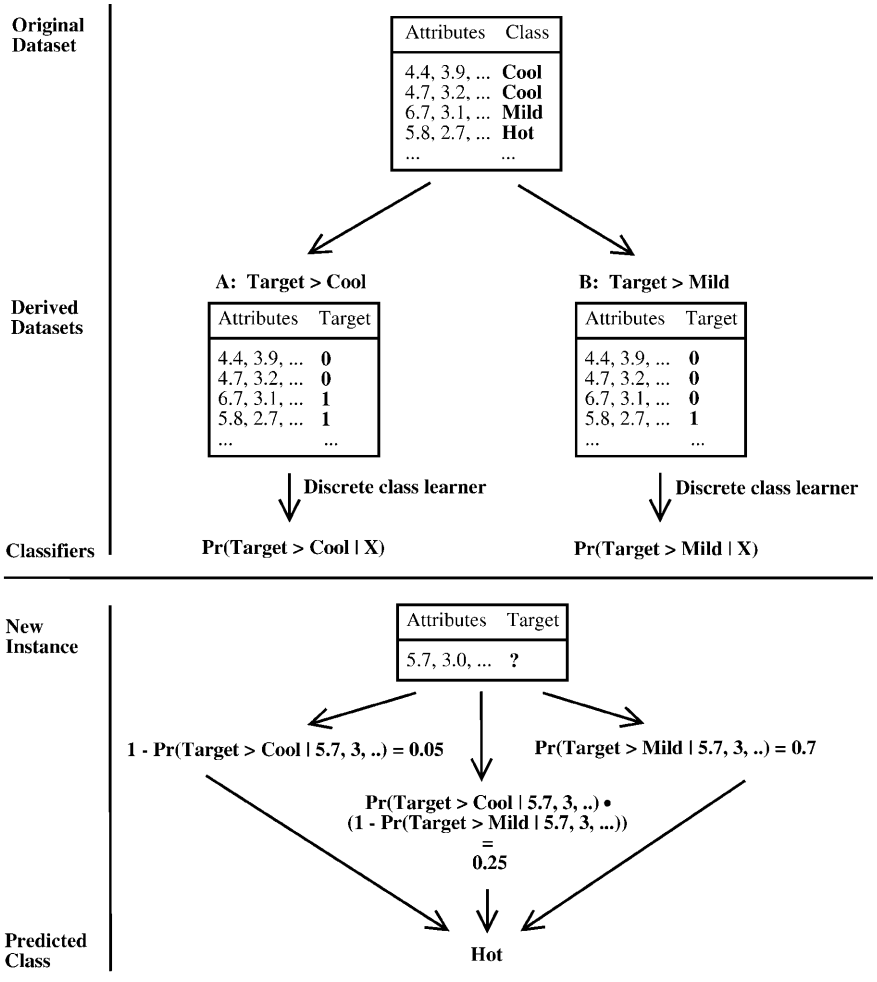

Fig. 1. How standard classification algorithms are applied to ordinal prediction

step the classification algorithm is applied to generate a model for each of the new datasets.

To predict the class value of an unseen instance we need to estimate the probabilities of the k original ordinal classes using our $k-1$ models. Estimation of the probability for the first and last ordinal class value depends on a single classifier. The probability of the first ordinal value ($Cool$) is given by $1-Pr(Target > Cool)$. Similarly, the last ordinal value (Hot) is computed from $Pr(Target > Mild)$. For class values in the middle of the range—in this case there is only one ($Mild$)—the probability depends on a pair of classifiers. In this example it is given by $Pr(Target > Cool) \times (1 - Pr(Target > Mild))$. In general, for class values V_i,

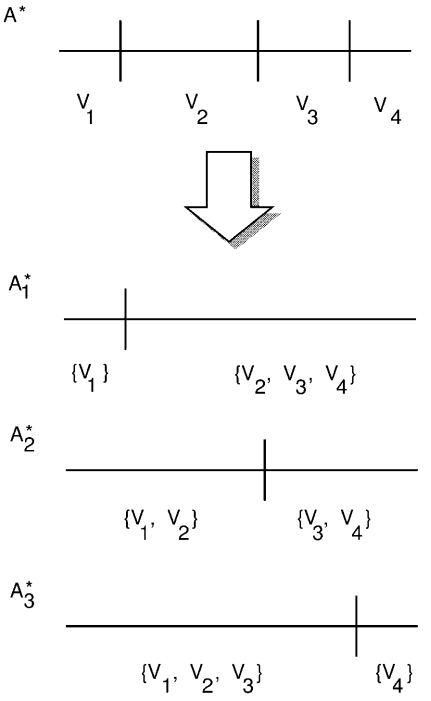

Fig. 2. Transformation of an ordinal attribute with four values into three binary attributes

$$
\begin{aligned}
Pr(V_1) &= 1 - Pr(Target > V_1) \\
Pr(V_i) &= Pr(Target > V_{i-1}) \times (1 - Pr(Target > V_i)) \quad , 1 < i < k \\
Pr(V_k) &= Pr(Targe > V_{k-1})
\end{aligned}
$$

At prediction time, an instance of unknown class is processed by each of the $k-1$ classifiers and the probability of each of the k ordinal class values is calculated using the above method. The class with maximum probability is assigned to the instance.

3 Experimental Results

To test the hypothesis that the above method improves the generalization performance of a standard classification algorithm on ordinal prediction problems, we performed experiments on artificial and real-world datasets in conjunction with the C4.5 decision tree learner [7]. We used a collection of benchmark datasets representing numeric prediction problems and converted the numeric target values into ordinal quantities using equal-frequency binning. This unsupervised discretization method divides the range of observed values into a given number of intervals so that the number of instances in each interval is approximately

Table 1. Datasets and their characteristics

Dataset	Instances	Attributes	Numeric	Nominal
Abalone	4177	9	7	2
Ailerons	13750	41	40	1
Delta Ailerons	7129	6	5	1
Elevators	16599	19	18	1
Delta Elevators	9517	7	6	1
2D Planes	40768	11	10	1
Pole Telecomm	15000	49	48	1
Friedman Artificial	40768	11	10	1
MV Artificial	40768	11	7	4
Kinematics of Robot Arm	8192	9	8	1
Computer Activity (1)	8192	13	12	1
Computer Activity (2)	8192	22	21	1
Census (1)	22784	9	8	1
Census (2)	22784	17	16	1
Auto MPG	398	8	4	4
Auto Price	159	16	15	1
Boston Housing	506	14	12	2
Diabetes	43	3	2	1
Pyrimidines	74	28	27	1
Triazines	186	61	60	1
Machine CPU	209	7	6	1
Servo	167	5	0	5
Wisconsin Breast Cancer	194	33	32	1
Pumadyn Domain (1)	8192	9	8	1
Pumadyn Domain (2)	8192	33	32	1
Bank Domain (1)	8192	9	8	1
Bank Domain (2)	8192	33	32	1
California Housing	20640	9	8	1
Stocks Domain	950	10	9	1

constant. The resulting class values are ordered, representing variable-size intervals of the original numeric quantity. This method was chosen because of the lack of benchmark datasets involving ordinal class values.

The datasets were taken from a publicly available collection of regression problems [8]. The properties of these 29 datasets are shown in Table 1. For each dataset we created three different versions by discretizing the target value into three, five, and ten intervals respectively. This was done to evaluate the influence of different numbers of classes on the schemes' relative performance.

All accuracy estimates were obtained by averaging the results from 10 separate runs of stratified 10-fold cross-validation. In other words, each scheme was applied 100 times to generate an estimate for a particular dataset. Throughout, we speak of two results for a dataset as being "significantly different" if the difference is statistically significant at the 1% level according to a paired two-sided t-test, each pair of data points consisting of the estimates obtained in one ten-fold cross-validation run for the two learning schemes being compared. A

significant difference in this sense means that, with high probability, a complete cross-validation [4] on the dataset would result in a difference in accuracy.[1]

Table 2 shows the accuracy estimates for the decision tree learner C4.5[2] in the five-class situation, applied (a) without any modification of the input and output (C4.5), (b) in conjunction with the ordinal classification method presented in Section 2 (C45-ORD), and (c) using the standard one-per-class method (see, e.g., [9]) for applying a two-class learner to a multi-class problem (C4.5-1PC). We have included C4.5-1PC in the comparison to ascertain whether overall differences in performance are due to the fact that we transform the multi-class problem into several two-class problems. Standard deviations are also shown (based on the 10 accuracy estimates obtained from the 10 cross-validation runs).

Results for C4.5 and C4.5-1PC are marked with $\circ$ if they show significant improvement over the corresponding results for C4.5-ORD, and with $\bullet$ if they show significant degradation. Table 3 shows how the different methods compare with each other. Each entry indicates the number of datasets for which the method associated with its column is (significantly) more accurate than the method associated with its row.

The results show that the ordinal classification method frequently improves performance compared to plain C4.5. On 17 datasets, C4.5-ORD is significantly more accurate than C4.5, and significantly worse on only one. The results also show that the performance difference is not due to the fact that each learning problem has been converted into several two-class problems: C45-ORD is significantly more accurate than C4.5-1PC on 16 datasets, and significantly worse on only five.

A sign test confirms the hypothesis that the ordinal classification procedure from Section 2 improves performance. Note that the results for the two different versions of the computer activity, census domain, pumadyn domain, and bank domain datasets are highly correlated. Consequently we ignore the smaller version of each of these four datasets when we perform the sign test (i.e. we only consider version number 2 in each case). If this is done, C4.5-ORD wins against plain C4.5 on 19 datasets and looses on only four (not taking the significance of the individual differences into account). According to a two-sided sign test this ratio is significant at the 0.005%-level. The win/loss-ratio between C4.5-ORD and C4.5-1PC is 18/6 and significant at the 0.05%-level.

An interesting question is whether the difference in accuracy depends on the number of class values in the problem. A reasonable hypothesis is that the performance difference increases as the number of classes increases. To investigate this we also ran the three schemes on the same datasets with different discretizations of the target value.

Tables 4 and 5 summarize the results for the three-bin discretization. They show that there is relatively little to be gained by exploiting the ordering information in the class. Compared to C4.5, C4.5-ORD is significantly more accurate on 15 datasets, and significantly worse on none. However, C4.5-ORD does not

[1] A complete cross-validation is performed by averaging across all possible cross-validation runs that can be performed for a particular dataset.

[2] We used the implementation of C4.5 that is part of the WEKA machine learning workbench.

Table 2. Experimental results for target value discretized into five bins: percentage of correct classifications, and standard deviation

Dataset	C4.5-ORD	C4.5	C4.5-1PC
Abalone	47.86±0.45	46.34±0.73 •	49.55±0.65 ○
Ailerons	59.24±0.30	56.97±0.35 •	55.58±0.34 •
Delta Ailerons	55.76±0.34	55.54±0.50	56.77±0.15 ○
Elevators	50.32±0.24	47.76±0.29 •	50.72±0.33 ○
Delta Elevators	49.78±0.31	47.63±0.42 •	50.34±0.29 ○
2D Planes	75.37±0.10	75.37±0.06	75.29±0.07
Pole Telecomm	95.05±0.12	95.05±0.10	94.94±0.07
Friedman Artificial	66.50±0.19	64.83±0.18 •	64.01±0.13 •
MV Artificial	99.19±0.04	99.20±0.04	99.19±0.02
Kinematics of Robot Arm	47.28±0.34	43.69±0.41 •	42.58±0.70 •
Computer Activity (1)	65.96±0.38	63.75±0.32 •	65.03±0.32 •
Computer Activity (2)	68.69±0.47	66.80±0.47 •	66.76±0.44 •
Census Domain (1)	53.32±0.22	50.20±0.36 •	51.46±0.34 •
Census Domain (2)	51.49±0.26	48.96±0.33 •	50.97±0.21 •
Auto MPG	59.74±0.98	59.58±1.86	59.46±1.25
Auto Price	66.80±2.20	62.39±2.71 •	63.50±1.43 •
Boston Housing	60.97±1.41	59.34±1.49 •	59.70±1.65
Diabetes	29.00±3.14	26.55±5.21	33.80±2.63 ○
Pyrimidines	43.27±2.85	42.27±3.51	42.68±2.78
Triazines	40.03±2.51	38.90±3.07	37.14±2.40 •
Machine CPU	58.10±1.32	56.78±2.78	56.62±2.43
Servo	52.45±1.60	55.16±2.09 ○	49.82±1.65 •
Wisconsin Breast Cancer	21.36±2.04	22.92±3.48	21.71±1.40
Pumadyn Domain (1)	49.83±0.30	46.04±0.43 •	48.28±0.34 •
Pumadyn Domain (2)	65.75±0.35	62.67±0.42 •	63.51±0.37 •
Bank Domain (1)	74.04±0.24	73.14±0.31 •	73.27±0.36 •
Bank Domain (2)	38.68±0.52	37.37±0.59 •	36.01±0.22 •
California Housing	64.83±0.24	63.30±0.18 •	64.36±0.18 •
Stocks Domain	86.85±0.67	87.05±0.88	85.19±0.53 •

○, • statistically significant improvement or degradation

Table 3. Pair-wise comparison for target value discretized into five bins: number indicates how often method in column (significantly) outperforms method in row

	C4.5-ORD	C4.5	C4.5-1PC
C4.5-ORD	–	**4 (1)**	**6 (5)**
C4.5	**23 (17)**	–	15 (11)
C4.5-1PC	**22 (16)**	14 (7)	–

Table 4. Experimental results for target value discretized into three bins: percentage of correct classifications, and standard deviation

Dataset	C4.5-ORD	C4.5	C4.5-1PC
Abalone	66.07±0.26	63.90±0.24 •	65.91±0.34
Ailerons	75.37±0.31	74.78±0.37 •	73.79±0.34 •
Delta Ailerons	80.54±0.26	80.34±0.17	80.9 ±0.17 ○
Elevators	64.43±0.17	62.22±0.25 •	63.63±0.36 •
Delta Elevators	70.83±0.22	69.89±0.26 •	70.73±0.27
2D Planes	86.61±0.04	86.61±0.05	86.52±0.09 •
Pole Telecomm	95.90±0.12	95.64±0.10 •	95.58±0.1 •
Friedman Artificial	80.78±0.09	80.23±0.14 •	80.3 ±0.18 •
MV Artificial	99.51±0.02	99.53±0.02	99.55±0.03 ○
Kinematics of Robot Arm	64.37±0.37	63.76±0.24 •	64.88±0.38 ○
Computer Activity (1)	79.04±0.39	78.44±0.43 •	78.38±0.31 •
Computer Activity (2)	81.02±0.42	80.76±0.31	80.21±0.4 •
Census Domain (1)	70.53±0.17	69.58±0.27 •	70.95±0.18 ○
Census Domain (2)	69.53±0.21	68.19±0.28 •	69.61±0.23
Auto MPG	78.81±1.26	78.12±1.14	79.16±1.32
Auto Price	86.25±1.38	85.36±1.60	85.87±1.27
Boston Housing	75.52±1.07	74.83±1.72	74.79±1.18
Diabetes	54.45±2.61	51.00±3.91	54.45±2.61
Pyrimidines	56.89±3.98	50.11±2.95 •	55.12±3.31
Triazines	54.47±3.28	54.18±3.20	54.11±2.84
Machine CPU	73.88±2.39	71.85±2.44	74.26±2.77
Servo	77.24±1.16	75.56±1.26 •	79.22±1.27 ○
Wisconsin Breast Cancer	35.98±2.33	37.40±3.23	35.71±2.11
Pumadyn Domain (1)	66.71±0.37	66.02±0.28 •	67.37±0.32 ○
Pumadyn Domain (2)	78.83±0.15	77.65±0.41 •	77.64±0.3 •
Bank Domain (1)	85.89±0.28	86.02±0.28	85.61±0.11 •
Bank Domain (2)	57.15±0.45	55.84±0.38 •	53.94±0.26 •
California Housing	78.94±0.11	79.02±0.17	79.42±0.16 ○
Stocks Domain	91.74±0.53	91.24±0.53	91.77±0.45

○, • statistically significant improvement or degradation

Table 5. Pair-wise comparison for target value discretized into three bins: number indicates how often method in column (significantly) outperforms method in row

	C4.5-ORD	C4.5	C4.5-1PC
C4.5-ORD	–	**4 (0)**	**11 (7)**
C4.5	**24 (15)**	–	18 (11)
C4.5-1PC	**17 (10)**	11 (4)	–

perform markedly better than C4.5-1PC: it is significantly more accurate on 10 datasets, and significantly worse on seven. It is interesting to see that the one-per-class encoding outperforms plain C4.5 on these three-class datasets.

If the significance of the individual differences is not taken into account and ignoring the smaller version of each pair of datasets from the same domain, C4.5-ORD wins against C4.5 on 20 datasets, and looses on four. This difference is statistically significant according to a two-sided sign test: the corresponding p-value is 0.0008. However, compared to C4.5-1PC, the win/loss-ratio for C4.5-ORD is 15/10, with a corresponding p-value of 0.21. Thus there is only very weak evidence that the ordinal classification method improves on the standard one-per-class encoding.

For the ten-bin case we expect a more significant difference, in particular when compared to the one-per-class method. This is confirmed in the experimental results summarized in Tables 6 and 7. The difference in performance between C4.5-ORD and C4.5 remains high but increases only slightly when moving from five to ten bins. C4.5-ORD outperforms C4.5 on all but seven of the datasets. It is significantly more accurate on 22 datasets, and significantly less accurate on only two. Compared to C4.5-1PC, C4.5-ORD is significantly more accurate on 25 datasets, and significantly worse on two. This is a marked improvement over the five-bin case and suggests that ordering information becomes more useful as the number of classes increases.

Not considering the significance of individual differences and ignoring the smaller version of each pair of datasets from the same domain, C4.5-ORD wins against C4.5 on 19 datasets, and looses on six. According to a two-sided sign test this ratio is significant at the 0.01%-level. Thus there is strong evidence that C4.5-ORD outperforms C4.5 on a collection of datasets of this type. Compared to C4.5-1PC, the win/loss-ratio for C4.5-ORD is 21/4, with a corresponding p-value of 0.0005. Consequently there is very strong evidence that the ordinal classification method improves on the standard one-per-class encoding.

4 Related Work

The ordinal classification method discussed in this paper is applicable in conjunction with any base learner that can output class probability estimates. Kramer *et al.* [5] investigate the use of a learning algorithm for regression tasks—more specifically, a regression tree learner—to solve ordinal classification problems. In this case each class needs to be mapped to a numeric value. Kramer *et al.* [5] compare several different methods for doing this. However, if the class attribute represents a truly ordinal quantity—which, by definition, cannot be represented as a number in a meaningful way—there is no principled way of devising an appropriate mapping and this procedure is necessarily *ad hoc*.

Herbrich *et al.* [3] propose a strategy for ordinal classification that is based on a loss function between pairs of true ranks and predicted ranks. They present a corresponding algorithm similar to a support vector machine, and mention that their approach can be extended to other types of linear classifiers.

Potharst and Bioch [6] present a decision tree learning algorithm for *monotone* learning problems. In a monotone learning problem both the input at-

Table 6. Experimental results for target value discretized into ten bins: percentage of correct classifications, and standard deviation

Dataset	C4.5-ORD	C4.5	C4.5-1PC
Abalone	29.48±0.36	26.68±0.61 •	27.43±0.58 •
Ailerone	39.36±0.33	36.64±0.37 •	35.76±0.32 •
Delta Ailerons	43.06±0.37	41.31±0.61 •	43.30±0.30
Elevators	31.41±0.34	28.58±0.35 •	29.77±0.38 •
Delta Elevators	39.86±0.33	36.9 ±0.44 •	41.44±0.23 ○
2D Planes	54.95±0.14	53.00±0.14 •	51.25±0.32 •
Pole Telecomm	91.18±0.08	90.85±0.14 •	89.34±0.15 •
Friedman Artificial	44.55±0.17	41.06±0.10 •	32.79±0.38 •
MV Artificial	98.11±0.03	98.17±0.05	97.36±0.06 •
Kinematics of Robot Arm	25.83±0.36	24.39±0.28 •	20.37±0.38 •
Computer Activity (1)	45.61±0.38	42.20±0.46 •	42.12±0.34 •
Computer Activity (2)	48.77±0.55	45.58±0.65 •	45.60±0.48 •
Census (1)	32.58±0.18	30.5 ±0.23 •	29.00±0.21 •
Census (2)	31.51±0.27	29.33±0.22 •	28.95±0.19 •
Auto MPG	36.55±1.18	39.63±1.70 ○	24.65±1.09 •
Auto Price	48.40±2.01	36.82±2.40 •	34.37±3.01 •
Boston Housing	42.66±1.00	38.61±1.25 •	35.93±1.65 •
Diabetes	14.55±3.12	23.00±3.12 ○	19.80±2.06 ○
Pyrimidines	19.39±2.95	23.89±2.69	15.93±1.83 •
Triazines	20.51±1.21	16.50±1.94 •	12.22±1.74 •
Machine CPU	36.35±2.47	36.23±1.48	30.59±1.93 •
Servo	34.57±0.98	34.60±1.47	13.18±0.03 •
Wisconsin Breast Cancer	10.73±1.91	11.24±3.15	11.33±1.24
Pumadyn Domain (1)	26.49±0.38	23.69±0.61 •	16.6 ±0.43 •
Pumadyn Domain (2)	45.76±0.44	41.87±0.42 •	41.21±0.54 •
Bank Domain (1)	52.76±0.37	49.57±0.44 •	43.58±0.82 •
Bank Domain (2)	25.51±0.42	24.20±0.33 •	24.06±0.39 •
California Housing	44.77±0.28	42.67±0.32 •	39.47±0.27 •
Stocks Domain	74.83±1.33	72.69±0.76 •	72.09±0.70 •

○, • statistically significant improvement or degradation

Table 7. Pair-wise comparison for target value discretized into ten bins: number indicates how often method in column (significantly) outperforms method in row

	C4.5-ORD	C4.5	C4.5-1PC
C4.5-ORD	–	**6 (2)**	**4 (2)**
C4.5	**23 (22)**	–	6 (3)
C4.5-1PC	**25 (25)**	22 (17)	–

tributes and the class attribute are assumed to be ordered. This is different from the setting considered in this paper because we do not assume that the input is ordered.

Although machine learning algorithms for ordinal classification are rare, there are many statistical approaches to this problem. However, they all rely on specific distributional assumptions for modeling the class variable and also assume a stochastic ordering of the input space [3].

The technique of generating binary "dummy" attributes to replace an ordered attribute can also be applied to the attributes making up the input space. Frank and Witten [2] show that this often improves performance compared to treating ordered attributes as nominal quantities. In cases where both the input and the output are ordered, this technique can be applied in addition to the method discussed in this paper to obtain further performance improvements.

The method presented in this paper is also related to the use of error-correcting output codes for (unordered) multi-class problems [1]. Instead of using error-correcting bit vectors to represent each class, we use bit vectors that reflect the ordering of the class values. As opposed to choosing the bit vector with the closest Hamming distance when making a prediction, our method selects the vector which corresponds to the largest estimated class probability (computed according to the procedure discussed in Section 2).

5 Conclusions

This paper presents a simple method that enables standard classification algorithms to make use of ordering information in ordinal class attributes. The method converts the original ordinal class problem into a series of binary class problems that encode the ordering of the original classes. Empirical results based on C4.5 show that this procedure is significantly more accurate than plain C4.5, and C4.5 used in conjunction with the standard one-per-class method. They also show that the performance gap increases with the number of classes. Our findings demonstrate that the improvement in performance is a result of exploiting ordering information and not simply as a side effect of transforming the problem into a series of binary-class problems.

References

1. T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.
2. E. Frank and I. H. Witten. Making better use of global discretization. In *Proceedings of the Sixteenth International Conference on Machine Learning*, Bled, Slovenia, 1999. Morgan Kaufmann.
3. R. Herbrich, T. Graepel, and K. Obermayer. Regression models for ordinal data: A machine learning approach. Technical report, TU Berlin, 1999.
4. R. Kohavi. *Wrappers for Performance Enhancement and Oblivious Decision Graphs.* PhD thesis, Stanford University, Department of Computer Science, 1995.
5. S. Kramer, G. Widmer, B. Pfahringer, and M. DeGroeve. Prediction of ordinal classes using regression trees. *Fundamenta Informaticae*, 2001.

6. R. Potharst and J.C. Bioch. Decision trees for ordinal classification. *Intelligent Data Analysis*, 4(2):97–112, 2000.
7. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, 1993.
8. L. Torgo. Regression Data Sets. University of Porto, Faculty of Economics, Porto, Portugal, 2001. [`http://www.ncc.up.pt/~ltorgo/Regression/DataSets.html`].
9. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, 2000.