Local Ordinal Classification

Sotiris B. Kotsiantis
Educational Software Development Laboratory
Department of Mathematics
University of Patras, Greece
sotos@math.upatras.gr

Abstract. Given ordered classes, one is not only concerned to maximize the classification accuracy, but also to minimize the distances between the actual and the predicted classes. This paper offers an organized study on the various methodologies that have tried to handle this problem and presents an experimental study of these methodologies with the proposed local ordinal technique, which locally converts the original ordinal class problem into a set of binary class problems that encode the ordering of the original classes. The paper concludes that the proposed technique can be a more robust solution to the problem because it minimizes the distances between the actual and the predicted classes as well as improves the classification accuracy.

1 Introduction

Ordinal classification can be viewed as a bridging problem between the two standard machine-learning tasks of classification and regression. In ordinal classification, the target values are in a finite set (like in classification) but there is an ordering among the elements (like in regression, but unlike classification).

Although Machine Learning (ML) algorithms for ordinal classification are rare, there are a number of 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 [9]. The ML community has mainly addressed the issue of ordinal classification in two ways. One is to apply classification algorithms by discarding the ordering information in the class attribute [2]. The other is to apply regression algorithms by transforming class values to real numbers [9]. This paper proposes a local ordinal technique that locally converts the original ordinal problem into a set of binary problems encoding the ordering of the original classes. Experimental results show that this technique minimizes the distances between the actual and the predicted class, as well as improves the prediction accuracy.

Please use the following format when citing this chapter:
Kotsiantis, Sotiris, 2006, in IFIP International Federation for Information Processing, Volume 204, Artificial Intelligence Applications and Innovations, eds. Maglogiannis, I., Karpouzis, K., Bramer, M., (Boston: Springer), pp. 1–8

This paper is organized as follows: the next section discusses the different techniques that have been presented for handling ordinal classification problems. In section 3, we describe the proposed technique. In Section 4, we present the experimental results of our methodology using different distribution algorithms and compare these results with those of other approaches. In the final section of the paper we discuss further work and some conclusions.

2 Techniques for Dealing with Ordinal Problems

Classification algorithms can be applied to ordinal prediction problems by discarding the ordering information in the class attribute. However, some information that could improve the performance of a classifier is lost when this is done.

The use of regression algorithms to solve ordinal problems has been examined in [9]. In this case each class needs to be mapped to a numeric value. 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 upright way of devising an appropriate mapping and this procedure is ad hoc.

Another approach is to reduce the multi-class ordinal problem to a set of binary problems using the one-against-all approach [2]. In the one-against-all approach, a classifier is trained for each of the classes using as positive examples the training examples that belong to that class, and as negatives all the other training examples. The estimates given by each binary classifier are then coupled in order to obtain class probability membership estimates for the multi-class problem [2].

A more sophisticated approach that enables classification algorithms to make use of ordering information in ordinal class attributes is presented in [7]. Similarly with previous method, this method converts the original ordinal class problem into a set of binary class problems that encode the ordering of the original classes. However, to predict the class value of an unseen instance this algorithm needs to estimate the probabilities of the m original ordinal classes using $m - 1$ models. For example, for a three class ordinal problem, estimation of the probability for the first ordinal class value depends on a single classifier: Pr(*Target* < *first value*) as well as for the last ordinal class: Pr(*Target* > *second value*). Whereas, for class value in the middle of the range, the probability depends on a pair of classifiers and is given by

Pr(*Target* > *first value*) * (1 − Pr(*Target* > *second value*)).

3 Proposed Technique

The proposed technique is based on the previous referred sophisticated technique [7]; however, we do not apply this technique globally but locally. If all training instances are taken into account when classifying a new test case, the classifier works as a global method, while when the nearest training instances are taken into account, the classifier works as a local method, since only data local to the area around the testing instance contribute to the classification.

Generally, local methods have significant advantages when the probability measure defined on the space of symbolic features for each class is very complex, but can still be described by a collection of less complex local approximations [1]. The proposed algorithm builds the required number of classifiers for each point to be estimated, taking into account only a subset of the training points. This subset is chosen on the basis of the preferable distance metric between the testing point and the training point in the input space.

In other words, the proposed technique consists of the four steps in Fig. 1.

1. Determine a suitable distance metric.
2. Find the k nearest neighbors using the selected distance metric.
3. Estimate the probabilities of the m original ordinal classes with m − 1 models using as training instances these k instances
4. The estimates given by each binary classifier are then coupled in order to obtain class probability membership estimates

Fig. 1. Local Ordinal Technique

The proposed ensemble has some free parameters such as the distance metric. In our experiments, we used the most well known -Euclidean similarity function- as distance metric. We also used k=50 since about this size of instances is appropriate for a simple algorithm to built a precise model [6].

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. In the following section, we empirically evaluate the performance of our approach with the other well known techniques.

4 Experiments

To test the hypothesis that the above method improves the generalization performance on ordinal prediction problems, we performed experiments on real-world ordinal datasets donated by Dr. Arie Ben David (http://www.cs.waikato.ac.nz/ml/weka/). We also used well-known datasets from many domains from the UCI repository [3]. However, the used UCI datasets represented numeric prediction problems and for this reason we converted the numeric target values into ordinal quantities using equal-size binning. This unsupervised discretization method divides the range of observed values into three equal size intervals. The resulting class values are ordered, representing variable-size intervals of the original numeric quantity. This method was chosen because of the lack of numerous benchmark datasets involving ordinal class values.

All accuracy estimates were obtained by averaging the results from 10 separate runs of stratified 10-fold cross-validation. It must be mentioned that we used the free available source code for most algorithms by the book [11]. In the following we present the empirical results obtained using Decision Stump (DS) [8], RepTree [11] and Naïve Bayes (NB) [5] algorithms as base learners. All of them produce class probability estimates.

Table 1 shows the results for the DS algorithm applied (a) without any modification of DS, (b) in conjunction with the ordinal classification method presented in Section 2 (Ordinal DS), (c) in conjunction with the multiclass classification method presented in Section 2 (Multiclass DS) and (d) using the proposed technique (Local Ordinal DS).

In Table 1, for each data set the algorithms are compared according to classification accuracy (the rate of correct predictions) and to mean absolute error:

$$\frac{|p_1 - a_1| + |p_2 - a_2| + ... + |p_n - a_n|}{n}$$

where p: predicted values and a: actual values. Moreover, in Table 1, we represent as "v" that the specific algorithm performed statistically better than the proposed method according to t-test with $p<0.05$. Throughout, we speak of two results for a dataset as being "significant different" if the difference is statistical significant at the 5% level according to the corrected resampled t-test [10], with each pair of data points consisting of the estimates obtained in one of the 100 folds for the two learning methods being compared. On the other hand, "*" indicates that proposed method performed statistically better than the specific algorithm according to t-test with $p<0.05$.

As one can observe from the aggregated results in Table 1, the proposed technique is more accurate than the remaining approaches from 2% to 5%. Moreover, it manages to minimize the distances between the actual and the predicted classes. The reduction of the mean absolute error is about 27% compared to the Ordinal DS and 30% compared to the simple DS, while it exceeds the 138% compared to the Multiclass DS. It must be also mentioned that the proposed method is statistically more accurate and has statistically less mean absolute error than the remaining methods in numerous datasets.

Similarly, Table 2 shows the results for the NB algorithm applied (a) without any modification of NB, (b) in conjunction with the ordinal classification method presented in Section 2 (Ordinal NB), (c) in conjunction with the multiclass classification method presented in Section 2 (Multiclass NB) and (d) using the proposed technique (Local Ordinal NB).

As one can see from the aggregated results in Table 2, the proposed technique is more accurate in classification accuracy than the remaining techniques from 2% to 5%. Furthermore, it minimizes the distances between the actual and the predicted classes. In detail, the reduction of the mean absolute error is about 25% compared to the Ordinal NB and 17% compared to simple NB, while it overcomes the 158% compared to Multiclass NB. It must be also stated that the proposed method is statistically more accurate and has statistically less mean absolute error than the remaining methods in a lot of datasets.

Similarly, Table 3 shows the results for the RepTree algorithm applied (a) without any modification of RepTree, (b) in conjunction with the ordinal classification method presented in Section 2 (Ordinal RepTree), (c) in conjunction with the multiclass classification method presented in Section 2 (Multiclass RepTree) and (d) using the proposed technique (Local Ordinal RepTree).

As one can notice from the aggregated results in Table 3, the proposed technique is more accurate in classification accuracy than the remaining techniques from 1% to 2%. What is more, it minimizes the distances between the actual and the predicted classes since the reduction of the mean absolute error is about 15% compared to the Ordinal RepTree and simple RepTree, while it overcomes the 138% compared to Multiclass RepTree. The proposed method is also statistically more accurate and has statistically less mean absolute error than the remaining methods in many datasets.

Table 1. Results for DS algorithm

Dataset		Local Ordinal DS	Multiclass DS	Ordinal DS	DS
auto93	accuracy	80.90	80.57	79.59	81.32
	MeanError	0.14	0.34*	0.18	0.18
autoHorse	accuracy	95.24	91.17	89.63*	91.17
	MeanError	0.04	0.30*	0.09*	0.09*
autoMpg	accuracy	79.67	79.76	78.01	79.61
	MeanError	0.14	0.35*	0.20*	0.21*
autoPrice	accuracy	88.11	89.80	89.80	86.05
	MeanError	0.09	0.31*	0.10	0.13*
bodyfat	accuracy	97.57	99.12	99.12	91.98*
	MeanError	0.02	0.29*	0.01	0.10*
cleveland	accuracy	70.32	71.63	71.14	71.93
	MeanError	0.21	0.37*	0.26*	0.26*
Cloud	accuracy	84.69	87.72	83.43	84.51
	MeanError	0.11	0.32*	0.13	0.14*
Cpu	accuracy	98.09	97.76	97.76	98.24
	MeanError	0.01	0.28*	0.02	0.02
Era	accuracy	25.69	22.08*	24.13	21.81*
	MeanError	0.18	0.20*	0.18*	0.19*
Esl	accuracy	65.53	44.48*	53.72*	43.03*
	MeanError	0.09	0.20*	0.13*	0.16*
fishcatch	accuracy	97.35	92.37*	92.37*	90.56*
	MeanError	0.03	0.30*	0.07*	0.10*
housing	accuracy	79.58	74.81	75.77	70.39*
	MeanError	0.15	0.36*	0.23*	0.28*
hungarian	accuracy	79.06	81.78	81.78	81.78
	MeanError	0.15	0.34*	0.20*	0.20*
Lev	accuracy	61.79	43.86*	49.03*	42.40*
	MeanError	0.20	0.31*	0.25*	0.26*
lowbwt	accuracy	57.25	61.80	61.90	61.90
	MeanError	0.30	0.39*	0.31	0.31
pharynx	accuracy	68.98	73.85	73.85	73.85
	MeanError	0.25	0.37*	0.25	0.25
servo	accuracy	89.72	83.36*	83.24*	83.36*
	MeanError	0.09	0.31*	0.13*	0.12*
Strike	accuracy	98.85	99.06	99.06	99.06
	MeanError	0.01	0.27*	0.01	0.01
swd	accuracy	56.11	51.38*	54.56	51.80*
	MeanError	0.26	0.36*	0.29*	0.30*

Veteran	accuracy	90.45	91.26	90.80	91.26
	MeanError	0.10	0.31*	0.11	0.11
AVERAGE	accuracy	78.25	75.88	76.43	74.80
	MeanError	0.13	0.31	0.16	0.17

Table 2. Results for NB algorithm

Dataset		Local Ordinal NB	Multiclass NB	Ordinal NB	NB
auto93	accuracy	84.36	76.28	74.01	76.18
	MeanError	0.10	0.33*	0.17*	0.16
autoHorse	accuracy	95.14	91.06	90.87	90.67*
	MeanError	0.03	0.29*	0.06*	0.06*
autoMpg	accuracy	82.56	80.65	70.11*	78.89
	MeanError	0.12	0.32*	0.20*	0.15*
autoPrice	accuracy	90.31	91.51	91.45	90.25
	MeanError	0.07	0.30*	0.06	0.07
bodyfat	accuracy	88.96	79.64*	77.22*	81.34*
	MeanError	0.08	0.32*	0.16*	0.13*
cleveland	accuracy	72.45	74.82	75.51	73.31
	MeanError	0.19	0.34*	0.18	0.19
Cloud	accuracy	90.30	91.70	92.04	89.95
	MeanError	0.07	0.30*	0.07	0.08
Cpu	accuracy	97.81	97.56	94.87	97.56
	MeanError	0.01	0.28*	0.04*	0.02
Era	accuracy	23.25	24.73	25.07	24.88
	MeanError	0.18	0.20*	0.18	0.18
Esl	accuracy	67.37	66.84	54.65*	67.52
	MeanError	0.09	0.19*	0.12*	0.10*
fishcatch	accuracy	97.42	89.92*	88.13*	90.10*
	MeanError	0.02	0.30*	0.08*	0.07*
housing	accuracy	81.44	74.76*	56.15*	73.14*
	MeanError	0.13	0.34*	0.29*	0.19*
hungarian	accuracy	81.17	83.95	83.95	83.95
	MeanError	0.13	0.31*	0.12v	0.12v
Lev	accuracy	59.95	56.24*	57.95	56.12*
	MeanError	0.20	0.31*	0.23*	0.23*
lowbwt	accuracy	60.10	58.79	58.52	59.53
	MeanError	0.29	0.39*	0.30	0.30
pharynx	accuracy	70.17	71.09	71.13	70.52
	MeanError	0.24	0.36*	0.25	0.25*
servo	accuracy	87.59	87.24	86.48	87.12
	MeanError	0.10	0.31*	0.12*	0.12*
Strike	accuracy	99.19	99.06	99.06	99.05
	MeanError	0.01	0.27*	0.02*	0.02*
swd	accuracy	50.17	57.31v	56.01v	56.77v
	MeanError	0.27	0.35*	0.26v	0.26v
Veteran	accuracy	89.31	88.48	88.70	86.88
	MeanError	0.09	0.32*	0.12*	0.13*
AVERAGE	accuracy	78.45	77.08	74.59	76.69
	MeanError	0.12	0.31	0.15	0.14

Table 3. Results for RepTree algorithm

Dataset		Local Ordinal RepTree	Multiclass RepTree	Ordinal RepTree	RepTree
auto93	accuracy	82.41	79.73	80.14	80.06
	MeanError	0.14	0.35*	0.20*	0.19*
autoHorse	accuracy	94.45	92.34	94.01	93.17
	MeanError	0.05	0.29*	0.07	0.07
autoMpg	accuracy	81.68	81.34	80.66	80.41
	MeanError	0.14	0.34*	0.17*	0.17*
autoPrice	accuracy	88.86	87.99	88.35	87.81
	MeanError	0.09	0.31*	0.10	0.11
bodyfat	accuracy	96.78	98.88	98.88	98.80
	MeanError	0.03	0.27*	0.01v	0.01v
cleveland	accuracy	71.08	71.73	68.39	71.36
	MeanError	0.21	0.36*	0.26*	0.24*
Cloud	accuracy	86.32	88.54	87.78	88.70
	MeanError	0.12	0.31*	0.11	0.10
Cpu	accuracy	98.04	97.00	96.95	97.29
	MeanError	0.01	0.28*	0.04*	0.03*
Era	accuracy	25.68	19.24*	26.20	26.60
	MeanError	0.18	0.20*	0.18	0.18
Esl	accuracy	66.08	60.59*	62.65	62.37
	MeanError	0.10	0.19*	0.11	0.11*
fishcatch	accuracy	96.71	94.88	94.05	94.70
	MeanError	0.03	0.28*	0.05	0.04
housing	accuracy	80.43	79.51	79.03	78.65
	MeanError	0.16	0.34*	0.18	0.18
hungarian	accuracy	78.62	78.70	78.46	78.46
	MeanError	0.17	0.34*	0.19	0.19
Lev	accuracy	63.16	60.43*	60.79	59.87*
	MeanError	0.20	0.31*	0.20	0.21*
lowbwt	accuracy	56.87	58.89	58.47	58.63
	MeanError	0.32	0.40*	0.34	0.33
pharynx	accuracy	69.79	65.06*	65.01 *	65.31*
	MeanError	0.28	0.40*	0.34*	0.34*
servo	accuracy	93.31	91.42	92.71	90.72
	MeanError	0.06	0.30*	0.07	0.08*
Strike	accuracy	98.97	99.21	99.21	99.21
	MeanError	0.01	0.27*	0.01	0.01
swd	accuracy	56.99	57.45	57.68	56.46
	MeanError	0.27	0.35*	0.26	0.27
Veteran	accuracy	89.20	91.26	91.19	90.90
	MeanError	0.11	0.31*	0.11	0.12
AVERAGE	accuracy	78.77	77.71	78.03	77.97
	MeanError	0.13	0.31	0.15	0.15

5 Conclusion

This paper is devoted to the problem of learning to predict ordinal (i.e., ordered discrete) classes. The local ordinal classification method discussed in this paper is applicable in conjunction with any learning algorithm that can output class probability estimates. According to our experiments in synthetic and real ordinal data sets, it manages to minimize the distances between the actual and the predicted classes, without harming but actually improving the classification accuracy in conjunction with DS, RepTree and NB algorithms. Drawing more general conclusions from these experimental data seems unwarranted. Our results so far show that the proposed methodology for predicting ordinal classes can be naturally derived from classification algorithms, but more extensive experiments will be needed to establish the precise capabilities and relative advantages of this methodology.

For large datasets, the benefit of local ordinal models is somewhat offset by the cost of storing and querying the training dataset for each test set instance. For this reason, in a following project we will focus on the problem of reducing the size of the stored set of instances while trying to maintain or even improve generalization performance by avoiding noise and over-fitting. In [4], numerous instance selection methods that can be combined with the proposed technique can be found.

References

1. C. G. Atkeson, A.W. Moore, S. Schaal, Locally weighted learning. Artificial Intelligence Review 11 (1997) 11–73.
2. E. L. Allwein, R. E. Schapire, Y. Singer, Reducing multiclass to binary: A unifying approach for margin classifiers. Journal of Machine Learning Research 1 (2000) 113–141.
3. C.L. Blake, C.J. Merz, UCI Repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science. [http://www.ics.uci.edu/~mlearn/MLRepository.html] (1998).
4. H. Brighton, C. Mellish, Advances in Instance Selection for Instance-Based Learning Algorithms, Data Mining and Knowledge Discovery, 6 (2002) 153–172.
5. P. Domingos, M. Pazzani, On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning, 29 (1997) 103-130.
6. E. Frank, M. Hall, B. Pfahringer, Locally weighted naive Bayes. Proc. of the 19th Conference on Uncertainty in Artificial Intelligence. Acapulco, Mexico. Morgan Kaufmann (2003).
7. E. Frank, M. Hall, A simple approach to ordinal prediction, L. De Raedt and P. Flach (Eds.): ECML 2001, LNAI 2167, (2001) 145-156, Springer-Verlag Berlin.
8. W. Iba, P. Langley, Induction of one-level decision trees. Proc. of the Ninth International Machine Learning Conference (1992). Aberdeen, Scotland: Morgan Kaufmann.
9. S. Kramer, G. Widmer, B. Pfahringer, M. DeGroeve, Prediction of ordinal classes using regression trees. Fundamenta Informaticae (2001).
10. C. Nadeau, Y. Bengio, Inference for the Generalization Error. Machine Learning 52(3): 239-281 (2003).
11. I. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann, San Mateo (2000).