International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:1, No:12, 2007

An ensemble of Weighted Support Vector Machines for Ordinal Regression

Willem Waegeman, and Luc Boullart

Abstract**—** Instead of traditional (nominal) classification we investigate the subject of ordinal classification or ranking. An enhanced method based on an ensemble of Support Vector Machines (SVM's) is proposed. Each binary classifier is trained with specific weights for each object in the training data set. Experiments on benchmark datasets and synthetic data indicate that the performance of our approach is comparable to state of the art kernel methods for ordinal regression. The ensemble method, which is straightforward to implement, provides a very good sensitivity-specificity trade-off for the highest and lowest rank.

Keywords**—**Ordinal regression, support vector machines, ensemble learning.

I. Introduction

CLASSIFICATION and (metric) regression are by far the most studied supervised learning problems in the machine learning domain. In the case of regression or function estimation labels can have continuous values. For classification class labels might take binary or nominal values. Ordinal regression shows resemblance to both of them because labels are chosen from a finite, in general small number of discrete, ordered values. Therefore the words ordinal classification and ranking are used as synonyms for ordinal regression. Categories typically correspond to quotations or linguistic terms — varying from "very bad" to "brilliant" for example — that express a difference in correctness, quality, beauty or any other characteristic of the analyzed objects.

Ordinal response variables are frequently observed in medicine, psychology and related sciences where human interaction often forms a part of the data generation process. As a consequence this learning task has received much attention by statisticians, which resulted in standard data analysis tools like cumulative logit models and its variants [1]. Information retrieval can also be mentioned as an important application of ranking learning and less common applications can be found in the literature, such as analyzing beef meat quality [2] or determining the skills of computer game players [3].

Research on this topic is relatively new in the machine learning community with only a few papers published in the last decade. Different authors present methods in which tree structures are used as classifiers. Kramer et al. [4] report various ways of transforming classification and regression trees (CART) into effective learners for ordinal classification tasks. They map the ordinal scale to a continuous scale in a preprocessing step and hence ignore the absence of a metric for ordinal values. Cao-van [5] uses decision trees and instance based methods to solve ranking problems and makes a distinction between ordinal regression and ranking by adding additional constraints on the ranking problem setting. A similar approach with decision trees as classifiers is found in [6]. Preference learning, which is another related problem setting, is discussed by Cohen et al. [7]. They developed greedy and online algorithms to learn preference relations. Crammer and Singer [8] proposed an online algorithm for ranking that is called PRank or perceptron ranking. Harrington [9] extended this algorithm to a large margin version by using averaging strategies like bagging.

W. Waegeman and L. Boullart are with Ghent University, Dept. of Electrical Energy, Systems and Automation, Technologiepark 913 9052 Ghent, Belgium (phone: + 32 9 264 55 86, email: willem.waegeman@ugent.be).

W. Waegeman is supported by a grant of the "Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen)".

II. Kernel Methods for Ordinal Regression

Ordinal regression has been studied before in the framework of kernel methods and statistical learning theory. Herbrich et al. [10] make a theoretical study on this subject and suggest to model ranks by intervals on a continuous scale. They present a large margin algorithm that formulates an ordering by preference judgments and derive generalization bounds for ordinal regression. In [11] other algorithms are given for batch and online ranking. Shashua and Levin [12] embed the continuous scale as a reference in kernel space as well and extend C-Support Vector Regression (C-SVR) for ordinal response variables. Therefore, a direction w and $r-1$ thresholds b_i are chosen (for ranks $1,...,r$). According to structural risk minimization the weighted sum of the norm $||w||$ and the training errors ξ and ξ^* are minimized with regularization parameter C. Consecutive ranks are separated in kernel space by $r-1$ parallel hyperplanes that lie perpendicular to the direction defined by the unobserverd latent variable.

In some cases it can happen that the thresholds are unordered. Chu and Keerthi [13] propose a solution to tackle this problem by adding the ordering of the thresholds as a constraint to the original optimization problem. They also derive another algorithm that penalizes errors of more than one rank heavier, i.e.

$$min_{w,b,\xi,\xi^*} \quad \frac{1}{2}||w||^2 + C\sum_{j=1}^{r-1}(\sum_{k=1}^{j}\sum_{i=1}^{n^k}\xi_{ki}^j + \sum_{k=1}^{j}\sum_{i=1}^{n^k}\xi_{ki}^{*j})$$

$$\text{subject to} \begin{cases} w\cdot\Phi(x_i^k) - b_j \leq -1 + \xi_{ki}^j \\ \xi_{ki}^j \geq 0, \\ \text{for } k = 1,...,j \text{ and } i = 1,...,n^k; \\ w\cdot\Phi(x_i^k) - b_j \geq +1 - \xi_{ki}^{*j} \\ \quad \xi_{ki}^{*j} \geq 0, \\ \text{for } k = j+1,...,r \text{ and } i = 1,...,n^k; \end{cases} \tag{1}$$

The transformation function Φ will be replaced by a kernel function $K(.,.)$ in the dual formulation and n^k denotes the number of training examples for class k. The function $f(x) = w \cdot \Phi(x)$ lies in a Reproducing Kernel Hilbert Space (RKHS) induced by the kernel function $K(.,.)$. From the Representer Theorem follows that f can be written as a function of the training objects: $f(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i)$ with α_i unknown variables that need to be estimated. This (kernel) property holds for a wide branch of pattern analysis methods of which the support vector machine is the most frequently used (see [14] and [15] for further information). We will compare our approach with this ranking algorithm in section V.

III. Ensembles of Support Vector Machines for Ordinal Regression

It has been shown that kernel based ordinal regression methods perform very well, but the proposed methods solve specific optimization problems for which new algorithms need to be derived to find estimates in a reasonable time. We show that a comparable performance can be obtained with a rather simple ensemble of SVM's. Ensembles with weak classifiers as base learners have been proposed for ordinal regression (see some of the references above). Frank and Hall [16] took decision trees as base learners to estimate a ranking function. In a first step we will do the same with kernel machines as underlying classifiers. Therefore $r-1$ SVM's are trained for a ranking problems with r ranks. For each binary classifier the training instances are transformed to the positive or the negative class depending on whether their label is greater than p or not, with p varying from 1 to $r-1$.

$$y_{pi} = \begin{cases} -1 & \text{if } y_i \leq p \\ 1 & \text{if } y_i > p \end{cases} \tag{2}$$

In the following step we attach weights v_{pi} to each of the n training examples. The weighting of an object differs for each binary classifier. As a result $r-1$ optimization problems of the following form are solved, i.e.

$$\begin{aligned} & min_{w_p, b_p, \xi_{pi}} \quad \frac{1}{2}||w_p||^2 + C\sum_{i=1}^{n} v_{pi}\xi_{pi} \\ & \text{subject to} \begin{cases} y_{pi} \cdot w_p \cdot \Phi(x_i) - b_p \geq 1 - \xi_{pi}, \\ \xi_{pi} \geq 0, \\ y_i = 1, ..., r \\ p = 1, ..., r-1 \\ \text{and } i = 1, ..., n; \end{cases} \end{aligned} \tag{3}$$

The weights for each binary system are assigned in such a way that errors on training objects are penalized proportional to the absolute difference between their rank and the value of p.

$$v_{pi} = \begin{cases} (p+1-y_i)\frac{|\{x_i|y_i \leq p\}|}{\sum_{i=1;y_i \leq p}^{n}(p+1-y_i)} & \text{if } y_i \leq p \\ (y_i - p)\frac{|\{x_i|y_i > p\}|}{\sum_{i=1;y_i > p}^{n}(y_i - p)} & \text{if } y_i > p \end{cases} \tag{4}$$

We also scale the weights so that the sum of training errors of the positive class and the sum of training errors of the negative class stays the same, compared to the same ensemble of binary SVM's without object weights.

$$\sum_{i=1;y_i \leq p}^{n} v_{pi} + \sum_{i=1;y_i > p}^{n} v_{pi} = n \quad p = 1, ..., r-1 \tag{5}$$

In this way ensembles of weighted and unweighted SVM's can be easily compared for a particular choice of the regularization parameter C and the kernel function. As a consequence assigning different weights to objects in our way doesn't have the side effect that the $r-1$ binary optimization problems become more balanced. Balancing the subproblems will in general lead to undesirable outcomes on an independent test set, although the class distributions of the binary systems might be very unbalanced. For example when $p = 1$, the positive class exists of the training objects with rank 1 and the negative class is formed by training objects with rank $2, ..., r$ and. In this case we are dealing with a very unbalanced problem. Penalizing errors of the positive class more severely will obviously increase the power to correctly identify objects of rank 1, but it will also affect the global performance of the ensemble drastically. This will be further discussed in the experiments section.

Rank estimates for the test set are obtained by combining the estimated outcomes $\widehat{y}_{pi}$ of the $r-1$ binary classifiers. The intuitive interpretation of these binary outcomes says that $\widehat{y}_i > k$ if $\widehat{y}_{ki} = 1$. We will assign rank k to object x_i so that $\widehat{y}_{pi} = 1$ for all p smaller than k and $\widehat{y}_{pi} = -1$ for all p equal to or bigger than k. From a theoretical perspective it can happen that this strategy leads to an ambiguity for few test objects. However, on the data sets that were investigated (see section 5), such objects were not detected. For multi-class classification these ambiguities often occur and are solved by choosing the class with the highest probability after training the ensemble with binary SVM's that generate probability estimates for each class. Abe and Inoue [17] present fuzzy techniques to tackle this problem in a One-versus-All ensemble. Similar strategies could be applied for our ranking algorithm but this wasn't necessary for the analyzed datasets. Such objects could also be seen as outliers in the test set.

IV. Implementation

The weights v_{pi} give rise to a slightly different optimization problem than the original SVM quadratic program. The Lagrange primal function is given by:

$$\begin{aligned} L_P \quad = \quad & \frac{1}{2}||w_p||^2 + C\sum_{i=1}^{n} v_{pi}\xi_{pi} \\ & - \sum_{i=1}^{n} \alpha_{pi} y_{pi} w_p \cdot \Phi(x_i) + b_p - (1 - \xi_{pi}) \\ & - \sum_{i=1}^{n} \mu_{pi}\xi_{pi} \end{aligned} \tag{6}$$

and can be minimized w.r.t. the primal variables by setting the respective derivatives to zero. We get

$$w_p = \sum_{i=1}^{n} \alpha_{pi} y_{pi} \Phi(x_i), \quad (7)$$

$$0 = \sum_{i=1}^{n} \alpha_{pi} y_{pi}, \quad (8)$$

$$\alpha_{pi} = C v_{pi} - \mu_i, \forall i \quad (9)$$

with constraints $\alpha_{pi}, \mu_{pi}, \xi_{pi} \geq 0$. By substituting this into (6) we obtain the Wolfe dual:

$$L_D = \sum_{i=1}^{n} \alpha_{pi} - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_{pi} \alpha_{pj} y_{pi} y_{pj} \Phi(x_i) \Phi(x_j). \quad (10)$$

There is no need to compute the dot-product in kernel space explicitly. It can be replaced by the kernel function. We conclude that L_D stays the same when object weights are included in the primal objective function. The only difference is that we maximize L_D subject to $0 \leq \alpha_{pi} \leq C v_{pi}$. To implement this optimization problem, we carried out some modifications to the SMO-algorithm (JAVA version) of the LibSVM-software [18].

V. Experiments

We compared our approach — further referred to as multi-class ordinal support vector machines (MCOSvm) — to the method summarized in section II, which is called support vector ordinal regression with implicit constraints (Svorim). We chose to compare with Svorim for three reasons: (a) the method is very similar to our approach, (b) the authors claim to achieve good generalization performance and (c) their software is publicly accessible.

A. Generalization Performance

We picked 4 data sets from the ones that were analyzed by [13]: *boston housing, abalone, machine cpu and computer*[1].

These data sets are all (metric) regression data sets. The continuous labels were discretized into 5 ranks so that each rank contains an equal number of instances. In the next phase 20 random partitions into training set and test set were created from the converted data.

We have a strong comment with the procedure of discretizing metric regression data sets for evaluating ranking methods. By discretizing one will never achieve a good seperation between successive ranks. This conclusion also holds for some real world ordinal regression problems, but definitely not for all. Therefore we also gave attention to the generalization performance on 2 synthetic data sets. For the first data set we sampled $4 * 100$ instances from 4 bivariate Gaussian clusters with consecutive ranks. The mean of the clusters is set to (10,10), (20,10), (20,20) and (30,20) respectively, σ_1,σ_2 and ρ are fixed to 3, 3 and 0. This process was repeated 40 times, as a result of which 20 train sets and 20 test sets of size 400 were obtained. Figure 1 shows a scatter plot of one of the training data sets. For the second synthetic problem again for Gaussian clusters with means (10,10), (20,10), (30,10) and (40,10) were considered. This time 10% uniform noise was added to the data to test the robustness of the ranking methods. This gave 20 train sets of size 440 and 20 test sets of size 400. One of the training data sets is displayed in figure 2.

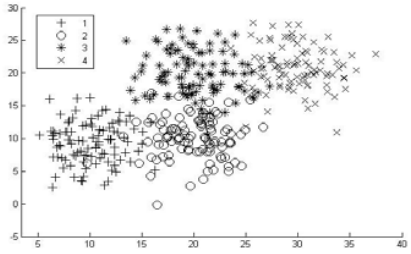

Fig. 1 One of the training data sets generated for the first experiment with synthethic data

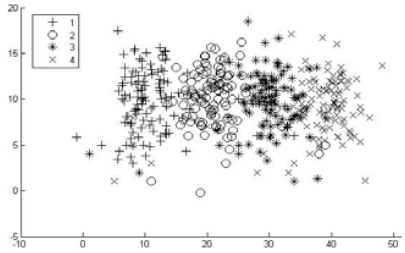

Fig. 2 One of the training data sets generated for the second experiment with synthethic data

We only examined the Gaussian RBF-kernel in our experiments.

$$K(x, z) = e^{-\gamma ||x-z||^2} \quad (11)$$

The original software was used to obtain the results for Svorim. This program does a coarse grid search followed by a fine search in the zone with initial best settings to determine the optimal values for C and γ. Our JAVA extension of LibSVM carries out a grid search in the same region ($-10 \leq log_2 \gamma \leq 5$ and $-5 \leq log_2 C \leq 10$) as the LibSVM software and with the same step size ($log_{10} 0.5$), but without zooming. Five fold stratified cross-validation on train set was done to evaluate each search point[2].

Different measures are used to compare the generalization performance of ordinal regression methods, We looked at three types of measures. Accuracy (Acc) is the standard measure for (nominal) classification and is simply the percentage of correct predictions. Mean absolute error takes the difference between

[1]The other data sets of [13] were not considered for various reasons. We believe that *pyrimidines* is too small for the ranking problem setting because it contains only 74 objects. Two versions of the data set *bank* were available. The data sets *california housing* and *census* need too much computing power. All data sets can be downloaded from http://www.liacc.up.pt/ltorgo/Regression/DataSets.html.

[2]For the large data set *computer* we took the search region smaller for both methods to reduce computation time

the real rank and the predicted rank into account, i.e.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y}_i| \quad (12)$$

In addition "local" performance measures were also investigated. The sensitivity and the specificity between two consecutive ranks give an impression of the kind of incorrect rank predictions that occur the most often. We define the sensitivity and the specificity of a two consecutive ranks p and $p+1$ as:

$$Sens_p^{p+1} = \frac{|\{x_i|y_i \leq r \wedge \widehat{y}_i \leq r\}|}{|\{x_i|\widehat{y}_i \leq r\}|} \quad (13)$$

$$Spec_p^{p+1} = \frac{|\{x_i|y_i > r \wedge \widehat{y}_i > r\}|}{|\{x_i|\widehat{y}_i > r\}|} \quad (14)$$

Table I and II give an overview of the results averaged over twenty test sets for Acc. and MAE. A Wilcoxon rank sum test was used to determine whether the difference between MCOSvm and Svorim is statistically significant. For accuracy both methods win on some of the data sets, but the difference is only significant at the 0.05-level for *computer*. For MAE Svorim is slightly better on three of the analyzed problems. MCOSvm is again significantly better on this measure for *computer*.

TABLE I
ACCURACY FOR THE ANALYZED METHODS ON BENCHMARK DATASETS TOGETHER WITH THE P-VALUE OF THE WILCOXON RANK TEST

Dataset	Acc. MCOSvm	Acc Svorim	p-value
Boston Housing	66,52 %	67,44 %	0.34
Abalone	47.4 %	47.3 %	0.96
Computer	73.06 %	71.22 %	< 0.001
Machine CPU	41.37 %	40.08 %	0.59
Synthetic1.	91.25 %	90.95 %	0.71
Synthetic2.	86.03 %	85.22 %	0.16

Consequently the sensitivity and specificity between rank one and two was computed. We noticed that MCOSvm always gave better results than Svorim for the "border" ranks (ranks close to rank 1 and rank r). This leads to a much higher sensitivity on these ranks. Table III gives an overview of the sensitivity and the specificity for rank 1. One can see that the sensitivity is much higher for MCOSvm for all data sets, while the specificity doesn't change much. Similar results were obtained for the highest rank with a much higher specificity for MCOSvm while the sensitivity is equal for both methods, except for *computer*. It seems that MAE on the one hand and $Sens_1^2$ and $Spec_{r-1}^r$ on the other hand are

TABLE II
MEAN ABSOLUTE ERROR OF THE ANALYZED METHODS ON BENCHMARK DATASETS TOGETHER WITH THE P-VALUE OF THE WILCOXON RANK TEST

Dataset	MAE MCOSvm	MAE Svorim	p-value
Boston Housing	0.38	0.35	0.02
Abalone	0.66	0.65	0.02
Computer	0.29	0.30	< 0.001
Machine CPU	0.49	0.43	0.02
Synthetic1	0.092	0.093	0.74
Synthetic2	0.18	0.18	0.88

TABLE III
SENSITIVITY AND SPECIFICITY OF THE LOWEST RANK FOR MCOSVM AND SVORIM

	MCOSvm		Svorim	
Dataset	$Sens_1^2$	$Spec_1^2$	$Sens_1^2$	$Spec_1^2$
Boston Housing	0.80	0.95	0.75	0.96
Abalone	0.45	0.95	0.39	0.96
Computer	0.86	0.98	0.87	0.97
Machine CPU	0.63	0.93	0.51	0.97
3clusters	0.95	0.98	0.92	0.98
4clusters	0.89	0.96	0.86	0.96

contradictory performance measures. The data set *computer* is a good example to confirm this point of view because it is the only experiment where MCOSvm did'n give a better sensitivity at the lowest rank and simultaneously MCOSvm achieved a significant lower value for MAE. Svorim is a bit "biased" towards the middle ranks and this is the reason why this method gives better results for MAE on three of the data sets.

We claim that a high sensitivity at the border ranks is very important for ranking systems. In information retrieval for example, ordinal regression methods are often used to rank web sites in accordance with their relevance to a search query that is typed by a user of the search engine. In many cases the user will only look at the first few results, so sensitivity and specificity of the highest ranks are very important in this case. Another example can be found in the domain of quality control, where it is crucial to detect products of very low quality without decreasing the specificity. It is clear that our approach will give better results in these situations.

B. Computation Time

It is clear that time complexity of MCOSvm is comparable to the One-versus-all ensemble for multi-class classification. Here $r-1$ optimization problems with n constraints are solved (instead of r for One-versus-all). The quadratic program of Svorim has $(r-1)n$ constraints. Therefore our approach is also comparable to Svorim. Actually the quadratic program of Svorim is split into $r-1$ smaller problems. Some time could be gained because non support vectors slow down the optimization process not very much. Unfortunately it was not possible to compare both algorithms because they are written in different program languages. Allthough MCOSvm was written in JAVA and Svorim in C, we noticed that MCOSvm was much faster. MCOSvm could obviously take advantage of the speedup techniques built in LibSVM.

VI. CONCLUSION

An enhanced ensemble method for ordinal regression was proposed in this article. Weighted Support Vector Machines were used as base classifiers. Specific weights were assigned to each object in such a way that errors of more than one rank are heavier penalized. Therefore the weight of a training object differs for each binary SVM. The insertion of object weights doesn't affect the dual objective function. It only leads to slightly different constraints on the quadratic program. Experiments on benchmark data sets and synthetic dat confirm that

the ensemble performs good on unseen test data by comparison with another kernel based ranking method. A much higher sensitivity on the lowest rank was found, while the specificity of that rank is more or less the same for both methods. The high sensitivity on the lowest rank and the high specificity on the highest rank influence mean absolute error a little since both objectives work against each other. For accuracy no significant difference was found between both methods. Moreover the algorithm has a comparable time complexity then Svorim and will be faster in practice because existing (fast) implementations of SVM's can easily be modified.

REFERENCES

[1] A. Agresti, *Categorical Data Analysis, 2nd version*. John Wiley and Suns Publications, 2002.
[2] J. DelCoz, G. Bayn, J. Dez, O. Luaces, A. Bahamonde, and C. Saudo, "Trait selection for assessing beef meat quality using non-linear svm," in *Proceedings of the conference on Neural Information Processing Systems, Vancouver, Canada*, 2004, pp. 321–328.
[3] R. Herbrich, "The trueskill(tm) ranking system," 2005, http://www.research.microsoft.com/mlp/trueskill/.
[4] S. Kramer, G. Widmer, B. Pfahringer, and M. Degroeve, "Prediction of ordinal classes using regression trees," *Fundamenta Informaticae*, vol. 24, pp. 1–15, 2000.
[5] K. Cao-Van, "Supervised ranking, from semantics to algorithms," Ph.D. dissertation, Ghent University, Belgium, 2003.
[6] R. Potharst and J. Bioch, "Decision trees for ordinal classification," *Intelligent Data Processing*, vol. 4, no. 2, 2000.
[7] W. Cohen, R. Schapire, and Y. Singer, "Learning to order things," in *Advances in Neural Information Processing Systems 10*, 1998.
[8] K. Crammer and Y. Singer, "Pranking with ranking," in *Proceedings of the conference on Neural Information Processing Systems (NIPS)*, 2001.
[9] E. Harrington, "Online ranking/collaborative filtering using the perceptron algorithm," in *Proceedings of the 20th International Conference on Machine Learning, Washington, USA*, 2003.
[10] R. Herbrich, T. Graepel, and K. Obermayer, "Large margin rank boundaries for ordinal regression," in *Advances in Large Margin Classifiers*, 2000, pp. 115–132.
[11] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
[12] A. Shashua and A. Levin, "Ranking with the large margin principle: two approaches," *Advances in Neural Information Processing Systems*, vol. 15, 2003.
[13] W. Chu and S. Keerthi, "New approaches to support vector ordinal regression," in *Proceedings of the 22th International Conference of Machine Learning, Bonn, Germany*, 2005, pp. 321–328.
[14] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2001.
[15] B. Scholkopf and A. Smola, *Learning with Kernels, Support Vector Machines, Regularisation, Optimization and beyond*. The Mit Press, 2002.
[16] E. Frank and M. Hall, "A simple approach to ordinal classification," in *Proceedings of the European Conference on Machine Learning*, 2000.
[17] S. Abe and T. Inoue, "Fuzzy support vector machines for multiclass problems," in *European Symposium on Artificial Neural Networks*, 2002, pp. 113–118.
[18] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," 2001, http://www.csie.ntu.edu.tw/ cjlin/libsvm.