Journal of Machine Learning Research 20 (2019) 1-5 Submitted 6/18; Revised 8/19; Published 8/19

ORCA: A Matlab/Octave Toolbox for Ordinal Regression

Javier Sánchez-Monedero SANCHEZ-MONEDEROJ@CARDIFF.AC.UK
School of Journalism, Media and Culture
Cardiff University
Cardiff CF10 1FS, Wales, United Kingdom

Pedro A. Gutiérrez PAGUTIERREZ@UCO.ES
Department of Computer Science and Numerical Analysis
University of Córdoba
Córdoba, 14071, Spain

María Pérez-Ortiz MARIA.PEREZ@UCL.AC.UK
Department of Computer Science
University College London
London WC1E 6EA, United Kingdom

Editor: Cheng Soon Ong

Abstract

Ordinal regression, also named ordinal classification, studies classification problems where there exist a natural order between class labels. This structured order of the labels is crucial in all steps of the learning process in order to take full advantage of the data.

ORCA (Ordinal Regression and Classification Algorithms) is a Matlab/Octave framework that implements and integrates different ordinal classification algorithms and specifically designed performance metrics. The framework simplifies the task of experimental comparison to a great extent, allowing the user to: (i) describe experiments by simple configuration files; (ii) automatically run different data partitions; (iii) parallelize the executions; (iv) generate a variety of performance reports and (v) include new algorithms by using its intuitive interface. Source code, binaries, documentation, descriptions and links to data sets and tutorials (including examples of educational purpose) are available at `https://github.com/ayrna/orca`.

Keywords: Ordinal regression, ordinal classification, Matlab, Octave, threshold models

1. Introduction

The terms ordinal regression and ordinal classification refer to those supervised learning problems where labels show an ordinal arrangement (Gutiérrez et al., 2016), e.g. in an age estimation problem where the categories are: {*baby, child, teenager, adult*}. The aim in this case is not just to improve standard accuracy, but to reduce the magnitude of misclassification errors, in such a way that the order relation between labels is considered during model construction and evaluation. In the problem of age estimation, if true label is *baby*, predicted label *adult* should entail a more severe misclassification error than predicted label *child*. Exploiting the ordinal disposition of the labels has proven to yield better performance than simply treating them as nominal categories.

In this paper, we introduce ORCA (Ordinal Regression and Classification Algorithms), a Matlab/Octave framework that gathers an extensive collection of recent ordinal machine learning methods and ordinal performance metrics. It also presents a general framework to automate experiments that can be used both from an API or by describing experiments with configuration files. Online documentation includes extensive tutorials to introduce users to ordinal classification and the pipeline of the experimental framework. We also provide several small data sets for code testing purposes and a list of 44 ordinal data sets of different characteristics that can be used for algorithm comparison.

There are some alternative software toolboxes that implement certain features for ordinal regression. Those include: i) `mord`[1] in Python, ii) `ordinal`[2] in R, iii) `vgam`[3] in R, iv) `bmrm`[4] in R and v) `ocapis`[5] in Scala. `mord` and `ordinal` focus on well-established simple statistical methods (i.e. ordinal logistic regression, which is also included in our toolbox) rather than on novel machine learning approaches. `vgam` and `bmrm` focus on vector generalised linear and additive models and regularised empirical minimisation respectively, having a setting to deal with ordinal classification problems. Finally, `ocapis` implements 4 of the 15 methods that ORCA includes but lacks the experimental and parallelisation framework.

2. Architecture and Features

The main features can be grouped in the following categories:

Methods. The `Algorithm` class defines an API including different methods such as `fit` and `predict`, which process a pair of train and test partitions for a specific hyper-parameter configuration of the selected classifier. Adding a new method is easy, the user only needs to add a new class implementing the `fit` and `predict` methods.

Metrics. All metrics implement a `calculateMetric` method, which can be run using the true and predicted labels or the confusion matrix.

Automatic experiment running. ORCA automates the process of running one or many methods for a set of data sets. It processes every experiment by fitting a model with the training data (including model selection through cross-validation), evaluating the test error and creating performance reports of all the performance metrics, training time and hyper-parameter values.

Experiment configuration. ORCA can be used by directly interacting with the API or by describing experiments with configuration files in INI format. ORCA checks the consistency of these files by analysing the corresponding algorithm class, in such a way that, if a new method is developed, the researcher does not need to modify the INI parser.

Experiment parallelisation. When presenting new classifier proposals, researchers typically run several methods over a set of partitions. For example, if we perform 30 repetitions of a hold-out design for 10 data sets, we need at least 300 calls to the `fit` and `predict`

1. https://pythonhosted.org/mord/
2. https://cran.r-project.org/web/packages/ordinal/index.html
3. https://cran.r-project.org/web/packages/VGAM/index.html
4. https://cran.r-project.org/web/packages/bmrm/index.html
5. https://arxiv.org/abs/1810.09733

Method	Reference
Ordinal methods	
Support vector regression (SVR)	Gutiérrez et al. (2016)
Cost-sensitive support vector classifier (CSSVC)	Hsu and Lin (2002)
Support vector machine with ordered partition (SVMOP)	Waegeman and Boullart (2009)
Ordinal extreme learning machine (ELMOP)	Deng et al. (2010)
Linear logistic regression for ordinal data (POM)	McCullagh (1980)
Ordinal SVM with explicit constraints (SVOREX)	Chu and Keerthi (2007)
Ordinal SVM with implicit constraints (SVORIM)	Chu and Keerthi (2007)
Linear SVORIM	Chu and Keerthi (2007)
Kernel discriminant analysis for ordinal regression (KDLOR)	Sun et al. (2010)
Neural network based on the POM (NNPOM)	Mathieson (1996)
Neural network with ordered partitions (NNOP)	Cheng et al. (2008)
Reduction applied to SVM (REDSVM)	Lin and Li (2012)
Ordinal regression boosting (ORBoost)	Lin and Li (2006)
Ordinal projection based ensemble (OPBE)	Pérez-Ortiz et al. (2014)
Partial order methods	
Hierarchical Partial Order Label Decomposition (HPOLD)	Sánchez-Monedero et al. (2018)
Nominal methods	
Multi-class nominal SVM with 1vs1 formulation (SVC1V1)	Hsu and Lin (2002)
Multi-class nominal SVM with 1vsAll formulation (SVC1VA)	Hsu and Lin (2002)
Matlab wrapper for LIBLINEAR (LIBLINEAR)	Fan et al. (2008)

Table 1: Ordinal and nominal methods available in `ORCA`.

functions, which can be run in parallel. `ORCA` exploits this by using Matlab and Octave parallelisation toolboxes. In addition, a set of scripts is provided to perform the parallelisation in the HTCondor[6] distributed computing environment.

3. Implemented Methods and Performance Metrics

`ORCA` collects an extensive list of ordinal classification methods including naïve approaches, ordinal binary decompositions and threshold models (see Table 1). Further details of the methods can be found in Gutiérrez et al. (2016), together with running time of the algorithms. From this analysis, it was concluded that ELMOP, SVORLin and POM are the best option if computational cost is a priority. The training time of neural network methods (NNPOM and NNOP) and GPOR is generally the highest. This cost can be assumed for GPOR, since it obtains very good performance for balanced ordinal data sets, while neural network-based methods are generally beaten by the ordinal SVM variants. Concerning scalability, the experimental setup in Gutiérrez et al. (2016) also included some relatively large data sets, so the practitioner could check the time it took to train one of those models with the `ORCA` framework. In general, linear models such as POM and SVORLin perform very well in these scenarios where there is plenty of data while still having a reasonably low

6. `http://research.cs.wisc.edu/htcondor/`

running time (e.g. around 10 seconds for cross-validating, training and testing on a data set of almost 22.000 patterns).

ORCA provides a collection of performance metrics, which can also be used for hyper-parameter selection. Given that ordinal problems usually show a skewed class distribution specific imbalance classification metrics are also included in the framework. For more details, we refer the reader to the project documentation and ordinal metrics review in (Cruz-Ramírez et al., 2014).

4. Sample Code

The ORCA API includes methods to directly configure, train and test an ordinal classifier:

```
% Create an Algorithm object
addpath('src/Algorithms/')
kdlorAlgorithm = KDLOR();
% Load data set
load exampledata/1-holdout/toy/matlab/train_toy.0
load exampledata/1-holdout/toy/matlab/test_toy.0
train.patterns = train_toy(:,1:(size(train_toy,2)-1));
train.targets = train_toy(:,size(train_toy,2));
test.patterns = test_toy(:,1:(size(test_toy,2)-1));
test.targets = test_toy(:,size(test_toy,2));
% Fit the model and predict with test data
result = kdlorAlgorithm.fitpredict(train,test);
% Evaluate performance metrics
addpath('src/Measures/')
CCR.calculateMetric(result.predictedTest,test.targets)
MAE.calculateMetric(result.predictedTest,test.targets)
```

Moreover, as previously discussed, ORCA can be used to run a batch of experiments specified in an INI file:

```
Utilities.runExperiments('tutorial/config-files/pom.ini')
```

Acknowledgments

We would like to thank Professor Peter Tiňo and the three anonymous reviewers for their valuable feedback. Javier Sánchez-Monedero and María Pérez-Ortiz started the development of this software while working at the University of Córdoba (Spain). This work has been subsidized by the projects TIN2017-85887-C2-1-P and TIN2017-90567-REDT of the Spanish Ministry of Economy and Competitiveness (MINECO) and EU FEDER funds.

References

Jianlin Cheng, Zheng Wang, and Gianluca Pollastri. A neural network approach to ordinal regression. In *IEEE International Joint Conference on Neural Networks*, pages 1279–1284. IEEE Press, 2008.

Wei Chu and S. Sathiya Keerthi. Support Vector Ordinal Regression. *Neural Computation*, 19(3):792–815, 2007.

Manuel Cruz-Ramírez, César Hervás-Martínez, Javier Sánchez-Monedero, and Pedro A. Gutiérrez. Metrics to guide a multi-objective evolutionary algorithm for ordinal classification. *Neurocomputing*, 135:21–31, July 2014.

Wan-Yu Deng, Qing-Hua Zheng, Shiguo Lian, Lin Chen, and Xin Wang. Ordinal extreme learning machine. *Neurocomputing*, 74(1–3):447–456, 2010.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Pedro A. Gutiérrez, María Pérez-Ortiz, Javier Sánchez-Monedero, Francisco Fernandez-Navarro, and César Hervás-Martínez. Ordinal regression methods: survey and experimental study. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):127–146, 2016.

Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multi-class support vector machines. *IEEE Transaction on Neural Networks*, 13(2):415–425, 2002.

Hsuan-Tien Lin and Ling Li. Large-margin thresholded ensembles for ordinal regression: Theory and practice. In José L. Balcázar, Philip M. Long, and Frank Stephan, editors, *International Conference on Algorithmic Learning Theory*, volume 4264, pages 319–333. Springer-Verlag, October 2006.

Hsuan-Tien Lin and Ling Li. Reduction from cost-sensitive ordinal ranking to weighted binary classification. *Neural Computation*, 24(5):1329–1367, 2012.

Mark J. Mathieson. Ordinal models for neural networks. In J. Moody A.-P. N. Refenes, Y. Abu-Mostafa and A. Weigend, editors, *International Conference on Neural Networks in the Capital Markets*, pages 523–536. World Scientific, 1996.

Peter McCullagh. Regression models for ordinal data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 42(2):109–142, 1980.

María Pérez-Ortiz, Pedro A. Gutiérrez, and César Hervás-Martínez. Projection-based ensemble learning for ordinal regression. *IEEE Transactions on Cybernetics*, 44(5):681–694, May 2014.

Javier Sánchez-Monedero, María Pérez-Ortiz, Aurora Saez, Pedro A. Gutiérrez, and César Hervás-Martínez. Partial order label decomposition approaches for melanoma diagnosis. *Applied Soft Computing*, 64:341–355, March 2018.

Bing-Yu Sun, Jiuyong Li, Desheng Dash Wu, Xiao-Ming Zhang, and Wen-Bo Li. Kernel discriminant learning for ordinal regression. *IEEE Transactions on Knowledge and Data Engineering*, 22(6):906–910, 2010.

Willem Waegeman and Luc Boullart. An ensemble of weighted support vector machines for ordinal regression. *International Journal of Computer Systems Science and Engineering*, 3(1):47–51, 2009.