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](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\footnote{https://pythonhosted.org/mord/} in Python, ii) ordinal\footnote{https://cran.r-project.org/web/packages/ordinal/index.html} in R, iii) vgam\footnote{https://cran.r-project.org/web/packages/VGAM/index.html} in R, iv) bmrm\footnote{https://cran.r-project.org/web/packages/bmrm/index.html} in R and v) ocapis\footnote{https://arxiv.org/abs/1810.09733} 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
Table 1: Ordinal and nominal methods available in ORCA.

<table>
<thead>
<tr>
<th>Method</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support vector regression (SVR)</td>
<td>Gutiérrez et al. (2016)</td>
</tr>
<tr>
<td>Cost-sensitive support vector classifier (CSSVC)</td>
<td>Hsu and Lin (2002)</td>
</tr>
<tr>
<td>Support vector machine with ordered partition (SVMOP)</td>
<td>Waegeman and Boullart (2009)</td>
</tr>
<tr>
<td>Ordinal extreme learning machine (ELMOP)</td>
<td>Deng et al. (2010)</td>
</tr>
<tr>
<td>Linear logistic regression for ordinal data (POM)</td>
<td>McCullagh (1980)</td>
</tr>
<tr>
<td>Ordinal SVM with explicit constraints (SVOREX)</td>
<td>Chu and Keerthi (2007)</td>
</tr>
<tr>
<td>Ordinal SVM with implicit constraints (SVORIM)</td>
<td>Chu and Keerthi (2007)</td>
</tr>
<tr>
<td>Linear SVORIM</td>
<td>Chu and Keerthi (2007)</td>
</tr>
<tr>
<td>Kernel discriminant analysis for ordinal regression (KDLOR)</td>
<td>Sun et al. (2010)</td>
</tr>
<tr>
<td>Neural network based on the POM (NNPOM)</td>
<td>Mathieson (1996)</td>
</tr>
<tr>
<td>Neural network with ordered partitions (NNOP)</td>
<td>Cheng et al. (2008)</td>
</tr>
<tr>
<td>Reduction applied to SVM (REDSVM)</td>
<td>Lin and Li (2012)</td>
</tr>
<tr>
<td>Ordinal regression boosting (ORBoost)</td>
<td>Lin and Li (2006)</td>
</tr>
<tr>
<td>Ordinal projection based ensemble (OPBE)</td>
<td>Pérez-Ortiz et al. (2014)</td>
</tr>
</tbody>
</table>

Partial order methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-class nominal SVM with 1vs1 formulation (SVC1V1)</td>
<td>Hsu and Lin (2002)</td>
</tr>
<tr>
<td>Multi-class nominal SVM with 1vsAll formulation (SVC1VA)</td>
<td>Hsu and Lin (2002)</td>
</tr>
<tr>
<td>Matlab wrapper for LIBLINEAR (LIBLINEAR)</td>
<td>Fan et al. (2008)</td>
</tr>
</tbody>
</table>

Nominal methods

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

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:

```matlab
% 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:

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

### Acknowledgments

We would like to thank Professor Peter Tiño 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


