oADABOOST
An ADABOOST *variant for Ordinal Classification*

João Costa[1], Jaime S. Cardoso[2]

[1] *Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal*

[2] *INESC TEC and Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, n^o 378, 4200-465 Porto, Portugal*

ei09008@fe.up.pt, jaime.cardoso@inescporto.pt

Keywords: Supervised Learning, Ordinal Classification, Ensemble Methods, ADABOOST.

Abstract: Ordinal data classification (ODC) has a wide range of applications in areas where human evaluation plays an important role, ranging from psychology and medicine to information retrieval. In ODC the output variable has a natural order; however, there is not a precise notion of the distance between classes. The Data Replication Method was proposed as tool for solving the ODC problem using a single binary classifier. Due to its characteristics, the Data Replication Method is straightforwardly mapped into methods that optimize the decision function globally. However, the mapping process is not applicable when the methods construct the decision function locally and iteratively, like decision trees and ADABOOST (with decision stumps). In this paper we adapt the Data Replication Method for ADABOOST, by softening the constraints resulting from the data replication process. Experimental comparison with state-of-the-art ADABOOST variants in synthetic and real data show the advantages of our proposal.

1 INTRODUCTION

One of the most representative problems of supervised learning is classification, consisting of the estimation of a mapping from the feature space into a finite class space. Depending on the cardinality of the output space, we are left with binary or multiclass classification problems. Finally, the presence or absence of a “natural” order among classes differentiates nominal from ordinal problems.

A large number of real world classification problems can be seen as ordinal tasks, that is, we want to learn a function that is able to classify points in a finite set of classes, which have an ordering relation between them (*e.g.* we might want to classify a movie as good, average or bad).

Note that, since only the order is known, one cannot simply pick an arbitrary mapping from our classes to numbers (*e.g.* $bad \mapsto 0, average \mapsto 1, good \mapsto 2$) and solve a regression problem, as our error function would be highly dependent of that mapping. Also, our classes might not have a valid numerical interpretation, (*e.g.* is a good movie equivalent to two average movies?) therefore choosing an appropriate mapping is not trivial.

One solution to ordinal problems is to simply ignore the ordering relation, treat them as nominal classification problems and use a learning algorithm designed for this kind of tasks (*e.g.* C4.5 decision trees and ADABOOST). A better alternative is to devise solutions that take advantage of the order information in the design of the classifier. A possible approach is to transform our dataset in such a way that we force our learning algorithm to respect the ordering relation. One such transformation is the Data Replication Method (Cardoso and da Costa, 2007), which transforms an ordinal problem into a larger binary classification one. One of the limitations of the Data Replication Method is that it cannot be immediately applicable when the decision function is constructed iteratively and locally, like in decision trees or ADABOOST.

In this work, we present a new ADABOOST variant that performs its growth on the replicated feature space, and therefore is able to use the ordering relation when training the weak classifiers.

1.1 Related Work

Frank and Hall presented a simple method that enables standard classification algorithms to make use of ordering information in class attributes (Frank and

Hall, 2001). By applying it in conjunction with a decision tree learner, the authors show that it outperforms the naive approach, which treats the class values as an unordered set. Compared to special-purpose algorithms for ordinal classification, the method has the advantage that it can be applied without any modification to the underlying learning scheme. The rationale encompasses using $(K-1)$ standard binary classifiers to address the K-class ordinal data problem. Toward that end, the training of the i-th classifier is performed by converting the ordinal dataset with classes $\mathcal{C}_1,\ldots,\mathcal{C}_K$ into a binary dataset, discriminating $\mathcal{C}_1,\ldots,\mathcal{C}_i$ against $\mathcal{C}_{i+1},\ldots,\mathcal{C}_K$. To predict the class value of an unseen instance, the $(K-1)$ outputs are combined to produce a single estimation. Any binary classifier can be used as the building block of this scheme. Observe that the $(K-1)$ classifiers are trained in an independent fashion. This independence is likely to lead to intersecting boundaries, a topic to which we will return further on in this paper.

The Data Replication Method (Cardoso and da Costa, 2007) overcomes the limitations identified above by building all the boundaries at once. That guarantees that the boundaries of the classifiers will never intersect. This method, however, has limitations with methods that build the decision function iteratively (and greedily), and therefore cannot be easily mapped to ADABOOST.

In the ensemble approach to ordinal data classification, although not directly related to our work, it is worth mentioning the work introducing global constraints in the design of decision trees (Cardoso and Sousa, 2010; Sousa and Cardoso, 2011). The method consists on growing a tree (or an ensemble of trees) and relabeling the leaves according to certain constraints. Therefore, the trees are still built without taking the order into account and only post-processed to satisfy ordinality constraints. Moreover, the post-processing is very computationally demanding, only possible in low dimensional input spaces. More recently, the combination of multiple orthogonal directions has been suggested to boost the performance of a base classifier (Sun et al., 2014). Sequentially, multiple orthogonal directions are found; these different directions are combined in a final stage.

There are also some boosting-related approaches for ordinal ranking. For example, RankBoost (Freund et al., 2003) approach is based on the pairwise comparison perspective. Lin and Li proposed ordinal regression boosting (ORBoost) (Lin and Li, 2006), which is a special instance of the extended binary classification perspective. The ensemble method most in line with our work is ADABOOST.OR (Lin and Li, 2009). This method uses a primal-dual approach to solve an ordinal problem both in the binary space and the ordinal space, by taking into account the order relation when updating the binary point's weights. However, ADABOOST.OR is more constrained than our proposed approach; while ADABOOST.OR is closer to a single ADABOOST instantiated with an ordinal data classifier, our approach is closer to having multiple ADABOOST coupled in the construction of the weak classifier.

2 BACKGROUND

In this section we start by analysing the Frank and Hall's approach to ordinal classification (Frank and Hall, 2001), which facilitates the introduction of the Data Replication Method (Cardoso and da Costa, 2007). The Data Replication Method is a framework for ordinal data classification that allows the application of most binary classification algorithms to ordinal classification and imposes a parallelism constraint on the resulting boundaries. In the end, we summarize the ADABOOST ensemble method, paving the way to the presentation of the proposed adaptation of ADABOOST to ordinal data.

2.1 Frank and Hall Method

Suppose we want to learn a function $f:\mathcal{X}\to\mathcal{Y}$, where $\mathcal{X}$ is our feature space and $\mathcal{Y}=\{\mathcal{C}_1,\mathcal{C}_2,...,\mathcal{C}_K\}$ is our output space, where our labels are ordered according to $\mathcal{C}_1\prec\mathcal{C}_2\prec...\prec\mathcal{C}_K$. Also, assume that we have a dataset $\mathcal{D}=(D,f)$, where $D\subseteq\mathcal{X}$ is our set of examples and $f:D\to\mathcal{Y}$ gives us the label of each example.

The Frank and Hall method transforms the K class ordinal problem into $(K-1)$ binary problems by creating $(K-1)$ datasets $\mathcal{D}_k=(D,f_k)$ where:

$$f_k(\mathbf{x})=\begin{cases}\mathcal{C}_- & \text{if } f(\mathbf{x})\preceq\mathcal{C}_k\\ \mathcal{C}_+ & \text{if } f(\mathbf{x})\succ\mathcal{C}_k\end{cases}$$

Intuitively, learning a binary classifier from each of the $\mathcal{D}_k$ datasets will create $(K-1)$ classifiers that answer the questions "is the label of point $\mathbf{x}$ larger than $\mathcal{C}_k$?". This is to say, each classifier will give us an estimate of $P(f(\mathbf{x})\succ\mathcal{C}_k)$.

Frank and Hall then propose that one finds the $P(f(\mathbf{x})=\mathcal{C}_k)$ using the usual rule:

$$\begin{cases}1-P(f(\mathbf{x})\succ\mathcal{C}_1) & \text{if } k=1\\ P(f(\mathbf{x})\succ\mathcal{C}_{k-1})-P(f(\mathbf{x})\succ\mathcal{C}_k) & \text{if } k\in[2,K-1]\\ P(f(\mathbf{x})\succ\mathcal{C}_{K-1}) & \text{if } k=K\end{cases}$$

Even though our conversion from ordinal to binary guarantees that $f_k(\mathbf{x}) = \mathcal{C}_- \Rightarrow f_{k+1}(\mathbf{x}) = \mathcal{C}_-$ and $f_k(\mathbf{x}) = \mathcal{C}_+ \Rightarrow f_{k-1}(\mathbf{x}) = \mathcal{C}_+$, those rules do not always hold for the learnt probabilities. In practice, this means that it is possible that the combination rule proposed by Frank and Hall returns a negative probability for some classes. One solution to this problem is setting that negative probabilities to zero. Another possible way to combine our binary classifiers is by a simple counting method: $\hat{f}(\mathbf{x}) = \mathcal{C}_i$, with $i = 1 + \sum_{k=1}^{K-1} [\![f_k(\mathbf{x}) = \mathcal{C}_+]\!]$[1]. In either way, the main conceptual problem is not addressed.

2.2 Consistency and Parallelism

One important concept in ordinal data classification is the idea of *consistency* with the ordinal setting (Cardoso and Sousa, 2010). The idea behind it is intuitive: a small change in the input data should not lead to a 'big jump' in the output decision (e.g. it is not expected that a small change in a feature makes the estimated product quality to go from "bad" to "good" without going through "average"). One way to guarantee this restriction is by enforcing that there is no intersection between decision boundaries, which can, in turn, be guaranteed by enforcing our boundaries to be parallel.

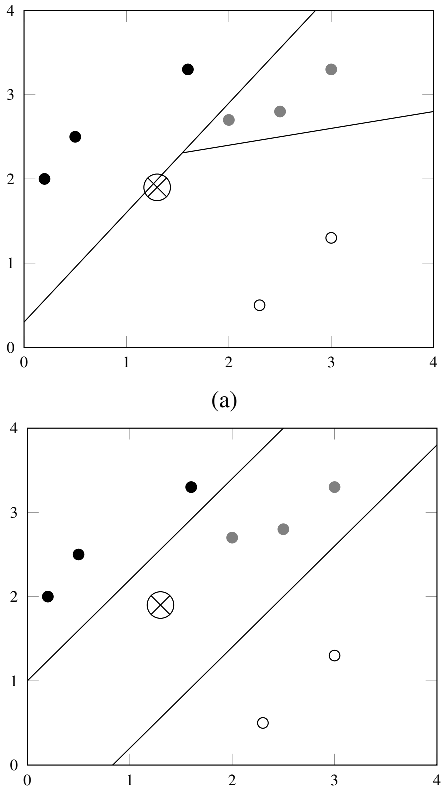
In Figure 1 it is possible to see two partitions of the input space corresponding to two different classifiers, one without the parallelism constraint and another one with it. In the first one, a small variation around the point marked with $\otimes$ can make an abrupt jump between two very distinct classes ("black" and "white"), while on the second classifier that is not possible.

It can be seen that the method proposed by Frank and Hall does not respect this concept, and therefore can lead to problematic classifiers.

2.3 The Data Replication Method

The Data Replication Method uses the idea behind the Frank and Hall method in a different way: instead of simply creating $(K-1)$ datasets, it extends the feature space so that all points from the $(K-1)$ replicas are present on the same dataset.

Assume $\mathbf{e}_0$ as a vector composed of $(K-2)$ zeros and a vector $\mathbf{e}_q$ as a vector composed by $(K-3)$ zeros and a positive constant (*e.g.* 1) on the q-th position (*e.g.* if $K = 5$, $e_2 = [0,1,0]$). We then transform each point $\mathbf{x} \in D$ into $(K-1)$ points $\mathbf{z}_q = (\mathbf{x}, \mathbf{e}_q)$. This does allows us to train a single binary classifier on the whole data and then combine the results in the same fashion as previously presented. Also, this replication guarantees that some algorithms such as Suport Vector Machines (SVMs) and Artificial Neural Networks (ANNs) will always produce parallel boundaries on the final ordinal space. A simple example of the Data Replication Method can be seen in Figure 2. Note that: a) $\mathbf{x} \in \mathbf{R}$ while $\mathbf{z} \in \mathbf{R}^2$; b) a single binary classifier is designed in the extended dataset to solve the multiclass problem in the original space. The intersection of the binary boundary with each of the $K-1$ replicas provide the necessary $K-1$ boundaries in the original space. Further details of the method can be found on the original paper (Cardoso and da Costa, 2007).



(a)

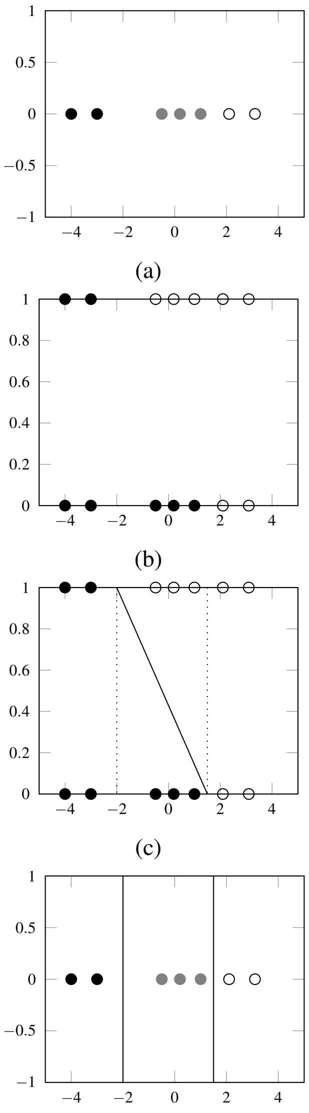
(b)

Figure 1: Comparison of a (a) non-consistent multiclass classifier vs. (b) a multiclass classifier with parallelism constraints

2.4 AdaBoost

AdaBoost is a boosting algorithm introduced by Freund and Schapire (Freund and Schapire, 1995). Boosting algorithms are a part of a big set of machine learning techniques called ensemble methods which general idea is to use several models to classify obser-

[1] $[\![\cdot]\!]$ is the indicator function. $[\![\cdot]\!]$ is 1 if the inner condition is true, 0 otherwise.



(a)

(b)

(c)

(d)

Figure 2: Toy example of the Data Replication Method. (a) Original Problem. (b) Problem on the replicated space. (c) Solution to the problem on the replicated space. (d) Result on the original space.

vations and combine them together to obtain a classifier with a predictive performance superior than any of its constituents. ADABOOST uses a *weak learner* to classify observations. A weak learner is defined as a classifier which is only slightly correlated with the true data labels. In the case of binary prediction, a weak learner is a classifier which is only slightly better than throwing a coin and deciding an object's class according to the trial's result.

During each iteration, the algorithm trains a weak learner $\hat{f}_t(\mathbf{x})$ using an iteratively determined distribution and selects the weak hypothesis minimizing the expected error rate. After selecting the best weak hypothesis $\hat{f}_t$ for the distribution D_t, the observations $\mathbf{x}_i$ correctly identified by $\hat{f}_t$ are weighted less than those misclassified, so that the algorithm will, when fitting a new weak hypothesis to D_{t+1} in the next iteration, select one such rule which identifies better those observations that its predecessor failed. The output of the ADABOOST algorithm is a final or combined hypothesis $\hat{f}_{\mathcal{F}}$. $\hat{f}_{\mathcal{F}}$ is simply the sign of a weighted combination of the weak hypothesis, i.e., H is a weighted majority rule of the weak classifiers, $\hat{f}_{\mathcal{F}}(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \hat{f}_t(\mathbf{x})\right)$.

3 oADABOOST

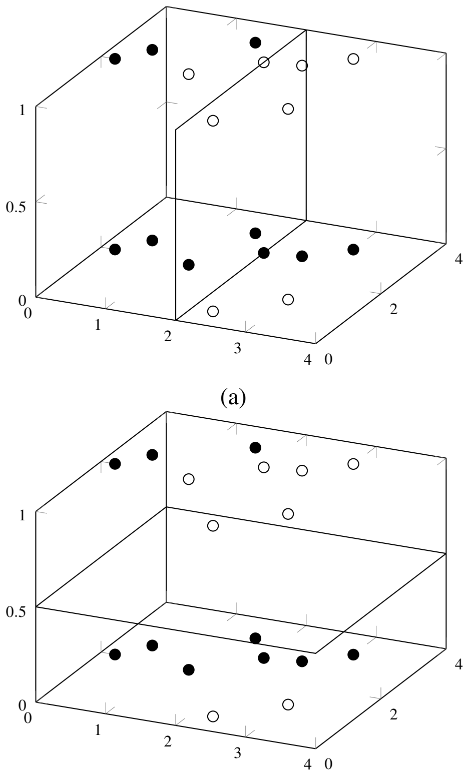
While the Data Replication Method (DRM) has already been instantiated with SVMs, ANNs and Kernel Discriminant Analysis (Cardoso and da Costa, 2007; Cardoso et al., 2012), its mapping to ADABOOST is not trivial, as illustrated in Figure 3 for an ADABOOST with a decision stump. This is due to the fact that a decision stump can only make cuts at one attribute at a time, and therefore, as it can be imagined, there are only two possible types of cuts in our data replicated space:

- If the cut happens on one of the original attributes, then the cuts on each replica will be on the same position (see Figure 3(a)).
- If the cut happens on one of the new attributes, then the cut will represent a constant factor on the original space (see Figure 3(b)).

Note that the difficulties of instantiating the DRM with ADABOOST remain true for *any weak learner* that use a single attribute. Since in each iteration a *weak learner* is designed without strong ties with the other *weak learners*, the strong classifier resulting from the boosting process may not possess the desired property of consistency.

In here we propose a soft DRM, where the non intersecting constraint is imposed on each iteration. Although the final strong classifier may not possess that property, intuitively, the final model is biased towards consistent models.

In order to exploit the parallelism constraint that is usually imposed by the Data Replication Method, instead of training one weak classifier, we can train $(K-1)$ weak classifiers (one for each replica), where we force them to use the same attribute but with different thresholds. This will in turn guarantee that our cuts are parallel. This makes our soft DRM a hybrid of the original DRM and the Frank and Hall method.



(a)

(b)

Figure 3: Problems with decision stumps and the data replication model. (a) Cut on one of the new attributes. (b) Cut on one of the original attributes.

We apply this idea to AdaBoost by independently boosting each replica while forcing that, at every iteration, all weak classifiers use the same attribute:

1. As before, we create $(K-1)$ binary replicas of our dataset;
2. Initialize the weights such as the sum of the weights of each replica is 1;
3. Coupled selection of the weak classifier between all replicas;
   (a) For a decision stump, pick the best attribute (possibly with different decision thresholds) $\hat{att}$ based on a combination of the errors of each replica;
4. Calculate the weight of each observation j in each replica k at iteration i;
5. If a replica has an error superior to 50%, stop boosting that replica;
6. The training stops if every replica has an error superior to 50% or after a set number of iterations.

Note that the $(K-1)$ binary AdaBoost are tied only in step 3, which can be considered as part of building the *weak learner*.Also note the error of each weak classifier is computed independently for each replica, resulting also in an independent update of the weights and in independent estimation of the contribution of the binary weak classifier to the final binary strong classifier. Moreover, the boosting process can stop earlier in one replica than in the others. Nothing of this is true for AdaBoost.OR.

When using the decision stump as weak classifier, all $(K-1)$ replicas of AdaBoost are constrained to use the same attribute *but* with (likely) different thresholds. The attribute is chosen to minimize a function (in our case we selected the average but the median or the maximum were also sensible options) of the individual $(K-1)$ errors:

$$\arg\min_{att} \frac{1}{K-1} \sum_{k=1}^{K-1} Err_k(att)$$

The individual error $Err_k(att)$ is the smallest misclassification error (by optimizing the threshold) in replica k when using attribute att.

A more detailed explanation of this algorithm can be seen on Algorithm 1.

4 EXPERIMENTAL STUDY

In order to compare the performance of the various AdaBoost variants, we performed experiments on both artificial and real-world datasets. All variants were implemented on top of *Weka*'s AdaBoost.M1 implementation. The results were obtained by performing 10 experiments using 10-fold cross validation, with the number of iterations limited to 100. The statistical significance analysis was done in *Weka*'s experimenter interface using a corrected paired t-test, with a confidence of 0.05. The error metrics used were the Mean Error Rate (MER) and the Mean Absolute Error (MAE) [2].

4.1 Ordinal Datasets

In our experiments we use two synthetic datasets and six real datasets. A more detailed description of each dataset is presented in Table 1.

The synthetic datasets are the following:

[2] We calculated the absolute error in the following way: If our predicted class is C_p and the correct class is C_c, then the absolute error is $|c-p|$. *Weka* comes with an implementation of MAE for classification that is different from the one we presented and is not suited for ordinal tasks, and therefore should not be used to reproduce our results.

```
Data: A dataset $\mathcal{D} = (D, f)$ with $N$ elements
Result: A classifier $\hat{f}_{\mathcal{F}}$
Replicate the dataset in $K-1$ binary replicas $\mathcal{D}_k = (D_k, f_k)$
Initialize the example weights as $w_k^j := \frac{1}{N}$;
Initialize $active_k = true$; $i := 1$;
while $\neg EndingCondition$ do
    forall the $att \in Attributes$ do
        forall the $k : 0 \leq k < K-1 \wedge active_k$ do
            $\hat{f}_{k,att}^i$ = train($\mathcal{D}_k, att$);
            $error_{k,att}^i = \sum_{j=1}^{N} w_k^j [\![\hat{f}_{k,att}^i(\mathbf{d}_k^j) \neq f(\mathbf{d}_k^j)]\!]$;
        end
    end
    $\hat{att}_i = \arg\min_{att}$ combination($error_{k,att}^i$) ;              // The attribute to split on
    forall the $k : 0 \leq k < K-1 \wedge active_k$ do
        $\hat{f}_k^i = \hat{f}_{k,\hat{att}_i}^i$;
        $error_k^i = error_{k,\hat{att}_i}^i$;
        if $error_k^i > 0.5$ then
            $active_k := false$;                                             // Stop boosting this replica
        else
            $\alpha_k^i = 0.5 log(\frac{error_k^i}{1-error_k^i})$ ;                                // Classifier weight
            forall the $j$ do
                $w_k^j := w_k^j (\alpha_k^i)^{1-[\![\hat{f}_k^i(\mathbf{d}_j) \neq f_k(\mathbf{d}_j)]\!]}$ ;             // Updates the example weights
            end
            $\mathcal{F}_k := \mathcal{F}_k \cup \{\hat{f}_k^i\}$ ;                                 // Binary ensemble for replica $k$
        end
    end
    Normalize the weights so that $\forall k : \sum_j w_k^j = 1$;
    $i := i + 1$
end
forall the $k$ do
    $\hat{f}_{\mathcal{F}_k}(\mathbf{x}) = sign(\sum_i \alpha_k^i \hat{f}_k^i(\mathbf{x}))$ ;                       // Binary classifier for replica $k$
end
$\hat{f}_{\mathcal{F}}$ = getOrdinalClassifier($\hat{f}_{\mathcal{F}_1}, ..., \hat{f}_{\mathcal{F}_{K-1}}$) ;              // Combines the binary results
```

Algorithm 1: oADABOOST with decision stump

Table 1: List of considered datasets. The number of attributes excludes the class attribute and the SWD dataset has an unused label.

Name	Points	Attributes	Labels	Class Distribution
Synthetic 1 (Circle)	1000	2	3	$[523, 413, 63]$
Synthetic 2 (Non-monotonic)	5000	2	5	$[115, 296, 225, 229, 135]$
Arie Ben David ERA	1000	3	9	$[92, 142, 181, 172, 158, 118, 88, 31, 18]$
Arie Ben David ESL	488	4	9	$[2, 12, 38, 100, 116, 135, 62, 19, 4]$
Arie Ben David LEV	1000	4	5	$[93, 280, 403, 197, 27]$
Arie Ben David SWD	1000	10	4(5)	$[32, 352, 399, 217]$
Balance-Scale	625	4	3	$[288, 49, 288]$
BCCT	1144	30	4	$[160, 592, 272, 120]$

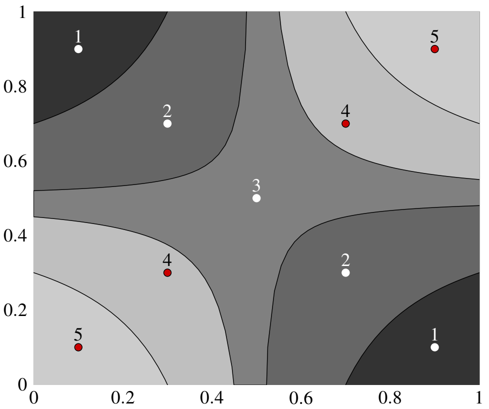


Figure 4: Synthetic dataset.

1. 1000 points $\mathbf{x} = (x_1, x_2)^t$ were randomly generated in the unit square $[0,1] \times [0,1] \in \mathbf{R}$ according to the uniform distribution. The class was assigned to each point according to its distance to $(0.5, 0.5)$: $f(\mathbf{x}) = \lfloor 6 \times ((x_1 - 0.5)^2 + (x_2 - 0.5)^2) \rfloor$
2. 1000 points from a dataset commonly used for evaluating ordinal data methods (Cardoso and da Costa, 2007) (this dataset is also shown on Figure 4).

In the Balance-Scale dataset each example representing a balanced scale tipped to the right, to the left or balanced, based on the relation between the left distance, right distance, left weight and right weight. It is available on the UCI repository (`https://archive.ics.uci.edu/ml/`) The Arie Ben David datasets are available on the MLData Repository (`https://mldata.org/`) and consist of examples classified on a ordinal scale according to subjective judgements (*e.g.* the degree of fitness of a candidate to a certain job). The BCCT dataset, encompassing 1144 observations, expresses the aesthetic evaluation of Breast Cancer Conservative Treatment (Cardoso and Cardoso, 2007; Cardoso and Sousa, 2011). For each patient submitted to BCCT, 30 measurements were recorded, capturing visible skin alterations or changes in breast volume or shape. The aesthetic outcome of the treatment for each and every patient was classified in one of the four categories: Excellent, Good, Fair and Poor.

4.2 Results

In Table 2 we show a comparison of the following AdaBoost variants, instantiated with Decision Stumps, limited to 100 iterations:

oAdaBoost Our AdaBoost variant.

AdaBoost.M1 One of the most common AdaBoost variants with support for multiple classes (Freund et al., 1996).

AdaBoost.M1W A small variant of the AdaBoost.M1 algorithm designed to have better performance on problems with many possible classes (Eibl and Pfeiffer, 2002).

AdaBoost.OR A variant of the AdaBoost.M1 designed for ordinal classification (Lin and Li, 2009). Since it needs an ordinal classifier as the weak learner, we use a decision stump that picks $(K-1)$ cuts on the same attribute.

Frank and Hall (with AdaBoost) The method proposed by Frank and Hall (Frank and Hall, 2001) instantiated with AdaBoost. Note that in this method the $(K-1)$ *weak learners* are independently designed, while in our proposed oAdaBoost method they are coupled.

Results seem to also indicate a slight superiority of AdaBoost.M1W over AdaBoost.M1, more clear in the datasets with more classes. AdaBoost.OR seems superior to AdaBoost.M1W and AdaBoost.M1, suggesting that the integration of the knowledge of the order in the design brings performance advantages. It is also possible to see that oAdaBoost and Frank and Hall methods present the most favorable results when compared to the other boosting algorithms under comparison. The performance difference between our method

Table 2: Comparison of oAdaBoost with AdaBoost variants

(a) Mean Error Rate: mean (standard deviation) of 10 experiments

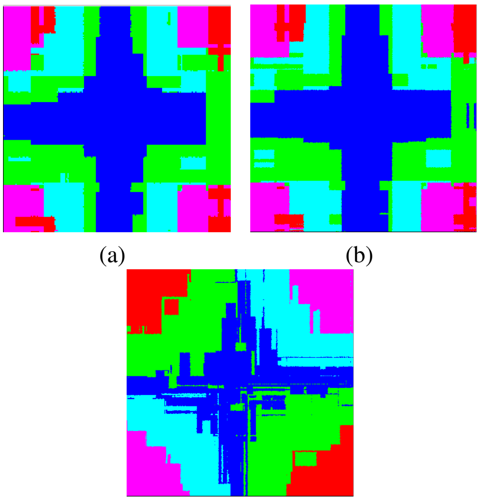
Dataset	oAdaBoost	AdaBoost.M1	AdaBoost.M1W	AdaBoost.OR	Frank and Hall
Circle	0.07(0.03)	0.40(0.03)●	0.55(0.01)●	0.16(0.04)●	0.07(0.03)
Non-mon.	0.66(0.03)	0.70(0.02)●	0.61(0.05)○	0.76(0.02)●	0.50(0.04)○
ERA	0.75(0.04)	0.78(0.02)	0.78(0.04)	0.78(0.02)	0.73(0.05)
ESL	0.33(0.06)	0.57(0.03)●	0.47(0.06)●	0.45(0.05)●	0.33(0.06)
LEV	0.38(0.04)	0.58(0.03)●	0.42(0.05)●	0.50(0.04)●	0.38(0.05)
SWD	0.43(0.05)	0.48(0.04)●	0.48(0.05)●	0.48(0.04)●	0.43(0.05)
Balance	0.03(0.02)	0.28(0.04)●	0.08(0.02)●	0.17(0.08)●	0.04(0.02)
BCCT	0.13(0.03)	0.37(0.03)●	0.38(0.05)●	0.32(0.03)●	0.13(0.03)

(b) Mean Absolute Error: mean (standard deviation) of 10 experiments

Dataset	oAdaBoost	AdaBoost.M1	AdaBoost.M1W	AdaBoost.OR	Frank and Hall
Circle	0.07(0.03)	0.44(0.03)●	0.55(0.01)●	0.16(0.04)●	0.07(0.03)
Non-Mon.	0.99(0.07)	1.30(0.08)●	1.19(0.14)●	1.03(0.04)	1.02(0.03)
ERA	1.24(0.10)	1.43(0.07)●	1.44(0.12)●	1.43(0.07)●	1.34(0.13)●
ESL	0.35(0.07)	0.73(0.06)●	0.56(0.08)●	0.51(0.07)●	0.35(0.07)
LEV	0.41(0.05)	0.71(0.03)●	0.46(0.06)●	0.57(0.05)●	0.42(0.06)
SWD	0.45(0.05)	0.50(0.04)●	0.54(0.06)●	0.50(0.04)●	0.46(0.05)
Balance	0.03(0.02)	0.49(0.09)●	0.08(0.02)●	0.18(0.09)●	0.04(0.02)
BCCT	0.13(0.03)	0.38(0.03)●	0.40(0.07)●	0.33(0.03)●	0.14(0.03)

○, ● statistically significant improvement or degradation

oAdaBoost and Frank and Hall instantiated with AdaBoost seems negligible. However, note that since oAdaBoost is constrained to use the same attribute in all $(K-1)$ *weak learners* (which is not the case in the Frank and Hall method), it results in simpler models without loss of performance.


(a)

(b)

(c)

Figure 5: Boundaries generated by (a) oAdaBoost, (b) Frank and Hall (with AdaBoost) and (c) AdaBoost.OR on our synthetic non-monotonic dataset

Based on the boundaries of the various classifiers trained on our synthetic non-monotonic dataset (Figure 5), it appears that AdaBoost.OR has more problems with noise, while both oAdaBoost and the Frank and Hall method have problems with non-monotonicity.

5 CONCLUSIONS

In this work we have presented a new variant of the well known AdaBoost algorithm designed for ordinal classification. In the proposed methodology, $(K-1)$ binary AdaBoost are built in parallel, tied in phase of designing the *weak learner*. In the end, the $(K-1)$ strong binary classifiers are combined to yield the multiclass model.

Based on our results, it appears that by enforcing local constraints at each boosting iteration (in this case, by enforcing local parallelism) and by working on the replicated space, one can achieve better results on ordinal classification tasks, when compared to other AdaBoost variants instantiated with decision stumps. We plan now to extend these ideas to other learning methodologies whose behaviour is similar to AdaBoost (*i.e.* the final classifier is built from a set of weak classifiers that only use one attribute), such as decision trees (which should have less issues with non-monotonic datasets, since the recursive division of the space should lead to small monotonic cells). We also plan to study the impact of stronger restrictions on the set of weak classifiers (*e.g.* enforce the splits to be ordered), as that should lead to classifiers more similar to the ones generated via the original DRM.

ACKNOWLEDGEMENTS

This work is financed by the ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT within project PTDC/SAU-ENB/114951/2009. The work leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement n FP7–600948. The authors would also like to thank the Masters in Informatics and Computing Engineering and Department of Informatics Engineering of the Faculty of Engineering of the University of Porto for supporting some of the costs of this work.

REFERENCES

Cardoso, J. S. and Cardoso, M. J. (2007). Towards an intelligent medical system for the aesthetic evaluation of breast cancer conservative treatment. *Artificial Intelligence in Medicine*, 40:115–126.

Cardoso, J. S. and da Costa, J. F. P. (2007). Learning to classify ordinal data: the data replication method. *Journal of Machine Learning Research*, 8:1393–1429.

Cardoso, J. S. and Sousa, R. (2010). Classification models with global constraints for ordinal data. In *Proceedings of The Ninth International Conference on Machine Learning and Applications (ICMLA)*, pages 71–77.

Cardoso, J. S. and Sousa, R. (2011). Measuring the performance of ordinal classification. *International Journal of Pattern Recognition and Artificial Intelligence*, 25(8):1173–1195.

Cardoso, J. S., Sousa, R., and Domingues, I. (2012). Ordinal data classification using kernel discriminant analysis: A comparison of three approaches. In *Proceedings of The Eleventh International Conference on Machine Learning and Applications (ICMLA)*, pages 473–477.

Eibl, G. and Pfeiffer, K. P. (2002). How to make adaboost. m1 work for weak base classifiers by changing only one line of the code. In *Machine Learning: ECML 2002*, pages 72–83. Springer.

Frank, E. and Hall, M. (2001). *A simple approach to ordinal classification*. Springer.

Freund, Y., Iyer, R., Schapire, R. E., and Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969.

Freund, Y. and Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, EuroCOLT '95, pages 23–37. Springer-Verlag.

Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156.

Lin, H.-T. and Li, L. (2006). Large-margin thresholded ensembles for ordinal regression: Theory and practice. In Balcázar, J., Long, P., and Stephan, F., editors, *Algorithmic Learning Theory*, volume 4264 of *Lecture Notes in Computer Science*, pages 319–333.

Lin, H.-T. and Li, L. (2009). Combining ordinal preferences by boosting. In *Proceedings ECML/PKDD 2009 Workshop on Preference Learning*, pages 69–83.

Sousa, R. and Cardoso, J. S. (2011). Ensemble of decision trees with global constraints for ordinal classification. In *International Conference on Intelligent Systems Design and Applications (ISDA)*.

Sun, B.-Y., Wang, H.-L., Li, W.-B., Wang, H.-J., Li, J., and Du, Z.-Q. (2014). Constructing and combining orthogonal projection vectors for ordinal regression. *Neural Processing Letters*, pages 1–17.