Calibrating AdaBoost for asymmetric learning

Nikolaos Nikolaou and Gavin Brown

School of Computer Science, University of Manchester,
Kilburn Building, Oxford Road, Manchester, M13 9PL, UK
{nikolaos.nikolaou,gavin.brown}@manchester.ac.uk

Abstract. *Asymmetric classification* problems are characterized by class imbalance or unequal costs for different types of misclassifications. One of the main cited weaknesses of *AdaBoost* is its *perceived* inability to handle asymmetric problems. As a result, a multitude of asymmetric versions of AdaBoost have been proposed, mainly as heuristic modifications to the original algorithm. In this paper we challenge this approach and propose instead handling asymmetric tasks by properly *calibrating* the scores of the original AdaBoost so that they correspond to probability estimates. We then account for the asymmetry using classic decision theoretic approaches. Empirical comparisons of this approach against the most representative asymmetric Adaboost variants show that it compares favorably. Moreover, it retains the theoretical guarantees of the original AdaBoost and it can easily be adjusted to account for changes in class imbalance or costs without need for retraining.

Keywords: Boosting; Cost-sensitive; Class imbalance; Classifier calibration

1 Introduction

Most real world classification problems are *asymmetric*. This asymmetry means that either the classes have different *prior probabilities* or the *costs* of different types of misclassifications are unequal, or both. A doctor testing a patient for a life-threatening disease, is faced with a cost-sensitive decision: a false positive will lead to further tests which will eventually reveal the misdiagnosis, while a false negative can be lethal. An astrophysicist predicting whether a telescope image contains a supernova or not faces an imbalanced class problem, as supernovae are rare.

AdaBoost [4] is a powerful, popular and recognized meta-learning technique. However it is often regarded as *skew-insensitive* [15, 17], meaning it is unable to handle asymmetric tasks. There exist many *skew-sensitive* AdaBoost variants, including *AdaCost* [2, 17], *CSB0*, *CSB1*, *CSB2* [17], *Asymmetric-Adaboost* [18], *RareBoost* [6], *AdaC1*, *AdaC2*, *AdaC3* [16], *CS-AdaBoost* [9, 10]. However, most of them are heuristic and as a result they lack the theoretical guarantees of the original AdaBoost [7].

It is also unclear if we really *need* to modify AdaBoost, or if asymmetric problems are better tackled by *calibrating* the *scores* of the original AdaBoost.

The final publication is available at:
http://link.springer.com/chapter/10.1007/978-3-319-20248-8_10

Calibrated scores can be treated as *probability estimates* and can thus be used to handle the asymmetric nature of the task following *decision theoretic* approaches, similar to the classical work by Elkan [1]. For one, this approach can easily be adjusted to different class and cost imbalance setups, without the need to retrain the ensemble. Another benefit is that all favorable theoretical properties of the original AdaBoost will be preserved. The goal of this work is thus to investigate whether we can achieve comparable results to the asymmetric AdaBoost variants by calibrating the scores produced by the original AdaBoost and then choosing an appropriate classification threshold.

2 Background

2.1 Asymmetric Learning

In this paper we will be examining binary classification asymmetric problems, where an example can be either positive, denoted by a label $y = 1$ or negative, denoted by $y = -1$. The class imbalance can be captured by the different priors, $p(y = -1)$ and $p(y = 1)$, while the cost imbalance can be modeled with a *cost matrix* of the form

$$C = \begin{bmatrix} 0 & c \\ 1 & 0 \end{bmatrix}, \tag{1}$$

where 1 is the cost of a *false positive* and c the cost of a *false negative*[1]. The above matrix assigns a zero cost to all correct classifications, as is commonly the case [1].

Although *skewed class* and *skewed cost* problems are different [8], they can be formulated and treated in a similar way, by using a *skew ratio* c, that captures the *relative importance* of positives w.r.t. negatives to adjust for either [3]. We commonly assume w.l.o.g. that the important class (the 'rare' one in an imbalanced class scenario, or the 'expensive to misclassify' in a cost-sensitive scenario) is the positive one, $y = 1$. One difference is the evaluation measures used in each type of asymmetric problem. When facing a skewed cost task, the main goal is to minimize the total cost. When faced with a skewed class problem, the goal could instead be to achieve good performance on all classes.

Our analysis will focus on cost-sensitive tasks under a cost matrix of the form C, with skew ratio $c = c_{FN}/c_{FP} \geq 1$. This means that the cost of misclassifying the i-th example is

$$c(y_i) = \begin{cases} c, & \text{if } y_i = 1 \\ 1, & \text{if } y_i = -1 \end{cases},$$

[1] A more intuitive equivalent form is $\begin{bmatrix} 0 & c_{FN} \\ c_{FP} & 0 \end{bmatrix}$. Scaling the cost matrix has no effect on the decision problem, so we can divide its entries with c_{FP}, thus assigning a cost of 1 to false positives and a cost of $c = c_{FN}/c_{FP}$ to false negatives.

where y_i is the label of the instance. The primary evaluation measure we will use in this paper is the *average cost* incurred on the test set

$$L_{Avg}(\mathbf{x}'_i, y_i; H, c) = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \mathbb{I}[H(\mathbf{x}'_i) \neq y_i] c(y_i),$$

where $H(\mathbf{x}'_i)$ is the prediction of the ensemble on the test example $\mathbf{x}'_i$ and $\mathbb{I}[\cdot]$ is the *indicator function* which outputs '1' iff its argument is true. We will also use *precision*, i.e. the fraction of positive predictions that are correct and *recall*, the fraction of positives correctly classified, to shed light on the different behaviors of the methods we examine. Precision and recall are given by

$$Prec = \frac{TP}{TP + FP}, \quad Rec = \frac{TP}{TP + FN},$$

where TP, FP and FN are the numbers of *true positives*, *false positives* and *false negatives* on the test set, respectively. Both quantities need to be high for prediction to be reliable. The harmonic mean of precision and recall,

$$F - measure = \frac{2 \cdot Rec \cdot Prec}{Rec + Prec},$$

can be used to capture both precision and recall at once. A high *F-measure* is desirable, as it implies that the values of both precision and recall are high.

2.2 AdaBoost

AdaBoost [4] is an *ensemble learning technique* which constructs a *strong classifier* H sequentially by combining multiple *weak classifiers* h_t, $t = 1, \ldots, M$. A weak classifier is one that is marginally more accurate than random guessing and a strong classifier is one that achieves arbitrarily high accuracy. AdaBoost achieves this by training each subsequent model h_t on a new dataset in which the examples misclassified by the previous model are assigned more weight and the ones that were correctly classified are assigned less weight. This can be achieved either by reweighing or by resampling the dataset on each round. This work uses the reweighing approach and focuses on AdaBoost with *confidence rated predictions* [14], where each *base learner* h_t is assigned a *confidence score* α_t.

The algorithm is given as input a set of training examples of the form $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, N$ where $\mathbf{x}_i$ is the feature vector of the i-th example and y_i is its class label. On the first round of AdaBoost, all training examples are assigned equal weights $D_i^1 = \frac{1}{N}$. On each round $t = 1, \ldots, M$, the weak learner h_t that minimizes the misclassification error $\epsilon_t = \sum_{i:h_t(\mathbf{x}_i) \neq y_i} D_i^t$, where $h_t(\mathbf{x}_i)$ is the predicted class of the i-th example by the t-th weak learner, is added to the ensemble. The confidence of weak learner h_t is computed as

$$\alpha_t = \frac{1}{2} \log \Big(\frac{1 - \epsilon_t}{\epsilon_t} \Big). \tag{2}$$

The weight of each example $i = 1, \ldots, N$ is then updated to

$$D_i^{t+1} = e^{-y_i h_t(\mathbf{x}_i)\alpha_t} D_i^t \tag{3}$$

and renormalized by $D_i^{t+1} \leftarrow \frac{D_i^{t+1}}{\sum_{i=1}^N D_i^{t+1}}$ so that $\sum_{i=1}^N D_i^{t+1} = 1$. These will be the weights of each example on the next round. The algorithm terminates when the maximum number M of weak learners have been added to the ensemble or when a base learner h_t with $\epsilon_t < 1/2$ cannot be found[2]. The *final prediction* on a test datapoint $\mathbf{x}'$ is given by the sign of the weighted sum of the weak learner predictions $h_t(\mathbf{x}')$ *weighted* by their corresponding confidence scores

$$H(\mathbf{x}') = sign\left[\sum_{t=1}^{M} \alpha_t h_t(\mathbf{x}')\right]. \tag{4}$$

2.3 Classifier Calibration

Many classifiers can have their output normalized to return a *score* $s(\mathbf{x}') \in [0, 1]$ for each test example $\mathbf{x}'$ indicating 'how positive' it is. In the case of AdaBoost, this score is the quantity $s(\mathbf{x}') = \frac{\sum_{t=1}^M \alpha_t \frac{h_t(\mathbf{x}')+1}{2}}{\sum_{t=1}^M \alpha_t}$. However, when a cost-sensitive decision needs to be made on the instance $\mathbf{x}'$, the score $s(\mathbf{x}')$ is of little use. Instead, we need to estimate the probability of $\mathbf{x}'$ belonging to the positive class $\hat{p}(y = 1|\mathbf{x}')$. This will allow us to assign $\mathbf{x}'$ to the class that *minimizes the expected cost*. In other words, in binary classification, $\mathbf{x}'$ is assigned to the positive class only if

$$\hat{p}(y = 1|\mathbf{x}')c > \hat{p}(y = -1|\mathbf{x}') \iff \hat{p}(y = 1|\mathbf{x}') > \frac{1}{1+c},$$

under the cost matrix of Eq. (1), making use of $\hat{p}(y = -1|\mathbf{x}') = 1 - \hat{p}(y = 1|\mathbf{x}')$. Otherwise, $\mathbf{x}'$ is assigned to the negative class.

The procedure of converting classifier scores to actual probability estimates is called *calibration*. A classifier is calibrated if $\hat{p}(y = 1|\mathbf{x}') \to s(\mathbf{x}')$, as $N \to \infty$, for any $\mathbf{x}'$ [21]. However, it has been previously noted [11] that as the number of boosting rounds increases, the scores $s(\mathbf{x}')$ get more pushed away from 0 or 1, exhibiting an increasing "sigmoid distortion". In other words, the scores produced by AdaBoost are a sigmoid transformation of actual probability estimates. A theoretical justification for this effect is based on the statistical interpretation of AdaBoost by Friedman et al. [5], under which AdaBoost is a *stagewise* procedure of constructing an *additive logistic regression model* which finds the weak learners h_t and their corresponding confidence scores α_t that minimize the *average exponential loss* across all training examples.

The results of Niculescu-Mizil and Caruana [11] showed empirically that once properly calibrated, AdaBoost produced better probability estimates than any other model examined. The authors corrected for the "sigmoid distortion" of

[2] Note that in the binary classification case, a hypothesis h_t with error $\epsilon_t > 1/2$ can be turned into one with $\epsilon_t < 1/2$ simply by flipping its predictions.

the AdaBoost scores using three different approaches. The first approach was to directly apply a *logistic correction* implied by the framework of Friedman et al. [5]. The second calibration method was *Platt scaling* [12], originally used to map *SVM* outputs[3] to posterior probabilities. Platt scaling consists of finding the parameters A and B for a sigmoid mapping $\hat{p}(y=1|\mathbf{x}') = \frac{1}{1+e^{As(\mathbf{x}')+B}}$, such that the likelihood of the data is maximized. Fitting the parameters A and B requires the use of a separate validation set. Finally, they also performed calibration using *isotonic regression* [13]. The latter is non-parametric and more general as it can be used to calibrate scores which exhibit any form of monotonic distortion [20]. Platt scaling produced the most reliable probability estimates on small sample sizes among the three methods, closely followed by isotonic regression. In this paper we will therefore be calibrating the scores of AdaBoost using Platt scaling. The *Calibrated AdaBoost* algorithm is given in Figure 1.

3 Asymmetric boosting algorithms

In the introduction we mentioned a number of AdaBoost variants proposed to handle asymmetric learning tasks. Most of these methods are proposed heuristically, i.e. by introducing ad-hoc changes to steps of the AdaBoost algorithm, rather than by starting from a cost-sensitive problem formulation, e.g. by defining a different *loss function* in place of AdaBoost's exponential loss. Asymmetric boosting methods can broadly be classified into two groups: those that modify the prediction rule Eq. (4) of AdaBoost and those that introduce modifications in the training phase, either by modifying the weight update rule of Eq. (3), or the calculation of the α_t coefficients of Eq. (2).

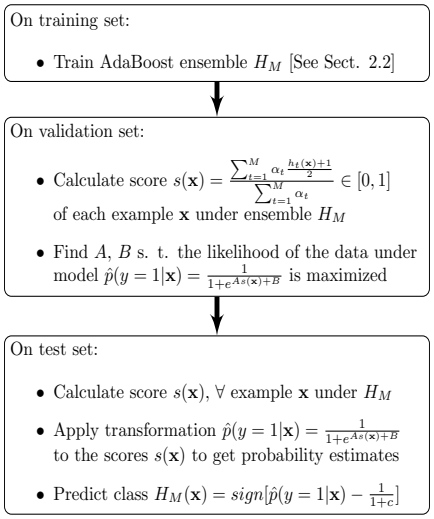

Fig. 1. Calibrated AdaBoost

3.1 Methods that modify the prediction rule

A straightforward way to make AdaBoost skew-sensitive is to substitute the weighted majority vote prediction rule of Eq. (4) with the *minimum expected cost (MEC)* prediction rule

$$H_M(\mathbf{x}') = sign\left[\sum_{y\in\{-1,1\}} c(y) \sum_{t=1}^{M} \alpha_t h_t(\mathbf{x}')\right], \tag{5}$$

[3] The mapping of outputs of SVMs to posterior probability estimates exhibits a similar sigmoid distortion to that observed in AdaBoost.

which reduces to Eq. (4) for $c = 1$, i.e. when the costs of false positives and false negatives are equal. This method trains a standard AdaBoost ensemble and only changes the decision rule used for the final prediction. The idea is to assign $\mathbf{x}'$ to the class that minimizes the expected cost. This approach has been mentioned in [17] without being given a specific name. In this paper we refer to it as *AdaMEC*.

3.2 Methods that modify the training algorithm

CSB2 [17], changes the weight update rule of the original AdaBoost, given in Eq. (3), to

$$D_i^{t+1} = e^{-y_i h_t(\mathbf{x}_i)\alpha_t} C_{\delta(i)} D_i^t, \;\; where \;\; C_{\delta(i)} = \begin{cases} 1, & \text{if } h_t(\mathbf{x}_i) = y_i \\ c(y_i), & \text{if } h_t(\mathbf{x}_i) \neq y_i \end{cases}. \quad (6)$$

The form of the update rule of *CSB2* is the same as that of the original AdaBoost only for correctly classified examples, hence true positives and true negatives are not treated differently. On the other hand, misclassified examples have their weight updates adjusted by an multiplicative cost factor $c(y_i)$, thus false positives and false negatives are treated differently. *CSB2* reduces to AdaBoost for $c = 1$.

AdaC2 [16] substitutes the weight update rule of the original AdaBoost, given in Eq. (3), by

$$D_i^{t+1} = e^{-y_i h_t(\mathbf{x}_i)\alpha_t} c(y_i) D_i^t, \quad (7)$$

which also treats true positives and true negatives differently, unlike Eq. (6). The method also modifies the calculation of the α_t coefficients of Eq. (2) to

$$\alpha_t = \frac{1}{2} \log \frac{\sum_{i:h_t(\mathbf{x}_i)=y_i} D_i^t c(y_i)}{\sum_{i:h_t(\mathbf{x}_i)\neq y_i} D_i^t c(y_i)}. \quad (8)$$

It is worth noting that *AdaC2* can be justified theoretically [15] as a stage-wise minimization of a *cost-weighted* version of the exponential loss, which for a classifier h_t, on an example $(\mathbf{x}_i, y_i)$ has the form $L(h_t(\mathbf{x}_i), y_i) = c(y_i)e^{-y_i h_t(\mathbf{x}_i)}$. Under this definition, the α_t calculated by Eq. (8) is *optimal*. Like the other two variants we described, when $c = 1$, *AdaC2* reduces to AdaBoost.

4 Empirical evaluation

4.1 Experimental setup

In our experiments we compare the performance of *AdaMEC*, *CSB2* and *AdaC2* to that of the *original AdaBoost calibrated with Platt scaling* under various degrees of cost skew, namely $c \in \{1, 1.5, 2, 2.5, 5, 10\}$. As a primary measure of performance we use the average cost attained on the test set. We also provide precision, recall and F-measure results to better demonstrate the different behavior of each method. As a base learner, we used univariate logistic regression

trained with batch gradient descent. The maximum number of base learners M was set to 100.

We used 7 datasets from the *UCI repository*. Any entries with missing values were discarded. Our goal is to investigate the performance of each approach under various degrees of cost skew c. The datasets are originally imbalanced, so we selected an equal number of positive and negative examples, to suppress the additional effects of class imbalance. This was achieved by uniformly undersampling the majority class rather than by oversampling the minority class, as it avoids overfitting due to occurrences of identical examples in training, testing and validation sets. A summary of the datasets is given in Table 1.

We use a random 25% of the data for testing. The remaining 75% was used for training. In the case of calibration using Platt scaling, we needed to also reserve a separate validation set to fit the parameters of the sigmoid without overfitting. A third of the training data was used to this end. After training the models and –where applicable– calibrating on the validation set, we evaluated them on the test set. The entire procedure is repeated 30 times. For each method and evaluation measure, we report average values across all 30 runs as well as 95% confidence intervals.

Table 1. Characteristics of the datasets used in this study. The table indicates the number of instances used, the number of features, and the class we chose to be 'positive' according to the naming convention in the original file. For example, in *semeion*, class '1' was chosen as 'positive' and the rest grouped under the 'negative' label.

Dataset	# Instances	Positive Class	Negative Class	# Features
survival	162	2	1	3
liver	290	1	2	6
pima	576	1	0	8
heart	240	1	0	13
wdbc	424	1	0	31
sonar	194	0	2	60
semeion	322	1	$\{2, ..., 10\}$	256

4.2 Analysis of experimental results

Average Cost: In terms of average cost, we observe different trends on the lower-dimensional datasets *survival*, *liver* and *pima* and on the higher-dimensional datasets, *wdbc*, *heart*, *semeion* and *sonar*. Results for *liver*, *pima* and *sonar* are omitted due to lack of space. When the problem is cost-insensitive ($c = 1$), all methods exhibit more or less the same performance. The performance of most methods also tends to be equivalent when the cost ratio is very high ($c = 10$), since for such high degrees of imbalance all examples tend to be assigned to the positive class. Our results are summarized in Figure 2.

On low-dimensional datasets, the performance of *calibrated AdaBoost* is on par with that of *CSB2* and *AdaC2* and all three methods clearly outperform

AdaMEC, which exhibits a high variance as c increases. This can partly be explained by the fact that *AdaMEC* used on average a much smaller number of weak learners than *CSB2* and *AdaC2*. On the other hand, *calibrated AdaBoost* tended to slightly fewer weak learners than *AdaMEC*. So its improved performance over *AdaMEC* can only be attributed to the calculation of more reliable probability estimates.

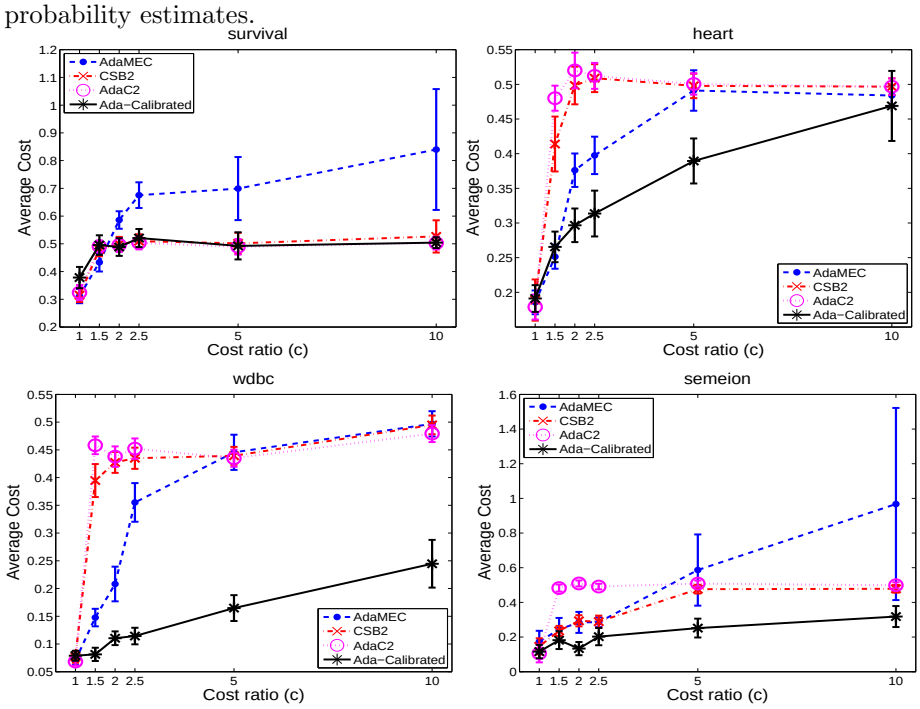

Fig. 2. Average cost results under various degrees of cost imbalance c. The cost attained by *calibrated AdaBoost* is lower than that of *AdaMEC*, *CSB2* and *AdaC2* on higher dimensional datasets like *heart*, *wdbc* and *semeion* and comparable on lower dimensional datasets like *survival*.

On higher-dimensional datasets, the performance of *calibrated AdaBoost* is even more impressive, as it clearly outperforms all other methods. *AdaMEC* exhibits the second-best performance at low degrees of skew c. *CSB2* and *AdaC2* produce the highest average cost with the former producing marginally lower average cost than the latter for low values of c.

Precision and Recall: The precision and recall curves reveal more details about the different behaviour of each method. *Calibrated AdaBoost* has the overall highest precision scores, even for high degrees of cost skew c. *AdaMEC* achieves the second-best overall precision. *CSB2* typically gives poor precision values, but on *semeion*, *heart* and marginally on *wdbc*, it still outperforms *AdaC2* for low values of c. We can again notice the two different trends regarding the *calibrated AdaBoost*. It does not outperform its competitors on the low-dimensional datasets *survival*, *pima* and *liver*, but its performance is comparable to theirs.

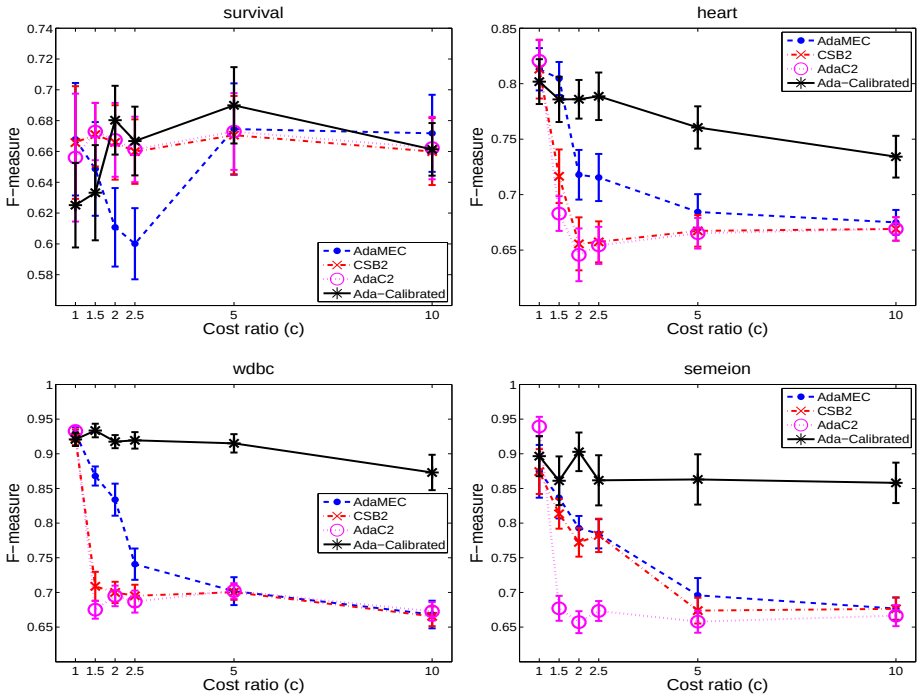

Fig. 3. F-measure results under various degrees of cost imbalance c. *Calibrated AdaBoost* produces higher F-measure scores than its competitors on the higher dimensional datasets *heart*, *wdbc* and *semeion* and comparable on the lower dimensional datasets like *survival*. It also shows a remarkable robustness to changes in c.

In terms of recall, all methods exhibit very high scores, close to the maximal value of 1. This is indicative of the cost-sensitive methods' eagerness to assign test instances to the positive class. *AdaC2* and *CSB2* have the overall highest recall, reaching the value of 1 even for small values of c. *AdaMEC* has the second-highest overall scores and *calibrated AdaBoost* exhibits the lowest recall values among the compared methods. This does not mean that *calibrated AdaBoost* behaves poorly in terms of recall, just that it is less *aggressive* than its competitors.

These results indicate that *AdaC2* is the most aggressive among the compared methods, as it tends to assign all test examples to the positive class even for relatively low values of c. This leads to zero false negatives, hence maximal recall, but also to many false positives, hence low precision. This behaviour of *AdaC2* is largely mimicked by *CSB2*. The next most aggressive method is *AdaMEC* and the least aggressive is calibrated AdaBoost. The results are summarized in Figure 3.

F-measure: On the F-measure curves of Figure 3, we can again observe that on the low-dimensional datasets *survival*, *pima* and *liver* all methods exhibit comparable performance. This is not surprising, as problems with small numbers of features are generally easier than high-dimensional ones. Where the *calibrated*

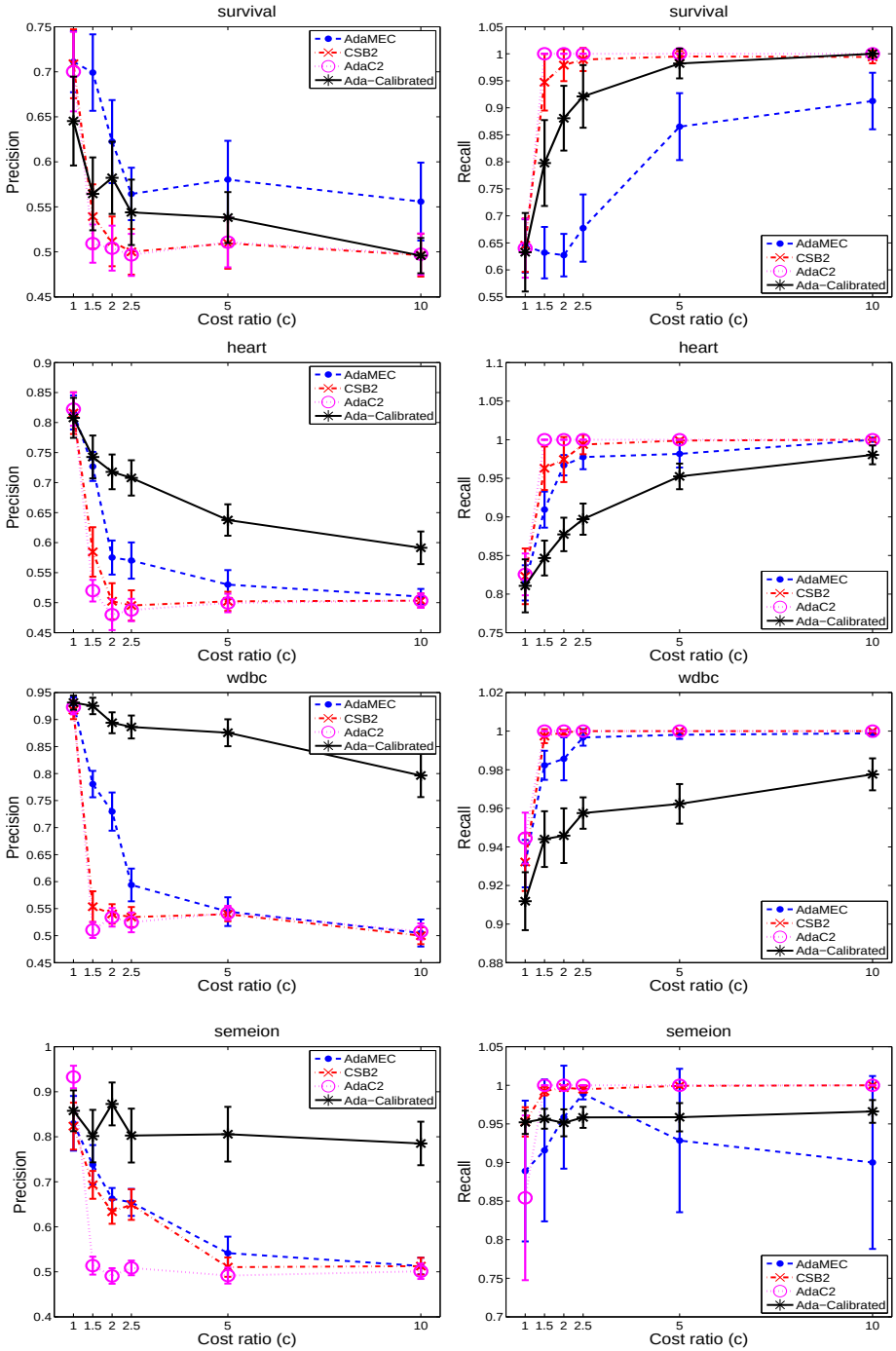

Fig. 4. Precision and recall results under various degrees of cost imbalance c. *Calibrated AdaBoost* achieves higher precision and lower recall than its competitors, especially on the higher dimensional datasets *heart*, *wdbc* and *semeion*.

AdaBoost shines is on the higher dimensional datasets *wdbc*, *heart*, *semeion* and *sonar*. On these datasets it attains F-measure values far higher than the asymmetric AdaBoost variants, exhibiting an admirably small sensitivity to the cost ratio c. The F-measure values of *AdaMEC* are the second highest overall, with *CSB2* and *AdaC2* having low scores. Of these two, *CSB2* has a slightly higher F-measure overall.

5 Discussion and conclusion

Calibration, the act of adjusting the scores of classifiers so that they correspond to reliable probability estimates, is an often overlooked aspect of classification. We can use it to improve classification performance, especially in asymmetric situations. In the case of boosting, the results we obtained clearly show that *calibrated AdaBoost* can be used as a viable alternative to asymmetric versions of AdaBoost. In our experiments, we found that *calibrated AdaBoost* outperforms the asymmetric AdaBoost variants on datasets with large numbers of features while it performs comparably on datasets with few features. Furthermore, the theoretical properties of the original AdaBoost are preserved. Finally, we can also easily adjust our predictions without the need to retrain the model. This is also true for AdaBoost variants that modify only the prediction rule (*AdaMEC*), but not for those that modify the training phase (*CSB2*, *AdaC2*).

This study was limited to comparing *AdaMEC*, *CSB2* and *AdaC2* to *calibrated AdaBoost*, by virtue of being the most successful representatives of their families. As for the other variants, they are all methods that modify the training algorithm. *CSB0* and *CSB1* [17] do not use confidence rated predictions and based on the results of comparative studies [9, 10, 15], the two variants are typically dominated by *CSB2*. *Asymmetric-Adaboost* [18] was excluded from said studies as being similar to *CSB2*. AdaCost [2,17] is also outperformed by *AdaC2* and *CSB2* and so is *AdaC3* [16]. *CS-AdaBoost* [9, 10], despite being the only method other than *AdaC2* with a solid theoretical basis, has been characterized as 'time-consuming and imprecise' [19], as it lacks a closed form solution for α_t and the optimization of its parameters is therefore computationally intensive.

To our knowledge, the only previous attempt at directly comparing asymmetric AdaBoost variants to *calibrated AdaBoost* was by Masnadi-Shirazi and Vasconselos [10]. The comparison was performed on imbalanced data, it included *AdaC2* and *CSB2* and the performance of *calibrated AdaBoost* was found to be slightly inferior to theirs. However the authors were solving a quite different problem from the one we do in the present paper. They *fixed the desired precision* of their ensembles and based on that they chose the appropriate cost setup for each method (i.e the cost ratio c *differed* from one method to the other) so as to minimize the total number of errors on the test set.

On the other hand, we solve a cost-sensitive problem. We therefore use a *fixed* cost ratio c, taken directly from the cost matrix of the problem and our goal is to minimize the average cost of misclassifications. Our findings and those of Masnadi-Shirazi and Vasconselos are complementary, not contradictory. We observed that *AdaC2* and *CSB2* favor more aggressively the positive class compared to *calibrated AdaBoost*. Masnadi-Shirazi and Vasconselos, by fixing the

precision to a high value, allow the ensemble to commit only a small number of false positives. *AdaC2* and *CSB2* are thus forced to select c values that limit their aggressiveness. Under this light, asymmetric AdaBoost variants can outperform *calibrated AdaBoost* on imbalanced data, if costs are allowed to vary, but with fixed costs, *calibrated AdaBoost* produces lower average costs.

Acknowledgments. This work was supported by EPSRC grant [EP/I028099/1]. We also thank Peter Flach for suggesting the idea that inspired this paper.

References

1. C. Elkan. The foundations of cost-sensitive learning. In *IJCAI*, 2001.
2. W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan. Adacost: Misclassification cost-sensitive boosting. In *ICML*, pages 97–105, 1999.
3. P. A. Flach. The geometry of roc space: understanding machine learning metrics through roc isometrics. In *AAAI*, pages 194–201, 2003.
4. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comp. Syst. Sci.*, 55 (1):119–139, 1997.
5. J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:337–407, 2000.
6. M. V. Joshi, V. Kumar, and R. C. Agarwal. Evaluating boosting algorithms to classify rare classes: Comparison and improvements., 2001.
7. I. Landesa-Vázquez and J. L. Alba-Castro. Shedding light on the asymmetric learning capability of adaboost. *Pat. Rec. Letters*, 33 (3):247–255, 2012.
8. M. A. Maloof. Learning when data sets are imbalanced and when costs are unequal and unknown. In *ICML*, 2003.
9. H. Masnadi-Shirazi and N. Vasconcelos. Asymmetric boosting. In *ICML*, pages 609–619, 2007.
10. H. Masnadi-Shirazi and N. Vasconcelos. Cost-sensitive boosting. *IEEE Trans. Pat. Anal. Mach. Intell.*, 33 (2):294–309, 2011.
11. A. Niculescu-Mizil and R. Caruana. Obtaining calibrated probabilities from boosting. In *UAI*, 2005.
12. J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.
13. T. Robertson, F. Wright, and R. Dykstra. *Order Restricted Statistical Inference*. Probability and Statistics Series. Wiley, 1988.
14. R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37 (3):297–336, 1999.
15. Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang. Cost-sensitive boosting for classification of imbalanced data. *Pat. Recogn.*, 40 (12):3358–3378, 2007.
16. Y. Sun, A. K. C. Wong, and Y. Wang. Parameter inference of cost-sensitive boosting algorithms. In *MLDM*, pages 21–30, 2005.
17. K. M. Ting. A comparative study of cost-sensitive boosting algorithms. In *ICML*, pages 983–990, 2000.
18. P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In *NIPS*, 2002.
19. Z. Wang, C. Fang, and X. Ding. Asymmetric real adaboost. In *ICPR*, 2008.
20. B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *ICML*, pages 609–616, 2001.
21. B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates, 2002.