Accepted in *Machine Learning* 104(2), 2016. Final publication available at:
http://link.springer.com/article/10.1007/s10994-016-5572-x

Cost-sensitive boosting algorithms: Do we really need them?

Nikolaos Nikolaou · Narayanan Edakunni · Meelis Kull · Peter Flach · Gavin Brown

Received: date / Accepted: date

Abstract We provide a unifying perspective for two decades of work on cost-sensitive Boosting algorithms. When analyzing the literature 1997-2016, we find 15 distinct cost-sensitive variants of the original algorithm; each of these has its own motivation and claims to superiority — so who should we believe? In this work we critique the Boosting literature using *four* theoretical frameworks: Bayesian decision theory, the functional gradient descent view, margin theory, and probabilistic modelling. Our finding is that only *three* algorithms are fully supported – and the probabilistic model view suggests that all require their outputs to be *calibrated* for best performance. Experiments on 18 datasets across 21 degrees of imbalance support the hypothesis – showing that once calibrated, they perform equivalently, and outperform all others. Our final recommendation – based on simplicity, flexibility and performance – is to use the *original* Adaboost algorithm with a shifted decision threshold and *calibrated* probability estimates.

Keywords Boosting · Cost-sensitive · Class imbalance · Classifier calibration

1 Introduction

Cost-sensitive prediction tasks are everywhere in real life applications – e.g. medical applications where *false positives* are dangerous, or rare classes in astrophysical data where a *false negative* can mean missing a key scientific observation. *Ensemble* learning algorithms are equivalently ubiquitous in the Machine Learning literature, being a key principle in the winning entries of every major ML competition [2]. The *Adaboost* algorithm [9] stands out in the field of ensemble learning – named

N. Nikolaou, N. Edakunni and G. Brown
School of Computer Science, University of Manchester,
Kilburn Building, Oxford Road, Manchester, M13 9PL, UK.
E-mail: [nikolaos.nikolaou, gavin.brown]@manchester.ac.uk, narayanan.unny@gmail.com

M. Kull and P. Flach
Department of Computer Science, University of Bristol,
The Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, UK.
E-mail: [meelis.kull, peter.flach]@bristol.ac.uk

in a community survey as one of the top ten algorithms in data mining [36], whilst also having a rich theoretical depth, winning the 2003 Gödel prize for the authors. It is no surprise therefore, that significant international research effort has been dedicated to adapting Adaboost for cost sensitive tasks. At the time of writing this article, we identify 15 distinct variants proposed in a sequence of papers [7, 13, 14, 17, 18, 31, 32, 33, 35] published 1997-2016. Each of these is derived from different underlying perspectives, principles or assumptions, and each makes its own claim to superiority in some sense – either empirically, theoretically, or pragmatically.

In this work we analyse the literature, using tools from *four* theoretical frameworks: decision theory, functional gradient descent, margin theory, and probabilistic modelling. Each one of these will turn out to have its own advantages and perspective in the analysis, and the work could not be complete without all four.

The functional gradient descent view ensures the steps of the algorithm are consistent with greedy minimisation of a well defined loss function which is itself a function of the margin, thus ensuring an efficient path toward good generalisation properties. Analysis of the loss functions from the perspective of margin theory ensures the generalisation behaviour is in line with the defined cost-sensitive problem. The decision theoretic view ensures that the predictions generated by boosting are used in a way that is consistent with the goal of minimizing the expected cost of future classifications. A probabilistic analysis shows that the models generate *uncalibrated* outputs, which we proceed to remedy; key to this argument is rephrasing Adaboost as a *Product of Experts (PoE)*. Different cost-sensitive boosting variants translate to different PoE models but they all suffer from the same systematic distortion of the resulting probability estimates due to their PoE nature.

The paper is structured as follows. In Section 2 we introduce the problem of *asymmetric* learning and the basic Adaboost algorithm. We will also briefly introduce the variety of cost sensitive Boosting variants in the literature, and consolidate them into a common notation and terminology. Section 3 is our main contribution, identifying desirable and undesirable properties of the algorithms from the perspective of decision theory, margin theory and functional gradient descent and probabilistic modelling. Section 4 discusses the calibration of probability estimates. Section 5 provides a large scale experimental study to test our hypotheses – 18 datasets, 15 variants of boosting, across 21 different degrees of class imbalance. Finally, Section 6 draws together the conclusions of the work, identifying directions for future work.

2 Background

In the current section we introduce the terminology and notation behind the two key concepts of the paper – learning in *asymmetric* tasks, and the ensemble learning principle known as *Boosting*.

2.1 Asymmetric Learning

Throughout this work we will be focusing on binary classification, for ease of exposition and clarity. Extension to the multiclass case is often handled by breaking

down the problem into multiple binary ones, so our analysis and its results can carry over to the multiclass case. In a binary classification task, 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 *class priors*: $\pi_- = N_-/N$ and $\pi_+ = N_+/N$, where N_+ is the number of positive training examples, N_- is the number of negative training examples and N the size of the training set. The cost imbalance can be modelled with a *cost matrix* of the form

$$C = \begin{bmatrix} 0 & c_{FN} \\ c_{FP} & 0 \end{bmatrix}, \tag{1}$$

where $c_{FP} > 0$ is the cost of a *false positive* and $c_{FN} > 0$ the cost of a *false negative*. The above matrix assigns a zero cost ($c_{TP} = c_{TN} = 0$) to all correct classifications –*true positives* and *true negatives*– as is common practice [6]. Although *skewed class* and *skewed cost* problems are seen as different, they can be understood in a similar way [6, 8]. For example, suppose that the false positive cost c_{FP} associated with misclassifying negatives is twice the false negative cost c_{FN}. This can be simulated by an adjusted class prior which duplicates every negative, leading to $\pi'_- = \frac{2 \cdot N_-}{N_+ + 2 \cdot N_-} = \frac{(2/3) \cdot N_-}{(1/3) \cdot N_+ + (2/3) \cdot N_-}$ and $\pi'_+ = \frac{N_+}{N_+ + 2 \cdot N_-} = \frac{(1/3) \cdot N_+}{(1/3) \cdot N_+ + (2/3) \cdot N_-}$. More generally, if we define

$$c = \frac{c_{FP}}{c_{FP} + c_{FN}}, \tag{2}$$

then this can be simulated by an adjusted class distribution $\pi'_- = \frac{c \cdot N_-}{(1-c) \cdot N_+ + c \cdot N_-}$ and $\pi'_+ = \frac{(1-c) \cdot N_+}{(1-c) \cdot N_+ + c \cdot N_-}$. Hence we can focus on skewed *cost* problems in this paper without loss of generality. Our analysis will focus on tasks with a fixed cost matrix of the form C, under which, the cost of misclassifying the i-th example only depends on its class label y_i – what Landesa-Vázquez and Alba-Castro [15] refer to as *class-level asymmetry*– and is given by

$$c(y_i) = \begin{cases} c_{FN}, & \text{if } y_i = 1 \\ c_{FP}, & \text{if } y_i = -1. \end{cases} \tag{3}$$

Given the probability $p(y = 1|\mathbf{x})$, we should assign $y = 1$ iff the expected cost of a positive prediction, i.e. $c_{TP} \cdot p(y = 1|\mathbf{x}) + c_{FP} \cdot p(y = -1|\mathbf{x})$ is lower than that of a negative prediction, $c_{TN} \cdot p(y = -1|\mathbf{x}) + c_{FN} \cdot p(y = 1|\mathbf{x})$. Under the cost matrix above, $c_{TP} = c_{TN} = 0$, so we assign $\mathbf{x}$ to $y = 1$ iff :

$$\begin{aligned} & p(y = 1|\mathbf{x}) \cdot c_{FN} > p(y = -1|\mathbf{x}) \cdot c_{FP} \iff \\ & p(y = 1|\mathbf{x}) \cdot c_{FN} > (1 - p(y = 1|\mathbf{x})) \cdot c_{FP} \iff \\ & p(y = 1|\mathbf{x}) \cdot (c_{FN} + c_{FP}) > c_{FP} \iff \\ & p(y = 1|\mathbf{x}) > c, \end{aligned} \tag{4}$$

where we made use of $p(y = -1|\mathbf{x}) = 1 - p(y = 1|\mathbf{x})$ and Eq. (2). Otherwise, $\mathbf{x}$ is assigned to the negative class – when $c_{FP} = c_{FN}$ this reduces to the familiar threshold of 0.5. This all follows straightforwardly from Bayesian decision theory and simply translates to shifting the threshold from 0.5 to c to account for the costs. The output of most learning algorithms can be used to *estimate* the probability $p(y = 1|\mathbf{x})$, so the decision rule of Eq. (4) in practice uses estimates $\hat{p}(y = 1|\mathbf{x})$. A key point here, which will play a major role in the current paper, is that 'good'

probability estimates are not always straightforward to obtain from a classifier, even if it has high classification accuracy – such estimates are referred to as *calibrated*, and will be discussed in more detail later.

2.2 AdaBoost

AdaBoost [9] is an *ensemble learning technique* which constructs a *strong classifier* H from a weighted vote of multiple *weak classifiers* h_t, $t = 1, ..., M$. A weak classifier is one that is marginally more accurate than random guessing and a strong classifier is one that achieves error rates arbitrarily close to the irreducible Bayes error rate. The idea is to train each subsequent model on a new dataset in which the weights of examples misclassified by the previous model are increased and the weights of the correctly classified instances are decreased. This can be achieved either by reweighting or by resampling the dataset on each round. In this work we followed the reweighting approach as some evidence exists that it works better in practice [24]. We use the version of AdaBoost that employs *confidence rated predictions* [30], where each *base learner* is assigned a *confidence coefficient* α_t. The lower the weighted error of the learner, the higher its α_t and the larger its contribution to the final decision.

The algorithm is given N training examples of the form $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \{-1, +1\}$ and a maximum number of rounds M. On the first round of AdaBoost, all training examples are assigned equal weights $D_i^1 = \frac{1}{N}$. On each round t, we learn a model h_t to minimize the weighted misclassification error $\epsilon_t = \sum_{i:h_t(\mathbf{x}_i) \neq y_i} D_i^t$, and add it to the ensemble, with a voting weight

$$\alpha_t = \frac{1}{2} \log \Big(\frac{1 - \epsilon_t}{\epsilon_t} \Big). \tag{5}$$

The distribution D^t is then updated for timestep $t + 1$ as

$$D_i^{t+1} \propto e^{-y_i \alpha_t h_t(\mathbf{x}_i)} \times D_i^t, \tag{6}$$

and normalised by $Z_t = \sum_{j=1}^{N} e^{-y_j \alpha_t h_t(\mathbf{x}_j)} D_j^t$ to make D^{t+1} a valid distribution.

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 with $\epsilon_t < 1/2$ cannot be found[1]. The *final prediction* on a test datapoint $\mathbf{x}$ is given by the sign of the sum of the weak learner predictions $h_t(\mathbf{x})$ *weighted* by their corresponding confidence coefficients

$$H(\mathbf{x}) = sign \left[\sum_{t=1}^{M} \alpha_t h_t(\mathbf{x}) \right]. \tag{7}$$

In a seminal paper [10], Friedman et al. interpreted the original AdaBoost algorithm as a procedure to minimize the expected exponential loss:

$$L_{Ada}(F_t) = \mathbb{E}_{\mathbf{x},y}[e^{-yF_t(\mathbf{x})}], \tag{8}$$

[1] 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.

by iterative fitting of terms in the additive model $F_t = \sum_{\tau=1}^{t} \alpha_\tau h_\tau(\mathbf{x})$.

Friedman et al. showed that if we minimize $L_{Ada}(F_t)$ in a *greedy stagewise* manner –i.e. if at stage t we choose the optimal h_t and α_t considering all h_τ and α_τ, for $\tau < t$ as constants– we naturally derive the steps of AdaBoost described above. This was generalized by Mason et al. [19] to show that *functional gradient descent* derives the same steps, but allows us to choose a different loss function if we wish.

2.3 Cost-sensitive Boosting Algorithms

Adaboost iteratively constructs an ensemble model by weighted combination of a sequence of base learners. Each base learner is guided by a weighted distribution over training examples, that leads it to focus on the mistakes of its predecessors. To make this *cost-sensitive* there are, in general, two strategies adopted.

Strategy 1: Modifying the model learnt from data.
Strategy 1 includes methods that change the training phase of the algorithm. They do so either by changing how the weights of training examples are updated alone or in conjunction to changing the α_t weights in a class-dependent manner. This can be achieved by specifying an alternative loss function to Eq. (8), and deriving what the corresponding weight update and α_t should be. However, we could also simply manually modify the update rule without regard of a loss function, but in a way that makes sense in an application-dependent situation.

Tables 1 & 2 summarize the changes in cost sensitive Boosting literature spanning 1997-2016, with numerous innovative class-dependent solutions to this challenge[2]. Table 1 gives the proposed weight update rule for each method and the corresponding entry in Table 2 gives the proposed formula for α_t for the same method. All of the methods presented here, with the exception of CSB0/CSB1 and AdaCost/AdaCost(β_2), reduce to the original AdaBoost when the task is symmetric, i.e. when $c_{FP} = c_{FN} = 1$.

Strategy 2: Modifying how the model is used to make a decision.
Ting [33] proposes an appealing alternative, to train with the *original* Adaboost, but modify the decision rule in a cost-respecting decision-theoretic manner. This is the ADAboost with Minimum Expected Cost (AdaMEC) rule:

$$H_{AdaMEC}(\mathbf{x}) = sign\left[\sum_{y\in\{-1,1\}} c(y) \sum_{\tau:h_\tau(\mathbf{x})=y} \alpha_\tau h_\tau(\mathbf{x})\right]. \tag{9}$$

Eq. (9) reduces to the original AdaBoost decision rule of Eq. (7) when the task is symmetric. AdaMEC exploits Bayesian decision theory – assuming the weighted votes are proportional to class probabilities – that is, $\sum_{\tau:h(\mathbf{x})=1} \alpha_\tau \propto p(y=1|\mathbf{x})$ and $\sum_{\tau:h(\mathbf{x})=-1} \alpha_\tau \propto p(y=-1|\mathbf{x})$. This will be expanded upon in later sections,

[2]Minor variations in e.g. the weight initialization of different approaches have also been examined [15,33] further increasing the number of variants used in practice. These are excluded from our analysis as, for reasons that will become clear in the subsequent sections, these changes are not sufficient to grant any of the favourable missing properties to a variant.

but for now, we have completed our survey of the existing cost sensitive boosting literature, and will proceed to engage in analyzing it from different theoretical frameworks in the next section.

Table 1: A summary of cost-sensitive variants showing how they modify the weight updates. The original Adaboost is included for reference. The definition of α_t also varies across methods and can be looked up in Table 2.

Algorithm	**Weight Update Rule** $D_i^{t+1} \propto [...] \times D_i^t$	**Initial Weights** D_i^1 **& Cost Adjustment Functions**
Adaboost [29]	$e^{-y_i \alpha_t h_t(\mathbf{x}_i)}$	here $D_i^1 = 1/N$
CGAda [13]	$e^{-y_i \alpha_t h_t(\mathbf{x}_i)}$	where $D_i^1 = c(y_i)$
AdaC1 [31] CSAda [17] AdaDB [14]	$e^{-c(y_i) y_i \alpha_t h_t(\mathbf{x}_i)}$	where $D_i^1 = c(y_i)$
AdaC2 [32]	$c(y_i) e^{-y_i \alpha_t h_t(\mathbf{x}_i)}$	where $D_i^1 = c(y_i)$
AdaC3 [32]	$c(y_i) e^{-c(y_i) y_i \alpha_t h_t(\mathbf{x}_i)}$	where $D_i^1 = c(y_i)$
AsymAda [35]	$c(y_i)^{1/M} e^{-y_i \alpha_t h_t(\mathbf{x}_i)}$	where $D_i^1 = c(y_i)^{1/M}$ (fixed M)
CSB0 [33] CSB1 [33] CSB2 [33]	γ_t^i $\gamma_t^i e^{-y_i h_t(\mathbf{x}_i)}$ $\gamma_t^i e^{-y_i \alpha_t h_t(\mathbf{x}_i)}$	where $D_i^1 = c(y_i)$ and $\gamma_t^i = \begin{cases} c(y_i), & \text{if } h_t(\mathbf{x}_i) \neq y_i \\ 1, & \text{if } h_t(\mathbf{x}_i) = y_i \end{cases}$
AdaCost [7] AdaCost(β_2) [33]	$e^{-\beta_t^i y_i \alpha_t h_t(\mathbf{x}_i)}$	where $D_i^1 = c(y_i)$ and $\beta_t^i = \begin{cases} \frac{1+c(y_i)}{2}, & \text{if } h_t(\mathbf{x}_i) \neq y_i \\ \frac{1-c(y_i)}{2}, & \text{if } h_t(\mathbf{x}_i) = y_i \end{cases}$

3 Examining the Literature from Different Views

In this section we use tools from *four* theoretical frameworks: decision theory, functional gradient descent, margin theory, and probabilistic modelling. This is useful to identify properties of the methods, as well as interesting in its own right, since it reflects the diversity of positions from which Adaboost can be derived [28].

3.1 The Functional Gradient Descent View

We first present the view of boosting as functional gradient descent (FGD) [10,19], following Mason et al's formulation. This perspective views boosting as a procedure

Table 2: A summary of cost-sensitive variants for base learner weights α_t.

Algorithm	**Base learner weight** α_t
Adaboost [29] AdaCost(β_2) [33] CSB(0,1,2) [33] CGAda [13]	$\alpha_t = \frac{1}{2}\log\frac{\sum_{i:h_t(\mathbf{x}_i)=y_i} D_i^t}{\sum_{i:h_t(\mathbf{x}_i)\neq y_i} D_i^t}$
AdaC1 [32]	$\alpha_t = \frac{1}{2}\log\frac{1+\sum_{i:h_t(\mathbf{x}_i)=y_i} D_i^t c(y_i)-\sum_{i:h_t(\mathbf{x}_i)\neq y_i} D_i^t c(y_i)}{1-\sum_{i:h_t(\mathbf{x}_i)=y_i} D_i^t c(y_i)+\sum_{i:h_t(\mathbf{x}_i)\neq y_i} D_i^t c(y_i)}$
AdaC2 [32]	$\alpha_t = \frac{1}{2}\log\frac{\sum_{i:h_t(\mathbf{x}_i)=y_i} D_i^t c(y_i)}{\sum_{i:h_t(\mathbf{x}_i)\neq y_i} D_i^t c(y_i)}$
AdaC3 [32]	$\alpha_t = \frac{1}{2}\log\frac{\sum_{i=1}^N D_i^t c(y_i)+\sum_{i:h_t(\mathbf{x}_i)=y_i} D_i^t c(y_i)^2-\sum_{i:h_t(\mathbf{x}_i)\neq y_i} D_i^t c(y_i)^2}{\sum_{i=1}^N D_i^t c(y_i)-\sum_{i:h_t(\mathbf{x}_i)=y_i} D_i^t c(y_i)^2+\sum_{i:h_t(\mathbf{x}_i)\neq y_i} D_i^t c(y_i)^2}$
AsymAda [35]	$\alpha_t = \frac{1}{2}\log\frac{\sum_{i:h_t(\mathbf{x}_i)=y_i} D_i^t c(y_i)^{1/M}}{\sum_{i:h_t(\mathbf{x}_i)\neq y_i} D_i^t c(y_i)^{1/M}}$ (fixed M)
AdaCost [7]	$\alpha_t = \frac{1}{2}\log\frac{1+\sum_{i:h_t(\mathbf{x}_i)=y_i} D_i^t \beta_t^i-\sum_{i:h_t(\mathbf{x}_i)\neq y_i} D_i^t \beta_t^i}{1-\sum_{i:h_t(\mathbf{x}_i)=y_i} D_i^t \beta_t^i+\sum_{i:h_t(\mathbf{x}_i)\neq y_i} D_i^t \beta_t^i}$ See β_t^i in Table 1.
CSAda [17]	Numerical solution of hyperbolic equation. No closed form.
AdaDB [14]	Numerical solution of polynomial equation. No closed form.

that greedily fits an additive model $F_t(\mathbf{x}) = \sum_{\tau=1}^t \alpha_\tau h_\tau(\mathbf{x})$ by minimizing the *empirical risk* (i.e the average loss on a training set),

$$J(F_t(\mathbf{x})) = \frac{1}{N}\sum_{i=1}^N L(y_i F_t(\mathbf{x}_i)), \tag{10}$$

where $L(y_i F_t(\mathbf{x}_i))$ is a *monotonically decreasing* loss function of $y_i F_t(\mathbf{x}_i)$, the *margin* on the i-th example.
The FGD approach tells us to add the model h_{t+1} that most rapidly reduces Eq. (10). This model turns out to be that which minimizes the weighted error for a new weight distribution D^{t+1}, which can be written, as Mason et al. observed, in terms of the functional derivative:

$$D_i^{t+1} = \frac{\frac{\partial}{\partial y_i F_t(\mathbf{x}_i)} J(F_t(\mathbf{x}))}{\sum_{j=1}^N \frac{\partial}{\partial y_j F_t(\mathbf{x}_j)} J(F_t(\mathbf{x}))} = \frac{\frac{\partial}{\partial y_i F_t(\mathbf{x}_i)} L(y_i F_t(\mathbf{x}_i))}{\sum_{j=1}^N \frac{\partial}{\partial y_j F_t(\mathbf{x}_j)} L(y_j F_t(\mathbf{x}_j))}. \tag{11}$$

The voting weight α_t is the step size in the direction of the new weak model h_t, such that it minimizes the empirical risk on the training set, i.e.

$$\alpha_t^* = \arg\min_{\alpha_t} \Big[\frac{1}{N}\sum_{i=1}^{N} L\Big(y_i(F_{t-1}(\mathbf{x}_i) + \alpha_t h_t(\mathbf{x}_i))\Big)\Big]. \tag{12}$$

Under Eq. (11), a given loss function $L(y_i F_t(\mathbf{x}_i))$ implies a specific form of weight updates D_i^{t+1} and conversely the weight updates D_i^{t+1} imply a specific family of equivalent loss functions via

$$D_i^{t+1} \propto -\frac{\partial}{\partial y_i F_t(\mathbf{x}_i)} L(y_i F_t(\mathbf{x}_i)) \implies L(y_i F_t(\mathbf{x}_i)) \propto \int -D_i^{t+1} d(y_i F_t(\mathbf{x}_i)). \tag{13}$$

For example, in AdaBoost the loss w.r.t. the margin of the i-th example is

$$L_{Ada}(y_i F_t(\mathbf{x}_i)) = e^{-y_i F_t(\mathbf{x}_i)}, \tag{14}$$

so the weight update rule will be

$$D_i^{t+1} = \frac{e^{-y_i h_t(\mathbf{x}_i)\alpha_t} \times D_i^t}{\sum_{j=1}^{N} e^{-y_j h_t(\mathbf{x}_j)\alpha_t} \times D_j^t}. \tag{15}$$

Following the inverse derivation, taking the weight update rule of Eq. (15) as given and inferring the loss function via Eq. (13), we recover that any member of the family of loss functions

$$L(y_i F_t(\mathbf{x}_i)) \propto e^{-y_i F_t(\mathbf{x}_i)} + K, \tag{16}$$

where K is some constant w.r.t. the margin $y_i F_t(\mathbf{x}_i)$, would lead to weights updated via Eq. (15). Setting the integration constant to $K = 0$, we obtain the familiar loss of Eq. (14), scaled by some constant. However, any function of the family defined by Eq. (16) once minimized w.r.t. α_t across the entire training set as per Eq. (12) will give us the original α_t from Adaboost, in Eq. (5).

The functional gradient descent viewpoint divides the literature in two families – those which are *consistent* with it and those that are not.

Definition: FGD-consistent *A boosting method is functional gradient descent (FGD)-consistent if it uses a distribution update rule and voting weights α_t that are both consequences of greedily optimising the same, monotonically decreasing, loss function of the margin. Otherwise the method is FGD-inconsistent.*

Only a handful of the existing cost-sensitive boosting variants, CSAda [17,18], AdaDB [14] & CGAda [13, 15] have been derived by first *explicitly* specifying a loss function L and then following the steps of FGD. Hence the first step in our analysis is to follow the inverse derivation for the remaining methods, taking their modified weight update rule and inferring the loss function via Eq. (13). Since modifying the weight updates implies a specific family of equivalent loss functions by Eq. (13), then any change in the distribution update should be reflected in the calculation of voting weights α_t, according to Eq. (12). Otherwise, the chosen α_t are sub-optimal for the purpose of the stagewise minimization of the loss.

Results are shown below. Due to space limitations, we cannot present proofs for each method individually, a general scheme for checking a given boosting algorithm for FGD-consistency is presented in the Supplementary Material.

FGD-consistent	**FGD-inconsistent**
Ada, AdaMEC, CGAda, AsymAda, AdaC2, CSAda[3], AdaDB[3]	CSB0, CSB1, CSB2, AdaC1, AdaC3, AdaCost, AdaCost(β_2)

For the rest of the paper, the treatment of FGD-consistency serves two purposes. Firstly, to identify the precise loss function an algorithm is minimizing. Once this is known, additional properties can be evaluated, revealing whether this loss is sensible in a cost-sensitive scenario. Secondly, FGD-consistency is useful for knowing whether the loss is being *efficiently* optimized.

Our initial hypothesis was that methods that are FGD-inconsistent will be outperformed by those that are FGD-consistent. However, as experiments will show, this only reveals part of the story. As the history of Machine Learning has shown many times, an intuitive choice of a good heuristic can result in practical advances that outperform a sophisticated theory. We certainly do not regard the methods we name above as 'inconsistent' as *necessarily* inferior to 'consistent' ones.

3.2 The Decision Theoretic View

Decision theory gives us straightforward and optimal rules for dealing with cost sensitive binary problems [6, 8]. We have reviewed this in Section 2.1, the result being a simple rule: given the probability $p(y=1|\mathbf{x})$, or –in practice– a probability estimate $\hat{p}(y=1|\mathbf{x})$, we should make predictions using the rule:

$$\hat{p}(y=1|\mathbf{x}) \begin{cases} > c, & \text{predict } y=+1 \\ < c, & \text{predict } y=-1. \end{cases} \tag{17}$$

Definition: Cost-consistent *A method is cost-consistent, if the prediction rule it constructs is equivalent to the rule of Eq. (17), for any given cost matrix of the form of Eq. (1). Otherwise the method is cost-inconsistent.*

All methods with the exception of AdaMEC, which will be discussed separately, use the decision rule:

$$H(\mathbf{x}) = sign[F(\mathbf{x})]. \tag{18}$$

The argument about whether a method is cost-consistent (or not) hinges on the form of the population minimizer of its loss function, i.e. *what the method is attempting to approximate*. The loss function is either explicitly specified by the authors (e.g. CSAda, CGAda) or we can infer it by the proposed weight update rule via the FGD mechanism in Eq. (13). If we plug that population minimizer in the decision rule of Eq.(18) and the rule can be rearranged into the form of Eq. (17), then the method is cost consistent, otherwise it is not. For example, the

[3] CSAda & AdaDB are FGD-consistent under our definition. However, the α_t under the loss function they minimize has no closed form and requires approximation. The two variants use different approximations; CSAda requires the solution of a hyperbolic equation, AdaDB that of a polynomial. This makes both methods computationally intensive and subject to approximation error in the α coefficients. Thus *in practice* they are not guaranteed to be FGD-consistent.

population minimizer of the loss of AdaC2 is $F^*(\mathbf{x}) = \frac{1}{2}\log\frac{p(y=1|\mathbf{x})}{p(y=-1|\mathbf{x})} + \frac{M}{2}\log\frac{c_{FN}}{c_{FP}}$, so it is cost-inconsistent. On the other hand, in the case of CSAda it is $F^*(\mathbf{x}) = \frac{1}{c_{FP}+c_{FN}}\log\frac{p(y=1|\mathbf{x})c_{FN}}{p(y=-1|\mathbf{x})c_{FP}}$, so it is cost-consistent. Table 3, shows the decision rule implemented by each method in terms of probability estimates –as in practice we do not have access to true probabilities. We can classify the methods as so[4]:

Cost-consistent	**Cost-inconsistent**
AdaMEC, CGAda, AsymAda, AdaC1, CSAda, AdaDB	Ada, CSB0, CSB1, CSB2, AdaC2, AdaC3, AdaCost, AdaCost(β_2)

We now present an interesting observation on the AdaMEC procedure [33]. Acknowledging that Eq. (17), is the optimal decision strategy for a given probability estimate, we can reformulate AdaMEC's Eq. (9) in a slightly different way:

$$H_{AdaMEC}(\mathbf{x}) = sign\left[c_{FN} \times \hat{p}(y=1|\mathbf{x}) - c_{FP} \times \hat{p}(y=-1|\mathbf{x})\right]. \tag{19}$$

where $\hat{p}(y=1|\mathbf{x}) = \frac{\sum_{\tau:h_\tau(\mathbf{x})=1}\alpha_\tau}{\sum_{\tau=1}^{t}\alpha_\tau}$. This highlights that Ting's formulation of AdaMEC makes optimal decisions, but only when estimates of probabilities are made in a very specific way – which we are not necessarily constrained to.

Theorem 1: *The AdaMEC rule of Eq. (19) is a special case of the more general*

$$H_{AdaMEC}(\mathbf{x}) = sign\left[\hat{p}(y=1|\mathbf{x}) - c\right], \tag{20}$$

This generalised formulation of AdaMEC, Eq. (20), reduces to Eq. (9) when probability estimates are raw scores of the form $\hat{p}(y=1|\mathbf{x}) = \frac{\sum_{\tau:h_\tau(\mathbf{x})=1}\alpha_\tau}{\sum_{\tau=1}^{t}\alpha_\tau}$.

By viewing AdaMEC in this form, we have *separated* the cost matrix C from the estimation of the probabilities $\hat{p}(y=1|\mathbf{x})$, whereas in Eq. (9) they are somewhat tangled. In this way, we can *choose* how we estimate the probabilities from the base learner outputs. One such way is to do exactly as Ting proposes and use the (normalized) weighted vote to represent each class probability. We are not restricted to this choice. This will be discussed in more detail later.

3.3 The Margin Theoretic View

In Table 3 we summarize the loss function of all methods examined. A closer inspection of the loss function, leads us to interesting observations regarding the training behaviour of the method that minimizes it, which will result in different margin optimization properties. High margin values have been linked to good generalization performance by Schapire et al. [29].

[4]Some methods in practice are used in conjunction with replacing c_{FP} & c_{FN} with hyper-parameters to be set via cross-validation. This has been criticized in e.g. [27] as *heuristic*. Our decision theoretic analysis shows why they resort to this choice. Being cost-inconsistent their decision rule is not geared towards minimizing the expected cost using c_{FP} & c_{FN} directly from the cost matrix, despite them being fixed, known problem characteristics.

Table 3: A summary of the cost-sensitive variants showing the loss function minimized w.r.t. the ensemble constructed on round t and the final decision rule, written in terms of probability estimates and misclassification costs.

Method	Loss Function $L(yF_t(\mathbf{x}))$	Decision rule $sign[\hat{p}(y=1\|\mathbf{x})-\theta]$, where $\theta=...$
Adaboost [29]	$e^{-yF_t(\mathbf{x})}$	$\frac{1}{2}$
AdaMEC	"	$\frac{c_{FP}}{c_{FP}+c_{FN}}$
CGAda [13, 15, 16]	$c(y)e^{-yF_t(\mathbf{x})}$	"
AsymAda [35]	$c(y)^{t/M}e^{-yF_t(\mathbf{x})}$	"
CSAda [17, 18]	$e^{-c(y)yF_t(\mathbf{x})}$	"
AdaDB [14]	"	"
AdaC1 [31, 32]	"	"
CSB2 [33]	$c(y)^{q-1}e^{-yF_t(\mathbf{x})}$ where q models have misclassified $\mathbf{x}$.	$\frac{(c_{FP})^{q-1}}{(c_{FP})^{q-1}+(c_{FN})^{q-1}}$
AdaC2 [31, 32]	$c(y)^t e^{-yF_t(\mathbf{x})}$	$\frac{(c_{FP})^t}{(c_{FP})^t+(c_{FN})^t}$
AdaC3 [31, 32]	$c(y)^{t-1}e^{-c(y)yF_t(\mathbf{x})}$	"
CSB0 [34] CSB1 [33] AdaCost(β_2) [33] AdaCost [7]	Cannot express loss as a function of $yF_t(\mathbf{x})$.	Cannot express decision rule as a function of c_{FP}, c_{FN} & $\hat{p}(y=1\|\mathbf{x})$

In Figures (1 & 2) we present loss functions of two different types versus the margin $y_iF_t(\mathbf{x}_i)$. In Figure 2, we notice that the lines indicating the loss on positive and negative examples cross. When this happens, the emphasis placed on the two classes is reversed: the weights of correctly classified examples of the expensive class are penalized more than those of correctly classified examples of the low-cost class, contrary to the fact that the cost matrix dictates that preserving the former is more important. This was first observed in the case of CSAda & AdaDB by Landesa-Vázquez & Alba-Castro [15, 16] and motivates the following definition which will again allow us to divide the methods into two categories:

Definition: Asymmetry-preserving *A method is asymmetry-preserving if the ratio $r_L(m)$ of the loss on an example of the important class over the loss on an example of the unimportant one – given equal $m = yF_t(\mathbf{x})$– remains greater or equal to 1 during training. Otherwise the method is asymmetry-swapping.*

Variants that minimize a loss of the form $K_1(i)e^{-K_2(i)y_iF_t(\mathbf{x}_i)}$, where $K_1(i)$ is a non-decreasing and $K_2(i)$ an increasing function of $c(y_i)$ (i.e. AdaC1, AdaC3, CSAda & AdaDB) exhibit asymmetry-swapping behaviour as can be demonstrated by Figure 2. Conversely, variants that minimize a loss of the form $K_1(i)e^{-y_iF_t(\mathbf{x}_i)}$, where $K_1(i)$ is a non-decreasing function of $c(y_i)$ (i.e. AdaBoost, AdaMEC, CGAda, AsymAda, AdaC2 & CSB2) are asymmetry preserving, as it is always the case that $r_L(m) = K_1(i)/K_1(j) \geq 1$, as shown in Figure 1. A proof sketch for checking a method for asymmetry preservation can be found in the Supplementary Material.

Of the methods whose loss function cannot be expressed in terms of $y_iF_t(\mathbf{x}_i)$, CSB0 & CSB1 are asymmetry-preserving as their weight updates can only increase

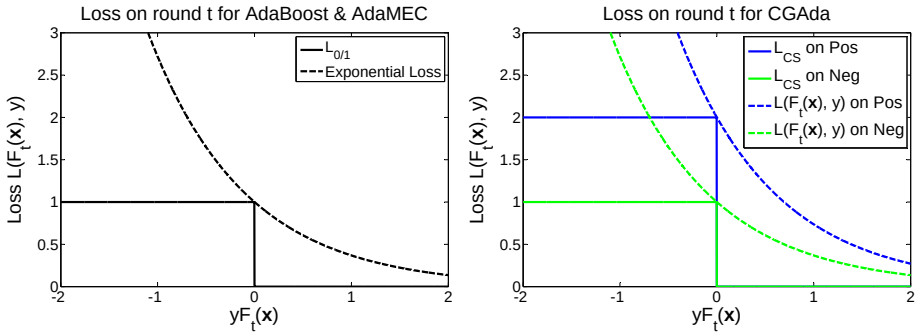

Fig. 1: LEFT: The loss function used in Adaboost. This illustrates the reason why Adaboost is seen as a margin-maximising method – the loss is non-zero even when an example has been correctly classified ($y_i F(\mathbf{x}_i) > 0$). RIGHT: The loss for CGAda in a $2:1$ cost ratio – note that the same margin maximising properties hold, and that examples of the positive (expensive) class always have a loss greater than that of examples of the negative (cheap) class, given an equal $y_i F(\mathbf{x}_i)$.

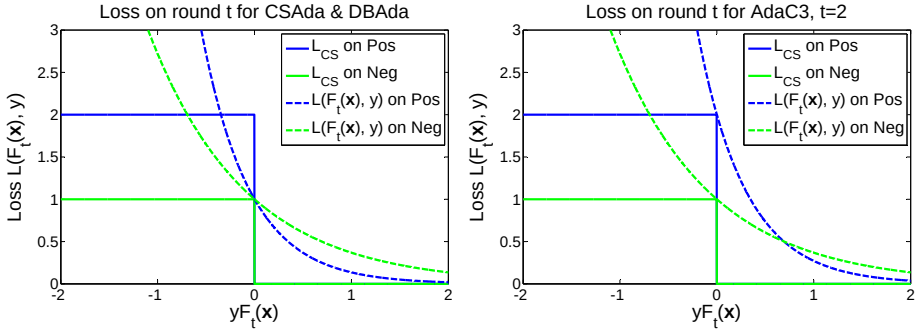

Fig. 2: LEFT: The loss of CSAda & AdaDB does not preserve the class asymmetry leading to asymmetry swapping when $y_i F_t(\mathbf{x_i}) > 0$. Beyond that point, examples of the positive (expensive) class have a lower loss than that of examples of the negative (cheap) class, given an equal $y_i F(\mathbf{x}_i)$. RIGHT: The loss of AdaC3, plotted here for $t = 3$, also exhibits asymmetry swapping, more specifically it does so when $y_i F_t(\mathbf{x_i}) > ((t-1)/(c_{FN} - c_{FP})) \log(\frac{c_{FN}}{c_{FP}})$.

the relative importance of the important class over the unimportant one. On the other hand AdaCost & AdaCost(β_2) do not offer such a guarantee, hence are classified as asymmetry-swapping.

Asymmetry-preserving	**Asymmetry-swapping**
Ada, AdaMEC, CGAda, AsymAda, AdaC2, CSB0, CSB1, CSB2	AdaC1, CSAda, AdaDB, AdaC3, AdaCost, AdaCost(β_2)

Following the observation of Landesa-Vázquez & Alba-Castro [15, 16] that asymmetry swapping behaviour has an adverse effect on the the generalization properties of the final ensemble constructed, we will now investigate the effect of each loss function from an empirical margin-theoretic perspective.

Normalizing the margin $y_i F_t(\mathbf{x_i})$ by dividing by the *1-norm* of the vector comprised of all confidence coefficients $\alpha_1, \ldots, \alpha_t$, we get $m_i = y_i F_t(\mathbf{x_i})/||\boldsymbol{\alpha}||_1 \in [-1, 1]$. The effect that the different loss functions have on the resulting normalized *margin distribution* $\{m_i | i = 1, \ldots, N\}$ is demonstrated by the results shown in Figure 3. The figure shows the cumulative margin distributions[5] produced by *AdaBoost*/AdaMEC, CGAda, AsymAda, CSAda & AdaC3 for four different degrees of imbalance on the *congress* dataset. The results demonstrate that CSAda & AdaC3 are generating lower average margins than the rest of the methods. This is attributed to the asymmetry swapping effect we analyzed earlier. It also agrees with the observation of Landesa-Vázquez & Alba-Castro that asymmetry swapping is having a detrimental effect on the generalization behavior, the latter being dependent on the margin distribution.

3.4 The Probabilistic Model View

The probability estimates produced by boosting algorithms are subject to a systematic distortion. To demonstrate this, we will make use of the view of Adaboost as a *Product of Experts (PoE)* [5]. The *PoE* is a probabilistic model introduced by Hinton [12], under which the probability of an outcome y is expressed as a normalized product of *–not necessarily normalized–* probabilities of y as assigned by different experts. The conclusion of this section is that the probability estimates of the ensemble need to be properly *calibrated*.

To better understand the nature of the probability estimates generated by an AdaBoost ensemble, we now follow the inverse process of [5]. Starting from the probability estimates themselves, we show, without introducing any assumptions, that they correspond to a specific PoE model. The same reasoning allows us to derive similar models for cost sensitive boosting variants thus extending [5] to cover these variants as well.

AdaBoost builds an additive model $F_t(\mathbf{x}) = \sum_\tau \alpha_\tau h_\tau(\mathbf{x})$ to approximate

$$F^*(\mathbf{x}) \;=\; \arg\min_{F(.)} E_{\mathbf{x}y}\Big\{e^{-yF(\mathbf{x})}\Big\} = \frac{1}{2}\log\frac{p(y=1|\mathbf{x})}{p(y=-1|\mathbf{x})}. \tag{21}$$

We can get an estimate of $p(y = 1|\mathbf{x})$ using the AdaBoost outputs $F_t(\mathbf{x}) \approx F^*(\mathbf{x})$,

$$\hat{p}_{Ada}(y=1|\mathbf{x}) = \frac{1}{1+e^{-2F_t(\mathbf{x})}}. \tag{22}$$

This is the form of probability estimates proposed by Friedman et al. and the most popular choice in the boosting literature.

[5] In Figure 3 we do not distinguish between the margin distributions of positive and negative examples. Considering an equal number of positives and negatives, the margin distribution produced by cost-sensitive methods on high cost examples is on average higher than that of low cost examples, as a larger number of low cost examples than high cost ones tend to be misclassified (hence have negative margin). The effect is more pronounced as the skew ratio increases. AdaBoost/AdaMEC, being cost-insensitive in its training phase, produces margin distributions for the two classes that are -in expectation- identical.

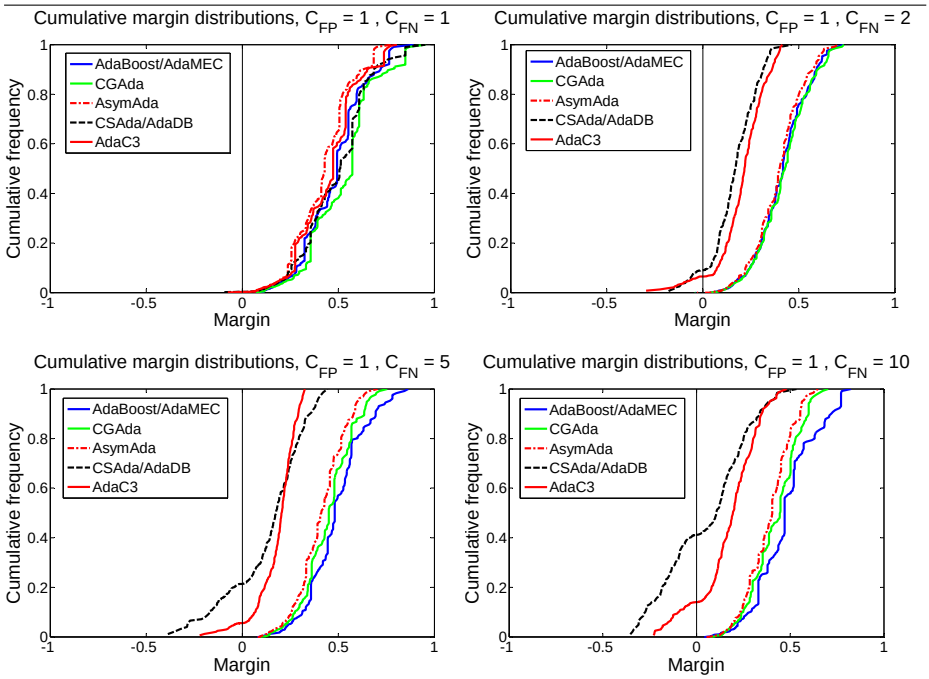

Fig. 3: Cumulative (normalized) margin distributions for *AdaBoost*/AdaMEC, CGAda, AsymAda, CSAda & AdaC3 for degrees of imbalance $\frac{c_{FN}}{c_{FP}} = \{1, 2, 5, 10\}$. When $\frac{c_{FN}}{c_{FP}} = 1$ all methods reduce to the original AdaBoost and produce similar distributions, indicative of margin maximization. But when $\frac{c_{FN}}{c_{FP}} > 1$, CSAda and AdaC3 produce margins closer to 0. This is a result of asymmetry swapping and margin theory suggests it has a negative effect on generalization.

If we substitute $F_t(\mathbf{x}) = \sum_\tau \alpha_\tau h_\tau(\mathbf{x})$ into Eq. (22), we find that the conditional probability estimates under AdaBoost have the form of a *PoE*.

Theorem 2: *The probability estimate of Eq. (22), assigned to class* $y = 1$ *by an AdaBoost ensemble* F_t *on an example* $\mathbf{x}$ *has the form of a product of experts:*

$$\hat{p}_{Ada}(y = 1|\mathbf{x}) \;=\; \frac{\prod_{\tau=1}^{t} \hat{p}_\tau(y = 1|\mathbf{x})}{\prod_{\tau=1}^{t} \hat{p}_\tau(y = 1|\mathbf{x}) + \prod_{\tau=1}^{t} \hat{p}_\tau(y = -1|\mathbf{x})},$$

with experts of the form

$$\hat{p}_\tau(y = 1|\mathbf{x}) = \begin{cases} \epsilon_\tau \quad , & \text{if } h_\tau(\mathbf{x}) = -1 \\ 1 - \epsilon_\tau, & \text{if } h_\tau(\mathbf{x}) = 1, \end{cases}$$

$$\hat{p}_\tau(y = -1|\mathbf{x}) = \begin{cases} 1 - \epsilon_\tau, & \text{if } h_\tau(\mathbf{x}) = -1 \\ \epsilon_\tau \quad , & \text{if } h_\tau(\mathbf{x}) = 1, \end{cases}$$

where ϵ_τ *is the weighted error of the* τ*-th weak learner and* $h_\tau(\mathbf{x}) \in \{-1, 1\}$ *its prediction on example* $\mathbf{x}$.

Table 4: Properties of cost-sensitive Boosting algorithms.

Method	FGD-consistent	Cost-consistent	Asymmetry-preserving	Calibrated estimates
Ada [9]	✓		✓	
AdaCost [7]				
AdaCost(β_2) [33]				
CSB0 [34]			✓	
CSB1 [33]			✓	
CSB2 [33]			✓	
AdaC1 [31,32]		✓		
AdaC2 [31,32]	✓		✓	
AdaC3 [31,32]				
CSAda [17,18]	✓	✓		
AdaDB [14]	✓	✓		
AdaMEC [22,33]	✓	✓	✓	
CGAda [13,15,16]	✓	✓	✓	
AsymAda [35]	✓	✓	✓	
Calibrated AdaMEC	✓	✓	✓	✓
Calibrated CGAda	✓	✓	✓	✓
Calibrated AsymAda	✓	✓	✓	✓

A detailed proof can be found in the Supplementary Material.

Since typically ϵ_τ is only slightly smaller than 0.5 for weak learners (this is how a weak learner is defined), Theorem 2 suggests that the overall $\hat{p}(y=1|\mathbf{x})$ remains close to 0.5. If on the other hand the base learners 'are powerful enough' i.e. at least one ϵ_τ tends to 0, then it dominates the *PoE* and the overall probability estimate tends to 0 or 1. Moreover, we can expect the distortion to be more pronounced as more experts are added to the ensemble. These are exactly the effects observed by Rosset et al. [26] and Caruana et al. [21].

Following the same reasoning, we can construct *PoE* models for the probability estimates of other boosting variants, thus showing that they are also subject to the same form of distortion. For example, as the model constructed by AdaMEC is just a threshold-shifted version of the one built by AdaBoost, $F_t(\mathbf{x}) = [F_t(\mathbf{x})]_{Ada} + \frac{1}{2}\log\frac{c_{FN}}{c_{FP}}$, substituting $F_t(\mathbf{x})$ into Eq. (22), we get that the resulting probability estimates have the form

$$\hat{p}_{AdaMEC}(y=1|\mathbf{x}) = \frac{c_{FP}\cdot\prod_{\tau=1}^{t}\hat{p}_\tau(y=1|\mathbf{x})}{c_{FP}\cdot\prod_{\tau=1}^{t}\hat{p}_\tau(y=1|\mathbf{x}) + c_{FN}\cdot\prod_{\tau=1}^{t}\hat{p}_\tau(y=-1|\mathbf{x})}, \quad (23)$$

with expert probabilities as given in Theorem 2. Thus, an alternative view of AdaMEC is that it classifies examples as $H(\mathbf{x}) = sign[\hat{p}(y=1|\mathbf{x}) - 0.5]$, as AdaBoost does, but changes the underlying probabilistic model of the conditional probabilities for the positive class to that of Eq. (23). This new model is a weighted version of the *PoE* of Theorem 2, where the probability of each class is reinforced by a multiplicative factor equal to its relative importance. We can think of this as adding an additional expert to the *PoE* that captures our *prior knowledge* over the asymmetry. Thus *the probability estimates of boosting algorithms are distorted in a systematic way*. The next step will be to correct for that distortion and this will be the focus of Section 4.

Before we close this section, note that of all the methods proposed, as we can see in Table 4, only three satisfy all the properties of FGD-consistency, cost-

consistency and asymmetry-preservation: CGAda, AsymAda & AdaMEC. Interestingly, each of these is drawn from one of the three main approaches for making a learning algorithm cost-sensitive: cost-proportional resampling/reweighting of the dataset, modifying the training algorithm to take costs into account, and shifting the decision threshold to account for the cost imbalance, respectively.

FGD-consistency ensures that the steps of the algorithm are coherent and geared towards greedily minimizing a monotonically decreasing loss function of the margin. Asymmetry-preservation grants them good generalization by connecting said loss to margin maximization in a cost-sensitive setting. Cost-consistency ensures that the probability estimates are used in a way that is consistent with the goal of minimizing the expected cost of future classifications. What remains is a reasonable guarantee that said probability estimates are good approximations of the true underlying probabilities. This is achieved by *calibration*.

4 Calibration

Probability estimates are not always straightforward to obtain from the outputs of a classifier. The majority of classifiers allow for their output to be treated as a *score* for each test example $\mathbf{x}$ that indicates '*how positive*' $\mathbf{x}$ is. One choice for the score of an AdaBoost ensemble on a given instance $\mathbf{x}$ is

$$s(\mathbf{x}) = \frac{\sum_{\tau:h_\tau(\mathbf{x})=1} \alpha_\tau}{\sum_{\tau=1}^{t} \alpha_\tau} \in [0,1], \tag{24}$$

the weighed fraction of base learners voting for the positive class. Another is

$$s'(\mathbf{x}) = \frac{1}{1+e^{-2F_t(\mathbf{x})}} \in [0,1] \tag{25}$$

which is the quantity we have been denoting with $\hat{p}(y=1|\mathbf{x})$ and using as the estimate of the probability of $\mathbf{x}$ belonging to the positive class throughout the previous sections, following the framework of Friedman et al. [10] and a large body of the boosting literature. The act of converting *raw scores* to actual probability estimates is called *calibration*. Denoting with N the total number of examples, N_s the number of examples with score s and $N_{+,s}$ the number of positives with score s, Zadrozny & Elkan [38] give the following definition:

Definition: Calibrated classifier *A classifier is said to be calibrated if the empirical probability of an example with score $s(\mathbf{x}) = s$ belonging to the positive class, $N_{+,s}/N_s$, tends to the score value s, as $N \to \infty$, $\forall s$.*

A *raw score*, be it of the form $s(\mathbf{x})$ or $s'(\mathbf{x})$, is not sufficient for making a cost-sensitive decision regarding the instance $\mathbf{x}$. What we need is a calibrated estimate of $p(y=1|\mathbf{x})$, given which, classifying $\mathbf{x}$ according to the decision rule of Eq. (17) would give us a *Bayes-optimal* decision.

Niculescu-Mizil & Caruana [21] showed empirically that the scores produced by AdaBoost exhibit a '*sigmoid distortion*', meaning that the scores produced by AdaBoost are a sigmoid transformation of actual probability estimates. – which agrees with the *PoE* model we derived in Theorem 2. The authors also showed that once properly calibrated, AdaBoost produced superior probability estimates to any other model included in their study.

Niculescu-Mizil & Caruana calibrated the scores using three different approaches. The first approach, which they dubbed *logistic correction*, was to use scores of the form $s'(\mathbf{x})$ as implied by the framework of Friedman et al. [10]. The second method used scores of the form $s(\mathbf{x})$ and *Platt scaling* [23], originally used to map *SVM* outputs to conditional class 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 A and B requires the use of a separate validation set. The third method, was to use again scores of the form $s(\mathbf{x})$ and calibrate them via *isotonic regression* [25]. The latter is non-parametric and more general as it can be used to calibrate scores which exhibit any form of monotonic distortion [37]. Among the three methods, Platt scaling produced the best probability estimates on small sample sizes, closely followed by isotonic regression [21].

In this paper we generate scores of the form $s(\mathbf{x})$ under AdaMEC, CGAda & AsymAda, being the only methods in our analysis that are FGD-consistent, cost-consistent and asymmetry preserving. We then apply Platt scaling to calibrate them. We account for class imbalance in the calibration set by the following correction detailed in [23]: if the calibration set has N_+ positive examples and N_- negatives, Platt calibration uses values $\frac{N_++1}{N_++2}$ and $\frac{1}{N_-+2}$, rather than 1 and 0, for the *target probability estimates* of positive and negative examples, respectively. Pseudocode for this full process is provided in the supplementary material.

With these final changes, the calibrated versions of the AdaMEC, CGAda & AsymAda algorithms now satisfy all the properties shown on Table 4 – we proceed to our experimental section, where we compare these methods to all other variants discussed.

5 Empirical evaluation

5.1 Experimental setup

In our empirical analysis we extend our initial explorations found in [22] and compare the performance of all methods[6] we discussed to those of calibrated AdaMEC, AsymAda & CGAda under various degrees of cost imbalance. Univariate logistic regression models, trained with conjugate gradient descent, were chosen as base learners. Their maximum number M was set to 100.

We used 18 datasets from the *UCI repository*. Multiclass problems were handled with a *one-vs-rest* approach: one class was deemed as positive and all others formed the negative one. Our goal is to compare the methods under 21 different cost setups, namely $\frac{c_{FN}}{c_{FP}} \in \{\frac{100}{1}, \frac{50}{1}, \frac{25}{1}, \frac{20}{1}, \frac{15}{1}, \frac{10}{1}, \frac{5}{1}, \frac{2.5}{1}, \frac{2}{1}, \frac{1.5}{1}, \frac{1}{1}, \frac{1}{1.5}, \frac{1}{2}, \frac{1}{2.5}, \frac{1}{5}, \frac{1}{10}, \frac{1}{15}, \frac{1}{20}, \frac{1}{25}, \frac{1}{50}, \frac{1}{100}\}$. We selected an equal number of positive and negative examples, to suppress the additional effects of class imbalance, the same approach followed by Landesa-Vázquez and Alba-Castro [16]. This was achieved by uniformly undersampling the majority class rather than by oversampling the minority class. Thus we avoid overfitting due to duplicates in training/testing/validation sets. A summary of the datasets used can be found in the Supplementary Material.

[6] As CSAda & AdaDB are equivalent within numerical precision [14], we only present results for CSAda. The α_t values were calculated using *Newton steps* and a tolerance of 10^{-6}.

We use a random 25% of the data for testing. The remaining 75% is used for training. To perform 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. For uncalibrated methods, the entire training set was used to fit the models. After training the models (and calibrating on the validation set, where applicable), we evaluated them on the test set. The entire procedure is repeated 30 times. For each method, we report average values and 95% confidence intervals.

To assess the performance of the different approaches, we use the *normalized cost-sensitive loss* $Q(\theta)$ [4, 11, 16], for a given decision threshold θ, averaged over the test set. The loss is given by

$$Q(\theta) = FNR(\theta) \cdot (1 - z) + FPR(\theta) \cdot z \in [0, 1], \tag{26}$$

where $FNR(\theta)$ is the *false negative rate* of the classifier at *decision threshold* θ, $FPR(\theta)$ its *false positive rate* at that threshold and z the *skew* [4, 11] defined as,

$$z = \frac{\pi_- \cdot c_{FP}}{\pi_- \cdot c_{FP} + \pi_+ \cdot c_{FN}} \in [0, 1]. \tag{27}$$

Lower *loss* values are desirable, regardless of the asymmetry. When false positives and false negatives have equal costs, i.e. for $c_{FN} = c_{FP}$, the loss reduces to the expected error rate as measured on the test set. It is normalized in the sense that $Q \in [0, 1]$, regardless of the values of c_{FN} and c_{FP}. A skew $z < 0.5$ signifies that negative examples are more important than positives, values $z > 0.5$ that positive examples are more important and $z = 0.5$ corresponds to the symmetric case of all examples being equally important. For each cost setup (c_{FN}, c_{FP}) , we compute the corresponding skew z and use it to quantify the asymmetry of the task. In our analysis we explore the scenario in which class priors are balanced i.e. $\pi_+ = \pi_- = 0.5$, but costs are not, therefore the skew simplifies to the *cost skew*

$$z = \frac{c_{FP}}{c_{FP} + c_{FN}} = c \in [0, 1]. \tag{28}$$

Since a decision threshold θ that is optimal in training may not necessarily be optimal on a test set, $Q(\theta)$ would constitute an optimistic assessment of the cost-sensitive performance. To avoid this, we follow the '*probabilistic threshold choice*' practice suggested by [11] and instead of *cost curves* ($Q(\theta)$ vs. z for optimal θ on training set), we produce *Brier curves* ($Q(z')$ vs. z). That is, we set the threshold θ for each classifier to be equal to the *change in skew from training to deployment* z', which relates to the skew ratio during training z_{tr} and the skew ratio during deployment z_{dep} via $z_{dep} = \frac{z_{tr} \cdot z'}{z_{tr} \cdot z' + (1 - z_{tr}) \cdot (1 - z')}$. In our experiments, all methods but AdaMEC are trained under a skew ratio z_{tr} which is almost the same as that encountered in deployment, subject to the balance of positives and negatives in the train and test sets on each split, so $z_{dep} \approx z_{tr}$ and the threshold used in Eq. (26) was $\theta = z' \approx 0.5$. AdaMEC was trained under a skew ratio of $z_{tr} = 0.5$ (cost-insensitive training), so on deployment the change in skew would be $z' \approx z_{dep}$, so $\theta \approx z_{dep}$. Abusing notation, from now on we will write $Q(z)$ to refer to the loss of a classifier under *operating condition* (i.e. skew in deployment phase) $z_{dep} = z$.

The *area below the Brier curve* is equal[7] to the *Brier Score* (BS) [11], a common measure for assessing the quality of probability estimates, first proposed by Brier [1]. The BS is defined as the mean squared difference between the probability estimates and the actual class labels of the test examples (if the negative class is denoted with '0'), hence the lower it is, the better the estimates of the model. Thus the difference of the area under the Brier curve of AdaMEC, AsymAda & CGAda and that of their calibrated counterpart is due to the reduction of the *calibration loss* component of the BS. The $Q(z) = z$ diagonal of a Brier curve corresponds to the '*all-negatives*' classifier, while the $Q(z) = 1 - z$ diagonal to the '*all-positives*'.

5.2 Analysis of experimental results

In total we examined $15 \times 18 \times 21$ combinations of $(method, dataset, skew\ ratio)$. Due to space limitations, we cannot present all results in this paper. As an overall measure of the performance of each method across all degrees of imbalance, we calculate the average *area under the Brier curve* it attains per dataset, which equals its expected BS. The results are shown in the Supplementary Material. In Figure 4 we rank all methods according to their area under the Brier curve, across all datasets –higher rank meaning lower area under the curve, i.e. better performance.

Overall, in Figure 4 we see that calibrated versions of AdaMEC, CGAda & AsymAda outrank their uncalibrated counterparts. This can solely be attributed to the decrease of the *calibration loss* component of the *BS*. These results agree with our theoretical observations about the probability estimates of boosting variants being distorted and requiring to be calibrated. Moreover, the uncalibrated versions of AdaMEC, CGAda & AsymAda dominate the remaining variants. This is again in accordance with our theoretical findings as these three methods are the only ones that satisfy all three properties of FGD-consistency, cost-consistency and asymmetry-preservation.

Among AdaMEC, CGAda & AsymAda, the third variant outranks the other two. This pattern carries over to the calibrated versions of the three algorithms. This might indicate some benefit of making the training phase itself cost-sensitive, but there could be a simpler explanation: AsymAda creates an ensemble of predefined size $M = 100$. AdaMEC & CGAda typically use about 40 weak learners of the maximum M allowed. This results in AsymAda producing higher margin values than the other two methods, which leads to better generalization. Some evidence for this is provided in Figure 5, where we have plotted both the average rank attained by each method and the average ensemble size. As we can see calibrated AdaMEC and AsymAda lie on the *pareto front* of the two objectives: attaining a high rank while building a parsimonious model.

To verify the above hypothesis we also included another set of experiments: we fixed the ensemble size for the calibrated versions of AdaMEC, CGAda & AsymAda to be equal for each run[8]. We then compared the statistical significance of the difference in the average ranking of their Brier scores (low rank indicating better

[7] In our experiments it will be an approximation, as we will be generating the Brier curve with non-uniform skew samples to compare the algorithms on a wide range of skew values.

[8] More precisely, AdaMEC & AsymAda were forced to use as many weak learners as CGAda with a maximum of 100.

performance) across all datasets. The resulting *critical difference diagrams* [3], showing when the differences in loss ranking are statistically significant at the 0.01 level under a Nemenyi post-hoc test [20] are shown in Figure 6. The results suggest that there is no clearly dominant method among these three[9]. Therefore, any performance benefit AsymAda had over the other two methods in our main results was due to its larger number of weak learners. A table showing the average Brier Score for each method on each dataset is included in the Supplementary Material.

AdaMEC has been largely overlooked despite having some clear practical benefits. It is more flexible than AsymAda, allowing us to e.g. grow the ensemble after the original training if needed. It is faster than AsymAda, as fewer weak learners will be added to the ensemble–weak learner optimization being the computational bottleneck of training. Finally and perhaps most importantly, unlike all other methods included in this study, AdaMEC does not need to retrain the model if the cost ratio changes in deployment. For these reasons, as a representative of this group of methods we pick AdaMEC in subsequent comparisons, including both its calibrated and uncalibrated versions to showcase the benefits of calibration.

Returning to the Brier score results, excluding AdaMEC, CGAda & AsymAda, the best performing variants are CSB2 & AdaC2. AdaC1 exhibits an erratic behaviour; on about half of the datasets examined it ranks among the top-performing methods in terms of average *BS*, while in the rest of them its performance is among the poorest. AdaCost, AdaCost(β_2), CSAda and on some datasets AdaC1, were found to yield the highest *BS*. What these methods have in common is the asymmetry-swapping effect[10].

The area under the Brier curve does not take into account the variance across runs, neither does it allow us to observe the different behaviours exhibited by each method under the different degrees of skew. To observe these effects we need to examine the entire Brier curve. For clarity we will only compare some representative methods against AdaMEC & calibrated AdaMEC. CSB2 is chosen for its relatively good performance across datasets. AdaC1 is chosen despite its erratic behaviour, since, on some datasets it ranks among the top-performing methods. The Brier curves for some characteristic datasets can be found in Figure 7. The Brier curves of the remaining datasets are in the Supplementary Material. Some additional comparisons of calibrated AdaMEC, AdaMEC, CSB2 and AdaC2 under different evaluation measures can also be found in [22].

CSB2 performs very poorly under low values of skew ($0.3 < z < 0.7$). This is because of the *saturation* phenomenon, also observed by [16], i.e. the tendency of CSB2 to construct '*all-positives*' or '*all-negatives*' models. The eagerness of CSB2

[9] This is not very surprising. All 3 approaches are approximating the minimizer $F^*(\mathbf{x}) = \frac{1}{2}\log\frac{p(y=1|\mathbf{x})}{p(y=-1|\mathbf{x})} + \frac{1}{2}\log\frac{c_{FN}}{c_{FP}}$ of the loss $L(F_M) = \mathbb{E}_{\mathbf{x},y}[c(y)e^{-yF_M(\mathbf{x})}]$, albeit in different ways: AdaMEC by shifting the decision threshold, CGAda by reweighting and AsymAda by modifying the base algorithm to simulate splitting the asymmetry equally among all M rounds.

[10] AdaCost & AdaC1 have been criticised in recent studies as being 'unstable, repeatedly producing meaningless negative, or even imaginary, α_t values' [17, page 8]. AdaCost was found in [16] to exhibit a similar 'worst-than-baseline' behaviour. The authors also attribute it to the α_t coefficients not being guaranteed to be positive reals. In our experiments we prevented this situation from occurring by forcing the ensemble to terminate training if it cannot find a learner with a positive real α_t. Rather than adding 'meaningless' learners, our strategy led to ensembles consisting of few experts, as can also be verified by Figure 5, a potential reason for the poor performance of AdaCost, AdaCost(β_2) and –on some datasets– AdaC1.

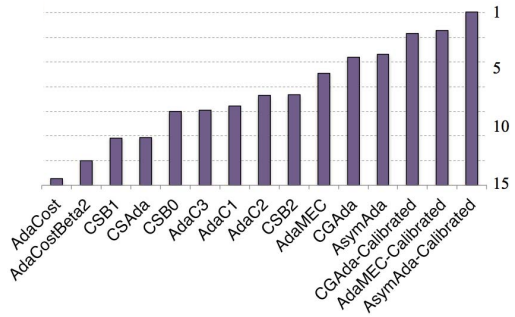

Fig. 4: All methods, across all datasets, ranked by their area under the Brier curve – higher is better. This illustrates the resilience of each method to a spectrum of changing cost ratios. The highest ranked method is *AsymAda-Calibrated*.

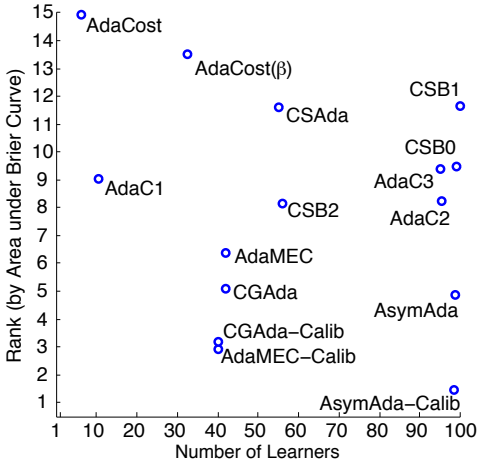

Fig. 5: Pareto plot showing the tradeoff between the size of the final ensemble and the rank by area under the Brier curve. The plot shows that *calibrated* Adaboost and *calibrated* AsymAda are pareto-optimal, with the former to be preferred for parsimonious models.

to classify examples to the costly class is explained by our theory by the fact that CSB2 overemphasises the costs as we saw in Table 3. This strategy starts paying off when the degree of skew becomes very high ($z \leq 0.1$ or $z \geq 0.9$) when, it becomes one of the dominant methods, second only to calibrated AdaMEC. On the other hand, AdaC1 exhibits particularly poor performance when the skew ratio is high ($z < 0.2$ or $z > 0.8$). AdaMEC tends to perform well for low values of skew, but for higher degrees of imbalance ($z \leq 0.3$ or $z \geq 0.7$), it is outranked.

Calibrated AdaMEC sacrifices part of the dataset to solve the harder problem of probability estimation. As a result, on average it is outranked by the other meth-

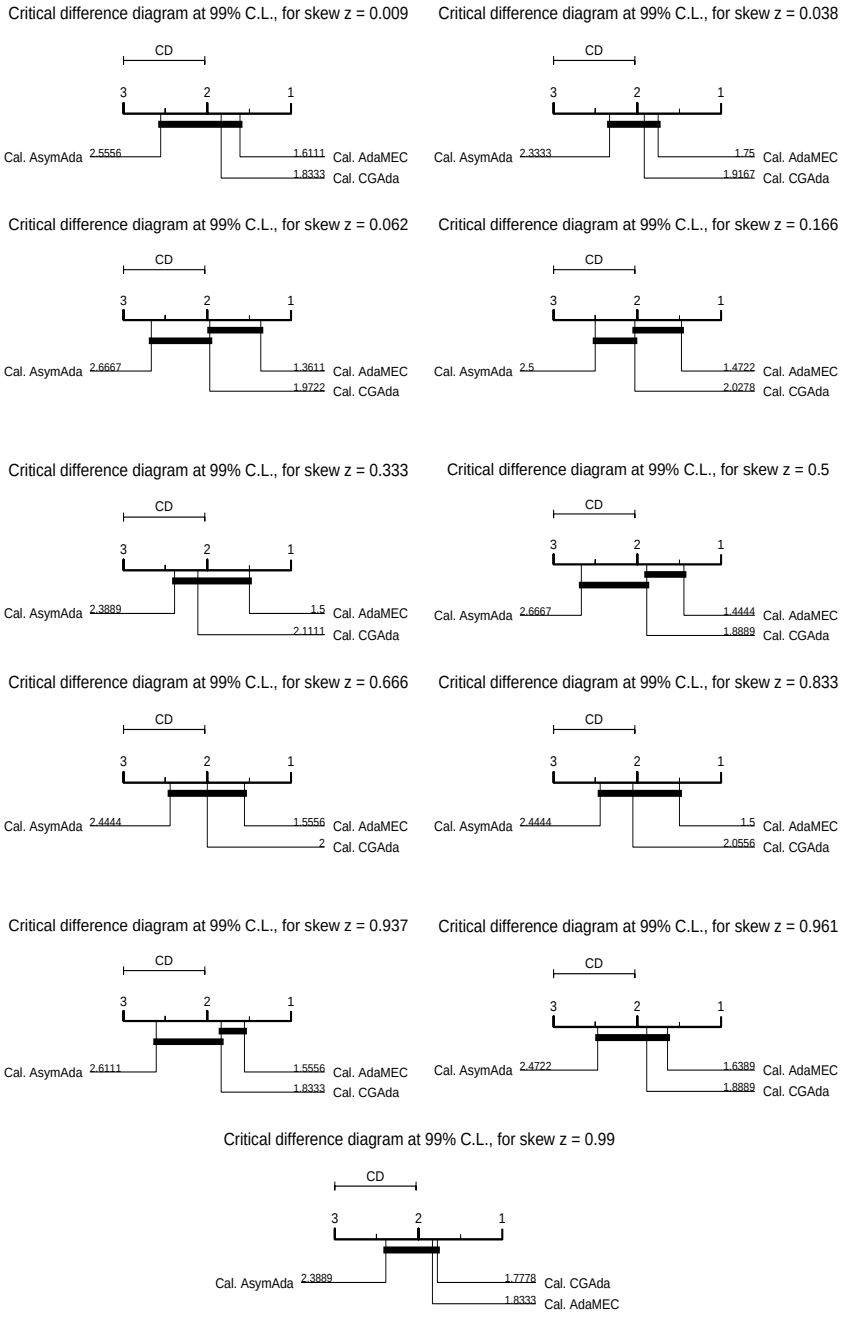

Fig. 6: Comparison of the average Brier Score ranks across all datasets attained by AdaMEC, CGAda & AsymAda against each other under the Nemenyi test, for every other skew ratio examined. All methods were calibrated and used the same ensemble size on each run. Groups of methods that are not significantly different at the 0.01 level are connected. We see that no method clearly dominates the others.

ods when the task is skew-insensitive ($z = 0.5$). The effect is barely detectable on most datasets as the confidence intervals overlap. A simple solution is the *use of cross-validation for calibration*. But as the imbalance increases, the investment of *calibrated AdaBoost*[11] in estimating calibrated probabilities pays off and it clearly dominates all other methods. On nearly all datasets and for all values of z examined, it ranks first or tied for first among the 4 methods studied here.

A closer inspection of the individual datasets shows that calibrated AdaMEC is most clearly outperforming the competitors on larger datasets, since a large training set allows it to compute better probability estimates. This effect is more pronounced in datasets which are also high-dimensional (*splice*, *musk2*, *krvskp*, *waveform*, *spambase*). In lower-dimensional datasets, like *mushroom*, the confidence intervals for most methods tend to overlap as the problem is easier.

6 Conclusion & Future Work

We analysed the cost sensitive boosting literature spanning the last two decades under a variety of theoretical frameworks. We used tools from four different perspectives to understand this: decision theory, functional gradient descent, margin theory, and probabilistic modelling. Our main finding is that cost-sensitive modifications seem unnecessary for Adaboost, if proper calibration is applied.

The tools from the different theory frameworks each provide different perspectives and strengths for the analysis. Algorithms that do not fit the functional gradient descent framework cannot be viewed as efficient procedures to greedily minimize a loss function of the margin. The decision theoretic analysis shows that certain methods are not implementing decision rules that align with the goal of minimizing the expected cost of future classifications. Finally, margin theory predicts that methods that invert class importance during their execution will exhibit poor generalization performance.

Only three algorithms turn out to be consistent with the rules of these theoretical perspectives – AdaMEC [33], CGAda [13] & AsymAda [35]. However, in our final theory angle, we find they share a common flaw: they assume that AdaBoost produces well-calibrated probability estimates. By reinterpreting Boosting as a *Product of Experts* [5] we showed that this assumption is *violated* and the estimates produced by AdaBoost deviate from true posterior probabilities in a *predictable* fashion. To correct for this, we applied calibration using *Platt scaling* [23] – a detailed pseudocode for its implementation has been provided in the Supplementary Material[12].

Experiments on 18 datasets across 21 degrees of imbalance support the hypothesis – showing that once calibrated, these three algorithms perform equivalently, and outperform all others. Our final recommendation – based on simplicity, flexibility and performance – is *calibrated* AdaMEC, i.e. to use the *original* Adaboost algorithm with a shifted decision threshold, and *calibrated* probability estimates.

[11] Here we refer to *calibrated AdaMEC* as *calibrated AdaBoost*. We remind the reader that AdaMEC builds the same model as AdaBoost, but while the latter does not shift the threshold to account for the asymmetry, AdaMEC does. Conversely, calibrated AdaMEC builds exactly the same model as calibrated AdaBoost, but applies threshold-shifting in the decision rule.

[12] Matlab code can also be found in the link: http://www.cs.man.ac.uk/∼gbrown/software/

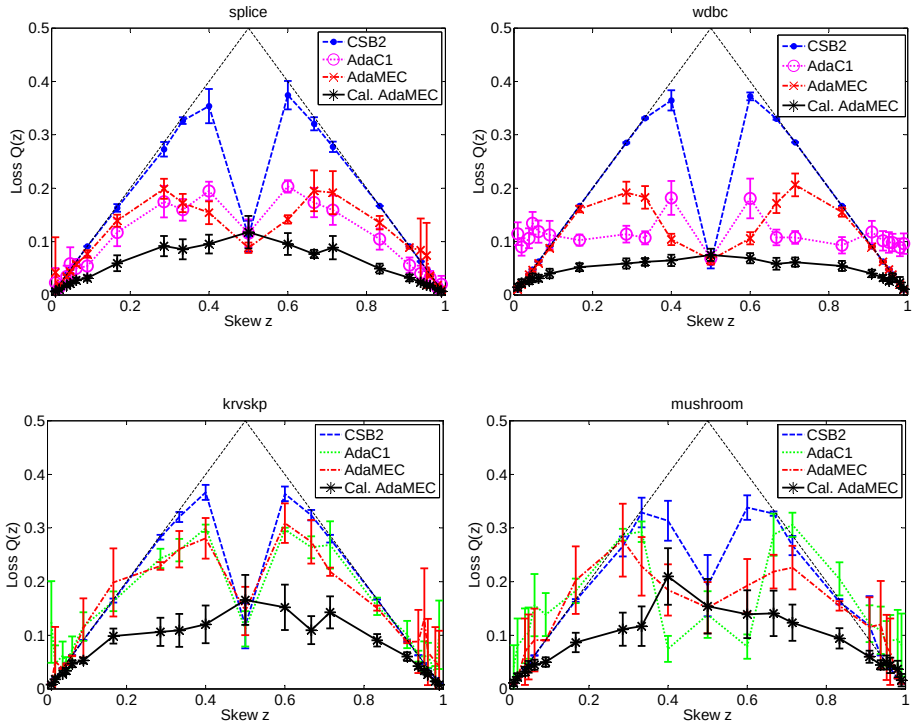

Fig. 7: Loss Q under various degrees of skew z for some characteristic datasets included in the study. Lower values indicate better cost-sensitive classification performance. The area under each (Brier) curve corresponds to the *Brier Score* of the final model. The difference of the areas under the curve of AdaMEC and that of *calibrated AdaMEC* is due to the reduction of the *calibration loss* component of the BS. The $Q(z) = z$ diagonal, corresponds to the '*all-negatives*' classifier, while the $Q(z) = 1-z$ diagonal to the '*all-positives*'. CSB2 is prone to saturating leading to high loss when the skew is high. AdaC1 is prone to ignoring the asymmetry, leading to high loss as z moves away from 0.5. *Calibrated AdaMEC*, adopts in each case an asymmetric behaviour that leads to low loss. As a result it consistently attains the lowest –or tied for lowest– loss.

For future work, we note that, even though the calibrated algorithms examined match or exceed the performance of other cost-sensitive variants, there are various parameters of the calibration procedure we left unoptimized - suggesting room for improvement. Examples include the choice of calibration method, optimizing the split between training and calibration set and the use of *leave-one-out cross-validation*. Another interesting research direction would be to investigate performing training and calibration in a single step, i.e. choosing at each round the weak learner so that the new ensemble produces calibrated probability estimates.

Finally, our key findings have the potential to carry over to other learning algorithms beyond Adaboost. Identifying when this is the case would mean re-

ducing cost-sensitive learning to probability estimation, with all the theoretical guarantees and practical advantages discussed in this paper.

Acknowledgments

Nikolaos Nikolaou and Gavin Brown were supported by the EPSRC Centre for Doctoral Training [EP/I028099/1] and AnyScale Applications [EP/L000725/1] grants. Meelis Kull and Peter Flach were supported by the REFRAME project granted by the European Coordinated Research on Long-term Challenges in Information and Communication Sciences & Technologies ERA-Net (CHIST-ERA), and funded by the Engineering and Physical Sciences Research Council in the UK under grant EP/K018728/1. The authors would like to thank Konstantinos Sechidis, Henry Reeve, Sarah Nogueira, Andrew Webb, Joe Mellor and Janez Demšar for their useful comments and suggestions.
Data Access Statement All research data supporting this publication are directly available within this publication.

References

1. G. W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78:1–3, 1950.
2. R. Caruana and A. Niculescu-Mizil. An Empirical Comparison of Supervised Learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 161–168. ACM, 2006.
3. J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
4. C. Drummond and R. C. Holte. Explicitly representing expected cost: An alternative to ROC representation. In *In Proc. of the 6th ACM SIGKDD*, pages 198–207, 2000.
5. N. U. Edakunni, G. Brown, and T. Kovacs. Boosting as a product of experts. *UAI*, 2011.
6. C. Elkan. The foundations of cost-sensitive learning. In *IJCAI*, 2001.
7. W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan. AdaCost: Misclassification cost-sensitive boosting. In *ICML*, pages 97–105, 1999.
8. P. A. Flach. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012.
9. 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.
10. J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:337–407, 2000.
11. J. Hernández-Orallo, P. A. Flach, and C. Ferri. Brier curves: a new cost-based visualisation of classifier performance. In *Proc. of the 28th International Conference on Machine Learning, ICML*, pages 585–592, 2011.
12. G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.
13. 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.
14. I. Landesa-Vázquez and J. L. Alba-Castro. Double-base asymmetric AdaBoost. *Neurocomputing*, 118:101 – 114, 2013.
15. I. Landesa-Vázquez and J. L. Alba-Castro. Revisiting AdaBoost for cost-sensitive classification. part i: Theoretical perspective. *arXiv:1507.04125v1*, 2015.
16. I. Landesa-Vázquez and J. L. Alba-Castro. Revisiting AdaBoost for cost-sensitive classification. part ii: Empirical analysis. *arXiv:1507.04126v1*, 2015.
17. H. Masnadi-Shirazi and N. Vasconcelos. Asymmetric boosting. In *ICML*, 2007.
18. H. Masnadi-Shirazi and N. Vasconcelos. Cost-sensitive boosting. *IEEE Trans. Pat. Anal. Mach. Intell.*, 33 (2):294–309, 2011.

19. L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. *In NIPS 12*, pages 512–518, 2000.
20. P. B. Nemenyi. Distribution-free multiple comparisons. *PhD, Princeton University*, 1963.
21. A. Niculescu-Mizil and R. Caruana. Obtaining calibrated probabilities from boosting. In *UAI*, 2005.
22. N. Nikolaou and G. Brown. Calibrating AdaBoost for asymmetric learning. In *In Proc. of Multiple Classifier Systems 2015*, pages 112–124, 2015.
23. 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.
24. J. R. Quinlan. Bagging, boosting, and C4.5. In *Proc. of the 13th Nat. Conf. on AI - Vol. 1*, pages 725–730, 1996.
25. T. Robertson, F. Wright, and R. Dykstra. *Order Restricted Statistical Inference*. Probability and Statistics Series. Wiley, 1988.
26. S. Rosset, J. Zhu, T. Hastie, and R. Schapire. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973, 2004.
27. M. J. Saberian and N. Vasconcelos. Learning optimal embedded cascades. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 34, 2012.
28. R. E. Schapire. *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, chapter Explaining AdaBoost, pages 37–52. 2013.
29. R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. 1997.
30. R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37 (3):297–336, 1999.
31. 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.
32. Y. Sun, A. K. C. Wong, and Y. Wang. Parameter inference of cost-sensitive boosting algorithms. In *Mach. Lear. and Data Min. in Pat. Recogn.*, pages 21–30, 2005.
33. K. M. Ting. A comparative study of cost-sensitive boosting algorithms. In *ICML*, pages 983–990, 2000.
34. K. M. Ting and Z. Zheng. Boosting cost-sensitive trees. In *Discovery Science: First International Conference*, pages 244–255, 1998.
35. P. Viola and M. Jones. Fast and robust classification using asymmetric AdaBoost and a detector cascade. In *NIPS*, 2002.
36. Wu, X. and Kumar, V. and Quinlan, J. R. and Ghosh, J. and Yang, Q. and Motoda, H. and McLachlan, G. J. and Ng, A. and Liu, B. and Philip, S. Y. and others. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.
37. B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *ICML*, pages 609–616, 2001.
38. B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates, 2002.