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

Nikolaos Nikolaou[1], Narayanan Edakunni[1], Meelis Kull[2], Peter Flach[2], and Gavin Brown[1]

[1] School of Computer Science, University of Manchester,
Kilburn Building, Oxford Road, Manchester, M13 9PL, UK.
[nikolaos.nikolaou,gavin.brown]@manchester.ac.uk,narayanan.unny@gmail.com
[2] Department of Computer Science, University of Bristol,
The Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, UK.
[meelis.kull,peter.flach]@bristol.ac.uk

Proofs & Proof Sketches

Proof of Theorem 1

Theorem 1: *The generalised formulation of AdaMEC,*

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

reduces to

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

where

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

when probability estimates are raw scores of the form $\hat{p}(y=1|\mathbf{x}) = \frac{\sum_{\tau:h_\tau(\mathbf{x})=1}\alpha_\tau}{\sum_{\tau=1}^{t}\alpha_\tau}$.

Proof. Our generalized formulation of AdaMEC's prediction rule is

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

where $\hat{p}(y=1|\mathbf{x})$ denotes the probability estimate the AdaBoost ensemble F_t assigns to example $\mathbf{x}$ belonging to the positive class, regardless of how it is estimated.

Suppose we opt to use as probability estimates raw scores of the form

$$\hat{p}(y=1|\mathbf{x}) = s(\mathbf{x}) = \frac{\sum_{\tau:h_\tau(\mathbf{x})=1}\alpha_\tau}{\sum_{\tau=1}^{t}\alpha_\tau}.$$

Then Eq. (1) becomes

$$H_{AdaMEC}(\mathbf{x}) = sign\left[(1-c)\hat{p}(y=1|\mathbf{x}) - c(1-\hat{p}(y=1|\mathbf{x}))\right]$$
$$= sign\left[\frac{c_{FN}}{c_{FP}+c_{FN}}\frac{\sum_{\tau:h_\tau(\mathbf{x})=1}\alpha_\tau}{\sum_{\tau=1}^{t}\alpha_\tau} - \frac{c_{FP}}{c_{FP}+c_{FN}}(1-\frac{\sum_{\tau:h_\tau(\mathbf{x})=1}\alpha_\tau}{\sum_{\tau=1}^{t}\alpha_\tau})\right].$$

Since $c_{FP}+c_{FN}$ is bounded, constant and non-negative, this is equivalent to

$$H_{AdaMEC}(\mathbf{x}) = sign\left[c_{FN}\frac{\sum_{\tau:h_\tau(\mathbf{x})=1}\alpha_\tau}{\sum_{\tau=1}^{t}\alpha_\tau} - c_{FP}(1-\frac{\sum_{\tau:h_\tau(\mathbf{x})=1}\alpha_\tau}{\sum_{\tau=1}^{t}\alpha_\tau})\right]$$
$$= sign\left[c_{FN}\frac{\sum_{\tau:h_\tau(\mathbf{x})=1}\alpha_\tau}{\sum_{\tau=1}^{t}\alpha_\tau} - c_{FP}\frac{\sum_{\tau:h_\tau(\mathbf{x})=-1}\alpha_\tau}{\sum_{\tau=1}^{t}\alpha_\tau}\right].$$

As $\sum_{\tau=1}^{t}\alpha_\tau$ is bounded, constant and non-negative, we get the equivalent rule

$$H_{AdaMEC}(\mathbf{x}) = sign\left[c_{FN}\sum_{\tau:h_\tau(\mathbf{x})=1}\alpha_\tau - c_{FP}\sum_{\tau:h_\tau(\mathbf{x})=-1}\alpha_\tau\right]. \quad (2)$$

Rearranging gives us

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

where

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

□

Proof sketch: Checking for FGD-Consistency

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.*

We now present a general scheme for checking a given boosting variant for FGD-consistency. We assume that the weight update rule D_i^{t+1} and the form of the voting weights α_t are given in the description of the algorithm.

Mason et al. [1] noted that the weight update rule D_i^{t+1} and the monotonically decreasing loss function $L(y_iF_t(\mathbf{x}_i))$ of the margin $y_iF_t(\mathbf{x}_i)$ are related via

$$D_i^{t+1} = \frac{\frac{\partial}{\partial y_iF_t(\mathbf{x}_i)}L(y_iF_t(\mathbf{x}_i))}{\sum_{j=1}^{N}\frac{\partial}{\partial y_jF_t(\mathbf{x}_j)}L(y_jF_t(\mathbf{x}_j))}. \quad (3)$$

Under Eq. (3), a given form of weight updates D_i^{t+1} implies a specific family of equivalent loss functions via

$$D_i^{t+1} \propto -\frac{\partial}{\partial y_i F_t(\mathbf{x}_i)} L(y_i F_t(\mathbf{x}_i)) \implies L(y_i F_t(\mathbf{x}_i)) \propto \int -D_i^{t+1} d(y_i F_t(\mathbf{x}_i)). \quad (4)$$

If the RHS of Eq. (4) is not a monotonically decreasing function of the margin $y_i F_t(\mathbf{x}_i)$, then neither is the loss $L(y_i F_t(\mathbf{x}_i))$ and the method is FGD-inconsistent under our definition. This is the case for some of the methods examined here, namely CSB0, CSB1, AdaCost & AdaCost(β_2).

For all other methods examined here, by recursively applying the weight update given it can be shown that D_i^{t+1} can be written as a monotonically decreasing function of the margin. In all methods it is of the general form:

$$D_i^{t+1} \propto K_1(i) e^{-K_2(i) y_i F_t(\mathbf{x}_i)} \quad (5)$$

where $K_1(i)$ and $K_2(i)$ are non-decreasing functions of $c(y_i)$, the cost of the i-th example (e.g. $K_1(i) = 1$ or $K_2(i) = 1$ are also admissible).

Combining Eq. (4) and Eq. (5), we obtain

$$\begin{aligned} L(y_i F_t(\mathbf{x}_i)) &\propto \int -K_1(i) e^{-K_2(i) y_i F_t(\mathbf{x}_i)} d(y_i F_t(\mathbf{x}_i)) \\ &= \frac{K_1(i)}{K_2(i)} e^{-K_2(i) y_i F_t(\mathbf{x}_i)} + K, \end{aligned} \quad (6)$$

where K is constant w.r.t. the margin $y_i F_t(\mathbf{x}_i)$.

Setting $K = 0$ and ignoring the scaling factor, we can limit ourselves for simplicity to a single member of the family of functions $L(y_i F_t(\mathbf{x}_i))$ and use

$$L(y_i F_t(\mathbf{x}_i)) = \frac{K_1(i)}{K_2(i)} e^{-K_2(i) y_i F_t(\mathbf{x}_i)}. \quad (7)$$

The loss functions of this form are the ones we present in Table 3 of the main paper. This is of course an optional step, as any loss of the form of Eq. (6) is equivalent for optimization purposes, i.e. it is minimized by the same $F_t(\mathbf{x}_i)$.

If the given form of the voting weight of the t-th base learner minimizes the empirical risk under $L(y_i F_t(\mathbf{x}_i))$ on the training set, i.e.

$$\alpha_t^* = \arg\min_{\alpha_t} \; \Big[\frac{1}{N} \sum_{i=1}^{N} L(y_i (F_{t-1}(\mathbf{x}_i) + \alpha_t h_t(\mathbf{x}_i)))\Big]. \quad (8)$$

then the method is by our definition FGD consistent. Otherwise it is FGD-inconsistent.

Proof sketch: Checking for Cost-Consistency

Definition: Cost-Consistent *A method is cost-consistent, if the prediction rule it constructs is equivalent to*

$$\hat{p}(y=1|\mathbf{x}) > \frac{c_{FP}}{c_{FP}+c_{FN}}, \tag{9}$$

for any given cost matrix of the form

$$C = \begin{bmatrix} c_{TP}=0 & c_{FN} \\ c_{FP} & c_{TN}=0 \end{bmatrix}$$

with $c_{FP}, c_{FN} > 0$*. Otherwise the method is cost-inconsistent.*

We now present a general scheme for checking a given boosting variant for cost-consistency. We assume that we know the loss function $L(yF_t(\mathbf{x}))$ that the algorithm minimizes in stages[3].

For most methods examined here, the minimizer F^* of their expected loss $L(yF_t(\mathbf{x}))$ can be written as a function Φ of the true conditional class probability $p(y=1|\mathbf{x})$ and the cost setup, i.e.

$$F^*(\mathbf{x}) \;=\; \arg\min_F E_{\mathbf{x}y}\Big\{L(yF_t(\mathbf{x}))\Big\} = \Phi(p(y=1|\mathbf{x}), c_{FN}, c_{FP}). \tag{10}$$

In Table 1 we summarize the population minimizer F^* under the loss function minimized by each method.

Boosting methods greedily approximate the true minimizer F^* by an additive model $F_t(\mathbf{x}) = \sum_{\tau=1}^{t} \alpha_\tau h_\tau(\mathbf{x})$ estimated on a finite training set. So we can replace true probabilities with estimates[4] regardless of how they are estimated:

$$F^*(\mathbf{x}) = \Phi(p(y=1|\mathbf{x}), c_{FN}, c_{FP}) \approx F_t(\mathbf{x}) = \Phi(\hat{p}(y=1|\mathbf{x}), c_{FN}, c_{FP}). \tag{11}$$

Once the final ensemble $F_t(\mathbf{x})$ is constructed, all boosting methods make predictions using the rule

$$H(\mathbf{x}) = sign[F_t(\mathbf{x})] = sign[\Phi(\hat{p}(y=1|\mathbf{x}), c_{FN}, c_{FP})]. \tag{12}$$

Simplifying Eq. (12), we derive the decision rules given in Table 3 of the main paper. If a method implements a decision rule equivalent to that of Eq. (9) under any cost matrix C it is cost-consistent. Otherwise it is not.

Finally, if $L(yF_t(\mathbf{x}))$ cannot be expressed as a monotonically decreasing function of the margin $yF_t(\mathbf{x})$, where $F_t(\mathbf{x}) = \sum_{\tau=1}^{t} \alpha_\tau h_\tau(\mathbf{x})$ is the actual ensemble

[3] Given only the weight update rule of a method, we saw how to derive $L(yF_t(\mathbf{x}))$ in the previous proof sketch.

[4] To examine a method for cost-consistency it would have sufficed to inspect the decision rule under the optimal model F^* shown in Table 1, i.e. a decision rule involving true probabilities rather than estimates. We chose to use estimates for notational consistency. It should be clear though that cost-consistency is ultimately only due to the loss function L rather than the specific way probabilities are estimated.

Table 1. The population minimizer of the loss function of each method. This is directly derived from the loss function of Table 3 of the main paper and results to the decision rule shown on the same table (after replacing true probabilities with estimates as is the case in practice). In the case of AdaMEC -included here for completeness- the decision rule is defined this way, motivated by decision theory, as explained in the main paper.

Method	Population Minimizer $F^*(\mathbf{x})$ of Loss $L(F)$
Adaboost	$\frac{1}{2}\log\frac{p(y=1\vert\mathbf{x})}{p(y=-1\vert\mathbf{x})}$
AdaMEC	$\frac{1}{2}\log\frac{p(y=1\vert\mathbf{x})}{p(y=-1\vert\mathbf{x})} + \frac{1}{2}\log\frac{c_{FN}}{c_{FP}}$
CGAda	”
AsymAda	”
CSAda	$\frac{1}{c_{FN}+c_{FP}}\log\frac{p(y=1\vert\mathbf{x})}{p(y=-1\vert\mathbf{x})} + \frac{1}{c_{FN}+c_{FP}}\log\frac{c_{FN}}{c_{FP}}$
AdaDB	”
AdaC1	”
CSB2	$\frac{1}{2}\log\frac{p(y=1\vert\mathbf{x})}{p(y=-1\vert\mathbf{x})} + \frac{q-1}{2}\log\frac{c_{FN}}{c_{FP}}$, where q models have misclassified $\mathbf{x}$.
AdaC2	$\frac{1}{2}\log\frac{p(y=1\vert\mathbf{x})}{p(y=-1\vert\mathbf{x})} + \frac{t}{2}\log\frac{c_{FN}}{c_{FP}}$, where t is the number of boosting rounds.
AdaC3	$\frac{1}{c_{FN}+c_{FP}}\log\frac{p(y=1\vert\mathbf{x})}{p(y=-1\vert\mathbf{x})} + \frac{t}{c_{FN}+c_{FP}}\log\frac{c_{FN}}{c_{FP}}$, where t is the number of boosting rounds.
CSB0 CSB1 AdaCost(β_2) AdaCost	Cannot express population minimizer as a function of c_{FP}, c_{FN} & $p(y=1\vert\mathbf{x})$

constructed, then we cannot derive a decision rule as a function of $\hat{p}(y=1|\mathbf{x})$, c_{FN} & c_{FP}. This is the case for some of the methods examined here, namely CSB0, CSB1, AdaCost & AdaCost(β_2)[5]. As there can be no guarantee that they satisfy Eq. (9), the methods are classified as cost-inconsistent.

Proof sketch: Checking for Asymmetry-Preservation

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.*

We now present a general scheme for checking a given boosting variant for asymmetry-preservation. We assume that we know the loss function $L(yF_t(\mathbf{x}), c(y))$ that the algorithm minimizes in stages[6]. Here we make explicit the fact that L also depends on the cost of each example, unlike the rest of the paper where we simplified notation for clarity.

Based on the above definition, a method is asymmetry preserving if for any two examples $(\mathbf{x}_i, y_i)$ and $(\mathbf{x}_j, y_j)$ such that $c(y_i) > c(y_j)$ and $y_iF_t(\mathbf{x}_i) = y_jF_t(\mathbf{x}_j) = m$, its loss function L satisfies the following property:

$$r_L(m) = \frac{L(y_iF_t(\mathbf{x}_i), c(y_i))}{L(y_jF_t(\mathbf{x}_j), c(y_j))} = \frac{L(m, c(y_i))}{L(m, c(y_j))} \geq 1, \forall m \tag{13}$$

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 $c(y_i)$ are asymmetry preserving, as it is always the case that $r_L(m) = \frac{K_1(i)}{K_1(j)} \geq 1$.

On the other hand, variants that minimize a loss of the form $K_1(i)e^{-K_2(i)y_iF_t(\mathbf{x}_i)}$, where $K_1(i)$ and $K_2(i)$ are a non-decreasing and an increasing function of $c(y_i)$, respectively, have

$$r_L(m) = \frac{K_1(i)e^{-K_2(i)y_iF_t(\mathbf{x}_i)}}{K_1(j)e^{-K_2(j)y_jF_t(\mathbf{x}_j)}} = \frac{K_1(i)}{K_1(j)}e^{m(K_2(j)-K_2(i))}. \tag{14}$$

It can be shown that $\exists m : r_L < 1$. More specifically when

$$m > \frac{1}{K_2(i) - K_2(j)} \log\Big(\frac{K_1(i)}{K_1(j)}\Big), \tag{15}$$

[5] More specifically, CSB0 uses the loss $L(y, \mathbf{x}) = \frac{1}{N}c(y)^{q-1}$, where $q \geq 0$ is the number of models that misclassified example $\mathbf{x}$, which is not a function of $F_t(\mathbf{x})$. CSB1 uses the loss $L(y, \mathbf{x}) = \frac{1}{N}c(y)^{q-1}e^{-y\sum_{\tau=1}^{t} h_\tau(\mathbf{x})}$, which does not depend on the weighted model $F_t(\mathbf{x})$, but rather on its unweighted counterpart and AdaCost & AdaCost(β_2) use the loss $L(y, \mathbf{x}) = \frac{1}{N}e^{-y\sum_{\tau=1}^{t} \beta_\tau(\mathbf{x})\alpha_\tau h_\tau(\mathbf{x})}$, where the additional $\beta_\tau(\mathbf{x})$ factors break the direct dependence of the loss w.r.t. $F_t(\mathbf{x})$.

[6] Given only the weight update rule of a method, we saw how to derive $L(yF_t(\mathbf{x}), c(y))$ in the FGD-consistency proof sketch.

the importance of the two classes is flipped. Hence such methods are asymmetry-swapping.

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

Proof of Theorem 2

Theorem 2: *The probability estimate assigned to class $y = 1$ by an AdaBoost ensemble F_t on an example $\mathbf{x}$ constitutes a product of experts*

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

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 $\mathbf{x}$.

Proof. Assume an unknown distribution $p(\mathbf{x}, y)$. Define $F^*(\mathbf{x})$ as the population minimiser of an exponential loss function:

$$F^*(\mathbf{x}) \;=\; \arg\min_F E_{\mathbf{x}y}\Big\{e^{-yF(\mathbf{x})}\Big\} \tag{16}$$

To find $F^*(\mathbf{x})$, it is sufficient to minimize $E_{y|\mathbf{x}}\Big\{e^{-yF(\mathbf{x})}\Big\}$, for any $\mathbf{x}$. Therefore,

$$\frac{\partial E_{y|\mathbf{x}}\Big\{e^{-yF(\mathbf{x})}\Big\}}{\partial F(\mathbf{x})} = 0 \implies \frac{\partial(p(y=1|\mathbf{x})e^{-F(\mathbf{x})} + p(y=-1|\mathbf{x})e^{F(\mathbf{x})})}{\partial F(\mathbf{x})} = 0 \implies$$
$$-\,p(y=1|\mathbf{x})e^{-F^*(\mathbf{x})} + p(y=-1|\mathbf{x})e^{F^*(\mathbf{x})} = 0 \implies$$
$$\frac{e^{F^*(\mathbf{x})}}{e^{-F^*(\mathbf{x})}} = \frac{p(y=1|\mathbf{x})}{1-p(y=1|\mathbf{x})} \implies F^*(\mathbf{x}) \;=\; \frac{1}{2}\log\frac{p(y=1|\mathbf{x})}{1-p(y=1|\mathbf{x})} \tag{17}$$

which also implies that

$$p(y=1|\mathbf{x}) \;=\; \frac{1}{1+e^{-2F^*(\mathbf{x})}} \tag{18}$$

Now assume $F^*(\mathbf{x})$ is approximated by an additive model:

$$F^*(\mathbf{x}) \approx F_t(\mathbf{x}) = \sum_{\tau=1}^{t} \alpha_\tau h_\tau(\mathbf{x}) \tag{19}$$

where $\forall \tau$, we have $\alpha_\tau \in \mathbb{R}$ and $h_\tau(\mathbf{x}) \in \{-1, +1\}$, then we have,

$$p(y=1|\mathbf{x}) \approx \hat{p}(y=1|\mathbf{x}) \ = \ \frac{1}{1+e^{-2\sum_{\tau=1}^{t}\alpha_\tau h_\tau(\mathbf{x})}} \tag{20}$$

Adaboost minimises $E_{\mathbf{x}y}\Big\{e^{-yF(\mathbf{x})}\Big\}$ via a greedy stage-wise addition of terms to the model $F_M(\mathbf{x})$, using an empirical risk approximation:

$$E_{\mathbf{x}y}\Big\{e^{-yF(\mathbf{x})}\Big\} \approx \frac{1}{N}\sum_{i=1}^{N} e^{-y_i\sum_{\tau=1}^{t}\alpha_\tau h_\tau(\mathbf{x}_i)} = J_{Ada}$$

Under the greedy optimization scheme, the optimal value for α_τ is

$$\frac{\partial J_{Ada}(\alpha_\tau)}{\partial \alpha_\tau} = 0 \implies \alpha_\tau = \frac{1}{2}\log\frac{1-\epsilon_\tau}{\epsilon_\tau}, \tag{21}$$

where $\epsilon_\tau = \frac{\sum_{i:h_\tau(\mathbf{x}_i)\neq y_i} D_i^\tau}{\sum_{i=1}^{N} D_i^\tau}$ is the weighted error of the weak learner added on round τ.

Substituting α_τ from Eq. (21) into Eq. (20) gives us that the probability estimate assigned to class $y=1$ by an AdaBoost ensemble on an example $\mathbf{x}$ is

$$\hat{p}(y=1|\mathbf{x}) = \frac{1}{1+e^{-2\sum_{\tau=1}^{t}\frac{1}{2}\log\frac{1-\epsilon_\tau}{\epsilon_\tau}h_\tau(\mathbf{x})}}$$

which can be rearranged to

$$\begin{aligned}\hat{p}(y=1|\mathbf{x}) \ &= \ \frac{1}{1+\prod_{\tau=1}^{t}(\frac{\epsilon_\tau}{1-\epsilon_\tau})^{h_\tau(\mathbf{x})}} \\ &= \ \frac{\prod_{\tau=1}^{t}(1-\epsilon_\tau)^{h_\tau(\mathbf{x})}}{\prod_{\tau=1}^{t}(1-\epsilon_\tau)^{h_\tau(\mathbf{x})}+\prod_{\tau=1}^{t}(\epsilon_\tau)^{h_\tau(\mathbf{x})}}.\end{aligned}$$

So the probability estimates of AdaBoost have the form for a product of (unnormalized) experts

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

$$\phi_\tau(y=1|\mathbf{x}) = (1-\epsilon_\tau)^{h_\tau(\mathbf{x})} \tag{22}$$

$$\phi_\tau(y=-1|\mathbf{x}) = \epsilon_\tau^{h_\tau(\mathbf{x})}, \tag{23}$$

which can be normalised to give:

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

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

□

Datasets Used

Table 2. Characteristics of the datasets used in our experiments; number of instances used, number of features and number of classes. The class chosen as 'positive' was the minority class in the original file was chosen. In multiclass datasets, we followed a 1-vs-all approach, where the negative class consisted of uniformly sampled examples from the remaining classes.

Dataset	# Instances	# Features	# Classes
parkinsons	96	22	2
survival	162	3	2
sonar	194	60	2
heart	240	13	2
ionosphere	252	34	2
liver	290	6	2
semeion	322	256	10
congress	336	16	2
wdbc	424	31	2
pima	576	8	2
credit	600	24	2
landsat	1252	36	6
splice	1524	60	3
musk2	2034	166	2
krvskp	3054	36	2
waveform	3306	40	3
spambase	3626	57	2
mushroom	7832	21	2

Additional Experimental Results

Tables of Brier Scores

Table 3. Average *area under the Brier curves* produced by each of the 15 methods examined for all 18 datasets. The area is equal to the average *Brier score*, so lower values are desirable. The lowest value per dataset is marked in bold. AsymAda, AdaMEC & CGAda outperform the other approaches and are in turn outperformed by their calibrated versions. The best performing method overall is *calibrated AsymAda*.

Dataset	CSB0	CSB1	CSB2	AdaC1	AdaC2	AdaC3	AdaCost	AdaCost(β_2)
survival	0.2335	0.2537	0.2326	0.3277	0.2342	0.2341	0.3773	0.3248
ionosphere	0.2292	0.2251	0.2215	0.2830	0.2159	0.2213	0.4272	0.2974
congress	0.1981	0.2131	0.1883	0.0349	0.2036	0.2044	0.2151	0.2222
liver	0.2481	0.2493	0.2448	0.2719	0.2407	0.2446	0.4696	0.3276
pima	0.2396	0.2512	0.2369	0.3129	0.2363	0.2370	0.4241	0.3034
parkinsons	0.2012	0.2332	0.2162	0.2359	0.2199	0.2207	0.4099	0.2799
landsat	0.2132	0.2472	0.2225	0.2131	0.2178	0.2357	0.3619	0.3079
krvskp	0.2265	0.2431	0.2036	0.1838	0.2060	0.2117	0.4175	0.2632
heart	0.2294	0.2435	0.2160	0.2887	0.2180	0.2177	0.3831	0.2836
wdbc	0.2012	0.2117	0.2002	0.1128	0.1993	0.2065	0.2696	0.2482
credit	0.2384	0.2529	0.2370	0.2766	0.2316	0.2321	0.4555	0.3064
sonar	0.2274	0.2290	0.2232	0.2944	0.2216	0.2215	0.4173	0.2953
semeion	0.2111	0.2131	0.1944	0.1431	0.2077	0.2133	0.3581	0.2442
splice	0.2078	0.2325	0.2017	0.1234	0.2073	0.2096	0.3217	0.2495
spambase	0.2279	0.2413	0.2145	0.2343	0.2090	0.2242	0.4109	0.2834
waveform	0.1786	0.2465	0.2103	0.1910	0.2116	0.2108	0.3984	0.2683
musk2	0.2322	0.2394	0.2237	0.2641	0.2186	0.2206	0.4504	0.3126
mushroom	0.2306	0.2350	0.2037	0.1743	0.1965	0.2123	0.4851	0.3205

Dataset	CSAda	AdaMEC	AsymAda	CGAda	Calibrated AdaMEC	Calibrated AsymAda	Calibrated CGAda
survival	0.3593	0.2623	0.2337	**0.2260**	0.2302	0.2334	0.2287
ionosphere	0.3157	0.2043	0.2016	0.2090	0.1642	**0.1333**	0.1814
congress	0.0665	0.0840	0.1040	0.0906	**0.0336**	0.0348	**0.0336**
liver	0.3024	0.2729	**0.2378**	0.2454	0.2485	0.2391	0.2491
pima	0.3383	0.2243	0.2261	0.2297	0.2234	**0.2127**	0.2263
parkinsons	0.2693	0.1727	0.1833	0.1725	0.1388	0.1378	**0.1327**
landsat	0.2452	0.3474	0.1656	0.2065	0.2150	**0.1242**	0.2004
krvskp	0.2143	0.1846	0.1727	0.1859	0.1009	**0.0448**	0.1062
heart	0.3177	0.1643	0.1858	0.1802	0.1471	**0.1450**	0.1532
wdbc	0.1418	0.1230	0.1409	0.1243	0.0537	**0.0505**	0.0579
credit	0.3083	0.2239	0.2202	0.2223	0.2131	**0.2009**	0.2157
sonar	0.3304	0.1969	0.2029	0.2005	0.1826	**0.1823**	0.1839
semeion	0.1788	0.1626	0.1413	0.1600	0.0890	**0.0447**	0.0895
splice	0.1556	0.1286	0.1546	0.1400	0.0683	**0.0500**	0.0622
spambase	0.2663	0.2218	0.1785	0.1981	0.1461	**0.0682**	0.1537
waveform	0.2171	0.1317	0.1380	0.1340	0.0695	**0.0686**	0.0696
musk2	0.2943	0.1963	0.2031	0.1970	0.1432	**0.1225**	0.1344
mushroom	0.2066	0.1686	0.0374	0.1039	0.1071	**0.0353**	0.1118

Table 4. Average *area under the Brier curves* produced by the calibrated versions of AsymAda, AdaMEC & CGAda ensembles of equal ensemble size, for all 18 datasets. The area is equal to the average *Brier score*, so lower values are desirable. The lowest value per dataset is marked in bold. By constraining AsymAda to use as many weak learners as AdaMEC & CGAda, it loses its advantage over the other two methods. In the main paper we provide evidence that the three methods do not significantly differ in performance.

Dataset	Calibrated AdaMEC	Calibrated AsymAda	Calibrated CGAda
survival	0.2337	0.2343	**0.2328**
ionosphere	**0.1711**	0.1994	0.1931
congress	0.0330	0.0358	**0.0328**
liver	0.2494	0.2622	**0.2491**
pima	**0.2268**	0.2338	0.2330
parkinsons	**0.1431**	0.1534	0.1474
landsat	0.2182	0.2421	**0.2137**
krvskp	**0.0991**	0.1405	0.1178
heart	**0.1491**	0.1522	0.1524
wdbc	**0.0557**	0.0626	0.0620
credit	**0.2156**	0.2260	0.2200
sonar	**0.1828**	0.1846	0.1829
semeion	**0.0898**	0.1341	0.1120
splice	**0.0668**	0.1049	0.0729
spambase	**0.1421**	0.2060	0.1699
waveform	0.0699	**0.0688**	0.0702
musk2	0.1397	**0.1367**	0.1408
mushroom	**0.1051**	0.1817	0.1281

Brier Curves

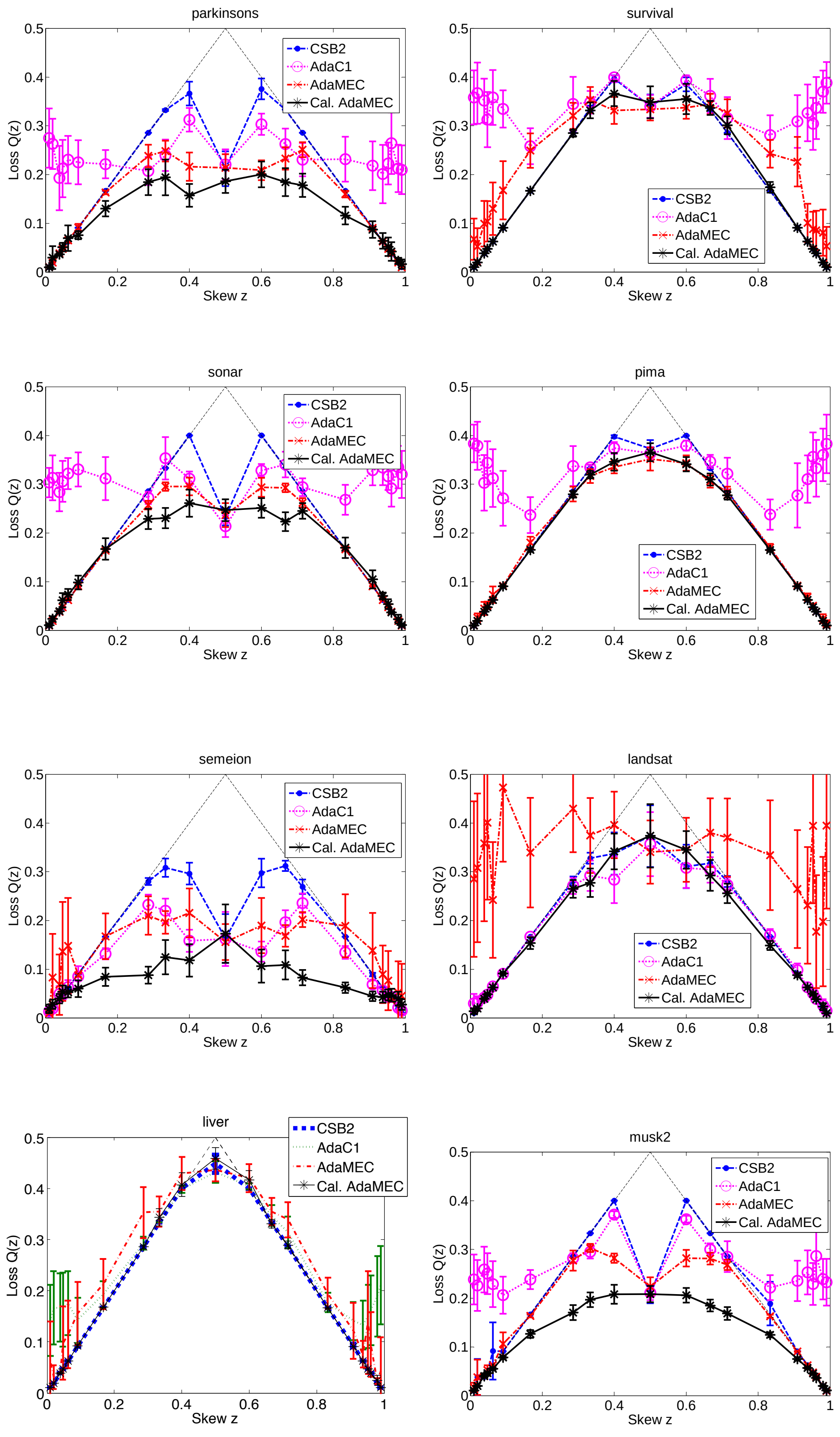

Fig. 1. Loss Q under various degrees of skew z for the datasets included in the study, omitted from the main paper. Lower values indicate better cost-sensitive classification performance. *Calibrated AdaMEC* consistently attains the lowest –or tied for lowest– loss.

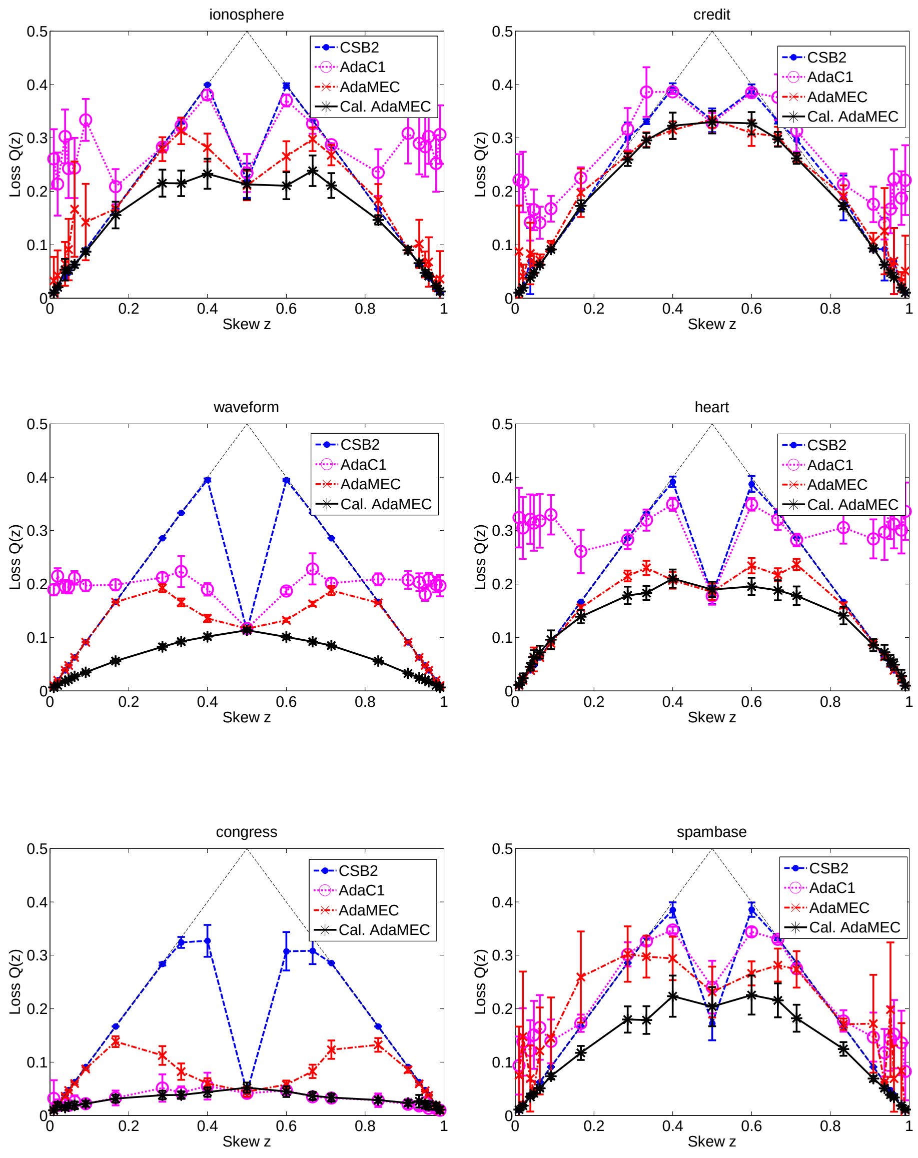

Fig. 2. Loss Q under various degrees of skew z for the datasets included in the study, omitted from the main paper. Lower values indicate better cost-sensitive classification performance. *Calibrated AdaMEC* consistently attains the lowest –or tied for lowest– loss.

Pseudocode for calibrated AdaMEC

In the paper, we chose a 50% / 50% split for Step 1. Step 3.3 was performed using the matlab command *nlinfit* with a tolerance of 10^{-10} and a maximum number of 600 iterations.

Algorithm 1 Platt-Calibrated AdaMEC

```
Input: Number of weak learners $M$, data $\{(\mathbf{x}_i, y_i) | i = 1, \dots, N\}$,
where $y_i \in \{-1, 1\}$, cost of false negatives $c_{FN}$, cost of false positives $c_{FP}$

Training Phase:
    1. Split data into training $D_{tr}$ & calibration set $D_{cal}$
    2. On $D_{tr}$:
        2.1. Train AdaBoost ensemble $F(\mathbf{x}) = \sum_{t=1}^{M} \alpha_t h_t(\mathbf{x})$
    3. On $D_{cal}$:
        3.1. Calculate scores $s(\mathbf{x_i}) = \frac{\sum_{\tau: h_\tau(\mathbf{x_i})=1} \alpha_t}{\sum_{\tau=1}^{t} \alpha_t} \in [0,1], \forall \mathbf{x}_i \in D_{cal}$
        3.2. Calculate the number of positives $N_+$ and negatives $N_-$ in $D_{cal}$
        3.3. Find $A, B$ s. t. $\sum_{i \in D_{cal}} (\hat{p}(y=1|\mathbf{x_i}) - y_i')^2$ is minimized,
             where $\hat{p}(y=1|\mathbf{x}) = \frac{1}{1+e^{As(\mathbf{x})+B}}$ and $y_i' = \begin{cases} \frac{N_+ + 1}{N_+ + 2}, & \text{if } y_i = 1 \\ \frac{1}{N_- + 2}, & \text{if } y_i = -1 \end{cases}$
Prediction Phase:
    4. On new example $\mathbf{x}$:
        4.1. Calculate score $s(\mathbf{x}) = \frac{\sum_{\tau: h_\tau(\mathbf{x})=1} \alpha_t}{\sum_{\tau=1}^{t} \alpha_t} \in [0,1]$
        4.2. Obtain probability estimate $\hat{p}(y=1|\mathbf{x}) = \frac{1}{1+e^{As(\mathbf{x})+B}}$
        4.3. Predict class $H(\mathbf{x}) = sign\left[\hat{p}(y=1|\mathbf{x}) > \frac{c_{FP}}{c_{FP}+c_{FN}}\right]$
```

References

1. L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. *In NIPS 12*, pages 512–518, 2000.