Cost-sensitive Boosting Algorithms: Do we really need them?

Nikolaos Nikolaou[1], Narayanan Edakunni[1], Meelis Kull[2], Peter Flach[2] and Gavin Brown[1]
[1]School of Computer Science, University of Manchester; [2]Department of Computer Science, University of Bristol
nikolaos.nikolaou@manchester.ac.uk

Basically, no.

We analyse **20** years of literature, with the axioms of **4** distinct frameworks:

Functional Gradient Descent *Decision Theory* *Margin Theory* *Probabilistic modelling*

From **15+** boosting variants over 20 years:

... only **3** are consistent with all axioms... and even then, only if we **calibrate** their outputs...

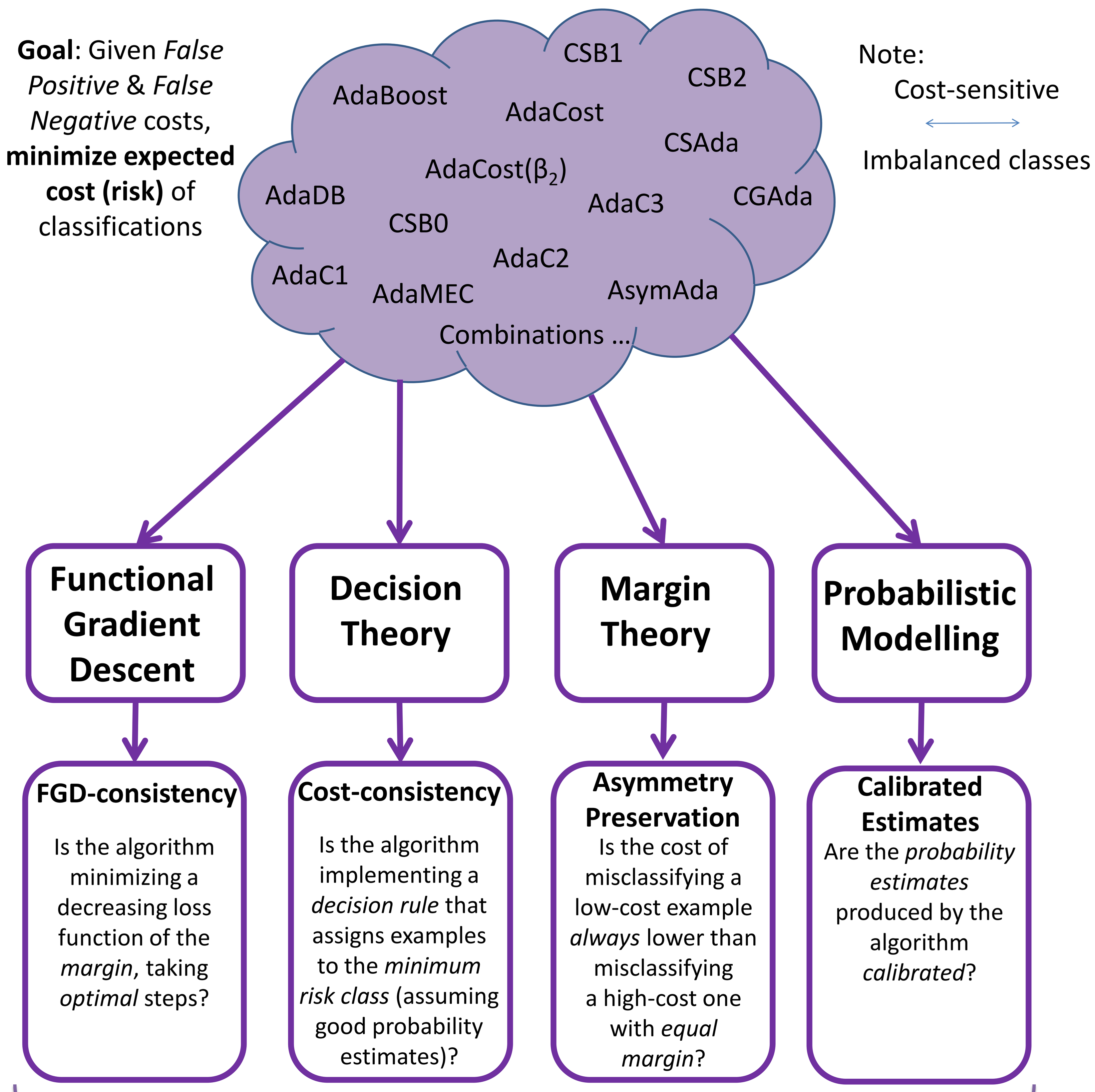
Final recommendation – use the ORIGINAL (Freund & Schapire 1997) and calibrate it.

Now... read on...

A Unified Perspective

Goal: Given *False Positive* & *False Negative* costs, **minimize expected cost (risk)** of classifications

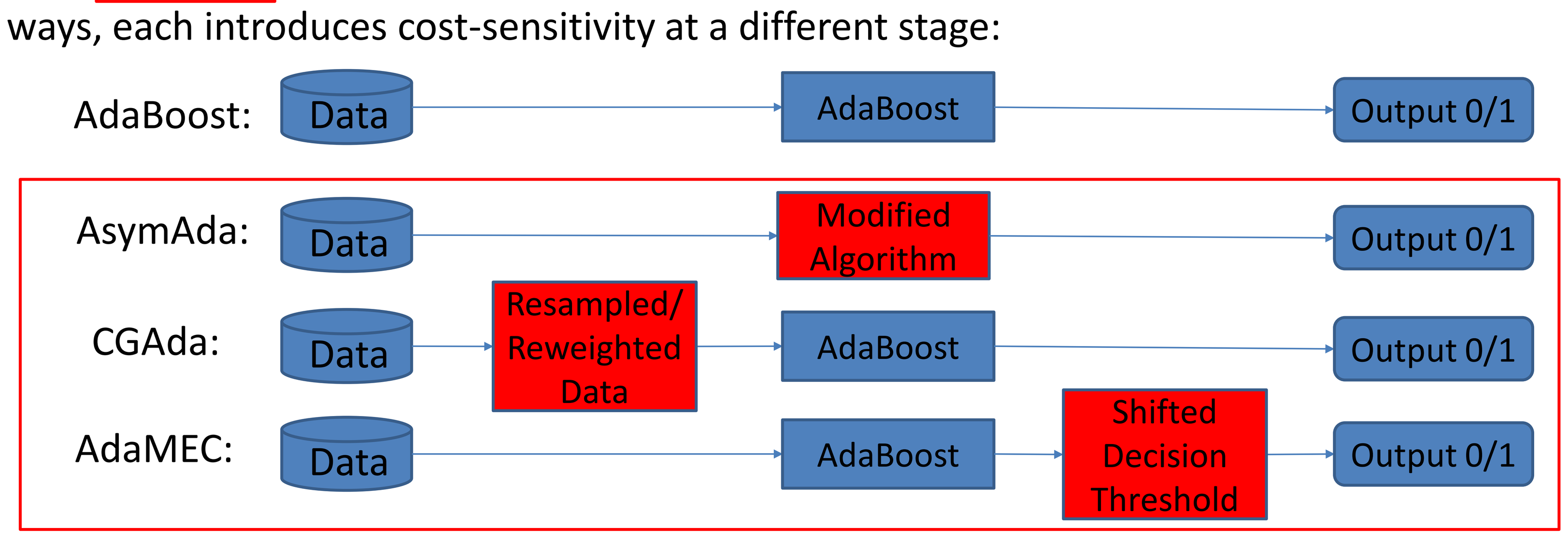
Note: Cost-sensitive ⟷ Imbalanced classes



Method	FGD-consistent	Cost-consistent	Asymmetry-preserving	Calibrated estimates
AdaBoost (Freund & Schapire 1997)	✓		✓	
AdaCost (Fan et al. 1999)				
AdaCost(β_2) (Ting 2000)				
CSB0 (Ting 1998)			✓	
CSB1 (Ting 2000)			✓	
CSB2 (Ting 2000)			✓	
AdaC1 (Sun et al. 2005, 2007)		✓		
AdaC2 (Sun et al. 2005, 2007)	✓		✓	
AdaC3 (Sun et al. 2005, 2007)				
CSAda (Mashnadi-Shirazi & Vasconselos 2007, 2011)	✓	✓		
AdaDB (Landesa-Vázquez & Alba-Castro 2013)	✓	✓		
AdaMEC (Ting 2000, Nikolaou & Brown 2015)	✓	✓	✓	
CGAda (Landesa-Vázquez & Alba-Castro 2012, 2015)	✓	✓	✓	
AsymAda (Viola & Jones 2002)	✓	✓	✓	

All boosting algorithms produce **uncalibrated** probability estimates (**scores**)

Only 3 variants satisfy all other properties – all approximate the same model in different ways, each introduces cost-sensitivity at a different stage:



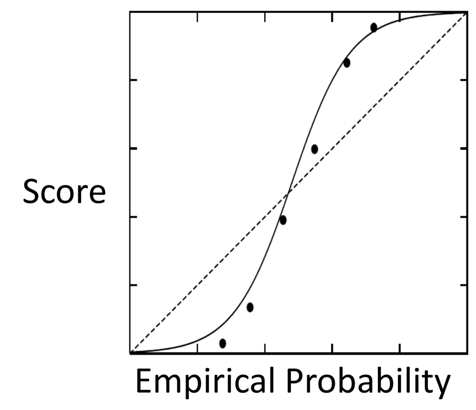
Once **calibrated**, AdaMEC, CGAda & AsymAda **satisfy all properties**:

Calibrated AdaMEC	✓	✓	✓	✓
Calibrated CGAda	✓	✓	✓	✓
Calibrated AsymAda	✓	✓	✓	✓

Calibration

The mapping of scores to empirical probabilities exhibits a **sigmoid** distortion

- **Platt scaling** (logistic calibration) to correct – need separate training & calibration sets

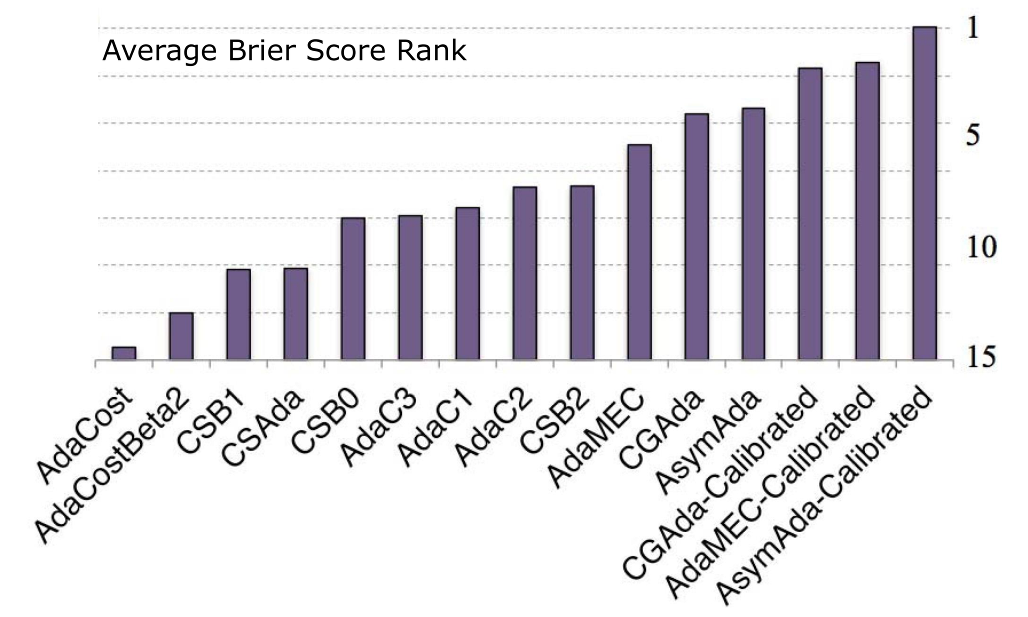


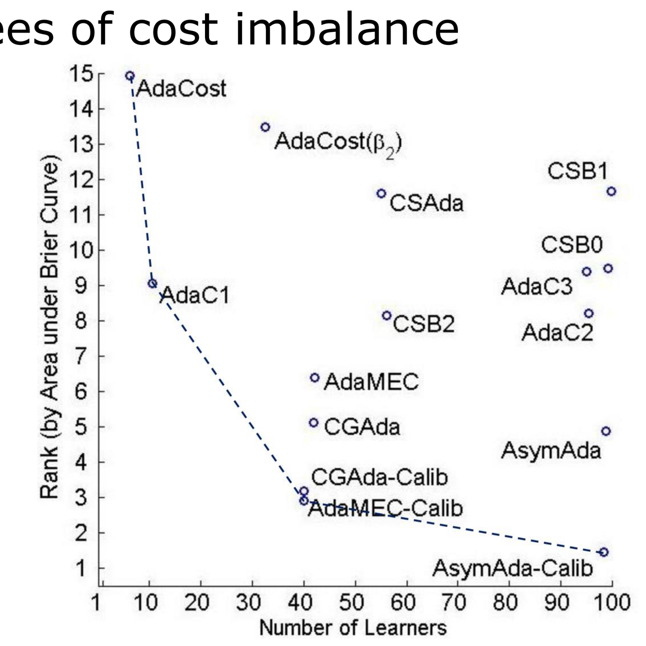
Find A, B for mapping raw scores $s(\boldsymbol{x})$ to calibrated probability estimates

$$\hat{p}(y=1|\mathbf{x}) = \frac{1}{1+e^{As(\mathbf{x})+B}}$$

Results

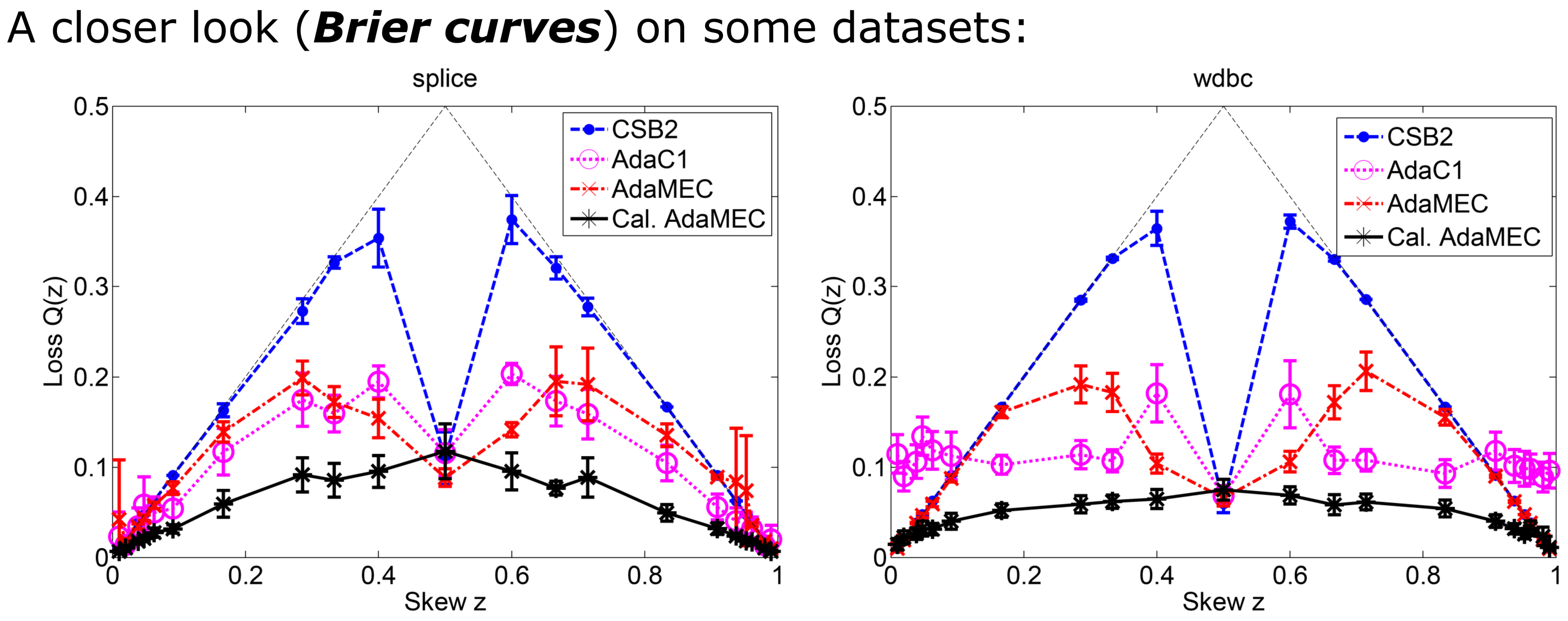
Experiments on **18** datasets, across **21** degrees of cost imbalance





- AdaMEC, CGAda & AsymAda **outperform all others**
- Their **calibrated** versions **outperform** the **uncalibrated** ones
- Among the 3, **AsymAda lowest *Brier score***, but uses **more weak learners**
- **Fixing Num. weak learners** AdaMEC, CGAda & AsymAda **similar performance**
- Above findings are **supported by statistical significance tests**

A closer look (***Brier curves***) on some datasets:



Advice for Practitioners

Based on theoretical soundness, flexibility, simplicity & results: **Calibrated AdaMEC**

Input: Number of weak learners M, data $\{(\mathbf{x}_i, y_i)|i=1,\ldots,N\}$, 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} — **Reserve part of the training data for calibration.**
2. On D_{tr}:
   2.1. Train AdaBoost ensemble $F(\mathbf{x}) = \sum_{t=1}^{M} \alpha_t h_t(\mathbf{x})$ — **Train original AdaBoost ensemble on training set.**
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} — **Train sigmoid parameters on calibration set.**
   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 $\mathbf{x}$: — **Obtain a score for the test example.**
   4.1. Calculate *non-prior-weighted* 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 *non-prior-weighted* probability estimate $\hat{p}(y=1|\mathbf{x}) = \frac{1}{1+e^{As(\mathbf{x})+B}}$ — **Calibrate score.**
   4.3. Predict class $H(\mathbf{x}) = sign\left[\hat{p}(y=1|\mathbf{x}) > \frac{c_{FP}}{c_{FP}+c_{FN}}\right]$ — **Use shifted decision threshold for predictions.**

Acknowledgements: NN & GB were supported by the EPSRC grants [EP/I028099/1 & EP/L000725/1], MK & PF were supported by the EPSRC grant [EP/K018728/1].

Implementation in Matlab available online at: http://www.cs.man.ac.uk/~gbrown/software/