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

Machine Learning Journal, Vol. 104, Issue 2, Sept 2016

Nikolaos Nikolaou, Narayanan Edakunni, Meelis Kull, Peter Flach and Gavin Brown

MANCHESTER 1824
The University of Manchester

University of BRISTOL

Adaboost (Freund & Schapire 1997)

Ensemble method - rich theoretical depth.

Train models sequentially.

Each model focuses on examples previously misclassified.

Combine by majority vote.

Define a distribution over the training set, $D_1(i) = \frac{1}{N}, \forall i.$

for $t = 1$ to T **do**

Build a classifier h_t from the training set, using distribution D_t.

Set $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ — **Majority voting confidence in classifier t**

Update D_{t+1} from D_t :

Set $D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$ — **Distribution update**

end for

$H(x') = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x')\right)$ — **Majority vote on test example x'**

Adaboost

How will it work on cost sensitive problems?

$$\begin{bmatrix} 0 & c_{FN} \\ c_{FP} & 0 \end{bmatrix}$$

i.e. with differing cost for a False Positive / False Negative …

…does it minimize the expected cost?

Cost sensitive Adaboost…

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)

15+ boosting variants over **20** years

Cost sensitive Adaboost...

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)

15+ boosting variants over **20** years

Some **re-invented** multiple times

Cost sensitive Adaboost…

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)

15+ boosting variants over **20** years

Some **re-invented** multiple times

Most proposed as **heuristic** modifications to original AdaBoost

Cost sensitive Adaboost...

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)

15+ boosting variants over **20** years

Some **re-invented** multiple times

Most proposed as **heuristic** modifications to original AdaBoost

Many treat FP/FN costs as **hyperparameters**

A step back… Why is Adaboost interesting?

Functional Gradient Descent (Mason et al., 2000)

Decision Theory (Freund & Schapire, 1997)

Margin Theory (Schapire et al., 1998)

Probabilistic Modelling (Lebanon & Lafferty 2001; Edakunni et al 2011)

$$\text{Set } \alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

$$\text{Update } D_{t+1} \text{ from } D_t :$$

$$\text{Set } D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

So for a cost sensitive boosting algorithm…

My new algorithm

Functional Gradient Descent

Decision Theory

Margin Theory

Probabilistic Modelling

"Does my new algorithm still follow from each?"

$$\text{Set } \alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$$

$$\text{Update } D_{t+1} \text{ from } D_t :$$

$$\text{Set } D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

So for a cost sensitive boosting algorithm…

	My new algorithm
Functional Gradient Descent	✓
Decision Theory	✓
Margin Theory	✓
Probabilistic Modelling	✓

"Does my new algorithm still follow from each?"

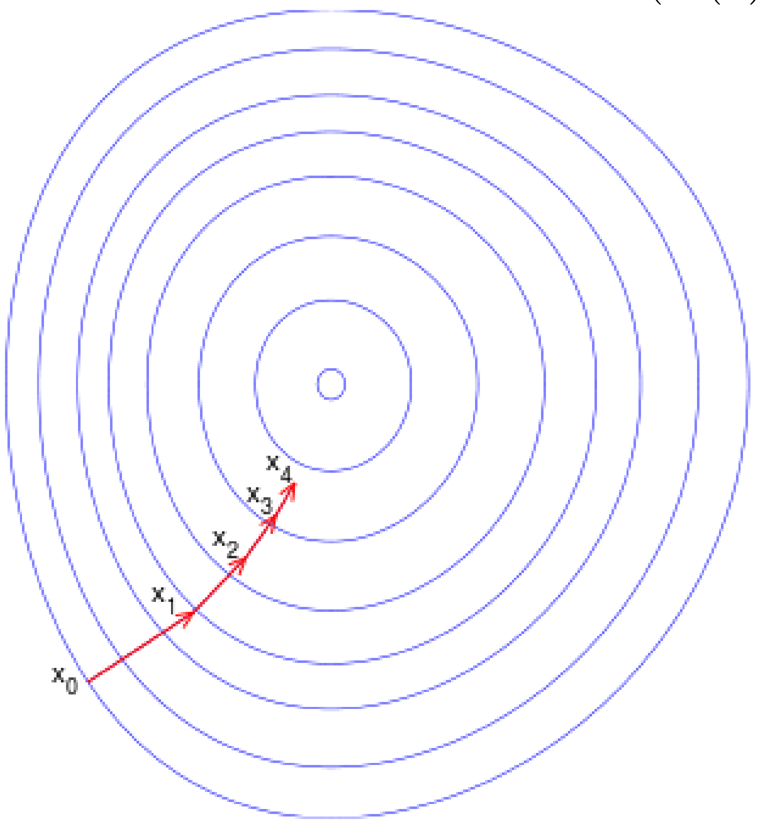
$$\text{Set } \alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

$$\text{Update } D_{t+1} \text{ from } D_t :$$

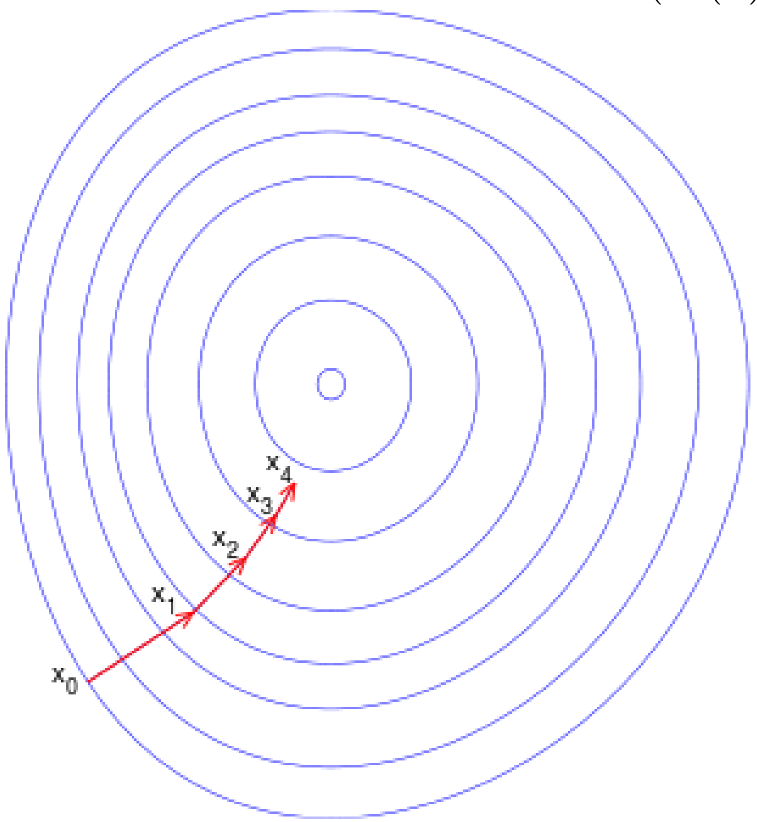
$$\text{Set } D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

Functional Gradient Descent

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



Functional Gradient Descent



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

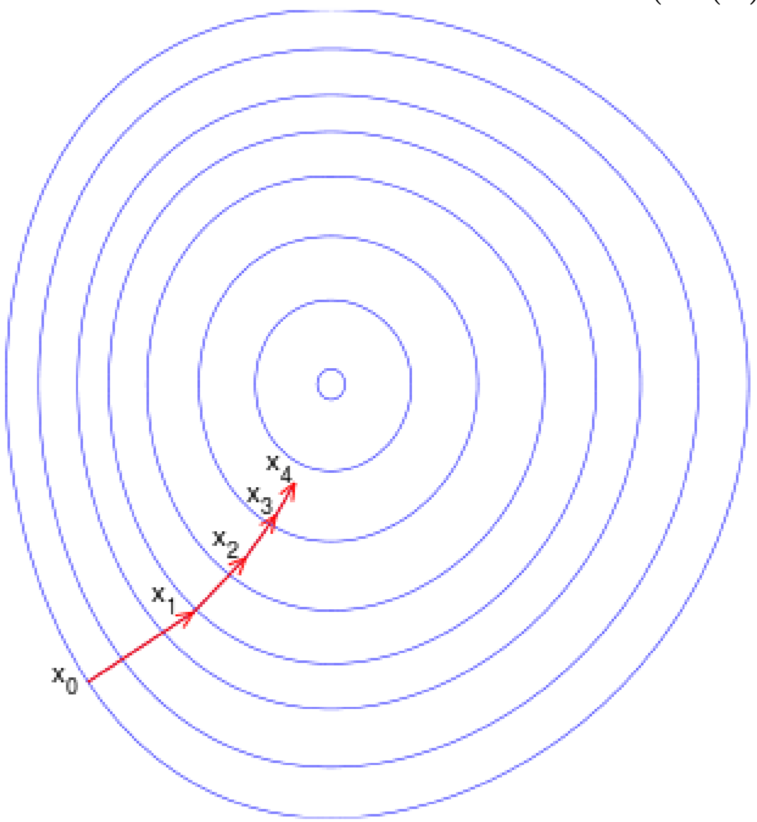
$$D_i^{t+1} = \frac{\frac{\partial}{\partial y_i F_t(\mathbf{x}_i)} J(F_t(\mathbf{x}))}{\sum_{j=1}^{N} \frac{\partial}{\partial y_j F_t(\mathbf{x}_j)} J(F_t(\mathbf{x}))}$$

Functional Gradient Descent

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

Direction in function space

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

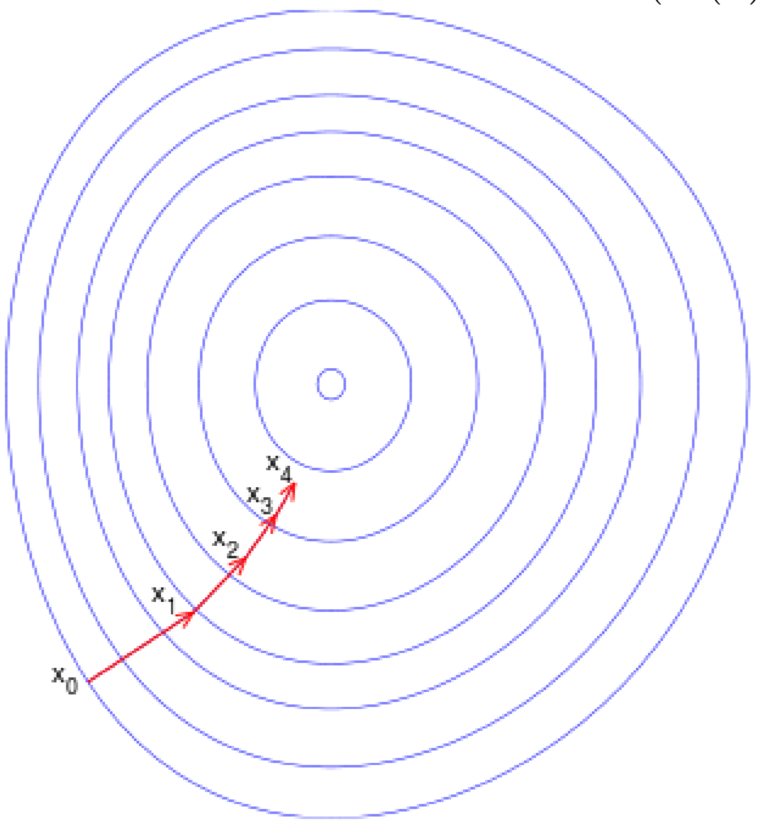


Functional Gradient Descent

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

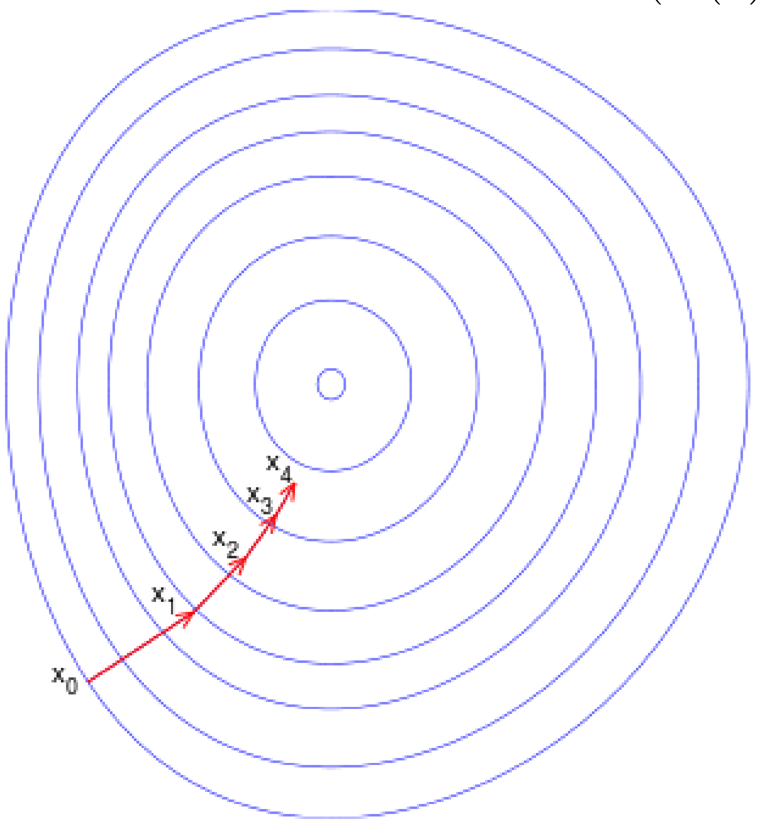
Direction in function space

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



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

Functional Gradient Descent



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

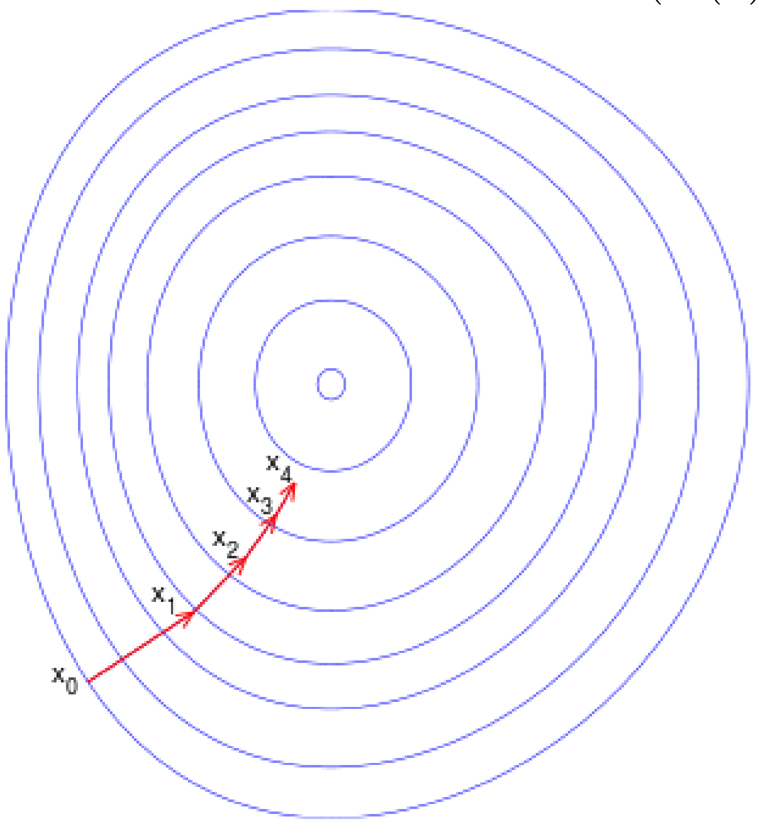
Direction in function space

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

Step size

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

Functional Gradient Descent



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

Direction in function space

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

Step size

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

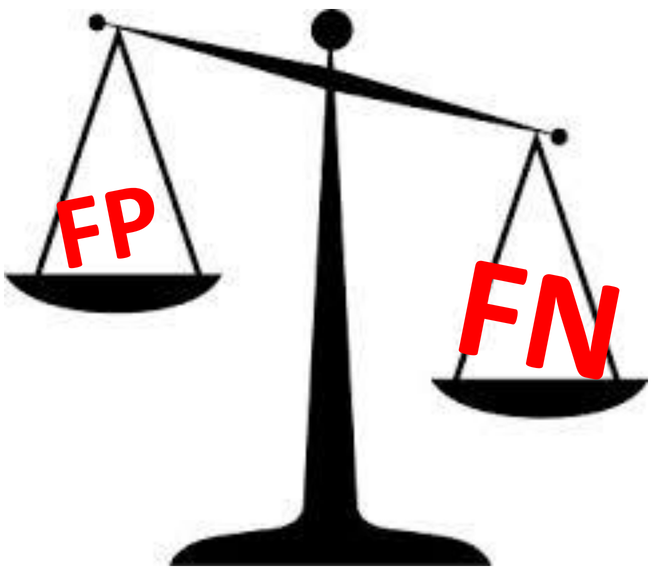
Property: FGD-consistency

Are the voting weights and distribution updates consistent with each other?

(i.e. both derivable from FGD on a given loss)

Decision Theory

Ideally: Assign each example to **risk-minimizing** class.

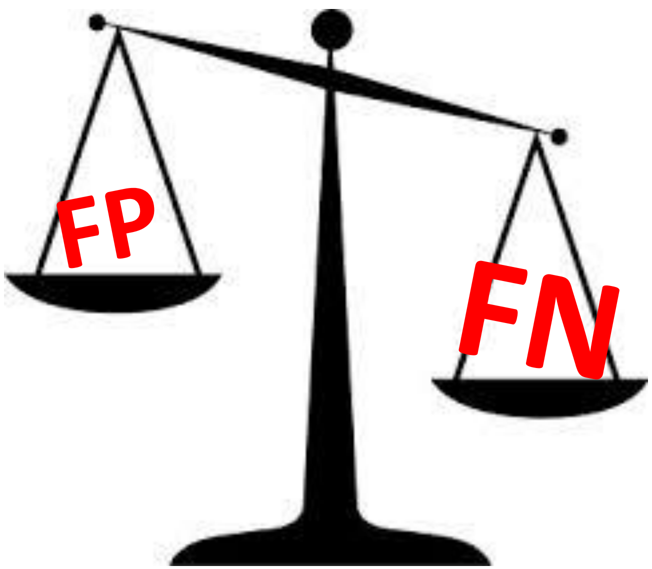


$$\hat{p}(y = 1|\mathbf{x}) \quad > \quad \frac{c_{FP}}{c_{FP} + c_{FN}}$$

$$\begin{bmatrix} 0 & c_{FN} \\ c_{FP} & 0 \end{bmatrix}$$

Decision Theory

Ideally: Assign each example to **risk-minimizing** class.



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

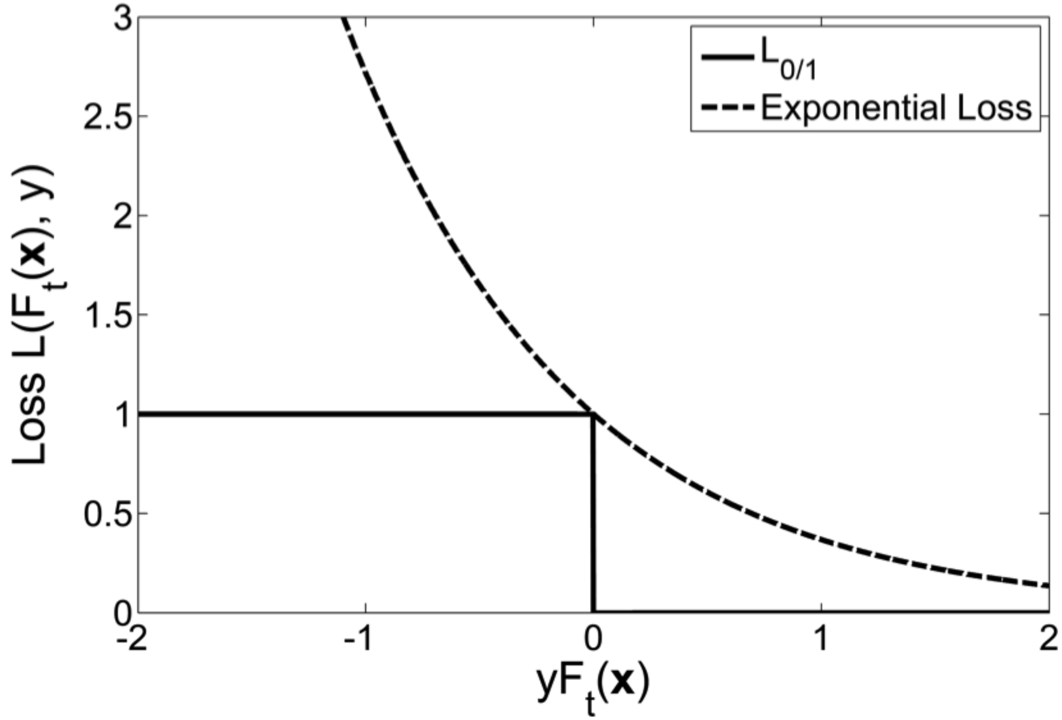
$$\begin{bmatrix} 0 & c_{FN} \\ c_{FP} & 0 \end{bmatrix}$$

Property: Cost-consistency

Does the algorithm use the above (Bayes optimal) rule to make decisions?

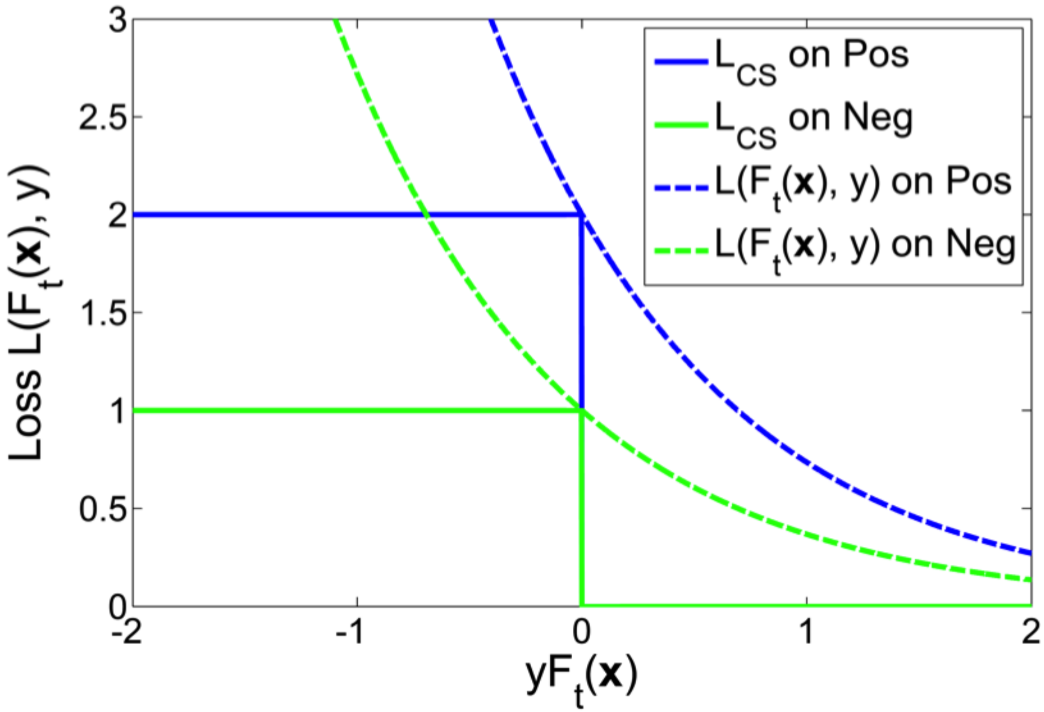
(assuming good probability estimates)

Margin Theory



Large margins encourage small **generalization error**.
Adaboost promotes **large margins**.

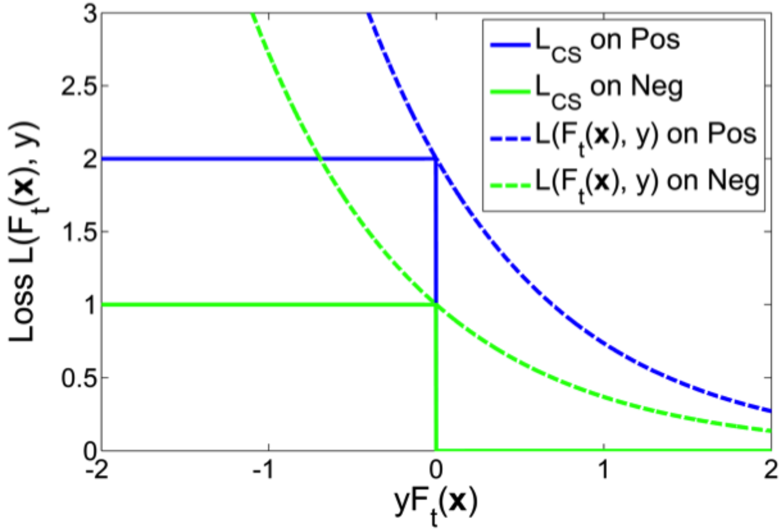
Margin Theory – with costs...



Different surrogate losses for each class.

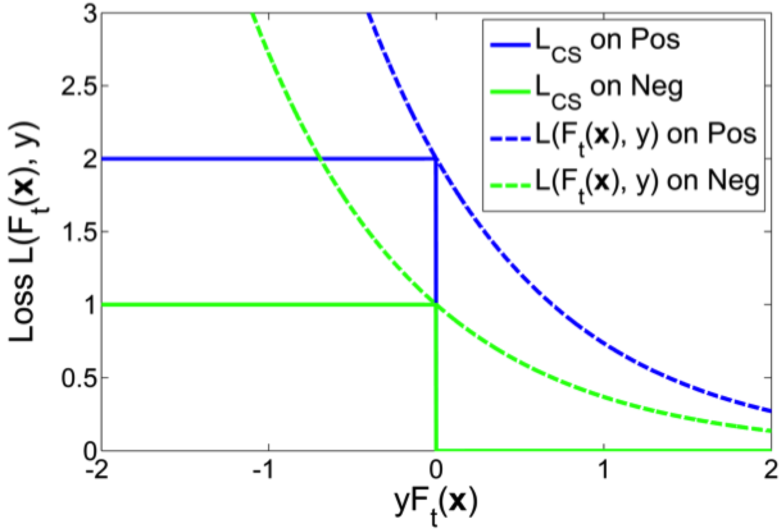
So for a cost sensitive boosting algorithm…

We expect this to be the case.

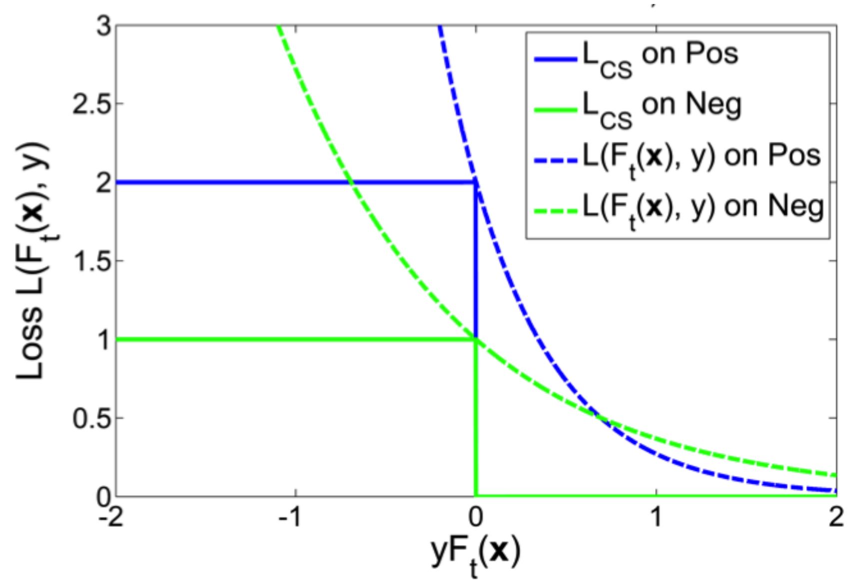


So for a cost sensitive boosting algorithm…

We expect this to be the case.

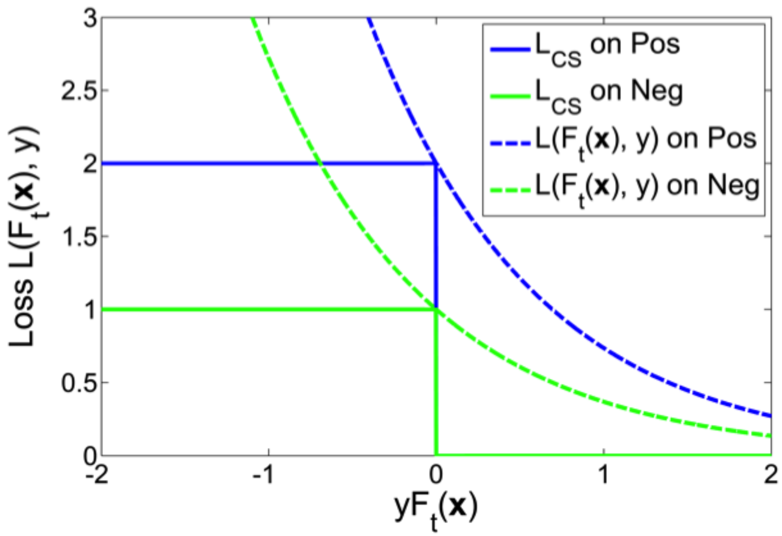


But some algorithms do this…

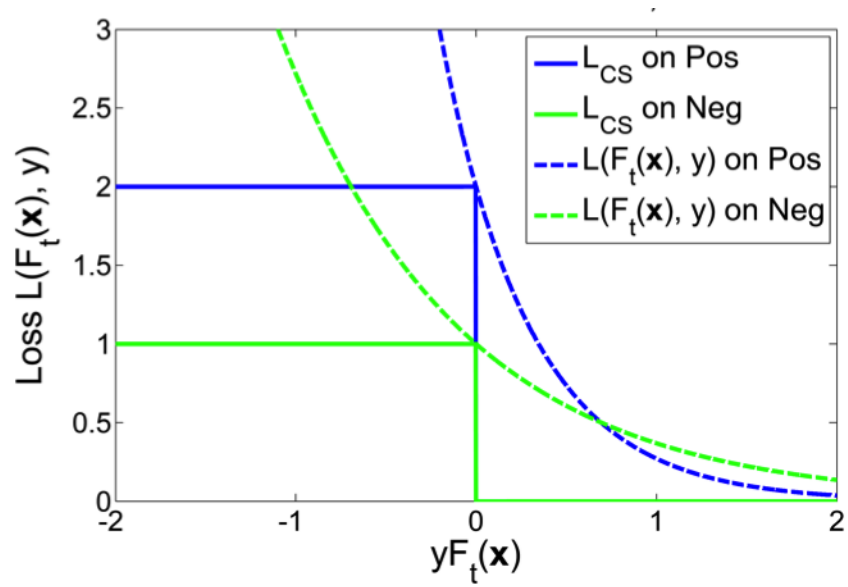


So for a cost sensitive boosting algorithm…

We expect this to be the case.



But some algorithms do this…



Property: Asymmetry preservation

Does the loss function preserve the **relative** importance of each class, for all margin values?

Probabilistic Models

Probabilistic Models

‘AdaBoost does not produce good probability estimates.’

Niculescu-Mizil & Caruana, 2005

Probabilistic Models

‘AdaBoost does not produce good probability estimates.’

Niculescu-Mizil & Caruana, 2005

‘AdaBoost is successful at [..] classification [..] but not class probabilities.’

Mease et al., 2007

Probabilistic Models

‘AdaBoost does not produce good probability estimates.’

Niculescu-Mizil & Caruana, 2005

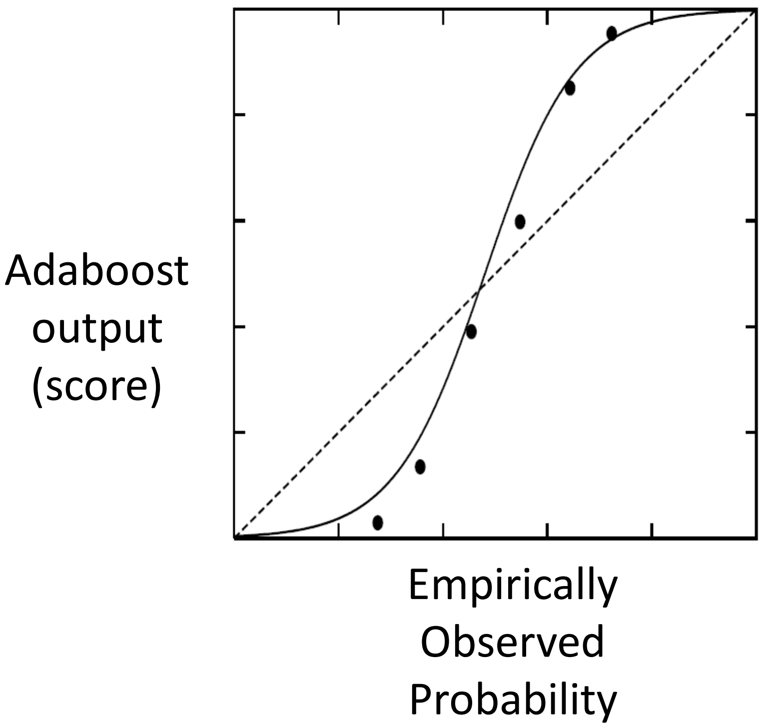
‘AdaBoost is successful at [..] classification [..] but not class probabilities.’

Mease et al., 2007

‘This increasing tendency of [the margin] impacts the probability estimates by causing them to quickly diverge to 0 and 1.’

Mease & Wyner, 2008

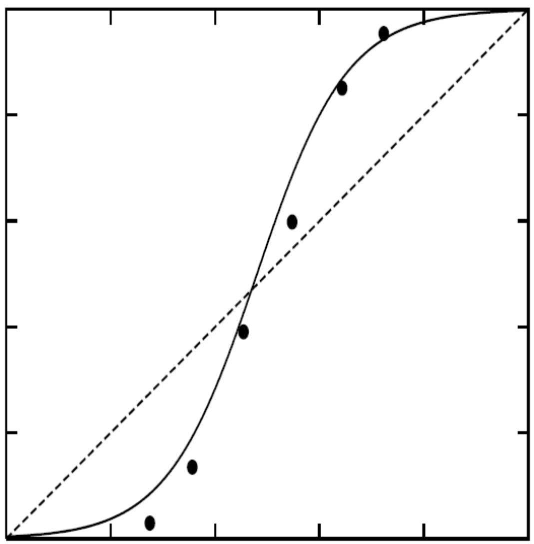
Probabilistic Models



Adaboost tends to produce over- or under-estimates of probabilities.

Probabilistic Models

Adaboost output (score)



Empirically Observed Probability

Adaboost tends to produce over- or under-estimates of probabilities.

Property: Calibrated estimates

Does the algorithm generate "calibrated" probability estimates?

The results are in...

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)	✓	✓	✓	

The results are in…

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)			✓	**All algorithms** produce **uncalibrated** probability estimates!
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)	✓	✓	✓	

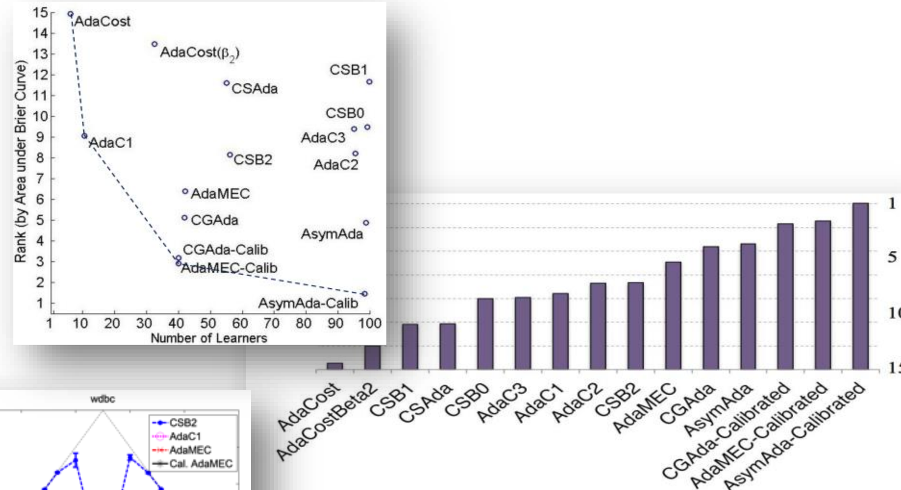
The results are in...

Method	FGD-consistent	Cost-consistent	Asymmetry-preserving	Calibrated estimates
AdaBoost (Freund & Schapire 1997)	✓		✓	**All algorithms** produce **uncalibrated** probability estimates!
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)	✓	✓	✓	

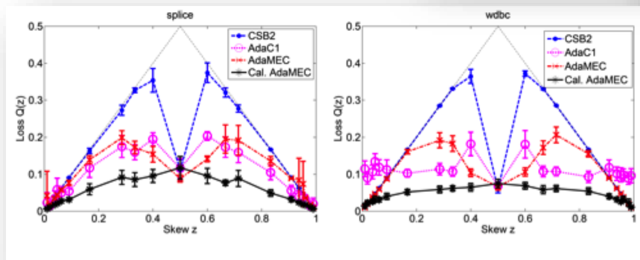
So could we just calibrate these last three? We use “Platt scaling”.

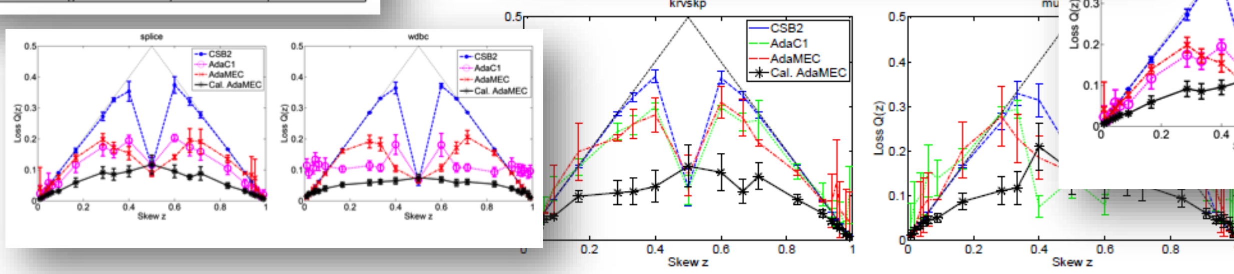
Experiments

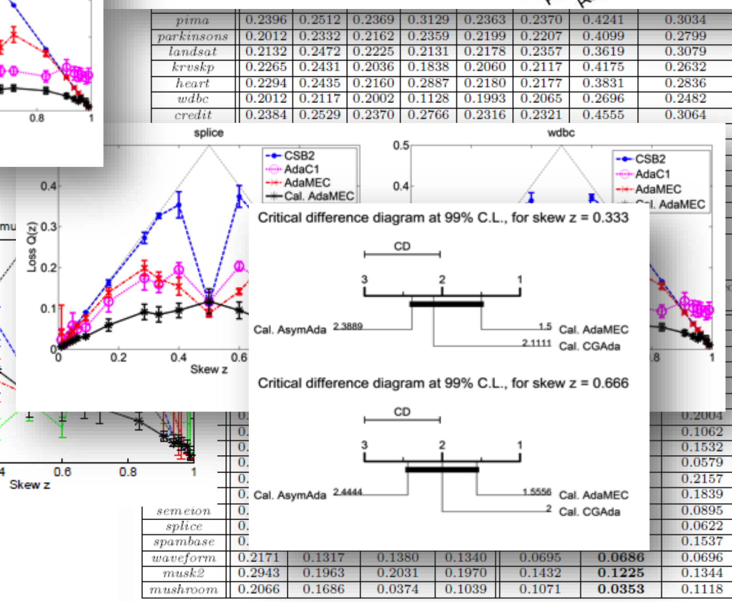
15 algorithms.
18 datasets.
21 degrees of cost imbalance.



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

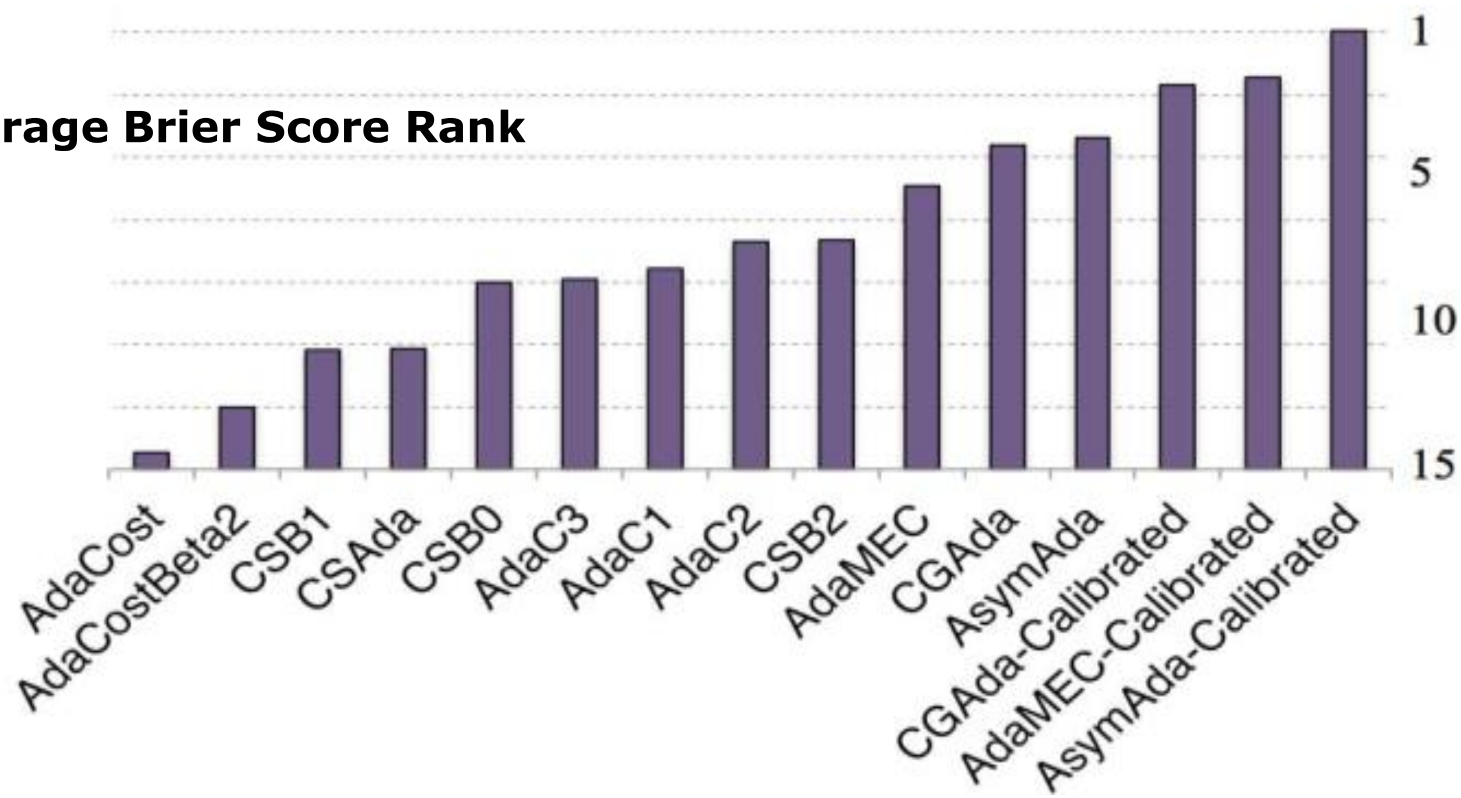






In summary…

Average Brier Score Rank



AdaMEC, CGAda & AsymAda **outperform all others.**

Their **calibrated** versions **outperform** the **uncalibrated** ones

In summary…

“Calibrated-AdaMEC” was one of the top methods.

In summary…

"Calibrated-AdaMEC" was one of the top methods.

1. Take original Adaboost.

2. Calibrate it (we use Platt scaling)

3. Shift the decision threshold…. : $\frac{c_{FP}}{c_{FP} + c_{FN}}$

In summary…

“Calibrated-AdaMEC” was one of the top methods.

1. Take <u>original</u> Adaboost.

2. Calibrate it (we use Platt scaling)

3. Shift the decision threshold…. : $\frac{c_{FP}}{c_{FP} + c_{FN}}$

Consistent with all theory perspectives.

No extra **hyperparameters** added.

No need to retrain if cost ratio changes.

Consistently **top (or joint top)** in empirical comparisons.

Conclusions

We analyzed the cost-sensitive boosting literature

… **15+** variants over **20** years, from **4** different theoretical perspectives

Conclusions

We analyzed the cost-sensitive boosting literature

… **15+** variants over **20** years, from **4** different theoretical perspectives

"Cost sensitive" modifications to the **original** Adaboost are not needed...

… if the scores are properly calibrated,
and the decision rule is shifted according to the cost matrix.

Conclusions

We analyzed the cost-sensitive boosting literature

… **15+** variants over **20** years, from **4** different theoretical perspectives

"Cost sensitive" modifications to the **original** Adaboost are not needed...

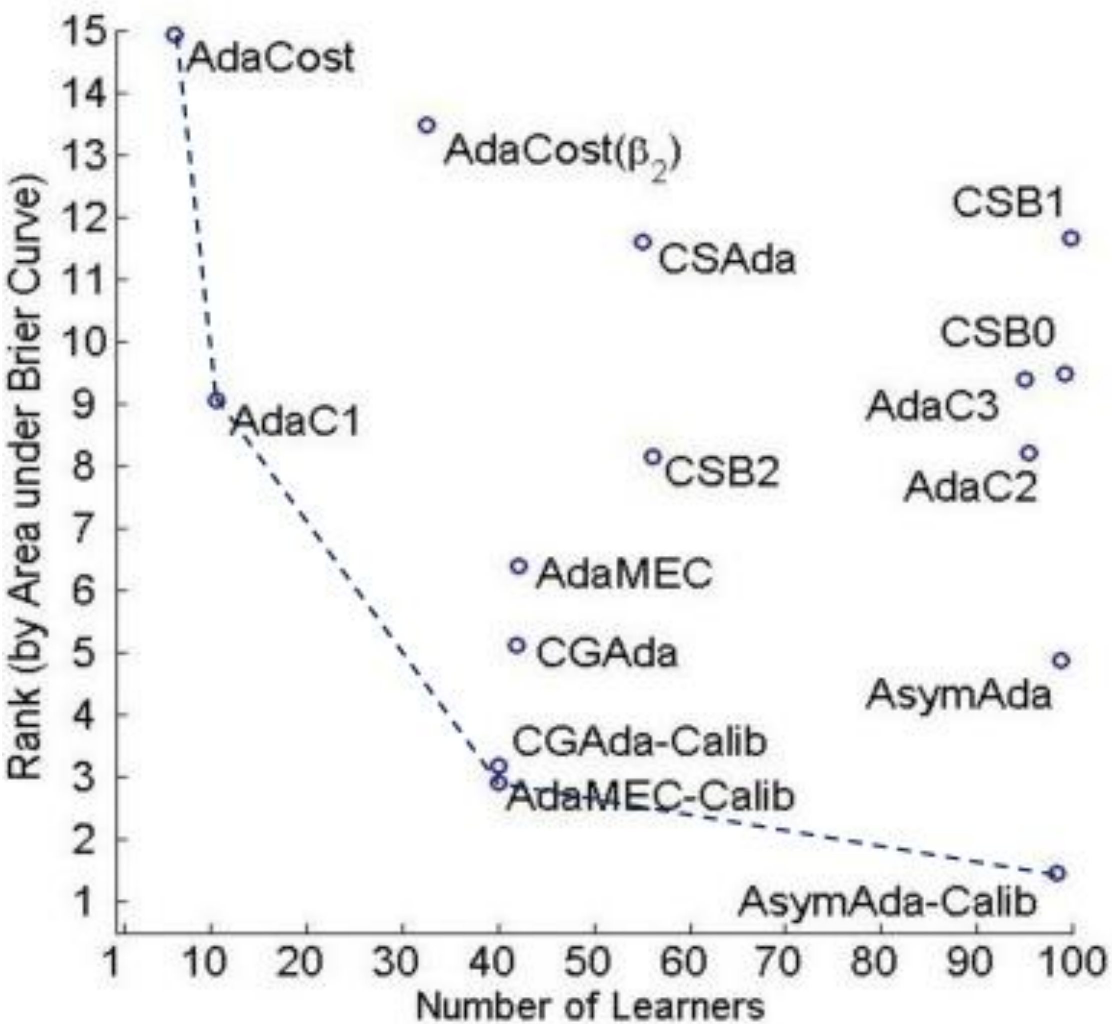
… if the scores are properly calibrated,
and the decision rule is shifted according to the cost matrix.

Thank you!

Grazie!

Area under the Brier (cost) curve

Look at the pareto front!



A closer look at the Brier Curves.