Boosting for Probability Estimation & Cost-Sensitive Learning

Nikos Nikolaou

EPSRC Doctoral Prize Fellow, University of Manchester

EPSRC

Engineering and Physical Sciences Research Council

MANCHESTER 1824

The University of Manchester

Part I:
What is wrong with cost-sensitive Boosting?

Boosting

Can we turn a **weak learner** into a **strong learner**? (Kearns, 1988)

Marginally more accurate than random guessing

Arbitrarily high accuracy

YES! **‘Hypothesis Boosting’** (Schapire, 1990)

AdaBoost (Freund & Schapire, 1997)

Gödel Prize 2003

Adaboost (Freund & Schapire 1997)

Ensemble method – very successful, rich theoretical depth.

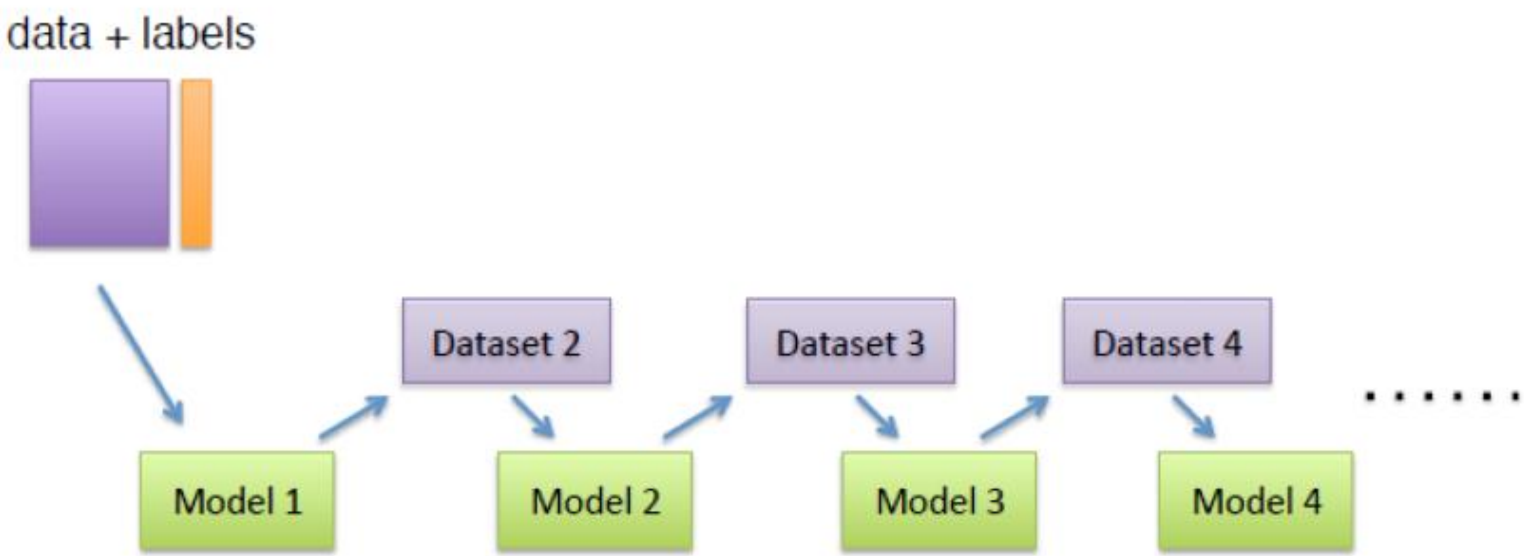
Train models **sequentially**.

Each model **focuses on examples previously misclassified**.

Combine by **weighted majority vote**.

AdaBoost: training

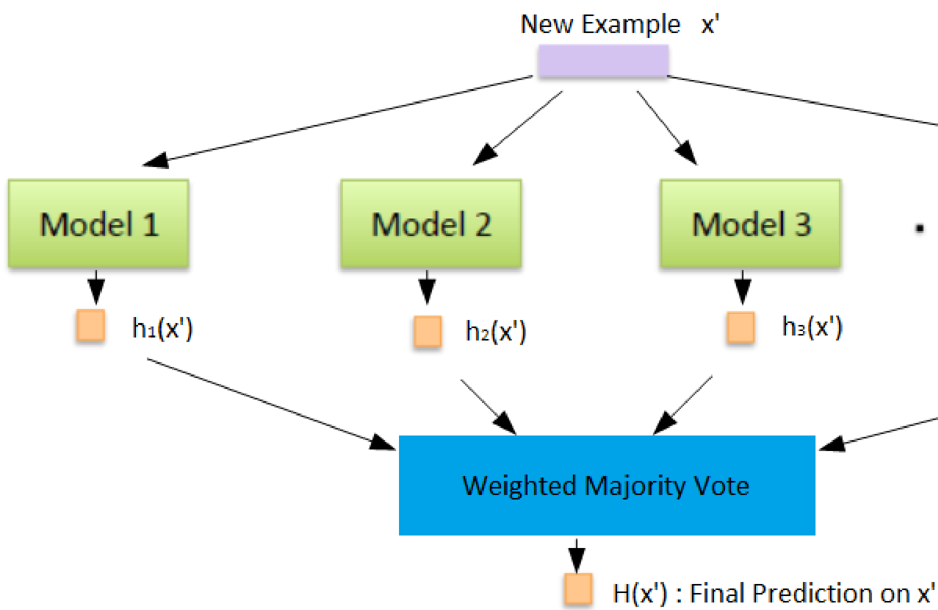
Construct strong model **sequentially** by combining multiple weak models



Each model tries to **correct the mistakes of the previous one**

AdaBoost: predictions

Prediction: **weighted majority vote** among M weak learners



AdaBoost: algorithm

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** (a.k.a. **risk**)?

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)

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

Update D_{t+1} 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	✓

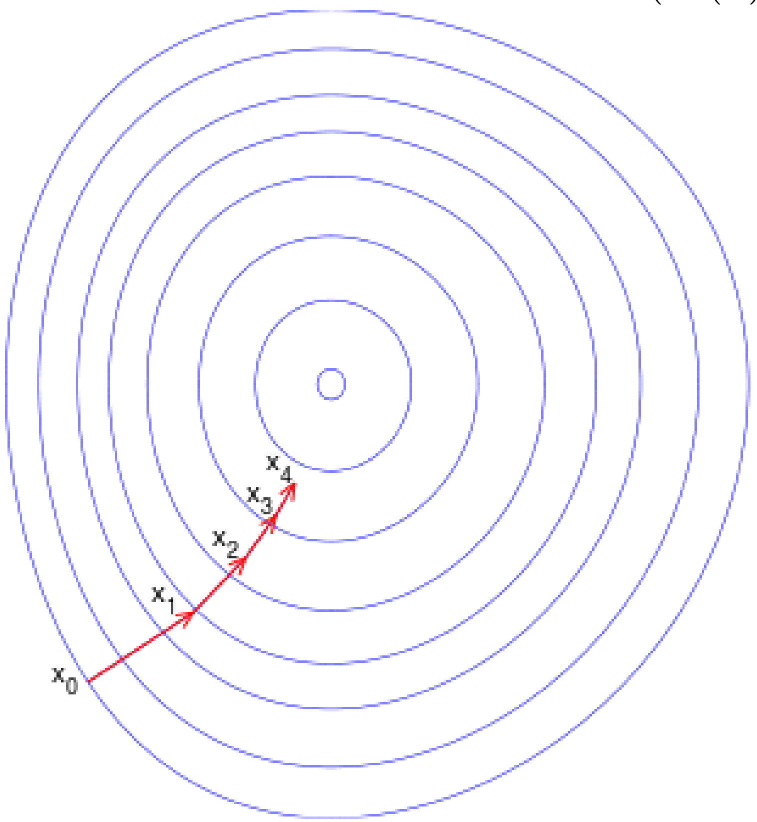
"Does my new algorithm still follow from each?"

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

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

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

Functional Gradient Descent



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

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

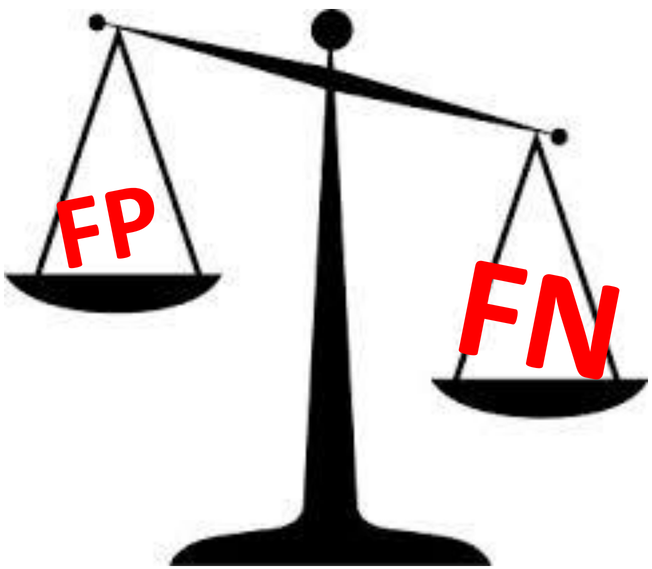
$$\alpha_t^* = \arg\min_{\alpha_t} \left[\frac{1}{N}\sum_{i=1}^{N} L\Big(y_i(F_{t-1}(\mathbf{x}_i) + \alpha_t h_t(\mathbf{x}_i))\Big)\right].$$

Property: FGD-consistency

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

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

Decision theory



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

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

Predict class $y = 1$ *iff*

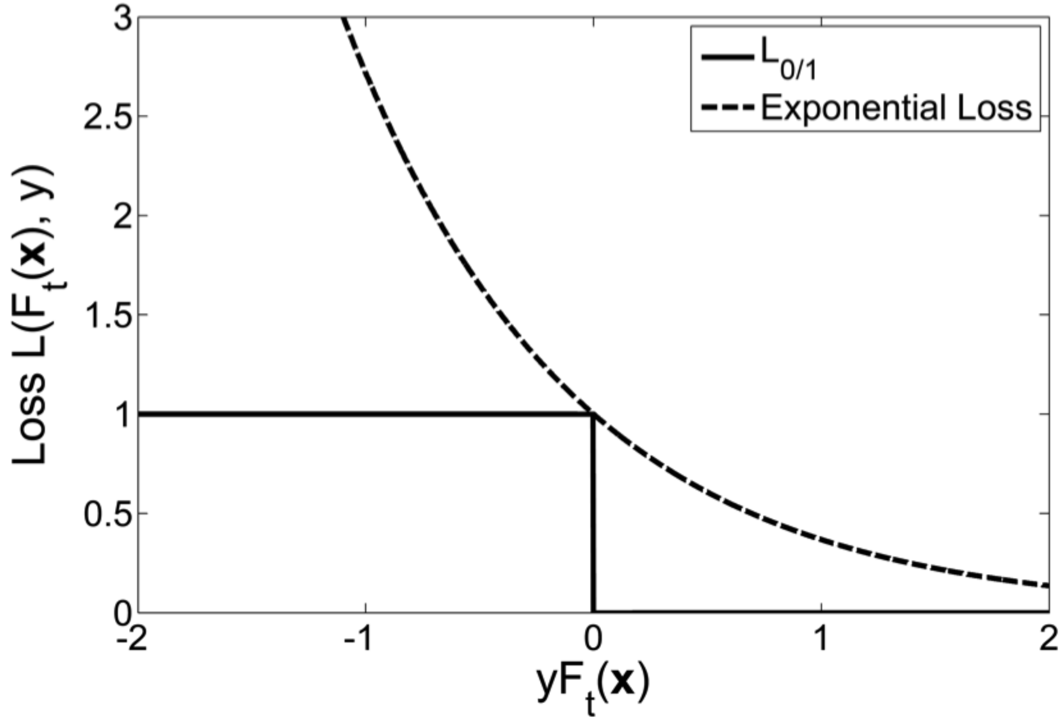
$$\hat{p}(y = 1|\mathbf{x}) > \frac{c_{FP}}{c_{FP} + c_{FN}}$$

Property: Cost-consistency

Does the algorithm use the above
(Bayes Decision Rule)
to make decisions?

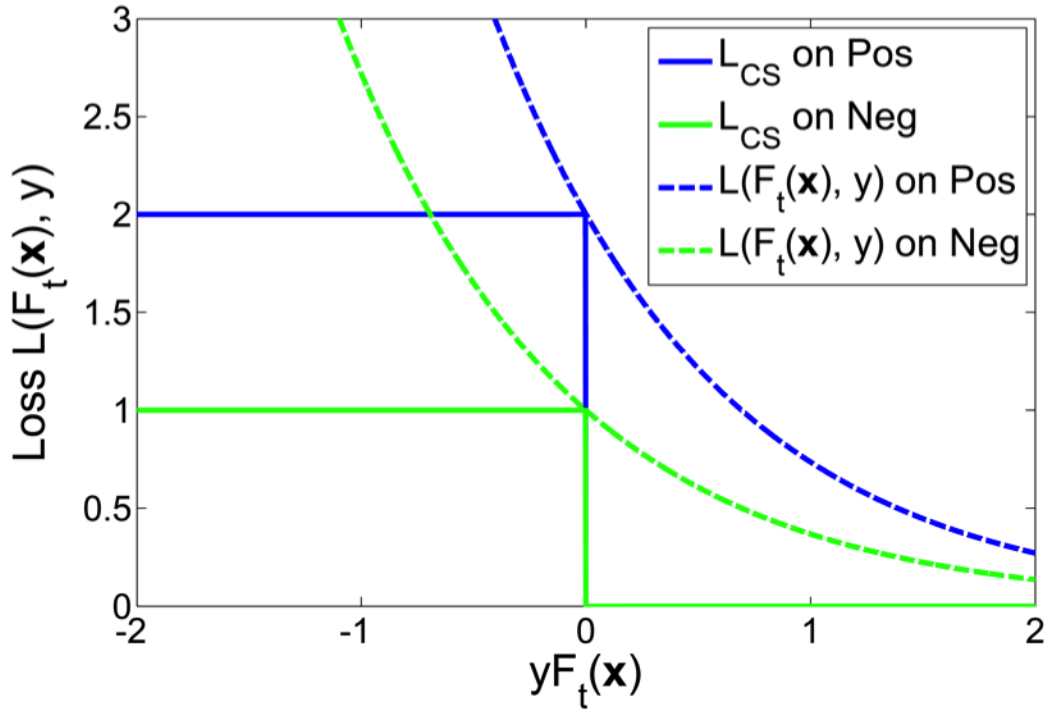
(assuming 'good' probability estimates)

Margin theory



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

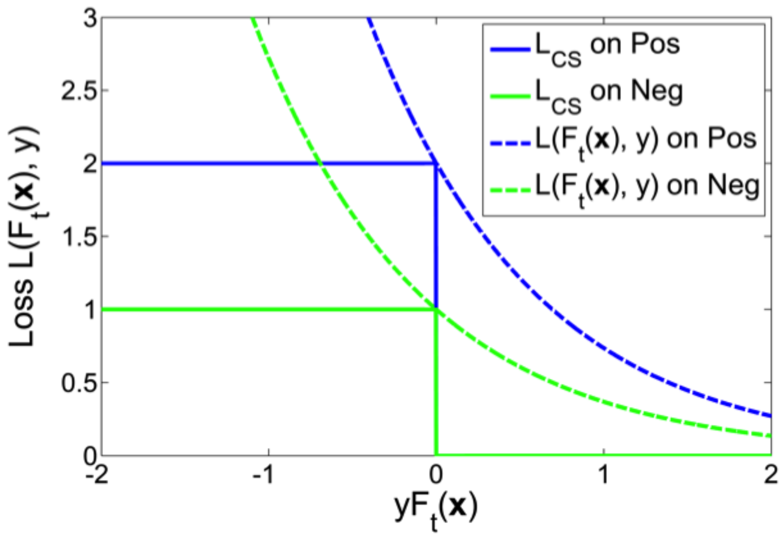
Margin theory – with costs...



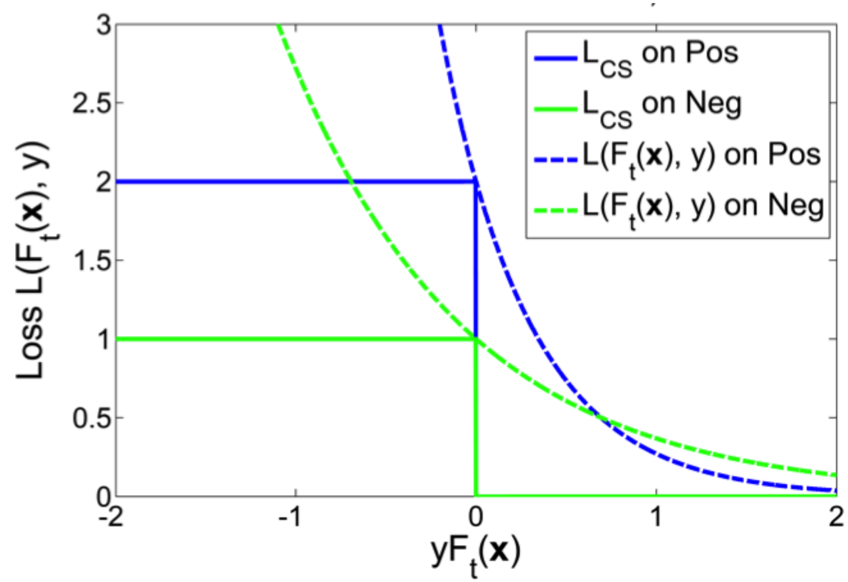
Different surrogate losses for each class.

So for a cost sensitive boosting algorithm…

We expect this to be the case.



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

'AdaBoost does not produce good probability estimates.'

Niculescu-Mizil & Caruana, 2005

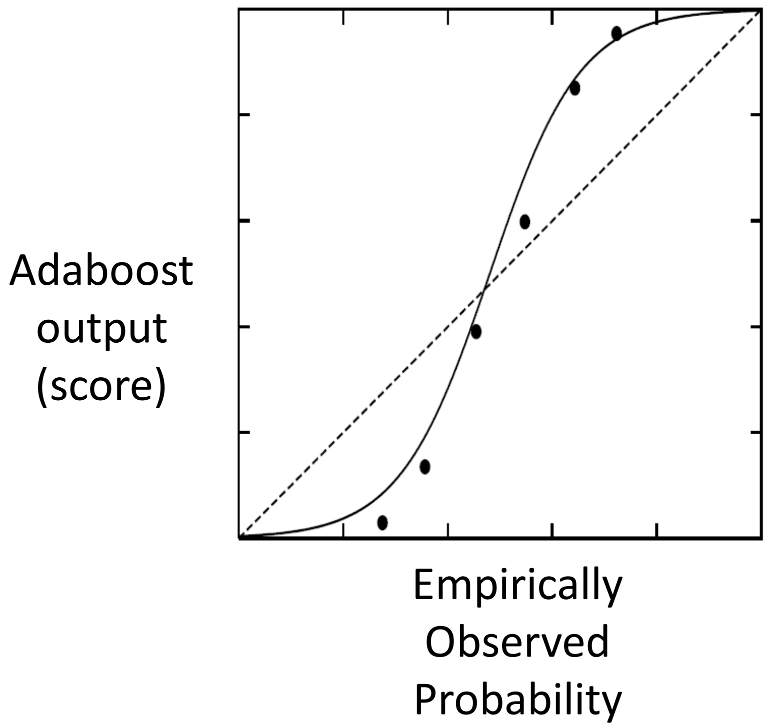
'AdaBoost is successful at [..] classification [..] but not class probabilities.'

Mease et al., 2007

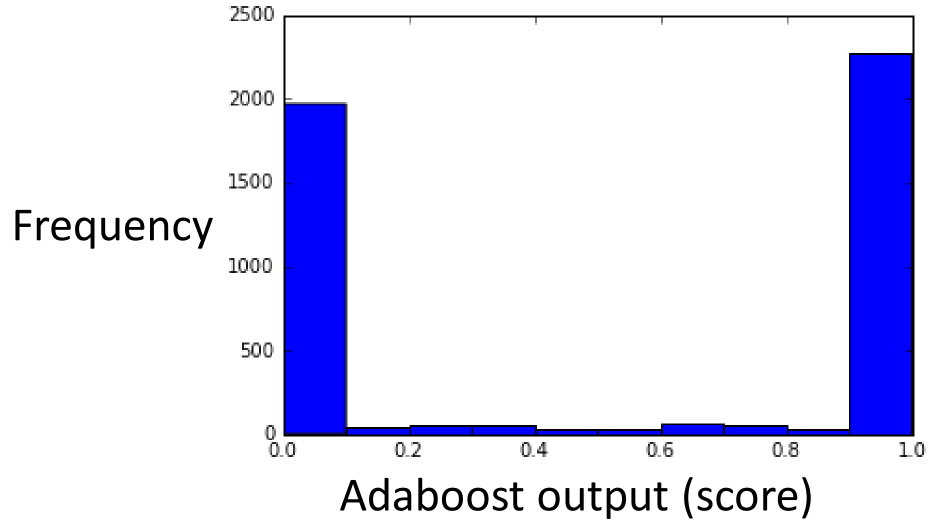
'This increasing tendency of [the margin] impacts the probability estimates by causing them to quickly diverge to 0 and 1.'

Mease & Wyner, 2008

Probabilistic models



Adaboost tends to produce probability estimates **close to 0 or 1**.



Why this distortion?

Estimates of form:

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

(Niculescu-Mizil & Caruana, 2005)

As **margin** is **maximized** on training set, scores will tend to 0 or 1.

Estimates of form:

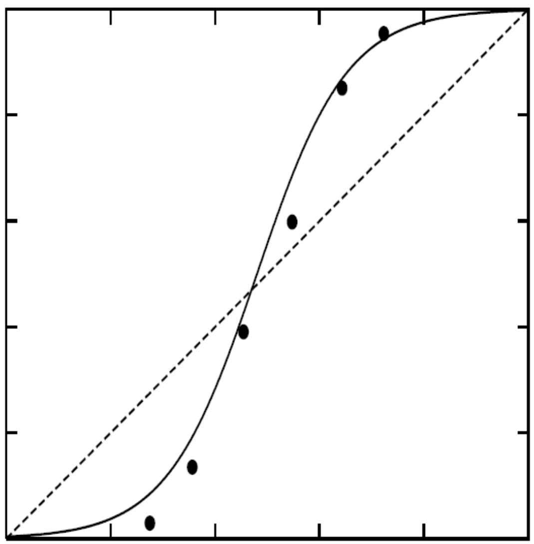
$$\hat{p}(y=1|\mathbf{x}) = \frac{1}{1+e^{-2F_t(\mathbf{x})}}$$

(Friedman, Hastie & Tibshirani, 2000)

Product of Experts; if one term close to 0 or 1, it dominates.

Probabilistic Models

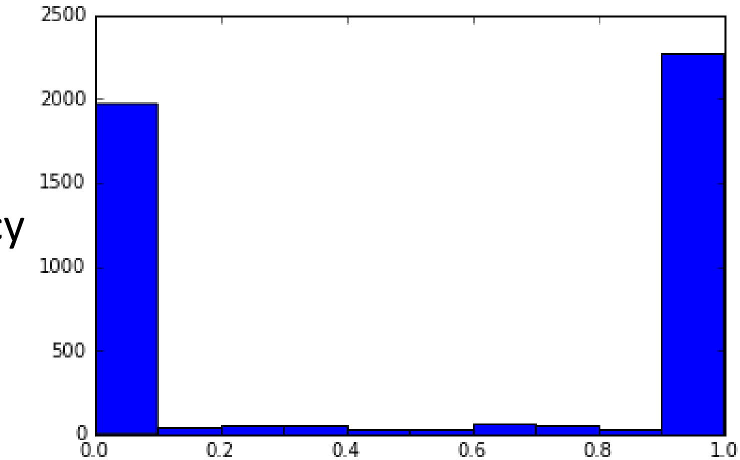
Adaboost output (score)



Empirically Observed Probability

Adaboost tends to produce probability estimates close to 0 or 1.

Frequency



Adaboost output (score)

Property: Calibrated estimates

Does the algorithm generate "calibrated" probability estimates?

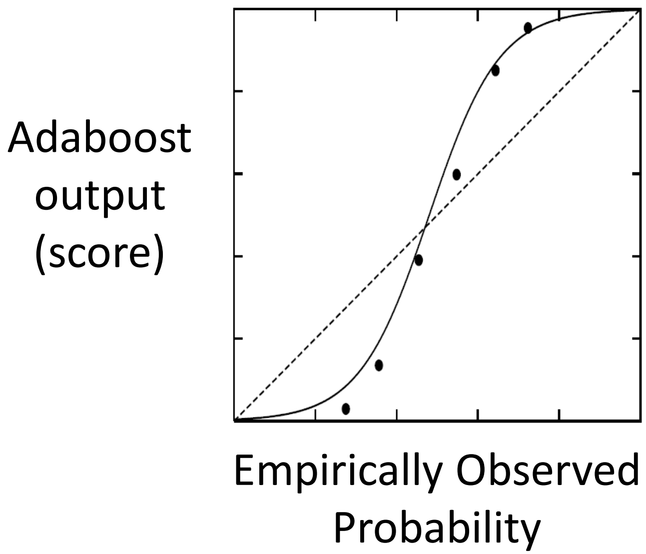
The results are in…

Method	FGD-consistent	Cost-consistent	Asymmetry-preserving	Calibrated estimates
AdaBoost (Freund & Schapire 1997)	✓		✓	**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”.

Platt scaling (logistic calibration)

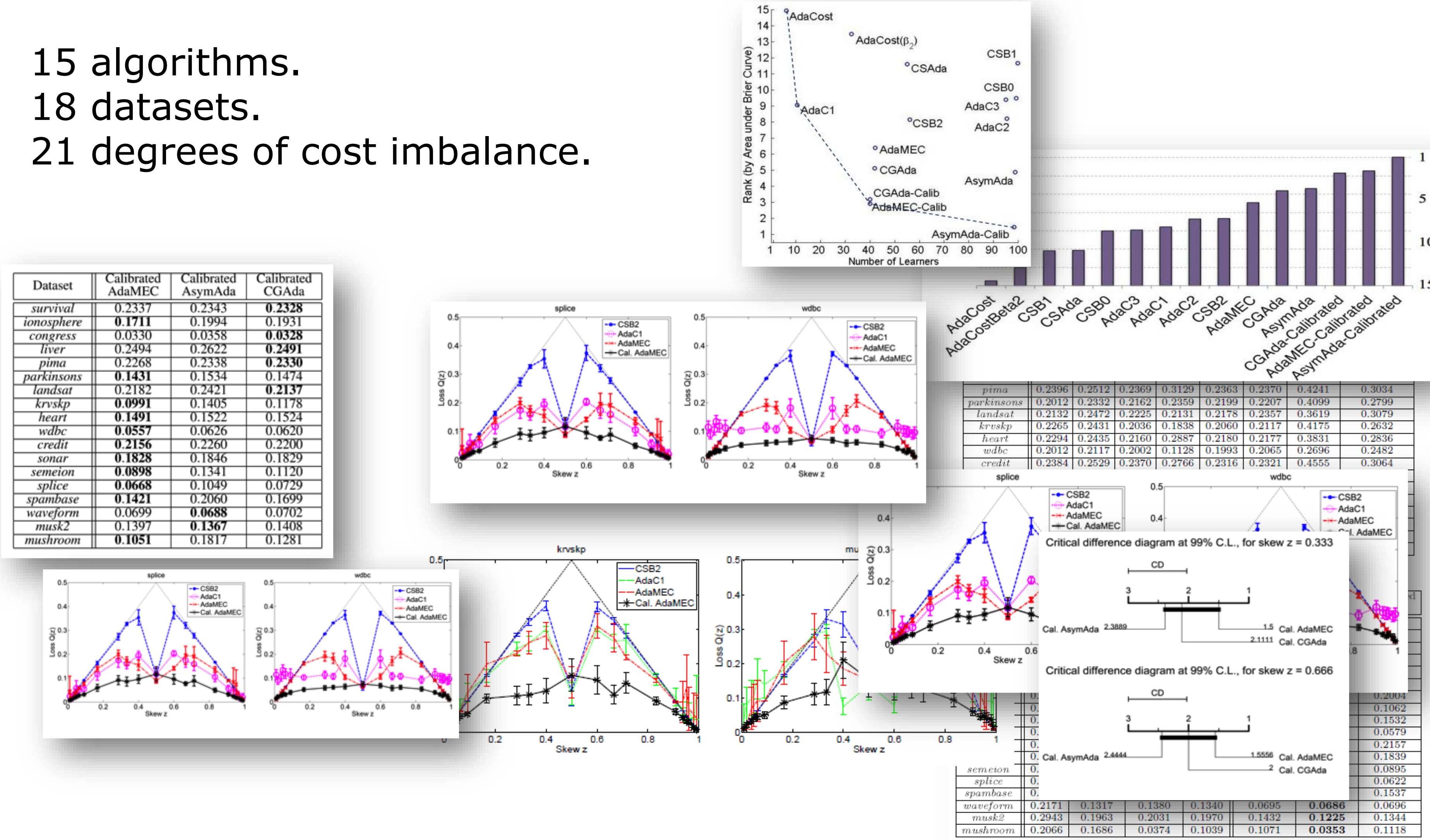
Training: Reserve part of training data (here 50% -more on this later) to fit a sigmoid to correct the distortion:



Prediction: Apply sigmoid transformation to score (output of ensemble) to get probability estimate

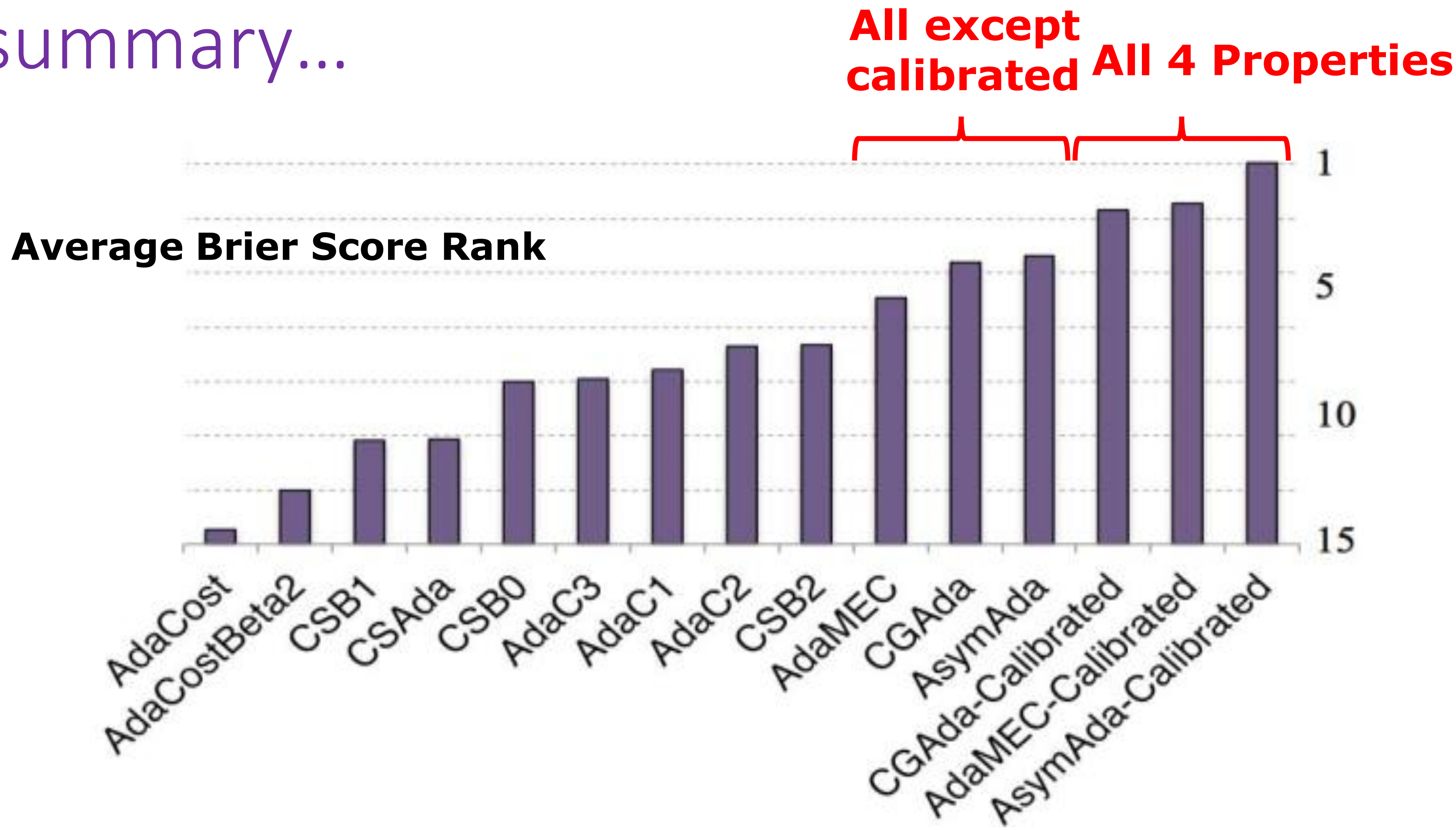
Experiments

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...



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

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

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.... $\frac{c_{FP}}{c_{FP} + c_{FN}}$

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.

Methods & properties

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

Q: What if we calibrate all methods?

A: In **theory**, …

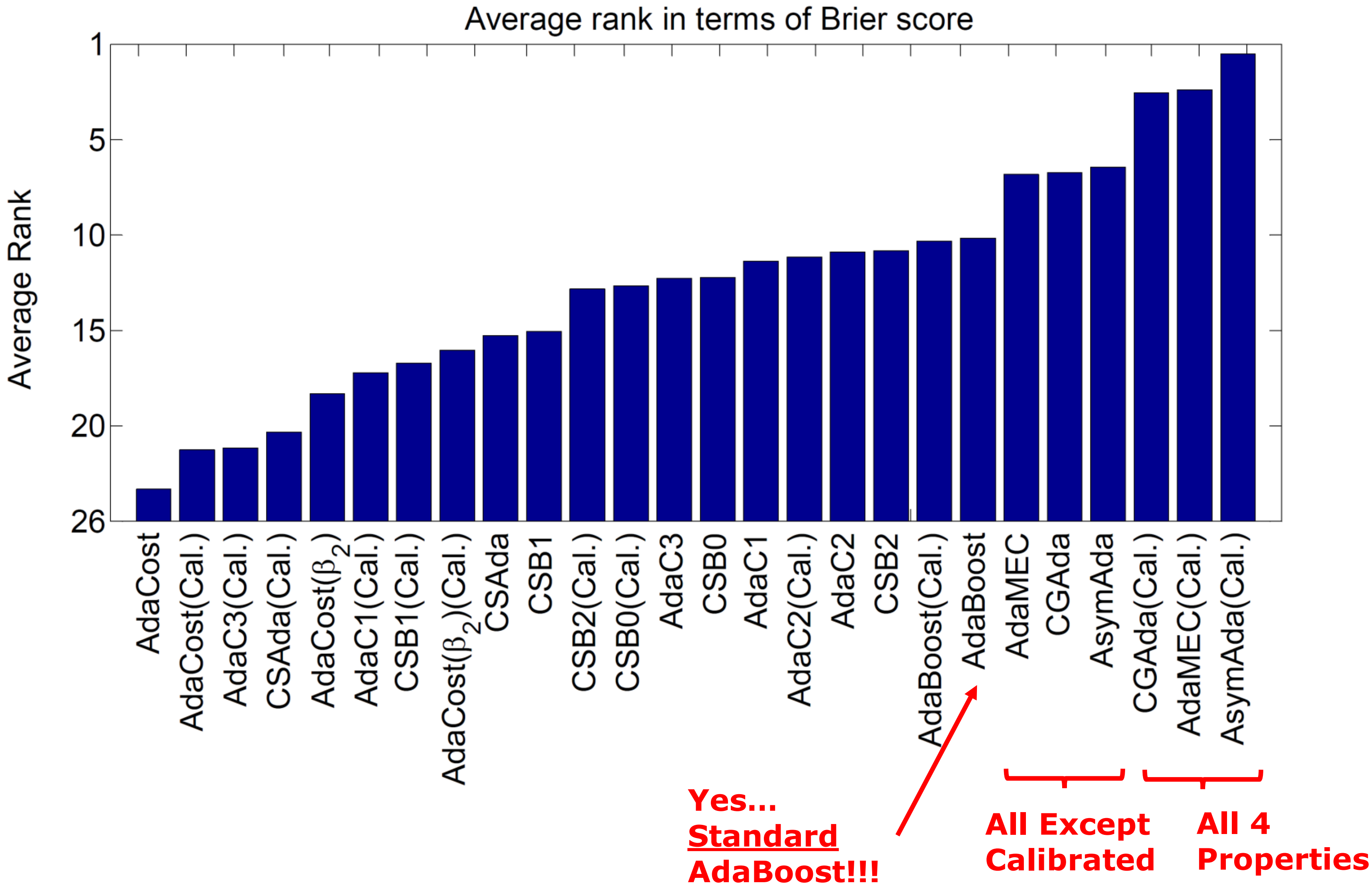
… calibration improves probability estimates.

… if a method is not cost-sensitive, will not make it.

… if the steps are not consistent, will not make them.

… if class importance is swapped during training, will not correct.

Results



Average rank in terms of Brier score
Average Rank
1
5
10
15
20
26
AdaCost
AdaCost(Cal.)
AdaC3(Cal.)
CSAda(Cal.)
AdaCost(β_2)
AdaC1(Cal.)
CSB1(Cal.)
AdaCost(β_2)(Cal.)
CSAda
CSB1
CSB2(Cal.)
CSB0(Cal.)
AdaC3
CSB0
AdaC1
AdaC2(Cal.)
AdaC2
CSB2
AdaBoost(Cal.)
AdaBoost
AdaMEC
CGAda
AsymAda
CGAda(Cal.)
AdaMEC(Cal.)
AsymAda(Cal.)
Yes…
Standard
AdaBoost!!!
All Except
Calibrated
All 4
Properties

Methods & properties

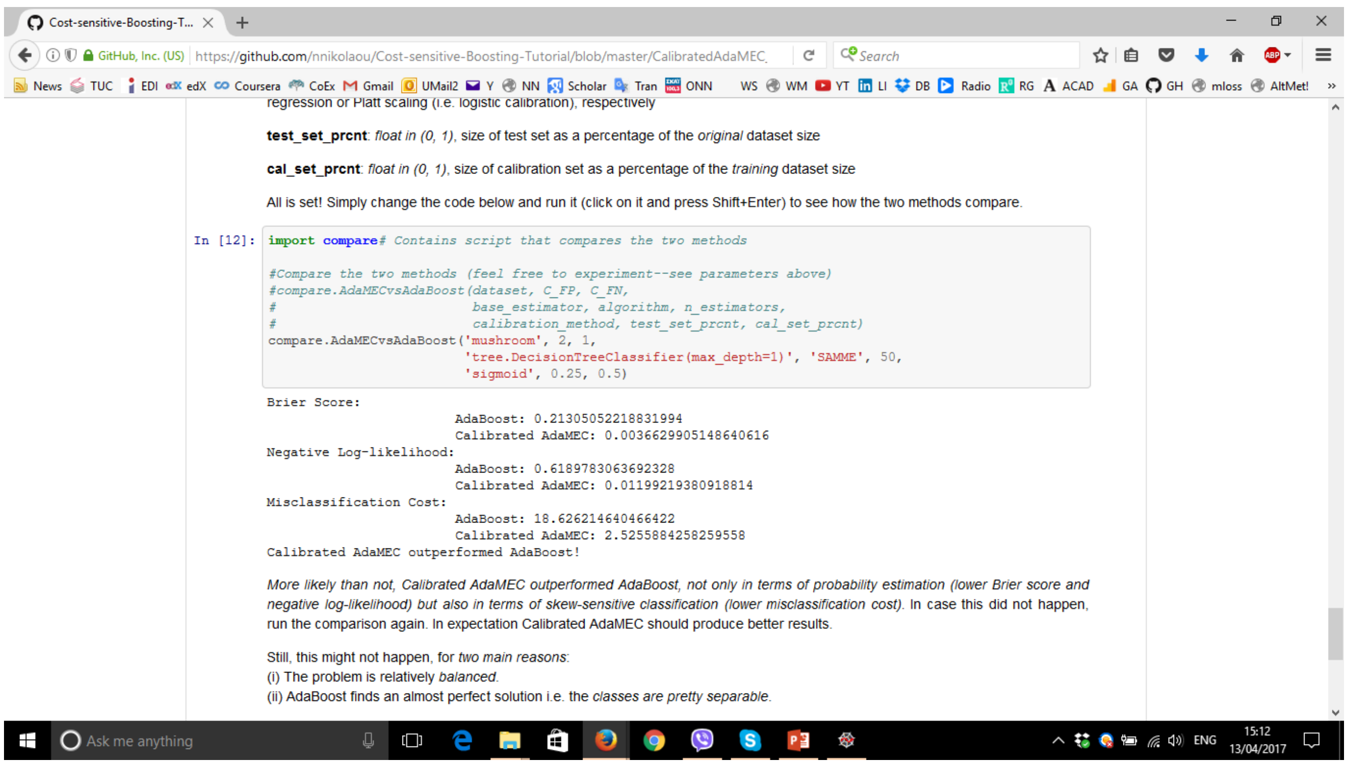
Method	FGD-consistent	Cost-consistent	Asymmetry-preserving	Calibrated estimates
AdaBoost (Freund & Schapire 1997)	✓		✓	**All algorithms** produce **uncalibrated** probability estimates!
AdaCost (Fan et al. 1999)				
AdaCost(β_2) (Ting 2000)				
CSB0 (Ting 1998)			✓	
CSB1 (Ting 2000)			✓	
CSB2 (Ting 2000)			✓	
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”.

Q: Sensitive to calibration choices?

A: Check it out on your own!

https://github.com/nnikolaou/Cost-sensitive-Boosting-Tutorial



Results

Isotonic regression > Platt scaling, for larger datasets

Can do better than 50%-50% train-calibration split by using fewer data to calibrate and more to train (problem dependent; see Part II)

(Calibrated) Real AdaBoost > (Calibrated) Discrete AdaBoost...

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…. $\frac{c_{FP}}{c_{FP} + c_{FN}}$

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

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

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

Relevant publications

- Nikolaos Nikolaou and Gavin Brown, ***Calibrating AdaBoost for Asymmetric Learning***, Multiple Classifier Systems, 2015

- Nikolaos Nikolaou, Narayanan Edakunni, Meelis Kull, Peter Flach and Gavin Brown, ***Cost-sensitive Boosting algorithms: Do we really need them?***, Machine Learning Journal, Vol. 104, Issue 2, Sept 2016
  - Best Poster Award, INIT/AERFAI summer school in ML 2014
  - Plenary Talk ECML 2016
  - Best Paper Award 2016, School of Computer Science, University of Manchester

- Nikolaos Nikolaou, ***Cost-sensitive Boosting: A Unified Approach***, PhD Thesis, University of Manchester, 2016
  - Best Thesis Award 2017, School of Computer Science, University of Manchester

Resources & code

- Easy-to-use but not so flexible ‘Calibrated AdaMEC’ python implementation (scikit-learn style):

  https://mloss.org/revision/view/2069/

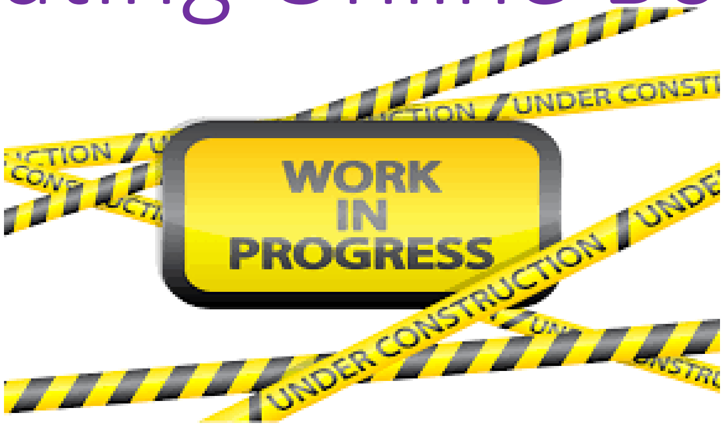
- i-python tutorial for all this with interactive code for ‘Calibrated AdaMEC’, where every choice can be tweaked:

  https://github.com/nnikolaou/Cost-sensitive-Boosting-Tutorial

End of Part I

Ερωτήσεις; - Questions?

Part II:
Calibrating Online Boosting



Online learning

Examples presented **one (or a few) @ a time**

Learner makes **predictions as examples are received**

Each '**minibatch**' used to **update model**, then discarded;
constant time & space complexity

Why?

- Data arrive this way (**streaming**)
- Problem (e.g. data distribution) **changes over time**
- To **speed up learning** in big data applications

Online learning

For each $minibatch\ n$ do:

1. **Receive** n
2. **Predict label / class probability** of examples in n
3. **Get true label** of examples in n
4. **Evaluate** learner's performance on n
5. **Update** learner **parameters** accordingly

Online Boosting (Oza, 2004)

Train weak learners **sequentially** on each datapoint x:

If weak learner misclassifies x*,*

Increase weight of x for the purposes *of updating parameters of next weak learner*

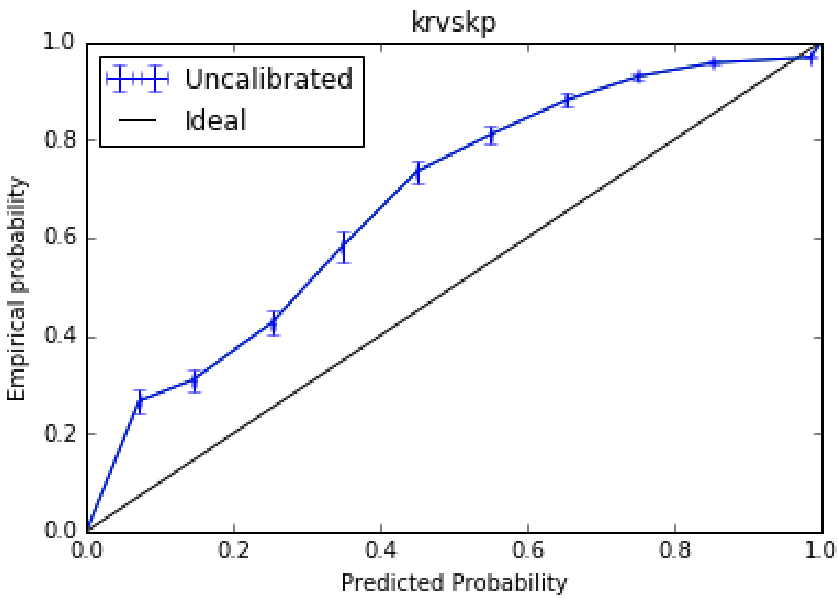
Else,

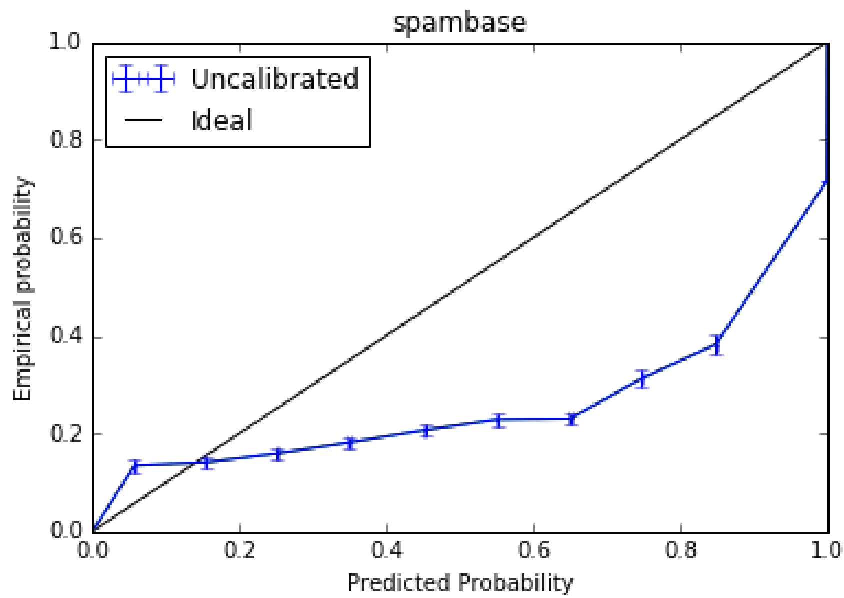
Decrease it…

Is it good at estimating probabilities?

Online Boosting probability estimates

Probability estimates -as in AdaBoost- are uncalibrated:





How to calibrate online Boosting?

Batch Learning: **reserve part of the dataset** to train calibrator function (logistic sigmoid, if Platt scaling)

Online learning: **cannot do this**; on each minibatch we must **decide** whether to **train ensemble or calibrator**

How to make this decision?

Naïve approach

- Calibrate **every N rounds**:

For each $minibatch$ n do:

1. **Receive** n
2. **Predict class probability** of examples in n
3. **Get true label** of examples in n
4. **Evaluate** learner's performance on n (e.g. **likelihood**)
5. **Every N-th round:**
   - **5.1 Update calibrator parameters** accordingly

   Every other round:
   - **5.2 Update ensemble parameters** accordingly

Complications

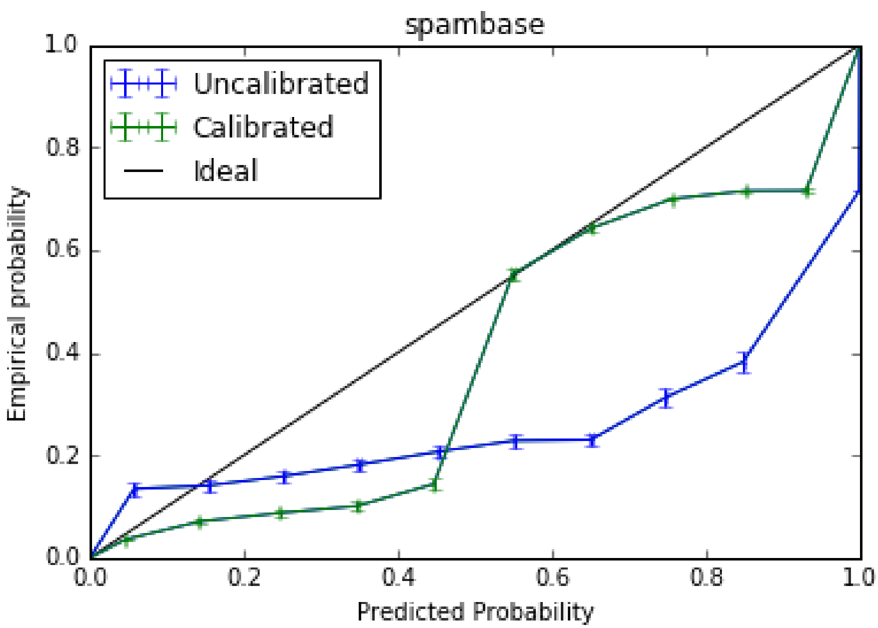
How to pick N?

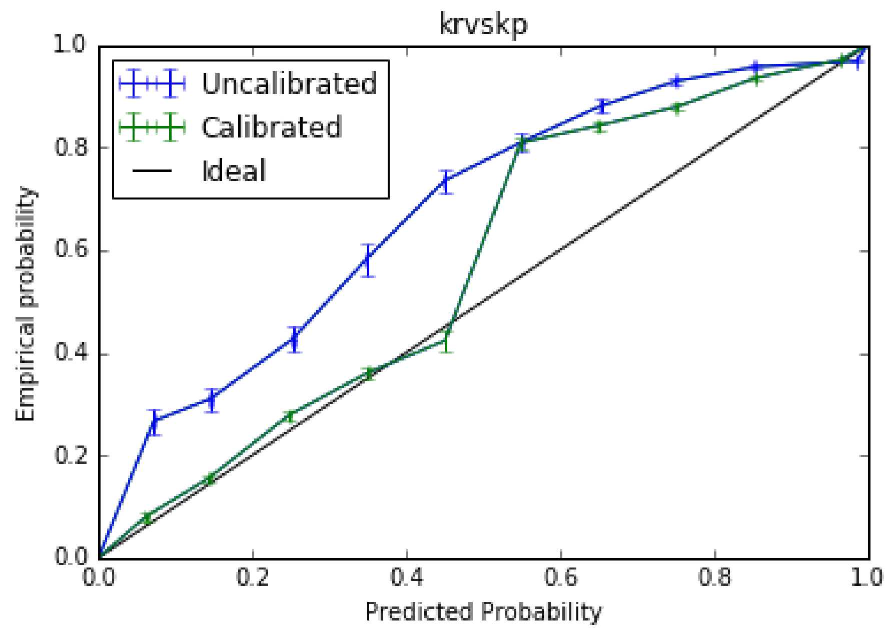
- Will depend on **problem**
- Will depend on **ensemble hyperparameters**
- Will depend on **calibrator hyperparameters**
- **Might change** during training…

In batch learning can **choose via cross-validation**; **not here**

Still, naïve better than nothing

Results with N = 2 (**not** best value):





A more refined approach

- What if we could **learn** a good sequence of alternating between actions?

Bandit Algorithms

Bandit optimization

A **set of actions (arms)** -on each round we choose one

Each action associated with a **reward distribution**

Each time an action taken we **sample** its reward distribution

Sequence of actions that **minimize cumulative regret?**

Exploration vs. Exploitation

In **online calibrated boosting**:

Two actions: **{ train , calibrate }**

Reward: **Increase in overall model likelihood** after action

Thomson sampling

A **Bayesian** take on bandits for learning reward distribution

Assume **rewards are Gaussian**; start with **Gaussian prior**, then update using **self-conjugacy of Gaussian distribution**

Take action with **highest expected reward**

UCB policies

‘Optimism in the face of uncertainty’

Choose not the action with best mean reward, but that with **highest upper bound on reward**

Bounds derived for arbitrary (UCB1, UCB1-Improved) or specific (KL-UCB) reward distributions

Discounted rewards

‘Forgeting the past’

Weigh past rewards less; protects from **non-stationarity**

Why non-stationary?

- **Data distribution** might **change**…
- …most importantly: **reward distributions** will **change**:
  if we perform one action many times, the relative reward for performing the other is expected to have increased

Some initial results

- Uncalibrated

  vs. Naively-Calibrated $N \in \{2, 4, 6, 8, 10, 12, 14\}$

  vs. UCB1, UCB1-Improved, Gaussian Thompson Sampling

  vs. Discounted versions of above

- Initial results:
  - calibrating (even naive) > not calibrating
  - discounted versions ≥ as 'Every N' policy (+ no need to set N)
  - Not discounted → one action (as expected)

In summary…

Online Boosting **poor probability estimates**; some **calibration** can improve

Learn a good sequence of calibration / training actions using **bandits**

Online, **fast**, **at least as good as 'best naïve'**

Easy to **adapt to other problems** (e.g. cost-sensitive learning)

Robust to ensemble/calibrator **hyperparameters**

Extensions: e.g. **adversarial**, **contextual**, **refine calibration**, …

Ευχαριστώ! - Thank you!

Ερωτήσεις; - Questions?