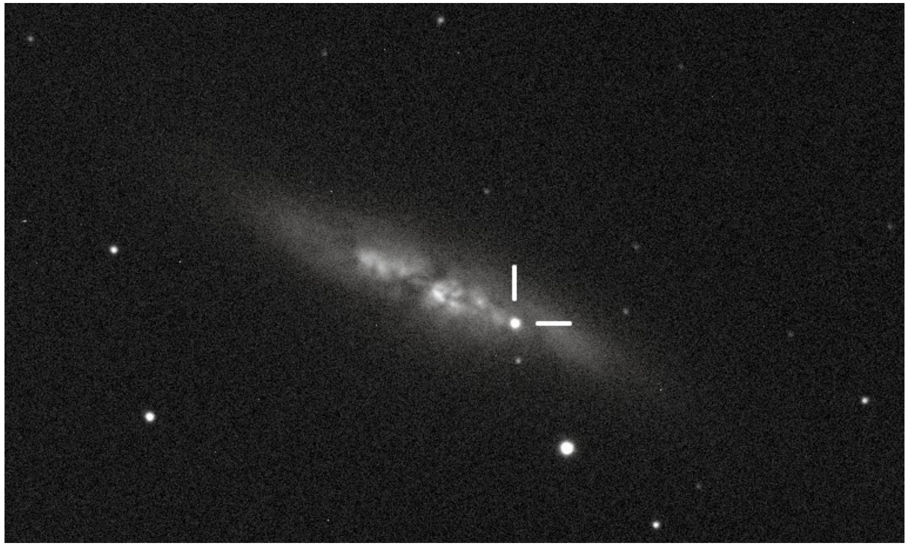
Cost-sensitive learning with AdaBoost

Nikos Nikolaou

The University of Manchester

Asymmetric Learning

Cost-sensitive
different errors have
have different costs



Imbalanced classes
different classes appear
with different frequency

...or both!

Motivation

I have symptoms of a serious disease…

…so I go to the doctor for a test

Binary decision :

H**ave** disease (**Positive,** $\boldsymbol{y = 1}$)

D**on't have** disease (**Negative,** $\boldsymbol{y = -1}$)

But tests (& doctors) make **mistakes**…

Possible Outcomes

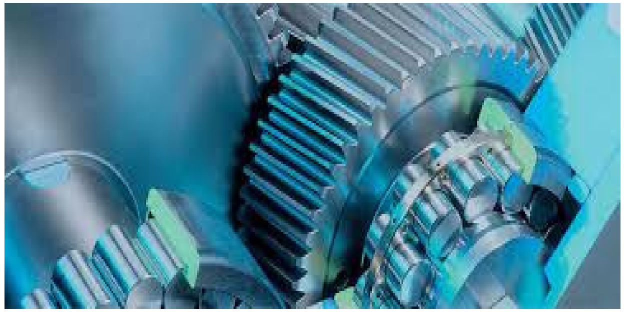
		Predicted Class: Positive	Predicted Class: Negative
True Class	**Positive**	TP	FN
	Negative	FP	TN

Two types of **misdiagnosis**:

FP: don’t have disease but test says I do (**BAD**)

FN: have disease but test says I don’t (**VERY BAD!**)

Other Applications



The Cost Matrix

Assign a **cost** to each **type of outcome**

Assumes **cost depends only on class**

		Predicted Class	
		Positive	**Negative**
True Class	**Positive**	C_{TP}	C_{FN}
	Negative	C_{FP}	C_{TN}

must satisfy: $C_{TP} < C_{FN}$ & $C_{TN} < C_{FP}$

The Cost Matrix

Most common case:

Predicted Class

True Class		Positive	Negative
	Positive	0	C_{FN}
	Negative	C_{FP}	0

must satisfy: $0 < C_{FN}$ & $0 < C_{FP}$

Solving Cost-Sensitive Learning

1. **Change classifier:** let it **take into account** the **cost matrix**

2. **Resample data:** create **class imbalance matching cost imbalance**

3. Get **class probability estimates** from classifier & assign to class that incurs the **minimum expected cost**

Boosting/AdaBoost Recap

- Ensemble method: sequentially combine multiple weak learners to build a strong one

- Weights over examples: on each round increase for previously misclassified examples, decrease for correctly classified ones

- Confidence coefficient on each learner, based on its error rate

- Nice theoretical properties, resistant to overfitting, extensively studied, successful applications

AdaBoost

Can it handle cost-sensitive problems?

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

$$\epsilon_t = \sum_{i:h_t(\mathbf{x}_i)\neq y_i} D_i^t$$

Assign a confidence score to each weak learner

$$D_i^{t+1} = e^{-y_i h_t(\mathbf{x}_i)\alpha_t} D_i^t$$

Update examples' weights

$$D_i^1 = \frac{1}{N}$$

Start with a uniform weight distribution over the examples

$$H(\mathbf{x}') = sign\left[\sum_{t=1}^{M} \alpha_t h_t(\mathbf{x}')\right]$$

Confidence weighted majority vote

Asymmetric Boosting Variants

$$\alpha_t = \frac{1}{2} \log \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

$$\epsilon_t = \sum_{i:h_t(\mathbf{x}_i) \neq y_i} D_i^t$$

Assign a confidence score to each weak learner

(Fan et al., 1999)
(Cohen & Singer, 1999)
(Ting, 2000)
(Joshi et al., 2001)
(Sun et al., 2005; 2007)
(Masnadi-Shirazi & Vasconcelos , 2007; 2011)

$$D_i^{t+1} = e^{-y_i h_t(\mathbf{x}_i) \alpha_t} D_i^t$$

Update examples' weights

(Ting & Zheng, 1998)
(Ting, 2000)
(Viola & Jones, 2001; 2002)

$$D_i^1 = \frac{1}{N}$$

Start with a uniform weight distribution over the examples

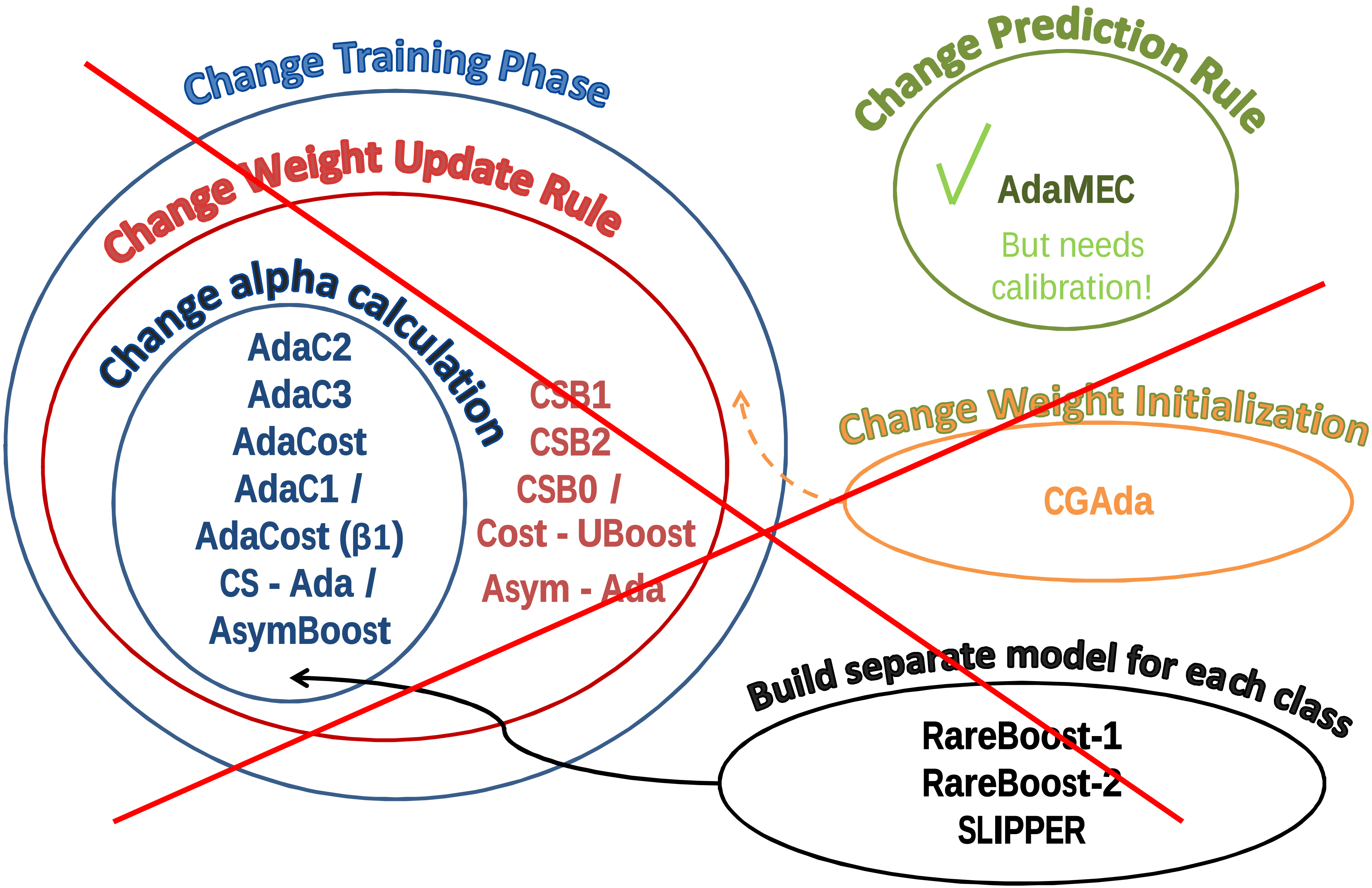
(Landesa-Vázquez & Alba-Castro, 2013;2015a;2015b)

$$H(\mathbf{x}') = sign \left[\sum_{t=1}^{M} \alpha_t h_t(\mathbf{x}') \right]$$

Confidence weighted majority vote

(Ting, 2000)

Asymmetric Boosting Variants



Issues with modifying training phase

- No **theoretical guarantees** of original AdaBoost
  - e.g. bounds on generalization error, convergence, confidence $\alpha_t \in R^+$, max num. weak learners M not fixed
- Most heuristic, **no decision-theoretic** motivation
  - ad-hoc changes, not apparent what they achieve
- Need to **retrain** if skew ratio changes
- Require **extra hyperparameters** to be set via CV

Boosting as a Product of Experts

AdaBoost: $$\hat{p}(y=1|\mathbf{x};F_M) = \frac{\prod_{t=1}^{M}\hat{p}(y=1|\mathbf{x};f_t)}{\prod_{t=1}^{M}\hat{p}(y=1|\mathbf{x};f_t)+\prod_{t=1}^{M}\hat{p}(y=-1|\mathbf{x};f_t)}$$

(Edakunni et al., 2011)

AdaMEC: $$\hat{p}(y=1|\mathbf{x};F_M) = \frac{c_{FN}\prod_{t=1}^{M}\hat{p}(y=1|\mathbf{x};f_t)}{c_{FN}\prod_{t=1}^{M}\hat{p}(y=1|\mathbf{x};f_t)+c_{FP}\prod_{t=1}^{M}\hat{p}(y=-1|\mathbf{x};f_t)}$$

AdaC2: $$\hat{p}(y=1|\mathbf{x};F_M) = \frac{{c_{FN}}^M\prod_{t=1}^{M}\hat{p}(y=1|\mathbf{x};f_t)}{{c_{FN}}^M\prod_{t=1}^{M}\hat{p}(y|\mathbf{x};f_t)+{c_{FP}}^M\prod_{t=1}^{M}\hat{p}(y=-1|\mathbf{x};f_t)}$$

⋮

Issues with modifying prediction rule

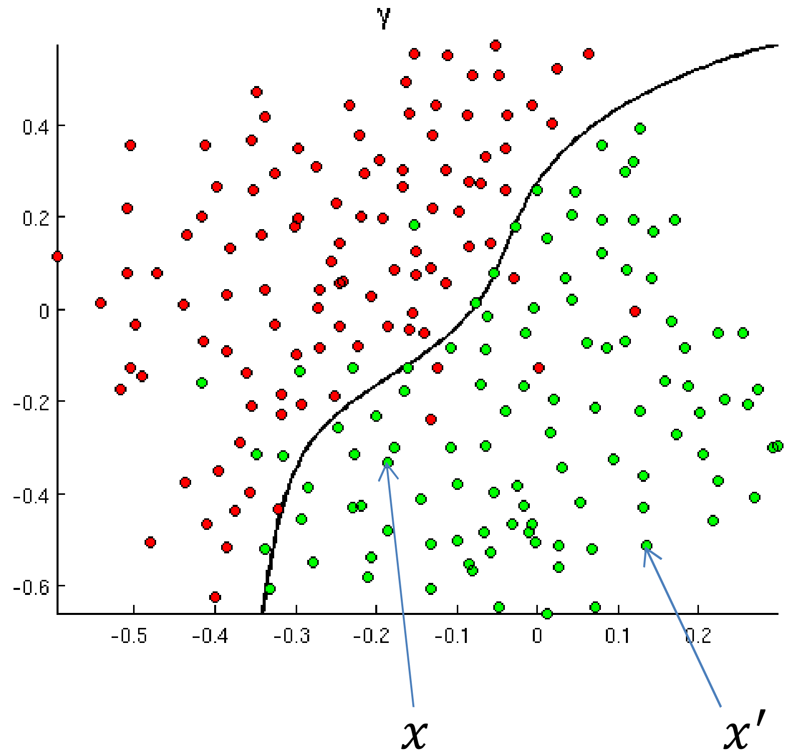
- *AdaMEC* changes prediction rule from **weighted majority** vote to **minimum expected cost** criterion

- Problem: **incorrectly assumes scores** are **probability estimates**...

- ...but can correct this via **calibration**

Things classifiers do...

- **Classify** examples
  - Is x positive?
- **Rank** examples
  - Is x 'more positive' than x'?
- Output a **score** for each example
  - 'How positive' is x?
- Output a **probability estimate** for each example
  - What is the (estimated) probability that x is positive?



Why estimate probabilities?

- Need **probabilities** when a **cost-sensitive decision** needs to be made; scores won't cut it

- Will assign to class that minimizes **expected** cost

  i.e. assign to $y = 1$ (Pos) only if:

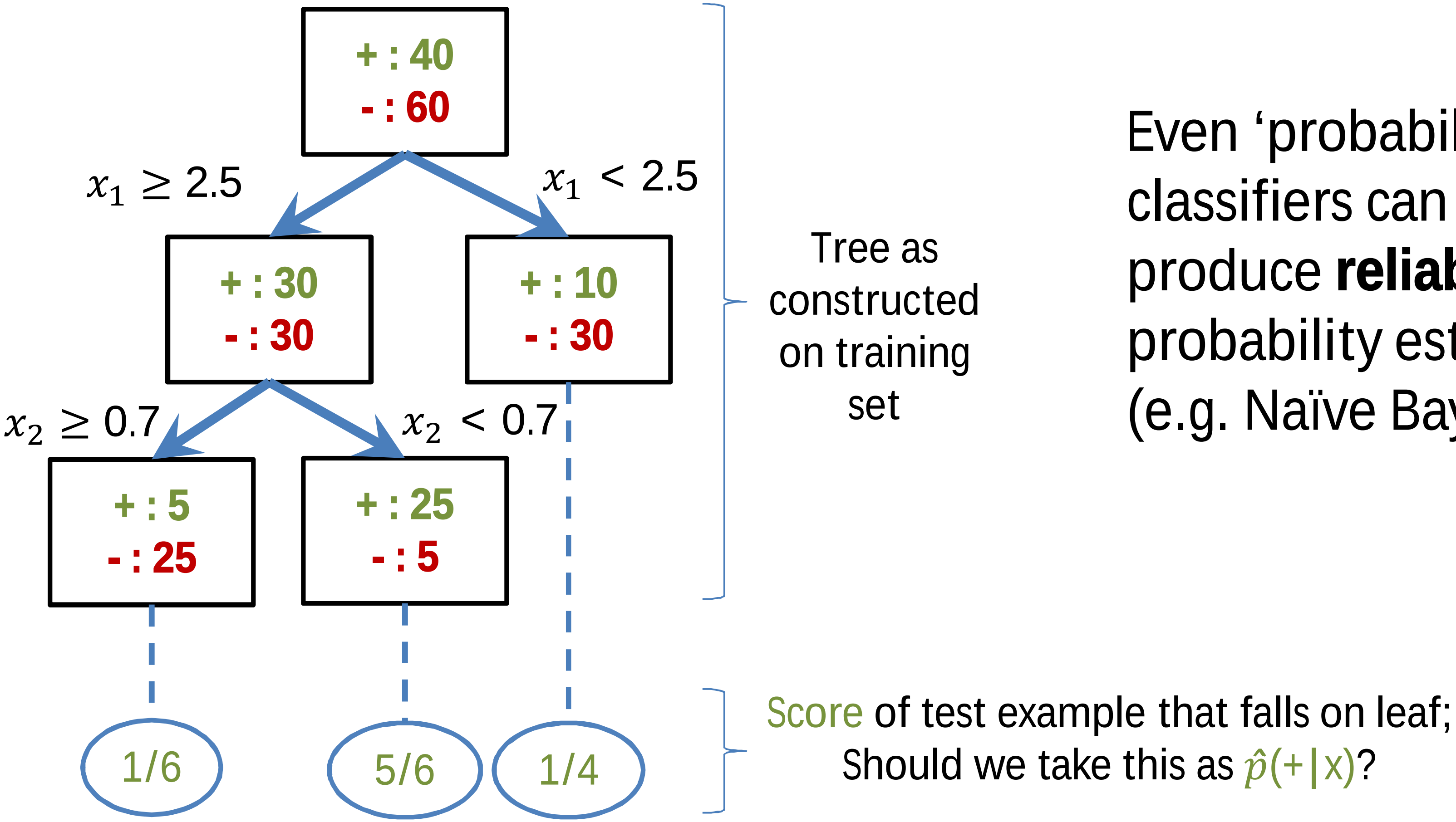
expected cost of assigning to Neg < expected cost of assigning to Pos

$$\Leftrightarrow$$

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

Probability estimation is not easy

Most classifiers don't produce probability estimates **directly** but we get them via scores, e.g. decision trees:



Even 'probabilistic' classifiers can fail to produce **reliable** probability estimates (e.g. Naïve Bayes)

Calibration

- $s(x) \in [0,1]$: score assigned by classifier to example x

- A classifier is **calibrated** if

  $\hat{p}(y = 1|x) \to s(x)$, as $N \to \infty$

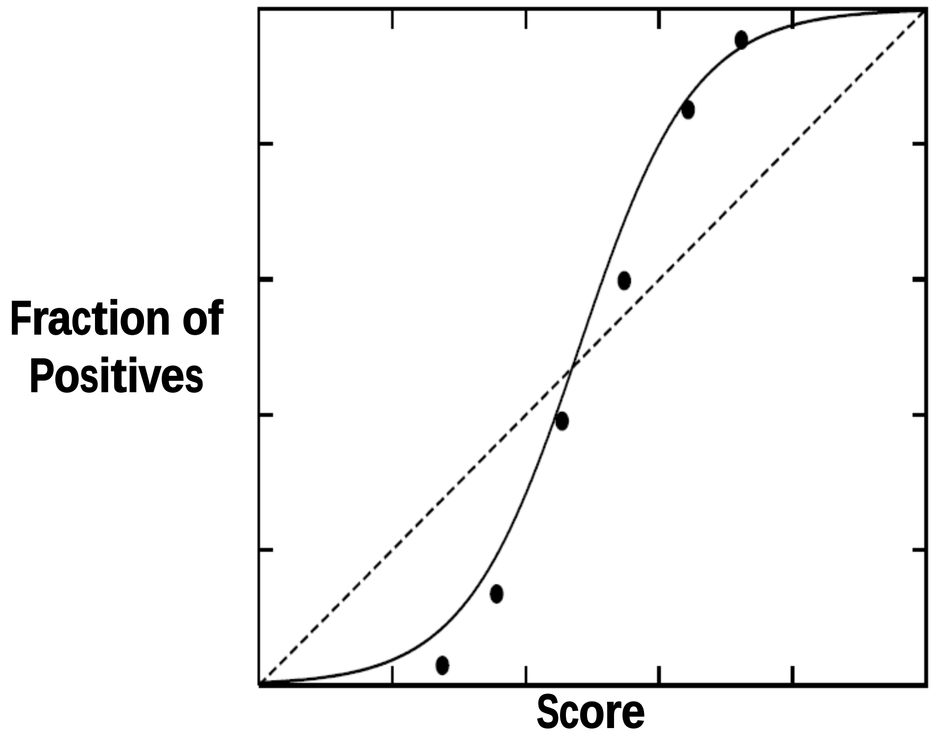
- Intuitively: consider all examples with $s(x) = 0.7$;

  70% of these examples **should** be positives

- Calibration **can only improve** classification (asymptotically)

Probability estimates of AdaBoost

Score for Boosting: $s(\mathbf{x}') = \frac{\sum_{t=1}^{M} \alpha_t \frac{h_t(\mathbf{x}')+1}{2}}{\sum_{t=1}^{M} \alpha_t} \in [0, 1]$

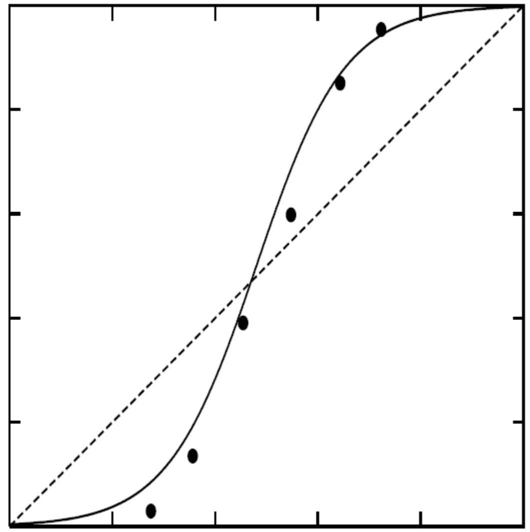


Boosted trees / stumps: **sigmoid distortion**; scores pushed more towards 0 or 1 as num. of boosting rounds increases

(Niculescu-Mizil & Caruana, 2006)

Calibrating AdaBoost: Platt Scaling

- Find A, B for $\hat{p}(y = 1|\mathrm{x}) = \frac{1}{1+e^{A\, s(x) + B}}$, s. t. likelihood of data is maximized



- **Separate sets** for train & calibration
- Motivation: undo sigmoid distortion
  observed in boosted trees
- Alternative: isotonic regression

Calibrating AdaBoost for asymmetric learning

On training set:

- Train AdaBoost ensemble H_M

↓

On validation set:

- Calculate score $s(\mathbf{x}) = \frac{\sum_{t=1}^{M} \alpha_t \frac{h_t(\mathbf{x})+1}{2}}{\sum_{t=1}^{M} \alpha_t} \in [0, 1]$ of each example $\mathbf{x}$ under ensemble H_M
- Find A, B s. t. the likelihood of the data under model $\hat{p}(y = 1|\mathbf{x}) = \frac{1}{1+e^{As(\mathbf{x})+B}}$ is maximized

↓

On test set:

- Calculate score $s(\mathbf{x})$, $\forall$ example $\mathbf{x}$ under H_M
- Apply transformation $\hat{p}(y = 1|\mathbf{x}) = \frac{1}{1+e^{As(\mathbf{x})+B}}$ to the scores $s(\mathbf{x})$ to get probability estimates
- Predict class $H_M(\mathbf{x}) = sign\ [\hat{p}(y = 1|x) - \frac{c_{FP}}{c_{FP}+c_{FN}}]$

Experimental Design

- AdaC2 vs. CGAda vs. AdaMEC vs. Calibrated AdaBoost
  75% Tr / 25% Te 50% Tr / 25% Cal / 25% Te

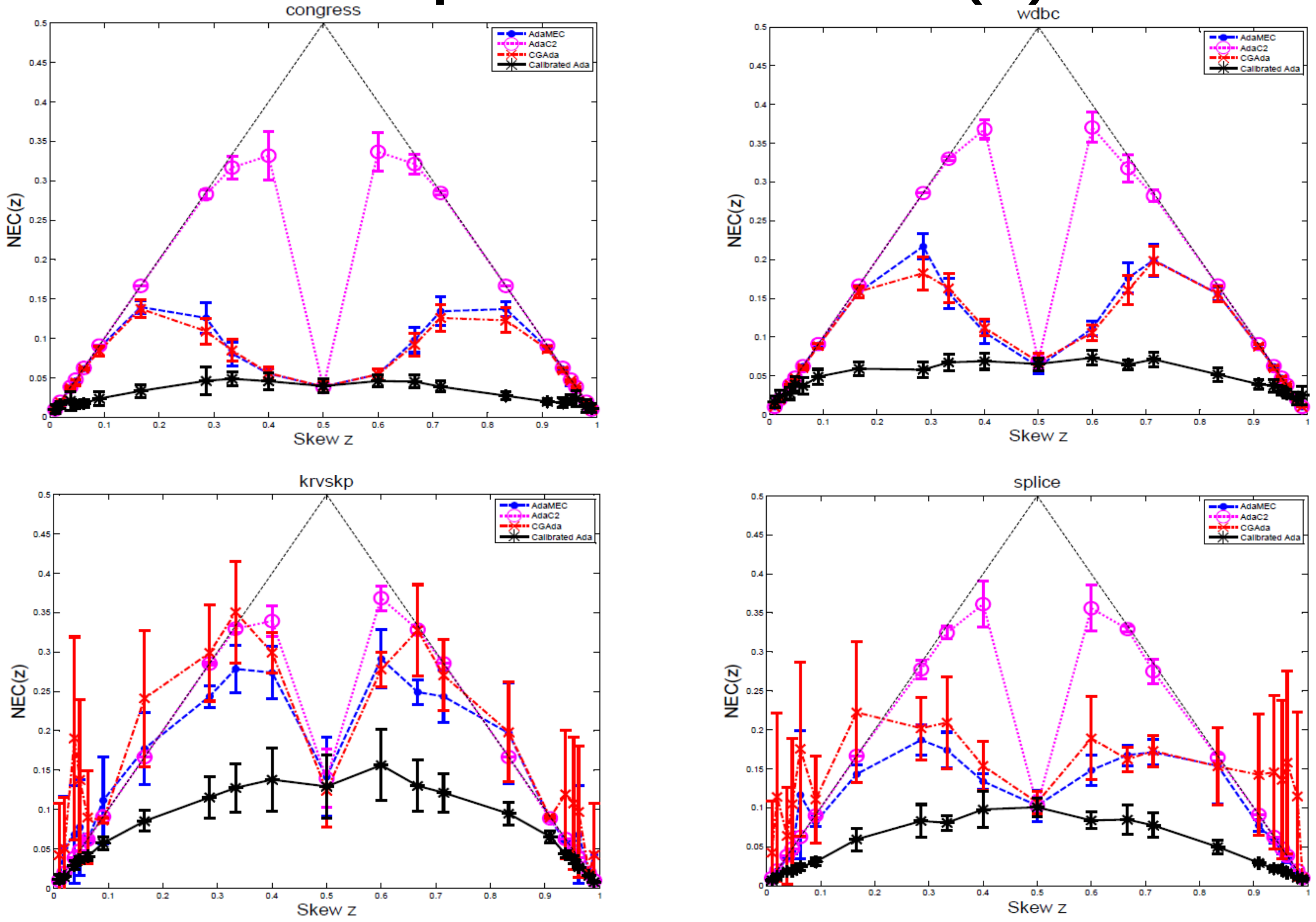
- Weak learner: univariate logistic regression

- 18 datasets

- Evaluation: normalized expected cost $\in [0, 1]$

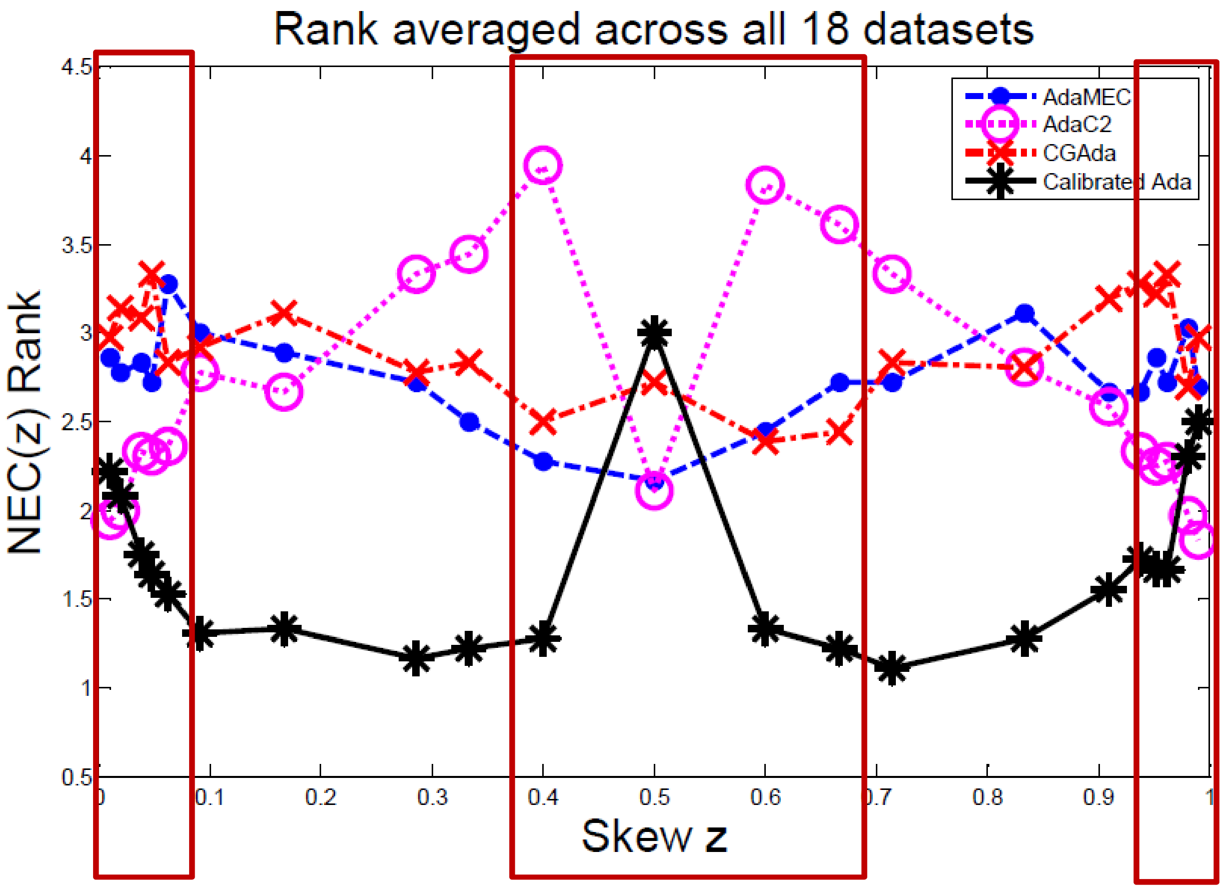
- Various skew ratios: $z = \frac{c_{FP}}{c_{FN}+c_{FP}}$

Empirical Results (1)



Ada-Calibrated at least as good as best, especially good on larger datasets

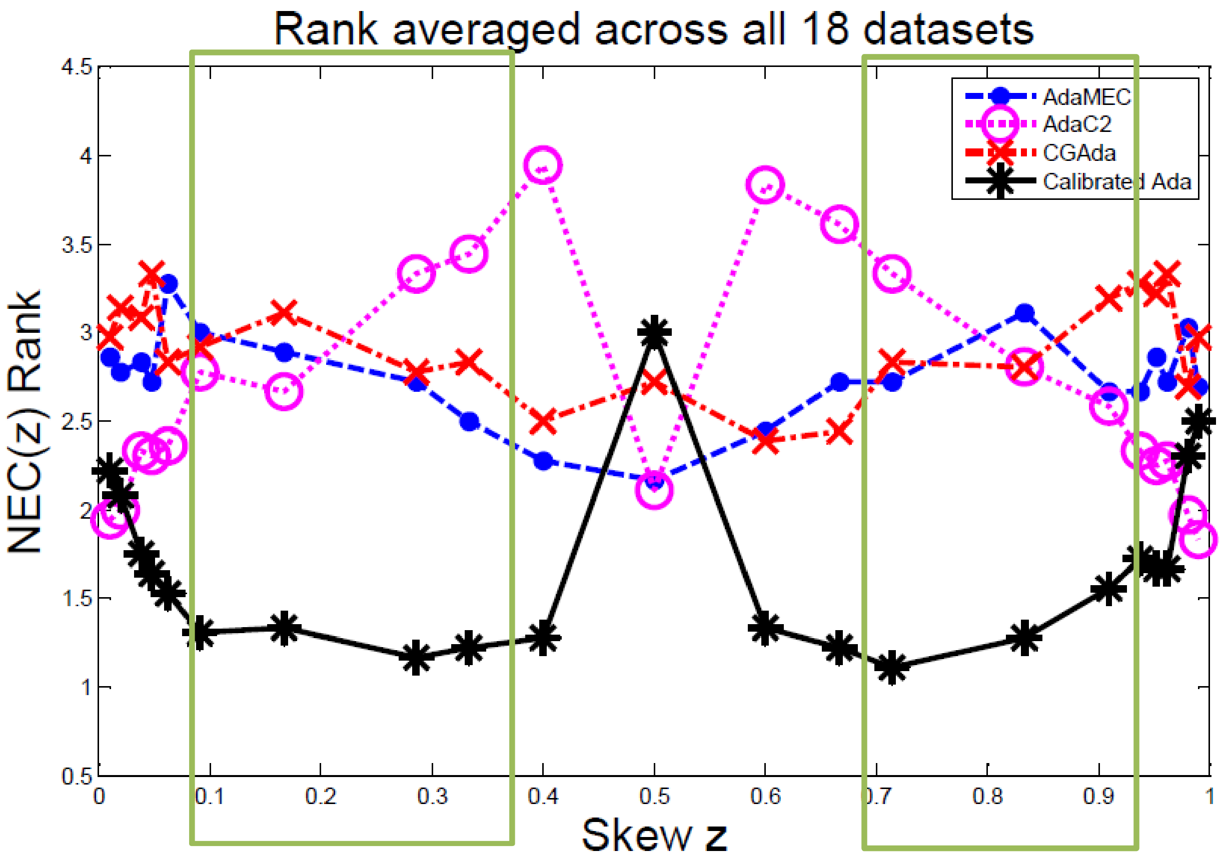
Empirical Results (2)



Nemenyi test at the 0.05 level on the differences

Ada-Calibrated at least as good as best (no sig. diff.) for very low /high skew

Empirical Results (2)



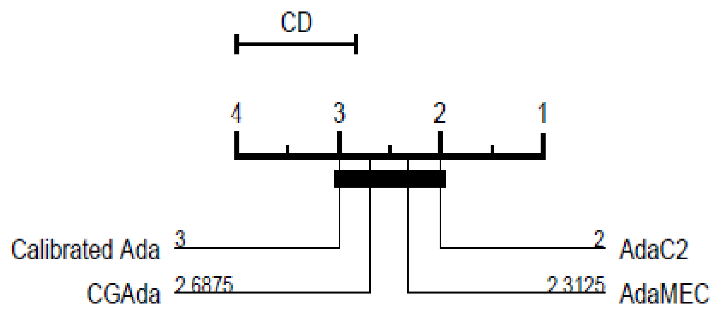
Nemenyi test at the 0.05 level on the differences

Ada-Calibrated at least as good as best (no sig. diff.) for very low \high skew

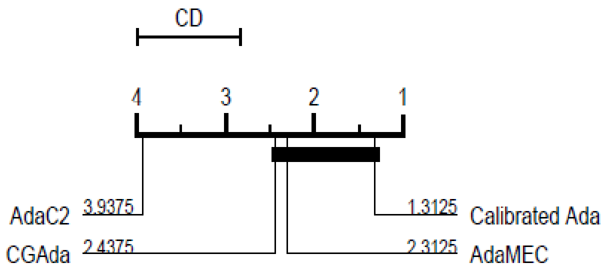
Ada-Calibrated superior to rest (sig. diff.) for medium skew

Empirical Results (3)

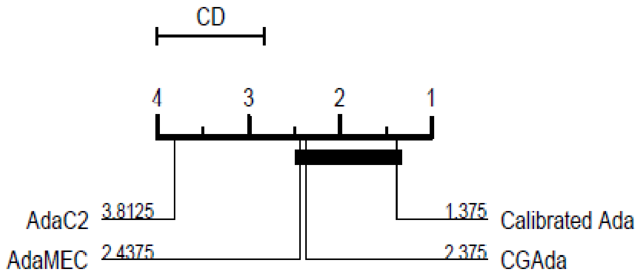
Critical difference diagram at 95% C.L., for skew z = 0.5



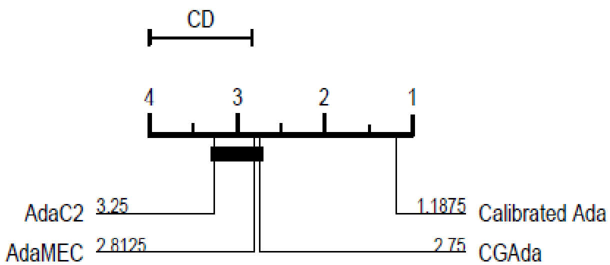
Critical difference diagram at 95% C.L., for skew z = 0.4



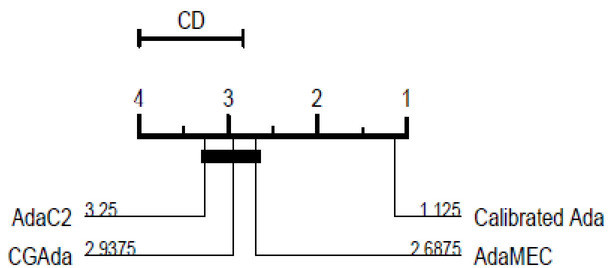
Critical difference diagram at 95% C.L., for skew z = 0.6



Critical difference diagram at 95% C.L., for skew z = 0.28571



Critical difference diagram at 95% C.L., for skew z = 0.71429



Conclusion

- Calibrating AdaBoost empirically **comparable** (small data & skew)/ **superior** (big data / skew) to alternatives published 1998 - 2015

- Conceptual **simplicity**; no need for new algorithms, or hyperparameter setting

- **No need to retrain** if skew ratio changes in deployment

- Retains **theoretical guarantees** of AdaBoost & decision theory

- Sound **probabilistic / decision-theoretic motivation**

Thank you!