Better Boosting with Bandits

Nikolaos Nikolaou[1], Joseph Mellor[2], Nikunj Oza[3] and Gavin Brown[1]
[1]School of Computer Science, University of Manchester; [2]School of Informatics, University of Edinburgh, [3]NASA Ames Research Center
nikolaos.nikolaou@manchester.ac.uk

Engineering and Physical Sciences Research Council

In brief…

- **Boosting is a successful family of ensemble learning algorithms. However, boosting algorithms despite being good classifiers – or rather because of it – exhibit poor probability estimation, which also makes them poor at cost-sensitive / imbalanced class learning.**
- **In recent work we established that all cost-sensitive boosting variants in the literature (1997-2016) fail to satisfy all desirable theoretical properties. The few that do satisfy all such properties, do so only after calibration of their scores.**
- **We proposed reserving part of the dataset to calibrate the scores of the ensemble and shifting the decision threshold accordingly to account for the cost / class imbalance – fast, easy to implement, satisfies all properties, can easily adjust to changes in costs / imbalance, matches or outperforms any other method on a wide range of cost-sensitive tasks.**
- **Current work focuses on extending these ideas to the online learning case. The main complication is that we can no longer split our data into a training & a calibration set; we need to decide for each minibatch whether to use it for training the ensemble or the calibrator.**
- **We propose the use of Bandit Algorithms resolve this decision. Our results show that bandit policies match or outperform naïve (fixed) policies in probability estimation on both stationary & non-stationary datasets. Moreover they are fast, robust to changes in the ensemble or the calibrator & easy to adapt to other tasks, such as cost-sensitive classification.**

Motivation

We want classifier scores to be **good probability estimates**
- To **quantify our uncertainty** over their predictions & know how much to **trust** them.
- To **combine predictions** from multiple sources.
- To make **better cost-sensitive decisions** (see study case below).

We want **online learning** algorithms
- To process **streaming data**.
- To deal with situations where the **problem** (e.g. data distribution) **changes over time**, **possibly** in an **adversarial** fashion.
- To **efficiently process big amounts of data**, even when they all are available at once.

Why **boosting**
- **Very successful in classification / regression** tasks – **60% of the winning Kaggle entries were based on boosting**.
- **Very good classifier but very poor at estimating probabilities** (no coincidence).
- **Very rich theory** motivating it & connecting it to other areas of machine learning. More recently **Residual Networks (ResNets), a state of the art Deep Neural Network architecture has been explained through boosting theory –also very good classifier but very poor at estimating probabilities** (we conjecture this is also not a coincidence).

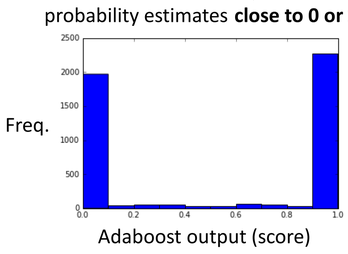
A study case: cost-sensitive boosting

All cost-sensitive boosting variants in the literature (1997-2016) fail to satisfy all desirable theoretical properties of (i) having **internally consistent steps** (ii) making **optimal decisions under decision theory**, (iii) preserving the **relative importance of the classes during training**, (iv) having **calibrated probability estimates**:

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)	✓	✓	✓	

Boosting probability estimates are inherently uncalibrated!
It minimizes monotonically decreasing loss functions of the margin. **Maximizing the margin increases the confidence of the classifier in its predictions**, which has been **connected to good generalization in classification**. But, the **generated probability estimates tend to 0 or 1 as the margins increase**.
What makes boosting a good classifier, also makes it a bad probability estimator!

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



Once **calibrated**, AdaMEC, CGAda & AsymAda (different approximations of same model) **satisfy all properties**. Based on theoretical soundness, flexibility, simplicity & results our suggestion for practitioners is **Calibrated AdaMEC**:

Reserve part of the dataset to calibrate the scores of the ensemble and shift the decision threshold accordingly to account for the cost / class imbalance.

<u>To find out more:</u>

- N. Nikolaou and G. Brown, ***Calibrating AdaBoost for Asymmetric Learning***, Multiple Classifier Systems, 2015
- N. Nikolaou, N. Edakunni, M. Kull, P. Flach and G. 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 – 12/129 eligible papers (9.3%)
  Best Paper Award 2016, School of Computer Science, University of Manchester
- N. Nikolaou, ***Cost-sensitive Boosting: A Unified Approach***, PhD Thesis, University of Manchester, 2016
  Best Thesis Award 2017, School of Computer Science, University of Manchester

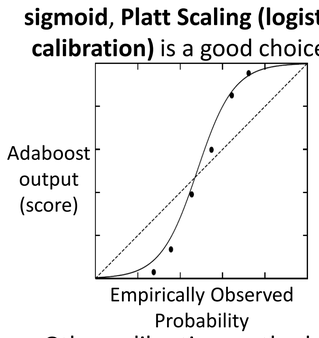
21st Annual BEST OF COMPUTING Notable Article ThinkLoud

- Calibrated AdaMEC python implementation: 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

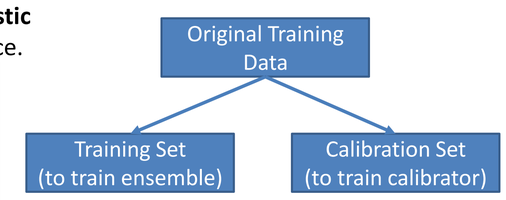
From batch to online calibration

So far (**batch learning**) we saw that boosting produces distorted probability estimates that need to be calibrated. Also, established that we can improve cost-sensitive boosting by first correcting the probability estimates. Can we apply the same principles to **online learning**?

Score distortion tends to be **sigmoid**, **Platt Scaling (logistic calibration)** is a good choice.



Other calibration methods (e.g. isotonic regression) are possible as well.



Batch learning: **split data into two separate parts** to avoid overfitting. **One to train ensemble, the other to train calibrator**.
(optimal split % can be decided by CV)

Online learning: **on each minibatch must decide whether to train ensemble or calibrator** – how to do this?

Fixed Policy: calibrate **every N rounds** – how to pick N?

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

What if we could **learn** a good sequence of alternating between **actions (train\calibrate)**?

Bandit algorithms to the rescue

Bandit problem: **set of actions (arms)** – on each round we choose one; each action associated with a **reward distribution;** each time an action is taken we **sample** its reward distribution.

Goal: **find the sequence of actions** that **minimize cumulative regret** (exploration vs. exploitation)

Algorithm : Deciding when to calibrate under a given *BanditPolicy*

For each round n **do:**
1 Receive unlabelled examples of $Minibatch_n$
2 Make predictions on examples of $Minibatch_n$
3 Receive labels of examples of $Minibatch_n$
4 Evaluate performance on $Minibatch_n$
5 **If** $n < 2$ **do:**
6 Update ensemble parameters on examples of $Minibatch_n$
7 **Else:**
8 Use reward X_n to update the parameters of *BanditPolicy*
9 Use ~~*BanditPolicy*~~ to decide action a_n
10 **If** a_n == *'TRAIN'* **do:**
11 Update ensemble parameters on examples of $Minibatch_n$
12 **Else:**
13 Update calibrator parameters on examples of $Minibatch_n$
14 Compute reward X_n of performing action a_n

Reward: increase in overall model likelihood after action is taken

Bandit Policies examined:

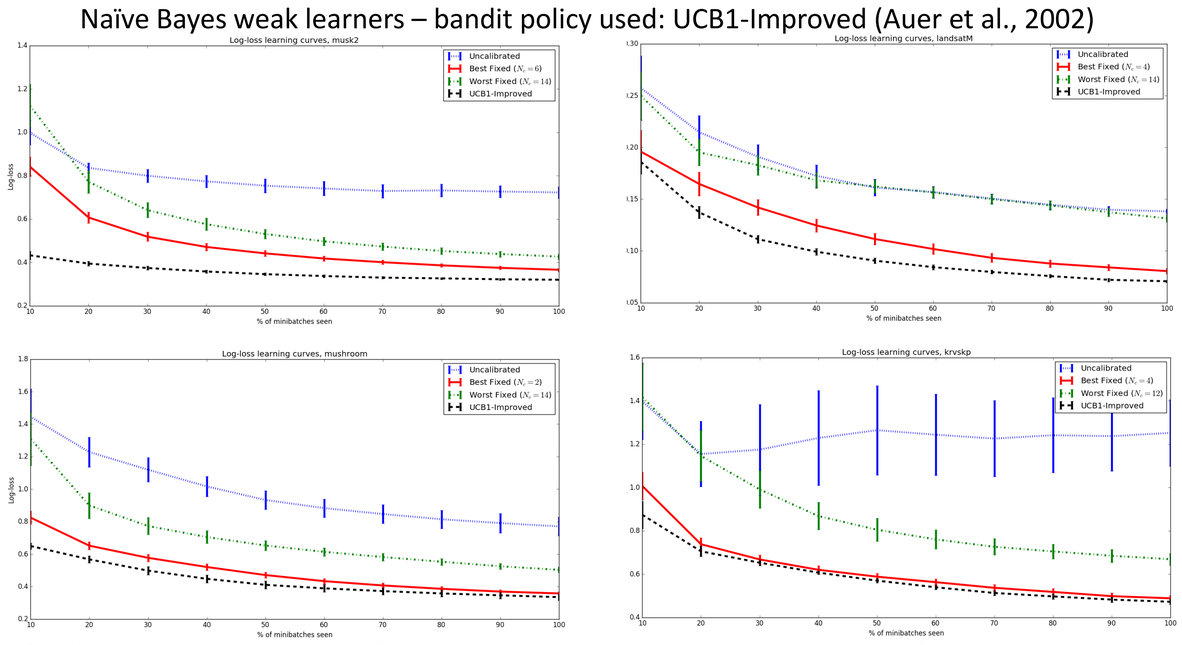
- **Thompson sampling**: **Bayesian** way of updating reward distribution; Assume **rewards Gaussian**; start with **Gaussian prior**, then **update** using **self-conjugacy of Gaussian distribution**
  Take action with highest posterior reward
- **UCB policies**: **'Optimism in the face of uncertainty'**
  Choose not the action with best expected reward, but that with **highest upper bound on reward**

- **Discounted rewards** (for all bandit policies examined):
  Weigh past rewards less; protects from **non-stationarity**

Why non-stationary? **Data distribution** might **change**…
most importantly: **reward distributions** will **change**

Results

Learning curves of avg log-loss per 10% of minibatches seen, for ensembles of M=10 Naïve Bayes weak learners – bandit policy used: UCB1-Improved (Auer et al., 2002)



Similar results for **other bandit policies**, **other weak learners**, **regularized weak learners**, **varying ensemble sizes**, **presence of inherent non-stationarity**

Some calibration (even worst naive) is better than no calibration

Bandit policies are online, fast, at least as good as 'best naïve' + adaptive to non-stationarity
Easy to adapt to other problems (e.g. cost-sensitive learning)
Robust to ensemble/calibrator hyperparameters

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

Acknowledgements: NN & GB were supported by the EPSRC grants [EP/I028099/1 & EP/L000725/1], NN Acknowledges the support of the EPSRC Doctoral Prize Fellowship.