Introduction

Balázs Kégl

LAL, CNRS/Université Paris Sud

SIMINOLE meeting

LAL, January 26, 2012

Introduction

- Discriminative learning
  - formalization, notation
- Short introduction to AdaBoost
  - the basic algorithm
  - multi-class / multi-label / multi-task
  - multiboost.org
- Extensions, applications, projects

The supervised learning model

- observation vector: $\mathbf{x} \in \mathbb{R}^d$
- class label: $y \in \{-1, 1\}$
- classifier: $g : \mathbb{R}^d \to \{-1, 1\}$
- discriminant function: $f : \mathbb{R}^d \to [-1, 1]$
  - $\longrightarrow$ classifier

$$g(\mathbf{x}) = \begin{cases} 1, & \text{if } f(\mathbf{x}) \geq 0, \\ -1, & \text{if } f(\mathbf{x}) < 0 \end{cases}$$

  - decision boundary: $\{\mathbf{x} : f(\mathbf{x}) = 0\}$

The supervised learning model

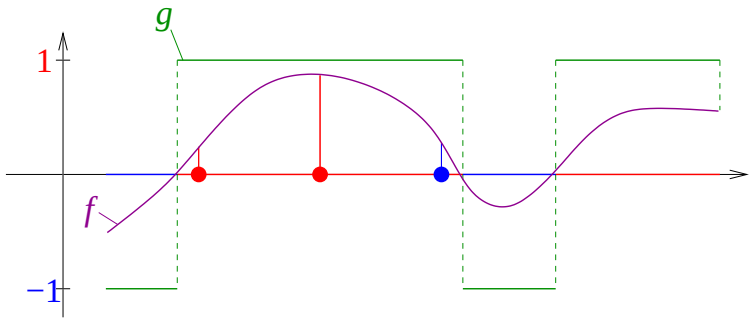
- Learning
  - training set : $D_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
  - function class : $\mathcal{F}$
  - learning algorithm : $\textsc{Algo} : \left(\mathbb{R}^d \times \{-1, 1\}\right)^n \to \mathcal{F}$

$$\textsc{Algo}(D_n) \mapsto f$$

  - goal: small generalization error $R(g) = P\left[g(\mathbf{X}) \neq Y\right] = P\left[f(\mathbf{X})Y \leq 0\right]$
  - learning principle: minimize the training error

$$\widehat{R}(g) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\{g(\mathbf{x}_i) \neq y_i\}$$

The supervised learning model

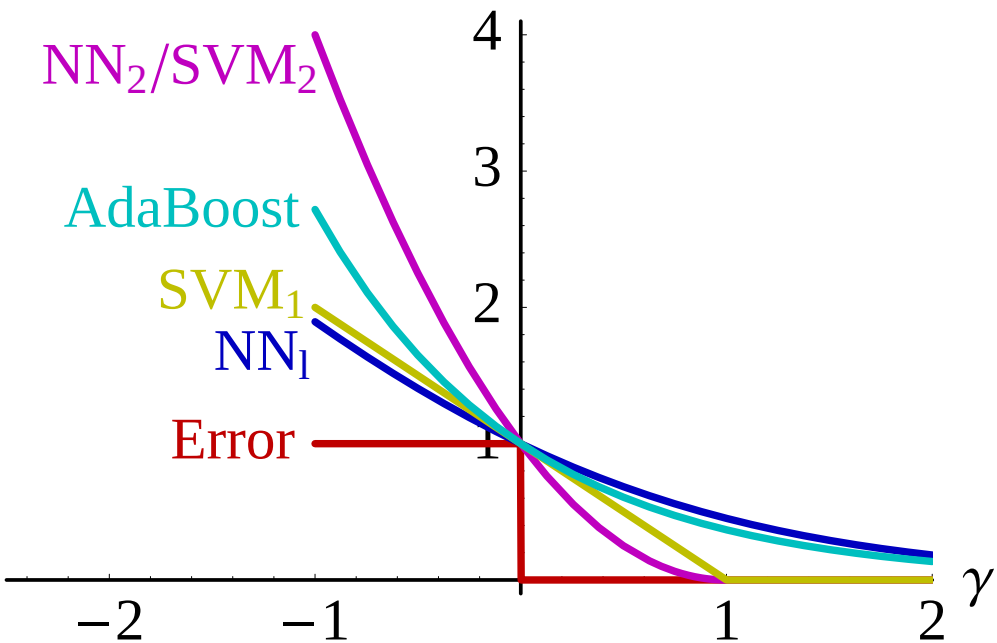


- Margin: $\gamma = y \cdot f(\mathbf{x})$
  - classification error $\equiv$ negative margin
  - the magnitude of a positive margin quantifies the confidence
  - learning principle: minimize a smooth loss function over the margin

$$\widehat{R}_\gamma(f) = \frac{1}{n} \sum_{i=1}^{n} L\big(f(\mathbf{x}_i) y_i\big)$$

The supervised learning model

- Margin loss functions



Generalized linear discriminant functions

- Model:

$$f(\mathbf{x}) = \sum_{j=1}^{N} \alpha^{(j)} h^{(j)}(\mathbf{x})$$

- $h^{(j)} : \mathbb{R}^d \to [-1, 1]$
  - simple classifiers/discriminant functions, features, experts
- $\alpha^{(j)} \in \mathbb{R}^+$
  - weight of the expert $h^{(j)}$ in the final vote

AdaBoost

- Model:

$$f(\mathbf{x}) = \sum_{j=1}^{N} \alpha^{(j)} h^{(j)}(\mathbf{x})$$

  - no restriction on the form of $h^{(j)}(\mathbf{x})$
  - often "decision stumps" :

$$h_{\ell,\theta}(\mathbf{x}) = \begin{cases} +1 & \text{if } x^{(\ell)} \geq \theta, \\ -1 & \text{otherwise} \end{cases}$$

    where $\mathbf{x} = \left(x^{(1)}, \ldots, x^{(d)}\right)$
  - trees, products, Haar filter

AdaBoost

- Intuitive elementary algorithm
  - add one expert at a time
  - add the best expert on training points mis-classified by previous experts
  - weight of the expert chosen proportionally to its correctness

AdaBoost

- Weighting over the training points $w_1, \ldots, w_n$
  - normalized: $\sum_{i=1}^{n} w_i = 1$
  - initialized uniformly : $\mathbf{w} = (1/n, \ldots, 1/n)$
  - if $\mathbf{x}_i$ is mis-classified by $h^{(j)}$, increase w_i
  - otherwise, decrease w_i
  - "difficult" training points get larger weights gradually

AdaBoost

- Base learning
  - binary base classifiers: $h^{(j)} : \mathcal{X} \to \{-1, 1\}$
  - weighted error

$$\varepsilon = \varepsilon(h) = \sum_{i=1}^{n} w_i \mathbb{I}\{h(\mathbf{x}_i) \neq y_i\}$$

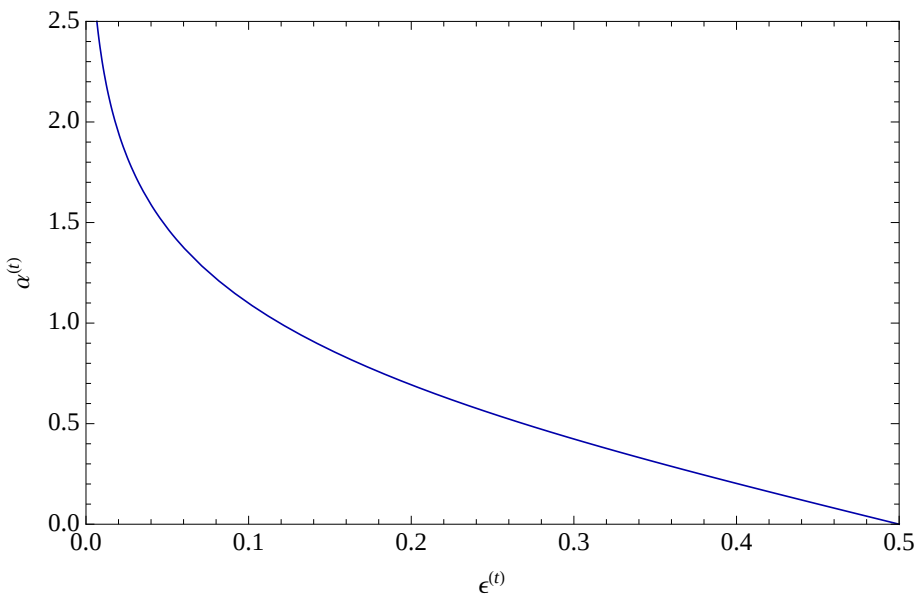
  - choose the base classifier that minimizes the weighted error

$$h^{(t)} = \arg\min_{h} \varepsilon(h)$$

AdaBoost

- Base learning
  - the coefficient of the base classifier

$$\alpha^{(t)} = \frac{1}{2} \ln \frac{1-\varepsilon}{\varepsilon}$$



AdaBoost

- Reweighting the points
  - si $h(\mathbf{x}_i) \neq y_i$ alors $w_i \leftarrow w_i \frac{1}{2\varepsilon}$
  - si $h(\mathbf{x}_i) = y_i$ alors $w_i \leftarrow w_i \frac{1}{2(1-\varepsilon)}$

AdaBoost

ADABOOST$(D_n, \text{BASE}(D_n, \mathbf{w}), T)$

1 $\mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ ▷ *initial weights*

2 **for** $t \leftarrow 1$ **to** T

3 $h^{(t)} \leftarrow \text{BASE}(D_n, \mathbf{w})$ ▷ *base learner*

4 $\varepsilon^{(t)} \leftarrow \sum_{h^{(t)}(\mathbf{x}_i) \neq y_i} w_i$ ▷ *weighted error*

5 $\alpha^{(t)} \leftarrow \frac{1}{2} \ln \left(\frac{1-\varepsilon^{(t)}}{\varepsilon^{(t)}} \right)$ ▷ *coefficient of* $h^{(t)}$

6 **for** $i \leftarrow 1$ **to** n ▷ *reweighting the points*

7 **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then**

8 $w_i^{(t+1)} \leftarrow \frac{w_i^{(t)}}{2\varepsilon^{(t)}}$

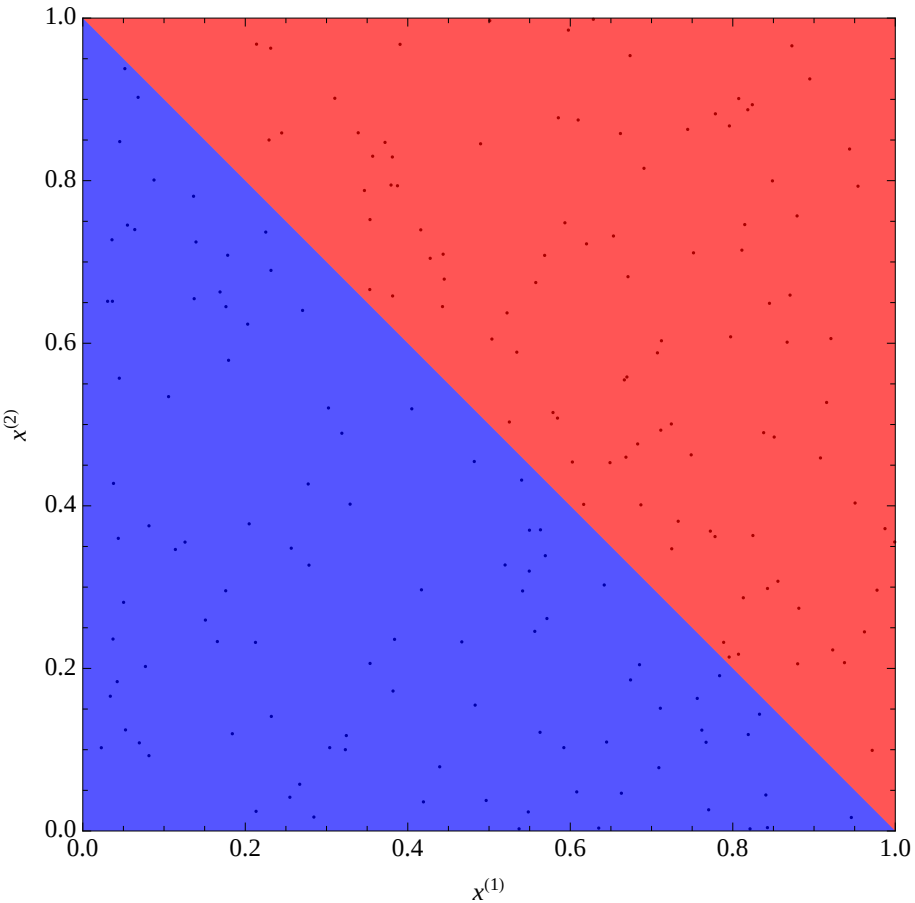
9 **else**

10 $w_i^{(t+1)} \leftarrow \frac{w_i^{(t)}}{2(1-\varepsilon^{(t)})}$

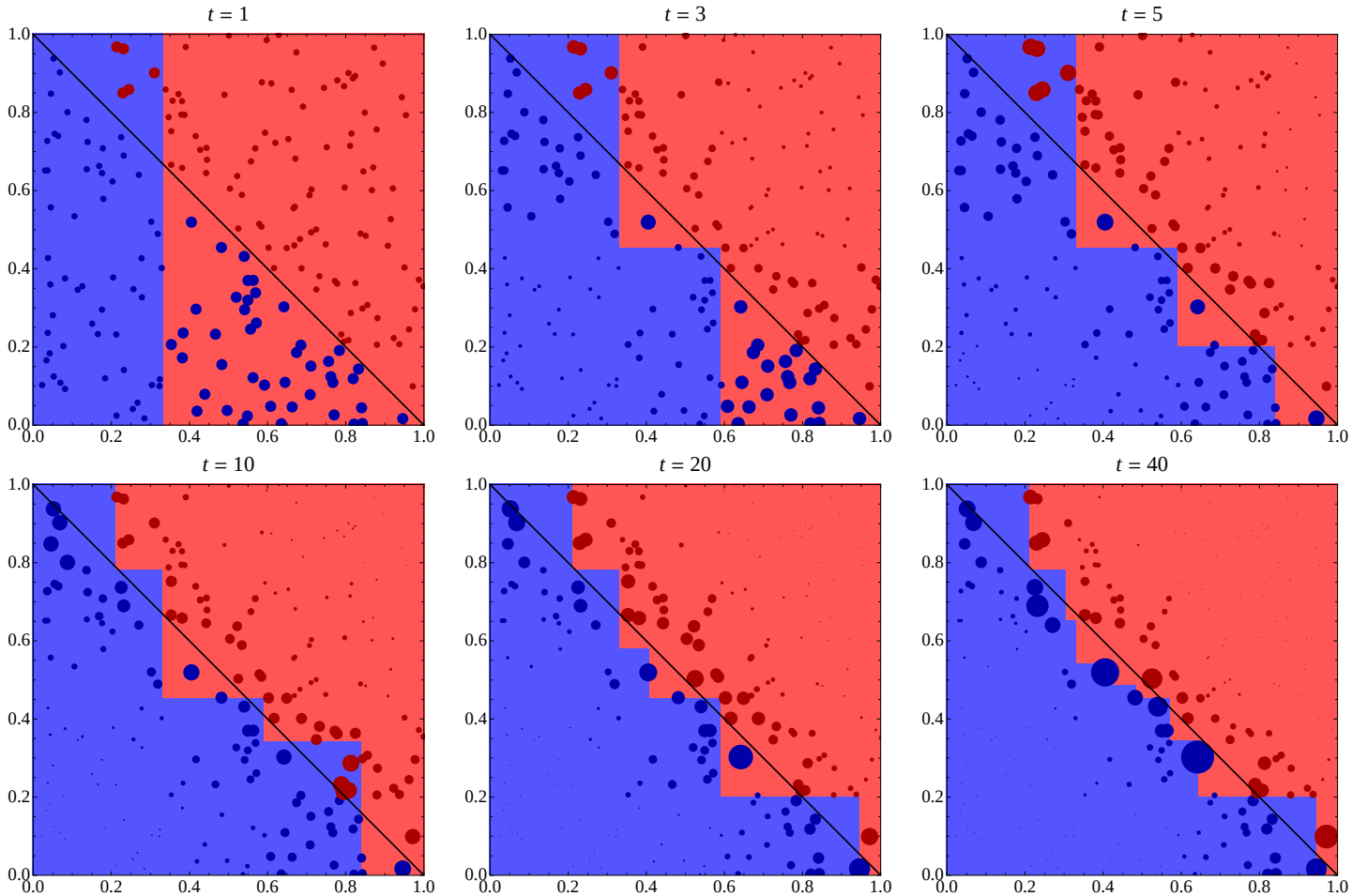
11 **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

AdaBoost

- Data set:

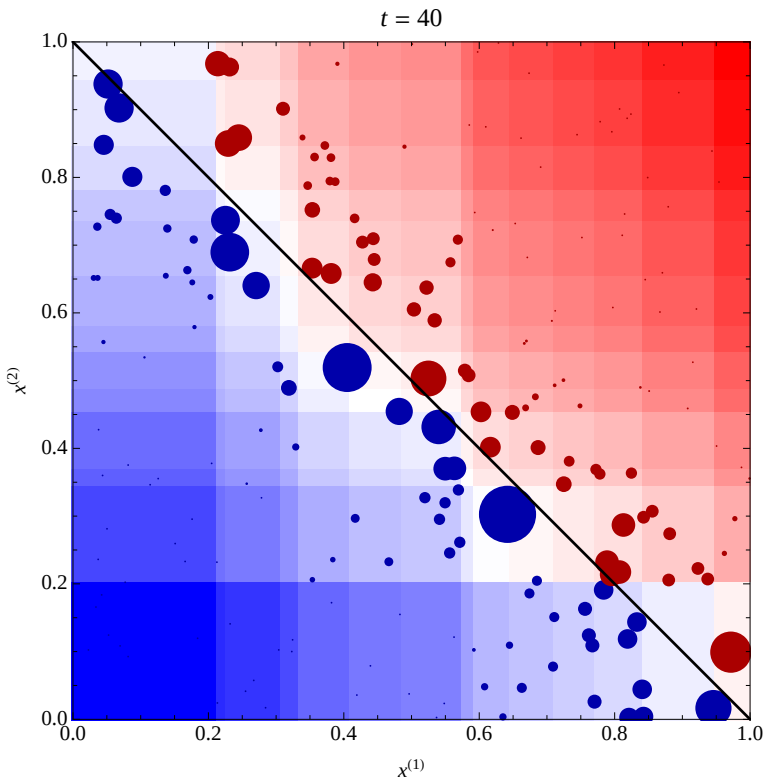


AdaBoost



AdaBoost

- Margins:



AdaBoost

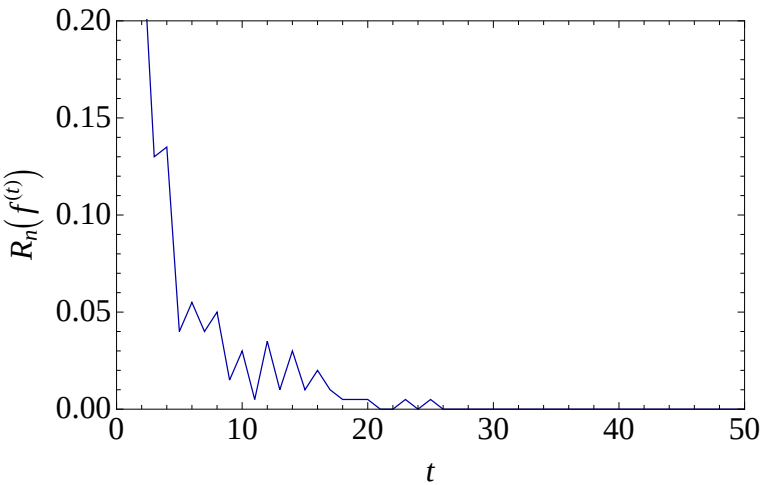
- Convergence
  - assume that $\varepsilon \leq \frac{1}{2} - \delta$ in every iteration: h is slightly than a random decision
  - the training error

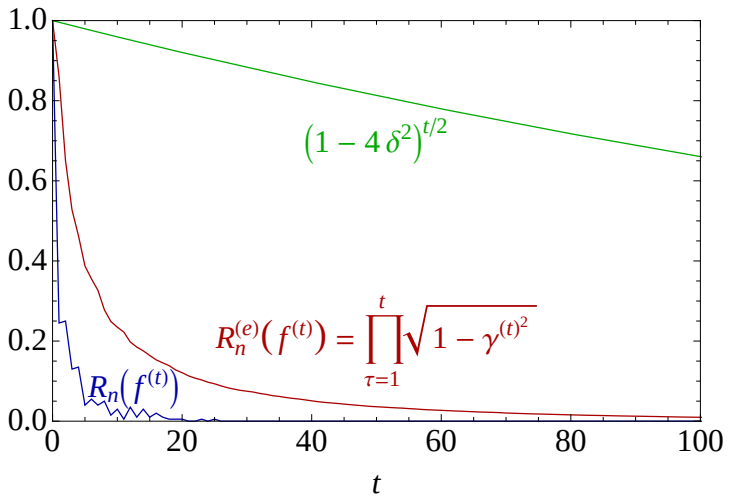
$$\widehat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{f(\mathbf{x}_i) y_i < 0\}$$

    is 0 after $\left\lceil \frac{\ln n}{2\delta^2} \right\rceil + 1$ iterations

AdaBoost

- Convergence





Multi-class

- Multi-class labels: $\mathbf{y} = \{-1, \ldots, -1, 1, -1, \ldots, -1\}$

$$y_\ell = \begin{cases} 1 & \text{if } \ell \text{ is the correct class } \ell(\mathbf{x}), \\ -1 & \text{otherwise} \end{cases}$$

- Multi-class discriminant function: $\mathbf{f} : \mathbf{x} \to \mathbb{R}^K$
  - notation: $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_K(\mathbf{x}))$
- Multi-class classifier:

$$g(\mathbf{x}) = \arg\max_{\ell \in \{1,\ldots,K\}} f_\ell(\mathbf{x})$$

Multi-class

- Multi-class base classifiers: $\mathbf{h} : \mathbf{x} \to \{-1, 1\}^K$
  - Convert a binary base classifer ϕ to multi-class:

$$\mathbf{h}(\mathbf{x}) = \mathbf{v} \cdot \phi(\mathbf{x})$$

  - $\mathbf{v} \in \{-1, 1\}^K$ is the vote vector

Multi-class

- Multi-class errors:

$$\widehat{R}(\mathbf{f}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left\{\ell(\mathbf{x}_i) \neq \widehat{\ell}(\mathbf{x}_i)\right\}$$

$$\widehat{R}\left(\mathbf{f}, \mathbf{w}^{(1)}\right) = \sum_{i=1}^{n}\sum_{\ell=1}^{K} w_{i,\ell}^{(1)} \mathbb{I}\left\{f_\ell(\mathbf{x}_i) y_\ell < 0\right\}$$

- Initial weighting

$$w_{i,\ell}^{(1)} = \begin{cases} \frac{1}{2n} & \text{if } \ell \text{ is the correct class (if } y_{i,\ell} = 1\text{),} \\ \frac{1}{2n(K-1)} & \text{otherwise (if } y_{i,\ell} = -1\text{).} \end{cases}$$

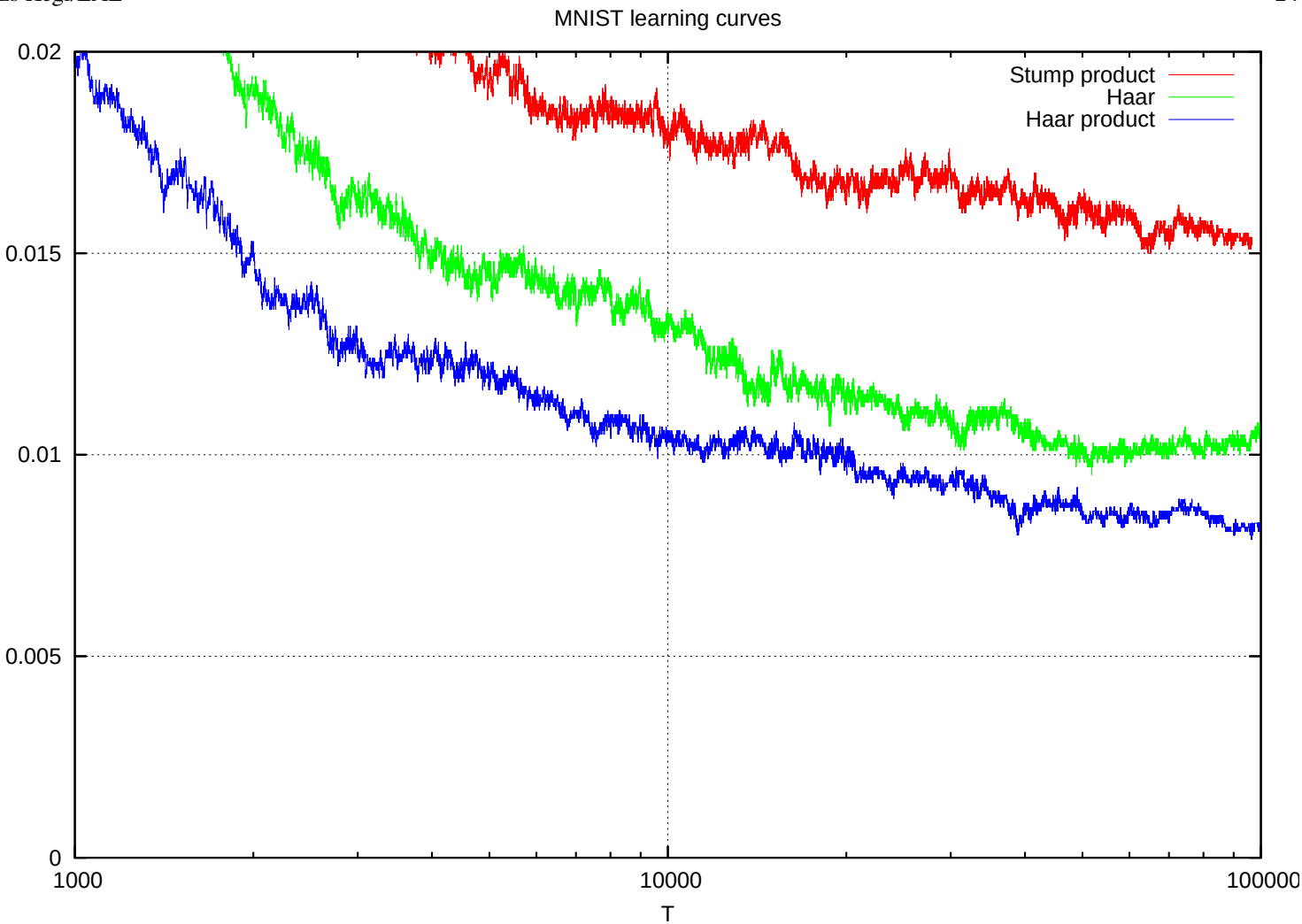
  - other initial weighting schemes for multi-label / multi-task

Multi-class

- AdaBoost.MH
  - weighting over the data points and over the classes:

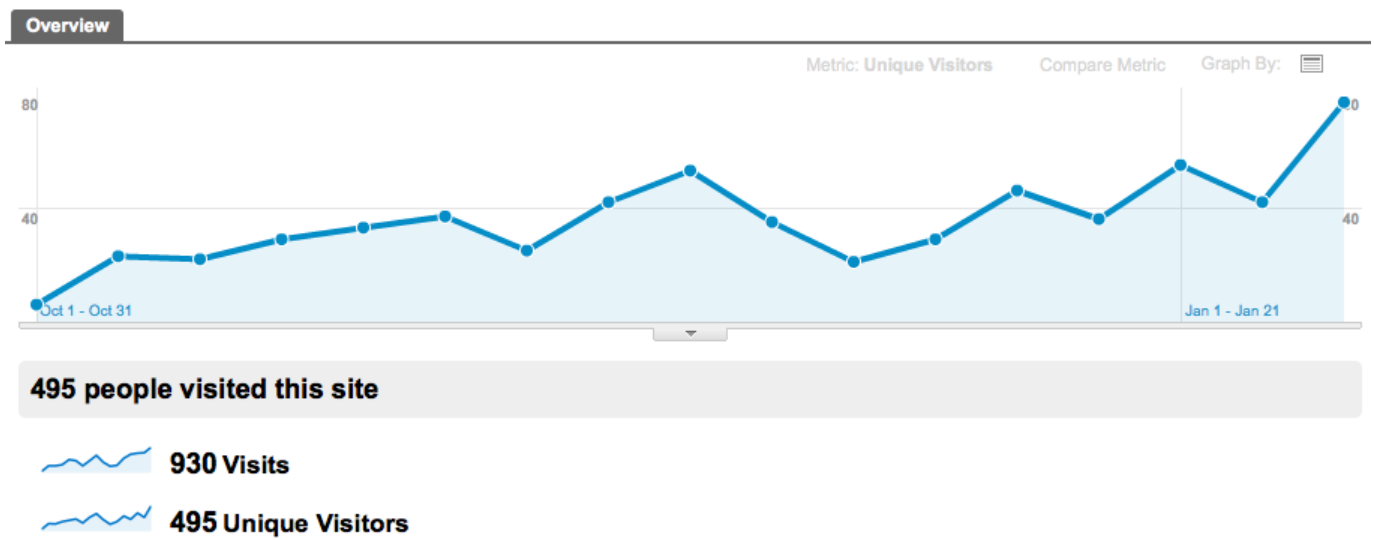
$$\left\{w_{i,\ell}\right\}_{i=1,\ldots n}^{\ell=1,\ldots,K}$$

  - $w_{i,\ell}$ is initialized to $w_{i,\ell}^{(1)}$
  - finding the best binary stump $\phi(\mathbf{x})$ and the best vote vector $\mathbf{v}$ remains efficient ($O(ndK)$)
  - trees ($O(ndKT\log N)$), products ($O(ndKTm)$)
  - Haar filters for images


MNIST learning curves
0.02
0.015
0.01
0.005
0
1000
10000
100000
T
Stump product
Haar
Haar product

multiboost.org

- Code, documentation, examples



D. Benbouzid, R. Busa-Fekete, N. Casagrande, F.-D. Collin, and B. Kégl, "Multiboost: a multi-purpose boosting package," *Journal of Machine Learning Research (submitted)*, 2011.

Bandit boosting (*Róbert*)

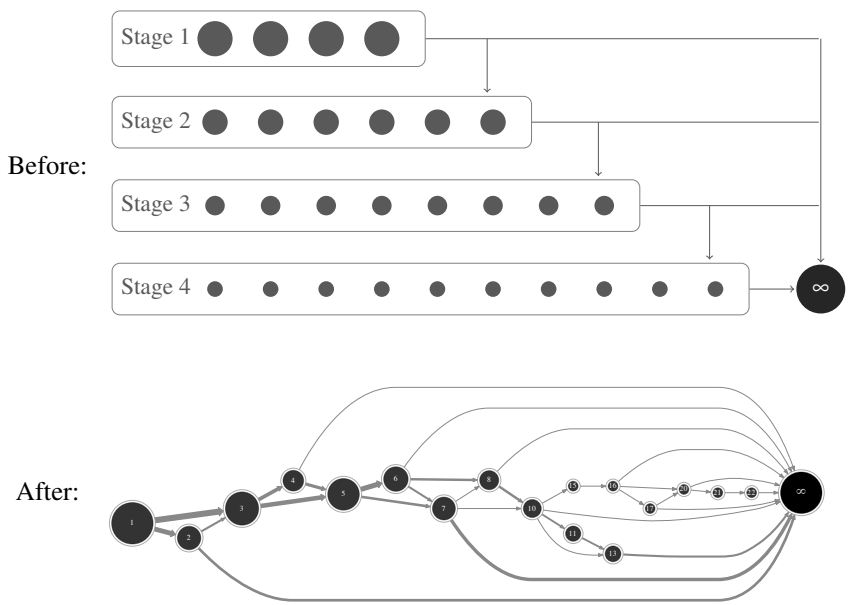
- Saving time at training
  - trees ($O(ndKT \log N)$), products ($O(ndKTm)$)
  - saving on n: stochastique boosting, FILTERBOOST
  - saving on d: LAZYBOOST, BANDITBOOST
  - adversarial bandits
  - Haar filters: we need adversarial Lipschitz bandits

Hyperparameter optimization (*Rémi, Matthias*)

- Saving time at tuning
  - find the best N, m, T using (cross-)validation
  - state of the art: grid search, random search
  - Gaussian-process-based surrogate optimization
  - ranking-based co-optimization of several problems in a global meta-feature space

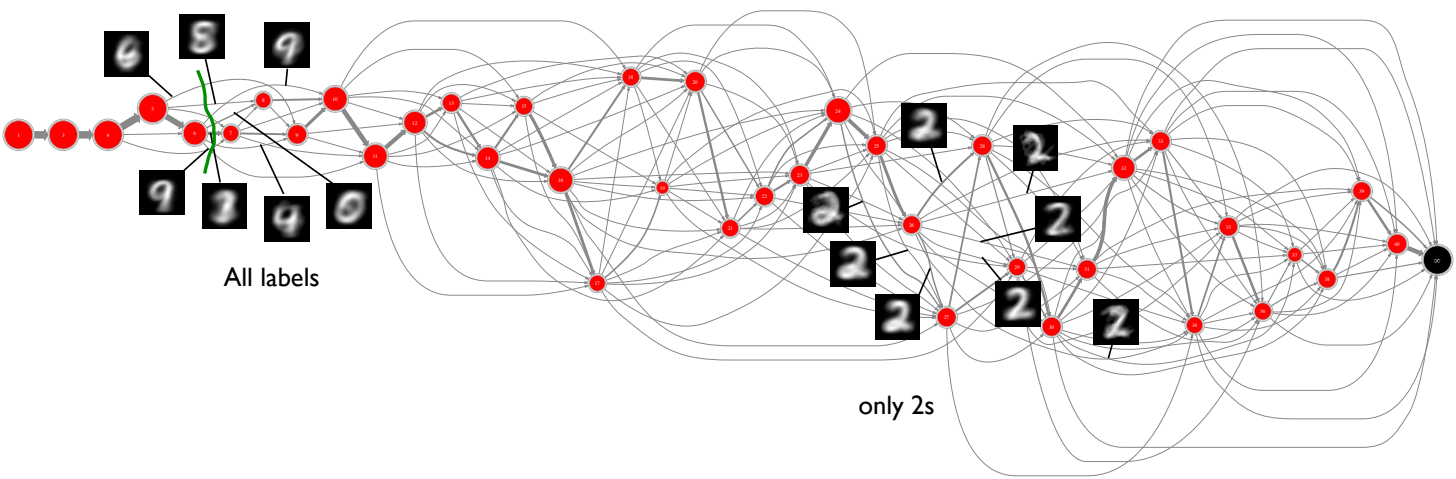
Sequential classifier design (*Djalel*)

- Saving time at testing: *lean classifiers*
  - object detection, trigger design, web-page ranking
  - state of the art: cascade classifiers



Sequential classifier design (*Djalel*)

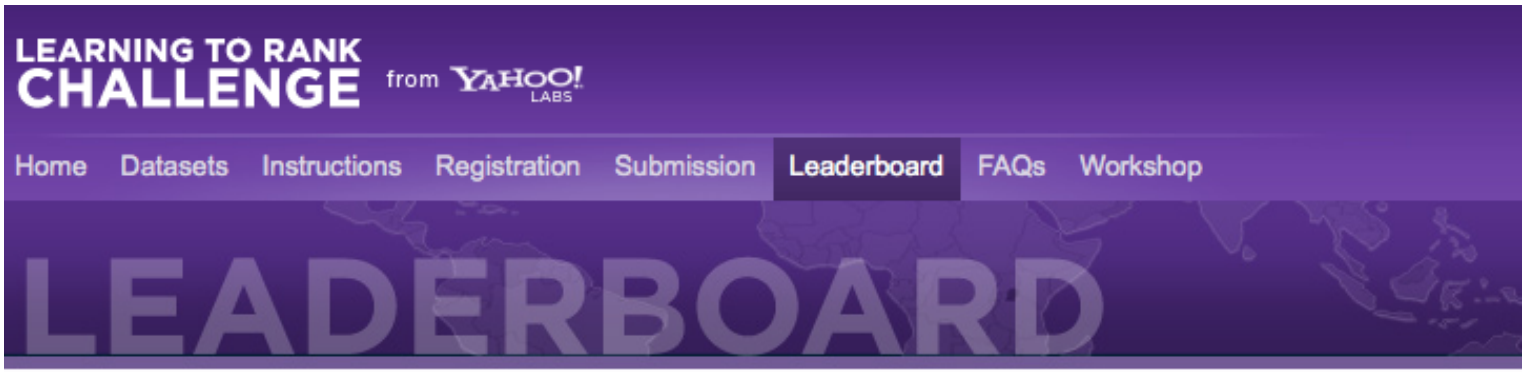
- Saving time at testing: *lean classifiers*
  - the Markov decision process setup
  - sparse coding, learning where to look, deep learning



Reconstructing $N_\mu(1000)$

- Extract 200 FADC signal observables (jumps, asymmetry, etc.)
- Preprocess (normalization, PCA, log label transformation)
- Train a (Bayesian) neural network for N_μ
- LDF reconstruction, bias correction

Ranking



Scores on the test sets: **Track 1** Track 2

Rank	Team Name	ERR Score	NDCG Score
1	Ca3Si2O7	0.468605	0.8041
2	catonakeyboardinspace	0.467857	0.806
3	MLG	0.466954	0.8026
4	Joker	0.466776	0.8053
5	AG	0.466157	0.8018
6	LAL	0.465748	0.7992
7	HotStepper	0.464658	0.7972
8	WashU in Saint Louis	0.464298	0.7983
9	ya	0.464258	0.7984
10	ULG-PG	0.464231	0.7967

Ranking

- Preprocessing (normalization, label grouping)
- Calibrated multiclass classifiers
  - multiple hyperparameter sets
  - diverse pointwise calibration
- Giant ensemble with exponential weighting
  - tuned for the listwise goal

Ranking

- Preprocessing (normalization, label grouping)
- Calibrated multiclass classifiers
  - multiple hyperparameter sets
  - diverse pointwise calibration
- Giant ensemble with exponential weighting
  - tuned for the listwise goal

Reinforcement learning

- Classification-based policy iteration
  - based on multiboost
- “Feels” natural for the sequential classifier design project
  - small dimensional continuous state space
  - two or three actions
  - near bimodal rewards
- Promising preliminary results