CS480/680: Introduction to Machine Learning

Lec 12: Boosting

Yaoliang Yu

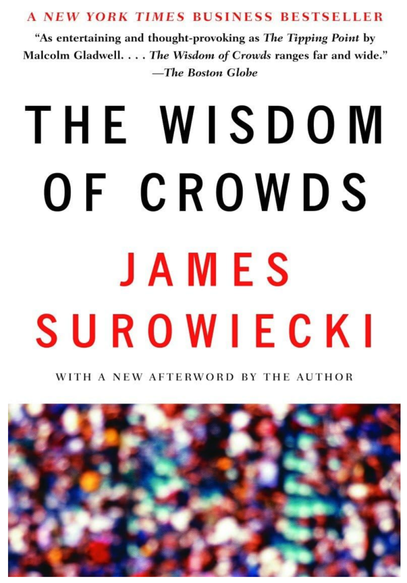
UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS
DAVID R. CHERITON SCHOOL OF COMPUTER SCIENCE

Feb 25, 2025

"Which Algorithm Should I Use for My Problem"

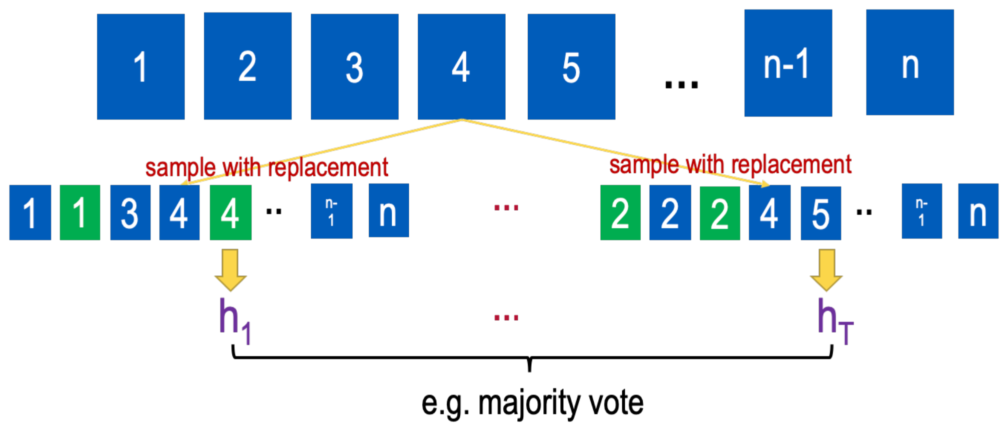
- Cheap answers
  - deep learning, but then which architecture?
  - I don't know; whatever the boss says
  - whatever I can find in xxxxx package
  - The one that runs fast!
  - Try a bunch and *pick* the "best"

- Why not combine a few algorithms? But how?



- Diversity
- Independence
- Trust

Bootstrap Aggregating



- Bootstrap if can't afford to have many independent training sets

L. Breiman. "Bagging Predictors". *Machine Learning*, vol. 24, no. 2 (1996), pp. 123–140.

Why and When Bagging Works

- With T i.i.d. classifiers h_t, averaging reduces variance by a factor of T
- Beneficial if classifiers have high variance (i.e. unstable)
  - performances change a lot if training set is slightly perturbed
  - simple models such as decision trees but not sophisticated ones

Randomized Output

- For regression, add small noise (e.g. Gaussian) to each y_i while leaving $\mathbf{x}_i$ unchanged
- For classification, can
  - use one-hot encoding and reduce to regression
  - randomly flip a small proportion of training labels
- Train many h_t and average/vote the results

L. Breiman. "Randomizing outputs to increase prediction accuracy". *Machine Learning*, vol. 40, no. 3 (2000), pp. 229–242.

Random Forest

- A collection of tree-structured classifiers $\{h(\mathbf{x}; \theta_t) : t = 1, \ldots, T\}$
  - θ_t are i.i.d. random
- Random feature split
- Random samples (bagging)

L. Breiman. "Random Forest". *Machine Learning*, vol. 45, no. 1 (2001), pp. 5–32.

Boosting

- Given a collection of classifiers h_t, each slightly better than random guessing
- Is it possible to construct a meta-classifier with nearly optimal accuracy?
- Yes!

```
if ϵ ≥ 1/2 − 1/p(n,s) then return WeakLearn(δ, EX)
α ← g^{-1}(ϵ)

EX_1 ← EX
h_1 ← Learn(α, δ/5, EX_1)
τ_1 ← ϵ/3
let â_1 be an estimate of a_1 = Pr_{v∈D}[h_1(v) ≠ c(v)]:
   choose a sample sufficiently large that |a_1 − â_1| ≤ τ_1 with probability ≥ 1 − δ/5
if â_1 ≤ ϵ − τ_1 then return h_1

defun EX_2()
   { flip coin
     if heads, return the first instance v from EX for which h_1(v) = c(v)
     else      return the first instance v from EX for which h_1(v) ≠ c(v) }
h_2 ← Learn(α, δ/5, EX_2)
τ_2 ← (1 − 2α)ϵ/8
let ê be an estimate of e = Pr_{v∈D}[h_2(v) ≠ c(v)]:
   choose a sample sufficiently large that |e − ê| ≤ τ_2 with probability ≥ 1 − δ/5
if ê ≤ ϵ − τ_2 then return h_2

defun EX_3()
   { return the first instance v from EX for which h_1(v) ≠ h_2(v) }
h_3 ← Learn(α, δ/5, EX_3)

defun h(v)
   { b_1 ← h_1(v), b_2 ← h_2(v)
     if b_1 = b_2 then return b_1
     else              return h_3(v) }
return h
```

1. Call **EX** m times to generate a sample $S = \{(x_1, l_1), \ldots, (x_m, l_m)\}$. To each example (x_j, l_j) in S corresponds a weight w_j and count r_j. Initially, all weights are $1/m$ and all counts are zero.
2. Find a (small) k that satisfies

$$\sum_{i=\lceil k/2 \rceil}^{k} \binom{k}{i} (1/2 - \gamma)^i (1/2 + \gamma)^{k-i} < \frac{1}{m}$$

   (For example, any $k > 1/(2\gamma^2)\ln(m/2)$ is sufficient.)
3. Repeat the following steps for $i = 1 \ldots k$.
   (a) repeat the following steps for $l = 1 \ldots (1/\lambda)\ln(2k/\delta)$ or until a weak hypothesis is found.
      i. Call **WeakLearn**, referring it to **FiltEX** as its source of examples, and save the returned hypothesis as h_i.
      ii. Sum the weights of the examples on which $h_i(x_j) \neq l_j$. If the sum is smaller than $1/2 - \gamma$ then declare h_i a weak hypothesis and exit the loop.
   (b) Increment r_j by one for each example on which $h_i(x_j) = l_j$.
   (c) Update the weights of the examples according to $w_j = \alpha^i_{r_j}$, α^i_r is defined in Equation (1).
   (d) Normalize the weights by dividing each weight by $\sum_{j=1}^{m} w_j$.
4. Return as the final hypothesis, h_M, the majority vote over $h_1, \ldots, h_k$.

Subroutine FiltEX

1. choose a real number x uniformly at random in the range $0 \leq x < 1$.
2. Perform a binary search for the index j for which

$$\sum_{i=1}^{j-1} w_i \leq x < \sum_{i=1}^{j} w_i$$

   ($\sum_{i=1}^{0} w_i$ is defined to be zero.)
3. Return the example (x_j, l_j)

R. E. Schapire. "The strength of weak learnability". *Machine Learning*, vol. 5, no. 2 (1990), pp. 197–227, Y. Freund. "Boosting a Weak Learning Algorithm by Majority". *Information and Computation*, vol. 121, no. 2 (1995), pp. 256–285.

Algorithm 1: Hedging.

Input: initial weight vector $\mathbf{w}_1 \in \mathbb{R}^n_{++}$, discount factor $\beta \in [0, 1]$

Output: last weight vector $\mathbf{w}_{T+1}$

for $t = 1, 2, \ldots, T$ **do**

  learner chooses probability vector $\mathbf{p}_t = \mathbf{w}_t / \langle \mathbf{1}, \mathbf{w}_t \rangle$ // normalization

  environment chooses loss vector $\boldsymbol{\ell}_t \in [0, 1]^n$ // $\boldsymbol{\ell}_t$ may depend on $\mathbf{p}_t$!

  learner suffers (expected) loss $\langle \mathbf{p}_t, \boldsymbol{\ell}_t \rangle$

  learner updates weights $\mathbf{w}_{t+1} = \mathbf{w}_t \odot \beta^{\boldsymbol{\ell}_t}$ // element-wise product $\odot$ and power

  optional scaling: $\mathbf{w}_{t+1} \leftarrow c_{t+1}\mathbf{w}_{t+1}$ // $c_{t+1} > 0$ can be arbitrary

- n horses in a race, repeated for T rounds
- p_{it} is the proportion of money bet on the i-th horse at round t
- ℓ_{it} is the loss on the i-th horse at round t
- 🥕 for the winning horses and 🏒 for the losing ones

Y. Freund and R. E. Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". *Journal of Computer and System Sciences*, vol. 55, no. 1 (1997), pp. 119–139.

Theorem: Hedging guarantee

Let $L_{\min} := \min_i L_i = \min_i \sum_{t=0}^{T} \ell_{it}$ and $\mathbf{w}_1 = \mathbf{1}$. We have

$$\sum_{t=1}^{T} \langle \mathbf{p}_t, \boldsymbol{\ell}_t \rangle \leq \frac{L_{\min} \ln \frac{1}{\beta} + \ln \mathsf{n}}{1-\beta}$$

With $\beta = \frac{1}{1+\sqrt{(2\ln n)/L_{\min}}}$, we have

$$\frac{1}{T} \sum_{t=1}^{T} \langle \mathbf{p}_t, \boldsymbol{\ell}_t \rangle \leq \frac{L_{\min}}{T} + \sqrt{\frac{2 \ln \mathsf{n}}{T}} + \frac{\ln \mathsf{n}}{T}$$

- Logarithmic dependence on n: can bet on many horses!
- Square root dependence on T
- In the long run, can do no worse than the best horse (with hindsight)

Algorithm 2: Adaptive Boosting.

Input: initial weight $\mathbf{w}_1 \in \mathbb{R}^{\mathsf{n}}_{++}$, training set $\mathcal{D}_{\mathsf{n}} = \{\!\{(\mathbf{x}_i, \mathsf{y}_i)\}\!\}_{i=1}^{\mathsf{n}} \subseteq \mathbb{R}^d \times \{0, 1\}$

Output: meta-classifier $\bar{h} : \mathbb{R}^d \to \{0, 1\}$, $\mathbf{x} \mapsto \left[\!\left[\sum_{t=1}^{T} (\ln \frac{1}{\beta_t})(h_t(\mathbf{x}) - \frac{1}{2}) \geq 0 \right]\!\right]$

for $t = 1, 2, \ldots, T$ **do**

  $\mathbf{p}_t = \mathbf{w}_t / \langle \mathbf{1}, \mathbf{w}_t \rangle$ // normalization

  $h_t \leftarrow \mathsf{WeakLearn}(\mathcal{D}_{\mathsf{n}}, \mathbf{p}_t)$ // t-th weak classifier $h_t : \mathbb{R}^d \to [0, 1]$

  $\forall i,\ \ell_{it} = 1 - |h_t(\mathbf{x}_i) - \mathsf{y}_i|$ // higher loss if more accurate!

  $\epsilon_t = 1 - \langle \mathbf{p}_t, \boldsymbol{\ell}_t \rangle = \sum_{i=1}^{\mathsf{n}} p_{it} |h_t(\mathbf{x}_i) - \mathsf{y}_i|$ // weighted error of h_t

  $\beta_t = \epsilon_t / (1 - \epsilon_t)$ // adaptive discounting $\beta_t \leq 1 \iff \epsilon_t \leq \frac{1}{2}$

  $\mathbf{w}_{t+1} = \mathbf{w}_t \odot \beta_t^{\boldsymbol{\ell}_t}$ // element-wise product $\odot$ and power

  optional scaling: $\mathbf{w}_{t+1} \leftarrow c_{t+1} \mathbf{w}_{t+1}$ // $c_{t+1} > 0$ can be arbitrary

Y. Freund and R. E. Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". *Journal of Computer and System Sciences*, vol. 55, no. 1 (1997), pp. 119–139.

Properties of Adaboost

- Expected error $\epsilon_t \leq \frac{1}{2} \iff \beta_t \in [0,1]$, adaptive and automatic
  - what if $\epsilon_t > \frac{1}{2}$?
- Each weak classifier focuses on hard examples that are *misclassified* before

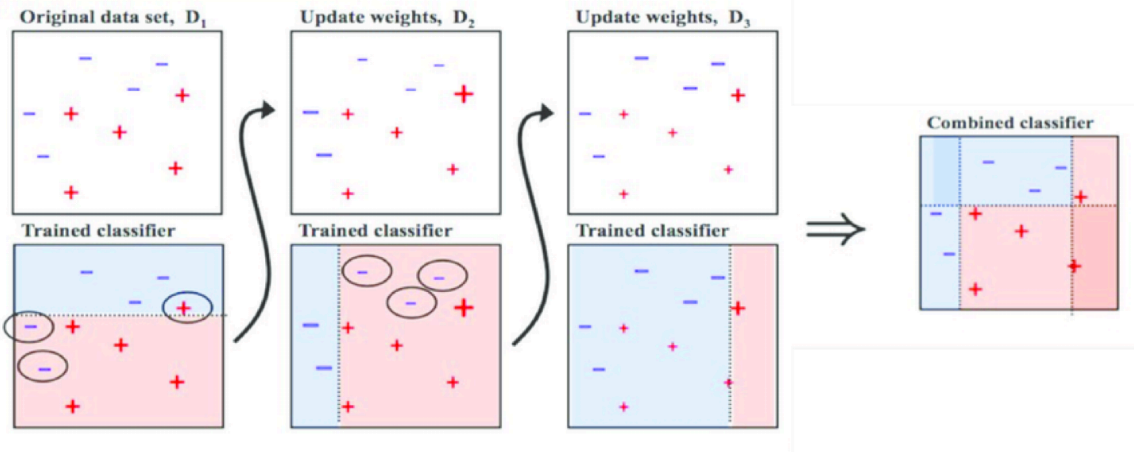
$$w_{i,t+1} = w_{i,t} \cdot \beta_t^{1-|h_t(\mathbf{x}_i)-\mathsf{y}_i|}$$

  - when will $w_{i,t}$ become 0?
- Meta-classifier $\bar{h}$ aggregates the history, with weight $\ln \frac{1}{\beta_t}$ for the t-th classifier
  - which classifier gets higher weight?
- No same classifier in a row (assuming $h_t \in \{0,1\}$):

$$\epsilon_{t+1}(h_t) \equiv \frac{1}{2}, \quad \text{where} \quad \epsilon_t(h) := \sum_{i=1}^{n} p_{it}|h(\mathbf{x}_i) - \mathsf{y}_i|$$

  - what happens to β_t and $\ln \frac{1}{\beta_t}$?

Does It Work?



Original data set, D_1
Trained classifier
Update weights, D_2
Trained classifier
Update weights, D_3
Trained classifier
Combined classifier

Theorem: Exponential decay of training error

The meta-classifier $\bar{h}$ of Adaboost satisfies:

$$\sum_{i=1}^{n} p_{i1} \left[\!\left[\bar{h}(\mathbf{x}_i) \neq y_i \right]\!\right] \leq \prod_{t=1}^{T} \sqrt{4\epsilon_t(1-\epsilon_t)}.$$

Assuming $|\epsilon_t - \frac{1}{2}| > \gamma_t$, then

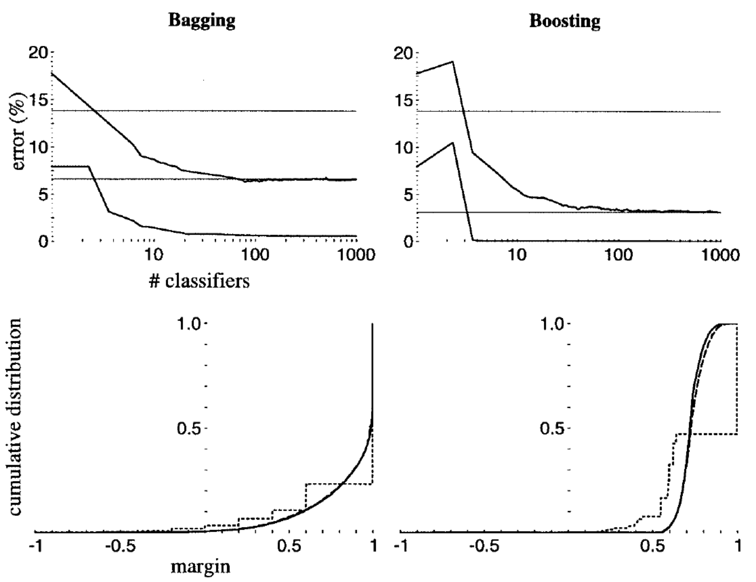
$$\sum_{i=1}^{n} p_{i1} \left[\!\left[\bar{h}(\mathbf{x}_i) \neq y_i \right]\!\right] \leq \prod_{t=1}^{T} \sqrt{1-4\gamma_t^2} \leq \exp\left(-2\sum_{t=1}^{T} \gamma_t^2\right).$$

In particular, if $\gamma_t \geq \gamma$ for all t, then

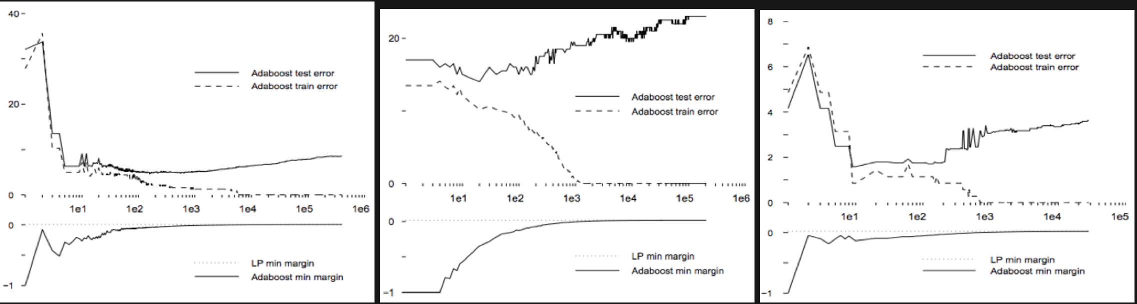
$$\sum_{i=1}^{n} p_{i1} \left[\!\left[\bar{h}(\mathbf{x}_i) \neq y_i \right]\!\right] \leq \exp(-2T\gamma^2).$$

- To achieve ϵ (weighted) training error, combine at most $T = \lceil \frac{1}{2\gamma^2} \ln \frac{1}{\epsilon} \rceil$ weak classifiers, each of which slightly better than random guessing (by a margin of γ)

Will Adaboost Overfit?



LPboost

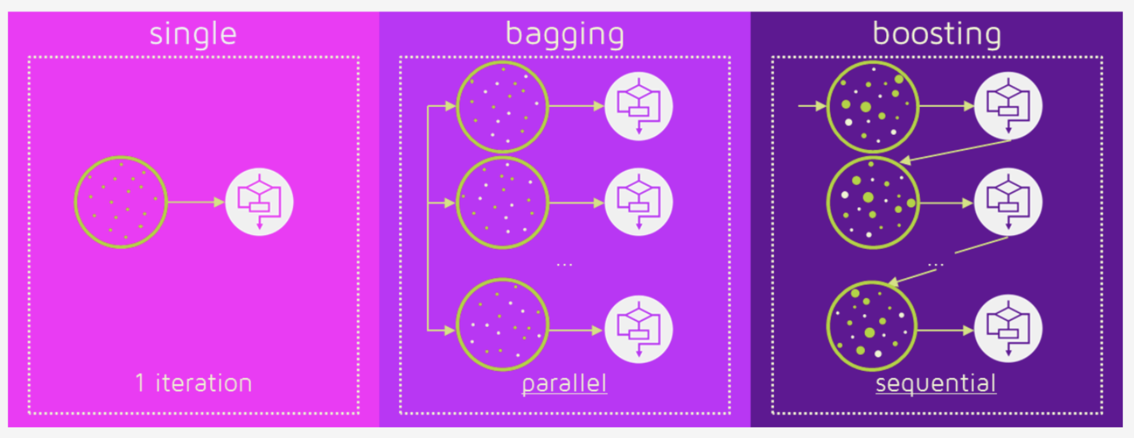


A. J. Grove and D. Schuurmans. "Boosting in the Limit: Maximizing the Margin of Learned Ensembles". In: *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. 1998, pp. 692–699, L. Breiman. "Prediction Games and Arcing Algorithms". *Neural Computation*, vol. 11, no. 7 (1999), pp. 1493–1517.

Data set	C4.5		Adaboost		LP-Adaboost			DualLPboost		
	error%	win%	error%	margin	error%	win%	margin	error%	win%	margin
Audiology	22.70	17.0	16.39	0.446	16.48	49.0	0.501	18.09	38.5	0.370
Banding	25.58	12.5	15.00	0.528	15.42	45.5	0.565	22.50	20.0	0.430
Chess	4.18	12.5	2.70	0.657	2.74	46.5	0.730	2.97	37.0	0.560
Colic	14.46	67.5	17.03	0.051	18.97	31.5	0.182	18.16	44.0	0.108
Glass	30.91	22.0	23.95	0.513	23.91	49.5	0.624	26.86	38.0	0.386
Hepatitis	21.06	38.0	18.94	0.329	17.56	59.0	0.596	20.00	45.5	0.385
Labor	15.33	43.0	12.83	0.535	13.83	47.0	0.684	15.17	42.0	0.599
Promoter	21.09	10.5	7.55	0.599	8.00	47.0	0.694	13.55	29.5	0.378
Sonar	28.81	16.0	18.10	0.628	18.62	48.0	0.685	25.00	23.0	0.478
Soybean	8.86	28.5	6.97	-0.005	6.55	62.0	0.017	8.41	33.5	0.003
Splice	16.18	0.0	6.83	0.535	7.00	25.0	0.569	11.01	0.0	0.393
Vote	4.95	51.0	5.02	0.723	5.30	44.5	0.795	5.27	44.5	0.756
Wine	9.11	27.0	4.61	0.869	4.89	47.5	0.912	4.50	50.5	0.814

Pros and Cons

- "Straightforward" way to boost performance
- Flexible: can work with any base classifiers
- Less interpretable
- Longer training time
  - harder to parallelize, compared to bagging



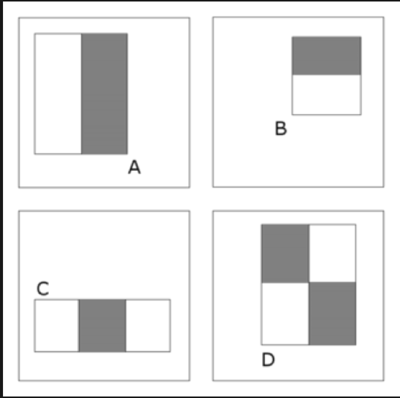
https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

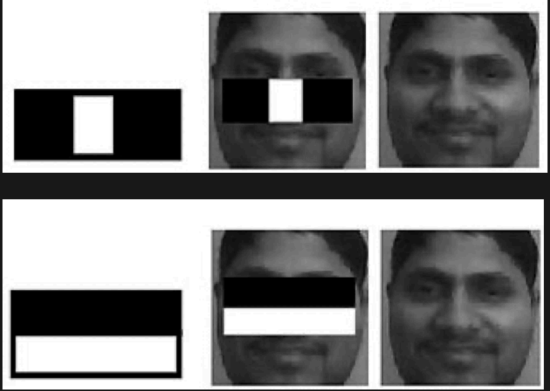
Extensions

- LogitBoost
- GradBoost
- L2Boost
- XGboost

- Multi-class
- Regression
- Ranking

Face Detection





- Each detection window results in $\approx$ 160k features
- Speed is crucial for real-time detection

P. Viola and M. J. Jones. "Robust Real-Time Face Detection". *International Journal of Computer Vision*, vol. 57, no. 2 (2004), pp. 137–154.