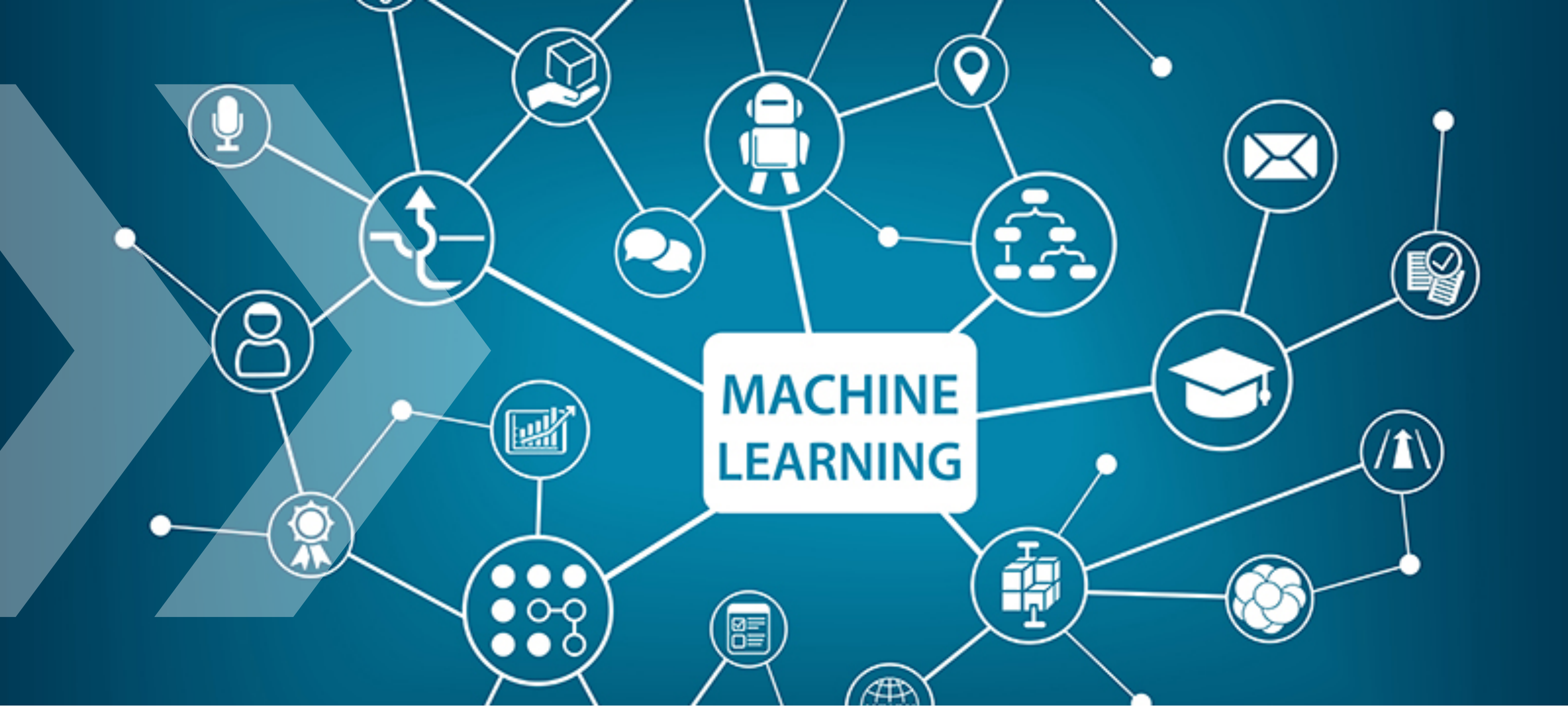


CS480/680: Intro to ML

Lecture 11: Bagging and Boosting

UNIVERSITY OF WATERLOO

 2020-06-16 Yao-Liang Yu

“Adults do not need to choose”

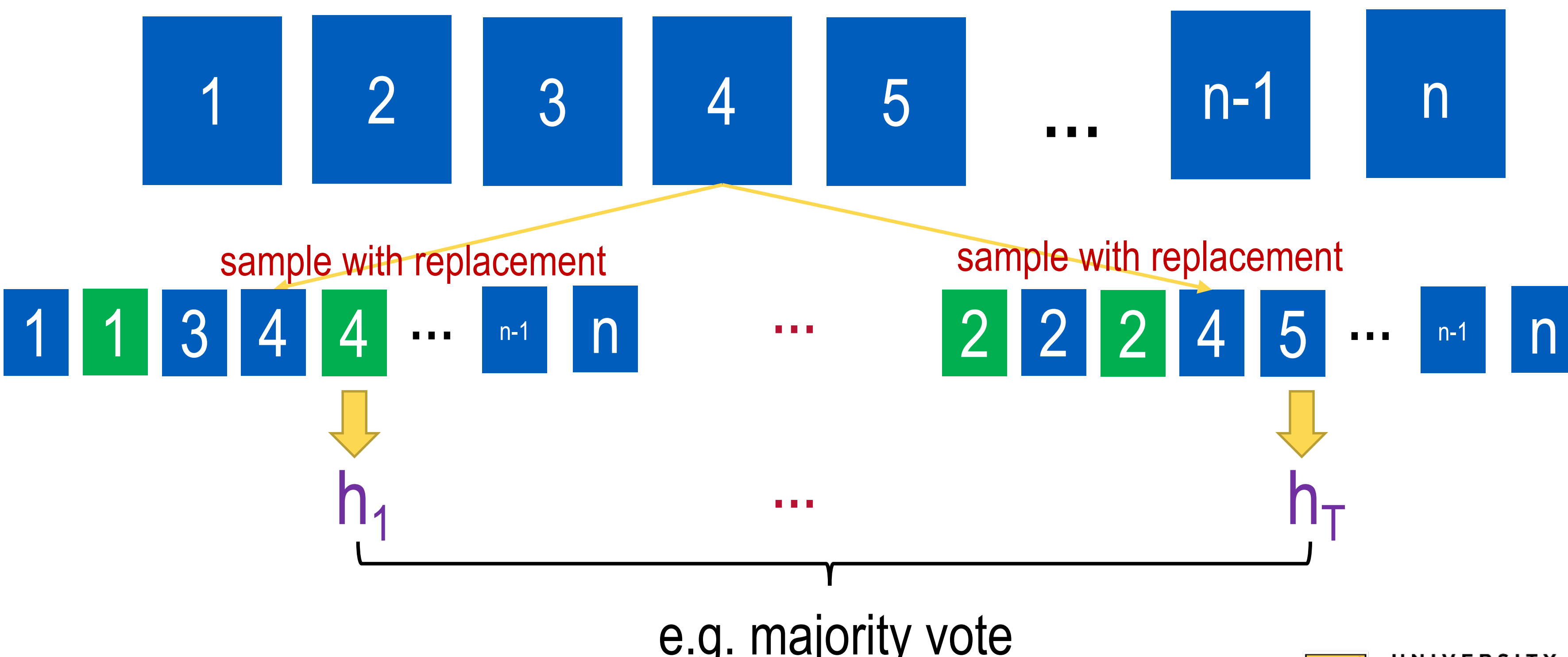
- Which algorithm to use for my problem?

- Cheap answers
  - Deep learning, but then which architecture?
  - I don’t know; whatever my boss tells me
  - Whatever I can find in *scikit-learn*
  - I try many and pick the “best”

- Why don’t we combine a few algs? But how?

Bootstrap Aggregating (Breiman'96)

- Can't afford to have many independent training sets
- Bootstrapping!



Bagging for Regression

- Simply average the outputs of h_t, each trained on a bootstrapped training set

- With T ~~independent~~ h_t, averaging would reduce variance by a factor of T

UNIVERSITY OF WATERLOO

When Bagging Works

- Bagging is essentially averaging

- Beneficial if base classifiers have high variance (instable); e.g., performance changes a lot if training set is slightly perturbed

- Like decision trees but not k-NNs

Randomizing Output (Breiman'00)

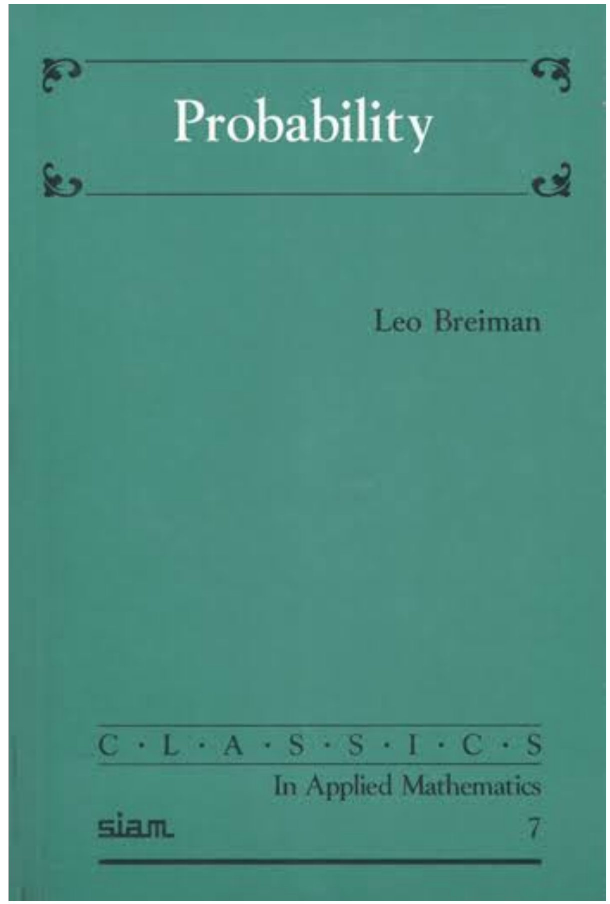
- For regression, add small Gaussian noise to each y_i (leaving x_i untouched)
- Train many h_t and average their outputs

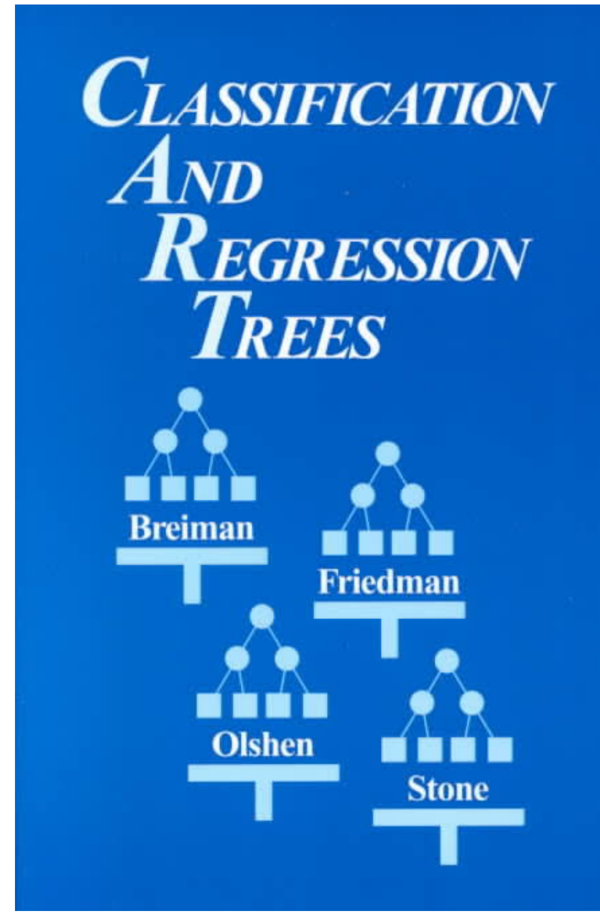
- For classification
  - Use one-hot encoding and reduce to regression
  - Randomly flip a small proportion of labels in training set
- Train many h_t and majority vote

Random Forest (Breiman'01)

- A collection of tree-structured classifiers $\{h(x; \Theta_t), t=1, \ldots, T\}$, where Θ_t are iid random vectors
- Bagging: random samples
- Random feature split
- Both

Leo Breiman (1928-2005)





Boosting

- Given many classifiers h_t, each slightly better than random guessing
- Is it possible to construct a classifier with nearly optimal accuracy?

- Yes! First shown by Schapire (1990)

The Power of Majority

if $\epsilon \geq 1/2 - 1/p(n,s)$ **then return** WeakLearn(δ, EX)
$\alpha \leftarrow g^{-1}(\epsilon)$

$EX_1 \leftarrow EX$
$h_1 \leftarrow$ Learn($\alpha, \delta/5, EX_1$)
$\tau_1 \leftarrow \epsilon/3$
let $\hat{a}_1$ be an estimate of $a_1 = \Pr_{v \in D}[h_1(v) \neq c(v)]$:
choose a sample sufficiently large that $|a_1 - \hat{a}_1| \leq \tau_1$ with probability $\geq 1 - \delta/5$
if $\hat{a}_1 \leq \epsilon - \tau_1$ **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 $h_1(v) \neq c(v)$ }
$h_2 \leftarrow$ Learn($\alpha, \delta/5, EX_2$)
$\tau_2 \leftarrow (1-2\alpha)\epsilon/8$
let $\hat{e}$ be an estimate of $e = \Pr_{v \in D}[h_2(v) \neq c(v)]$:
choose a sample sufficiently large that $|e - \hat{e}| \leq \tau_2$ with probability $\geq 1 - \delta/5$
if $\hat{e} \leq \epsilon - \tau_2$ **then return** h_2

defun $EX_3()$
{ **return** the first instance v from EX for which $h_1(v) \neq h_2(v)$ }
$h_3 \leftarrow$ Learn($\alpha, \delta/5, EX_3$)

defun $h(v)$
{ $b_1 \leftarrow h_1(v)$, $b_2 \leftarrow h_2(v)$
if $b_1 = b_2$ **then return** b_1
else **return** $h_3(v)$ }
return h

(Schapire, 1990)

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)

(Freund, 1995)

UNIVERSITY OF WATERLOO

Hedging (Freund & Schapire'97)

Algorithm Hedge(β)

Parameters: $\beta \in [0, 1]$

initial weight vector $\mathbf{w}^1 \in [0, 1]^N$ with $\sum_{i=1}^N w_i^1 = 1$

number of trials T

Do for $t = 1, 2, ..., T$

1. Choose allocation

$$\mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{i=1}^N w_i^t}$$

2. Receive loss vector $\boldsymbol{\ell}^t \in [0, 1]^N$ from environment.
3. Suffer loss $\mathbf{p}^t \cdot \boldsymbol{\ell}^t$.
4. Set the new weights vector to be

$$w_i^{t+1} = w_i^t \beta^{\ell_i^t}$$

UNIVERSITY OF WATERLOO

What Guarantee?

of rounds

# of experts

total loss of best expert

your total loss

$$\sum_{t=1}^{T} \mathbf{p}^t \cdot \boldsymbol{\ell}^t \le \frac{\ln N - \ln \beta \cdot \min_{i=1,\ldots,N} \sum_{t=1}^{T} \ell_i^t}{1-\beta}$$

goes to 0 as T → inf

choose beta appropriately

$$\frac{1}{T}\sum_{t=1}^{T} \mathbf{p}^t \cdot \boldsymbol{\ell}^t \le \min_{i=1,\ldots,N} \frac{1}{T}\sum_{t=1}^{T} \ell_i^t + \sqrt{\frac{2\ln N}{T} \cdot \min_{i=1,\ldots,N} \frac{1}{T}\sum_{t=1}^{T} \ell_i^t} + \frac{\ln N}{T}$$

Adaptive Boost (Freund & Schapire'97)

Input: sequence of N labeled examples $\langle(x_1, y_1), ..., (x_N, y_N)\rangle$
distribution D over the N examples
weak learning algorithm **WeakLearn**
integer T specifying number of iterations

Initialize the weight vector: $w_i^1 = D(i)$ for $i = 1, ..., N$.

Do for $t = 1, 2, ..., T$

1. Set

$$\mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{i=1}^N w_i^t}$$

2. Call **WeakLearn**, providing it with the distribution $\mathbf{p}^t$; get back a hypothesis h_t: $X \to [0, 1]$.
3. Calculate the error of h_t: $\varepsilon_t = \sum_{i=1}^N p_i^t |h_t(x_i) - y_i|$.
4. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$.
5. Set the new weights vector to be

$$w_i^{t+1} = w_i^t \beta_t^{1 - |h_t(x_i) - y_i|}$$

Output the hypothesis

$$h_f(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^T (\log 1/\beta_t) h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \log 1/\beta_t \\ 0 & \text{otherwise.} \end{cases}$$

Given: $(x_1, y_1), \ldots, (x_m, y_m)$; $x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

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

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(x)\right).$$

(Schapire & Singer, 1999)

Look Closely

Initialize the weight vector: $w_i^1 = D(i)$ for $i = 1, ..., N$.

Do for $t = 1, 2, ..., T$

$$h_f(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t}\right) \cdot h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \log \frac{1}{\beta_t} \\ 0 & \text{otherwise} \end{cases}$$

1. Set

$$\mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{i=1}^{N} w_i^t}$$

bigger ε_t, bigger β_t, smaller coefficient

can optimize

2. Call **WeakLearn**, providing it with the distribution $\mathbf{p}^t$; get back a hypothesis $h_t: X \to [0, 1]$.
3. Calculate the error of h_t: $\varepsilon_t = \sum_{i=1}^{N} p_i^t |h_t(x_i) - y_i|$.
4. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$.
5. Set the new weights vector to be

y = 1 or 0

expected error of h

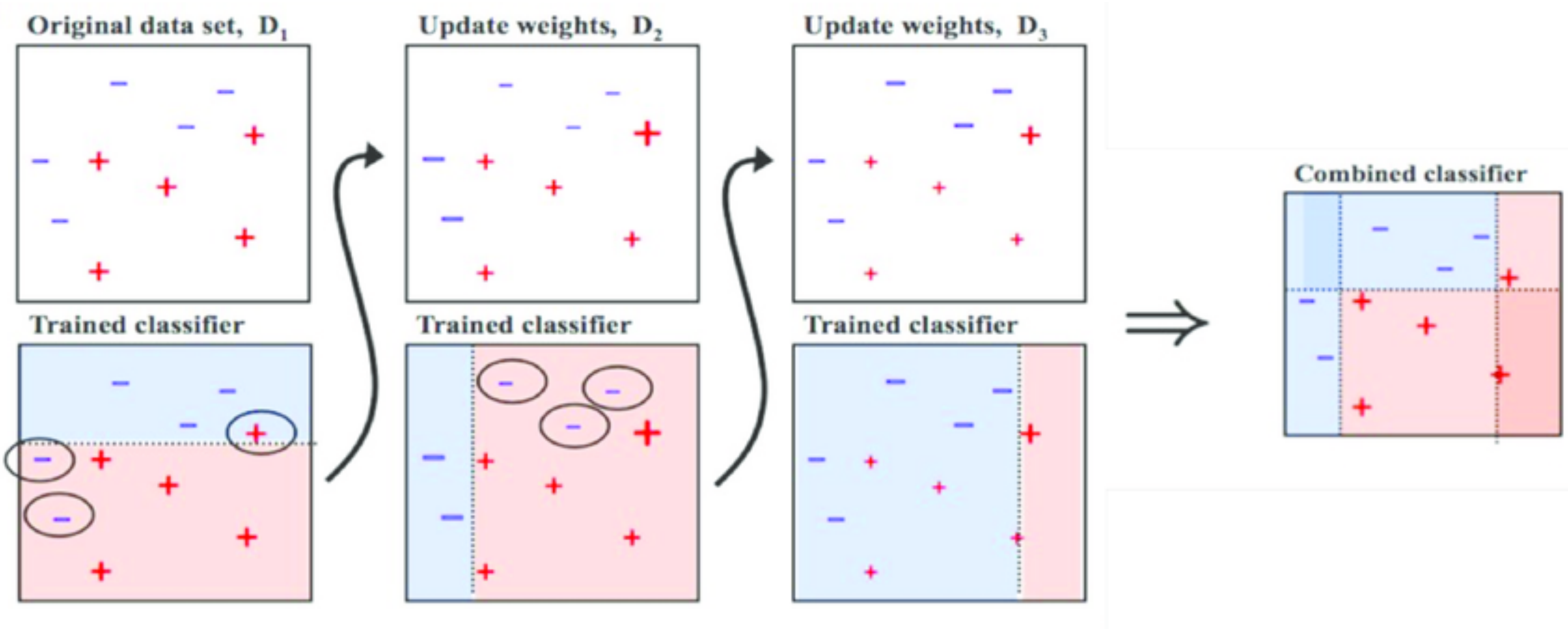
adaptive

when $w_i^t = 0$?

if $\varepsilon_t \leq \frac{1}{2}$ discount $w_i^{t+1} = w_i^t \beta_t^{1 - |h_t(x_i) - y_i|}$

h(x) closer to y, bigger exponent,

Does It Work?



Original data set, D1
Update weights, D2
Update weights, D3
Trained classifier
Trained classifier
Trained classifier
Combined classifier

Exponential decay of … training error

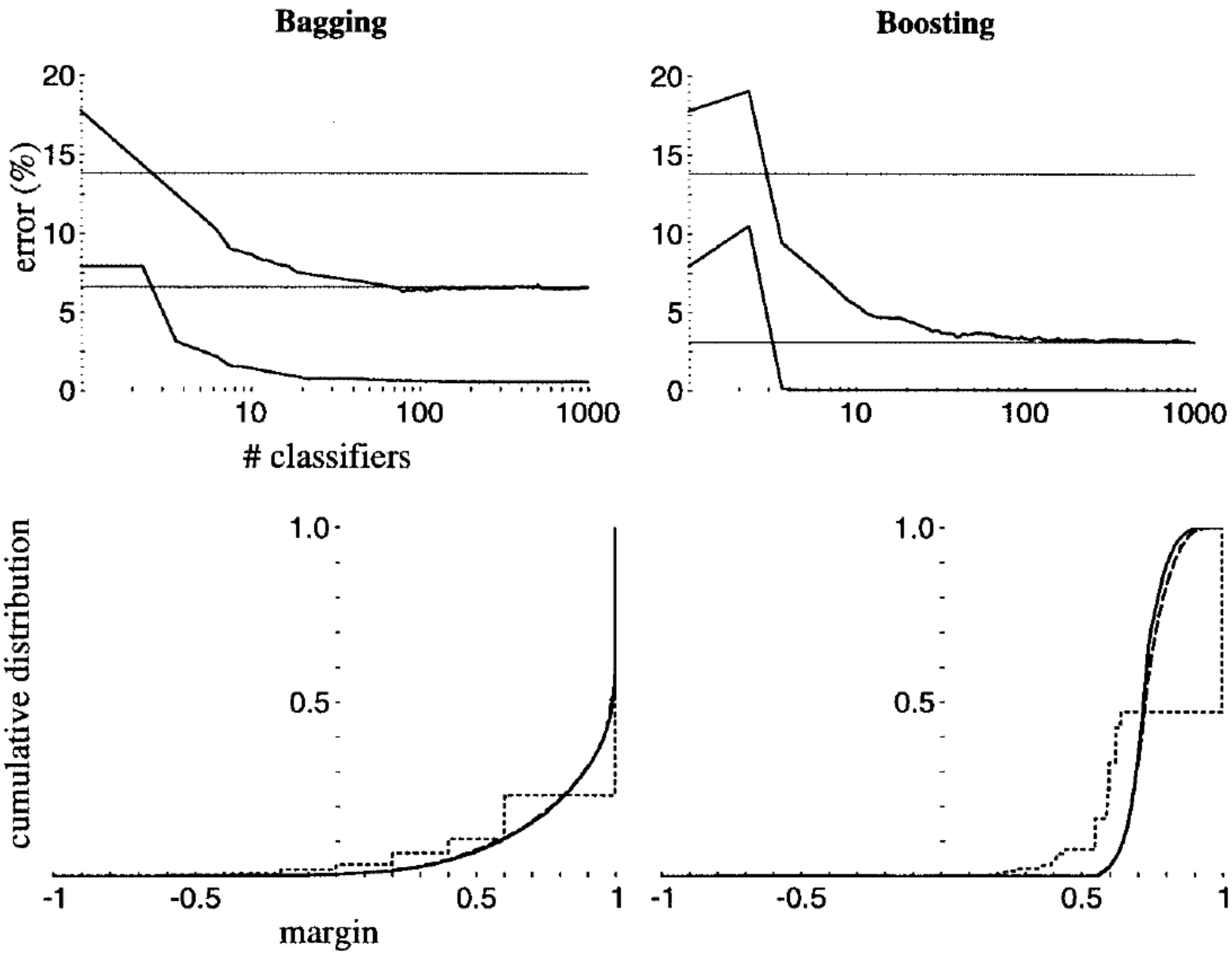
$$\Pr_{i\sim D}[h_f(x_i) \neq y_i] \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}$$

training error! # of weak classifiers slightly smaller than ½

$$\epsilon_t \leq \frac{1}{2} - \gamma \quad \Rightarrow \quad \leq \exp(-2T\gamma^2)$$

- Basically a form of gradient descent to minimize exponential loss: $\sum_i e^{-y_i h(x_i)}$, h in conic hull of h_t's
- Overfitting? Use simple base classifiers (e.g., decision stumps)

Will Adaboost Overfit?

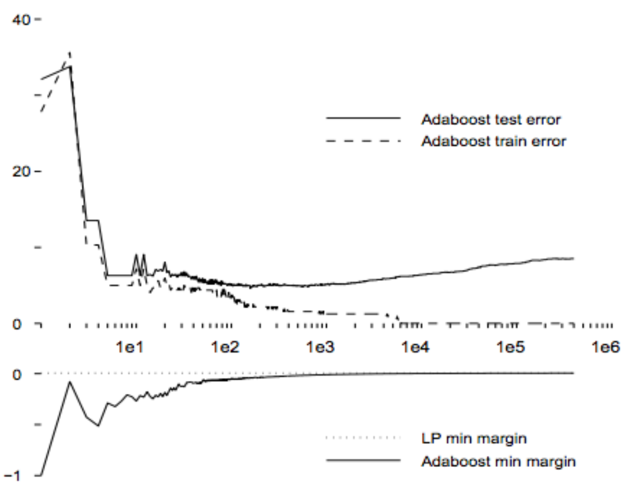


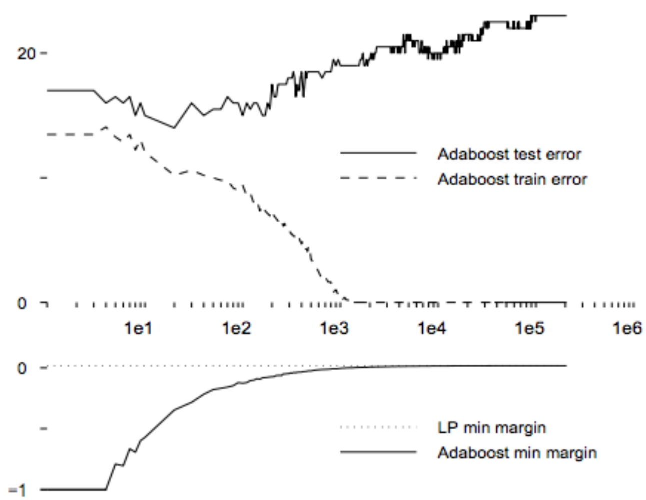
margin: y h(x)

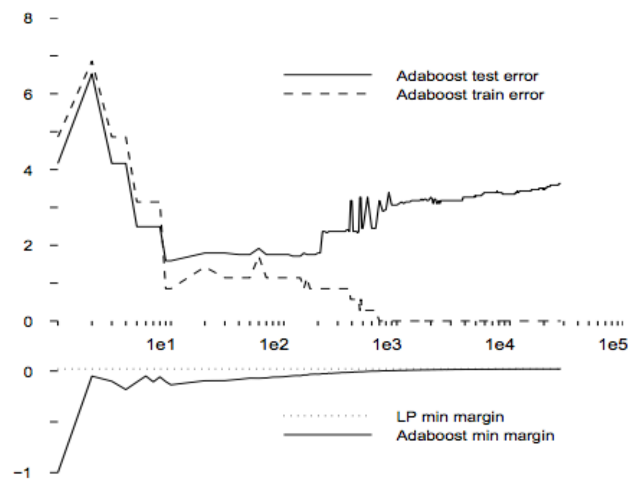
UNIVERSITY OF WATERLOO

Seriously? (Grove & Schuurmans'98; Breiman'99)

	C4.5		Adaboost		LP-Adaboost			DualLPboost		
Data set	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

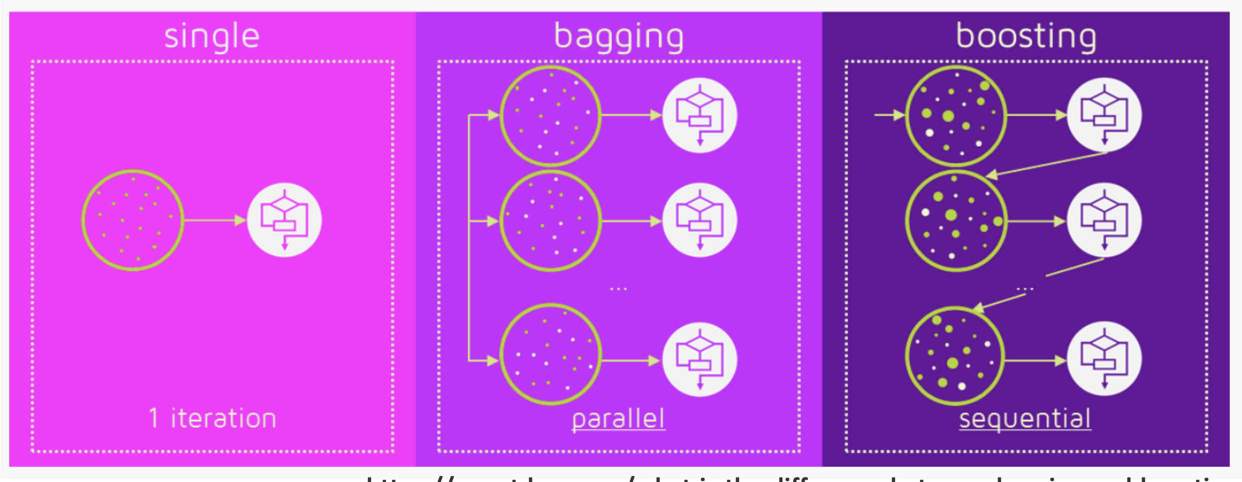






Pros and Cons

- “Straightforward” way to boost performance
- Flexible with any base classifier
- Less interpretable
- Longer training time
  - hard to parallelize (in contrast to bagging)



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

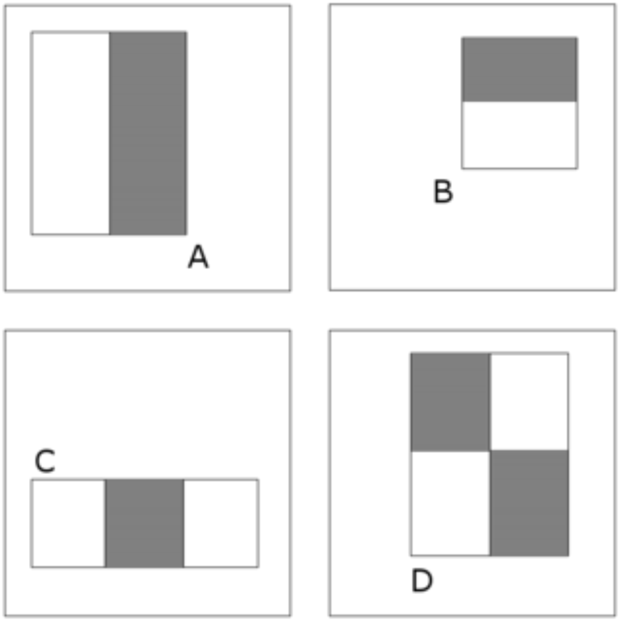
UNIVERSITY OF WATERLOO

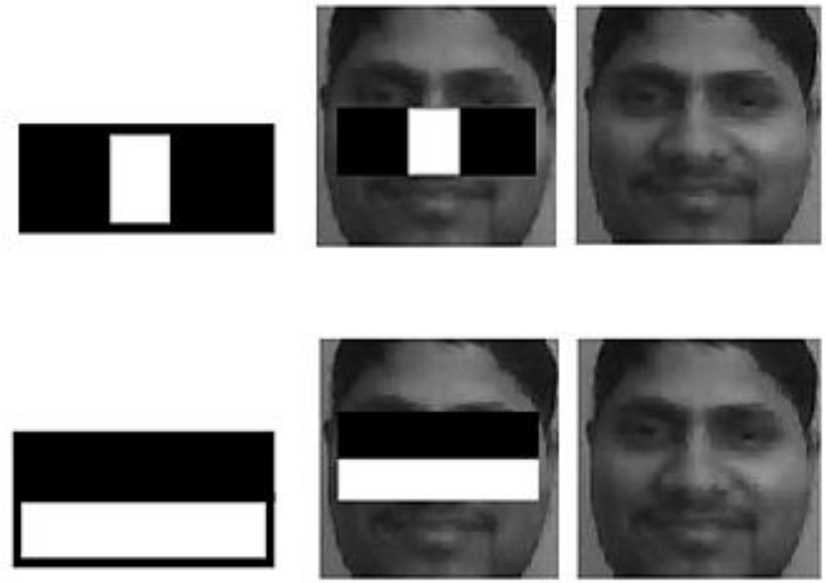
Extensions

- LogitBoost
- GradBoost
- L2Boost
- … you name it

- Multi-class
- Regression
- Ranking

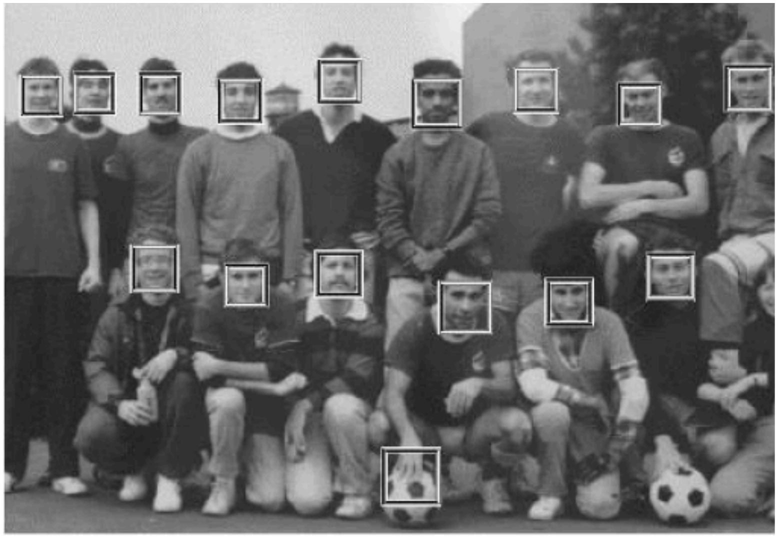
Face Detection (Viola & Jones'01)





- Each detection window results in ~160k features
- Speed is crucial for real-time detection!

Examples



Questions?

UNIVERSITY OF WATERLOO