Boosting Basics

Cynthia Rudin

Duke

- Question of Kearns: Can you turn a “weak” learning algorithm (barely better than random guessing) into a “strong” learning algorithm (whose error rate is arbitrarily close to 0)?

- We could ask the algorithm to create a lot of classifiers and figure out how to combine them… how to do that?

"The strength of weak learnability" (Schapire, 1990) answers this question

Theorem 1: A concept class C is weakly learnable if and only if it is strongly learnable

The proof was constructive (an algorithm!) but it wasn't practical.

Schapire and Freund's (1996) answer:

- Reweight the data in many specific ways
- Use the weak learning algorithm to create a weak classifier for each (reweighted) dataset
- Compute a weighted average of the weak classifiers.

Outline of a generic boosting algorithm

for $t = 1 \ldots T$

construct $\mathbf{d}_t$, where $\mathbf{d}_t$ is a discrete probability distribution over indices $\{1 \ldots n\}$.

WLA

run A on $\mathbf{d}_t$, producing $h_{(t)} : \mathcal{X} \to \{-1, 1\}$.

calculate

$$\begin{aligned} \epsilon_t &= \mathrm{error}_{\mathbf{d}_t}(h_{(t)}) = Pr_{i \sim \mathbf{d}_t}[h_{(t)}(x_i) \neq y_i] \\ &=: \frac{1}{2} - \gamma_t, \end{aligned}$$

where by the weak learning assumption, $\gamma_t > \gamma_{WLA}$.

end

output H

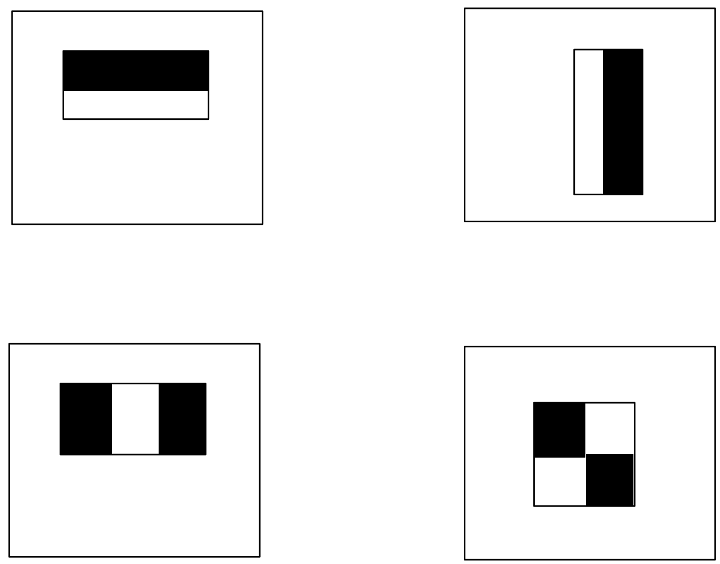
Weak learning assumption.

H is a combination of the $h_{(t)}$'s.

Weak classifiers used by Viola and Jones

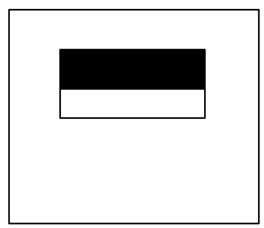
Weak classifiers used by Viola and Jones

- Subtract the white areas from the black ones



Weak classifiers used by Viola and Jones

- Subtract the white areas from the black ones



Weak classifiers used by Viola and Jones

- Subtract the white areas from the black ones

Weak classifiers used by Viola and Jones

- Subtract the white areas from the black ones

Doesn't detect anything

Black and white areas are very similar

Weak classifiers used by Viola and Jones

- Subtract the white areas from the black ones

Doesn't detect anything

Black and white areas are very similar

Weak classifiers used by Viola and Jones

- Subtract the white areas from the black ones

Weak classifiers used by Viola and Jones

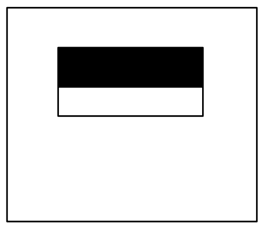
- Subtract the white areas from the black ones

Now it detects!

Weak classifiers used by Viola and Jones

- Subtract the white areas from the black ones

Weak classifiers

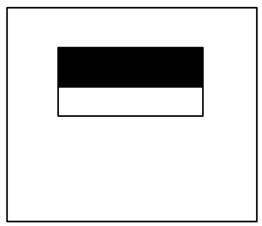


Detects eyes!

Weak classifiers used by Viola and Jones

- Subtract the white areas from the black ones

Weak classifiers



Detects eyes!

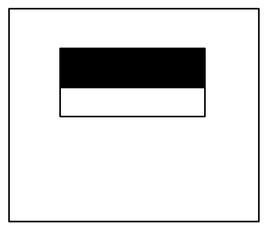
Credit: telegraph.co.uk

Credit: chinadaily.com.cn

Weak classifiers used by Viola and Jones

- Subtract the white areas from the black ones

Weak classifiers



Detects eyes!

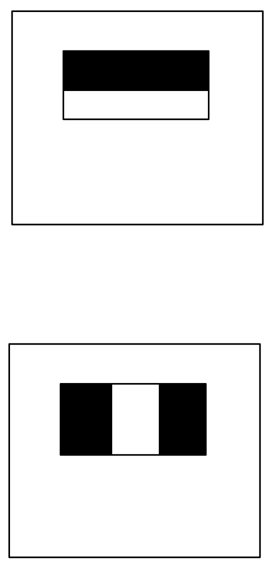
Credit: telegraph.co.uk

Credit: chinadaily.com.cn

Weak classifiers used by Viola and Jones

- Subtract the white areas from the black ones

Weak classifiers



Detects eyes!

Weak classifiers used by Viola and Jones

- Used hundreds of thousands of these weak classifiers at all different scales

Weak classifiers used by Viola and Jones

- Used hundreds of thousands of these weak classifiers at all different scales

Weak classifiers used by Viola and Jones

- Used hundreds of thousands of these weak classifiers at all different scales

AdaBoost Pseudocode

Assign observation i the weight of d_{1i}=$1/n$ (equal weights).

For $t = 1{:}T$

Train weak learning algorithm using data weighted by d_{ti}. This produces weak classifier h_t.

Choose coefficient α_t.

Update weights:

$y_i h_t(x_i)$ is 1 if correct, -1 if incorrect

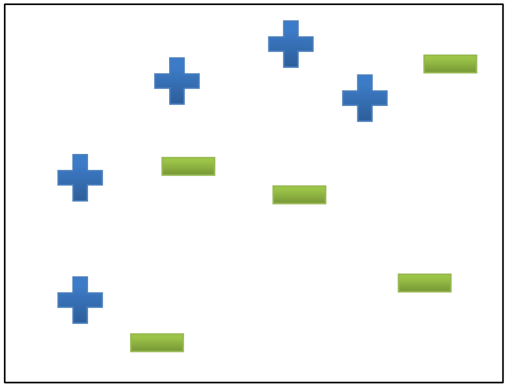
$$d_{t+1,i} = \frac{d_{t,i} \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Z_t is a normalization factor.

End

Output the final classifier: $H(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right)$

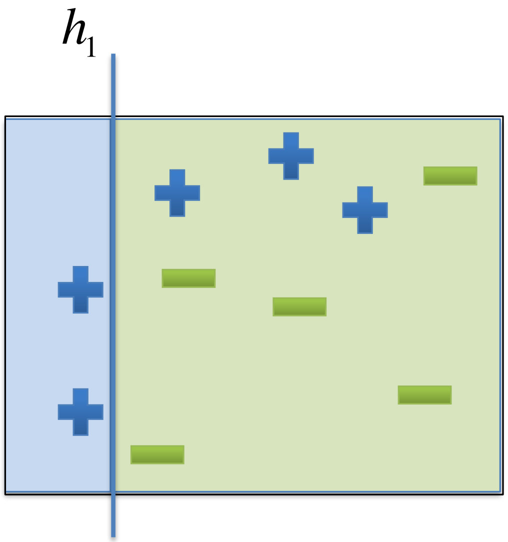
Boosting Example



All points start with equal weights.

(Credit: Example adapted from Freund and Schapire)

Boosting Example

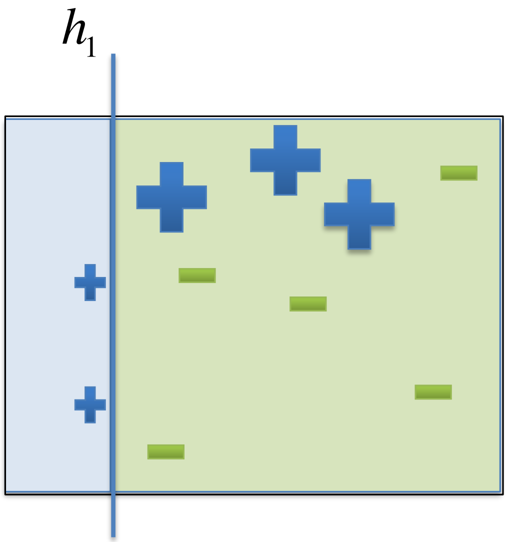


Run the weak learning algorithm to get a weak classifier.

Choose coefficient $\alpha_1 = .41$

(Credit: Example adapted from Freund and Schapire)

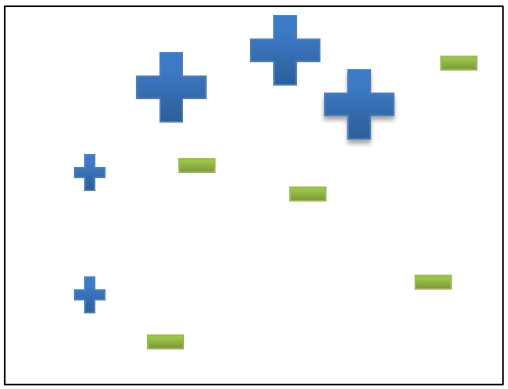
Boosting Example



Increase the weights on the misclassified points, decrease the weights on the correctly classified points.

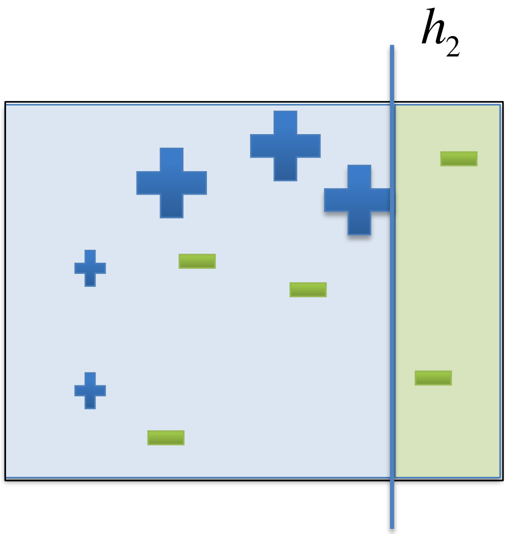
(Credit: Example adapted from Freund and Schapire)

Boosting Example



(Credit: Example adapted from Freund and Schapire)

Boosting Example

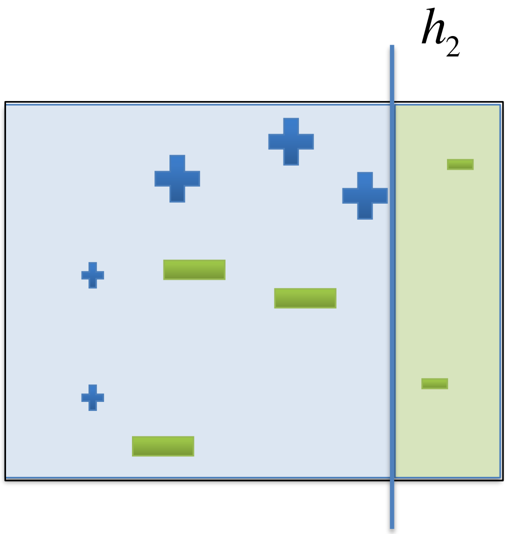


Run the weak learning algorithm to get a weak classifier for the weighted data.

Choose coefficient $\alpha_2 = .66$

(Credit: Example adapted from Freund and Schapire)

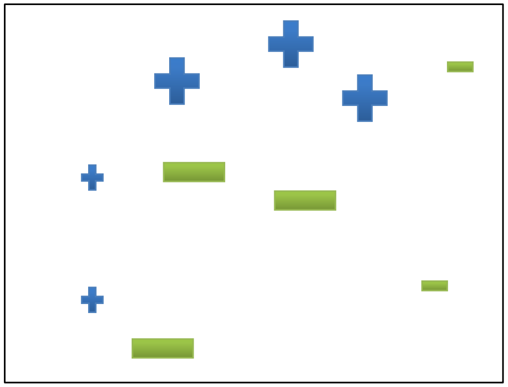
Boosting Example



Increase the weights on the misclassified points, decrease the weights on the correctly classified points.

(Credit: Example adapted from Freund and Schapire)

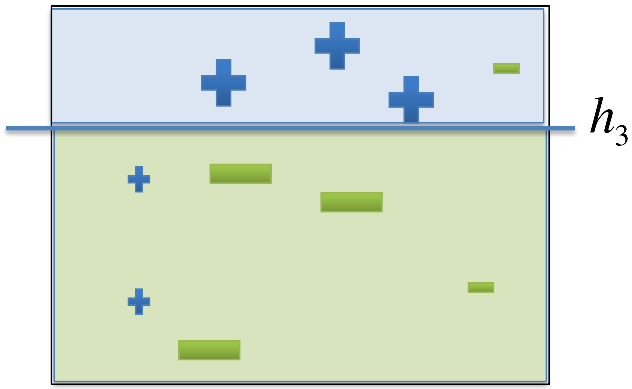
Boosting Example



Increase the weights on the misclassified points, decrease the weights on the correctly classified points.

(Credit: Example adapted from Freund and Schapire)

Boosting Example

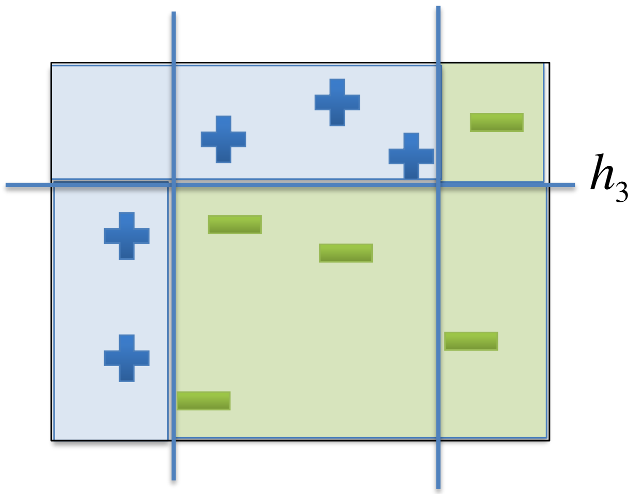


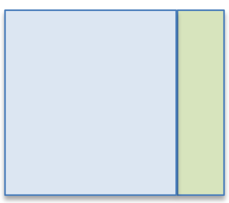
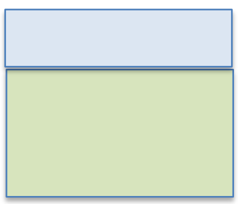
Increase the weights on the misclassified points, decrease the weights on the correctly classified points.

Choose coefficient $\alpha_3 = .93$

(Credit: Example adapted from Freund and Schapire)

Boosting Example



H=sign(.42  + .66  + .93 )

(Credit: Example adapted from Freund and Schapire)

AdaBoost Pseudocode

Assign observation *i* the weight of $d_{1i}=1/n$ (equal weights).

For $t=1:T$

Train weak learning algorithm using data weighted by d_{ti}. This produces weak classifier h_t.

Choose coefficient α_t.

Update weights:

$$d_{t+1,i} = \frac{d_{t,i}}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_{(t)}(x_i) \text{ (smaller weights for easy examples)} \\ e^{\alpha_t} & \text{if } y_i \neq h_{(t)}(x_i) \text{ (larger weights for hard examples)} \end{cases}$$

End

Z_t is a normalization factor.

Output the final classifier: $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

AdaBoost Pseudocode

Assign observation i the weight of $d_{1i}=1/n$ (equal weights).

For $t = 1{:}T$

Train weak learning algorithm using data weighted by d_{ti}. This produces weak classifier h_t.

Choose coefficient α_t.

Update weights:

$y_i h_t(x_i)$ is 1 if correct, -1 if incorrect

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

Z_t is a normalization factor.

End

Output the final classifier: $H(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

AdaBoost Pseudocode

Assign observation i the weight of $d_{1i}=1/n$ (equal weights).

For $t=1:T$

Train weak learning algorithm using data weighted by d_{ti}. This produces weak classifier h_t.

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\text{Error}_t}{\text{Error}_t}\right)$$

Choose coefficient α_t.

$y_i h_t(x_i)$ is 1 if correct, -1 if incorrect

Update weights:

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

Z_t is a normalization factor.

End

Output the final classifier: $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Historical Notes

Schapire/Freund came up with AdaBoost in 1996. Immediately after, 5 groups figured out that it was coordinate descent on the exponential loss (Breiman, 1997; Friedman et al., 2000; Raetsch et al., 2001; Duffy and Helmbold, 1999; Mason et al., 2000).

Coming up soon: derivation of AdaBoost as coordinate descent on exp loss.

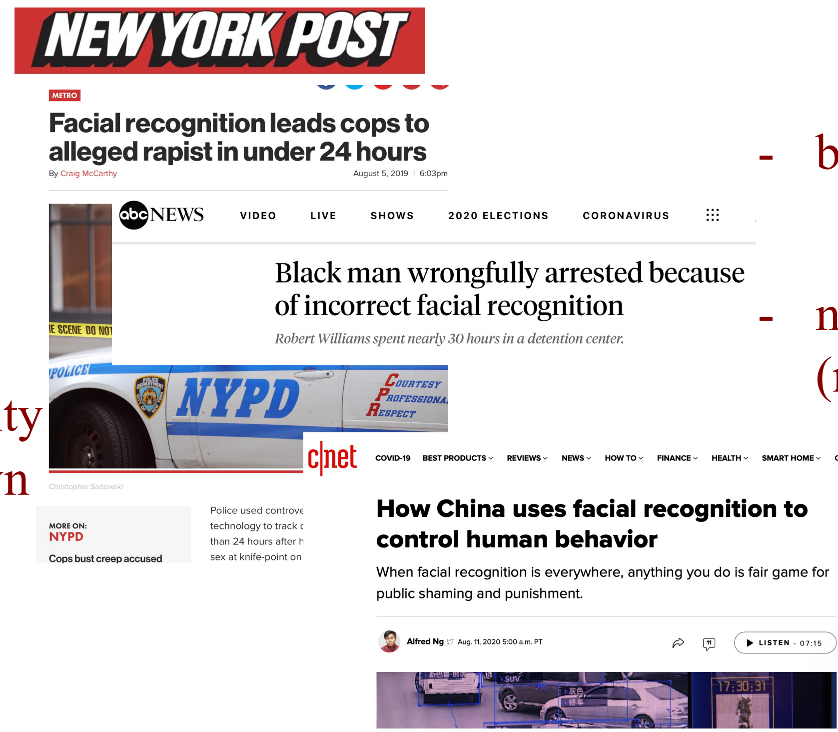
A short ethical interlude

- Face recognition
  - Possibly incredibly helpful, also incredibly dangerous.

Why is it useful?

Security

- school shootings
- kidnappings
- violent crime
- terrorist attacks
- public event security
- unlocking your own phone


NEW YORK POST

METRO

Facial recognition leads cops to alleged rapist in under 24 hours

By Craig McCarthy August 5, 2019 | 6:03pm

abc NEWS VIDEO LIVE SHOWS 2020 ELECTIONS CORONAVIRUS

Black man wrongfully arrested because of incorrect facial recognition

Robert Williams spent nearly 30 hours in a detention center.

MORE ON: NYPD

Cops bust creep accused

Police used controve technology to track than 24 hours after h sex at knife-point on

c|net COVID-19 BEST PRODUCTS REVIEWS NEWS HOW TO FINANCE HEALTH SMART HOME

How China uses facial recognition to control human behavior

When facial recognition is everywhere, anything you do is fair game for public shaming and punishment.

Alfred Ng Aug. 11, 2020 5:00 a.m. PT LISTEN · 07:15

Why is it harmful?

Data

- biometrics & privacy laws

Accuracy

- not always accurate, varies by race (note: this is getting better)

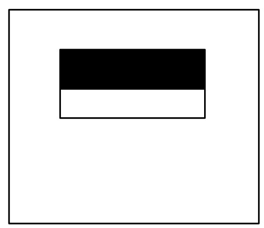
Bullying

- public shaming

Weak classifiers used by Viola and Jones

- Subtract the white areas from the black ones

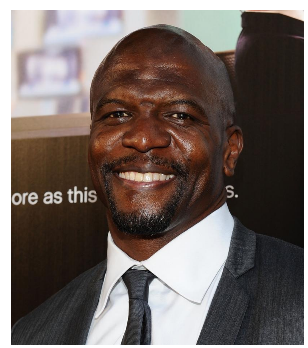
Weak classifiers



Detects eyes!

Credit: telegraph.co.uk

Credit: chinadaily.com.cn



Credit: itisweird.com

Question to think about

- How can we have it both ways?
  - Can we invent facial recognition technology for safe use at schools?
- What protections could you envision for facial data and other biometrics?
- How could biometric data with these protections be used for security purposes?

Coordinate Descent

$\min_{\mathbf{b}}$ R($\mathbf{b}$) by moving along one coordinate at a time

$$\min_{b_1,b_2,b_3,\ldots,b_j,\ldots b_p} R([b_1,b_2,b_3,\ldots,b_j,\ldots b_p])$$

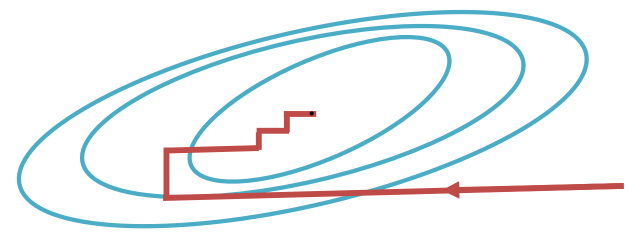
Adjust me

Then adjust me

Repeat until no more adjustments possible.

Coordinate Descent

$\min_{\mathbf{b}} R(\mathbf{b})$ by moving along one coordinate at a time



Until converged

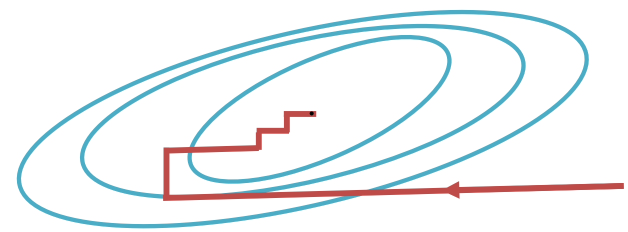
- Choose j
- Optimize R along direction j:

$$\min_\alpha R([b_1,b_2,b_3,\ldots,b_j+\alpha,\ldots b_p]) \;\; = \min_\alpha R(\mathbf{[b} + \alpha\ \mathbf{e}_j\mathbf{])}$$

Adjust me

Coordinate Descent

$\min_{\mathbf{b}} R(\mathbf{b})$ by moving along one coordinate at a time



Why would you use coordinate descent instead of gradient descent?

1) The gradient is impossible to calculate.

   E.g., boosted decision trees – optimizes over the whole space of decision trees (every possible decision tree is a "coordinate"!)

2) The feasible region is constrained

   E.g., SVM

3) You want to control the optimization

The two views of AdaBoost

Cynthia Rudin

Duke

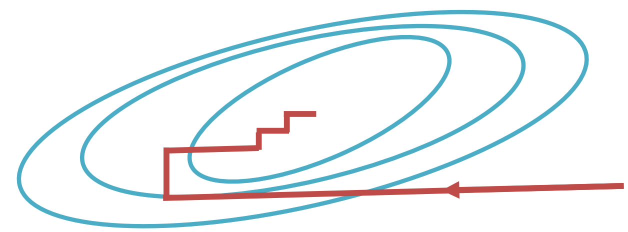
There are two ways to derive AdaBoost.

1. AdaBoost reweights the data and calls the weak learning algorithm to create a good weak classifier for the weighted data.
2. AdaBoost is coordinate descent on the exponential loss.

Reconciling these two views requires some understanding.

Coordinate Descent

$\min_{\mathbf{b}} R(\mathbf{b})$ by moving along one coordinate at a time



Until converged

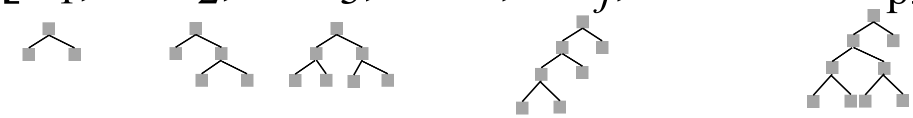
- Choose j
- Optimize R along direction j:

$$\min_{\alpha} R([b_1, b_2, b_3, \ldots, b_j + \alpha, \ldots b_p]) \;\; = \min_{\alpha} R(\mathbf{[b} + \alpha\ \mathbf{e}_j\mathbf{])}$$

Coordinate Descent

coefficient on tree

$$\min R([b_1, \quad b_2, \quad b_3, \quad \ldots, \quad b_j, \quad \ldots \quad b_p])$$



all trees

weak learning algorithm picks a tree

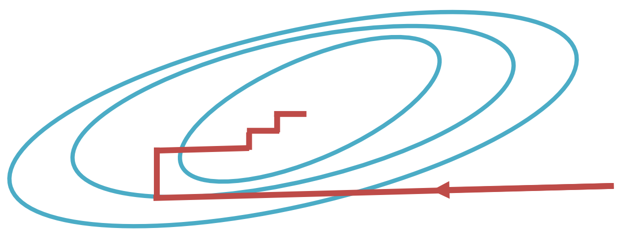
Weak classifier j = Coordinate j

Run weak learning algorithm = Choose weak classifier j = Choose coordinate j

Moving along direction j by α = adding α to the coefficient of weak classifier j

Coordinate Descent

$[b_1, \quad b_2, \quad b_3, \quad \ldots, \quad b_j + \alpha, \ldots \quad b_p]$



$$\min_\alpha R([b_1, b_2, b_3, \ldots, b_j + \alpha, \ldots b_p])$$

Weak classifier j = Coordinate j

Run weak learning algorithm = Choose weak classifier j = Choose coordinate j

Moving along direction j by α = adding α to the coefficient of weak classifier j

Coordinate Descent

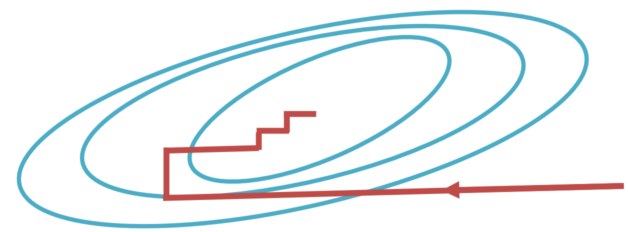
$[b_1, \quad b_2, \quad b_3, \quad \ldots, \quad b_j + \alpha, \ldots \quad b_p]$

coeff of h_1

coeff of h_2

coeff of h_3

coeff of h_j



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

Run weak learning algorithm = Choose weak classifier j = Choose coordinate j

Moving along direction j by α = adding α to the coefficient of weak classifier j

Coordinate Descent

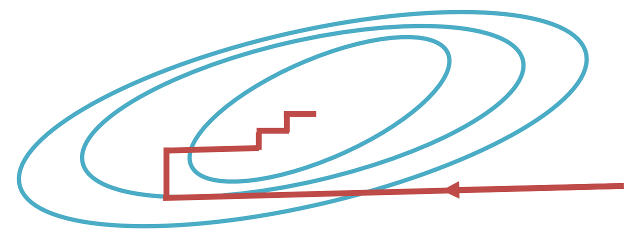
$$[b_1, \quad b_2, \quad b_3, \quad \ldots, \quad b_j+\alpha, \ldots \quad b_p]$$

coeff of h_1

coeff of h_2

coeff of h_3

coeff of h_j

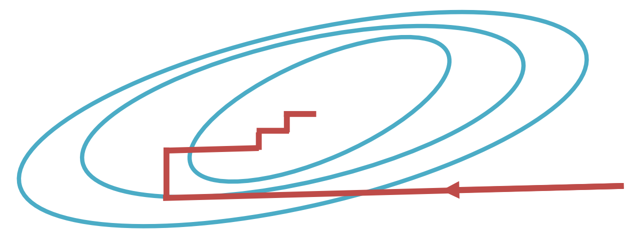


$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Update to coeff of h_t at iteration t

Coordinate Descent

$\min_{\mathbf{b}}$ R($\mathbf{b}$) by moving along one coordinate at a time



Until converged

- Choose j (run weak learning algorithm)
- Optimize R along direction j:

$$\min_{\alpha} R([b_1,b_2,b_3,\ldots,b_j+\alpha,\ldots b_p]) = \min_{\alpha} R(\mathbf{[b} + \alpha\ \mathbf{e}_j\mathbf{])}$$

(choose coefficient α)

Note: the weight vector d doesn't really have a meaning in the coordinate descent view.

AdaBoost Pseudocode

Assign observation *i* the weight of $d_{1i}=1/n$ (equal weights).

For t=1:T

Train weak learning algorithm using data weighted by d_{ti}. This produces weak classifier h_t.

Choose coefficient α_t.

Update weights:

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

Z_t is a normalization factor.

End

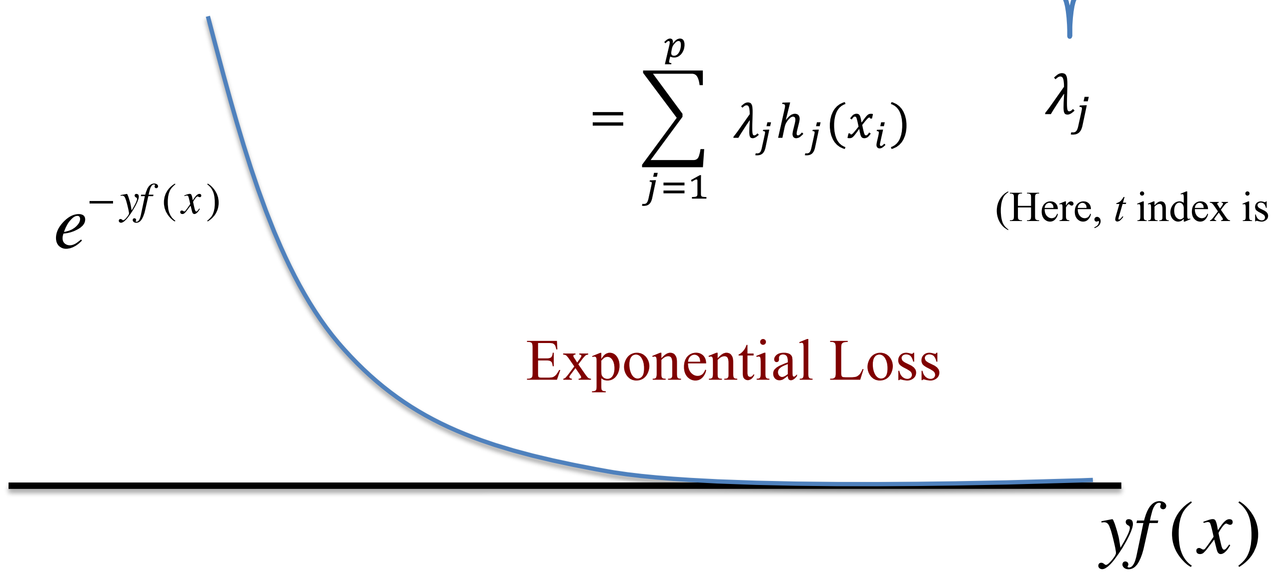
Output the final classifier: $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

AdaBoost's objective: $F(f) = \sum_{i=1}^{n} e^{-y_i f(x_i)}$

where: $$f(x_i) = \sum_{t=1}^{T} \alpha_t h_t(x_i) = \sum_{j=1}^{p} \underbrace{\sum_{t=1}^{T} 1_{[h_j \text{ was chosen at iteration } t]} \alpha_t}_{\lambda_j} h_j(x_i)$$

$$= \sum_{j=1}^{p} \lambda_j h_j(x_i)$$

(Here, t index is different from j index.)



Notes

- AdaBoost was derived using the weak learning perspective:
  - Reweight data at every iteration, choose the best weak classifier for the weighted data.
- It was only after AdaBoost was published did researchers notice that AdaBoost was coordinate descent on the exponential loss.

Notes on AdaBoost

Cynthia Rudin

Duke

AdaBoost can be used in 2 ways

- Where the weak classifiers are truly weak, e.g., $h_j(\mathbf{x}_i) = x_{ij}$
- Where the weak classifiers are strong and come from another learning algorithm, like a decision tree method (C4.5 or CART). Each possible tree it produces is an h_j.

Adding an Intercept Term

- An easy way to add an intercept to AdaBoost is to allow the weak learning algorithm to use a weak classifier that always predicts 1.

$$H(x) = \sum_{j=1}^{p} \lambda_j h_j(x)$$

- If the weak classifiers are just the features, append a new feature that is always 1.

$$\text{New } X = \begin{bmatrix} X & \begin{matrix} 1 \\ 1 \\ 1 \end{matrix} \end{bmatrix}$$

The WLA

The weak learning assumption doesn't usually hold, but AdaBoost works anyway (think of coordinate descent view).

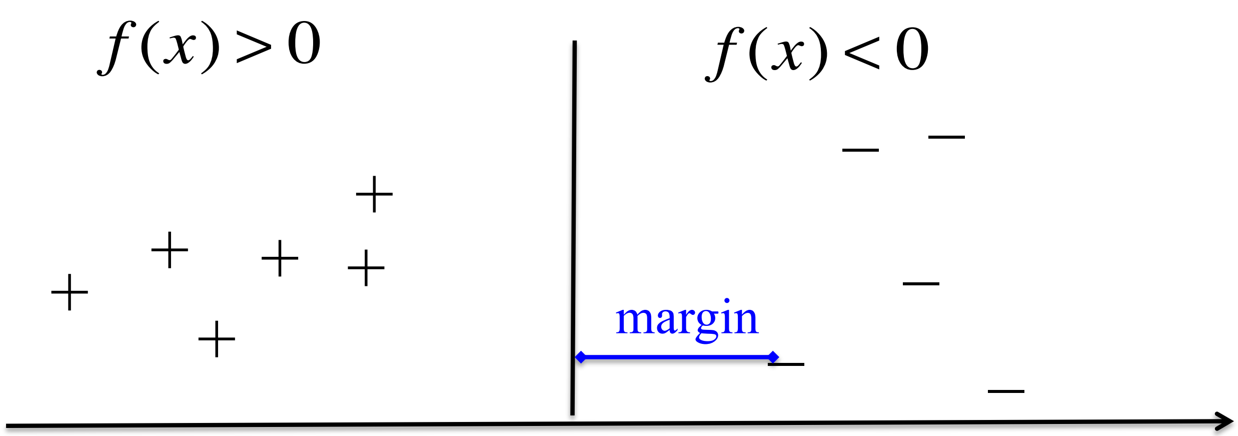
$$\begin{aligned} \epsilon_t &= \text{error}_{\mathbf{d}_t}(h_{(t)}) \\ &=: \frac{1}{2} - \gamma_t, \end{aligned}$$

$\gamma_t > \gamma_{WLA}$ for all t.

AdaBoost has no regularization

- Yet AdaBoost tends not to overfit. Why?

Does AdaBoost maximize the margin, like SVM?

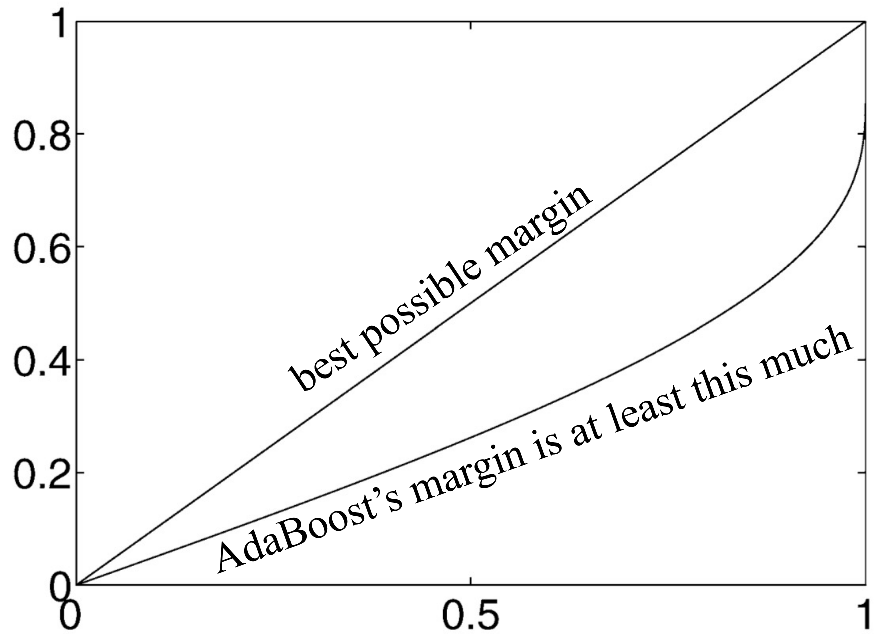


$$\min_i \frac{(\mathbf{M}\boldsymbol{\lambda})_i}{\|\boldsymbol{\lambda}\|_1}$$

AdaBoost has no regularization

- Yet AdaBoost tends not to overfit. Why?

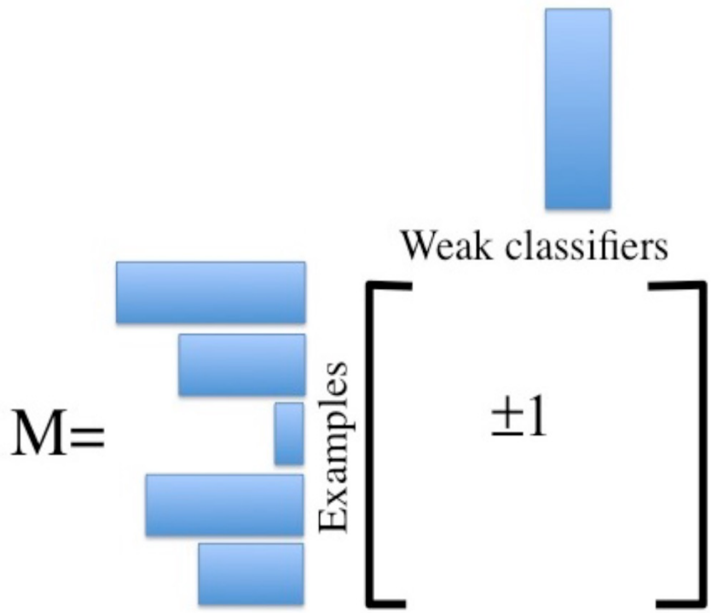
“Theorem”: AdaBoost achieves large margins, but not maximal margins.



$$\Upsilon(r) := \frac{-\ln(1-r^2)}{\ln((1+r)/(1-r))}.$$

margin (normalized)

AdaBoost has an interpretation as a 2-player repeated game.



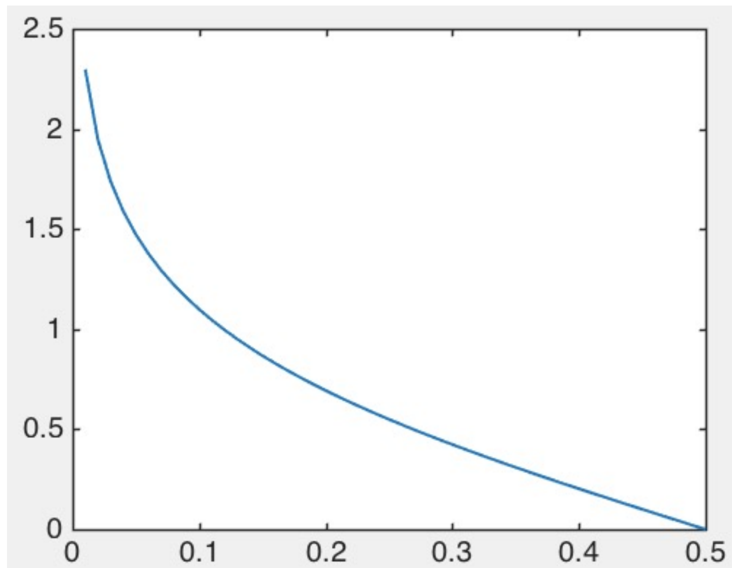
weak learning algorithm chooses $j_t \equiv$ column player chooses a pure strategy

$\mathbf{d}_t \equiv$ mixed strategy for row player

AdaBoost's Coefficients Update

$$\text{Error}_t = \sum_{i:h_t(x_i)\neq y_i} d_t = \text{sum of weights of misclassified points}$$

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\text{Error}_t}{\text{Error}_t}\right)$$



Error_t