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 (rewighted) dataset
- Compute a weighted average of the weak classifiers.
Outline of a generic boosting algorithm

for $t = 1...T$
construct $d_t$, where $d_t$ is a discrete probability distribution
over indices $\{1...n\}$.
run *A* on $d_t$, producing $h_t : \mathcal{X} \rightarrow \{-1, 1\}$.
calculate

$$
\epsilon_t = \text{error}_{d_t}(h_t) = \Pr_{i \sim d_t}[h_t(x_i) \neq y_i]
$$

$$
=: \frac{1}{2} - \gamma_t,
$$

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

$H$ 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

Black and white areas are very similar

Doesn’t detect anything
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

Detection of eyes!
Weak classifiers used by Viola and Jones

- Subtract the white areas from the black ones

Credit: telegraph.co.uk
Credit: chinadaily.com.cn

Dets 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!
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 $\alpha_t$.

Update weights:

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

$Z_t$ is a normalization factor.

End

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

All points start with equal weights.

(Credit: Example adapted from Freund and Schapire)
Run the weak learning algorithm to get a weak classifier.

Choose coefficient $\alpha_1 = .41$
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)
Run the weak learning algorithm to get a weak classifier for the weighted data.

Choose coefficient $\alpha_2 = .66$
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

\[ H = \text{sign}(0.42 + 0.66 + 0.93) \]

(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 $\alpha_t$.

Update weights:

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

$Z_t$ 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 $\alpha_t$.

Update weights:

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

$Z_t$ 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 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 $\alpha_t$.

Update weights:

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

$Z_t$ is a normalization factor.

End

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

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

Dects 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

\[ \min_b R(b) \text{ by moving along one coordinate at a time} \]

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

Adjust me

Then adjust me

Repeat until no more adjustments possible.
Coordinate Descent

\[ \min_b R(b) \text{ by moving along one coordinate at a time} \]

Until converged

- Choose \( j \)
- Optimize \( R \) along direction \( j \):

\[
\min_\alpha R([b_1, b_2, b_3, \ldots, b_j + \alpha, \ldots, b_p]) = \min_\alpha R([b + \alpha e_j])
\]
Coordinate Descent

\[ \min_b R(b) \text{ by moving along one coordinate at a time} \]

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

\[ \min_{b} R(b) \text{ by moving along one coordinate at a time} \]

Until converged

- Choose \(j\)
- Optimize \(R\) along direction \(j\):

\[ \min_{\alpha} R([b_1, b_2, b_3, \ldots, b_j + \alpha, \ldots b_p]) = \min_{\alpha} R([b + \alpha e_j]) \]
Coordinate Descent

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

All trees

Weak learning algorithm picks a tree

Weak classifier \( j = \text{Coordinate } j \)

Run weak learning algorithm = Choose weak classifier \( j = \text{Choose coordinate } j \)

Moving along direction \( j \) by \( \alpha \) = adding \( \alpha \) to the coefficient of weak classifier \( j \)
Coordinate Descent

\[ [b_1, b_2, b_3, ..., b_j + \alpha, ..., b_p] \]

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

Weak classifier \( j = \text{Coordinate } j \)

Run weak learning algorithm = Choose weak classifier \( j = \text{Choose coordinate } j \)

Moving along direction \( j \) by \( \alpha = \text{adding } \alpha \) to the coefficient of weak classifier \( j \)
Coordinate Descent

\[
[b_1, \ b_2, \ b_3, \ ... \ b_j + \alpha, \ ... \ b_p]
\]

\[H(x) = \text{sign} \left( \sum_{i=1}^{T} \alpha_i h_i(x) \right)\]

Run weak learning algorithm = Choose weak classifier \( j \) = Choose coordinate \( j \)

Moving along direction \( j \) by \( \alpha \) = adding \( \alpha \) to the coefficient of weak classifier \( j \)
Coordinate Descent

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

coeff of \(h_1\) \quad coeff of \(h_2\) \quad coeff of \(h_3\) \quad 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

\[ \min_b R(b) \] by moving along one coordinate at a time

Until converged

- Choose \( j \) \hspace{1cm} \text{(run weak learning algorithm)}
- Optimize \( R \) along direction \( j \):

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

(choose coefficient \( \alpha \))

Note: the weight vector \( d \) doesn’t really have a meaning in the coordinate descent view.
AdaBoost Pseudocode

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

For \( t = 1:T \)

1. Train weak learning algorithm using data weighted by \( d_{t,i} \). This produces weak classifier \( h_t \).
2. Choose coefficient \( \alpha_t \).
3. 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: 

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

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

Exponential Loss

$$e^{-y f(x)}$$

$$y f(x)$$
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., $h_j(x_i) = x_{ij}$

- 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.

\[
\text{New } X = \begin{bmatrix}
X & 1 \\
1 & 1
\end{bmatrix}
\]
The WLA

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

\[ \epsilon_t = \text{error}_d(h_{(t)}) \]
\[ =: \frac{1}{2} - \gamma_t, \]
\[ \gamma_t > \gamma_{WLA} \text{ for all } t. \]
AdaBoost has no regularization

- Yet AdaBoost tends not to overfit. Why?

Does AdaBoost maximize the margin, like SVM?

\[ f(x) > 0 \]

\[ f(x) < 0 \]

\[ \min_i \frac{(M\lambda)_i}{\|\lambda\|_1} \]
AdaBoost has no regularization

- Yet AdaBoost tends not to overfit. Why?

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

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

Graph:
- Best possible margin
- AdaBoost’s margin is at least this much
- 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 $d_t \equiv$ mixed strategy for row player
AdaBoost’s Coefficients Update

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)