Boosting

Machine Learning: Jordan Boyd-Graber
University of Maryland
SLIDES ADAPTED FROM ROB SCHAPIRE
Motivating Example

Goal

Automatically categorize type of call requested by phone customer (Collect, CallingCard, PersonToPerson, etc.)

- yes I’d like to place a collect call long distance please (Collect)
- operator I need to make a call but I need to bill it to my office (ThirdNumber)
- yes I’d like to place a call on my master card please (CallingCard)
- I just called a number in sioux city and I musta rang the wrong number because I got the wrong party and I would like to have that taken off of my bill (BillingCredit)
Boosting Approach

- devise computer program for deriving rough rules of thumb
- apply procedure to subset of examples
- obtain rule of thumb
- apply to second subset of examples
- obtain second rule of thumb
- repeat $T$ times
Details

- How to **choose** examples
- How to **combine** rules of thumb
Details

- How to \textit{choose} examples
  concentrate on \textit{hardest} examples (those most often misclassified by previous rules of thumb)
- How to \textit{combine} rules of thumb
Details

- How to **choose** examples
  concentrate on **hardest** examples (those most often misclassified by previous rules of thumb)

- How to **combine** rules of thumb
  take (weighted) majority vote of rules of thumb
## Definition

Boosting is a general method of converting rough rules of thumb into highly accurate prediction rules. It relies on the following:

- Assume given a weak learning algorithm that can consistently find classifiers (rules of thumb) at least slightly better than random, say, accuracy ≥ 55% (in two-class setting).
- Given sufficient data, a boosting algorithm can provably construct a single classifier with very high accuracy, say, 99%.
Formal Description

- Training set \((x_1, y_1) \ldots (x_m, y_m)\)
- \(y_i \in \{-1, +1\}\) is the label of instance \(x_i\)
Formal Description

- Training set \((x_1, y_1) \ldots (x_m, y_m)\)
- \(y_i \in \{-1, +1\}\) is the label of instance \(x_i\)
- For \(t = 1, \ldots T\):
  - Construct distribution \(D_t\) on \(\{1, \ldots, m\}\)
  - Find weak classifier
    \[
    h_t : \mathcal{X} \rightarrow \{-1, +1\}
    \]
    with small error \(\varepsilon_t\) on \(D_t\):
    \[
    \varepsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i]
    \]
Formal Description

- Training set \((x_1, y_1) \ldots (x_m, y_m)\)
- \(y_i \in \{-1, +1\}\) is the label of instance \(x_i\)
- For \(t = 1, \ldots T\):
  - Construct distribution \(D_t\) on \([1, \ldots, m]\)
  - Find weak classifier
    \[
    h_t : \mathcal{X} \mapsto \{-1, +1\}
    \]
    with small error \(\epsilon_t\) on \(D_t\):
    \[
    \epsilon_t = \Pr_{i \sim D_t} \left[ h_t(x_i) \neq y_i \right]
    \]
  - Output final classifier \(H_{\text{final}}\)
AdaBoost (Schapire and Freund)

- Data distribution $D_t$
AdaBoost (Schapire and Freund)

- Data distribution $D_t$
  - $D_1(i) = \frac{1}{m}$
  - Given $D_t$ and $h_t$:

$$D_{t+1}(i) \propto D_t(i) \cdot \exp \left\{ -\alpha_t y_i h_t(x_i) \right\}$$

where $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right) > 0$
AdaBoost (Schapire and Freund)

- Data distribution $D_t$
  - $D_1(i) = \frac{1}{m}$
  - Given $D_t$ and $h_t$:
    
    $$D_{t+1}(i) \propto D_t(i) \cdot \exp\left\{ -\alpha_t y_i h_t(x_i) \right\}$$  
    
    where $\alpha_t = \frac{1}{2} \ln\left( \frac{1 - \epsilon_t}{\epsilon_t} \right) > 0$

    Bigger if wrong, smaller if right
Algorithm

AdaBoost (Schapire and Freund)

- Data distribution $D_t$
  - $D_1(i) = \frac{1}{m}$
  - Given $D_t$ and $h_t$:

$$D_{t+1}(i) \propto D_t(i) \cdot \exp\left\{-\alpha_t y_i h_t(x_i)\right\}$$  \hspace{1cm} (3)

where $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) > 0$

Weight by how good the weak learner is
AdaBoost (Schapire and Freund)

- **Data distribution** $D_t$
  - $D_1(i) = \frac{1}{m}$
  - Given $D_t$ and $h_t$:
    \[
    D_{t+1}(i) \propto D_t(i) \cdot \exp\{-\alpha_t y_i h_t(x_i)\}
    \]  
    where $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right) > 0$

- **Final classifier**:
  \[
  H_{\text{fin}}(x) = \text{sign} \left( \sum_t \alpha_t h_t(x) \right)
  \]
Toy Example
Round 1

\[ h_1 \]

\[ \varepsilon_1 = 0.30 \]
\[ \alpha_1 = 0.42 \]

\[ D_2 \]
Round 2

\[ \varepsilon_2 = 0.21 \]

\[ \alpha_2 = 0.65 \]
Round 3
Example

Final Classifier

$H_{\text{final}} = \text{sign}(0.42 + 0.65 + 0.92)$
Generalization

![Graph showing error vs. number of rounds (T) for training and test data.](image)
Generalization

(boosting C4.5 on “letter” dataset)
Training Error

First, we can prove that the training error goes down. If we write the the error at time $t$ as $\frac{1}{2} - \gamma_t$,

$$
\hat{R}(h) \leq \exp \left\{ -2 \sum_t \gamma_t^2 \right\}
$$

- If $\forall t : \gamma_t \geq \gamma > 0$, then $\hat{R}(h) \leq \exp \{ -2\gamma^2 T \}$

AdaBoost: do not need $\gamma$ or $T$ a priori
Repeatedly expand the definition of the distribution.

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

\[
D_{t-1}(i) \exp \{-\alpha_{t-1} y_i h_{t-1}(x_i)\} \exp \{-\alpha_t y_i h_t(x_i)\} \frac{1}{Z_{t-1} Z_t} \cdot \exp \{-y_i \sum_{s=1}^{t} \alpha_s h_s(x_i)\} \frac{1}{m \prod_{s=1}^{t} Z_s} 
\]
Training Error Intuition

- On round $t$ weight of examples incorrectly classified by $h_t$ is increased.
- If $x_i$ incorrectly classified by $H_T$, then $x_i$ wrong on (weighted) majority of $h_t$'s.
  - If $x_i$ incorrectly classified by $H_T$, then $x_i$ must have large weight under $D_T$.
  - But there can’t be many of them, since total weight $\leq 1$. 
Training Error Proof: It’s all about the Normalizers

\[ \hat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} 1[y_i g(x_i) \leq 0] \]  

Definition of training error
Training Error Proof: It’s all about the Normalizers

\[
\hat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1} [y_i g(x_i) \leq 0] \leq \frac{1}{m} \sum_{i=1}^{m} \exp \{-y_i g(x_i)\} \leq \mathbb{1} [u \leq 0] \leq \exp -u \text{ is true for all real } u.
\]
Training Error Proof: It’s all about the Normalizers

\[
\hat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} 1[y_i g(x_i) \leq 0]
\]

\[
\leq \frac{1}{m} \sum_{i=1}^{m} \exp \{-y_i g(x_i)\}
\]

Final distribution \(D_{t+1}(i)\)

\[
D_{t+1}(i) = \frac{\exp \{-y_i \sum_{s=1}^{t} \alpha_s h_s(x_i)\}}{m \prod_{s=1}^{t} Z_s}
\]
Training Error Proof: It’s all about the Normalizers

\[ \hat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}[y_i g(x_i) \leq 0] \]  

\[ \leq \frac{1}{m} \sum_{i=1}^{m} \exp \{-y_i g(x_i)\} \]  

\[ = \frac{1}{m} \sum_{i=1}^{m} \left[ m \prod_{t=1}^{T} Z_t \right] D_{T+1}(i) \]

$m$’s cancel, $D$ is a distribution
Training Error Proof: It’s all about the Normalizers

\[ \hat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}[y_i g(x_i) \leq 0] \]  

\[ \leq \frac{1}{m} \sum_{i=1}^{m} \exp \{-y_i g(x_i)\} \]  

\[ = \frac{1}{m} \sum_{i=1}^{m} \left[ m \prod_{t=1}^{T} Z_t \right] D_{T+1}(i) \]  

\[ = \prod_{t=1}^{T} Z_t \]
Training Error Proof: Weak Learner Errors

Single Weak Learner

\[ Z_t = \sum_{i=1}^{m} D_t(i) \exp\{-\alpha_t y_i h_t(x_i)\} \]  \hspace{1cm} (13)

\[ = \]  \hspace{1cm} (14)

\[ = \]  \hspace{1cm} (15)

\[ = \]  \hspace{1cm} (16)
Training Error Proof: Weak Learner Errors

Single Weak Learner

\[ Z_t = \sum_{i=1}^{m} D_t(i) \exp \{-\alpha_t y_i h_t(x_i)\} \quad (13) \]

\[ = \sum_{i:\text{right}} D_t(i) \exp \{-\alpha_t\} + \sum_{i:\text{wrong}} D_t(i) \exp \{\alpha_t\} \quad (14) \]

\[ = \quad (15) \]

\[ = \quad (16) \]
Theoretical Analysis

Training Error Proof: Weak Learner Errors

Single Weak Learner

\[
Z_t = \sum_{i=1}^{m} D_t(i) \exp \{-\alpha_t y_i h_t(x_i)\} 
\]

\[
= \sum_{i:\text{right}} D_t(i) \exp \{-\alpha_t\} + \sum_{i:\text{wrong}} D_t(i) \exp \{\alpha_t\} 
\]

\[
= (1 - \epsilon_t) \exp \{-\alpha_t\} + \epsilon_t \exp \{\alpha_t\} 
\]

(13) (14) (15) (16)
**Theoretical Analysis**

**Training Error Proof: Weak Learner Errors**

### Single Weak Learner

\[
Z_t = \sum_{i=1}^{m} D_t(i) \exp \left\{ -\alpha_t y_i h_t(x_i) \right\} \\
= \sum_{i:\text{right}} D_t(i) \exp \left\{ -\alpha_t \right\} + \sum_{i:\text{wrong}} D_t(i) \exp \left\{ \alpha_t \right\} \\
= (1 - \epsilon_t) \exp \left\{ -\alpha_t \right\} + \epsilon_t \exp \left\{ \alpha_t \right\} \\
= (1 - \epsilon_t) \sqrt{\frac{\epsilon_t}{1 - \epsilon_t}} + \epsilon_t \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}}
\]
Training Error Proof: Weak Learner Errors

Single Weak Learner

\[ Z_t = \left(1 - \epsilon_t\right) \sqrt{\frac{\epsilon_t}{1 - \epsilon_t}} + \epsilon_t \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \quad (13) \]

Normalization Product

\[ \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1 - \epsilon_t)} = \sqrt{1 - 4 \left(\frac{1}{2} - \epsilon_t\right)^2} \quad (14) \]

\[ (15) \]
**Training Error Proof: Weak Learner Errors**

**Normalization Product**

\[
\prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1 - \epsilon_t)} = \sqrt{1 - 4\left(\frac{1}{2} - \epsilon_t\right)^2} \tag{13}
\]

\[
\leq \prod_{t=1}^{T} \exp \left\{-2\left(\frac{1}{2} - \epsilon_t\right)^2\right\} \tag{14}
\]

\[
\leq \exp \left\{-2\left(\frac{1}{2} - \epsilon_t\right)^2\right\} \tag{15}
\]
Training Error Proof: Weak Learner Errors

Normalization Product

\[
\prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)} = \sqrt{1 - 4\left(\frac{1}{2} - \epsilon_t\right)^2}
\]  
\leq \prod_{t=1}^{T} \exp\left\{-2\left(\frac{1}{2} - \epsilon_t\right)^2\right\} 
\leq \exp\left\{-2 \sum_{t=1}^{T} \left(\frac{1}{2} - \epsilon_t\right)^2\right\}
\]
Generalization

VC Dimension

\[ \leq 2(d + 1)(T + 1)\lg[(T + 1)e] \]

Margin-based Analysis

AdaBoost maximizes a linear program maximizes an $L_1$ margin, and the weak learnability assumption requires data to be linearly separable with margin $2\gamma$. 
Practical Advantages of AdaBoost

- fast
- simple and easy to program
- no parameters to tune (except $T$)
- flexible: can combine with any learning algorithm
- no prior knowledge needed about weak learner
- provably effective, provided can consistently find rough rules of thumb
  - shift in mind set: goal now is merely to find classifiers barely better than random guessing
- versatile
  - can use with data that is textual, numeric, discrete, etc.
  - has been extended to learning problems well beyond binary classification
Caveats

- performance of AdaBoost depends on data and weak learner
- consistent with theory, AdaBoost can fail if
  - weak classifiers too complex
    - overfitting
  - weak classifiers too weak ($\gamma_t \to 0$ too quickly)
    - underfitting
    - low margins $\to$ overfitting
- empirically, AdaBoost seems especially susceptible to uniform noise