Boosting

Jordan Boyd-Graber
University of Colorado Boulder

LECTURE 10

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

• How to **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

general method of converting rough rules of thumb into highly accurate prediction rule

• assume given weak learning algorithm that can consistently find classifiers (rules of thumb) at least slightly better than random, say, accuracy $\geq 55\%$ (in two-class setting)

• given sufficient data, a boosting algorithm can provably construct single classifier with very high accuracy, say, 99\%
Plan

Algorithm

Example

Generalization

Theoretical Analysis
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

\[
h_t : X \mapsto \{-1, +1\}
\]

(1)

with small error \(\epsilon_t\) on \(D_t\):

\[
\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i]
\]

(2)
Algorithm

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
    \[ h_t : \mathcal{X} \mapsto \{-1, +1\} \] (1)
    with small error $\epsilon_t$ on $D_t$:
    \[ \epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i] \] (2)
  - Output final classifier $H_{final}$
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 $h_t$:
    $$D_{t+1}(i) \propto D_t(i) \cdot \exp \{ - \alpha_t y_i h_t(x_i) \}$$
    (3)
    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 $h_t$:

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

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

Bigger if wrong, smaller if right
AdaBoost (Schapire and Freund)

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

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

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

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 $h_t$:
    \[
    D_{t+1}(i) \propto D_t(i) \cdot \exp \{ -\alpha_t y_i h_t(x_i) \} \tag{3}
    \]
    where $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right) > 0$

- Final classifier:
  \[
  H_{fin}(x) = \text{sign} \left( \sum_t \alpha_t h_t(x) \right) \tag{4}
  \]
Plan

Algorithm

Example

Generalization

Theoretical Analysis
Toy Example
Example

Round 1

\[ h_1 \]

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

\[ D_2 \]
Round 2

\[
\begin{align*}
\epsilon_2 &= 0.21 \\
\alpha_2 &= 0.65
\end{align*}
\]
Round 3

$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$
Final Classifier

\[ H_{\text{final}} = \text{sign}(0.42 + 0.65 + 0.92) \]
Plan

Algorithm

Example

Generalization

Theoretical Analysis
Generalization

![Graph showing the relationship between error and the number of rounds (T). The graph illustrates the decrease in error for both training and test data as the number of rounds increases.]
Generalization

C4.5 test error

(boosting C4.5 on “letter” dataset)
Theoretical Analysis

Plan

Algorithm

Example

Generalization

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

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

(5)

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

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

\[ D_{t+1}(i) = \frac{D_t(i) \exp \{-\alpha_t y_i h_t(x_i)\}}{Z_t} \]  \hspace{1cm} (6)

\[ D_{t-1}(i) \exp \{-\alpha_{t-1} y_i h_{t-1}(x_i)\} \exp \{-\alpha_t y_i h_t(x_i)\} \]  \hspace{1cm} (7)

\[ \frac{Z_{t-1} Z_t}{m \prod_{s=1}^t Z_s} \exp \{-y_i \sum_{s=1}^t \alpha_s h_s(x_i)\} \]  \hspace{1cm} (8)
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

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

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

\[ \hat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} 1 \left[ y_i g(x_i) \leq 0 \right] \] (9)

\[ \leq \frac{1}{m} \sum_{i=1}^{m} \exp \{ -y_i g(x_i) \} \] (10)

(11)

\[ 1 \left[ u \leq 0 \right] \leq \exp -u \text{ is true for all real } u. \]
Training Error Proof: It’s all about the Normalizers

\[ \hat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} 1 \left[ y_i g(x_i) \leq 0 \right] \] (9)

\[ \leq \frac{1}{m} \sum_{i=1}^{m} \exp \left\{ -y_i g(x_i) \right\} \] (10)

\[ \leq \frac{1}{m} \sum_{i=1}^{m} \exp \left\{ -y_i g(x_i) \right\} \] (11)

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

\[ D_{t+1}(i) = \frac{\exp \left\{ -y_i \sum_{s=1}^{t} \alpha_s h_s(x_i) \right\}}{m \prod_{s=1}^{t} Z_s} \] (12)
Training Error Proof: It’s all about the Normalizers

\[ \hat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} 1 \left[ y_i g(x_i) \leq 0 \right] \] (9)

\[ \leq \frac{1}{m} \sum_{i=1}^{m} \exp \left\{ -y_i g(x_i) \right\} \] (10)

\[ = \frac{1}{m} \sum_{i=1}^{m} \left[ m \prod_{t=1}^{T} Z_t \right] D_{T+1}(i) \] (11)

\[ m \text{'s cancel, } D \text{ is a distribution} \] (12)
Training Error Proof: It’s all about the Normalizers

\[ \hat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1} [y_i g(x_i) \leq 0] \]  \hspace{1cm} (9)

\[ \leq \frac{1}{m} \sum_{i=1}^{m} \exp \{-y_i g(x_i)\} \]  \hspace{1cm} (10)

\[ = \frac{1}{m} \sum_{i=1}^{m} \left[ m \prod_{t=1}^{T} Z_t \right] D_{T+1}(i) \]  \hspace{1cm} (11)

\[ = \prod_{t=1}^{T} Z_t \]  \hspace{1cm} (12)
Single Weak Learner

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

= \quad (14)

= \quad (15)

= \quad (16)
Theoretical Analysis

Training Error Proof: Weak Learner Errors

Single Weak Learner

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

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

\[ = \quad (15) \]

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

Training Error Proof: Weak Learner Errors

Single Weak Learner

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

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

\[ = (1 - \epsilon_t) \exp \{-\alpha_t\} + \epsilon_t \exp \{\alpha_t\} \tag{15} \]

\[ = \tag{16} \]
Theoretical Analysis

Training Error Proof: Weak Learner Errors

Single Weak Learner

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

Single Weak Learner

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

Normalization Product

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

\[ \sqrt{1 - 4 \left(\frac{1}{2} - \epsilon_t\right)^2} \] (15)
Training Error Proof: Weak Learner Errors

Normalization Product

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

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

\[ \leq \exp \left\{ -2 \left( \frac{1}{2} - \epsilon_t \right)^2 \right\} \quad (15) \]
Theoretical Analysis

Training Error Proof: Weak Learner Errors

Normalization Product

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

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

\[
= \exp \left\{ -2 \sum_{t=1}^{T} \left(\frac{1}{2} - \epsilon_t\right)^2 \right\}
\tag{15}
\]
Generalization

**VC Dimension**

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

**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 $2\gamma$
Theoretical Analysis

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 ($\gamma_t \rightarrow 0$ too quickly)
  - underfitting
  - low margins $\rightarrow$ overfitting
- empirically, AdaBoost seems especially susceptible to uniform noise