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

Details

- How to **choose** examples
  concentrate on <u>hardest</u> examples (those most often misclassified by previous rules of thumb)
- How to **combine** rules of thumb
  take (weighted) majority vote of rules of thumb

Boosting

Definition

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

- assume given <u>weak learning algorithm</u> 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%

Formal Description

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

Formal Description

- Training set $(x_1, y_1) \dots (x_m, y_m)$
- $y_i \in \{-1, +1\}$ is the label of instance x_i
- For $t = 1, \dots T$:
    - Construct distribution D_t on $\{1, \dots, m\}$
    - Find weak classifier

$$h_t : \mathscr{X} \mapsto \{-1, +1\} \tag{1}$$

with small error ϵ_t on D_t:

$$\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i] \tag{2}$$

Formal Description

- Training set $(x_1, y_1) \dots (x_m, y_m)$
- $y_i \in \{-1, +1\}$ is the label of instance x_i
- For $t = 1, \dots T$:
    - Construct distribution D_t on $\{1, \dots, m\}$
    - Find weak classifier

      $$h_t : \mathscr{X} \mapsto \{-1, +1\} \tag{1}$$

      with small error ϵ_t on D_t:

      $$\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i] \tag{2}$$

    - 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:

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

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:

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

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:

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

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)

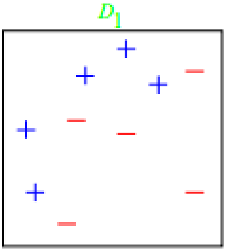
- Data distribution D_t
  - $D_1(i) = \frac{1}{m}$
  - Given D_t and h_t:

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

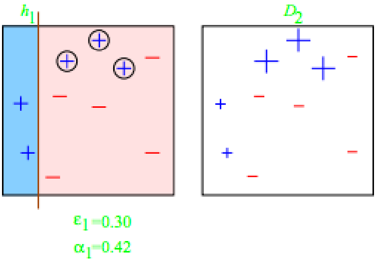
  where $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$
- Final classifier:

$$H_{fin}(x) = \text{sign}\left(\sum_t \alpha_t h_t(x)\right) \tag{4}$$

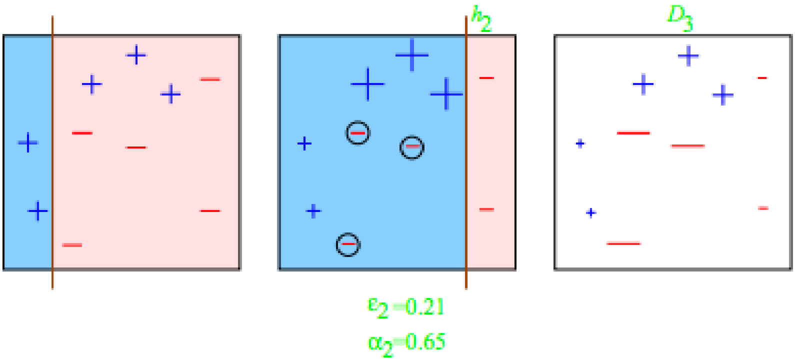
Toy Example



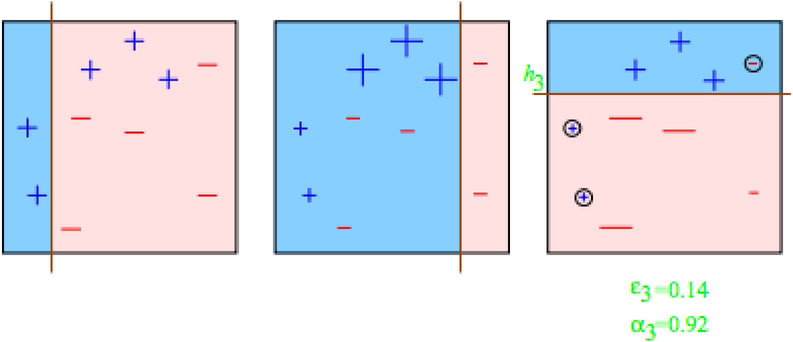
Round 1



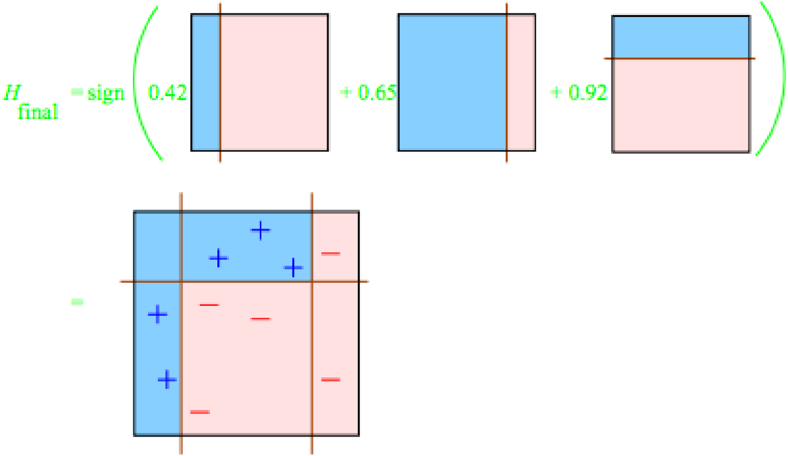
Round 2



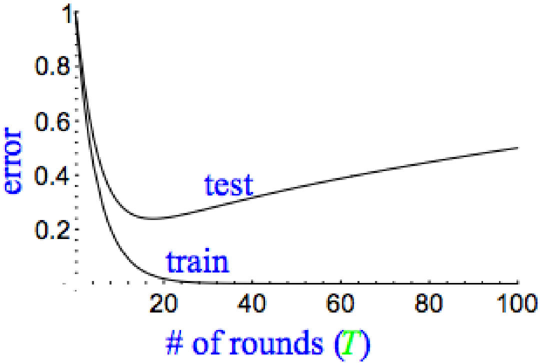
Round 3



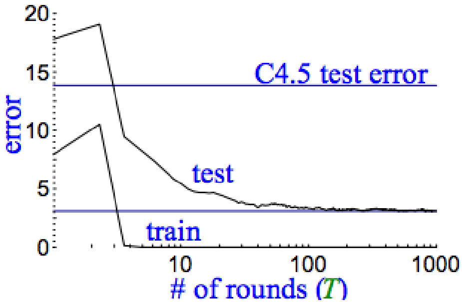
Final Classifier



Generalization



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

$$\hat{R}(h) \leq \exp\left\{-2\sum_t \gamma_t^2\right\} \tag{5}$$

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

Adaboost: do not need γ or T a priori

Training Error Proof: Preliminaries

Repeatedly expand the definition of the distribution.

$$D_{t+1}(i) = \frac{D_t(i)\exp\left\{-\alpha_t y_i h_t(x_i)\right\}}{Z_t} \tag{6}$$

$$\frac{D_{t-1}(i)\exp\left\{-\alpha_{t-1} y_i h_{t-1}(x_i)\right\}\exp\left\{-\alpha_t y_i h_t(x_i)\right\}}{Z_{t-1} Z_t} \tag{7}$$

$$\frac{\exp\left\{-y_i \sum_{s=1}^{t} \alpha_s h_s(x_i)\right\}}{m\prod_{s=1}^{t} Z_s} \tag{8}$$

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 ≤ 1

Training Error Proof: It's all about the Normalizers

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

$$\tag{10}$$

Definition of training error

Training Error Proof: It's all about the Normalizers

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

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

$$\tag{11}$$

$\mathbb{1}\left[u \leq 0\right] \leq \exp -u$ is true for all real u.

Training Error Proof: It's all about the Normalizers

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

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

$$\tag{11}$$

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

$$D_{t+1}(i) = \frac{\exp\left\{-y_i \sum_{s=1}^{t} \alpha_s h_s(x_i)\right\}}{m \prod_{s=1}^{t} Z_s} \tag{12}$$

Training Error Proof: It's all about the Normalizers

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

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

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

$$\tag{12}$$

m's cancel, *D* is a distribution

Training Error Proof: It's all about the Normalizers

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

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

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

$$= \prod_{t=1}^{T} Z_t \tag{12}$$

Training Error Proof: Weak Learner Errors

Single Weak Learner

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp\left\{-\alpha_t y_i h_t(x_i)\right\} \tag{13}$$
$$= \tag{14}$$
$$= \tag{15}$$
$$= \tag{16}$$

Training Error Proof: Weak Learner Errors

Single Weak Learner

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

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

$$= \tag{15}$$

$$= \tag{16}$$

Training Error Proof: Weak Learner Errors

Single Weak Learner

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

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

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

$$= \tag{16}$$

Training Error Proof: Weak Learner Errors

Single Weak Learner

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

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

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

$$= (1-\epsilon_t) \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} + \epsilon_t \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \tag{16}$$

Training Error Proof: Weak Learner Errors

Single Weak Learner

$$Z_t = (1-\epsilon_t)\sqrt{\frac{\epsilon_t}{1-\epsilon_t}} + \epsilon_t\sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \tag{13}$$

Normalization Product

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

$$\tag{15}$$

Training Error Proof: Weak Learner Errors

Normalization Product

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

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

$$\tag{15}$$

Training Error Proof: Weak Learner Errors

Normalization Product

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

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

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

Generalization

VC Dimension

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

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 2γ

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