

Theory and Applications of Boosting

Rob Schapire

Example: "How May I Help You?"

[Gorin et al.]

- **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**)

Example: “How May I Help You?”

[Gorin et al.]

- **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**)
- **observation:**
 - **easy** to find “rules of thumb” that are “often” correct
 - e.g.: “**IF** ‘card’ **occurs in utterance**
THEN predict ‘**CallingCard**’ ”
 - **hard** to find **single** highly accurate prediction rule

The Boosting Approach

- devise computer program for deriving rough rules of thumb
- apply procedure to subset of examples
- obtain rule of thumb
- apply to 2nd subset of examples
- obtain 2nd rule of thumb
- repeat T times

Details

- how to choose examples on each round?
 - concentrate on “hardest” examples (those most often misclassified by previous rules of thumb)
- how to combine rules of thumb into single prediction rule?
 - take (weighted) majority vote of rules of thumb

Boosting

- **boosting** = general method of converting rough rules of thumb into highly accurate prediction rule
- **technically:**
 - **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) [“**weak learning assumption**”]
 - given sufficient data, a **boosting algorithm** can **provably** construct single classifier with very high accuracy, say, 99%

Outline of Tutorial

- brief background
- basic algorithm and core theory
- other ways of understanding boosting
- experiments, applications and extensions

Brief Background

Strong and Weak Learnability

- boosting's roots are in “PAC” (Valiant) learning model
- get random examples from unknown, arbitrary distribution
- strong PAC learning algorithm:
 - for any distribution
with high probability
given polynomially many examples (and polynomial time)
can find classifier with arbitrarily small generalization
error

Strong and Weak Learnability

- boosting's roots are in “PAC” (Valiant) learning model
- get random examples from unknown, arbitrary distribution
- **strong** PAC learning algorithm:
 - for **any** distribution
with high probability
given polynomially many examples (and polynomial time)
can find classifier with **arbitrarily small** generalization error
- **weak** PAC learning algorithm
 - same, but generalization error only needs to be **slightly better than random guessing** ($\frac{1}{2} - \gamma$)
- [Kearns & Valiant '88]:
 - does weak learnability imply strong learnability?

Early Boosting Algorithms

- [Schapire '89]:
 - first provable boosting algorithm
- [Freund '90]:
 - “optimal” algorithm that “boosts by majority”
- [Drucker, Schapire & Simard '92]:
 - first experiments using boosting
 - limited by practical drawbacks

AdaBoost

- [Freund & Schapire '95]:
 - introduced “AdaBoost” algorithm
 - strong practical advantages over previous boosting algorithms
- experiments and applications using AdaBoost:

[Drucker & Cortes '96]

[Jackson & Craven '96]

[Freund & Schapire '96]

[Quinlan '96]

[Breiman '96]

[Maclin & Opitz '97]

[Bauer & Kohavi '97]

[Schwenk & Bengio '98]

[Schapire, Singer & Singhal '98]

[Abney, Schapire & Singer '99]

[Haruno, Shirai & Ooyama '99]

[Cohen & Singer '99]

[Dietterich '00]

[Schapire & Singer '00]

[Collins '00]

[Escudero, Márquez & Rigau '00]

[Iyer, Lewis, Schapire et al. '00]

[Onoda, Rätsch & Müller '00]

[Tieu & Viola '00]

[Walker, Rambow & Rogati '01]

[Rochery, Schapire, Rahim & Gupta '01]

[Merler, Furlanello, Larcher & Sboner '01]

[Di Fabrizio, Dutton, Gupta et al. '02]

[Qu, Adam, Yasui et al. '02]

[Tur, Schapire & Hakkani-Tür '03]

[Viola & Jones '04]

[Middendorf, Kundaje, Wiggins et al. '04]

⋮

- continuing development of theory and algorithms:

[Breiman '98, '99]

[Schapire, Freund, Bartlett & Lee '98]

[Grove & Schuurmans '98]

[Mason, Bartlett & Baxter '98]

[Schapire & Singer '99]

[Cohen & Singer '99]

[Freund & Mason '99]

[Domingo & Watanabe '99]

[Mason, Baxter, Bartlett & Frean '99]

[Duffy & Helmbold '99, '02]

[Freund & Mason '99]

[Ridgeway, Madigan & Richardson '99]

[Kivinen & Warmuth '99]

[Friedman, Hastie & Tibshirani '00]

[Rätsch, Onoda & Müller '00]

[Rätsch, Warmuth, Mika et al. '00]

[Allwein, Schapire & Singer '00]

[Friedman '01]

[Koltchinskii, Panchenko & Lozano '01]

[Collins, Schapire & Singer '02]

[Demiriz, Bennett & Shawe-Taylor '02]

[Lebanon & Lafferty '02]

[Wyner '02]

[Rudin, Daubechies & Schapire '03]

[Jiang '04]

[Lugosi & Vayatis '04]

[Zhang '04]

[Rosset, Zhu & Hastie '04]

[Zhang & Yu '05]

[Bartlett & Traskin '07]

[Mease & Wyner '07]

[Shalev-Shwartz & Singer '08]

⋮

Basic Algorithm and Core Theory

- introduction to AdaBoost
- analysis of training error
- analysis of test error based on margins theory

A Formal Description of Boosting

- given training set $(x_1, y_1), \dots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$

A Formal Description of Boosting

- given **training set** $(x_1, y_1), \dots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- for $t = 1, \dots, T$:
 - construct distribution D_t on $\{1, \dots, m\}$
 - find **weak classifier** (“rule of thumb”)

$$h_t : X \rightarrow \{-1, +1\}$$

with small **error** ϵ_t on D_t :

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

A Formal Description of Boosting

- given **training set** $(x_1, y_1), \dots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- for $t = 1, \dots, T$:
 - construct distribution D_t on $\{1, \dots, m\}$
 - find **weak classifier** (“rule of thumb”)

$$h_t : X \rightarrow \{-1, +1\}$$

with small **error** ϵ_t on D_t :

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

- output **final classifier** H_{final}

AdaBoost

[with Freund]

- constructing D_t :
 - $D_1(i) = 1/m$

AdaBoost

[with Freund]

- constructing D_t :
 - $D_1(i) = 1/m$
 - given D_t and h_t :

$$\begin{aligned}D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases} \\ &= \frac{D_t(i)}{Z_t} \exp(-\alpha_t y_i h_t(x_i))\end{aligned}$$

where $Z_t =$ normalization factor

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

- constructing D_t :
 - $D_1(i) = 1/m$
 - given D_t and h_t :

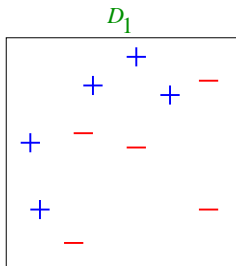
$$\begin{aligned}D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases} \\ &= \frac{D_t(i)}{Z_t} \exp(-\alpha_t y_i h_t(x_i))\end{aligned}$$

where $Z_t =$ normalization factor

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

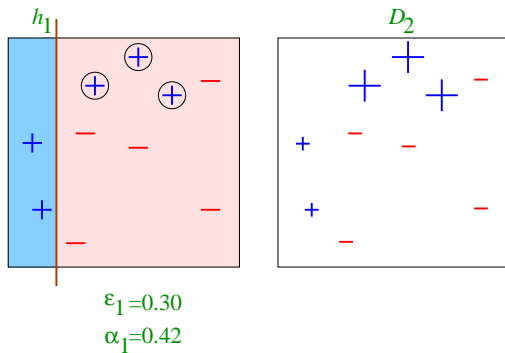
- final classifier:
 - $H_{\text{final}}(x) = \text{sign} \left(\sum_t \alpha_t h_t(x) \right)$

Toy Example

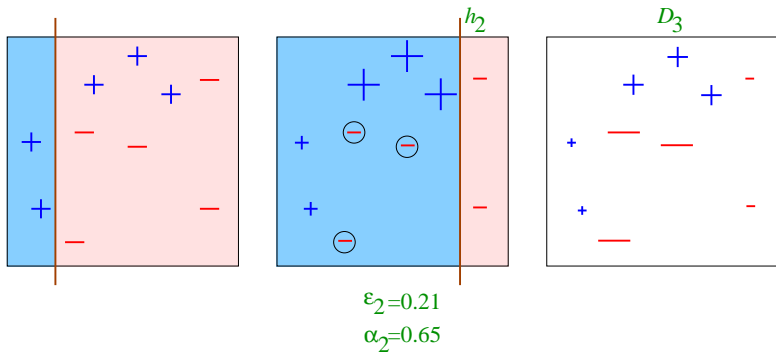


weak classifiers = vertical or horizontal half-planes

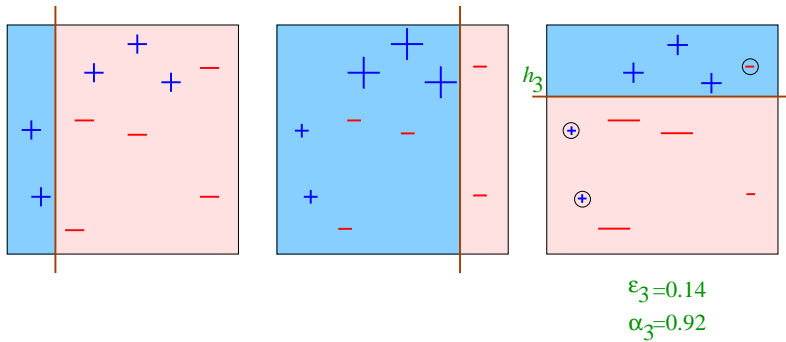
Round 1



Round 2



Round 3



Final Classifier

$$H_{\text{final}} = \text{sign} \left(0.42 \begin{array}{|c|} \hline \text{blue} \\ \hline \end{array} + 0.65 \begin{array}{|c|} \hline \text{blue} \\ \hline \end{array} + 0.92 \begin{array}{|c|} \hline \text{blue} \\ \hline \end{array} \right)$$

$$=$$

Analyzing the Training Error

[with Freund]

- Theorem:
 - write ϵ_t as $1/2 - \gamma_t$ [$\gamma_t = \text{"edge"}$]

Analyzing the Training Error

[with Freund]

- **Theorem:**

- write ϵ_t as $1/2 - \gamma_t$ [$\gamma_t =$ "edge"]
- then

$$\begin{aligned} \text{training error}(H_{\text{final}}) &\leq \prod_t \left[2\sqrt{\epsilon_t(1-\epsilon_t)} \right] \\ &= \prod_t \sqrt{1-4\gamma_t^2} \\ &\leq \exp\left(-2\sum_t \gamma_t^2\right) \end{aligned}$$

Analyzing the Training Error

[with Freund]

- **Theorem:**

- write ϵ_t as $1/2 - \gamma_t$ [$\gamma_t =$ “edge”]
- then

$$\begin{aligned} \text{training error}(H_{\text{final}}) &\leq \prod_t \left[2\sqrt{\epsilon_t(1-\epsilon_t)} \right] \\ &= \prod_t \sqrt{1-4\gamma_t^2} \\ &\leq \exp\left(-2\sum_t \gamma_t^2\right) \end{aligned}$$

- so: if $\forall t: \gamma_t \geq \gamma > 0$
then $\text{training error}(H_{\text{final}}) \leq e^{-2\gamma^2 T}$
- **AdaBoost is adaptive:**
 - does **not** need to know γ or T a priori
 - can exploit $\gamma_t \gg \gamma$

Proof

- let $F(x) = \sum_t \alpha_t h_t(x) \Rightarrow H_{\text{final}}(x) = \text{sign}(F(x))$
- *Step 1*: unwrapping recurrence:

$$\begin{aligned} D_{\text{final}}(i) &= \frac{1}{m} \frac{\exp\left(-y_i \sum_t \alpha_t h_t(x_i)\right)}{\prod_t Z_t} \\ &= \frac{1}{m} \frac{\exp(-y_i F(x_i))}{\prod_t Z_t} \end{aligned}$$

Proof (cont.)

- *Step 2:* training error(H_{final}) $\leq \prod_t Z_t$

Proof (cont.)

- *Step 2*: training error(H_{final}) $\leq \prod_t Z_t$
- Proof:

$$\text{training error}(H_{\text{final}}) = \frac{1}{m} \sum_i \begin{cases} 1 & \text{if } y_i \neq H_{\text{final}}(x_i) \\ 0 & \text{else} \end{cases}$$

Proof (cont.)

- *Step 2*: training error(H_{final}) $\leq \prod_t Z_t$
- Proof:

$$\begin{aligned} \text{training error}(H_{\text{final}}) &= \frac{1}{m} \sum_i \begin{cases} 1 & \text{if } y_i \neq H_{\text{final}}(x_i) \\ 0 & \text{else} \end{cases} \\ &= \frac{1}{m} \sum_i \begin{cases} 1 & \text{if } y_i F(x_i) \leq 0 \\ 0 & \text{else} \end{cases} \end{aligned}$$

Proof (cont.)

- *Step 2*: training error(H_{final}) $\leq \prod_t Z_t$
- Proof:

$$\begin{aligned} \text{training error}(H_{\text{final}}) &= \frac{1}{m} \sum_i \begin{cases} 1 & \text{if } y_i \neq H_{\text{final}}(x_i) \\ 0 & \text{else} \end{cases} \\ &= \frac{1}{m} \sum_i \begin{cases} 1 & \text{if } y_i F(x_i) \leq 0 \\ 0 & \text{else} \end{cases} \\ &\leq \frac{1}{m} \sum_i \exp(-y_i F(x_i)) \end{aligned}$$

Proof (cont.)

- *Step 2*: training error(H_{final}) $\leq \prod_t Z_t$
- Proof:

$$\begin{aligned} \text{training error}(H_{\text{final}}) &= \frac{1}{m} \sum_i \begin{cases} 1 & \text{if } y_i \neq H_{\text{final}}(x_i) \\ 0 & \text{else} \end{cases} \\ &= \frac{1}{m} \sum_i \begin{cases} 1 & \text{if } y_i F(x_i) \leq 0 \\ 0 & \text{else} \end{cases} \\ &\leq \frac{1}{m} \sum_i \exp(-y_i F(x_i)) \\ &= \sum_i D_{\text{final}}(i) \prod_t Z_t \end{aligned}$$

Proof (cont.)

- *Step 2*: training error(H_{final}) $\leq \prod_t Z_t$
- Proof:

$$\begin{aligned}\text{training error}(H_{\text{final}}) &= \frac{1}{m} \sum_i \begin{cases} 1 & \text{if } y_i \neq H_{\text{final}}(x_i) \\ 0 & \text{else} \end{cases} \\ &= \frac{1}{m} \sum_i \begin{cases} 1 & \text{if } y_i F(x_i) \leq 0 \\ 0 & \text{else} \end{cases} \\ &\leq \frac{1}{m} \sum_i \exp(-y_i F(x_i)) \\ &= \sum_i D_{\text{final}}(i) \prod_t Z_t \\ &= \prod_t Z_t\end{aligned}$$

Proof (cont.)

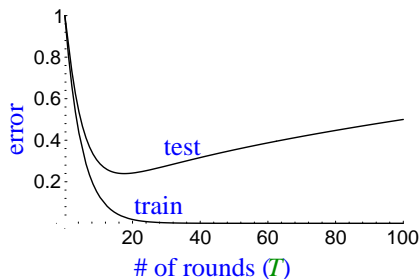
- *Step 3:* $Z_t = 2\sqrt{\epsilon_t(1 - \epsilon_t)}$

Proof (cont.)

- *Step 3:* $Z_t = 2\sqrt{\epsilon_t(1 - \epsilon_t)}$
- Proof:

$$\begin{aligned} Z_t &= \sum_i D_t(i) \exp(-\alpha_t y_i h_t(x_i)) \\ &= \sum_{i:y_i \neq h_t(x_i)} D_t(i) e^{\alpha_t} + \sum_{i:y_i = h_t(x_i)} D_t(i) e^{-\alpha_t} \\ &= \epsilon_t e^{\alpha_t} + (1 - \epsilon_t) e^{-\alpha_t} \\ &= 2\sqrt{\epsilon_t(1 - \epsilon_t)} \end{aligned}$$

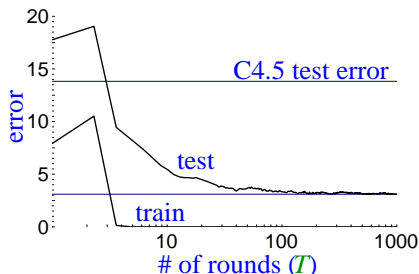
How Will Test Error Behave? (A First Guess)



expect:

- training error to continue to drop (or reach zero)
- test error to **increase** when H_{final} becomes “too complex”
 - “Occam's razor”
 - **overfitting**
 - hard to know when to stop training

Actual Typical Run



(boosting C4.5 on
"letter" dataset)

- test error does **not** increase, even after 1000 rounds
 - (total size > 2,000,000 nodes)
- test error continues to drop even after training error is zero!

	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1

- Occam's razor **wrongly** predicts "simpler" rule is better

A Better Story: The Margins Explanation

[with Freund, Bartlett & Lee]

- key idea:
 - training error only measures whether classifications are right or wrong
 - should also consider **confidence** of classifications

A Better Story: The Margins Explanation

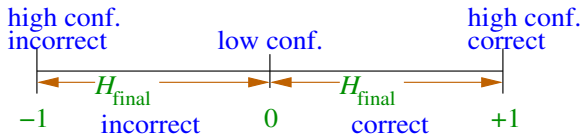
[with Freund, Bartlett & Lee]

- key idea:
 - training error only measures whether classifications are right or wrong
 - should also consider **confidence** of classifications
- recall: H_{final} is weighted majority vote of weak classifiers

A Better Story: The Margins Explanation

[with Freund, Bartlett & Lee]

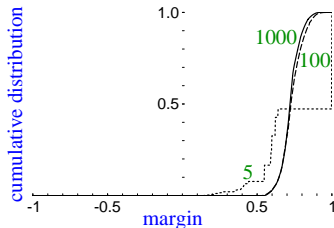
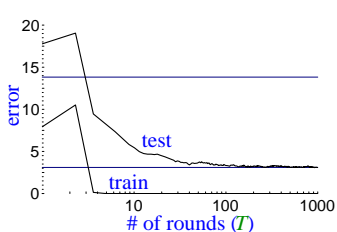
- key idea:
 - training error only measures whether classifications are right or wrong
 - should also consider **confidence** of classifications
- recall: H_{final} is weighted majority vote of weak classifiers
- measure confidence by **margin** = strength of the vote
= (weighted fraction voting correctly)
− (weighted fraction voting incorrectly)



Empirical Evidence: The Margin Distribution

- margin distribution

= cumulative distribution of margins of training examples



	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1
% margins ≤ 0.5	7.7	0.0	0.0
minimum margin	0.14	0.52	0.55

Theoretical Evidence: Analyzing Boosting Using Margins

- **Theorem:** large margins \Rightarrow better bound on generalization error (independent of number of rounds)

Theoretical Evidence: Analyzing Boosting Using Margins

- **Theorem:** large margins \Rightarrow better bound on generalization error (independent of number of rounds)
 - **proof idea:** if all margins are large, then can approximate final classifier by a much smaller classifier (just as polls can predict not-too-close election)

Theoretical Evidence: Analyzing Boosting Using Margins

- **Theorem:** large margins \Rightarrow better bound on generalization error (independent of number of rounds)
 - **proof idea:** if all margins are large, then can approximate final classifier by a much smaller classifier (just as polls can predict not-too-close election)
- **Theorem:** boosting tends to increase margins of training examples (given weak learning assumption)
 - moreover, larger edges \Rightarrow larger margins

Theoretical Evidence: Analyzing Boosting Using Margins

- **Theorem:** large margins \Rightarrow better bound on generalization error (independent of number of rounds)
 - **proof idea:** if all margins are large, then can approximate final classifier by a much smaller classifier (just as polls can predict not-too-close election)
- **Theorem:** boosting tends to increase margins of training examples (given weak learning assumption)
 - moreover, larger edges \Rightarrow larger margins
 - **proof idea:** similar to training error proof

Theoretical Evidence: Analyzing Boosting Using Margins

- **Theorem:** large margins \Rightarrow better bound on generalization error (independent of number of rounds)
 - **proof idea:** if all margins are large, then can approximate final classifier by a much smaller classifier (just as polls can predict not-too-close election)
- **Theorem:** boosting tends to increase margins of training examples (given weak learning assumption)
 - moreover, larger edges \Rightarrow larger margins
 - **proof idea:** similar to training error proof
- so:
although final classifier is getting larger,
margins are likely to be increasing,
so final classifier actually getting close to a simpler classifier,
driving down the test error

More Technically...

- with high probability, $\forall \theta > 0$:

$$\text{generalization error} \leq \hat{\mathbb{P}}_r[\text{margin} \leq \theta] + \tilde{O}\left(\frac{\sqrt{d/m}}{\theta}\right)$$

($\hat{\mathbb{P}}_r[\] = \text{empirical probability}$)

- bound depends on
 - $m = \#$ training examples
 - $d =$ “complexity” of weak classifiers
 - **entire** distribution of margins of training examples
- $\hat{\mathbb{P}}_r[\text{margin} \leq \theta] \rightarrow 0$ exponentially fast (in T) if
(error of h_t on D_t) $< 1/2 - \theta$ ($\forall t$)
 - so: if weak learning assumption holds, then all examples will quickly have “large” margins

Other Ways of Understanding AdaBoost

- game theory
- loss minimization
- an information-geometric view

Just a Game

[with Freund]

- can view boosting as a **game**, a formal interaction between **booster** and **weak learner**
- on each round t :
 - booster chooses distribution D_t
 - weak learner responds with weak classifier h_t
- **game theory**: studies interactions between all sorts of “players”

Game Theory

- game defined by matrix \mathbf{M} :

	Rock	Paper	Scissors
Rock	1/2	1	0
Paper	0	1/2	1
Scissors	1	0	1/2

- row player chooses row i
- column player chooses column j
(simultaneously)
- row player's goal: minimize loss $\mathbf{M}(i,j)$

Game Theory

- game defined by matrix **M**:

	Rock	Paper	Scissors
Rock	1/2	1	0
Paper	0	1/2	1
Scissors	1	0	1/2

- row player chooses row i
- column player chooses column j
(simultaneously)
- row player's goal: minimize loss $M(i, j)$
- usually allow randomized play:
 - players choose distributions **P** and **Q** over rows and columns
- row player's (expected) loss

$$\begin{aligned} &= \sum_{i,j} P(i)M(i,j)Q(j) \\ &= \mathbf{P}^T \mathbf{M} \mathbf{Q} \equiv M(\mathbf{P}, \mathbf{Q}) \end{aligned}$$

The Minmax Theorem

- von Neumann's minmax theorem:

$$\begin{aligned}\min_P \max_Q \mathbf{M}(\mathbf{P}, \mathbf{Q}) &= \max_Q \min_P \mathbf{M}(\mathbf{P}, \mathbf{Q}) \\ &= v \\ &= \text{"value" of game } \mathbf{M}\end{aligned}$$

The Minmax Theorem

- von Neumann's minmax theorem:

$$\begin{aligned}\min_P \max_Q \mathbf{M}(\mathbf{P}, \mathbf{Q}) &= \max_Q \min_P \mathbf{M}(\mathbf{P}, \mathbf{Q}) \\ &= v \\ &= \text{"value" of game } \mathbf{M}\end{aligned}$$

- in words:

- $v = \min \max$ means:

- row player has "minmax" strategy \mathbf{P}^*
such that \forall column strategy \mathbf{Q}
loss $\mathbf{M}(\mathbf{P}^*, \mathbf{Q}) \leq v$

The Minmax Theorem

- von Neumann's minmax theorem:

$$\begin{aligned}\min_P \max_Q \mathbf{M}(\mathbf{P}, \mathbf{Q}) &= \max_Q \min_P \mathbf{M}(\mathbf{P}, \mathbf{Q}) \\ &= v \\ &= \text{"value" of game } \mathbf{M}\end{aligned}$$

- in words:

- $v = \min \max$ means:

- row player has "minmax" strategy \mathbf{P}^*
such that \forall column strategy \mathbf{Q}
loss $\mathbf{M}(\mathbf{P}^*, \mathbf{Q}) \leq v$

- $v = \max \min$ means:

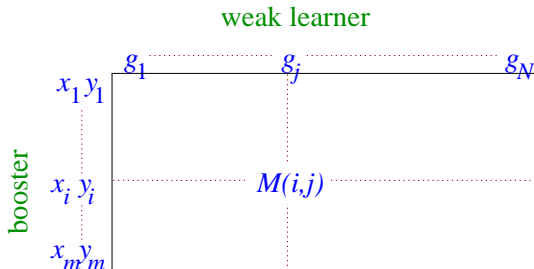
- this is optimal in sense that
column player has "maxmin" strategy \mathbf{Q}^*
such that \forall row strategy \mathbf{P}
loss $\mathbf{M}(\mathbf{P}, \mathbf{Q}^*) \geq v$

The Boosting Game

- row player \leftrightarrow booster
- column player \leftrightarrow weak learner

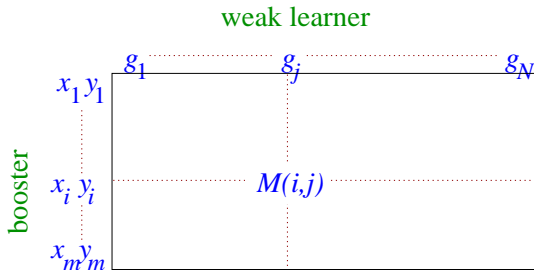
The Boosting Game

- row player \leftrightarrow booster
- column player \leftrightarrow weak learner
- let $\{g_1, \dots, g_N\}$ = space of **all** weak classifiers
- matrix **M**:
 - row \leftrightarrow example (x_i, y_i)
 - column \leftrightarrow weak classifier g_j



The Boosting Game

- row player \leftrightarrow booster
- column player \leftrightarrow weak learner
- let $\{g_1, \dots, g_N\}$ = space of **all** weak classifiers
- matrix **M**:
 - row \leftrightarrow example (x_i, y_i)
 - column \leftrightarrow weak classifier g_j
 - $M(i, j) = \begin{cases} 1 & \text{if } y_i = g_j(x_i) \\ 0 & \text{else} \end{cases}$
 - encodes which weak classifiers correct on which examples



Boosting and the Minmax Theorem

- γ -weak learning assumption says:
 \forall distribution \mathbf{P} over examples
 $\exists g_j$ such that

$$\Pr_{i \sim \mathbf{P}} [g_j(x_i) = y_i] \geq \frac{1}{2} + \gamma$$

Boosting and the Minmax Theorem

- γ -weak learning assumption says:
 \forall distribution \mathbf{P} over examples
 $\exists g_j$ such that

$$\Pr_{i \sim \mathbf{P}} [g_j(x_i) = y_i] \geq \frac{1}{2} + \gamma$$

- i.e.,

$$\min_{\mathbf{P}} \max_j \mathbf{M}(\mathbf{P}, j) \geq \frac{1}{2} + \gamma$$

Boosting and the Minmax Theorem

- γ -weak learning assumption says:
 \forall distribution \mathbf{P} over examples
 $\exists g_j$ such that

$$\Pr_{i \sim \mathbf{P}} [g_j(x_i) = y_i] \geq \frac{1}{2} + \gamma$$

- i.e., $\min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P}, \mathbf{Q}) = \min_{\mathbf{P}} \max_j \mathbf{M}(\mathbf{P}, j) \geq \frac{1}{2} + \gamma$

Boosting and the Minmax Theorem

- γ -weak learning assumption says:
 \forall distribution \mathbf{P} over examples
 $\exists g_j$ such that

$$\Pr_{i \sim \mathbf{P}} [g_j(x_i) = y_i] \geq \frac{1}{2} + \gamma$$

- i.e., $\min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P}, \mathbf{Q}) = \min_{\mathbf{P}} \max_j \mathbf{M}(\mathbf{P}, j) \geq \frac{1}{2} + \gamma$
- $\Rightarrow \max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{M}(\mathbf{P}, \mathbf{Q}) \geq \frac{1}{2} + \gamma$

Boosting and the Minmax Theorem

- γ -weak learning assumption says:
 \forall distribution \mathbf{P} over examples
 $\exists g_j$ such that

$$\Pr_{i \sim \mathbf{P}} [g_j(x_i) = y_i] \geq \frac{1}{2} + \gamma$$

- i.e., $\min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P}, \mathbf{Q}) = \min_{\mathbf{P}} \max_j \mathbf{M}(\mathbf{P}, j) \geq \frac{1}{2} + \gamma$
- $\Rightarrow \max_{\mathbf{Q}} \min_i \mathbf{M}(i, \mathbf{Q}) = \max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{M}(\mathbf{P}, \mathbf{Q}) \geq \frac{1}{2} + \gamma$

Boosting and the Minmax Theorem

- γ -weak learning assumption says:
 \forall distribution \mathbf{P} over examples
 $\exists g_j$ such that

$$\Pr_{i \sim \mathbf{P}} [g_j(x_i) = y_i] \geq \frac{1}{2} + \gamma$$

- i.e., $\min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P}, \mathbf{Q}) = \min_{\mathbf{P}} \max_j \mathbf{M}(\mathbf{P}, j) \geq \frac{1}{2} + \gamma$
- $\Rightarrow \max_{\mathbf{Q}} \min_i \mathbf{M}(i, \mathbf{Q}) = \max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{M}(\mathbf{P}, \mathbf{Q}) \geq \frac{1}{2} + \gamma$
- i.e., \exists distribution \mathbf{Q} over weak classifiers such that
 \forall example (x_i, y_i)

$$\Pr_{j \sim \mathbf{Q}} [g_j(x_i) = y_i] \geq \frac{1}{2} + \gamma$$

Boosting and the Minmax Theorem

- γ -weak learning assumption says:
 \forall distribution \mathbf{P} over examples
 $\exists g_j$ such that

$$\Pr_{i \sim \mathbf{P}} [g_j(x_i) = y_i] \geq \frac{1}{2} + \gamma$$

- i.e., $\min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{M}(\mathbf{P}, \mathbf{Q}) = \min_{\mathbf{P}} \max_j \mathbf{M}(\mathbf{P}, j) \geq \frac{1}{2} + \gamma$
- $\Rightarrow \max_{\mathbf{Q}} \min_i \mathbf{M}(i, \mathbf{Q}) = \max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{M}(\mathbf{P}, \mathbf{Q}) \geq \frac{1}{2} + \gamma$
- i.e., \exists distribution \mathbf{Q} over weak classifiers such that
 \forall example (x_i, y_i)

$$\Pr_{j \sim \mathbf{Q}} [g_j(x_i) = y_i] \geq \frac{1}{2} + \gamma$$

- i.e., \exists weighted majority of classifiers which correctly classifies
all examples with positive margin (2γ)

Idea for Boosting

- so:

γ -weak learning assumption holds

\Leftrightarrow (value of game \mathbf{M}) $\geq \frac{1}{2} + \gamma$

\Leftrightarrow every example classified correctly with margin $\geq 2\gamma$ by some weighted majority classifier

Idea for Boosting

- so:

γ -weak learning assumption holds

\Leftrightarrow (value of game \mathbf{M}) $\geq \frac{1}{2} + \gamma$

\Leftrightarrow every example classified correctly with margin $\geq 2\gamma$ by some weighted majority classifier

- further, weights are maxmin strategy for \mathbf{M}

Idea for Boosting

- so:
 - γ -weak learning assumption holds
 - \Leftrightarrow (value of game \mathbf{M}) $\geq \frac{1}{2} + \gamma$
 - \Leftrightarrow every example classified correctly with margin $\geq 2\gamma$ by some weighted majority classifier
 - further, weights are maxmin strategy for \mathbf{M}
- idea: use general algorithm for finding maxmin strategy through repeated play

Boosting and Repeated Play

- on each round t :
 - booster chooses:
 - distribution D_t over examples (x_i, y_i)
 - weak learner chooses:
 - weak classifier h_t
 - goal: h_t with high accuracy w.r.t. D_t

Boosting and Repeated Play

- booster and weak learner play game **M** repeatedly
- on each round t :
 - booster chooses:
 - distribution D_t over examples (x_i, y_i)
 - weak learner chooses:
 - weak classifier h_t
 - goal: h_t with high accuracy w.r.t. D_t

Boosting and Repeated Play

- booster and weak learner play game \mathbf{M} repeatedly
- on each round t :
 - booster chooses:
 - distribution D_t over examples (x_i, y_i)
 \leftrightarrow distribution \mathbf{P}_t over rows of \mathbf{M}
 - weak learner chooses:
 - weak classifier h_t
 - goal: h_t with high accuracy w.r.t. D_t

Boosting and Repeated Play

- booster and weak learner play game \mathbf{M} repeatedly
- on each round t :
 - booster chooses:
 - distribution D_t over examples (x_i, y_i)
 \leftrightarrow distribution \mathbf{P}_t over rows of \mathbf{M}
 - weak learner chooses:
 - weak classifier h_t
 \leftrightarrow column of \mathbf{M}
 \leftrightarrow (pure) distribution \mathbf{Q}_t over columns of \mathbf{M}
 - goal: h_t with high accuracy w.r.t. D_t

Boosting and Repeated Play

- booster and weak learner play game \mathbf{M} repeatedly
- on each round t :
 - booster chooses:
 - distribution D_t over examples (x_i, y_i)
 \leftrightarrow distribution \mathbf{P}_t over rows of \mathbf{M}
 - weak learner chooses:
 - weak classifier h_t
 \leftrightarrow column of \mathbf{M}
 \leftrightarrow (pure) distribution \mathbf{Q}_t over columns of \mathbf{M}
 - goal: h_t with high accuracy w.r.t. D_t
 \leftrightarrow high “loss” $\mathbf{M}(\mathbf{P}_t, \mathbf{Q}_t)$

Boosting and Repeated Play

- booster and weak learner play game \mathbf{M} repeatedly
- on each round t :
 - booster chooses:
 - distribution D_t over examples (x_i, y_i)
 \leftrightarrow distribution \mathbf{P}_t over rows of \mathbf{M}
 - weak learner chooses:
 - weak classifier h_t
 \leftrightarrow column of \mathbf{M}
 \leftrightarrow (pure) distribution \mathbf{Q}_t over columns of \mathbf{M}
 - goal: h_t with high accuracy w.r.t. D_t
 \leftrightarrow high “loss” $\mathbf{M}(\mathbf{P}_t, \mathbf{Q}_t)$
- in general, can show average of \mathbf{P}_t 's and \mathbf{Q}_t 's converge to approximate minmax/maxmin strategies

AdaBoost and Game Theory

- summarizing:
 - weak learning assumption implies maxmin strategy for **M** defines large-margin classifier
 - AdaBoost finds maxmin strategy by applying **general** algorithm for solving games through repeated play

AdaBoost and Game Theory

- summarizing:
 - weak learning assumption implies maxmin strategy for M defines large-margin classifier
 - AdaBoost finds maxmin strategy by applying general algorithm for solving games through repeated play
- consequences:
 - weights on weak classifiers converge to (approximately) maxmin strategy for game M
 - (average) of distributions D_t converges to (approximately) minmax strategy
 - margins and edges connected via minmax theorem
 - explains why AdaBoost maximizes margins

AdaBoost and Game Theory

- summarizing:
 - weak learning assumption implies maxmin strategy for M defines large-margin classifier
 - AdaBoost finds maxmin strategy by applying general algorithm for solving games through repeated play
- consequences:
 - weights on weak classifiers converge to (approximately) maxmin strategy for game M
 - (average) of distributions D_t converges to (approximately) minmax strategy
 - margins and edges connected via minmax theorem
 - explains why AdaBoost maximizes margins
- different instantiation of game-playing algorithm gives on-line learning algorithms (such as weighted majority algorithm)

AdaBoost and Loss Minimization

- many (most?) learning and statistical methods can be viewed as minimizing **loss** (a.k.a. cost or objective) function measuring **fit to data**:
 - e.g. least squares regression $\sum_i (F(x_i) - y_i)^2$

AdaBoost and Loss Minimization

- many (most?) learning and statistical methods can be viewed as minimizing **loss** (a.k.a. cost or objective) function measuring **fit to data**:
 - e.g. least squares regression $\sum_i (F(x_i) - y_i)^2$
- AdaBoost also minimizes a loss function

AdaBoost and Loss Minimization

- many (most?) learning and statistical methods can be viewed as minimizing **loss** (a.k.a. cost or objective) function measuring **fit to data**:
 - e.g. least squares regression $\sum_i (F(x_i) - y_i)^2$
- AdaBoost also minimizes a loss function
- **helpful** to understand because:
 - clarifies goal of algorithm and useful in proving convergence properties
 - decoupling of algorithm from its objective means:
 - faster algorithms possible for same objective
 - same algorithm may generalize for new learning challenges

What AdaBoost Minimizes

- recall proof of training error bound:
 - training error(H_{final}) $\leq \prod_t Z_t$
 - $Z_t = \epsilon_t e^{\alpha_t} + (1 - \epsilon_t) e^{-\alpha_t} = 2\sqrt{\epsilon_t(1 - \epsilon_t)}$

What AdaBoost Minimizes

- recall proof of training error bound:
 - training error(H_{final}) $\leq \prod_t Z_t$
 - $Z_t = \epsilon_t e^{\alpha_t} + (1 - \epsilon_t) e^{-\alpha_t} = 2\sqrt{\epsilon_t(1 - \epsilon_t)}$
- closer look:
 - α_t chosen to minimize Z_t
 - h_t chosen to minimize ϵ_t
 - same as minimizing Z_t
(since increasing in ϵ_t on $[0, 1/2]$)

What AdaBoost Minimizes

- recall proof of training error bound:
 - training error(H_{final}) $\leq \prod_t Z_t$
 - $Z_t = \epsilon_t e^{\alpha_t} + (1 - \epsilon_t) e^{-\alpha_t} = 2\sqrt{\epsilon_t(1 - \epsilon_t)}$
- closer look:
 - α_t chosen to minimize Z_t
 - h_t chosen to minimize ϵ_t
 - same as minimizing Z_t
(since increasing in ϵ_t on $[0, 1/2]$)
- so: both AdaBoost and weak learner minimize Z_t on round t
 - equivalent to **greedily** minimizing $\prod_t Z_t$

AdaBoost and Exponential Loss

- so AdaBoost is greedy procedure for minimizing

$$\prod_t z_t$$

AdaBoost and Exponential Loss

- so AdaBoost is greedy procedure for minimizing exponential loss

$$\prod_t Z_t = \frac{1}{m} \sum_i \exp(-y_i F(x_i))$$

where

$$F(x) = \sum_t \alpha_t h_t(x)$$

AdaBoost and Exponential Loss

- so AdaBoost is greedy procedure for minimizing exponential loss

$$\prod_t Z_t = \frac{1}{m} \sum_i \exp(-y_i F(x_i))$$

where

$$F(x) = \sum_t \alpha_t h_t(x)$$

- why exponential loss?
 - intuitively, strongly favors $F(x_i)$ to have same sign as y_i
 - upper bound on training error
 - smooth and convex (but very loose)

AdaBoost and Exponential Loss

- so AdaBoost is greedy procedure for minimizing exponential loss

$$\prod_t Z_t = \frac{1}{m} \sum_i \exp(-y_i F(x_i))$$

where

$$F(x) = \sum_t \alpha_t h_t(x)$$

- why exponential loss?
 - intuitively, strongly favors $F(x_i)$ to have same sign as y_i
 - upper bound on training error
 - smooth and convex (but very loose)
- how does AdaBoost minimize it?

Coordinate Descent

[Breiman]

- $\{g_1, \dots, g_N\}$ = space of **all** weak classifiers
- then can write $F(x) = \sum_t \alpha_t h_t(x) = \sum_{j=1}^N \lambda_j g_j(x)$
- want to find $\lambda_1, \dots, \lambda_N$ to minimize

$$L(\lambda_1, \dots, \lambda_N) = \sum_i \exp \left(-y_i \sum_j \lambda_j g_j(x_i) \right)$$

Coordinate Descent

[Breiman]

- $\{g_1, \dots, g_N\}$ = space of **all** weak classifiers
- then can write $F(x) = \sum_t \alpha_t h_t(x) = \sum_{j=1}^N \lambda_j g_j(x)$
- want to find $\lambda_1, \dots, \lambda_N$ to minimize

$$L(\lambda_1, \dots, \lambda_N) = \sum_i \exp \left(-y_i \sum_j \lambda_j g_j(x_i) \right)$$

- AdaBoost is actually doing **coordinate descent** on this optimization problem:
 - initially, all $\lambda_j = 0$
 - each round: choose **one** coordinate λ_j (corresponding to h_t) and update (increment by α_t)
 - choose update causing **biggest decrease** in loss
- powerful technique for minimizing over huge space of functions

Functional Gradient Descent

[Mason et al.][Friedman]

- want to minimize

$$L(F) = L(F(x_1), \dots, F(x_m)) = \sum_i \exp(-y_i F(x_i))$$

Functional Gradient Descent

[Mason et al.][Friedman]

- want to minimize

$$L(F) = L(F(x_1), \dots, F(x_m)) = \sum_i \exp(-y_i F(x_i))$$

- say have current estimate F and want to improve
- to do **gradient descent**, would like update

$$F \leftarrow F - \alpha \nabla_F L(F)$$

Functional Gradient Descent

[Mason et al.][Friedman]

- want to minimize

$$L(F) = L(F(x_1), \dots, F(x_m)) = \sum_i \exp(-y_i F(x_i))$$

- say have current estimate F and want to improve
- to do **gradient descent**, would like update

$$F \leftarrow F - \alpha \nabla_F L(F)$$

- but update **restricted** in class of weak classifiers

$$F \leftarrow F + \alpha h_t$$

Functional Gradient Descent

[Mason et al.][Friedman]

- want to minimize

$$L(F) = L(F(x_1), \dots, F(x_m)) = \sum_i \exp(-y_i F(x_i))$$

- say have current estimate F and want to improve
- to do **gradient descent**, would like update

$$F \leftarrow F - \alpha \nabla_F L(F)$$

- but update **restricted** in class of weak classifiers

$$F \leftarrow F + \alpha h_t$$

- so choose h_t “closest” to $-\nabla_F L(F)$
- equivalent to AdaBoost

Estimating Conditional Probabilities

[Friedman, Hastie & Tibshirani]

- often want to estimate **probability** that $y = +1$ given x
- AdaBoost minimizes (empirical version of):

$$E_{x,y} \left[e^{-yF(x)} \right] = E_x \left[\Pr [y = +1|x] e^{-F(x)} + \Pr [y = -1|x] e^{F(x)} \right]$$

where x, y random from true distribution

Estimating Conditional Probabilities

[Friedman, Hastie & Tibshirani]

- often want to estimate **probability** that $y = +1$ given x
- AdaBoost minimizes (empirical version of of):

$$E_{x,y} \left[e^{-yF(x)} \right] = E_x \left[\Pr [y = +1|x] e^{-F(x)} + \Pr [y = -1|x] e^{F(x)} \right]$$

where x, y random from true distribution

- over **all** F , minimized when

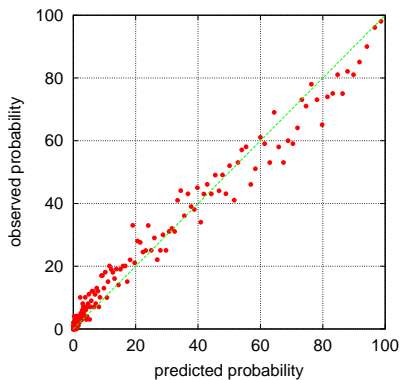
$$F(x) = \frac{1}{2} \cdot \ln \left(\frac{\Pr [y = +1|x]}{\Pr [y = -1|x]} \right)$$

or

$$\Pr [y = +1|x] = \frac{1}{1 + e^{-2F(x)}}$$

- so, to convert F output by AdaBoost to probability estimate, use same formula

Calibration Curve



- order examples by F value output by AdaBoost
- break into bins of fixed size
- for each bin, plot a point:
 - x -value: average estimated probability of examples in bin
 - y -value: actual fraction of positive examples in bin

Benefits of Loss-minimization View

- immediate generalization to other loss functions and learning problems
 - e.g. squared error for regression
 - e.g. logistic regression
(by only changing one line of AdaBoost)
- sensible approach for converting output of boosting into conditional probability estimates
- basis for proving AdaBoost is statistically “consistent”
 - i.e., under right assumptions, converges to best possible classifier [Bartlett & Traskin]

A Note of Caution

- tempting to conclude:
 - AdaBoost is **just** an algorithm for minimizing exponential loss
 - AdaBoost works **only because** of its loss function
 - ∴ more powerful optimization techniques for same loss should work even better

A Note of Caution

- tempting (but **incorrect!**) to conclude:
 - AdaBoost is **just** an algorithm for minimizing exponential loss
 - AdaBoost works **only because** of its loss function
 - ∴ more powerful optimization techniques for same loss should work even better
- **incorrect** because:
 - **other** algorithms that minimize exponential loss can give very **poor** generalization performance compared to AdaBoost
- for example...

An Experiment

- data:
 - instances x uniform from $\{-1, +1\}^{10,000}$
 - label $y =$ majority vote of three coordinates
 - weak classifier = single coordinate (or its negation)
 - training set size $m = 1000$

An Experiment

- data:
 - instances x uniform from $\{-1, +1\}^{10,000}$
 - label $y =$ majority vote of three coordinates
 - weak classifier = single coordinate (or its negation)
 - training set size $m = 1000$
- algorithms (all provably minimize exponential loss):
 - standard AdaBoost
 - gradient descent on exponential loss
 - AdaBoost, but in which weak classifiers chosen at random

An Experiment

- data:
 - instances x uniform from $\{-1, +1\}^{10,000}$
 - label $y =$ majority vote of three coordinates
 - weak classifier = single coordinate (or its negation)
 - training set size $m = 1000$
- algorithms (all provably minimize exponential loss):
 - standard AdaBoost
 - gradient descent on exponential loss
 - AdaBoost, but in which weak classifiers chosen at random
- results:

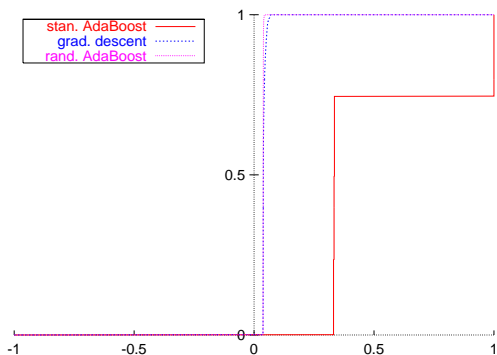
exp. loss	% test error [# rounds]		
	stand. AdaB.	grad. desc.	random AdaB.
10^{-10}	0.0 [94]	40.1 [2]	39.0 [23,115]
10^{-20}	0.0 [190]	40.2 [7]	40.3 [47,207]
10^{-40}	0.0 [382]	40.1 [29]	41.7 [96,163]
10^{-100}	0.0 [956]	40.2 [115]	—

An Experiment (cont.)

- conclusions:
 - not just **what** is being minimized that matters, but **how** it is being minimized
 - loss-minimization view has benefits and is fundamental to understanding AdaBoost
 - but is **limited** in what it says about generalization

An Experiment (cont.)

- conclusions:
 - not just **what** is being minimized that matters, but **how** it is being minimized
 - loss-minimization view has benefits and is fundamental to understanding AdaBoost
 - but is **limited** in what it says about generalization
- results **are** consistent with **margins theory**



A Dual Information-geometric Perspective

- loss minimization focuses on **function** computed by AdaBoost (i.e., weights on **weak classifiers**)

A Dual Information-geometric Perspective

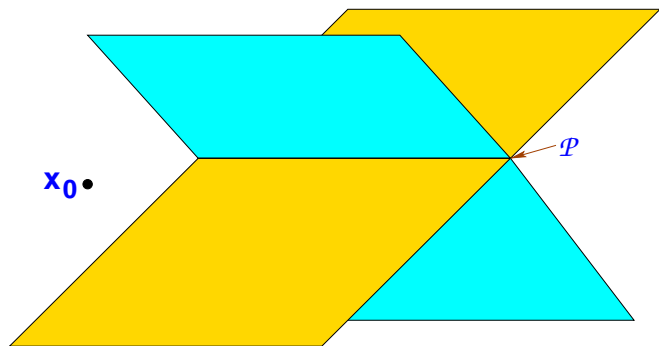
- loss minimization focuses on **function** computed by AdaBoost (i.e., weights on **weak classifiers**)
- **dual view**: instead focus on **distributions** D_t (i.e., weights on **examples**)

A Dual Information-geometric Perspective

- loss minimization focuses on **function** computed by AdaBoost (i.e., weights on **weak classifiers**)
- **dual view**: instead focus on **distributions** D_t (i.e., weights on **examples**)
- dual perspective combines **geometry** and **information theory**
- exposes underlying mathematical **structure**
- basis for proving **convergence**

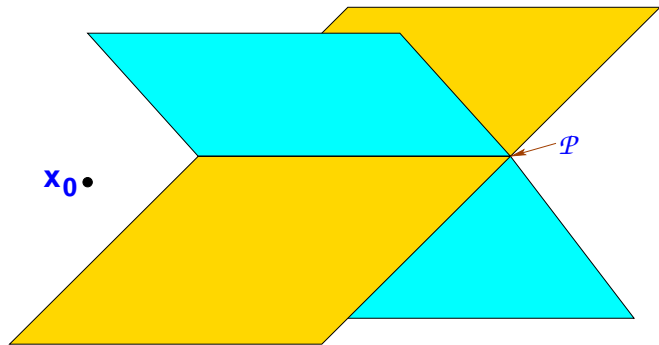
An Iterative-projection Algorithm

- say want to find point closest to \mathbf{x}_0 in set $\mathcal{P} = \{ \text{intersection of } N \text{ hyperplanes} \}$



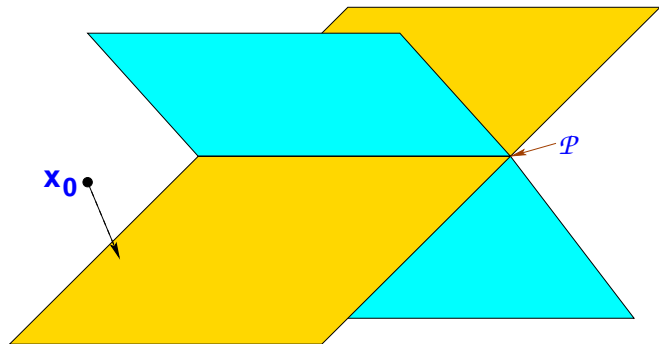
An Iterative-projection Algorithm

- say want to find point closest to \mathbf{x}_0 in set $\mathcal{P} = \{ \text{intersection of } N \text{ hyperplanes} \}$
- algorithm: [Bregman; Censor & Zenios]
 - start at \mathbf{x}_0
 - repeat: pick a hyperplane and project onto it



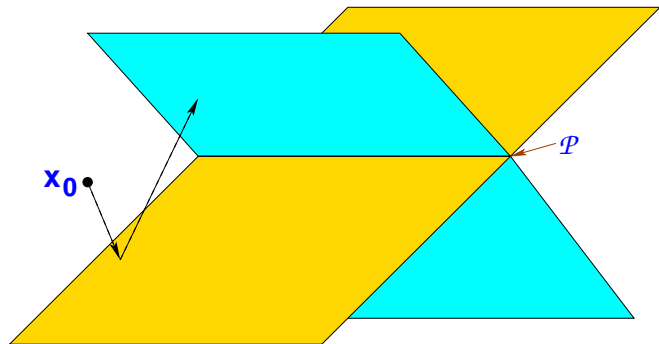
An Iterative-projection Algorithm

- say want to find point closest to \mathbf{x}_0 in set $\mathcal{P} = \{ \text{intersection of } N \text{ hyperplanes} \}$
- algorithm: [Bregman; Censor & Zenios]
 - start at \mathbf{x}_0
 - repeat: pick a hyperplane and project onto it



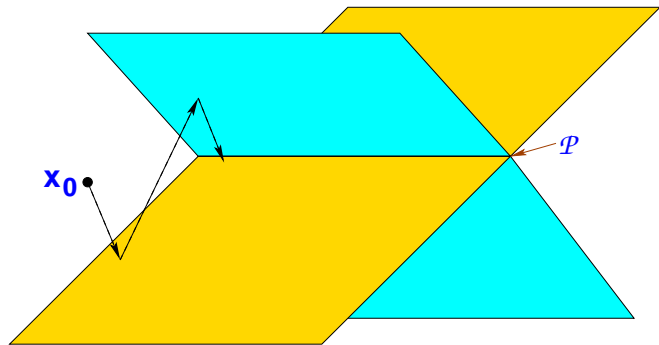
An Iterative-projection Algorithm

- say want to find point closest to \mathbf{x}_0 in set $\mathcal{P} = \{ \text{intersection of } N \text{ hyperplanes} \}$
- algorithm: [Bregman; Censor & Zenios]
 - start at \mathbf{x}_0
 - repeat: pick a hyperplane and project onto it



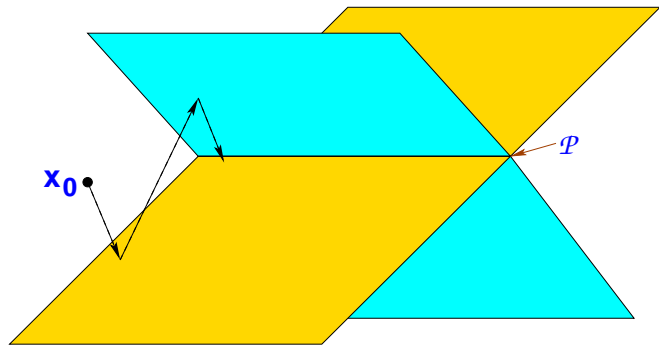
An Iterative-projection Algorithm

- say want to find point closest to \mathbf{x}_0 in set $\mathcal{P} = \{ \text{intersection of } N \text{ hyperplanes} \}$
- algorithm: [Bregman; Censor & Zenios]
 - start at \mathbf{x}_0
 - repeat: pick a hyperplane and project onto it



An Iterative-projection Algorithm

- say want to find point closest to \mathbf{x}_0 in set $\mathcal{P} = \{ \text{intersection of } N \text{ hyperplanes} \}$
- algorithm: [Bregman; Censor & Zenios]
 - start at \mathbf{x}_0
 - repeat: pick a hyperplane and project onto it



- if $\mathcal{P} \neq \emptyset$, under general conditions, will converge correctly

AdaBoost is an Iterative Projection Algorithm

[Kivinen & Warmuth]

- points = distributions D_t over training examples
- distance = relative entropy:

$$\text{RE}(P \parallel Q) = \sum_i P(i) \ln \left(\frac{P(i)}{Q(i)} \right)$$

- initial point \mathbf{x}_0 = uniform distribution

AdaBoost is an Iterative Projection Algorithm

[Kivinen & Warmuth]

- points = distributions D_t over training examples
- distance = relative entropy:

$$\text{RE}(P \parallel Q) = \sum_i P(i) \ln \left(\frac{P(i)}{Q(i)} \right)$$

- initial point \mathbf{x}_0 = uniform distribution
- hyperplanes defined by all possible weak classifiers g_j :

$$\sum_i D(i) y_i g_j(x_i) = 0$$

AdaBoost is an Iterative Projection Algorithm

[Kivinen & Warmuth]

- points = distributions D_t over training examples
- distance = relative entropy:

$$\text{RE}(P \parallel Q) = \sum_i P(i) \ln \left(\frac{P(i)}{Q(i)} \right)$$

- initial point \mathbf{x}_0 = uniform distribution
- hyperplanes defined by all possible weak classifiers g_j :

$$\sum_i D(i) y_i g_j(x_i) = 0 \Leftrightarrow \Pr_{i \sim D} [g_j(x_i) \neq y_i] = \frac{1}{2}$$

- intuition: looking for “hardest” distribution

AdaBoost as Iterative Projection (cont.)

- algorithm:
 - start at $D_1 = \text{uniform}$
 - for $t = 1, 2, \dots$:
 - pick hyperplane/weak classifier $h_t \leftrightarrow g_j$
 - $D_{t+1} = (\text{entropy})$ projection of D_t onto hyperplane
$$= \arg \min_{D: \sum_i D(i) y_i g_j(x_i) = 0} \text{RE}(D \parallel D_t)$$

AdaBoost as Iterative Projection (cont.)

- algorithm:
 - start at $D_1 = \text{uniform}$
 - for $t = 1, 2, \dots$:
 - pick hyperplane/weak classifier $h_t \leftrightarrow g_j$
 - $D_{t+1} = (\text{entropy})$ projection of D_t onto hyperplane
$$= \arg \min_{D: \sum_i D(i)y_i g_j(x_i)=0} \text{RE}(D \parallel D_t)$$
- claim: **equivalent** to AdaBoost
- further: choosing h_t with minimum error \equiv choosing **farthest** hyperplane

Boosting as Maximum Entropy

- corresponding optimization problem:

$$\min_{D \in \mathcal{P}} \text{RE}(D \parallel \text{uniform})$$

- where

$$\begin{aligned} \mathcal{P} &= \text{feasible set} \\ &= \left\{ D : \sum_i D(i) y_i g_j(x_i) = 0 \quad \forall j \right\} \end{aligned}$$

Boosting as Maximum Entropy

- corresponding optimization problem:

$$\min_{D \in \mathcal{P}} \text{RE}(D \parallel \text{uniform}) \leftrightarrow \max_{D \in \mathcal{P}} \text{entropy}(D)$$

- where

$$\begin{aligned} \mathcal{P} &= \text{feasible set} \\ &= \left\{ D : \sum_i D(i) y_i g_j(x_i) = 0 \quad \forall j \right\} \end{aligned}$$

Boosting as Maximum Entropy

- corresponding optimization problem:

$$\min_{D \in \mathcal{P}} \text{RE}(D \parallel \text{uniform}) \leftrightarrow \max_{D \in \mathcal{P}} \text{entropy}(D)$$

- where

$$\begin{aligned} \mathcal{P} &= \text{feasible set} \\ &= \left\{ D : \sum_i D(i) y_i g_j(x_i) = 0 \quad \forall j \right\} \end{aligned}$$

- $\mathcal{P} \neq \emptyset \Leftrightarrow$ weak learning assumption does **not** hold
 - in this case, $D_t \rightarrow$ (unique) solution

Boosting as Maximum Entropy

- corresponding optimization problem:

$$\min_{D \in \mathcal{P}} \text{RE}(D \parallel \text{uniform}) \leftrightarrow \max_{D \in \mathcal{P}} \text{entropy}(D)$$

- where

$$\begin{aligned} \mathcal{P} &= \text{feasible set} \\ &= \left\{ D : \sum_i D(i) y_i g_j(x_i) = 0 \quad \forall j \right\} \end{aligned}$$

- $\mathcal{P} \neq \emptyset \Leftrightarrow$ weak learning assumption does **not** hold
 - in this case, $D_t \rightarrow$ (unique) solution
- if weak learning assumption **does** hold then
 - $\mathcal{P} = \emptyset$
 - D_t can **never** converge
 - dynamics are fascinating but unclear in this case

Unifying the Two Cases

[with Collins & Singer]

- two distinct cases:
 - weak learning assumption holds
 - $\mathcal{P} = \emptyset$
 - dynamics unclear
 - weak learning assumption does **not** hold
 - $\mathcal{P} \neq \emptyset$
 - can prove convergence of D_t 's

Unifying the Two Cases

[with Collins & Singer]

- two distinct cases:
 - weak learning assumption holds
 - $\mathcal{P} = \emptyset$
 - dynamics unclear
 - weak learning assumption does **not** hold
 - $\mathcal{P} \neq \emptyset$
 - can prove convergence of D_t 's
- to **unify**: work instead with **unnormalized** versions of D_t 's

Unifying the Two Cases

[with Collins & Singer]

- two distinct cases:
 - weak learning assumption holds
 - $\mathcal{P} = \emptyset$
 - dynamics unclear
 - weak learning assumption does **not** hold
 - $\mathcal{P} \neq \emptyset$
 - can prove convergence of D_t 's
- to **unify**: work instead with **un**normalized versions of D_t 's
 - standard AdaBoost: $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{\text{normalization}}$

Unifying the Two Cases

[with Collins & Singer]

- two distinct cases:
 - weak learning assumption holds
 - $\mathcal{P} = \emptyset$
 - dynamics unclear
 - weak learning assumption does **not** hold
 - $\mathcal{P} \neq \emptyset$
 - can prove convergence of D_t 's
- to **unify**: work instead with **unnormalized** versions of D_t 's
 - standard AdaBoost: $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{\text{normalization}}$
 - instead:

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

$$D_{t+1}(i) = \frac{d_{t+1}(i)}{\text{normalization}}$$

Unifying the Two Cases

[with Collins & Singer]

- two distinct cases:
 - weak learning assumption holds
 - $\mathcal{P} = \emptyset$
 - dynamics unclear
 - weak learning assumption does **not** hold
 - $\mathcal{P} \neq \emptyset$
 - can prove convergence of D_t 's
- to **unify**: work instead with **unnormalized** versions of D_t 's
 - standard AdaBoost: $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{\text{normalization}}$
 - instead:

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

$$D_{t+1}(i) = \frac{d_{t+1}(i)}{\text{normalization}}$$

- algorithm is **unchanged**

Reformulating AdaBoost as Iterative Projection

- points = nonnegative vectors \mathbf{d}_t
- distance = unnormalized relative entropy:

$$\text{RE}(\mathbf{p} \parallel \mathbf{q}) = \sum_i \left[p(i) \ln \left(\frac{p(i)}{q(i)} \right) + q(i) - p(i) \right]$$

- initial point $\mathbf{x}_0 = \mathbf{1}$ (all 1's vector)
- hyperplanes defined by weak classifiers g_j :

$$\sum_i d(i) y_i g_j(x_i) = 0$$

Reformulating AdaBoost as Iterative Projection

- points = nonnegative vectors \mathbf{d}_t
- distance = unnormalized relative entropy:

$$\text{RE}(\mathbf{p} \parallel \mathbf{q}) = \sum_i \left[p(i) \ln \left(\frac{p(i)}{q(i)} \right) + q(i) - p(i) \right]$$

- initial point $\mathbf{x}_0 = \mathbf{1}$ (all 1's vector)
- hyperplanes defined by weak classifiers g_j :

$$\sum_i d(i) y_i g_j(x_i) = 0$$

- resulting iterative-projection algorithm is again equivalent to AdaBoost

Reformulated Optimization Problem

- optimization problem:

$$\min_{\mathbf{d} \in \mathcal{P}} \text{RE}(\mathbf{d} \parallel \mathbf{1})$$

- where

$$\mathcal{P} = \left\{ \mathbf{d} : \sum_i d(i) y_i g_j(x_i) = 0 \quad \forall j \right\}$$

- note: feasible set \mathcal{P} never empty (since $\mathbf{0} \in \mathcal{P}$)

Exponential Loss as Entropy Optimization

exponential loss:

$$\inf_{\lambda} \sum_i \exp \left(-y_i \sum_j \lambda_j g_j(x_i) \right)$$

Exponential Loss as Entropy Optimization

- all vectors \mathbf{d}_t created by AdaBoost have form:

$$d(i) = \exp \left(-y_i \sum_j \lambda_j g_j(x_i) \right)$$

- let $\mathcal{Q} = \{ \text{all vectors } \mathbf{d} \text{ of this form} \}$
- can rewrite exponential loss:

$$\inf_{\lambda} \sum_i \exp \left(-y_i \sum_j \lambda_j g_j(x_i) \right) = \inf_{\mathbf{d} \in \mathcal{Q}} \sum_i d(i)$$

Exponential Loss as Entropy Optimization

- all vectors \mathbf{d}_t created by AdaBoost have form:

$$d(i) = \exp \left(-y_i \sum_j \lambda_j g_j(x_i) \right)$$

- let $\mathcal{Q} = \{ \text{all vectors } \mathbf{d} \text{ of this form} \}$
- can rewrite exponential loss:

$$\begin{aligned} \inf_{\lambda} \sum_i \exp \left(-y_i \sum_j \lambda_j g_j(x_i) \right) &= \inf_{\mathbf{d} \in \mathcal{Q}} \sum_i d(i) \\ &= \min_{\mathbf{d} \in \overline{\mathcal{Q}}} \sum_i d(i) \end{aligned}$$

- $\overline{\mathcal{Q}}$ = closure of \mathcal{Q}

Exponential Loss as Entropy Optimization

- all vectors \mathbf{d}_t created by AdaBoost have form:

$$d(i) = \exp \left(-y_i \sum_j \lambda_j g_j(x_i) \right)$$

- let $\mathcal{Q} = \{ \text{all vectors } \mathbf{d} \text{ of this form} \}$
- can rewrite exponential loss:

$$\begin{aligned} \inf_{\lambda} \sum_i \exp \left(-y_i \sum_j \lambda_j g_j(x_i) \right) &= \inf_{\mathbf{d} \in \mathcal{Q}} \sum_i d(i) \\ &= \min_{\mathbf{d} \in \overline{\mathcal{Q}}} \sum_i d(i) \\ &= \min_{\mathbf{d} \in \overline{\mathcal{Q}}} \text{RE}(\mathbf{0} \parallel \mathbf{d}) \end{aligned}$$

- $\overline{\mathcal{Q}}$ = closure of \mathcal{Q}

Duality

[Della Pietra, Della Pietra & Lafferty]

- presented two optimization problems:
 - $\min_{\mathbf{d} \in \mathcal{P}} \text{RE}(\mathbf{d} \parallel \mathbf{1})$
 - $\min_{\mathbf{d} \in \mathcal{Q}} \text{RE}(\mathbf{0} \parallel \mathbf{d})$
- which is AdaBoost solving?

Duality

[Della Pietra, Della Pietra & Lafferty]

- presented two optimization problems:
 - $\min_{\mathbf{d} \in \mathcal{P}} \text{RE}(\mathbf{d} \parallel \mathbf{1})$
 - $\min_{\mathbf{d} \in \mathcal{Q}} \text{RE}(\mathbf{0} \parallel \mathbf{d})$
- which is AdaBoost solving? Both!
- problems have **same** solution

Duality

[Della Pietra, Della Pietra & Lafferty]

- presented two optimization problems:
 - $\min_{\mathbf{d} \in \mathcal{P}} \text{RE}(\mathbf{d} \parallel \mathbf{1})$
 - $\min_{\mathbf{d} \in \overline{\mathcal{Q}}} \text{RE}(\mathbf{0} \parallel \mathbf{d})$
- which is AdaBoost solving? Both!
- problems have **same** solution
- moreover: solution given by **unique** point in $\mathcal{P} \cap \overline{\mathcal{Q}}$

Duality

[Della Pietra, Della Pietra & Lafferty]

- presented two optimization problems:
 - $\min_{\mathbf{d} \in \mathcal{P}} \text{RE}(\mathbf{d} \parallel \mathbf{1})$
 - $\min_{\mathbf{d} \in \overline{\mathcal{Q}}} \text{RE}(\mathbf{0} \parallel \mathbf{d})$
- which is AdaBoost solving? Both!
- problems have **same** solution
- moreover: solution given by **unique** point in $\mathcal{P} \cap \overline{\mathcal{Q}}$
- problems are **convex duals** of each other

Convergence of AdaBoost

- can use to prove AdaBoost **converges** to common solution of both problems:

Convergence of AdaBoost

- can use to prove AdaBoost **converges** to common solution of both problems:
 - can argue that $\mathbf{d}^* = \lim \mathbf{d}_t$ is in \mathcal{P}

Convergence of AdaBoost

- can use to prove AdaBoost **converges** to common solution of both problems:
 - can argue that $\mathbf{d}^* = \lim \mathbf{d}_t$ is in \mathcal{P}
 - vectors \mathbf{d}_t are in \mathcal{Q} always $\Rightarrow \mathbf{d}^* \in \overline{\mathcal{Q}}$

Convergence of AdaBoost

- can use to prove AdaBoost **converges** to common solution of both problems:
 - can argue that $\mathbf{d}^* = \lim \mathbf{d}_t$ is in \mathcal{P}
 - vectors \mathbf{d}_t are in \mathcal{Q} always $\Rightarrow \mathbf{d}^* \in \overline{\mathcal{Q}}$
- $\therefore \mathbf{d}^* \in \mathcal{P} \cap \overline{\mathcal{Q}}$

Convergence of AdaBoost

- can use to prove AdaBoost **converges** to common solution of both problems:
 - can argue that $\mathbf{d}^* = \lim \mathbf{d}_t$ is in \mathcal{P}
 - vectors \mathbf{d}_t are in \mathcal{Q} always $\Rightarrow \mathbf{d}^* \in \overline{\mathcal{Q}}$
 - $\therefore \mathbf{d}^* \in \mathcal{P} \cap \overline{\mathcal{Q}}$
 - $\therefore \mathbf{d}^*$ solves both optimization problems

Convergence of AdaBoost

- can use to prove AdaBoost **converges** to common solution of both problems:
 - can argue that $\mathbf{d}^* = \lim \mathbf{d}_t$ is in \mathcal{P}
 - vectors \mathbf{d}_t are in \mathcal{Q} always $\Rightarrow \mathbf{d}^* \in \overline{\mathcal{Q}}$
 - $\therefore \mathbf{d}^* \in \mathcal{P} \cap \overline{\mathcal{Q}}$
 - $\therefore \mathbf{d}^*$ solves both optimization problems
- so:
 - AdaBoost **minimizes** exponential loss
 - exactly **characterizes** limit of unnormalized “distributions”
 - likewise for normalized distributions when weak learning assumption does not hold

Convergence of AdaBoost

- can use to prove AdaBoost **converges** to common solution of both problems:
 - can argue that $\mathbf{d}^* = \lim \mathbf{d}_t$ is in \mathcal{P}
 - vectors \mathbf{d}_t are in \mathcal{Q} always $\Rightarrow \mathbf{d}^* \in \overline{\mathcal{Q}}$
 - $\therefore \mathbf{d}^* \in \mathcal{P} \cap \overline{\mathcal{Q}}$
 - $\therefore \mathbf{d}^*$ solves both optimization problems
 - so:
 - AdaBoost **minimizes** exponential loss
 - exactly **characterizes** limit of unnormalized “distributions”
 - likewise for normalized distributions when weak learning assumption does not hold
 - also, provides additional link to **logistic regression**
 - only need slight change in optimization problem
- [with Collins & Singer; LeBannon & Lafferty]

Experiments, Applications and Extensions

- basic experiments
- multiclass classification
- confidence-rated predictions
- text categorization /
spoken-dialogue systems
- incorporating prior knowledge
- active learning
- face detection

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

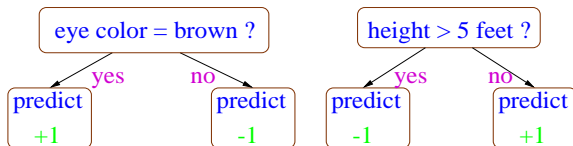
Caveats

- 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 → overfitting
- empirically, AdaBoost seems especially susceptible to uniform noise

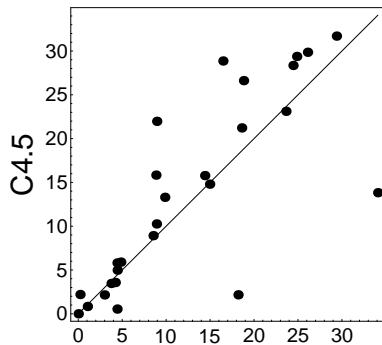
UCI Experiments

[with Freund]

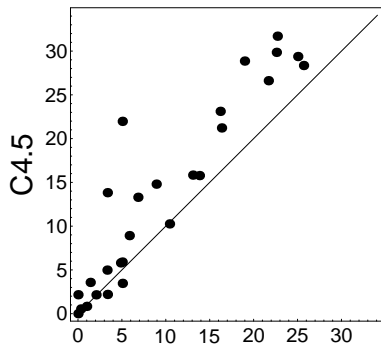
- tested AdaBoost on UCI benchmarks
- used:
 - C4.5 (Quinlan's decision tree algorithm)
 - “decision stumps”: very simple rules of thumb that test on single attributes



UCI Results



boosting Stumps



boosting C4.5

Multiclass Problems

[with Freund]

- say $y \in Y = \{1, \dots, k\}$
- direct approach (AdaBoost.M1):

$$h_t : X \rightarrow Y$$

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

$$H_{\text{final}}(x) = \arg \max_{y \in Y} \sum_{t: h_t(x)=y} \alpha_t$$

Multiclass Problems

[with Freund]

- say $y \in Y = \{1, \dots, k\}$
- direct approach (AdaBoost.M1):

$$h_t : X \rightarrow Y$$

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

$$H_{\text{final}}(x) = \arg \max_{y \in Y} \sum_{t: h_t(x)=y} \alpha_t$$

- can prove same bound on error if $\forall t : \epsilon_t \leq 1/2$
 - in practice, not usually a problem for “strong” weak learners (e.g., C4.5)
 - significant problem for “weak” weak learners (e.g., decision stumps)
- instead, reduce to binary

Reducing Multiclass to Binary

[with Singer]

- say possible labels are $\{a, b, c, d, e\}$
- each training example replaced by five $\{-1, +1\}$ -labeled examples:

$$x, c \rightarrow \begin{cases} (x, a), & -1 \\ (x, b), & -1 \\ (x, c), & +1 \\ (x, d), & -1 \\ (x, e), & -1 \end{cases}$$

- predict with label receiving most (weighted) votes

AdaBoost.MH

- can prove:

$$\text{training error}(H_{\text{final}}) \leq \frac{k}{2} \cdot \prod Z_t$$

- reflects fact that small number of errors in binary predictors can cause overall prediction to be incorrect
- extends immediately to **multi-label** case (more than one correct label per example)

Using Output Codes

[with Allwein & Singer][Dietterich & Bakiri]

- alternative: choose “code word” for each label

	π_1	π_2	π_3	π_4
a	-	+	-	+
b	-	+	+	-
c	+	-	-	+
d	+	-	+	+
e	-	+	-	-

- each training example mapped to one example per column

$$x, c \rightarrow \begin{cases} (x, \pi_1), & +1 \\ (x, \pi_2), & -1 \\ (x, \pi_3), & -1 \\ (x, \pi_4), & +1 \end{cases}$$

- to classify new example x :
 - evaluate classifier on $(x, \pi_1), \dots, (x, \pi_4)$
 - choose label “most consistent” with results

Output Codes (cont.)

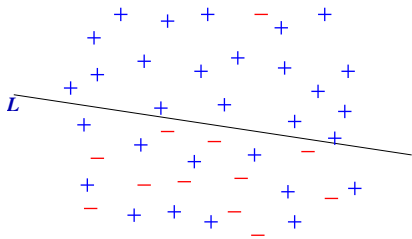
- training error bounds **independent** of # of classes
- overall prediction robust to large number of errors in binary predictors
- **but:** binary problems may be harder

Ranking Problems

[with Freund, Iyer & Singer]

- other problems can also be handled by reducing to binary
- e.g.: want to learn to **rank** objects (say, movies) from examples
- can **reduce** to multiple **binary** questions of form:
“is or is not object A preferred to object B?”
- now apply (binary) AdaBoost

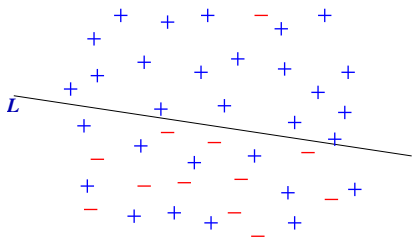
“Hard” Predictions Can Slow Learning



- ideally, want weak classifier that says:

$$h(x) = \begin{cases} +1 & \text{if } x \text{ above } L \\ \text{"don't know"} & \text{else} \end{cases}$$

“Hard” Predictions Can Slow Learning



- ideally, want weak classifier that says:

$$h(x) = \begin{cases} +1 & \text{if } x \text{ above } L \\ \text{"don't know"} & \text{else} \end{cases}$$

- **problem:** cannot express using “hard” predictions
- if must predict ± 1 below L , will introduce many “bad” predictions
 - need to “clean up” on later rounds
- dramatically increases time to convergence

Confidence-rated Predictions

[with Singer]

- useful to allow weak classifiers to assign **confidences** to predictions
- formally, allow $h_t : X \rightarrow \mathbb{R}$

$$\text{sign}(h_t(x)) = \text{prediction}$$

$$|h_t(x)| = \text{"confidence"}$$

Confidence-rated Predictions

[with Singer]

- useful to allow weak classifiers to assign **confidences** to predictions
- formally, allow $h_t : X \rightarrow \mathbb{R}$

$$\begin{aligned}\text{sign}(h_t(x)) &= \text{prediction} \\ |h_t(x)| &= \text{"confidence"}$$

- use identical update:

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

and identical rule for combining weak classifiers

- **question**: how to choose α_t and h_t on each round

Confidence-rated Predictions (cont.)

- saw earlier:

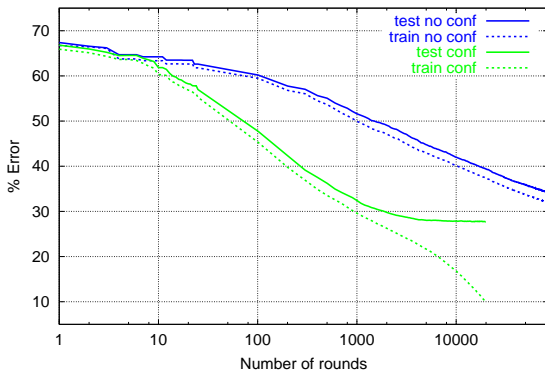
$$\text{training error}(H_{\text{final}}) \leq \prod_t Z_t = \frac{1}{m} \sum_i \exp \left(-y_i \sum_t \alpha_t h_t(x_i) \right)$$

- therefore, on each round t , should choose $\alpha_t h_t$ to minimize:

$$Z_t = \sum_i D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

- in many cases (e.g., decision stumps), best confidence-rated weak classifier has simple form that can be found efficiently

Confidence-rated Predictions Help a Lot



% error	round first reached		speedup
	conf.	no conf.	
40	268	16,938	63.2
35	598	65,292	109.2
30	1,888	>80,000	—

Application: Boosting for Text Categorization

[with Singer]

- **weak classifiers**: very simple weak classifiers that test on simple patterns, namely, (sparse) n -grams
 - find parameter α_t and rule h_t of given form which minimize Z_t
 - use efficiently implemented exhaustive search
- “How may I help you” data:
 - 7844 training examples
 - 1000 test examples
 - categories: AreaCode, AttService, BillingCredit, CallingCard, Collect, Competitor, DialForMe, Directory, HowToDial, PersonToPerson, Rate, ThirdNumber, Time, TimeCharge, Other.

Finding Outliers

examples with most weight are often **outliers** (mis-labeled and/or ambiguous)

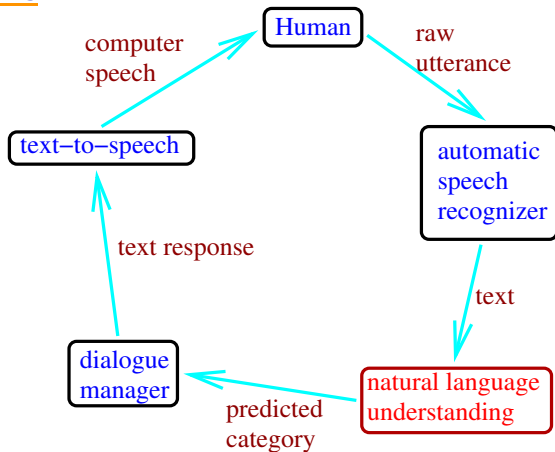
- I'm trying to make a credit card call (**Collect**)
- hello (**Rate**)
- yes I'd like to make a long distance collect call please (**CallingCard**)
- calling card please (**Collect**)
- yeah I'd like to use my calling card number (**Collect**)
- can I get a collect call (**CallingCard**)
- yes I would like to make a long distant telephone call and have the charges billed to another number (**CallingCard DialForMe**)
- yeah I can not stand it this morning I did oversea call is so bad (**BillingCredit**)
- yeah special offers going on for long distance (**AttService Rate**)
- mister allen please william allen (**PersonToPerson**)
- yes ma'am I I'm trying to make a long distance call to a non dialable point in san miguel philippines (**AttService Other**)

Application: Human-computer Spoken Dialogue

[with Rahim, Di Fabrizio, Dutton, Gupta, Hollister & Riccardi]

- **application:** automatic “store front” or “help desk” for AT&T Labs’ Natural Voices business
- caller can request demo, pricing information, technical support, sales agent, etc.
- interactive dialogue

How It Works



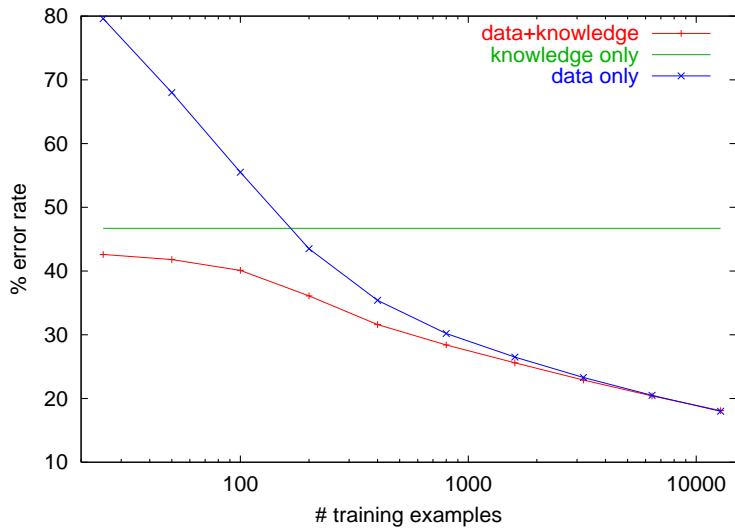
- NLU's job: classify caller utterances into 24 categories (demo, sales rep, pricing info, yes, no, etc.)
- weak classifiers: test for presence of word or phrase

Need for Prior, Human Knowledge

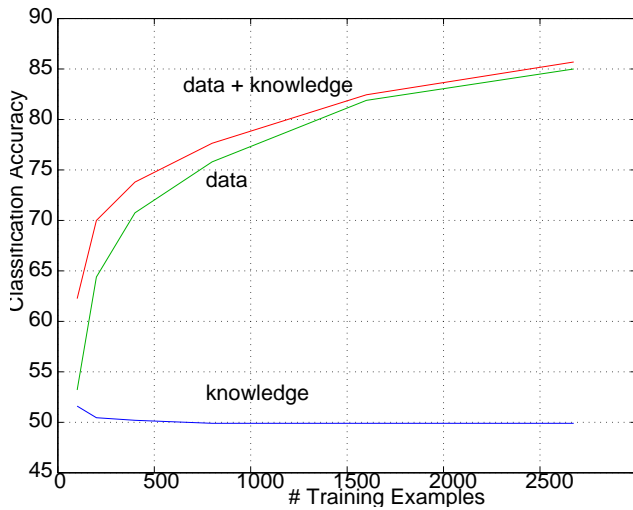
[with Rochery, Rahim & Gupta]

- building NLU: standard text categorization problem
- need **lots of data**, but for cheap, **rapid** deployment, can't wait for it
- **bootstrapping** problem:
 - need labeled data to deploy
 - need to deploy to get labeled data
- **idea**: use human knowledge to compensate for insufficient data
 - modify loss function to balance **fit to data** against **fit to prior model**

Results: AP-Titles



Results: Helpdesk



Problem: Labels are Expensive

- for spoken-dialogue task
 - getting examples is cheap
 - getting labels is expensive
 - must be annotated by humans
- how to reduce number of labels needed?

Active Learning

[with Tur & Hakkani-Tür]

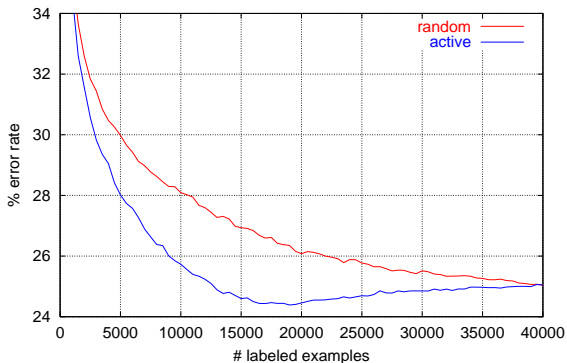
- idea:
 - use **selective sampling** to choose which examples to label
 - focus on **least confident** examples [Lewis & Gale]
- for boosting, use (absolute) margin as natural confidence measure

[Abe & Mamitsuka]

Labeling Scheme

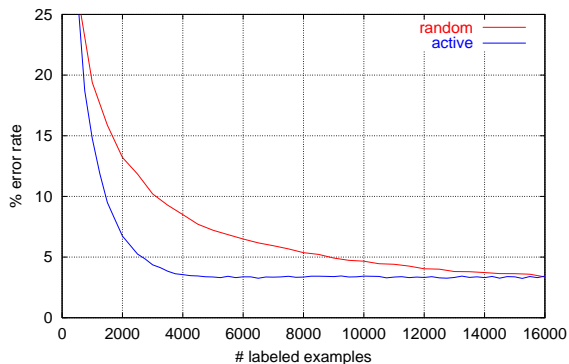
- start with pool of **unlabeled** examples
- choose (say) 500 examples at random for labeling
- run boosting on **all labeled** examples
 - get combined classifier F
- pick (say) 250 additional examples from pool for labeling
 - choose examples with minimum $|F(x)|$
(proportional to absolute margin)
- repeat

Results: How-May-I-Help-You?



% error	first reached		% label savings
	random	active	
28	11,000	5,500	50
26	22,000	9,500	57
25	40,000	13,000	68

Results: Letter

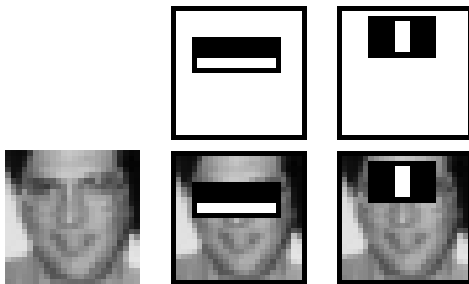


% error	first reached		% label savings
	random	active	
10	3,500	1,500	57
5	9,000	2,750	69
4	13,000	3,500	73

Application: Detecting Faces

[Viola & Jones]

- **problem**: find **faces** in photograph or movie
- **weak classifiers**: detect light/dark rectangles in image



- many clever tricks to make extremely fast and accurate

Conclusions

- from different perspectives, AdaBoost can be interpreted as:
 - a method for **boosting** the accuracy of a weak learner
 - a procedure for **maximizing margins**
 - an algorithm for playing **repeated games**
 - a numerical method for **minimizing exponential loss**
 - an **iterative projection** algorithm based on an information-theoretic geometry

Conclusions

- from different perspectives, AdaBoost can be interpreted as:
 - a method for **boosting** the accuracy of a weak learner
 - a procedure for **maximizing margins**
 - an algorithm for playing **repeated games**
 - a numerical method for **minimizing exponential loss**
 - an **iterative projection** algorithm based on an information-theoretic geometry
- none is entirely satisfactory by itself, but each useful in its own way
- taken together, create rich theoretical understanding
 - connect boosting to other learning problems and techniques
 - provide foundation for versatile set of methods with many extensions, variations and applications

References

- Ron Meir and Gunnar Rätsch.
An Introduction to Boosting and Leveraging.
In *Advanced Lectures on Machine Learning (LNAI2600)*, 2003.
<http://www.boosting.org/papers/MeiRae03.pdf>
- Robert E. Schapire.
The boosting approach to machine learning: An overview.
In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.
<http://www.cs.princeton.edu/~schapire/boost.html>

Coming soon (hopefully):

- Robert E. Schapire and Yoav Freund.
Boosting: Foundations and Algorithms.
MIT Press, 2010(?).