Boosting

Introduction to Statistical Machine Learning

11 Aug 2014

Note: Some slides are adapted from R. Schapire's Tutorial

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)
  - 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
- 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, Màrquez & 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 Fabbrizio, 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]

⋮

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

A Formal Description of Boosting

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

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

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

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

    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

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

AdaBoost

[Schapire & Freund]

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

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

where Z_t = normalization constant

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

AdaBoost

[Schapire & Freund]

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

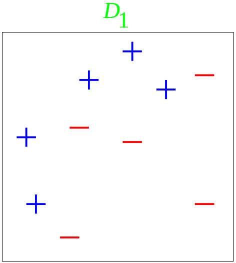
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$
$$= \frac{D_t(i)}{Z_t} \exp(-\alpha_t \, y_i \, h_t(x_i))$$

where $Z_t =$ normalization constant

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

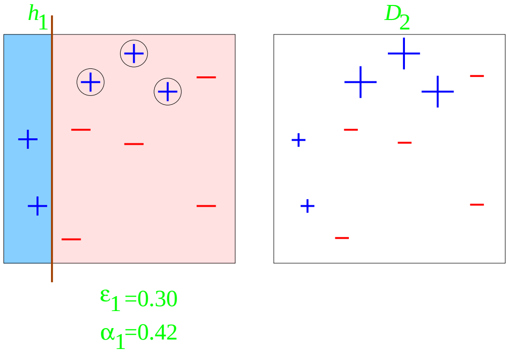
- final classifier:
  - $H_{\mathrm{final}}(x) = \mathrm{sign}\left(\sum_t \alpha_t h_t(x) \right)$

Toy Example

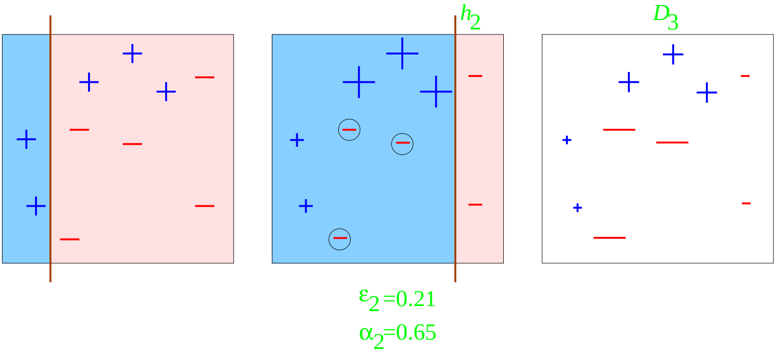


weak classifiers = vertical or horizontal half-planes

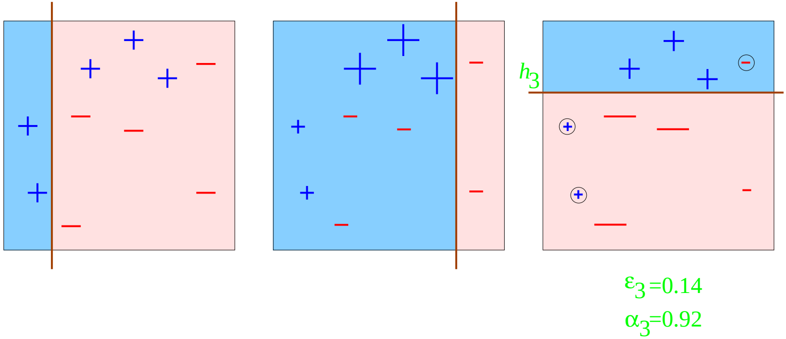
Round 1



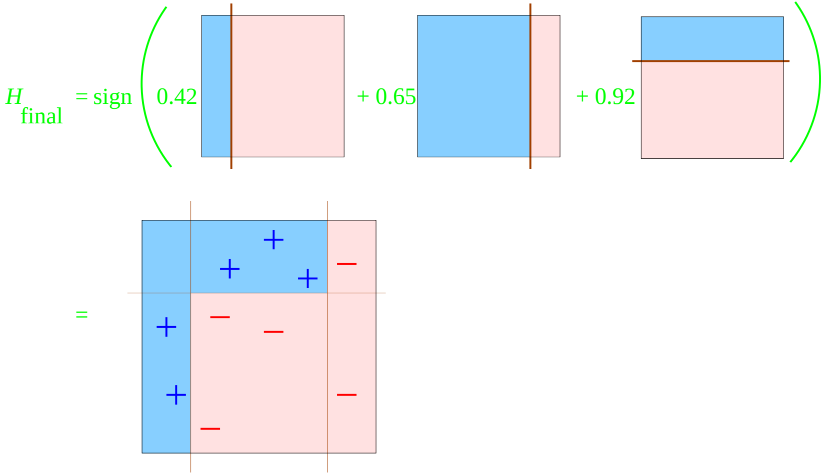
Round 2



Round 3



Final Classifier



Analyzing the training error

- Theorem:
  - write ϵ_t as $1/2 - \gamma_t$

Analyzing the training error

- Theorem:
  - write ϵ_t as $1/2 - \gamma_t$
  - 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

- Theorem:
  - write ϵ_t as $1/2 - \gamma_t$
  - then

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

- so: if $\forall t: \ \gamma_t \ge \gamma > 0$
  then $\text{training error}(H_{\text{final}}) \le e^{-2\gamma^2 T}$
- 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:

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

Proof (cont.)

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

Proof (cont.)

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

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

Proof (cont.)

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

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

Proof (cont.)

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

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

Proof (cont.)

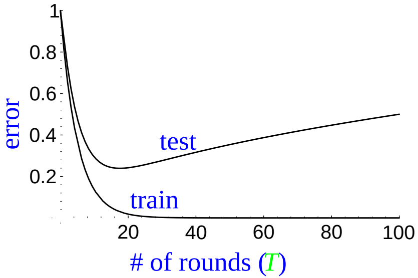
- *Step 3*: $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$

Proof (cont.)

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

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

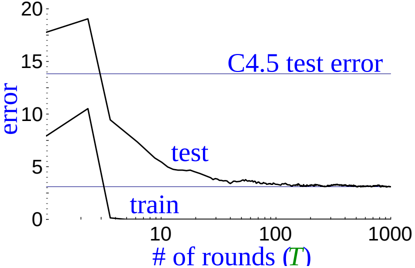
How Will Test Error Behave? (A First Guess)



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

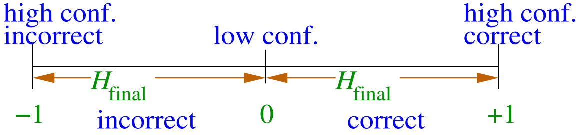
- key idea:
  - training error only measures whether classifications are right or wrong
  - should also consider confidence of classifications

A Better Story: The Margins Explanation

- 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

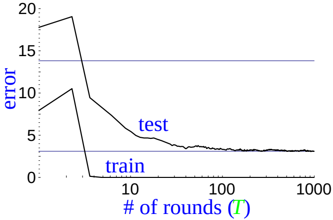
A Better Story: The Margins Explanation

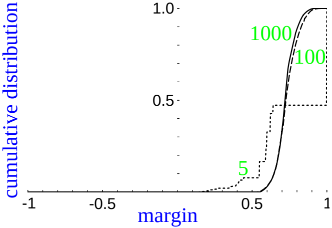
- key idea:
  - training error only measures whether classifications are right or wrong
  - should also consider confidence of classifications
- recall: H_{final} is weighted majority vote of weak classifiers
- measure confidence by margin = strength of the vote
  = (fraction voting correctly) − (fraction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

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)
  - 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)
  - 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, $\forall \theta > 0$:

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

($\hat{\Pr}[\,] =$ empirical probability)

  - bound depends on
    - $m = \#$ training examples
    - $d =$ "complexity" of weak classifiers
    - entire distribution of margins of training examples
- $\hat{\Pr}[\text{margin} \leq \theta] \to 0$ 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

AdaBoost and Exponential Loss

- many (most?) learning algorithms minimize a "loss" function
  - e.g. least squares regression
- training error proof shows AdaBoost actually minimizes

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

where $f(x) = \sum_t \alpha_t h_t(x)$

- on each round, AdaBoost greedily chooses α_t and h_t to minimize loss

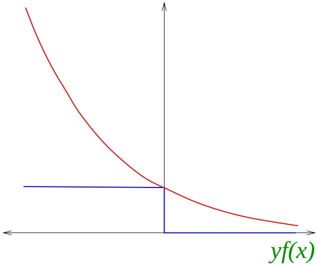
AdaBoost and Exponential Loss

- many (most?) learning algorithms minimize a "loss" function
  - e.g. least squares regression
- training error proof shows AdaBoost actually minimizes

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

where $f(x) = \sum_t \alpha_t h_t(x)$

- on each round, AdaBoost greedily chooses α_t and h_t to minimize loss
- exponential loss is an upper bound on 0-1 (classification) loss
- AdaBoost provably minimizes exponential loss



Coordinate Descent

[Breiman]

- $\{g_1, \ldots, g_N\}$ = space of all weak classifiers
- want to find $\lambda_1, \ldots, \lambda_N$ to minimize

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

Coordinate Descent

[Breiman]

- $\{g_1, \ldots, g_N\}$ = space of all weak classifiers
- want to find $\lambda_1, \ldots, \lambda_N$ to minimize

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

- 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

[Friedman][Mason et al.]

- want to minimize

$$L(f) = L(f(x_1), \ldots, f(x_m)) = \sum_i \exp(-y_i f(x_i))$$

Functional Gradient Descent

[Friedman][Mason et al.]

- want to minimize

$$L(f) = L(f(x_1), \ldots, f(x_m)) = \sum_i \exp(-y_i f(x_i))$$

- 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

[Friedman][Mason et al.]

- want to minimize

$$L(f) = L(f(x_1), \ldots, f(x_m)) = \sum_i \exp(-y_i f(x_i))$$

- 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

[Friedman][Mason et al.]

- want to minimize

$$L(f) = L(f(x_1), \ldots, f(x_m)) = \sum_i \exp(-y_i f(x_i))$$

- 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

Benefits of Model Fitting View

- immediate generalization to other loss functions
  - 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

Benefits of Model Fitting View

- immediate generalization to other loss functions
  - 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
- caveat: wrong to view AdaBoost as just an algorithm for minimizing exponential loss
  - other algorithms for minimizing same loss will (provably) give very poor performance
  - thus, this loss function cannot explain why AdaBoost "works"

Other Ways to Think about AdaBoost

- dynamical systems
- statistical consistency
- maximum entropy

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

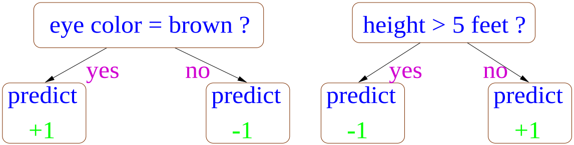
- fast
- simple and easy to program
- no parameters to tune (except T)
- flexible — can combine with any learning algorithm
- no prior knowledge needed about weak learner
- provably effective, provided can consistently find rough rules of thumb
  - $\rightarrow$ shift in mind set — goal now is merely 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

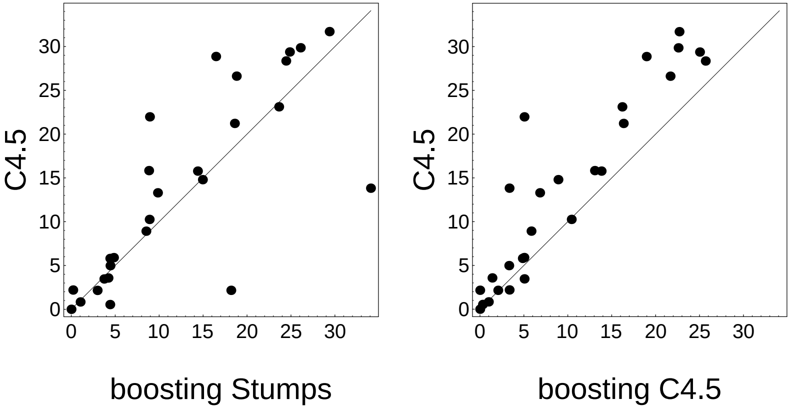
- performance of AdaBoost depends on data and weak learner
- consistent with theory, AdaBoost can fail if
  - weak classifiers too complex
    - $\longrightarrow$ overfitting
  - weak classifiers too weak ($\gamma_t \rightarrow 0$ too quickly)
    - $\longrightarrow$ underfitting
    - $\longrightarrow$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

UCI Experiments

- 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



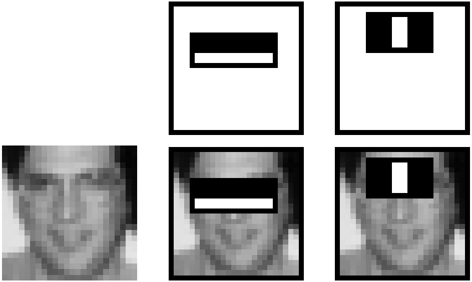
UCI Results


C4.5
0
5
10
15
20
25
30
boosting Stumps
C4.5
0
5
10
15
20
25
30
boosting C4.5

Application: Detecting Faces

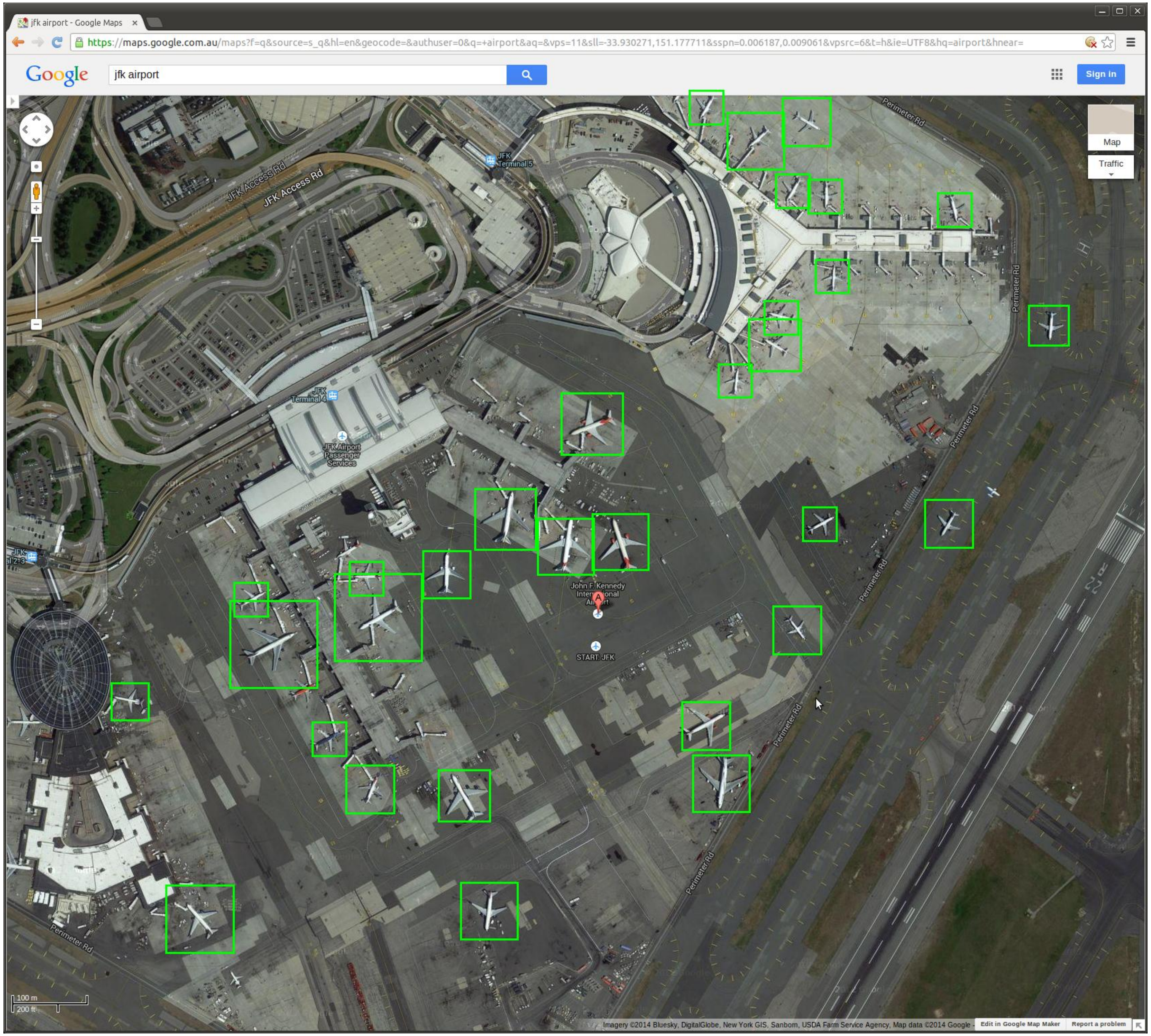
[Viola & Jones]

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



- many clever tricks to make extremely fast and accurate

Aircraft detection



Pedestrian detection

Railway sign detection

Conclusions

- boosting is a practical tool for classification and other learning problems
  - grounded in rich theory
  - performs well experimentally
  - often (but not always!) resistant to overfitting
  - many applications and extensions
- many ways to think about boosting
  - none is entirely satisfactory by itself, but each useful in its own way
  - considerable room for further theoretical and experimental work

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