Theory and Applications of Boosting

Yoav Freund
UCSD

Many slides from Rob Schapire
Plan

- **Day 1: Basics**
  - Boosting,
  - Adaboost,
  - Margins theory.
  - Confidence-rated boosting

- **Day 2: Applications**
  - ADTrees
  - JBoost
  - Viola and Jones
  - Active Learning and Pedestrian Detection
  - Genome Wide association studies
  - Online boosting and tracking.

- **Day 3: Advanced Topics**
  - Boosting and repeated matrix games
  - Boosting and Loss minimization.
  - Drifting games and Boost By Majority.
  - Brownboost and Boosting with High Noise.
Example: “How May I Help You?”

- **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
Key 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** = 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\%
Some History

• How it all began ...
Strong and Weak Learnability

- boosting’s roots are in “PAC” learning model [Valiant ’84]
- 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?
If Boosting Possible, Then...

- can use (fairly) wild guesses to produce highly accurate predictions
- if can learn “part way” then can learn “all the way”
- should be able to improve any learning algorithm
- for any learning problem:
  - either can always learn with nearly perfect accuracy
  - or there exist cases where cannot learn even slightly better than random guessing
First 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
- [Freund & Schapire '95]:
  - introduced “AdaBoost” algorithm
  - strong practical advantages over previous boosting algorithms
Basic Algorithm and Core Theory

- introduction to AdaBoost
- analysis of training error
- analysis of test error and the margins theory
- experiments and applications
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") \(h_t: X \rightarrow \{-1, +1\}\) with small error \(\epsilon_t\) on \(D_t\):
    \[
    \epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]
    \]
- output final classifier \(H_{final}\)
AdaBoost

- constructing $D_t$:
  - $D_1(i) = 1/m$
  - 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 $Z_t = \text{normalization factor}$

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

- final classifier:

$$H_{\text{final}}(x) = \text{sign} \left( \sum_t \alpha_t h_t(x) \right)$$
Toy Example

\[ D_1 \]

\[
\begin{array}{cccc}
+ & + & - \\
+ & - & - \\
+ & - & - \\
+ & - & - \\
\end{array}
\]

weak classifiers = vertical or horizontal half-planes
Round 1

$h_1$

$e_1 = 0.30$

$\alpha_1 = 0.42$

$D_2$
Round 2

\[ \varepsilon_2 = 0.21 \]
\[ \alpha_2 = 0.65 \]
Round 3

\[ h_3 \]

\[ \varepsilon_3 = 0.14 \]
\[ \alpha_3 = 0.92 \]
Final Classifier

\[ H_{\text{final}} = \text{sign} \begin{pmatrix} 0.42 \\ +0.65 \\ +0.92 \end{pmatrix} \]
http://cseweb.ucsd.edu/~yfreund/adaboost/index.html
Basic Algorithm and Core Theory

- introduction to AdaBoost
- analysis of training error
- analysis of test error and the margins theory
- experiments and applications
Analyzing the Training Error

- **Theorem:**
  - write $\epsilon_t$ as $\frac{1}{2} - \gamma_t$  \quad [\gamma_t = \text{“edge”}]  
  - then

  \[
  \text{training error}(H_{\text{final}}) \leq \prod_t \left[ 2\sqrt{\epsilon_t(1-\epsilon_t)} \right] 
  \]

  \[
  = \prod_t \sqrt{1 - 4\gamma^2_t} 
  \]

  \[
  \leq \exp \left( -2 \sum_t \gamma^2_t \right) 
  \]

- 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 $\gamma$ or $T$ a priori
  - can exploit $\gamma_t \gg \gamma$

[Freund & Schapire 96]
Proof

**Scoring function**

- let $F(x) = \sum_t \alpha_t h_t(x) \Rightarrow H_{\text{final}}(x) = \text{sign}(F(x))$

- *Step 1:* unwrapping recurrence:

\[
D_{\text{final}}(i) = \frac{1}{m} \exp \left( \frac{-y_i \sum_t \alpha_t h_t(x_i)}{\prod_t Z_t} \right)
\]

\[
= \frac{1}{m} \exp \left( -y_i F(x_i) \right) \prod_t Z_t
\]
Proof (cont.)

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

- Proof:

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

$$= \frac{1}{m} \sum_i \left\{ \begin{array}{ll} 1 & \text{if } y_i F(x_i) \leq 0 \\ 0 & \text{else} \end{array} \right.$$ 

$$\leq \frac{1}{m} \sum_i \exp(-y_i F(x_i))$$ 

$$= \sum_i D_{final}(i) \prod_t Z_t$$ 

$$= \prod_t Z_t$$
Proof (cont.)

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

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

\[
= \sum_{i: y_i \neq h_t(x_i)} D_t(i) e^{\alpha_t} + \sum_{i: y_i = h_t(x_i)} D_t(i) e^{-\alpha_t}
\]

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

\[
= 2 \sqrt{\epsilon_t(1 - \epsilon_t)}
\]
Basic Algorithm and Core Theory

- introduction to AdaBoost
- analysis of training error
- analysis of test error and the margins theory
- experiments and applications
How Will Test Error Behave? (A First Guess)

expect:

- training error to continue to drop (or reach zero)
- test error to increase when $H_{\text{final}}$ becomes “too complex”
  - “Occam’s razor”
  - overfitting
    - hard to know when to stop training
Technically...

- with high probability:

\[
\text{generalization error} \leq \text{training error} + \tilde{O}\left(\sqrt{\frac{dT}{m}}\right)
\]

- bound depends on
  - \(m = \# \text{ training examples}\)
  - \(d = \text{“complexity” of weak classifiers}\)
  - \(T = \# \text{ rounds}\)

- generalization error = \(E[\text{test error}]\)
- predicts overfitting
Overfitting Can Happen

(boostering “stumps” on heart-disease dataset)

- but often doesn’t…

Monday, July 16, 2012
Actual Typical Run

- 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!

<table>
<thead>
<tr>
<th># rounds</th>
<th>5</th>
<th>100</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train error</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Test error</td>
<td>8.4</td>
<td>3.3</td>
<td>3.1</td>
</tr>
</tbody>
</table>

- Occam’s razor wrongly predicts “simpler” rule is better

(Boosting C4.5 on “letter” dataset)
key idea:
- training error only measures whether classifications are right or wrong
- should also consider confidence of classifications

recall: \( H_{\text{final}} \) is weighted majority vote of weak classifiers

measure confidence by \( \text{margin} = \text{strength of the vote} \)
\[
= (\text{weighted fraction voting correctly}) - (\text{weighted fraction voting incorrectly})
\]

\( H_{\text{final}} \)

\[
\begin{array}{c|c|c}
\text{high conf.} & \text{low conf.} & \text{high conf.} \\
\text{incorrect} & \text{correct} & \text{correct} \\
-1 & 0 & +1
\end{array}
\]
Empirical Evidence: The Margin Distribution

• margin distribution
  \[ = \text{cumulative distribution of margins of training examples} \]

<table>
<thead>
<tr>
<th># rounds</th>
<th>5</th>
<th>100</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>train error</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>test error</td>
<td>8.4</td>
<td>3.3</td>
<td>3.1</td>
</tr>
<tr>
<td>% margins ( \leq 0.5 )</td>
<td>7.7</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>minimum margin</td>
<td>0.14</td>
<td>0.52</td>
<td>0.55</td>
</tr>
</tbody>
</table>
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, $\forall \theta > 0$:

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

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

- bound depends on
  - $m = \# \text{ training examples}$
  - $d = \text{“complexity” of weak classifiers}$
  - entire distribution of margins of training examples

- $\hat{Pr}[\text{margin} \leq \theta] \rightarrow 0$ exponentially fast (in $T$)
  if $\epsilon_t < \frac{1}{2} - \theta$ ($\forall t$)

- so: if weak learning assumption holds, then all examples will quickly have “large” margins
Consequences of Margins Theory

• predicts good generalization with no overfitting if:
  • weak classifiers have large edges (implying large margins)
  • weak classifiers not too complex relative to size of training set
• e.g., boosting decision trees resistant to overfitting since trees often have large edges and limited complexity
• overfitting may occur if:
  • small edges (underfitting), or
  • overly complex weak classifiers
• e.g., heart-disease dataset:
  • stumps yield small edges
  • also, small dataset
Improved Boosting with Better Margin-Maximization?

- can design algorithms more effective than AdaBoost at maximizing the minimum margin
- in practice, often perform worse
- why??
- more aggressive margin maximization seems to lead to:
  - more complex weak classifiers (even using same weak learner); or
  - higher minimum margins, but margin distributions that are lower overall

[Breiman]

[Reyzin & Schapire]
Comparison to SVM’s

• both AdaBoost and SVM’s:
  • work by maximizing “margins”
  • find linear threshold function in high-dimensional space

• differences:
  • margin measured slightly differently (using different norms)
  • SVM’s handle high-dimensional space using kernel trick; AdaBoost uses weak learner to search over space
Practical Extensions

- multiclass classification
- ranking problems
- confidence-rated predictions
“Hard” Predictions Can Slow Learning

- ideally, want weak classifier that says:

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

- problem: cannot express using “hard” predictions
- if must predict \( \pm 1 \) below \( L \), will introduce many “bad” predictions
  - need to “clean up” on later rounds
- dramatically increases time to convergence
Confidence-Rated Predictions

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

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

- 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 $\alpha_t$ 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

<table>
<thead>
<tr>
<th>% error</th>
<th>round first reached</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>conf.</td>
<td>no conf.</td>
</tr>
<tr>
<td>40</td>
<td>268</td>
<td>16,938</td>
</tr>
<tr>
<td>35</td>
<td>598</td>
<td>65,292</td>
</tr>
<tr>
<td>30</td>
<td>1,888</td>
<td>&gt;80,000</td>
</tr>
</tbody>
</table>
Application: Boosting for Text Categorization

[Schapire & Singer]

- **weak classifiers**: very simple weak classifiers that test on simple patterns, namely, (sparse) $n$-grams
  - find parameter $\alpha_t$ 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.
## Weak Classifiers

<table>
<thead>
<tr>
<th>rnd</th>
<th>term</th>
<th>AC</th>
<th>AS</th>
<th>BC</th>
<th>CC</th>
<th>CO</th>
<th>CM</th>
<th>DM</th>
<th>DI</th>
<th>HO</th>
<th>PP</th>
<th>RA</th>
<th>3N</th>
<th>TI</th>
<th>TC</th>
<th>OT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>collect</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>card</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>my home</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>4</td>
<td>person ? person</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>5</td>
<td>code</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>6</td>
<td>I</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Monday, July 16, 2012
### More Weak Classifiers

<table>
<thead>
<tr>
<th>rnd</th>
<th>term</th>
<th>AC</th>
<th>AS</th>
<th>BC</th>
<th>CC</th>
<th>CO</th>
<th>CM</th>
<th>DM</th>
<th>DI</th>
<th>HO</th>
<th>PP</th>
<th>RA</th>
<th>3N</th>
<th>TI</th>
<th>TC</th>
<th>OT</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>time</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>wrong number</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>how</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>call</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>seven</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>trying to</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>and</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Monday, July 16, 2012**
# More Weak Classifiers

<table>
<thead>
<tr>
<th>rnd</th>
<th>term</th>
<th>AC</th>
<th>AS</th>
<th>BC</th>
<th>CC</th>
<th>CO</th>
<th>CM</th>
<th>DM</th>
<th>DI</th>
<th>HO</th>
<th>PP</th>
<th>RA</th>
<th>3N</th>
<th>TI</th>
<th>TC</th>
<th>OT</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>third</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>to</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>for</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>charges</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>dial</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>just</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Monday, July 16, 2012
Finding Outliers

examples with most weight are often outliers (mislabeled 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)
Basic Algorithm and Core Theory

• introduction to AdaBoost
• analysis of training error
• analysis of test error and the margins theory
• experiments and applications
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
    \[ \rightarrow \text{overfitting} \]
  - weak classifiers too weak \((\gamma_t \rightarrow 0\) too quickly)
    \[ \rightarrow \text{underfitting} \]
    \[ \rightarrow \text{low margins} \rightarrow \text{overfitting} \]
- 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

---

**Decision Trees**

1. **Eye Color (Brown)**
   - **Yes**: predict +1
   - **No**: predict -1

2. **Height (> 5 feet)**
   - **Yes**: predict +1
   - **No**: predict -1
UCI Results

boosting Stumps

boosting C4.5
Tomorrow: more experiments and applications

• Download and play around with jboost (2.4): http://jboost.sourceforge.net