The Boosting Approach to Machine Learning

Rob Schapire
Princeton University

www.cs.princeton.edu/~schapire
Example: Spam Filtering

- **Problem**: filter out spam (junk email)

- **Gather large collection of examples of spam and non-spam**:
  - From: yoav@att.com  Rob, can you review a paper... non-spam
  - From: xa412@hotmail.com  Earn money without working!!!! ... spam

- **Main observation**:
  - Easy to find “rules of thumb” that are “often” correct
    - *If ‘buy now’ occurs in message, then predict ‘spam’*
  - Hard to find single rule that is very highly accurate
The Boosting Approach

- devise computer program for deriving rough rules of thumb
- apply procedure to subset of emails
- obtain rule of thumb
- apply to 2nd subset of emails
- 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

• more technically:
  • given “weak” learning algorithm that can consistently find classifier with error \( \leq 1/2 - \gamma \)
  • a boosting algorithm can **provably** construct single classifier with error \( \leq \epsilon \)

  \( (\epsilon, \gamma \text{ small}) \)
This Talk

- introduction to AdaBoost
- analysis of training error
- analysis of generalization error based on theory of margins
- applications, experiments and extensions
Background

- [Valiant ’84]:
  - introduced theoretical ("PAC") model for studying machine learning
- [Kearns & Valiant ’88]:
  - open problem of finding a boosting algorithm
- [Schapire ’89], [Freund ’90]:
  - first polynomial-time boosting algorithms
- [Drucker, Schapire & Simard ’92]:
  - first experiments using boosting
Background (cont.)

• [Freund & Schapire ’95]:
  • introduced “AdaBoost” algorithm
  • strong practical advantages over previous boosting algorithms

• experiments and applications using AdaBoost:

  [Drucker & Cortes ’96] [Schapire, Singer & Singhal ’98] [Iyer, Lewis, Schapire, Singer & Singhal ’00]
  [Jackson & Craven ’96] [Abney, Schapire & Singer ’99] [Onoda, Rätsch & Müller ’00]
  [Freund & Schapire ’96] [Haruno, Shirai & Ooyama ’99] [Tieu & Viola ’00]
  [Quinlan ’96] [Cohen & Singer’ 99] [Walker, Rambow & Rogati ’01]
  [Breiman ’96] [Dietterich ’00] [Rochery, Schapire, Rahim & Gupta ’01]
  [Maclin & Opitz ’97] [Schapire & Singer ’00] [Merler, Furlanello, Larcher & Sboner ’01]
  [Bauer & Kohavi ’97] [Collins ’00] :
  [Schwenk & Bengio ’98] [Escudero, Márquez & Rigau ’00]

• continuing development of theory and algorithms:

  [Schapire, Freund, Bartlett & Lee ’98] [Duffy & Helmbold ’99, ’02] [Friedman ’01]
  [Grove & Schuurmans ’98] [Freund & Mason ’99] [Koltchinskii, Panchenko & Lozano ’01]
  [Mason, Bartlett & Baxter ’98] [Ridgeway, Madigan & Richardson ’99] [Collins, Schapire & Singer ’02]
  [Schapire & Singer ’99] [Kivinen & Warmuth ’99] [Demiriz, Bennett & Shawe-Taylor ’02]
  [Cohen & Singer ’99] [Friedman, Hastie & Tibshirani ’00] [Lebanon & Lafferty ’02]
  [Freund & Mason ’99] [Rätsch, Onoda & Müller ’00] :
  [Domingo & Watanabe ’99] [Rätsch, Warmuth, Mika, Onoda, Lemm & Müller ’00] :

  ...
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 \(D_t\):
    \[ \epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i] \]
  - output final classifier \(H_{\text{final}}\)
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 $Z_t = \text{normalization constant}$

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

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

\[ D_1 \]

weak classifiers = vertical or horizontal half-planes
Round 1

\[ h_1 \]

\[ \varepsilon_1 = 0.30 \]
\[ \alpha_1 = 0.42 \]
Round 2

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

\[ \varepsilon_3 = 0.14 \]

\[ \alpha_3 = 0.92 \]
\[ H_{\text{final}} = \text{sign} \left( \begin{array}{c}
0.42 \\
+ 0.65 \\
+ 0.92 \\
\end{array} \right) \]
Analyzing the Training Error

- **Theorem:**
  - write $\epsilon_t$ as $1/2 - \gamma_t$
  - then
    \[
    \text{training error}(H_{\text{final}}) \leq \exp\left(-2\sum_t \gamma_t^2\right)
    \]

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

- **AdaBoost is adaptive:**
  - does not need to know $\gamma$ or $T$ a priori
  - can exploit $\gamma_t \gg \gamma$
Proof Intuition

- on round $t$:
  increase weight of examples incorrectly classified by $h_t$
- if $x_i$ incorrectly classified by $H_{\text{final}}$
  then $x_i$ incorrectly classified by (weighted) majority of $h_t$’s

\[ 	herefore \text{if } x_i \text{ incorrectly classified by } H_{\text{final}} \]
then $x_i$ must have “large” weight under final distribution $D_{T+1}$

\[ \therefore \text{number of incorrectly classified examples “small” (since total weight } \leq 1) \]
Proof

\textbf{Step 1}: unwrapping recurrence:

\[ DT_{T+1}(i) = \frac{1}{m} \exp\left(-y_if(x_i)\right) \prod_t Z_t \]

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

\textbf{Step 2}: 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} \]

\[ = \frac{1}{m} \sum_i \begin{cases} 1 & \text{if } y_if(x_i) \leq 0 \\ 0 & \text{else} \end{cases} \]

\[ \leq \frac{1}{m} \sum_i \exp(-y_if(x_i)) \]

\[ = \sum_i DT_{T+1}(i)\prod_t Z_t \]

\[ = \prod_t Z_t \]

\textbf{Step 3}: \( Z_t = 2\sqrt{\epsilon_t(1 - \epsilon_t)} = \sqrt{1 - 4\gamma_t^2} \leq e^{-2\gamma_t^2} \)
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
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.
A Better Story: Theory of Margins

[with Freund, Bartlett & Lee]

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

• can write: \( H_{\text{final}}(x) = \text{sign}(f(x)) \)

where \( f(x) = \frac{\sum_t \alpha_t h_t(x)}{\sum_t \alpha_t} \in [-1, +1] \)

• define margin of example \((x, y)\) to be \( y f(x) \)
  = measure of confidence of classifications

\[
\text{high conf. incorrect} \quad \text{low conf.} \quad \text{high conf. correct}
\]

\[-1 \quad H_{\text{final}} \quad 0 \quad H_{\text{final}} \quad +1\]

incorrect \quad correct

Empirical Evidence: The Margin Distribution

- margin distribution
  = cumulative distribution of margins of training examples

![Graph showing margin distribution with error on the y-axis and number of rounds (T) on the x-axis.](image)

<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

- if all training examples have large margins, then can approximate final classifier by a much smaller classifier
  - (similar to how polls can predict outcome of a not-too-close election)
- can use this to prove that larger margins $\Rightarrow$ better test error, regardless of number of weak classifiers
- can also prove that boosting tends to increase margins of training examples by concentrating on those with smallest margin
- so:
  although final classifier is getting larger, margins are likely to be increasing, so final classifier is actually getting close to a simpler classifier, driving down the test error
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 \((\gamma_t \to 0\) too quickly) → underfitting
  → low margins → 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

```
+1 predict

<table>
<thead>
<tr>
<th>eye color = brown ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
</tr>
<tr>
<td>predict +1</td>
</tr>
<tr>
<td>no</td>
</tr>
<tr>
<td>predict -1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>height &gt; 5 feet ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
</tr>
<tr>
<td>predict -1</td>
</tr>
<tr>
<td>no</td>
</tr>
<tr>
<td>predict -1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>predict +1</td>
</tr>
</tbody>
</table>
```
UCI Results

boosting Stumps

boosting C4.5
Multiclass Problems

- most direct extension effective only if all weak classifiers have error $\leq 1/2$
  - difficult to achieve for “weak” weak learners
- instead, reduce to binary problem by creating several binary questions for each example:
  “does or does not example $x$ belong to class 1?”
  “does or does not example $x$ belong to class 2?”
  ...
Application: Text Categorization

- weak classifiers are decision stumps
  - test for presence of word or short phrase in document
  - e.g.:
    
    “If the word Clinton appears in the document predict document is about politics”

- in our experiments, consistently beat or tied tested competitors
Extension: Confidence-rated Predictions

• useful to allow weak classifiers to express **confidences** about predictions

• formally, allow $h_t : X \rightarrow \mathbb{R}$

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

$$|h_t(x)| = \text{“confidence”}$$

• proposed **general principle** for:
  • modifying AdaBoost
  • designing weak learner to find (confidence-rated) $h_t$’s

• sometimes makes learning much faster since removes need to undo under-confident predictions of earlier weak classifiers
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: Human-computer Spoken Dialogue
[with Rahim, Di Fabbrizio, Dutton, Gupta, Hollister & Riccardi]

- phone “helpdesk” for AT&T’s Natural Voices text-to-speech business (1-877-741-4321)

- NLU’s job: classify caller utterances into 24 categories (demo, sales rep, pricing info, yes, no, etc.)
Incorporating Human Knowledge [with Rochery, Rahim & Gupta]

- boosting is data-driven
  - so works best with lots of data
- for rapid deployment, can’t wait to gather lots of data
  - want to compensate with human knowledge
- idea: balance fit to data against fit to prior, human-built model
Results

Classification Accuracy vs # Training Examples

- **data + knowledge**
- **data**
- **knowledge**
Application: Bidding Agents [with Stone, Csirik, Littman & McAllester]

- trading agent competition (TAC)
- 8 agents in each game
- must purchase flights, hotel rooms and entertainment tickets for 8 clients in complicated, interacting auctions
- value of one good depends on price of others
  - e.g., need both ingoing and outgoing flights
- so: need to predict prices, especially of hotel rooms
  - used boosting
- second place in tournament using straight scores
  - (first place with “handicapped” scores)
Predicting Hotel Prices

- predicting real numbers (prices)
- want to estimate entire distribution of prices, given current conditions

- main ideas:
  - reduce to multiple binary classification problems:
    - “is price above or below $100?”
    - “is price above or below $150?”
      :
  - extract probabilities using modification of boosting for logistic regression [Collins, Schapire & Singer, ’02] [Duffy & Helmbold ’99]

- can be applied to any conditional density estimation problem
Conclusions

- boosting is a useful new 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

- other stuff:
  - theoretical connections to:
    - game theory and linear programming
    - support-vector machines
    - logistic regression
    - convex analysis and Bregman distances
  - tool for data cleaning:
    - very effective at finding outliers (mislabeled or ambiguously labeled examples)