Boosting

Can we make weak learners smart?

Instructors: Pradeep Ravikumar, Manuela Veloso

Machine Learning 10-701

Slides Courtesy: Carlos Guestrin, Freund & Schapire
Weak Learners

**Goal:** Automatically categorize type of call requested (Collect, Calling card, Person-to-person, 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)*

- What is a quick “rule of thumb” a lay person might come up with to predict this?
  - E.g. If ‘card’ occurs in utterance, then predict ‘calling card’
  - Each such “rule of thumb” is a *weak* learner
  - Easy to find “rules of thumb” that are “often” correct
Can we “boost” such weak learners?

Goal: Automatically categorize type of call requested
(Collect, Calling card, Person-to-person, etc.)

- Easy to find “rules of thumb” that are “often” correct.
  E.g. If ‘card’ occurs in utterance, then predict ‘calling card’

- Hard to find single highly accurate prediction rule.

- QUESTION: Given access to a mechanism that gives us weak rules of thumb, can we “boost” these to a highly predictive learner?
What are weak learners good and bad at?

- **Simple (a.k.a. weak) learners** e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees)

  ![Diagram](image)

  **Are good 😊**
  
  Low variance (loosely: are very close to their expectation)

  **Are bad 😞**
  
  High bias (loosely: their expectation is far away from the truth)
  Can’t solve hard learning problems

- **Can we make weak learners always good??**
  
  – **No!!!**
  
  But often yes...
Voting (Ensemble Methods)

• Wisdom of the crowds!
• Instead of learning a single (weak) classifier, learn **many weak classifiers** that are **good at different parts of the input space**

• **Output class:** (Weighted) vote of each classifier
  – Classifiers that are most “sure” will vote with more conviction
  – Classifiers will be most “sure” about a particular part of the space
  – On average, do better than single classifier!

\[ h_{\text{combined}} : X \mapsto Y \equiv \{ -1, 1 \} \]

\[ h_{\text{combined}} = \text{sign} \left( \sum_t \alpha_t h_t(X) \right) \]

weights

\[
\begin{array}{c|c}
1 & -1 \\
? & ? \\
\end{array}
\begin{array}{c|c}
? & ? \\
1 & -1 \\
\end{array}
\]
Voting (Ensemble Methods)

• Instead of learning a single (weak) classifier, learn many weak classifiers that are good at different parts of the input space

• Output class: (Weighted) vote of each classifier
  – Classifiers that are most “sure” will vote with more conviction
  – Classifiers will be most “sure” about a particular part of the space
  – On average, do better than single classifier!

• But how do you ???
  – force classifiers $h_t$ to learn about different parts of the input space?
  – weigh the votes of different classifiers? $\alpha_t$
Boosting [Schapire’89]

• Idea: given a weak learner, run it multiple times on (rewighted) training data, then let learned classifiers vote

• On each iteration $t$:
  – weight each training example by how incorrectly it was classified
  – Learn a weak hypothesis: $h_t$
  – A strength for this hypothesis: $\alpha_t$

• Final classifier: $H(X) = \text{sign}(\sum \alpha_t h_t(X))$

• Practically useful
• Theoretically interesting
Learning from weighted data

• Consider a weighted dataset
  – $D(i)$ – weight of $i$th training example $(x^i,y^i)$
  – Interpretations:
    • $i$th training example counts as $D(i)$ examples
    • If I were to “resample” data, I would get more samples of “heavier” data points

• Now, in all calculations, whenever used, $i$th training example counts as $D(i)$ “examples”
  e.g. weighted MLE: $\max_\theta \sum_{i=1}^{n} D(i) \log p(X_i; \theta)$

• An option in many off-the-shelf packages e.g. in scikit-learn
Given: \((x_1, y_1), \ldots, (x_m, y_m)\) where \(x_i \in X, y_i \in Y = \{-1, +1\}\)

Initialize \(D_1(i) = 1/m.\)

For \(t = 1, \ldots, T:\)

- Train weak learner using distribution \(D_t.\)
- Get weak classifier \(h_t : X \rightarrow \mathbb{R}.\)
- Choose \(\alpha_t \in \mathbb{R}.\)
- Update:

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

\[
= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\]

where \(Z_t\) is a normalization factor.
Given: \((x_1, y_1), \ldots, (x_m, y_m)\) where \(x_i \in X, y_i \in Y = \{-1, +1\}\)

Initialize \(D_1(i) = 1/m\).  

For \(t = 1, \ldots, T\):

- Train weak learner using distribution \(D_t\).
- Get weak classifier \(h_t : X \rightarrow \mathbb{R}\).
- Choose \(\alpha_t \in \mathbb{R}\).
- Update:

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

where \(Z_t\) is a normalization factor

\[
Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))
\]

Weights for all pts must sum to 1

\[
\sum_{t} D_{t+1}(i) = 1
\]
AdaBoost [Freund & Schapire’95]

Given: \((x_1, y_1), \ldots, (x_m, y_m)\) where \(x_i \in X, y_i \in Y = \{-1, +1\}\)

Initialize \(D_1(i) = 1/m\). \textcolor{red}{\textbf{Initially equal weights}}

For \(t = 1, \ldots, T\):

- Train weak learner using distribution \(D_t\). \textcolor{red}{\textbf{Naïve bayes, decision stump}}
- Get weak classifier \(h_t : X \to \mathbb{R}\).
- Choose \(\alpha_t \in \mathbb{R}\). \textcolor{red}{\textbf{Magic (+ve)}}
- Update:

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

where \(Z_t\) is a normalization factor

Increase weight if wrong on pt \(i\)
\(y_i h_t(x_i) = -1 < 0\)

Output the final classifier:

\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right).
\]
What $\alpha_t$ to choose for hypothesis $h_t$?

Weight Update Rule:

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

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

[Freund & Schapire’95]

Weighted training error

$$\epsilon_t = P_{i \sim D_t(i)}[h_t(x^i) \neq y^i] = \sum_{i=1}^{m} D_t(i) \delta(h_t(x_i) \neq y_i)$$

Does $h_t$ get $i^{th}$ point wrong

$\epsilon_t = 0$ if $h_t$ perfectly classifies all weighted data pts

$\epsilon_t = 1$ if $h_t$ perfectly wrong $\Rightarrow$ $-h_t$ perfectly right

$\epsilon_t = 0.5$

$\alpha_t = \infty$

$\alpha_t = -\infty$

$\alpha_t = 0$
Boosting Example (Decision Stumps)

$D_1$

$D_2$

$D_3$

$h_1$

$h_2$

$h_3$

$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$

$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$

$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$
Boosting Example (Decision Stumps)

\[ H_{\text{final}} = \text{sign}(0.42 + 0.65 + 0.92) \]
Analysis reveals:

• What $\alpha_t$ to choose for hypothesis $h_t$?

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

$\epsilon_t$ - weighted training error

• If each weak learner $h_t$ is slightly better than random guessing ($\epsilon_t < 0.5$), then training error of AdaBoost decays exponentially fast in number of rounds $T$.

THEOREM: $\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \exp \left( -2 \sum_{t=1}^{T} (1/2 - \epsilon_t)^2 \right)$

Training Error

Edge of weak learner in iteration $t$
Training error bound

Training error of final classifier is bounded by:

\[
\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f(x_i))
\]

Where

\[
f(x) = \sum_{t} \alpha_t h_t(x); \ H(x) = \text{sign}(f(x))
\]

If boosting can make upper bound → 0, then training error → 0.
Training error bound

Training error of final classifier is bounded by:

**Lemma:** 
\[
\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f(x_i)) = \prod_{t} Z_t
\]

Where \( f(x) = \sum_{t} \alpha_t h_t(x); \quad H(x) = \text{sign}(f(x)) \)

**Recall:**

where \( Z_t \) is a normalization factor

\[
Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))
\]
Training error bound

Training error of final classifier is bounded by:

**LEMMA:**
\[
\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f(x_i)) = \prod_{t} Z_t
\]

Where
\[
f(x) = \sum_{t} \alpha_t h_t(x); \quad H(x) = \text{sign}(f(x))
\]

**Proof:** Using Weight Update Rule

\[
D_T + 1(i) = \frac{1}{m} \frac{\exp(-y_i f(x_i))}{\prod_{t} Z_t}
\]

Wts of all pts add to 1

\[
\sum_{i=1}^{m} D_T + 1(i) = 1
\]
Analyzing training error

Training error of final classifier is bounded by:

\[
\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f(x_i)) = \prod_{t} Z_t
\]

Where \( f(x) = \sum_{t} \alpha_t h_t(x); H(x) = \text{sign}(f(x)) \)

If \( Z_t < 1 \), training error decreases exponentially (even though weak learners may not be good \( \varepsilon_t \sim 0.5 \))
What $\alpha_t$ to choose for hypothesis $h_t$?

Training error of final classifier is bounded by:

$$\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f(x_i)) = \prod_t Z_t$$

Where $f(x) = \sum_t \alpha_t h_t(x)$; $H(x) = \text{sign}(f(x))$

If we minimize $\prod_t Z_t$, we minimize our training error

We can tighten this bound greedily, by choosing $\alpha_t$ and $h_t$ on each iteration to minimize $Z_t$.

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$
What $\alpha_t$ to choose for hypothesis $h_t$?

We can minimize this bound by choosing $\alpha_t$ on each iteration to minimize $Z_t$.

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

For boolean target function, this is accomplished by [Freund & Schapire ’97]:

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

**Proof:**

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

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

$$\frac{\partial Z_t}{\alpha_t} = \epsilon_t e^{\alpha_t} - (1 - \epsilon_t) e^{-\alpha_t} = 0 \Rightarrow e^{2\alpha_t} = \frac{1 - \epsilon_t}{\epsilon_t}$$
What $\alpha_t$ to choose for hypothesis $h_t$?

We can minimize this bound by choosing $\alpha_t$ on each iteration to minimize $Z_t$.

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

For boolean target function, this is accomplished by [Freund & Schapire ’97]:

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

**Proof:**

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

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

$$= 2\sqrt{\epsilon_t(1 - \epsilon_t)} = \sqrt{1 - (1 - 2\epsilon_t)^2}$$
Dumb classifiers made Smart

Training error of final classifier is bounded by:

\[
\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \prod_{t} Z_t = \prod_{t} \sqrt{1 - (1 - 2\epsilon_t)^2} \\
\leq \exp \left( -2 \sum_{t=1}^{T} (1/2 - \epsilon_t)^2 \right)
\]

If each classifier is (at least slightly) better than random \( \epsilon_t < 0.5 \)

AdaBoost will achieve zero training error exponentially fast (in number of rounds \( T \)) !!

What about test error?
Boosting results – Digit recognition

• Boosting often, but not always
  – Robust to overfitting
  – Test set error decreases even after training error is zero

[Schapire, 1989]
Generalization Error Bounds

$error_{true}(H) \leq error_{train}(H) + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$

<table>
<thead>
<tr>
<th>bias</th>
<th>variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>large</td>
<td>small</td>
</tr>
<tr>
<td>small</td>
<td>large</td>
</tr>
</tbody>
</table>

T small
T large

• T – number of boosting rounds
• d – VC dimension of weak learner, measures complexity of classifier
• m – number of training examples

[Freund & Schapire’95]
Generalization Error Bounds

$\text{error}_{true}(H) \leq \text{error}_{train}(H) + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$

With high probability

Boosting can overfit if $T$ is large

Boosting often, 

- Robust to overfitting
- Test set error decreases even after training error is zero

Need better analysis tools – *margin based bounds*
Comparison of C4.5 (decision trees) vs Boosting decision stumps (depth 1 trees)

C4.5 vs Boosting C4.5

27 benchmark datasets
AdaBoost and AdaBoost.MH on Train (left) and Test (right) data from Irvine repository. [Schapire and Singer, ML 1999]
Logistic regression assumes:

\[ P(Y = 1|X) = \frac{1}{1 + \exp(f(x))} \]

And tries to maximize data likelihood:

\[ P(D|f) \overset{iid}{=} \prod_{i=1}^{m} \frac{1}{1 + \exp(-y_i f(x_i))} \]

Equivalent to minimizing log loss

\[ - \log P(D|f) = \sum_{i=1}^{m} \ln(1 + \exp(-y_i f(x_i))) \]
Boosting and Logistic Regression

Logistic regression equivalent to minimizing log loss

\[
\sum_{i=1}^{m} \ln(1 + \exp(-y_if(x_i)))
\]

\[f(x) = w_0 + \sum_{j} w_jx_j\]

Boosting minimizes similar loss function!!

\[
\frac{1}{m} \sum_{i=1}^{m} \exp(-y_if(x_i)) = \prod_{t} Z_t
\]

\[f(x) = \sum_{t} \alpha_th_t(x)\]

Weighted average of weak learners

Both smooth approximations of 0/1 loss!
Boosting and Logistic Regression

Logistic regression:

- Minimize log loss
  \[ \sum_{i=1}^{m} \ln(1 + \exp(-y_i f(x_i))) \]
- Define
  \[ f(x) = \sum_j w_j x_j \]
  where \( x_j \) predefined features
  (linear classifier)
- Jointly optimize over all weights \( w_0, w_1, w_2... \)

Boosting:

- Minimize exp loss
  \[ \sum_{i=1}^{m} \exp(-y_i f(x_i)) \]
- Define
  \[ f(x) = \sum_t \alpha_t h_t(x) \]
  where \( h_t(x) \) defined dynamically to fit data
  (not a linear classifier)
- Weights \( \alpha_t \) learned per iteration incrementally
Effect of Outliers

**Good 😊**: Can identify outliers since focuses on examples that are hard to categorize

**Bad 😞**: Too many outliers can degrade classification performance dramatically increase time to convergence
Bagging

[Breiman, 1996]

Related approach to combining classifiers:

1. Run independent weak learners on bootstrap replicates (sample with replacement) of the training set
2. Average/vote over weak hypotheses

<table>
<thead>
<tr>
<th>Bagging</th>
<th>vs.</th>
<th>Boosting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resamples data points</td>
<td></td>
<td>Reweights data points (modifies their distribution)</td>
</tr>
<tr>
<td>Weight of each classifier is the same</td>
<td></td>
<td>Weight is dependent on classifier’s accuracy</td>
</tr>
<tr>
<td>Only variance reduction</td>
<td></td>
<td>Both bias and variance reduced – learning rule becomes more complex with iterations</td>
</tr>
</tbody>
</table>
Boosting Summary

• Combine weak classifiers to obtain very strong classifier
  – Weak classifier – slightly better than random on training data
  – Resulting very strong classifier – can eventually provide zero training error

• AdaBoost algorithm

• Boosting v. Logistic Regression
  – Similar loss functions
  – Single optimization (LR) v. Incrementally improving classification (B)

• Most popular application of Boosting:
  – Boosted decision stumps!
  – Very simple to implement, very effective classifier