Linear separation, drifting games & Boosting

Yoav Freund
UCSD
Adaboost is sensitive to label noise

- Letter / Irvine Database
- Focus on a binary problem: \{F,I,J\} vs. other letters.

<table>
<thead>
<tr>
<th>Label Noise</th>
<th>Adaboost</th>
<th>Logitboost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>0.8% ±0.2%</td>
<td>0.8% ±0.1%</td>
</tr>
<tr>
<td>20%</td>
<td>33.3% ±0.7%</td>
<td>31.6% ±0.6%</td>
</tr>
</tbody>
</table>

- Boosting puts too much weight on outliers.
- Need to give up on outliers.
Robustboost - A new boosting algorithm

<table>
<thead>
<tr>
<th>Label Noise</th>
<th>Adaboost</th>
<th>Logitboost</th>
<th>Robustboost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>0.8% ±0.2%</td>
<td>0.8% ±0.1%</td>
<td>2.9% ±0.2%</td>
</tr>
<tr>
<td>20%</td>
<td>33.3% ±0.7%</td>
<td>31.6% ±0.6%</td>
<td>22.2 ±0.8%</td>
</tr>
</tbody>
</table>

error with respect to original (noiseless) labels

| 20%          | 22.1% ±1.2%    | 19.4% ±1.3%    | 3.7% ±0.4%     |

Wednesday, June 10, 2009
Plan of talk

• Label noise and convex loss functions.
• Boost by Majority and drifting games.
• Boosting in continuous time.
• RobustBoost
• Experimental results.
Plan of talk

- Label noise and convex loss functions.
- Boost by Majority and drifting games.
- Boosting in continuous time.
- RobustBoost
- Experimental results.
Label noise and convex loss functions

- Algorithms for learning a classifier based on minimizing a convex loss function: perceptron, Adaboost, Logitboost, soft margins SVM.
- Work well when data is linearly separable.
- Get into trouble when not linearly separable.
- **Problem**: Convex loss functions are a poor approximation for classification error.
- **But**: No efficient algorithms for minimizing a non-convex loss function.
Loss functions

![Graph showing the relationship between loss and margin for various methods: classification error, AdaBoost, LogitBoost, Perceptron, and Soft-Margins. The x-axis represents the margin, and the y-axis represents the loss. The graph illustrates how these methods handle classification error under different margins.]
A hard case
Long & Servedio ICML 2008

Large Margin

Penalizers

puller

penalizers

puller

Large Margin
Theorem: for any convex loss function there exists a linearly separable distribution such that when independent label noise is added, the linear classifier that minimizes the loss function has very poor classification error.
Plan of talk

- Label noise and convex loss functions.
- Boost by Majority and drifting games.
- Boosting in continuous time.
- RobustBoost
- Experimental results.
Boost by Majority (BBM)

[Freund 95]

game between a **booster** and a weak **learner**.
- Boosting generates a simple (unweighted) majority rule over weak learners.
- \( T \) Number of iterations is **set in advance**
- On iteration \( t=1..T \)
  - **booster** assigns weights to the training examples.
  - **learner** chooses a rule whose error wrt the chosen weights is smaller than \( 1/2 - \gamma \)
  - Rule is added to majority rule
- Goal of booster is to minimize number of errors of final majority rule.
BBM as a drifting game

- Chips = examples
- bin $i$ contains the examples for which the difference between the number of correct and incorrect rules is $i$
The continuous chip limit

- Number of examples increases to infinity.
- Alternatively - think of examples as a probability mass with probability measure \( \mu \) defined on it.

Initial Configuration

\[ y \sum_{t=1}^{T} h_t(x) \]
The boosting game lattice

\[ s = y \sum_{t=1}^{T} h_t(x) \]

Assume \( T \) is odd to avoid ties
$\gamma = 0.1$

Initial configuration

incorrect | correct

$-3 \ -2 \ -1 \ 0 \ 1 \ 2 \ 3$
Booster assigns weights to examples

$\gamma = 0.1$

-3 -2 -1 0 1 2 3

incorrect correct
Weak learner chooses subset with weight $\frac{1}{2} + \gamma$ which $h_1(x)$ classifies correctly.
Weak learner chooses subset with weight $\frac{1}{2} + \gamma$ which $h_1(x)$ classifies correctly.

$\gamma = 0.1$
γ = 0.1

Booster assigns weights to examples

s

-3 -2 -1 0 1 2 3

incorrect  correct

0.6  0.4
\[\gamma = 0.1\]

Weak learner chooses subset with weight \(\frac{1}{2} + \gamma\) which \(h_2(x)\) classifies correctly.
\[ \gamma = 0.1 \]

Weak learner chooses subset with weight \( \frac{1}{2} + \gamma \) which \( h_2(x) \) classifies correctly.
\( \gamma = 0.1 \)

Booster assigns weights to examples.
Weak learner chooses subset with weight $1/2 + \gamma$ which $h_3(x)$ classifies correctly.

$\gamma = 0.1$
\( \gamma = 0.1 \)

Weak learner chooses subset with weight \( \frac{1}{2} + \gamma \) which \( h_3(x) \) classifies correctly.
$\gamma = 0.1$

Majority[$h_3(x) + h_3(x) + h_3(x)$]  
incorrect

Majority[$h_3(x) + h_3(x) + h_3(x)$]  
correct

Wednesday, June 10, 2009
Weak Learner’s min/max strategy

- **AdOpt** - Choose $1/2 + \gamma$ from each bin to be correct.

- Equivalent to producing the correct answer independently at random with
  
  $$P(\text{correct}) = 1/2 + \gamma$$
Potential

**Total potential:** $\Psi(t, \text{configuration}) - \mu$-prob of the examples on which the final majority vote is incorrect given the configuration after iteration $t$ is configuration and on the remaining steps the learner plays AdOpt.

$\Psi(0, \text{all at origin}) = \text{Initial potential.}$

$\Psi(T, \text{configuration}) = \text{Training error of final majority rule.}$

Boosting algorithm chooses weights so that the total potential does not increase.

Initial potential $\geq$ final training error.
**Bin Potential:** $\psi(s,t)$ - fraction of examples in bin $s$ after iteration $t$ on which the final majority rule will be incorrect assuming AdOpt play

**Equivalently:** $\psi(s,t) = \text{probability of } \leq (T-t-s)/2$ heads in $T-t$ flips of a coin with $p(\text{head}) = 1/2 + \gamma$

\[
\psi(t,s) = \text{Binom} \left( T-t, \frac{T-t-s}{2}, \frac{1}{2} + \gamma \right);
\quad \text{Binom} \left( n, k, p \right) = \sum_{j=0}^{\lfloor k \rfloor} \binom{n}{j} p^j (1-p)^{n-j}
\]

\[
\psi(t-1,s) = \left( \frac{1}{2} - \gamma \right) \psi(t,s-1) + \left( \frac{1}{2} + \gamma \right) \psi(t,s+1)
\]

\[
\psi(T,s) = \begin{cases} 
0 & s > 0 \\
1 & s \leq 0 
\end{cases}
\]

$\psi(0,0) = \text{Binom} \left( T, \frac{T}{2}, \frac{1}{2} + \gamma \right)$
Evolution of total potential

\(f(t, s)\) The \(\mu\)-prob of examples in bin \(s\) after iteration \(t\)

\[\Psi(t) = \sum_i f(t, s)\psi(t, s)\]

\(d(t, s)\) \(\mu((x,y) \text{ in bin } s \mid h_t(x) = y) - \mu((x,y) \text{ in bin } s \mid h_t(x) \neq y)\)

\[\Psi(t + 1) = \Psi(t) + \sum_s \left(\frac{1}{2} + \gamma - d(t, s)\right)w(t, s)\]

\(w(t, s) \triangleq \psi(t + 1, s - 1) - \psi(t + 1, s + 1)\)

If \[\sum_s d(t, s)w(t, s) \geq \frac{1}{2} + \gamma\] then \(\Psi(t + 1) \leq \Psi(t)\)
Potential and weight for the boosting game

setting the boosting weights at iteration $t$ to be

$$w(t,s) = \left( \begin{array}{c} T-t \\ \frac{T-t-s+1}{2} \end{array} \right) \left( \frac{1}{2} + \gamma \right)^{\frac{T-t-s+1}{2}} \left( \frac{1}{2} - \gamma \right)^{\frac{T-t+s-1}{2}}$$

guarantees

Initial potential $= \Psi(0) \geq \Psi(1) \geq \cdots \geq \Psi(T) = \text{training error of sign} \left( \sum_{t=1}^{T} h_t(x) \right)$

$$\varepsilon = \Psi(0) = \psi(0,0) = \text{Binom} \left( T, \frac{T}{2}, \frac{1}{2} + \gamma \right)$$
t=71
\[ \psi_{Ada}(s) = w_{Ada}(s) = e^{-s} \]

\[ \psi_{Logit}(s) = \ln(1 + e^{-s}) \]

\[ w_{Logit}(s) = \frac{1}{1 + e^s} \]
High level summary

- The worst case adversary splits each bin into:
  \( \frac{1}{2} - \gamma \) incorrect / \( \frac{1}{2} + \gamma \) correct

- Alternative interpretation: Random walk with IID steps.

- Algorithm is derived as optimal response to this simple worst-case adversary.
Plan of talk

• Label noise and convex loss functions.
• Boost by Majority and drifting games.
• **Boosting in continuous time.**
• RobustBoost
• Experimental results.
Why is BBM not practical?

- BBM needs to know $\varepsilon, \gamma$ before starting.

\[ T = \frac{1}{\gamma^2} \ln \frac{1}{\varepsilon} \]

- Adaboost = adaptive boosting, Adapts to the sequence, $\gamma_1, \gamma_2, \gamma_3, \ldots$.
  - No need to set parameters in advance.
  - Generates a weighted majority rule.
  - Decide when to stop using cross-validation.

- How can we make BBM adaptive?
Letting time step decrease to zero.

- Number of iterations required by BBM: 
  \[ T = \frac{1}{\gamma^2} \ln \frac{1}{\varepsilon} \]

- Keep \( \varepsilon \) fixed and let \( \gamma \to 0, T \to \infty \)

- Instead of \( t=1,2,...,T \) use \( t=\frac{1}{T},\frac{2}{T},...,1 \)

- The same weak rule is added many times, until it’s advantage falls below \( \gamma \). 
  - yields adaptive boosting and a weighted majority rule.

- In the limit, adversary uses random walk in continuous time = Brownian Motion.
The game lattice

T=3, t=1,2,3
Using step \[ \Delta s = \pm \frac{1}{T} \]

\[ T = 1, \Delta t = 1, \Delta s = \pm 1 \]
Using step \[ \Delta s = \pm \frac{1}{T} \]

\[ T = 3, \Delta t = \frac{1}{3}, \Delta s = \pm \frac{1}{3} \]
Using step \( \Delta s = \pm \frac{1}{T} \)

\[
T = 9, \; \Delta t = \frac{1}{9}, \; \Delta s = \pm \frac{1}{9}
\]

Looks fine but \( \text{var}(s) = T \frac{1}{T^2} = \frac{1}{T} \rightarrow 0 \quad \text{as} \quad T \rightarrow \infty \)
Using step \( \Delta s = \pm \frac{1}{\sqrt{T}} \)

\( T = 1, \Delta t = 1, \Delta s = \pm 1 \)
Using step \( \Delta s = \pm \frac{1}{\sqrt{T}} \)

\[ T = 3, \Delta t = \frac{1}{3}, \Delta s = \pm \frac{1}{\sqrt{3}} \]

\[ \text{var}(s) = 3 \frac{1}{3} = 1 \]
Using step \( \Delta s = \pm \frac{1}{\sqrt{T}} \)

\[ T = 9, \quad \Delta t = \frac{1}{9}, \quad \Delta s = \pm \frac{1}{3} \]

\[ \text{var}(s) = 9 \cdot \frac{1}{9} = 1 \quad \text{but range}(s) \to \infty \]
Potentials in continuous time

- **Discrete time:** Eqns relating time $t$ to time $t+1$ based on random walks.

- **Continuous time:**
  - **Differential Equations** for the time evolution of the distribution (Kolmogorov forward and backward equations).
  - **Stochastic Differential Equations** for describing the paths taken by individual experts / examples. (Ito Calculus).
Example: From BBM to Brownboost

Bin Potential for BBM:

\[
\psi(t-1,s) = \left( \frac{1}{2} - \gamma \right) \psi(t,s-1) + \left( \frac{1}{2} + \gamma \right) \psi(t,s+1)
\]

\[
\psi(T,s) = \begin{cases} 
0 & s > 0 \\
1 & s \leq 0
\end{cases} \quad \varepsilon = \psi(0,0) = \text{Binom}\left(T, \frac{T}{2}, \frac{1}{2} + \gamma \right)
\]

Bin Potential for Brownboost:

\[
\frac{\partial}{\partial t} \psi(t,s) = -\frac{1}{2} \frac{\partial^2}{\partial s^2} \psi(t,s) - 2\sqrt{\beta} \frac{\partial}{\partial s} \psi(t,s)
\]

Boundary conditions:

\[
\psi(1,s) = \begin{cases} 
0 & s > 0 \\
1 & s \leq 0
\end{cases} \quad \varepsilon = \psi(0,0)
\]
Plan of talk

• Label noise and convex loss functions.
• Boost by Majority and drifting games.
• Boosting in continuous time.
• RobustBoost
• Experimental results.
Robustboost

• Instead of minimizing training error, minimize number of training examples with margins smaller than $\theta$ (different from minimal margins and form soft margins.)

• Need to control the magnitude of the margins.

• Allow confidence-rated weak learners
Robustboost

\[ \psi(t,s) = \min\left\{ 1, 1 - \text{erf}\left( \frac{s - \mu(t)}{\sigma(t)} \right) \right\}; \quad \text{erf}(s) = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{s} e^{-x^2/2} \, dx \]

\[ w(t,s) = \begin{cases} 
\exp\left( -\left( \frac{s - \mu(t)}{\sigma(t)} \right)^2 \right) & \text{if } s > \mu(t) \\
0 & \text{if } s \leq \mu(t) 
\end{cases} \]

\[ \mu(t) = (\theta - 2\rho)e^{1-t} + 2\rho \quad \sigma^2(t) = (\sigma_f^2 + 1)e^{2(1-t)} - 1 \]

set \( \rho \) to satisfy \( \varepsilon = \psi(0,0) = 1 - \text{erf}\left( \frac{2(e-1)\rho - e\theta}{\sqrt{e^2(\sigma_f^2 + 1) - 1}} \right) \)
Plan of talk

• Label noise and convex loss functions.
• Boost by Majority and drifting games.
• Boosting in continuous time.
• RobustBoost
• Experimental results.
Experimental Results on Long/Servedio synthetic example
Adaboost on Long/Servedio

![Graph showing Adaboost on Long/Servedio](image)
LogitBoost on Long/Servedio
Robustboost on Long/Servedio
New in Version 2.0!

The following are the new features of JBoost 2.0:

- RobustBoost support added -- a new boosting algorithm that is resistant to label noise.
- A new visualization tool -- the score visualizer
- Support for stopping and restarting the boosting process while eliminating those examples with small weight from the restarted process.
- JBoost no longer supports Multi-class problems internally, but now offers a wrapper script.

Overview

JBoost is an easy to use and modify tool for boosting classification. JBoost includes state-of-the-art algorithms and can be used by researchers to quickly implement new boosting algorithms. JBoost also includes a set of easy to use scripts so that machine learning novices can quickly learn and utilize the power of boosting.

Some of the algorithms currently implemented include AdaBoost, LogitBoost, BoosTexter and RobustBoost. These algorithms are wrapped inside of an implementation of alternating decision trees (ADTrees), which allows for easy visualization of the final classifier, even for high dimensional data. Each of the algorithms comes with a set of options that allows for customization to your dataset.

To learn more, download JBoost or read the documentation.
Experimental Results on real-world data
Robustboost - A new boosting algorithm

<table>
<thead>
<tr>
<th>Label Noise</th>
<th>Adaboost</th>
<th>Logitboost</th>
<th>Robustboost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>0.8% ±0.2%</td>
<td>0.8% ±0.1%</td>
<td>2.9% ±0.2%</td>
</tr>
<tr>
<td>20%</td>
<td>33.3% ±0.7%</td>
<td>31.6% ±0.6%</td>
<td>22.2 ±0.8%</td>
</tr>
</tbody>
</table>

error with respect to original (noiseless) labels

| 20%        | 22.1% ±1.2%  | 19.4% ±1.3%  | 3.7% ±0.4%  |
Logitboost
0% Noise
Wednesday, June 10, 2009
Logitboost
20% Noise
Robustboost

20% Noise
Plan of talk

• Label noise and convex loss functions.
• Boost by Majority and drifting games.
• Boosting in continuous time.
• RobustBoost
• Experimental results.
Thank you!