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>
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.
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>
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
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%     |

Tuesday, June 16, 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.
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.
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.
- Can get into trouble when not linearly separable.
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.
- Can get into trouble when not linearly separable.
- **Problem**: Convex loss functions are a poor approximation for classification error.
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.

• Can 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 loss functions for classification error, Adaboost, Logitboost, Perceptron, and Soft-Margins.](graph.png)
A hard case

Long & Servedio ICML 2008

Tuesday, June 16, 2009
A hard case
Long & Servedio ICML 2008

Adding 10% label noise

Large Margin

Penalizers

puller

Penalizers

Large Margin

puller
A hard case
Long & Servedio ICML 2008

Adding 10% label noise
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 high 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]
Boost by Majority (BBM)

[Freund 95]

- game between a booster and a weak learner.
Boost by Majority (BBM)

[Freund 95]

- game between a booster and a weak learner.
- Boosting generates a simple (unweighted) majority rule over weak learners.
Boost by Majority (BBM)

- 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

[Freund 95]
Boost by Majority (BBM)

- 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 $t=1..T$

[Freund 95]
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.
Boost by Majority (BBM)

- 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 $1/2 - \gamma$

[Freund 95]
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
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

Initial Configuration

\[ s \doteq y \sum_{t=1}^{T} h_t(x) \]
BBM as a drifting game

- Chips = examples

Initial Configuration

\[ s = y \sum_{t=1}^{T} h_t(x) \]
BBM as a drifting game

- Chips = examples
- bin $s$ contains the examples for which the difference between the number of correct and incorrect base rules is $s$

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

Initial Configuration

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

- Number of examples increases to infinity.

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

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

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

\text{correct}
$\gamma = 0.1$

Booster assigns weights to examples

\[ s \]

-3 -2 -1 0 1 2 3

incorrect correct
\[ \gamma = 0.1 \]

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.
\[ \gamma = 0.1 \]

Booster assigns weights to examples

\[ \begin{align*}
0.6 & \quad \text{incorrect} \\
0.4 & \quad \text{correct}
\end{align*} \]
\[ \gamma = 0.1 \]

Weak learner chooses subset with weight \(1/2 + \gamma\) 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.

![Bar chart showing weights assigned to examples with labels incorrect and correct.](image)
Weak learner chooses subset with weight $1/2 + \gamma$ which $h_3(x)$ classifies correctly.

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

$\gamma = 0.1$
\( \gamma = 0.1 \)
Weak Learner’s min/max strategy
Weak Learner’s min/max strategy

- **AdOpt** - Choose $\frac{1}{2} + \gamma$ from each bin to be correct.
Weak Learner’s min/max strategy

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

- Equivalently: prediction of each base rule on each example is chosen independently at random

  $$P(h_t(x)=y) = \frac{1}{2} + \gamma$$
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.
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.

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

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

$\Psi(T, \text{configuration}) = \text{Training error of final majority rule}.$
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.
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: $\psi(t,s)$ - fraction of examples in bin $s$ after iteration $t$ on which the final majority rule will be incorrect assuming AdOpt play.
Bin Potential: $\psi(t,s)$ - fraction of examples in bin $s$ after iteration $t$ on which the final majority rule will be incorrect assuming AdOpt play

$f(t,s)$ The $\mu$-prob of examples in bin $s$ after iteration $t$
**Bin Potential:** \( \psi(t,s) \) - fraction of examples in bin \( s \) after iteration \( t \) on which the final majority rule will be incorrect assuming **AdOpt** play

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

\[ \Psi(t) = \sum_{i} f(t,s)\psi(t,s) \]
Bin Potential: $\psi(t,s)$ - fraction of examples in bin $s$ after iteration $t$ on which the final majority rule will be incorrect assuming AdOpt play.

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

$$\Psi(t) = \sum_i f(t,s)\psi(t,s)$$

$\psi(t,s)$ does not depend on the configuration.
**Bin Potential:** \( \psi(t,s) \) - fraction of examples in bin \( s \) after iteration \( t \) on which the final majority rule will be incorrect assuming \( \text{AdOpt} \) play

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

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

\( \psi(t,s) \) does not depend on the configuration

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

\[ \text{Binom}\left(n, k, p\right) = \sum_{j=0}^{[k]} \binom{n}{j} p^j (1 - p)^{n-j} \]
**Bin Potential:** $\psi(t,s)$ - fraction of examples in bin $s$ after iteration $t$ on which the final majority rule will be incorrect assuming AdOpt play

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

$\Psi(t) = \sum_i f(t,s)\psi(t,s)$

$\psi(t,s)$ does not depend on the configuration

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

$\psi(t - 1,s) = \left(\frac{1}{2} - \gamma\right)\psi(t, s - 1) + \left(\frac{1}{2} + \gamma\right)\psi(t, s + 1)$
**Bin Potential:** $\psi(t,s) -$ fraction of examples in bin $s$ after iteration $t$ on which the final majority rule will be incorrect assuming AdOpt play

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

$$
\Psi(t) = \sum_{i} f(t,s) \psi(t,s)
$$

$\psi(t,s)$ does not depend on the configuration

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

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

$$
\psi(T,s) = \begin{cases} 
0 & s > 0 \\
1 & s \leq 0 
\end{cases}
$$
**Bin Potential:** $\psi(t,s)$ - fraction of examples in bin $s$ after iteration $t$ on which the final majority rule will be incorrect assuming AdOpt play

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

$\Psi(t) = \sum_i f(t,s)\psi(t,s)$

$\psi(t,s)$ does not depend on the configuration

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

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

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

\[ d(t,s) = \mu(h_t(x) = y | (x,y) \text{ in bin } s \text{ after iteration } t) - \left( \frac{1}{2} + \gamma \right) \]
Definitions

\[ d(t,s) \doteq \mu(h_t(x) = y|(x,y) \text{ in bin } s \text{ after iteration } t) - \left( \frac{1}{2} + \gamma \right) \]

\[ w(t,s) \doteq \psi(t + 1, s - 1) - \psi(t + 1, s + 1) \]
Definitions

\[ d(t,s) \doteq \mu(h_t(x) = y \mid (x,y) \text{ in bin } s \text{ after iteration } t) - \left( \frac{1}{2} + \gamma \right) \]

\[ w(t,s) \doteq \psi(t + 1, s - 1) - \psi(t + 1, s + 1) \]

Theorem

If \[ \sum_{s} d(t,s)w(t,s) \geq 0 \] then \[ \Psi(t + 1) \leq \Psi(t) \]
Definitions

\[ d(t,s) \doteq \mu\left(h_t(x) = y \mid (x,y) \text{ in bin } s \text{ after iteration } t\right) - \left(\frac{1}{2} + \gamma\right) \]

\[ w(t,s) \doteq \psi(t+1,s-1) - \psi(t+1,s+1) \]

Theorem

If \( \sum_{s} d(t,s)w(t,s) \geq 0 \) then \( \Psi(t+1) \leq \Psi(t) \)

Corollary

If \( \forall t \) [weighted error of \( h_t(x) \)] \( \leq 1/2 - \gamma \)
Then
Initial potential \( \geq \) final training error.
Proof of Theorem
Proof of Theorem

\[ f(t + 1, s) = f(t, s - 1) \left( \frac{1}{2} + \gamma + d(t, s - 1) \right) + f(t, s + 1) \left( \frac{1}{2} - \gamma - d(t, s + 1) \right) \]
Proof of Theorem

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

\[ \Psi(t + 1) = \sum_s \left[ f(t, s - 1)\left(\frac{1}{2} + \gamma + d(t, s - 1)\right) + f(t, s + 1)\left(\frac{1}{2} - \gamma - d(t, s + 1)\right) \right] \psi(t + 1, s) \]
Proof of Theorem

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

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

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

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

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

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

\[
\begin{align*}
f(t+1,s) &= f(t,s-1)\left(\frac{1}{2} + \gamma + d(t,s-1)\right) + f(t,s+1)\left(\frac{1}{2} - \gamma - d(t,s+1)\right) \\
\Psi(t+1) &= \sum_s \left[ f(t,s-1)\left(\frac{1}{2} + \gamma + d(t,s-1)\right) + f(t,s+1)\left(\frac{1}{2} - \gamma - d(t,s+1)\right) \right] \psi(t+1,s) \\
\Psi(t+1) &= \sum_s \left[ f(t,s) \left(\frac{1}{2} + \gamma \right) \psi(t+1,s+1) + \left(\frac{1}{2} - \gamma \right) \psi(t+1,s-1) \right] - d(t,s) \left(\psi(t+1,s-1) - \psi(t+1,s+1)\right) \\
\psi(t,s) &= \left(\frac{1}{2} + \gamma \right) \psi(t+1,s+1) + \left(\frac{1}{2} - \gamma \right) \psi(t+1,s-1) \\
w(t,s) &= \psi(t+1,s-1) - \psi(t+1,s+1)
\end{align*}
\]
Proof of Theorem

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

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

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

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

\[ w(t, s) = \psi(t+1, s-1) - \psi(t+1, s+1) \]

\[ \Psi(t+1) = \sum_s \left[ f(t, s) \psi(t, s) + d(t, s) w(t, s) \right] = \Psi(t) - \sum_s d(t, s) w(t, s) \]
Proof of Theorem

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

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

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

\[ w(t,s) = \psi(t+1,s-1) - \psi(t+1,s+1) \]

\[ \Psi(t+1) = \sum_s [f(t,s)\psi(t,s) + d(t,s)w(t,s)] = \Psi(t) - \sum_s d(t,s)w(t,s) \]

If \( \sum_s d(t,s)w(t,s) \geq 0 \) then \( \Psi(t+1) \leq \Psi(t) \)
Theorem about BBM
Theorem about BBM

setting the boosting weights at iteration $t$ to be

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

setting the boosting weights at iteration \( t \) to be

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

guarantees

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

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

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

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

- The worst case adversary splits each bin into:
  \[ \frac{1}{2}^- \gamma \text{ incorrect} / \frac{1}{2}^+ \gamma \text{ correct} \]
High level summary

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

- Alternative interpretation: Random walk with IID steps.
High level summary

• The worst case adversary splits each bin into:
  \[ \frac{1}{2} - \gamma \text{ incorrect} / \frac{1}{2} + \gamma \text{ correct} \]

• 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?
Why is BBM not practical?

- BBM needs to know $\epsilon, \gamma$ before starting.
Why is BBM not practical?

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

$$T = \frac{1}{\gamma^2} \ln \frac{1}{\varepsilon}$$
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, $\gamma_1, \gamma_2, \gamma_3, \ldots$. 

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.
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, $\gamma_1$, $\gamma_2$, $\gamma_3$, ....

• No need to set parameters in advance.

• Generates a weighted majority rule.
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.
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.
Letting time step decrease to zero.

- Number of iterations required by BBM: \[ T = \frac{1}{\gamma^2 \ln \frac{1}{\epsilon}} \]
Letting time step decrease to zero.

- Number of iterations required by BBM: \[ T = \frac{1}{\gamma^2} \ln \frac{1}{\epsilon} \]
- Keep \( \epsilon \) fixed and let \( \gamma \to 0, T \to \infty \)
Letting time step decrease to zero.

- Number of iterations required by BBM: $T = \frac{1}{\gamma^2 \ln \frac{1}{\epsilon}}$

- Keep $\epsilon$ fixed and let $\gamma \to 0, T \to \infty$

- The same weak rule is added many times, until it’s advantage falls below $\gamma$. 

$T = \frac{1}{\gamma^2 \ln \frac{1}{\epsilon}}$
Letting time step decrease to zero.

- Number of iterations required by BBM: \( T = \frac{1}{\gamma^2 \ln \frac{1}{\epsilon}} \)

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

- The same weak rule is added many times, until it’s advantage falls below \( \gamma \).

- yields adaptive boosting and a weighted majority rule.
Letting time step decrease to zero.

- Number of iterations required by BBM: $T = \frac{1}{\gamma^2} \ln \frac{1}{\epsilon}$
- Keep $\epsilon$ fixed and let $\gamma \to 0, T \to \infty$
- 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.
Letting time step decrease to zero.

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

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

- 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.

- Instead of \( t=1,2,...,T \) use \( t=1/T,2/T,...,1 \)
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 \[ \Delta s = \pm \frac{1}{T} \]

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

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

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

\[ T = 9, \Delta t = \frac{1}{9}, \Delta s = \pm \frac{1}{9} \]
Using step $\Delta s = \pm \frac{1}{T}$

$$T = 9, \, \Delta t = \frac{1}{9}, \, \Delta s = \pm \frac{1}{9}$$
Using step \( \Delta s = \pm \frac{1}{T} \)

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

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

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

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

$T = 1, \Delta t = 1, \Delta s = \pm 1$

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

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

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

\[ T = 9, \Delta t = \frac{1}{9}, \Delta s = \pm \frac{1}{3} \]
Using step \( \Delta s = \pm \frac{1}{\sqrt{T}} \)

\[ T = 9, \Delta t = \frac{1}{9}, \Delta s = \pm \frac{1}{3} \]
Using step \( \Delta s = \pm \frac{1}{\sqrt{T}} \)

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

\[
\text{var}(s) = 9 \cdot \frac{1}{9} = 1
\]
Using step \( \Delta s = \pm \frac{1}{\sqrt{T}} \)

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

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

- **Discrete time:** Equations relating time $t$ to time $t+1$ based on random walks.
Potentials in continuous time

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

- **Continuous time:** Differential Equations describing the density evolution for Brownian motion with drift (known as the Kolmogorov forward and backward equations).
Example: From BBM to Brownboost
Example: From BBM to Brownboost

Potential function for BBM:
Example: From BBM to Brownboost

Potential function for BBM:

$$\psi(t-1, s) = \left(\frac{1}{2} - \gamma\right)\psi(t, s-1) + \left(\frac{1}{2} + \gamma\right)\psi(t, s+1)$$
Example: From BBM to Brownboost

Potential function for BBM:

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

\[ \psi(T,s) = \begin{cases} 
0 & s > 0 \\
1 & s \leq 0 
\end{cases} \]
Example: From BBM to Brownboost

Potential function for BBM:

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

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

\[\varepsilon = \psi(0,0) = \text{Binom}\left(T, \frac{T}{2}, \frac{1}{2} + \gamma\right)\]
Example: From BBM to Brownboost

Potential function for BBM:

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

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

Potential function for Brownboost:
Example: From BBM to Brownboost

Potential function for BBM:

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

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

\[
\epsilon = \psi(0,0) = \text{Binom} \left( T, \frac{T}{2}, \frac{1}{2} + \gamma \right)
\]

Potential function for Brownboost:

\[
\frac{\partial}{\partial t} \psi(t,s) = -\frac{1}{2} \frac{\partial^2}{\partial s^2} \psi(t,s) - 2\sqrt{\beta} \frac{\partial}{\partial s} \psi(t,s)
\]
Example: From BBM to Brownboost

Potential function for BBM:

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

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

Potential function 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:
Example: From BBM to Brownboost

Potential function for BBM:

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

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

\[ \varepsilon = \psi(0,0) = \text{Binom}\left( T, \frac{T}{2}, 1 + \gamma \right) \]

Potential function 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:

\[ \psi(1,s) = \begin{cases} 0 & s > 0 \\ 1 & s \leq 0 \end{cases} \]
Example: From BBM to Brownboost

Potential function for BBM:

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

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

\[
\epsilon = \psi(0,0) = \text{Binom}\left( T, \frac{T}{2}, \frac{1}{2} + \gamma \right)
\]

Potential function 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:

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

\[
\epsilon = \psi(0,0)
\]
Plan of talk

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

- Instead of minimizing training error, minimize number of examples whose margin $\leq \theta$. 
Robustboost

- Instead of minimizing training error, minimize number of examples whose margin $\leq \theta$.
- Control weights of base rules by restricting variance of score (instead of range).
Robustboost

- Instead of minimizing training error, minimize number of examples whose margin $\leq \theta$.
- Control weights of base rules by restricting variance of score (instead of range).
- Allow confidence-rated weak learners
Robustboost

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

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

$$\mu(t) = (\theta - 2\rho)e^{1-t} + 2\rho \quad \sigma^2(t) = \left( \sigma_f^2 + 1 \right)e^{2(1-t)} - 1$$

set $\rho$ to satisfy $\varepsilon = \psi(0,0) = 1 - \text{erf} \left( \frac{2(e-1)\rho - e\theta}{\sqrt{e^2 \left( \sigma_f^2 + 1 \right) - 1}} \right)$
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
LogitBoost on Long/Servedio

![Graph showing t=10](image)
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% |

Tuesday, June 16, 2009
Logitboost
0% Noise
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.
For more details see my publications page

1. Yoav Freund  
   A more robust boosting algorithm.  
   May 2009. Arxiv/0905.2138

2. Kamalika Chaudhuri, Yoav Freund, Daniel Hsu  
   A parameter-free hedging algorithm.  
   May 2009. Arxiv/0903.2851

3. Kamalika Chaudhuri, Yoav Freund, Daniel Hsu  
   Tracking using explanation-based modeling.  
   May 2009. Arxiv/0903.2862

4. Yoav Freund  
   A method for Hedging in continuous time.  
   May 2009. Arxiv/0904.3356
For more details see my publications page

1. Yoav Freund
   A more robust boosting algorithm.
   May 2009. Arxiv/0905.2138

2. Kamalika Chaudhuri, Yoav Freund, Daniel Hsu
   A parameter-free hedging algorithm.
   May 2009. Arxiv/0903.2851

3. Kamalika Chaudhuri, Yoav Freund, Daniel Hsu
   Tracking using explanation-based modeling.
   May 2009. Arxiv/0903.2862

4. Yoav Freund
   A method for Hedging in continuous time.
   May 2009. Arxiv/0904.3356

Rejected from ICML09
1. Yoav Freund
   A more robust boosting algorithm.
   May 2009. Arxiv/0905.2138

2. Kamalika Chaudhuri, Yoav Freund, Daniel Hsu
   A parameter-free hedging algorithm.
   May 2009. Arxiv/0903.2851

3. Kamalika Chaudhuri, Yoav Freund, Daniel Hsu
   Tracking using explanation-based modeling.
   May 2009. Arxiv/0903.2862

4. Yoav Freund
   A method for Hedging in continuous time.
   May 2009. Arxiv/0904.3356
For more details see my publications page

1. Yoav Freund
   A more robust boosting algorithm.
   May 2009. Arxiv/0905.2138

2. Kamalika Chaudhuri, Yoav Freund, Daniel Hsu
   A parameter-free hedging algorithm.
   May 2009. Arxiv/0903.2851

3. Kamalika Chaudhuri, Yoav Freund, Daniel Hsu
   Tracking using explanation-based modeling.
   May 2009. Arxiv/0903.2862

4. Yoav Freund
   A method for Hedging in continuous time.
   May 2009. Arxiv/0904.3356

Rejected from ICML09
Rejected from COLT09
Rejected from UAI09
For more details see my publications page

1. Yoav Freund  
   A more robust boosting algorithm.  
   May 2009. Arxiv/0905.2138  
   Rejected from ICML09

2. Kamalika Chaudhuri, Yoav Freund, Daniel Hsu  
   A parameter-free hedging algorithm.  
   May 2009. Arxiv/0903.2851  
   Rejected from COLT09

3. Kamalika Chaudhuri, Yoav Freund, Daniel Hsu  
   Tracking using explanation-based modeling.  
   May 2009. Arxiv/0903.2862  
   Rejected from UAI09

4. Yoav Freund  
   A method for Hedging in continuous time.  
   May 2009. Arxiv/0904.3356  
   Not rejected yet
Summary
Summary

• Don’t get too upset if your paper is rejected.
Summary

- Don’t get too upset if your paper is rejected.
- You might present it as an invited speaker!
Summary

- Don’t get too upset if your paper is rejected.
  - You might present it as an invited speaker!
- Try JBoost!
Summary

• Don’t get too upset if your paper is rejected.
  • You might present it as an invited speaker!

• Try JBoost!

• Questions: yfreund@ucsd.edu
Summary

• Don’t get too upset if your paper is rejected.
  • You might present it as an invited speaker!

• Try JBoost!

• Questions: yfreund@ucsd.edu

• Thank you!