A Theory of Multiclass Boosting

Indraneel Mukherjee*, R. E. Schapire
Princeton University
Wrigley Field prepared for college football game

Dublin warned over ECB liquidity

Newest senators Coons and Manchin sworn in
Wrigley Field prepared for college football game

Dublin warned over ECB liquidity

Newest senators Coons and Manchin sworn in
Wrigley Field prepared for college football game

Dublin warned over ECB liquidity

Newest senators Coons and Manchin sworn in
Wrigley Field prepared for college football game

Dublin warned over ECB liquidity

Newest senators Coons and Manchin sworn in
Goals of Boosting
Goals of Boosting

• Boost *simplest* weak classifiers
Goals of Boosting

- Boost *simplest* weak classifiers
- Use right *weak learning condition* (WLC)
Goals of Boosting

• Boost *simplest* weak classifiers
• Use right *weak learning condition (WLC)*
• Important for generalization error:
Goals of Boosting

• Boost *simplest* weak classifiers
• Use right *weak learning condition (WLC)*
• Important for generalization error:
  • Simple weak classifier may imply less overfitting
Goals of Boosting

• Boost *simplest* weak classifiers
• Use right *weak learning condition* (WLC)
• Important for generalization error:
  • Simple weak classifier may imply less overfitting
  • Too simple could lead to underfitting
Goals of Boosting

- Boost *simplest* weak classifiers
- Use right *weak learning condition (WLC)*
- Important for generalization error:
  - Simple weak classifier may imply less overfitting
  - Too simple could lead to underfitting
- Theory known for binary, not for multiclass
This Talk
This Talk

• Existing frameworks inadequate for multiclass
This Talk

• Existing frameworks inadequate for multiclass

• Most resulting WLC’s are *too weak* or *too strong*
This Talk

- Existing frameworks inadequate for multiclass
- Most resulting WLC’s are too weak or too strong
- Introduce new framework for multiclass boosting
This Talk

• Existing frameworks inadequate for multiclass
  
  • Most resulting WLC's are too weak or too strong
  
  • Introduce new framework for multiclass boosting
  
  • Captures the minimal WLC
This Talk

• Existing frameworks inadequate for multiclass
  • Most resulting WLC’s are too weak or too strong
• Introduce new framework for multiclass boosting
  • Captures the minimal WLC
• Boosting algorithm using the minimal WLC
This Talk

- Existing frameworks inadequate for multiclass
  - Most resulting WLC's are *too weak* or *too strong*

- Introduce *new framework for multiclass boosting*
  - Captures the *minimal WLC*

- *Boosting algorithm using the minimal WLC*
  - Provably drives down error efficiently
This Talk

• Existing frameworks inadequate for multiclass
  • Most resulting WLC’s are too weak or too strong
• Introduce new framework for multiclass boosting
  • Captures the minimal WLC
• Boosting algorithm using the minimal WLC
  • Provably drives down error efficiently
• Experiments to complement the theory
Binary boosting
Binary boosting

Input: \((x_1, y_1), \ldots, (x_m, y_m)\)

Booster

\(\mathcal{H} = \{\text{weak classifiers}\}\)
Binary boosting

Input: \((x_1, y_1), \ldots, (x_m, y_m)\)

\(d_1, \ldots, d_m\)

Booster \(\mathcal{H} = \{\text{weak classifiers}\}\)

\(h \in \mathcal{H}, \ h: \{\text{Example}\} \Rightarrow \{\text{Label}\}\)
Binary boosting

Input: \((x_1, y_1), \ldots, (x_m, y_m)\)

\(d_1, \ldots, d_m\)

\(h \in \mathcal{H}, h: \{Example\} \Rightarrow \{Label\}\)

Condition: \(\hat{\text{err}}_d(h) \leq \frac{1}{2} - \gamma\)
Binary boosting

Input: \((x_1, y_1), \ldots, (x_m, y_m)\)

\(d_1, \ldots, d_m\)

\(H = \{\text{weak classifiers}\}\)

\(h \in H, h: \{\text{Example}\} \Rightarrow \{\text{Label}\}\)

Condition: \(\hat{\text{err}}_d(h) \leq \frac{1}{2} - \gamma\)

Binary WLC
Binary boosting

Input: \((x_1, y_1), \ldots, (x_m, y_m)\)

d_1, \ldots, d_m

\(\mathcal{H} = \{\text{weak classifiers}\}\)

\(h \in \mathcal{H}, \; h : \{\text{Example}\} \rightarrow \{\text{Label}\}\)

Condition: \(\widehat{\text{err}}_d(h) \leq \frac{1}{2} - \gamma\)

Final model: (weighted) majority\(\{h_1, \ldots, h_T\}\)
Binary boosting

**Input:** \((x_1, y_1) , \ldots , (x_m , y_m)\)

\(d_1 , \ldots , d_m\)

**Booster**

**\(\mathcal{H} = \{\text{weak classifiers}\}\)**

**Condition:** \(\hat{\text{err}}_d(h) \leq \frac{1}{2} - \gamma\)

**Final model:** (weighted) majority\{h_1, \ldots , h_T\}

More weight on misclassified examples
Binary boosting

Input: \((x_1, y_1), \ldots, (x_m, y_m)\)

\(d_1, \ldots, d_m\)

\(h \in \mathcal{H}, \ h:\{\text{Example}\} \Rightarrow \{\text{Label}\}\)

Condition: \(\hat{\text{err}}_d(h) \leq \frac{1}{2} - \gamma\)

Final model: (weighted) majority\(\{h_1, \ldots, h_T\}\)

After T rounds, \(\hat{\text{err}}\) of \(\text{maj}\{h_1, \ldots, h_T\} \leq \exp(-T\gamma^2/2)\)

More weight on misclassified examples

\(\mathcal{H} = \{\text{weak classifiers}\}\)
Binary WLC Ideal
Binary WLC Ideal

• *Required tasks easy.* Only better than random
Binary WLC Ideal

- Required tasks easy. Only better than random
- Sufficient. $H$ satisfies binary WLC $\Rightarrow H$ is boostable
Binary WLC Ideal

- Required tasks easy. Only better than random
- Sufficient. $\mathcal{H}$ satisfies binary WLC $\Rightarrow \mathcal{H}$ is boostable
  - Boostable space: contains perfect combination
Binary WLC Ideal

- Required tasks easy. Only better than random
- Sufficient. $\mathcal{H}$ satisfies binary WLC $\Rightarrow$ $\mathcal{H}$ is boostable
  - Boostable space: contains perfect combination
- Necessary. Boostable space satisfies binary WLC
Binary WLC Ideal

- **Required tasks easy.** Only better than random
- **Sufficient.** $\mathcal{H}$ satisfies binary WLC $\Rightarrow$ $\mathcal{H}$ is boostable
  - Boostable space: contains perfect combination
- **Necessary.** Boostable space satisfies binary WLC
- **Effective.** Allows efficient boosting algorithm
Extending to Multiclass
Extending to Multiclass

Input: \((x_1, y_1), \ldots, (x_m, y_m)\)

d_1, \ldots, d_m

Booster

\(H = \{\text{weak classifiers}\}\)

h ∈ H, h: \{Example\} \Rightarrow \{\text{Multiclass Label}\}
Extending to Multiclass

Input: \((x_1, y_1), \ldots, (x_m, y_m)\)

d_1, \ldots, d_m

Booster

\(\mathcal{H} = \{\text{weak classifiers}\}\)

\(h \in \mathcal{H}, h: \{\text{Example}\} \Rightarrow \{\text{Multiclass Label}\}\)

\(\hat{\text{err}}_d(h) \leq 1 - \frac{1}{k} - \gamma\)

SAMME [Zhu, Zou, Rosset, Hastie ‘09]
Extending to Multiclass

Input: \((x_1, y_1), \ldots, (x_m, y_m)\)

d_1, \ldots, d_m

Booster

\(H = \{\text{weak classifiers}\}\)

\(h \in H, h: \{\text{Example}\} \Rightarrow \{\text{Multiclass Label}\}\)

\(\hat{err}_d(h) \leq 1 - \frac{1}{k} - \gamma\)

Too weak

SAMME [Zhu, Zou, Rosset, Hastie '09]
Extending to Multiclass

Input: \((x_1, y_1), \ldots, (x_m, y_m)\)

\(d_1, \ldots, d_m\)

Booster

\(H = \{\text{weak classifiers}\}\)

\(h \in H, h: \{\text{Example}\} \rightarrow \{\text{Multiclass Label}\}\)

\(\hat{err}_d(h) \leq 1 - \frac{1}{k} - \gamma\)

\(\hat{err}_d(h) \leq \frac{1}{2} - \gamma\)

Too weak

SAMME [Zhu, Zou, Rosset, Hastie '09]

AdaBoost.M1 [Freund, Schapire '96]
Extending to Multiclass

Input: \((x_1, y_1), \ldots, (x_m, y_m)\)

\[ d_1, \ldots, d_m \]

Booster \(\mathcal{H} = \{\text{weak classifiers}\}\)

\[ h \in \mathcal{H}, \ h: \{\text{Example}\} \Rightarrow \{\text{Multiclass Label}\} \]

\[ \hat{\text{err}}_d(h) \leq 1 - \frac{1}{k} - \gamma \]

SAMME [Zhu, Zou, Rosset, Hastie ‘09]

Too weak

\[ \hat{\text{err}}_d(h) \leq \frac{1}{2} - \gamma \]

AdaBoost.M1 [Freund, Schapire ‘96]

Too strong
Reduction to binary

Artificial binary problems

Diagram:
- Multiclass
  - Binary 1
    - Classifier 1
  - Binary 2
    - Classifier 2
  - Binary 3
    - Classifier 3
- Final Classifier

Boosting arrows:
- Binary 1 to Classifier 1
- Binary 2 to Classifier 2
- Binary 3 to Classifier 3
Reduction to binary

Artificial binary problems

• One-against-all, all-pairs, ECOC. E.g.
Reduction to binary

Artificial binary problems

• One-against-all, all-pairs, ECOC. E.g.
  • One-against-all: (AdaBoost.MH) [Schapire & Singer ‘99]
Reduction to binary

Artificial binary problems

• One-against-all, all-pairs, ECOC. E.g.
  • One-against-all: (AdaBoost.MH)[Schapire & Singer ‘99]
  • All-pairs: (AdaBoost.MR)[Freund & Schapire ’96, Schapire & Singer ‘99]
Reduction to binary

Artificial binary problems

- One-against-all, all-pairs, ECOC. E.g.
  - One-against-all: (AdaBoost.MH) [Schapire & Singer ‘99]
  - All-pairs: (AdaBoost.MR) [Freund & Schapire ’96, Schapire & Singer ‘99]
- Practical, but poorly understood
Reduction to binary

Artificial binary problems

• One-against-all, all-pairs, ECOC. E.g.
  • One-against-all: (AdaBoost.MH)[Schapire & Singer ‘99]
  • All-pairs: (AdaBoost.MR)[Freund & Schapire ’96, Schapire & Singer ‘99]
• Practical, but poorly understood
• Sometimes too strong
Reduction to binary

Artificial binary problems

- One-against-all, all-pairs, ECOC. E.g.
  - One-against-all: (AdaBoost.MH)[Schapire & Singer ‘99]
  - All-pairs: (AdaBoost.MR)[Freund & Schapire ’96, Schapire & Singer ‘99]
- Practical, but poorly understood
- Sometimes too strong
  - e.g. One-against-all (AdaBoost.MH)
New Framework (simplified)
New Framework \textit{(simplified)}

- Booster sends \textit{cost matrix} $C$, not distribution
New Framework (simplified)

- Booster sends *cost matrix* $C$, not distribution
  - $C(i, \ell)$: cost of predicting label $\ell$ on example $i$
  - $\text{Cost}(C, h) = \sum_i C(i, h(x_i))$
New Framework *(simplified)*

- Booster sends *cost matrix* $C$, not distribution
  
  $C(i, \ell)$: cost of predicting label $\ell$ on example $i$

- **Cost**($C$, $h$) = $\sum_i C(i, h(x_i))$

- Perform as well as fixed *baseline* predictor $B$
New Framework  *(simplified)*

- Booster sends *cost matrix* $C$, not distribution
  - $C(i, \ell)$: cost of predicting label $\ell$ on example $i$
  - $\text{Cost}(C, h) = \sum_i C(i, h(x_i))$
- Perform as well as fixed *baseline* predictor $B$
  - $B(i, \ell)$: prob. with which $B$ predicts $\ell$ on $i$
  - $\text{Cost}(C, B) = \sum_i \mathbb{E}[C(i, B(x_i))] = \sum_i \sum_\ell C(i, \ell) B(i, \ell)$
New Framework \textit{(simplified)}

- Booster sends \textit{cost matrix} $C$, not distribution
  - $C(i, \ell)$: cost of predicting label $\ell$ on example $i$
  - Cost($C$, $h$) = $\sum_i C(i, h(x_i))$

- Perform as well as fixed \textit{baseline} predictor $B$
  - $B(i, \ell)$: prob. with which $B$ predicts $\ell$ on $i$
  - Cost($C$, $B$) = $\sum_i \mathbb{E}[C(i, B(x_i))] = \sum_i \sum_\ell C(i, \ell) B(i, \ell)$

- \textit{Restriction}: Cost ($C$, $h$) $\leq$ Cost ($C$, $B$)
New Framework  (simplified)
New Framework \textit{(simplified)}

Parameter: Fixed baseline B
New Framework (simplified)

Parameter: Fixed baseline B

\( \text{Booster} \quad \text{cost matrix } C \quad \mathcal{H} = \{\text{weak classifiers}\} \)

\( h \in \mathcal{H}, \ h: \{\text{Example}\} \Rightarrow \{\text{Label}\} \)
New Framework \textit{(simplified)}

Parameter: Fixed baseline $B$

$$C_{\text{Cost}}(C, h) \leq C_{\text{Cost}}(C, B)$$
Binary Boosting

\[ B(i, \ell) = \begin{cases} 
\frac{1}{2} + \gamma & \text{if } \ell \text{ correct} \\
\frac{1}{2} - \gamma & \text{if } \ell \text{ wrong}
\end{cases} \]

Cost matrix \( C \)

\[ \mathcal{H} = \{ \text{weak classifiers} \} \]

Booster

\[ h \in \mathcal{H}, h: \{ \text{Example} \} \Rightarrow \{ \text{Label} \} \]

\[ \text{Cost}(C, h) \leq \text{Cost}(C, B) \]
Edge-over-random WLC
Edge-over-random WLC

- Edge-over-random baseline $Q$
Edge-over-random WLC

- Edge-over-random baseline $Q$
  
  - $B(i, \text{correct}) \geq B(i, \text{wrong}) + 2\gamma$
  
  - $B(i, \cdot)$ is a distribution
Edge-over-random WLC

- Edge-over-random baseline $Q$
  - $B(i, \text{correct}) \geq B(i, \text{wrong}) + 2\gamma$
  - $B(i, \cdot)$ is a distribution
- Many choices for $B$ (only one for binary)
Edge-over-random WLC

- Edge-over-random baseline $Q$
  - $B(i, \text{correct}) \geq B(i, \text{wrong}) + 2\gamma$
  - $B(i, \cdot)$ is a distribution

- Many choices for $B$ (only one for binary)
- *Condition with such baseline:*
Edge-over-random WLC

• Edge-over-random baseline $Q$
  
  - $B(i, \text{correct}) \geq B(i, \text{wrong}) + 2\gamma$
  - $B(i, \cdot)$ is a distribution

• Many choices for $B$ (only one for binary)

• *Condition with such baseline:*

  Edge-over-random WLC
EOR nearly Ideal
EOR nearly Ideal

- *Required tasks easy. Only beat random*
EOR nearly Ideal

- Required tasks easy. Only beat random
- Sufficient. Satisfying EOR implies boostability
EOR nearly Ideal

- Required tasks easy. Only beat random
- Sufficient. Satisfying EOR implies boostability
- Effective. Allows efficient boosting
EOR nearly Ideal

• Required tasks easy. Only beat random
• Sufficient. Satisfying EOR implies boostability
• Effective. Allows efficient boosting
• Not Necessary. For any EOR (B), there is some boostable space $\mathcal{H}$ that does not satisfy it.
EOR nearly Ideal

- **Required tasks easy.** Only beat random
- **Sufficient.** Satisfying EOR implies boostability
- **Effective.** Allows efficient boosting
- **Not Necessary.** For any EOR \((B)\), there is some boostable space \(H\) that does not satisfy it.

- **Relaxed necessity.** For any boostable space \(H\), there is some EOR \((B)\) that \(H\) satisfies
EOR nearly Ideal

- **Required tasks easy.** Only beat random
- **Sufficient.** Satisfying EOR implies boostability
- **Effective.** Allows efficient boosting
- **Not Necessary.** For any EOR \((B)\), there is some boostable space \(\mathcal{H}\) that does not satisfy it.

- **Relaxed necessity.** For any boostable space \(\mathcal{H}\), there is some EOR \((B)\) that \(\mathcal{H}\) satisfies

- Combine to form **single minimal WLC**
EOR nearly Ideal

- *Required tasks easy.* Only beat random
- *Sufficient.* Satisfying EOR implies boostability
- *Effective.* Allows efficient boosting
- *Not Necessary.* For any EOR \((B)\), there is some boostable space \(\mathcal{H}\) that does not satisfy it.
- *Relaxed necessity.* For any boostable space \(\mathcal{H}\), there is some EOR \((B)\) that \(\mathcal{H}\) satisfies
- Combine to form *single minimal WLC*
  - *Necessary and sufficient for boostability*
Boosting Algorithms
Boosting Algorithms

- Optimally efficient algorithm for any fixed EOR
Boosting Algorithms

- *Optimally efficient* algorithm for any fixed EOR
- Like Boost-by-majority [Freund ‘95]
Boosting Algorithms

• *Optimally efficient* algorithm for any fixed EOR
• Like Boost-by-majority [Freund ‘95]
• Non-adaptive. Requires knowledge of $\gamma$
Boosting Algorithms

- **Optimally efficient** algorithm for any fixed EOR
- Like Boost-by-majority [Freund ‘95]
- Non-adaptive. Requires knowledge of $\gamma$
- **Adaptive algorithm assuming the minimal WLC**
Boosting Algorithms

- **Optimally efficient** algorithm for any fixed EOR
- Like Boost-by-majority [Freund ‘95]
- Non-adaptive. Requires knowledge of $\gamma$
- *Adaptive* algorithm *assuming the minimal WLC*
- Based on multiplicative updates, like AdaBoost
Boosting Algorithms

- **Optimally efficient** algorithm for any fixed EOR
- Like Boost-by-majority [Freund ‘95]
- Non-adaptive. Requires knowledge of $\gamma$
- **Adaptive** algorithm *assuming the minimal WLC*
- Based on multiplicative updates, like AdaBoost
- Not optimal, but still *provably very efficient*
Adaptive Algorithm
Adaptive Algorithm

• In each round $t$: 
Adaptive Algorithm

- In each round $t$:
  - Create cost matrix $C_t$
Adaptive Algorithm

- In each round $t$:
  - Create cost matrix $C_t$
  - Receive weak classifier $h_t$ with edge $\delta_t$
Adaptive Algorithm

• In each round $t$:
  • Create cost matrix $C_t$
  • Receive weak classifier $h_t$ with edge $\delta_t$
  • Compute weight $\alpha_t$ and update $f_t = f_{t-1} + \alpha_t h_t$
Adaptive Algorithm

• In each round $t$:
  • Create cost matrix $C_t$
  • Receive weak classifier $h_t$ with edge $\delta_t$
  • Compute weight $\alpha_t$ and update $f_t = f_{t-1} + \alpha_t h_t$

Weight

$$\alpha_t = \ln \left\{ \frac{1 + \delta_t}{1 - \delta_t} \right\}$$

Cost Matrix

$$C_{t+1}(i, l) = \begin{cases}  
e f_t(i, l) - f_t(i, y_i) & \text{if } l \neq y_i \\ - \sum_{l \neq y_i} e f_t(i, l') - f_t(i, y_i) & \text{if } l = y_i \end{cases}$$
Experiments

• Ran adaptive algorithm using minimal WLC
• Compared with AdaBoost.M1, AdaBoost.MH
• Tested on benchmark datasets
• Weak classifiers: bounded size decision trees
Future Work
Future Work

• What happens with multi-label / confidence rated weak classifiers?
Future Work

• What happens with multi-label / confidence rated weak classifiers?

• Consistency of the algorithms.
Future Work

• What happens with multi-label / confidence rated weak classifiers?
• Consistency of the algorithms.
• Extensions to ranking.
Thank you