Combining Ordinal Preferences by Boosting

Hsuan-Tien Lin and Ling Li

National Taiwan University/California Institute of Technology

Preference Learning Workshop, September 12, 2009
Ordinal Ranking Setup

Hot or Not?

http://www.hotornot.com

Select a rating to see the next picture.

Show me men and women ages 18-25

rank: representing human preferences by a finite ordered set of labels $\mathcal{Y} = \{1, 2, \cdots, K\}$
Hot or Not?

http://www.hotornot.com

Select a rating to see the next picture.

NOT 1 2 3 4 5 6 7 8 9 10 HOT

Show me men and women ages 18-25

rank: representing human preferences by a finite ordered set of labels $\mathcal{Y} = \{1, 2, \cdots, K\}$
How Much Did You Like These Movies?

http://www.netflix.com

Get Recommendations (27)  Rate Movies  Movies You've Rated (5)

How much did you like these movies?

<table>
<thead>
<tr>
<th>The Wedding Planner</th>
<th>How to Lose a Guy in 10 Days</th>
<th>Sweet Home Alabama</th>
<th>Pretty Woman</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="The Wedding Planner" /></td>
<td><img src="image2" alt="How to Lose a Guy in 10 Days" /></td>
<td><img src="image3" alt="Sweet Home Alabama" /></td>
<td><img src="image4" alt="Pretty Woman" /></td>
</tr>
</tbody>
</table>

goal: use “movies you’ve rated” to automatically predict your preferences (ranks) on future movies
Ordinal Ranking Setup

How Much Did You Like These Movies?

http://www.netflix.com

Get Recommendations (27) Rate Movies Movies You've Rated (5)

How much did you like these movies?

The Wedding Planner

How to Lose a Guy in 10 Days

Sweet Home Alabama

Pretty Woman

goal: use “movies you’ve rated” to automatically predict your preferences (ranks) on future movies
ranks represent order information
—general classification cannot property use such

rating 9 “hotter than” rating 8 “hotter than” rating 7

ranks do not carry numerical information
—general regression deteriorates without such

not 2.5 times better than
**Properties of Ordinal Ranking**

- Ranks represent **order** information
  — general classification cannot properly use such information

  ![Rating Scale](image)

  Rating 9 “hotter than” rating 8 “hotter than” rating 7

- Ranks do **not** carry numerical information
  — general regression deteriorates without such information

  ![Rating Scale](image)

  ★★★★★ not 2.5 times better than ★★★★★★
Given

$N$ examples (input $x_n$, rank $y_n) \in \mathcal{X} \times \mathcal{Y}$

- hotornot: $\mathcal{X} = \text{encoding(human pictures)}$, $\mathcal{Y} = \{1, \cdots, 10\}$
- netflix: $\mathcal{X} = \text{encoding(movies/users)}$, $\mathcal{Y} = \{1, \cdots, 5\}$

Goal

an ordinal ranker $r(x)$ that “closely predicts” the ranks $y$ associated with some unseen inputs $x$

no numerical information: how to say “close”?
Ordinal Ranking Setup

**Given**

\( N \) examples (input \( x_n, \text{rank } y_n \) \( \in X \times Y \))

- **hotornot**: \( X = \text{encoding(human pictures)}, Y = \{1, \cdots, 10\} \)
- **netflix**: \( X = \text{encoding(movies/users)}, Y = \{1, \cdots, 5\} \)

**Goal**

an ordinal ranker \( r(x) \) that “closely predicts” the ranks \( y \) associated with some **unseen** inputs \( x \)

no numerical information: how to say “close”?
Ordinal Ranking Setup

Given

$N$ examples $(\text{input } x_n, \text{rank } y_n) \in \mathcal{X} \times \mathcal{Y}$

- hotornot: $\mathcal{X} = \text{encoding(human pictures)}, \mathcal{Y} = \{1, \cdots, 10\}$
- netflix: $\mathcal{X} = \text{encoding(movies/users)}, \mathcal{Y} = \{1, \cdots, 5\}$

Goal

an ordinal ranker $r(x)$ that “closely predicts” the ranks $y$ associated with some unseen inputs $x$

no numerical information: how to say “close”?
Formalizing (Non-)Closeness: Cost

- artificially quantify the **cost** of being wrong

- cost vector $c$ of example $(x, y, c)$:
  $c[k] = $ cost when predicting $(x, y)$ as rank $k$

- or use general cost vectors:

<table>
<thead>
<tr>
<th>$c[k]$</th>
<th>$c[k]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lceil y \neq k \rceil$</td>
<td>$</td>
</tr>
<tr>
<td>classification</td>
<td>absolute</td>
</tr>
<tr>
<td>$(1, 0, 1, 1, 1)$</td>
<td>$(1, 0, 1, 2, 3)$</td>
</tr>
</tbody>
</table>

closely predict: small cost during testing
Formalizing (Non-)Closeness: Cost

- artificially quantify the cost of being wrong
  - e.g. loss of customer loyalty when the recommendation system says ★★★★★★ but you feel ★★★★★☆

- cost vector $c$ of example $(x, y, c)$:
  - $c[k] = \text{cost when predicting } (x, y) \text{ as rank } k$

- or use general cost vectors:
  - $c[k] = \begin{cases} 1, & y \neq k \\ 0, & y = k \\ 1, & y = k + 1 \\ 1, & y = k - 1 \\ 1, & y = k \\ \end{cases}$
  - $c[k] = |y - k|$
  - classification: $(1, 0, 1, 1, 1)$
  - absolute: $(1, 0, 1, 2, 3)$

closely predict: small cost during testing
Formalizing (Non-)Closeness: Cost

- artificially quantify the **cost** of being wrong
  e.g. loss of customer loyalty when the recommendation system says ★★★★★★ but you feel ★★★★★☆

- cost vector \( \mathbf{c} \) of example \((x, y, c)\):
  \[ c[k] = \text{cost when predicting } (x, y) \text{ as rank } k \]
  e.g. for \((\text{Sweet Home Alabama}, ★★★★★☆)\), a customer-oriented cost may be \( c = (1, 0, 2, 10, 15) \)

- or use general cost vectors:
  \[
  \begin{array}{c|c}
  c[k] & y \neq k \\ \hline
  \text{classification} & (1, 0, 1, 1, 1) \\
  \text{absolute} & (1, 0, 1, 2, 3)
  \end{array}
  \]
  closely predict: small cost during testing
Formalizing (Non-)Closeness: Cost

- artificially quantify the **cost** of being wrong
  
e.g. loss of customer loyalty when the recommendation system says ★★★★★★ but you feel ★★★★★★

- cost vector \( \mathbf{c} \) of example \((x, y, c)\):
  
  \[ c[k] = \text{cost when predicting } (x, y) \text{ as rank } k \]

  e.g. for (Sweet Home Alabama, ★★★★★★), a customer-oriented cost may be \( \mathbf{c} = (1, 0, 2, 10, 15) \)

- or use general cost vectors:

  \[
  \begin{align*}
  c[k] &= \mathbb{1}[y \neq k] \\
  c[k] &= |y - k|
  \end{align*}
  \]

  classification absolute

  \[
  (1, 0, 1, 1, 1) \quad (1, 0, 1, 2, 3)
  \]

  closely predict: small cost during testing
artificially quantify the cost of being wrong

e.g. loss of customer loyalty when the recommendation system says ★★★★★★ but you feel ★★★★★★

cost vector \( c \) of example \((x, y, c)\):

\[ c[k] = \text{cost when predicting } (x, y) \text{ as rank } k \]

e.g. for \((\text{Sweet Home Alabama}, ★★★★★★)\),
a customer-oriented cost may be \( c = (1, 0, 2, 10, 15) \)

or use general cost vectors:

\[
\begin{align*}
\text{classification} & \quad \text{absolute} \\
(1, 0, 1, 1, 1) & \quad (1, 0, 1, 2, 3)
\end{align*}
\]

closely predict: small cost during testing
some simple ordinal rankers that predict your preference on movies:

- $r_1(x)$ = a ranker based on actor performance
- $r_2(x)$ = a ranker based on actress performance
- $r_3(x)$ = a ranker based on an expert opinion
- $r_4(x)$ = a ranker based on box reports

no single ranker can explain your preference well, but an ensemble combination of them possibly can
some simple ordinal rankers that predict your preference on movies:

- \( r_1(x) \) = a ranker based on actor performance
- \( r_2(x) \) = a ranker based on actress performance
- \( r_3(x) \) = a ranker based on an expert opinion
- \( r_4(x) \) = a ranker based on box reports

no single ranker can explain your preference well, but an ensemble combination of them possibly can

how to construct a good ordinal ensemble?
some simple ordinal rankers that predict your preference on movies:

- \( r_1(x) \) = a ranker based on actor performance
- \( r_2(x) \) = a ranker based on actress performance
- \( r_3(x) \) = a ranker based on an expert opinion
- \( r_4(x) \) = a ranker based on box reports

no single ranker can explain your preference well, but an ensemble combination of them possibly can
Our Contributions

an algorithmic and theoretical development on ensemble learning for ordinal ranking, which ...

- extends AdaBoost to ordinal ranking: can construct ordinal ensemble from any (possibly application-specific) cost
- introduces new theoretical guarantee on the performance of ordinal ensemble
- leads to good experimental results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Base Ranker</th>
<th>Final Ranker</th>
</tr>
</thead>
<tbody>
<tr>
<td>RankBoost</td>
<td>real (pairwise)</td>
<td>real (pairwise)</td>
</tr>
<tr>
<td>ORBoost (Lin and Li, ALT ’06)</td>
<td>real (binary)</td>
<td>ordinal</td>
</tr>
<tr>
<td>AdaBoost.OR</td>
<td>ordinal</td>
<td>ordinal</td>
</tr>
</tbody>
</table>

H.-T. Lin and L. Li (NTU/Caltech)
Our Contributions

An algorithmic and theoretical development on ensemble learning for ordinal ranking, which ...

- extends AdaBoost to ordinal ranking: can construct ordinal ensemble from any (possibly application-specific) cost
- introduces new theoretical guarantee on the performance of ordinal ensemble
- leads to good experimental results

<table>
<thead>
<tr>
<th>algorithm</th>
<th>base ranker</th>
<th>final ranker</th>
</tr>
</thead>
<tbody>
<tr>
<td>RankBoost (Freund et al., JMLR '03)</td>
<td>real (pairwise)</td>
<td>real (pairwise)</td>
</tr>
<tr>
<td>ORBoost (Lin and Li, ALT '06)</td>
<td>real (binary)</td>
<td>ordinal</td>
</tr>
<tr>
<td>AdaBoost.OR</td>
<td>ordinal</td>
<td>ordinal</td>
</tr>
</tbody>
</table>
Our Contributions

an algorithmic and theoretical development on ensemble learning for ordinal ranking, which ...

- extends AdaBoost to ordinal ranking: can construct ordinal ensemble from any (possibly application-specific) cost
- introduces new theoretical guarantee on the performance of ordinal ensemble
- leads to good experimental results

<table>
<thead>
<tr>
<th>algorithm</th>
<th>base ranker</th>
<th>final ranker</th>
</tr>
</thead>
<tbody>
<tr>
<td>RankBoost</td>
<td>real (pairwise)</td>
<td>real (pairwise)</td>
</tr>
<tr>
<td>ORBoost</td>
<td>real (binary)</td>
<td>ordinal</td>
</tr>
<tr>
<td>AdaBoost.OR</td>
<td>ordinal</td>
<td>ordinal</td>
</tr>
</tbody>
</table>

H.-T. Lin and L. Li (NTU/Caltech)  Combining Ordinal Preferences by Boosting  2009/09/12
Our Contributions

an algorithmic and theoretical development on ensemble learning for ordinal ranking, which ...

- extends AdaBoost to ordinal ranking: can construct ordinal ensemble from any (possibly application-specific) cost
- introduces new theoretical guarantee on the performance of ordinal ensemble
- leads to good experimental results

<table>
<thead>
<tr>
<th>algorithm</th>
<th>base ranker</th>
<th>final ranker</th>
</tr>
</thead>
<tbody>
<tr>
<td>RankBoost (Freund et al., JMLR ’03)</td>
<td>real (pairwise)</td>
<td>real (pairwise)</td>
</tr>
<tr>
<td>ORBoost (Lin and Li, ALT ’06)</td>
<td>real (binary)</td>
<td>ordinal</td>
</tr>
<tr>
<td>AdaBoost.OR</td>
<td>ordinal</td>
<td>ordinal</td>
</tr>
</tbody>
</table>
original problem

What is the rank of the movie $x$? ($r(x) = ?$)

reduced problems (Li and Lin, NIPS ’06)

Is the rank of movie $x$ greater than $k$? ($r(x) > k$?)

- traditional: combine probabilistic outputs (Frank and Hall, ECML ’01)
- ours: use counting of deterministic binary outputs

- simple and efficient
- good theoretical guarantee:
  1. absolutely good binary classifier $\Rightarrow$ absolutely good ranker
     (Li and Lin, NIPS ’06)
  2. relatively good binary classifier $\Rightarrow$ relatively good ranker
     (proved in this paper)
## Original Problem

What is the rank of the movie $x$? ($r(x) = ?$)

## Reduced Problems (Li and Lin, NIPS ’06)

Is the rank of movie $x$ greater than $k$? ($r(x) > k$?)

- **Traditional:** combine probabilistic outputs (Frank and Hall, ECML ’01)
- **Ours:** use counting of deterministic binary outputs

- **Simple and efficient**
- **Good theoretical guarantee:**
  1. Absolutely good binary classifier $\implies$ Absolutely good ranker (Li and Lin, NIPS ’06)
  2. Relatively good binary classifier $\implies$ Relatively good ranker (proved in this paper)
original problem

What is the rank of the movie \( x \)? \((r(x) = ?)\)

reduced problems (Li and Lin, NIPS ’06)

Is the rank of movie \( x \) greater than \( k \)? \((r(x) > k)\)

- traditional: combine probabilistic outputs (Frank and Hall, ECML ’01)
- ours: use counting of deterministic binary outputs

**simple and efficient**

- good theoretical guarantee:
  1. absolutely good binary classifier \(\rightarrow\) absolutely good ranker
     (Li and Lin, NIPS ’06)
  2. relatively good binary classifier \(\rightarrow\) relatively good ranker
     (proved in this paper)
**original problem**

What is the rank of the movie $x$? ($r(x) = ?$)

**reduced problems** (Li and Lin, NIPS '06)

Is the rank of movie $x$ greater than $k$? ($r(x) > k$?)

- traditional: combine probabilistic outputs (Frank and Hall, ECML '01)
- ours: use counting of deterministic binary outputs

- **simple** and **efficient**
- **good theoretical guarantee:**
  1. absolutely good binary classifier $\implies$ absolutely good ranker (Li and Lin, NIPS '06)
  2. relatively good binary classifier $\implies$ relatively good ranker (proved in this paper)
Goal
rankers $r_1(x) = 1$, $r_2(x) = 6$, $r_3(x) = 5$;
what does ensemble $R = \{r_1, r_2, r_3\}$ say on $x$?

Possible Solutions
- majority? $R(x) = 1$ or $5$ or $6$
- mean? $R(x) = 4$
- median? $R(x) = 5$
- ...?
Ordinal Ensemble: Prediction (1/2)

**Goal**

rankers \( r_1(x) = 1, \ r_2(x) = 6, \ r_3(x) = 5; \)
what does ensemble \( R = \{r_1, r_2, r_3\} \) say on \( x \)?

**Possible Solutions**

- majority? \( R(x) = 1 \) or 5 or 6
- mean? \( R(x) = 4 \)
- median? \( R(x) = 5 \)
- ...?
### Goal

rankers $r_1(x) = 1$, $r_2(x) = 6$, $r_3(x) = 5$;
what does ensemble $R = \{r_1, r_2, r_3\}$ say on $x$?

### Possible Solutions

- majority? $R(x) = 1$ or 5 or 6
- mean? $R(x) = 4$
- median? $R(x) = 5$
- ...?
Goal

rankers \( r_1(x) = 1 \), \( r_2(x) = 6 \), \( r_3(x) = 5 \);
what does ensemble \( R = \{r_1, r_2, r_3\} \) say on \( x \)?

Possible Solutions

- majority? \( R(x) = 1 \) or 5 or 6
- mean? \( R(x) = 4 \)
- median? \( R(x) = 5 \)
- ...?
Goal

rankers \( r_1(x) = 1, r_2(x) = 6, r_3(x) = 5; \)
what does ensemble \( R = \{r_1, r_2, r_3\} \) say on \( x \)?

Known

binary classifiers \( g_1(x) = Y, g_2(x) = N, g_3(x) = Y; \)
what does ensemble \( G = \{g_1, g_2, g_3\} \) say on \( x \)?
—majority vote \( G(x) = Y \)

<table>
<thead>
<tr>
<th>( r &gt; 1 )</th>
<th>( r &gt; 2 )</th>
<th>( r &gt; 3 )</th>
<th>( r &gt; 4 )</th>
<th>( r &gt; 5 )</th>
<th>( r &gt; 6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_1(x) = 1 )</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>( r_2(x) = 6 )</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>( r_3(x) = 5 )</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

majority

\( R(x) = 5 \) (provably, the median)
—can be applied to any ordinal ensemble
Ordinal Ensembles

Ordinal Ensemble: Prediction (2/2)

Goal

rankers $r_1(x) = 1$, $r_2(x) = 6$, $r_3(x) = 5$;
what does ensemble $R = \{r_1, r_2, r_3\}$ say on $x$?

Known

binary classifiers $g_1(x) = Y$, $g_2(x) = N$, $g_3(x) = Y$;
what does ensemble $G = \{g_1, g_2, g_3\}$ say on $x$?
—majority vote $G(x) = Y$

<table>
<thead>
<tr>
<th>$[r &gt; 1]$</th>
<th>$[r &gt; 2]$</th>
<th>$[r &gt; 3]$</th>
<th>$[r &gt; 4]$</th>
<th>$[r &gt; 5]$</th>
<th>$[r &gt; 6]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1(x) = 1$</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>$r_2(x) = 6$</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>$r_3(x) = 5$</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>majority</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

$R(x) = 5$ (provably, the median)
—can be applied to any ordinal ensemble
Goal

rankers \( r_1(x) = 1, r_2(x) = 6, r_3(x) = 5 \);
what does ensemble \( R = \{r_1, r_2, r_3\} \) say on \( x \)?

Known

binary classifiers \( g_1(x) = Y, g_2(x) = N, g_3(x) = Y \);
what does ensemble \( G = \{g_1, g_2, g_3\} \) say on \( x \)?
—**majority vote** \( G(x) = Y \)

<table>
<thead>
<tr>
<th>( r &gt; 1 )</th>
<th>( r &gt; 2 )</th>
<th>( r &gt; 3 )</th>
<th>( r &gt; 4 )</th>
<th>( r &gt; 5 )</th>
<th>( r &gt; 6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_1(x) = 1 )</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>( r_2(x) = 6 )</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>( r_3(x) = 5 )</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>majority</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

\( R(x) = 5 \) (**provably**, the median)
—can be applied to any ordinal ensemble
Ordinal Ensembles

Ordinal Ensemble: Prediction (2/2)

Goal

rankers $r_1(x) = 1, r_2(x) = 6, r_3(x) = 5$;
what does ensemble $R = \{r_1, r_2, r_3\}$ say on $x$?

Known

binary classifiers $g_1(x) = Y, g_2(x) = N, g_3(x) = Y$;
what does ensemble $G = \{g_1, g_2, g_3\}$ say on $x$?
—majority vote $G(x) = Y$

<table>
<thead>
<tr>
<th>$[r &gt; 1]$</th>
<th>$[r &gt; 2]$</th>
<th>$[r &gt; 3]$</th>
<th>$[r &gt; 4]$</th>
<th>$[r &gt; 5]$</th>
<th>$[r &gt; 6]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1(x) = 1$</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>$r_2(x) = 6$</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>$r_3(x) = 5$</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

majority $Y$

$R(x) = 5$ (provably, the median)
—can be applied to any ordinal ensemble
Ordinal Ensembles

Ordinal Ensemble: Prediction (2/2)

Goal

rankers $r_1(x) = 1$, $r_2(x) = 6$, $r_3(x) = 5$; what does ensemble $R = \{r_1, r_2, r_3\}$ say on $x$?

Known

binary classifiers $g_1(x) = Y$, $g_2(x) = N$, $g_3(x) = Y$; what does ensemble $G = \{g_1, g_2, g_3\}$ say on $x$? —majority vote $G(x) = Y$

<table>
<thead>
<tr>
<th>$[r &gt; 1]$</th>
<th>$[r &gt; 2]$</th>
<th>$[r &gt; 3]$</th>
<th>$[r &gt; 4]$</th>
<th>$[r &gt; 5]$</th>
<th>$[r &gt; 6]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1(x) = 1$</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>$r_2(x) = 6$</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>$r_3(x) = 5$</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>majority</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

$R(x) = 5$ (provably, the median) —can be applied to any ordinal ensemble
Goal

rankers \( r_1(x) = 1, r_2(x) = 6, r_3(x) = 5; \)
what does ensemble \( R = \{r_1, r_2, r_3\} \) say on \( x \)?

Known

binary classifiers \( g_1(x) = Y, g_2(x) = N, g_3(x) = Y; \)
what does ensemble \( G = \{g_1, g_2, g_3\} \) say on \( x \)?
—majority vote \( G(x) = Y \)

\[
\begin{array}{ccccccc}
\text{[r > 1]} & \text{[r > 2]} & \text{[r > 3]} & \text{[r > 4]} & \text{[r > 5]} & \text{[r > 6]} \\
\hline
\text{r}_1(x) = 1 & N & N & N & N & N & N \\
\text{r}_2(x) = 6 & Y & Y & Y & Y & Y & N \\
\text{r}_3(x) = 5 & Y & Y & Y & Y & N & N \\
\text{majority} & Y & Y & Y & Y & N & N \\
\end{array}
\]

\( R(x) = 5 \) (provably, the median)
—can be applied to any ordinal ensemble
Goal

locate ordinal rankers \( r_1(x), r_2(x), \ldots, r_T(x) \)
as well as their importance \( v_1, v_2, \ldots, v_T \)

Known: AdaBoost

locate binary classifiers \( g_1(x), g_2(x), \ldots, g_T(x) \)
as well as their importance \( v_1, v_2, \ldots, v_T \)
with weighted binary examples \((x_n, z_n, w_n^{(t)})\)

- binary classifier \(\Leftrightarrow\) ordinal ranker?
- weighted binary examples \(\Leftrightarrow\) cost-sensitive ordinal examples?

tools: reduction and reverse reduction
Ordinal Ensembles

Ordinal Ensemble: Training (1/4)

Goal

locate ordinal rankers $r_1(x), r_2(x), \cdots, r_T(x)$
as well as their importance $v_1, v_2, \cdots, v_T$

Known: AdaBoost

locate binary classifiers $g_1(x), g_2(x), \cdots, g_T(x)$
as well as their importance $v_1, v_2, \cdots, v_T$

with weighted binary examples $(x_n, z_n, w_n^{(t)})$

- binary classifier $\Leftrightarrow$ ordinal ranker?
- weighted binary examples $\Leftrightarrow$ cost-sensitive ordinal examples?

tools: reduction and reverse reduction
Goal

locate ordinal rankers \( r_1(x), r_2(x), \ldots, r_T(x) \)
as well as their importance \( v_1, v_2, \ldots, v_T \)

Known: AdaBoost

locate binary classifiers \( g_1(x), g_2(x), \ldots, g_T(x) \)
as well as their importance \( v_1, v_2, \ldots, v_T \)
with weighted binary examples \( (x_n, z_n, w_n^{(t)}) \)

- binary classifier \( \Leftrightarrow \) **ordinal ranker**?
- weighted binary examples \( \Leftrightarrow \) **cost-sensitive ordinal examples**?

**tools: reduction and reverse reduction**
Goal
locate ordinal rankers $r_1(x), r_2(x), \ldots, r_T(x)$
as well as their importance $v_1, v_2, \ldots, v_T$

Known: AdaBoost
locate binary classifiers $g_1(x), g_2(x), \ldots, g_T(x)$
as well as their importance $v_1, v_2, \ldots, v_T$
with weighted binary examples $(x_n, z_n, w^{(t)}_n)$

- binary classifier $\Leftrightarrow$ **ordinal ranker**?
- weighted binary examples $\Leftrightarrow$ **cost-sensitive ordinal examples**?

tools: reduction and reverse reduction
Ordinal Ensemble: Training (2/4)

1. Transform ordinal examples \((x_n, y_n, c_n)\) to weighted binary ones \((x_{nk}, z_{nk}, w_{nk})\)

2. Use your favorite algorithm on the weighted binary examples to get a binary classifier \(g(x, k)\)

3. For each new input \(x\), predict its rank using 
\[
    r_g(x) = 1 + \sum_k [g(x, k) = Y]
\]
1. Transform ordinal examples \((x_n, y_n, c_n)\) to weighted binary ones \((x_{nk}, z_{nk}, w_{nk})\).

2. Use your favorite algorithm on the weighted binary examples to get a binary classifier \(g\).

3. For each new input \(x\), predict its rank using

\[
r_g(x) = 1 + \sum_k [g(x, k) = Y]
\]
1. Transform ordinal examples \((x_n, y_n, c_n)\) to weighted binary ones \((x_{nk}, z_{nk}, w_{nk})\).

2. Use your favorite algorithm on the weighted binary examples to get a binary classifier \(g(x, k)\).

3. For each new input \(x\), predict its rank using \(r_g(x) = 1 + \sum_k [g(x, k) = Y]\).
Ordinal Ensemble: Training (3/4)

Reduction:
apply transforms on ordinal examples and binary classifiers

Reverse Reduction:
apply inverse transforms on binary examples and ordinal rankers

Ordinal example $(x_n, y_n, c_n) \Rightarrow$ weighted binary examples $(x_{nk}, z_{nk}, w_{nk}) \Rightarrow$ core binary classification algorithm $\Rightarrow$ related binary classifiers $g(x_k) \Rightarrow$ ordinal ranker $r_g(x)$
Ordinal Ensemble: Training (3/4)

**Reduction:**
apply transforms on ordinal examples and binary classifiers

**Reverse Reduction:**
apply inverse transforms on binary examples and ordinal rankers
Ordinal Ensembles

Ordinal Ensemble: Training (4/4)

AdaBoost.OR Derivation in a Nut Shell

1. plug AdaBoost into **reduction**
2. decompose AdaBoost as a series of binary base learners
3. cast ordinal base learner as binary one with **reverse reduction**
Ordinal Ensembles

Ordinal Ensemble: Training (4/4)

**AdaBoost.OR Derivation in a Nut Shell**

1. plug AdaBoost into reduction
2. decompose AdaBoost as a series of binary base learners
3. cast ordinal base learner as binary one with reverse reduction
Ordinal Ensemble: Training (4/4)

AdaBoost.OR Derivation in a Nut Shell

1. plug AdaBoost into reduction
2. decompose AdaBoost as a series of binary base learners
3. cast ordinal base learner as binary one with reverse reduction
**AdaBoost.OR: Further Simplifications**

### Reduction + Reverse Reduction

- **Examples**
  \[(x_n, y_n, c_n)\]
- **(Reduction)**
  \[\Rightarrow (x_{nk}, z_{nk}, w_{nk})\]
- **(AdaBoost)**
  \[\Rightarrow (x_{nk}, z_{nk}, w_{nk}^{(t)})\]
- **(Rev. Red.)**
  \[\Rightarrow (x_n, y_n, c_n^{(t)})\]

### AdaBoost.OR

- **Examples**
  \[(x_n, y_n, c_n)\]
- **(AdaBoost.OR)**
  \[\Rightarrow (x_n, y_n, c_n^{(t)})\]
  (Maintain \(c_n^{(t)}\) directly)

- **Ensemble**
  \[\{(v_t, r_t)\}\]
- **(Rev. Red.)**
  \[\Rightarrow \{(v_t, g_t)\}\]
- **(AdaBoost)**
  \[\Rightarrow G(x, k)\]
- **(Reduction)**
  \[\Rightarrow R_G(x)\]
- **(AdaBoost.OR)**
  \[\Rightarrow R(x)\]
  (Compute weighted median)
### AdaBoost.OR: Further Simplifications

#### Reduction + Reverse Reduction

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>examples</td>
<td>$(x_n, y_n, c_n)$</td>
<td>$(x_n, y_n, c_n)$</td>
</tr>
<tr>
<td>(reduction)</td>
<td>$\Rightarrow (x_{nk}, z_{nk}, w_{nk})$</td>
<td>$(x_{nk}, z_{nk}, w_{nk})$</td>
</tr>
<tr>
<td>(AdaBoost)</td>
<td>$\Rightarrow (x_{nk}, z_{nk}, w_{nk}^{(t)})$</td>
<td>$(x_{nk}, z_{nk}, w_{nk}^{(t)})$</td>
</tr>
<tr>
<td>(rev. red.)</td>
<td>$\Rightarrow (x_n, y_n, c_n^{(t)})$</td>
<td>$(x_n, y_n, c_n^{(t)})$</td>
</tr>
</tbody>
</table>

#### AdaBoost.OR

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>examples</td>
<td>$(x_n, y_n, c_n)$</td>
<td>$(x_n, y_n, c_n^{(t)})$</td>
</tr>
<tr>
<td>(AdaBoost.OR)</td>
<td>$\Rightarrow (x_n, y_n, c_n^{(t)})$</td>
<td>$(x_n, y_n, c_n^{(t)})$</td>
</tr>
<tr>
<td>(maintain $c_n^{(t)}$ directly)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

H.-T. Lin and L. Li (NTU/Caltech)  Combining Ordinal Preferences by Boosting  2009/09/12  15 / 20
### Reduction + Reverse Reduction

- **Examples**
  \[ (x_n, y_n, c_n) \]
  
- **(Reduction)**
  \[ \rightarrow (x_{nk}, z_{nk}, w_{nk}) \]

- **(AdaBoost)**
  \[ \rightarrow (x_{nk}, z_{nk}, w_{nk}^{(t)}) \]

- **(Reverse Reduction)**
  \[ \rightarrow (x_n, y_n, c_n^{(t)}) \]

### AdaBoost.OR

- **Examples**
  \[ (x_n, y_n, c_n) \]
  
- **(AdaBoost.OR)**
  \[ \rightarrow (x_n, y_n, c_n^{(t)}) \]

  *(maintain \( c_n^{(t)} \) directly)*

### Reduction + Reverse Reduction

- **Ensemble**
  \[ \{(v_t, r_t)\} \]
  
- **(Reverse Reduction)**
  \[ \rightarrow \{(v_t, g_t)\} \]

- **(AdaBoost)**
  \[ \rightarrow G(x, k) \]

- **(Reduction)**
  \[ \rightarrow R_G(x) \]

### AdaBoost.OR

- **Ensemble**
  \[ \{(v_t, r_t)\} \]
  
- **(AdaBoost.OR)**
  \[ \rightarrow R(x) \]

  *(compute weighted median)*
### Reduction + Reverse Reduction

<table>
<thead>
<tr>
<th>Reduction + Reverse Reduction</th>
<th>AdaBoost.OR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>examples</strong> ((x_n, y_n, c_n))</td>
<td><strong>examples</strong> ((x_n, y_n, c_n))</td>
</tr>
<tr>
<td>(reduction) (\Rightarrow) ((x_{nk}, z_{nk}, w_{nk}))</td>
<td>(AdaBoost.OR) (\Rightarrow) ((x_n, y_n, c_n^{(t)}))</td>
</tr>
<tr>
<td>(AdaBoost) (\Rightarrow) ((x_{nk}, z_{nk}, w_{nk}^{(t)}))</td>
<td>(maintain (c_n^{(t)}) directly)</td>
</tr>
<tr>
<td>(rev. red.) (\Rightarrow) ((x_n, y_n, c_n^{(t)}))</td>
<td></td>
</tr>
</tbody>
</table>

### Reduction + Reverse Reduction

<table>
<thead>
<tr>
<th>Reduction + Reverse Reduction</th>
<th>AdaBoost.OR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ensemble</strong> ({(v_t, r_t)})</td>
<td><strong>ensemble</strong> ({(v_t, r_t)})</td>
</tr>
<tr>
<td>(rev. red.) (\Rightarrow) ({(v_t, g_t)})</td>
<td>(AdaBoost.OR) (\Rightarrow) (R(x))</td>
</tr>
<tr>
<td>(AdaBoost) (\Rightarrow) (G(x, k))</td>
<td>(compute weighted median)</td>
</tr>
<tr>
<td>(reduction) (\Rightarrow) (R_G(x))</td>
<td></td>
</tr>
</tbody>
</table>
AdaBoost.OR versus AdaBoost

**AdaBoost**

for $t = 1, 2, \ldots, T$,

1. find a simple $g_t$ that matches best with the current “view” of $\{(x_n, y_n)\}$
2. give a larger weight $v_t$ to $g_t$ if the match is stronger
3. update “view” by emphasizing the weights of those $(x_n, y_n)$ that $g_t$ doesn’t predict well

prediction: majority vote of $\{(v_t, g_t(x))\}$

**AdaBoost.OR**

= reduction + any cost + AdaBoost + derivations
### AdaBoost.OR

for $t = 1, 2, \cdots, T$,

1. find a simple $r_t$ that matches best with the current “view” of $\{(x_n, y_n)\}$

2. give a larger weight $v_t$ to $r_t$ if the match is stronger

3. update “view” by emphasizing the costs $c_n$ of those $(x_n, y_n)$ that $r_t$ doesn’t predict well

prediction:

- weighted median of $\{(v_t, r_t(x))\}$

---

### AdaBoost

for $t = 1, 2, \cdots, T$,

1. find a simple $g_t$ that matches best with the current “view” of $\{(x_n, y_n)\}$

2. give a larger weight $v_t$ to $g_t$ if the match is stronger

3. update “view” by emphasizing the weights of those $(x_n, y_n)$ that $g_t$ doesn’t predict well

prediction:

- majority vote of $\{(v_t, g_t(x))\}$

---

AdaBoost.OR = reduction + any cost + AdaBoost + derivations
AdaBoost.OR versus AdaBoost

**AdaBoost.OR**

for $t = 1, 2, \ldots, T$,

1. find a simple $r_t$ that matches best with the current “view” of $\{(x_n, y_n)\}$
2. give a larger weight $v_t$ to $r_t$ if the match is stronger
3. update “view” by emphasizing the costs $c_n$ of those $(x_n, y_n)$ that $r_t$ doesn’t predict well

prediction: weighted median of $\{(v_t, r_t(x))\}$

**AdaBoost**

for $t = 1, 2, \ldots, T$,

1. find a simple $g_t$ that matches best with the current “view” of $\{(x_n, y_n)\}$
2. give a larger weight $v_t$ to $g_t$ if the match is stronger
3. update “view” by emphasizing the weights of those $(x_n, y_n)$ that $g_t$ doesn’t predict well

prediction: majority vote of $\{(v_t, g_t(x))\}$

AdaBoost.OR = reduction + any cost + AdaBoost + derivations
For AdaBoost.OR, if rankers $r_t$ always achieve normalized training cost $\leq \frac{1}{2} - \gamma$, the training cost of ensemble is bounded by:

$$\leq \text{constant} \cdot \exp(-2\gamma^2 T)$$

For AdaBoost, if classifiers $g_t$ always achieve weighted training error $\leq \frac{1}{2} - \gamma$, the training error of ensemble is bounded by:

$$\leq \text{constant} \cdot \exp(-2\gamma^2 T)$$

Many other useful properties are inherited:
algorithmic structure; boosting property; generalization bounds

Any future improvements in AdaBoost imply parallel improvements in AdaBoost.OR.
Ordinal Ensembles

Boosting Property of AdaBoost.OR

**Ordinal Ranking**

For AdaBoost.OR, if rankers $r_t$ always achieve normalized training cost $\leq \frac{1}{2} - \gamma$,

training cost of ensemble $\leq \text{constant} \cdot \exp(-2\gamma^2 T)$

**Bin. Class. (Freund and Schapire, 1997)**

For AdaBoost, if classifiers $g_t$ always achieve weighted training error $\leq \frac{1}{2} - \gamma$,

training error of ensemble $\leq \text{constant} \cdot \exp(-2\gamma^2 T)$

- many other useful properties inherited: algorithmic structure; boosting property; generalization bounds

any future improvements in AdaBoost $\Rightarrow$ parallel improvements in AdaBoost.OR
Experimental Performance

ORStump v.s. AdaBoost.OR + ORStump

- ORStump: a simple algorithm for ordinal ranking
- AdaBoost.OR: a good ensemble learning algorithm for ordinal ranking

- boosts ORStump in both training and testing
- efficient and sometimes outperforms benchmark
ORStump v.s. AdaBoost.OR + ORStump

- ORStump: a simple algorithm for ordinal ranking
- AdaBoost.OR: a good ensemble learning algorithm for ordinal ranking

- boosts ORStump in both training and testing
- efficient and sometimes outperforms benchmark
ORStump: a simple algorithm for ordinal ranking

AdaBoost.OR: a good ensemble learning algorithm for ordinal ranking

- boosts ORStump in both training and testing
- efficient and sometimes outperforms benchmark
reduction + reverse reduction:
- proved: relatively good binary classifier $\Rightarrow$ relatively good ranker
- derived AdaBoost.OR
  - training: update costs instead of weights
  - prediction: weighted median (wider application)

proved boosting and generalization properties of AdaBoost.OR
obtained good experimental results

more general reduction results:
(H.-T. Lin & L. Li, Reduction from Ordinal Ranking to Binary Classification, 2009)
reduction + reverse reduction:

- proved: relatively good binary classifier $\implies$ relatively good ranker
- derived AdaBoost.OR
  - training: update costs instead of weights
  - prediction: weighted median (wider application)

proved boosting and generalization properties of AdaBoost.OR

obtained good experimental results

more general reduction results:

(H.-T. Lin & L. Li, Reduction from Ordinal Ranking to Binary Classification, 2009)
Conclusion

- reduction + reverse reduction:
  - proved: relatively good binary classifier $\rightarrow$ relatively good ranker
  - derived AdaBoost.OR
    - training: update costs instead of weights
    - prediction: weighted median (wider application)

- proved boosting and generalization properties of AdaBoost.OR

- obtained good experimental results

more general reduction results:
(H.-T. Lin & L. Li, Reduction from Ordinal Ranking to Binary Classification, 2009)
Conclusion

- reduction + reverse reduction:
  - proved: relatively good binary classifier $\Rightarrow$ relatively good ranker
  - derived AdaBoost.OR
    - training: update costs instead of weights
    - prediction: weighted median (wider application)

- proved **boosting and generalization properties** of AdaBoost.OR

- obtained good experimental results

more general reduction results:
(H.-T. Lin & L. Li, Reduction from Ordinal Ranking to Binary Classification, 2009)
reduction + reverse reduction:
  - proved: relatively good binary classifier $\Rightarrow$ relatively good ranker
  - derived AdaBoost.OR
    - training: update costs instead of weights
    - prediction: **weighted median** (wider application)

proved **boosting and generalization properties** of AdaBoost.OR

obtained **good experimental results**

more general reduction results:
(H.-T. Lin & L. Li, Reduction from Ordinal Ranking to Binary Classification, 2009)
reduction + reverse reduction:
- proved: relatively good binary classifier $\rightarrow$ relatively good ranker
- derived AdaBoost.OR
  - training: update costs instead of weights
  - prediction: weighted median (wider application)

proved **boosting and generalization properties** of AdaBoost.OR

obtained **good experimental results**

more general reduction results:
(H.-T. Lin & L. Li, Reduction from Ordinal Ranking to Binary Classification, 2009)
Thank you. Questions?