Combining Ordinal Preferences by Boosting

Hsuan-Tien Lin and Ling Li

National Taiwan University/California Institute of Technology

Preference Learning Workshop, September 12, 2009
Hot or Not?

http://www.hotornot.com

Rate People | Meet People | Best Of | Meet Jim and James

HOT or NOT

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 \( \mathcal{Y} = \{1, 2, \cdots, K\} \)
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 ⭐⭐⭐⭐⭐⭐⭐⭐
Given

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

- 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 \textbf{unseen} inputs \( x \)

no numerical information: how to say “close”?
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 \((x, y, \mathbf{c})\):
  - \( \mathbf{c}[k] = \) cost when predicting \((x, y)\) as rank \(k\)
  - e.g. for \((\text{Sweet Home Alabama}, ★★★★★★)\), a customer-oriented cost may be \( \mathbf{c} = (1, 0, 2, 10, 15) \)

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

  *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

how to construct a good ordinal ensemble?
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
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? $R(x) = 5$
- ...?
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></th>
<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>$r_1(x)$</td>
<td>1</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>$r_2(x)$</td>
<td>6</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>$r_3(x)$</td>
<td>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>
<td>N</td>
</tr>
</tbody>
</table>

$R(x) = 5$ (provably, the median)
—can be applied to any ordinal ensemble
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
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 \(g\) for each new input \(x\), predict its rank using

\[
rg(x) = 1 + \sum_k [g(x, k) = Y]
\]
Ordinal Ensemble: Training (3/4)

 ordinal example \((x_n, y_n, c_n)\) ⇒ weighted binary examples \((x_{nk}, z_{nk}, w_{nk})\) ⇒ core binary classification algorithm ⇒ related binary classifiers \(g(x, k)\) ⇒ ordinal ranker \(r_g(x)\)

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

 weighted binary examples \((x_{nk}, z_{nk}, w_{nk})\) ⇒ ordinal example \((x_n, y_n, c_n)\) ⇒ core ordinal ranking algorithm ⇒ ordinal ranker \(r(x)\) ⇒ related binary classifiers \(g_r(x, k)\)

**reverse reduction:**
apply inverse transforms on binary examples and ordinal rankers
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

<table>
<thead>
<tr>
<th>Examples</th>
<th>Reduction</th>
<th>AdaBoost</th>
<th>Reverse Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x_n, y_n, c_n)$</td>
<td>$(x_{nk}, z_{nk}, w_{nk})$</td>
<td>$(x_n, y_n, c_n(t))$</td>
<td>$(x_n, y_n, c_n)$</td>
</tr>
<tr>
<td>(reduction)</td>
<td>(AdaBoost)</td>
<td>(maintain $c_n(t)$ directly)</td>
<td></td>
</tr>
<tr>
<td>(AdaBoost)</td>
<td>(rev. red.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(rev. red.)</td>
<td>(reduction)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### AdaBoost.OR

<table>
<thead>
<tr>
<th>Ensemble</th>
<th>Reduction</th>
<th>AdaBoost</th>
<th>Reverse Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>${(v_t, r_t)}$</td>
<td>${(v_t, g_t)}$</td>
<td>$R(x)$</td>
<td>${(v_t, r_t)}$</td>
</tr>
<tr>
<td>(rev. red.)</td>
<td>(AdaBoost)</td>
<td>(compute weighted median)</td>
<td></td>
</tr>
<tr>
<td>(AdaBoost)</td>
<td>(reduction)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(reduction)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
AdaBoost.\text{OR} versus AdaBoost

**AdaBoost.\text{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.\text{OR} = reduction + any cost + AdaBoost + derivations
**Ordinal Ranking**

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

training cost of ensemble

$\leq$ 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

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

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

any future improvements in AdaBoost

$\implies$ 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

---

H.-T. Lin and L. Li (NTU/Caltech)  
Combining Ordinal Preferences by Boosting  
2009/09/12 18 / 20
Conclusion

- 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)
Thank you. Questions?