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

rank: representing human preferences
by a finite **ordered** set of labels $\mathcal{Y} = \{1, 2, \cdots, K\}$

How Much Did You Like These Movies?

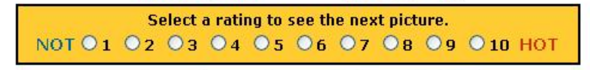
`http://www.netflix.com`

goal: use "movies you've rated" to automatically predict your **preferences (ranks)** on future movies

Properties of Ordinal Ranking

- 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 

Ordinal Ranking Setup

Given

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

- `hotornot`: $\mathcal{X} =$ encoding(human pictures), $\mathcal{Y} = \{1, \cdots, 10\}$
- `netflix`: $\mathcal{X} =$ 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

  > 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 (`Sweet Home Alabama` , ★★☆☆☆),
  > a customer-oriented cost may be $\mathbf{c} = (1, 0, 2, 10, 15)$

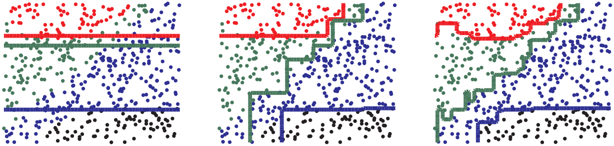
- or use general cost vectors:

$\mathbf{c}[k] = [\![y \neq k]\!]$	$\mathbf{c}[k] = \lvert y - k \rvert$
classification	absolute
$(1, 0, 1, 1, 1)$	$(1, 0, 1, 2, 3)$

closely predict: small cost during testing

Combining Ordinal Rankers

- 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**

algorithm	base ranker	final ranker
RankBoost (Freund et al., JMLR '03)	real (pairwise)	real (pairwise)
ORBoost (Lin and Li, ALT '06)	real (binary)	ordinal
AdaBoost.OR	**ordinal**	**ordinal**

From Ordinal Ranking to Binary Classification (and Back)

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 $\Longrightarrow$ absolutely good ranker (Li and Lin, NIPS '06)
  2. relatively good binary classifier $\Longrightarrow$ relatively good ranker (proved in this paper)

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? $R(x) = 5$
- ...?

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** $G(x) = Y$

	$[\![r > 1]\!]$	$[\![r > 2]\!]$	$[\![r > 3]\!]$	$[\![r > 4]\!]$	$[\![r > 5]\!]$	$[\![r > 6]\!]$
$r_1(x) = 1$	N	N	N	N	N	N
$r_2(x) = 6$	Y	Y	Y	Y	Y	N
$r_3(x) = 5$	Y	Y	Y	Y	N	N
majority	Y	Y	Y	Y	N	N

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

Ordinal Ensemble: Training (1/4)

Goal

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

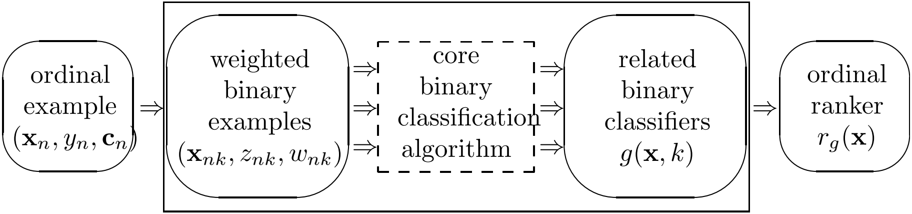
Known: AdaBoost

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**?

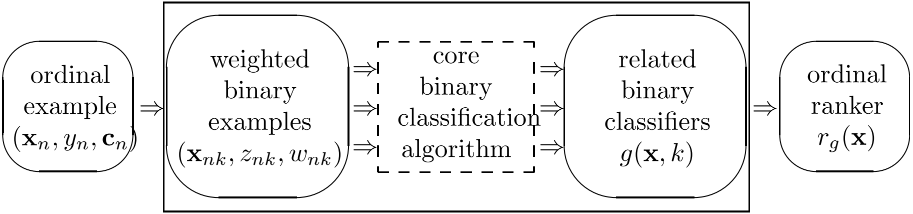
tools: reduction and reverse reduction

Ordinal Ensemble: Training (2/4)

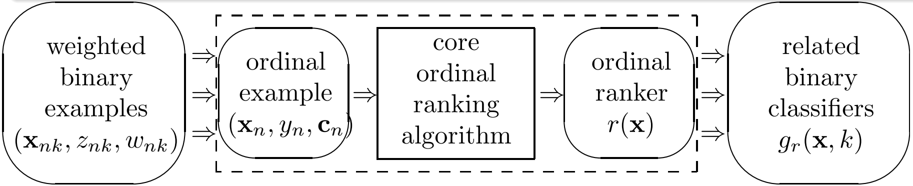


1. transform ordinal examples $(x_n, y_n, \mathbf{c}_n)$ to weighted binary ones $\left(x_{nk}, z_{nk}, w_{nk}\right)$
2. use your favorite algorithm on the weighted binary examples to get a binary classifier g
3. for each new input x, predict its rank using $r_g(x) = 1 + \sum_k [\![g(x, k) = \mathrm{Y}]\!]$

Ordinal Ensemble: Training (3/4)

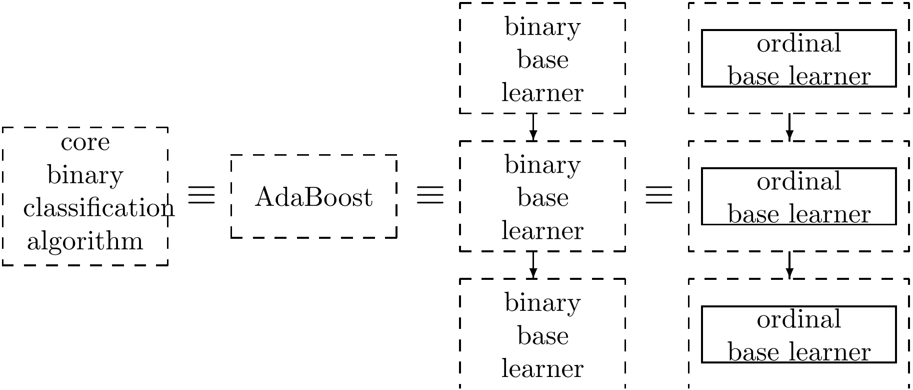


reduction:
apply transforms on ordinal examples and binary classifiers



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

Ordinal 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		$(x_n, y_n, \mathbf{c}_n)$
(reduction)	$\Longrightarrow$	(x_{nk}, z_{nk}, w_{nk})
(AdaBoost)	$\Longrightarrow$	$(x_{nk}, z_{nk}, w_{nk}^{(t)})$
(rev. red.)	$\Longrightarrow$	$(x_n, y_n, \mathbf{c}_n^{(t)})$

AdaBoost.OR

examples		$(x_n, y_n, \mathbf{c}_n)$
(AdaBoost.OR)	$\Longrightarrow$	$(x_n, y_n, \mathbf{c}_n^{(t)})$

(maintain $\mathbf{c}_n^{(t)}$ directly)

Reduction + Reverse Reduction

ensemble		$\{(v_t, r_t)\}$
(rev. red.)	$\Longrightarrow$	$\{(v_t, g_t)\}$
(AdaBoost)	$\Longrightarrow$	$G(x, k)$
(reduction)	$\Longrightarrow$	$R_G(x)$

AdaBoost.OR

ensemble		$\{(v_t, r_t)\}$
(AdaBoost.OR)	$\Longrightarrow$	$R(x)$

(compute weighted median)

AdaBoost.OR versus AdaBoost

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 $\mathbf{c}_n$ of those (x_n, y_n) that r_t doesn't predict well

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

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 $\left\{\left(v_t, g_t(x)\right)\right\}$

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

Boosting Property of AdaBoost.OR

Ordinal Ranking

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

$$\begin{aligned} & \text{training cost of ensemble} \\ \leq\ & \text{constant} \cdot \exp\left(-2\gamma^2 T\right) \end{aligned}$$

Bin. Class. (Freund and Schapire, 1997)

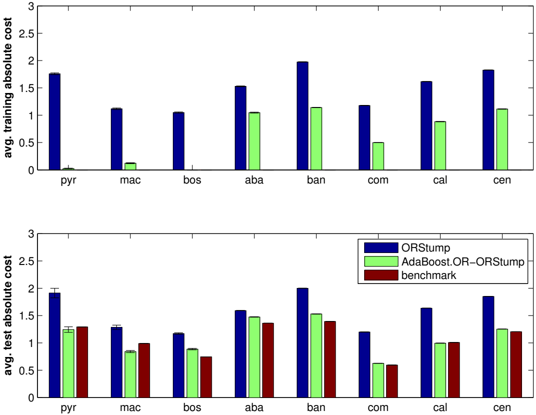
For AdaBoost, if classifiers g_t always achieve weighted training error $\leq \frac{1}{2} - \gamma$,

$$\begin{aligned} & \text{training error of ensemble} \\ \leq\ & \text{constant} \cdot \exp\left(-2\gamma^2 T\right) \end{aligned}$$

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

any future improvements in AdaBoost
$\Longrightarrow$ **parallel improvements in AdaBoost.OR**

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

Conclusion

- reduction + reverse reduction:
  - proved: relatively good binary classifier $\Longrightarrow$ 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?