An Ensemble Solution for Learning to Rank

Ming-Feng Tsai
National University of Singapore

Shang-Tse Chen, Yao-Nan Chen, Chun-Sung Ferng,
Chia-Hsuan Wang, Tszy-Yu Wen and Hsuan-Tien Lin
National Taiwan University

June 25, 2010

Three Specialties of Learning to Rank Challenge

- ordinally-ranked data: $(\mathbf{x}_{qn}, y_{qn})$ with $y_{qn} \in \{0, 1, 2, 3, 4\}$
  —y_{qn} carries ordinal but **no numerical meanings**

  include **ordinal ranking** approaches

- query-based criteria favoring top-ranked instances within
  —$(\mathbf{x}_{qn}, y_{qn})$ **not equally important**

  consider **weighting** and **cost-sensitive** schemes

- huge amount of data in set 1; limited amount of data in set 2
  —challenging **computationally** and **generalization-wise**

  build **ensemble solution**
  —divide-&-conquer set 1
  —mix-&-conquer set 2

Ensemble Solution for Set 1

Ensemble

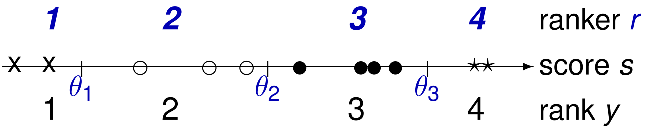
- pointwise: ORBoost
- pointwise: ORPolySVM
- pointwise: ORKernelSVM
- pairwise: RankPolyLR
- pairwise: RankLinearSVM
- listwise: BoltzRank

model diversity; method diversity

Pointwise: Ordinal Ranking Methods

Gist of Algorithms

- score each instance by some $s(\mathbf{x})$
- quantize score to $r(\mathbf{x}) = \underset{k}{\text{argmin}} \{ s(\mathbf{x}) < \theta_k \}$ to "match" rank



ORBoost

- score $s(\mathbf{x})$: **linear ensemble** of weak rankers $\sum \alpha_t h_t(\mathbf{x})$
- boosting-based; large-margin

ORSVM

- score $s(\mathbf{x})$: **linear function** in some Hilbert space $\mathcal{H}$
- SVM-based; large-margin

ORBoost (Lin and Li, 2006)

automatic feature selection with boosted performance

Basic Choices

- ORBoost-All: all margins in loss
- decision stump weak ranker, rather than soft perceptrons
- T by some validation

Special Tuning

- **across-query point weighting**: balance influence of each query
- **within-query point weighting**: focus on top-ranked instances ($\propto y_{qn} + 1$)

- time: 950 min. on $\approx$ 70% of set 1; memory: 5G
- public ERR: 0.4487

ORPolySVM (Chu and Keerthi, 2005; Li and Lin, 2007)

simple additive model that can be efficiently trained

Basic Choices

- 1st, 2nd, 3rd, 4th order terms, without cross-terms
- LIBLINEAR solver
- C by some validation

Special Tuning

- **query-level thresholding**: different "scales" for different queries

- time: 13 min. on $\approx$ 70% of set 1 (after transforming data); memory: $5G$
- public ERR: 0.4456 (worse than ORBoost)

ORKernelSVM

sophisticated model that yields the best single ranker

Basic Choices

- perceptron kernel
- LIBSVM solver
- C by some validation

Special Tuning

- **cost-sensitive** with cost generated from ERR (minor improvement)

- time: 4 * (1000 min. on $\approx$ 20% of set 1); memory: 40G
- public ERR: 0.4527 (our best single entry)

Pairwise: Relative Ranking Methods

Gist of Algorithms

- score each instance by some $s(\mathbf{x})$ such that s matches the "order" of the ranks

$$y > y' \Leftrightarrow \mathbf{x} \succ \mathbf{x}' \Leftrightarrow s(\mathbf{x}) > s(\mathbf{x}')$$

RankLR

- **logistic loss** on the linear score difference

RankSVM

- **hinge loss** on the linear score difference

RankPolyLR

fast solver (Sculley, 2009) **with pretty good performance**

Basic Choices

- within-query pairs
- 2nd order terms, with cross-terms
- fixed $\lambda = 0.01$, $T = 10^7$

Special Tuning

- **across-query point weighting** in sampling
- **within-query point weighting** in learning rate: emphasize (r_i, r_j) pair with $2^{r_i} - 2^{r_j}$

- time: 200 min. on $\approx 80\%$ of set 1; memory: $1G$
- public ERR: 0.4503 (our 2nd best single entry)

RankLinearSVM

a robust traditional choice with rooms for tuning

Basic Choices

- within-query pairs
- LIBLINEAR solver, fixed $C = 0.01$ (bigger C takes much longer)

Special Tuning

- **within-query point weighting**: emphasize (r_i, r_j) pair with $\max(r_i, r_j)^{|r_i - r_j|}$
- **within-query feature normalization**: capture instance relations within query

- time: 13 min. on $\approx 80\%$ of set 1 when using small C, after loading data; memory: $13G$
- public ERR: 0.4421 (behind previous five)

Listwise: Permutation Ranking Methods

Gist of Algorithms

- try to "match" the list order within each query with respect to the criteria of interest

BoltzRank

- gradient descent on ERR using Neural Networks

BoltzRank (Volkovs and Zemel, 2009)

a sophisticated model that may match the ERR criteria better

Basic Choices

- hand-written solver
- hidden layers and other parameters selected by validation

Special Tuning

- **feature selection** by AdaRank to speed up
- **regularization** by KL-divergence to avoid overfitting

- time: 800 min. on $\approx$ 80% of set 1; memory: $3G$
- public ERR: 0.4394 (worst)

Three Readouts on the Numbers

ORBoost	ORPolySVM	ORKernelSVM
0.4487	0.4456	0.4527

RankPolyLR	RankLinearSVM	BoltzRank
0.4503	0.4421	0.4394

- within pointwise models: ORKernelSVM best

  worth using if computationally feasible

- across models: pointwise promising

  fewer transformed examples than pairwise, but similar performance; much faster than listwise

- linear versus nonlinear: improvements when going nonlinear

  kernel design, feature transforms, or ensemble

Ensemble Solution for Set 1

Ensemble using 20% Holdout: RankPolyLR (0.4565)

- pointwise: ORBoost (0.4487)
- pointwise: ORPolySVM (0.4456)
- pointwise: ORKernelSVM (0.4527)
- pairwise: RankPolyLR (0.4503)
- pairwise: RankLinearSVM (0.4421)
- listwise: BoltzRank (0.4394)

ensemble better than individual

Ensemble Solution for Set 2

Ensemble: ORKernelSVM (0.4490)

- pointwise: ORBoost
- pointwise: ORPolySVM
- pointwise: ORKernelSVM
- pairwise: RankPolyLR
- pairwise: RankLinearSVM
- listwise: BoltzRank

model diversity; method diversity;
set diversity (set 1, set 2, domain adaptation)

Conclusion

- pointwise methods worked!
  —**can it be useful for similar applications?**
- weighting and cost-sensitive worked!
  —**how to design loss that better match ERR?**
- query-oriented tuning worked!
  —**can we improve if knowing more about queries?**
- ensemble learning by stacking worked!
  —**is there a better way of combining rankers w.r.t. ERR?**
- lots of things don't work, especially computationally!
  —AdaRank, BoltzRank with more nodes, ...

Thank you. Questions?