Ranking via Robust Binary Classification

Hyokun Yun[1], Parameswaran Raman[2], S.V.N. Vishwanathan[1,2]
Amazon[1], University of California Santa Cruz[2]

What is RoBiRank?

A Robust and Scalable Ranking algorithm:

- Optimizes for quality on top of the ranking list
- Directly bounds NDCG (popular evaluation metric for ranking)
- Can be efficiently parallelized and scales to very large datasets
- Demonstrates competitive results on both small-medium and large datasets

Robust Classification

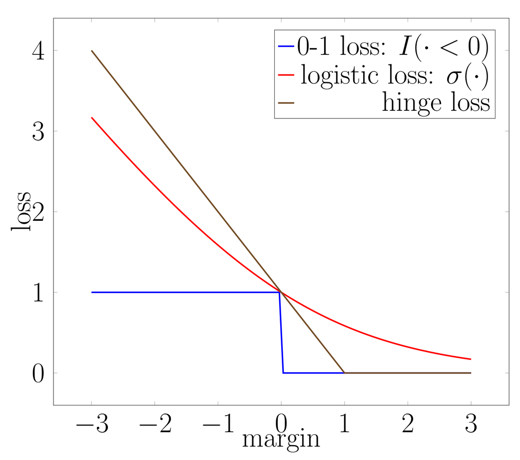
Setup: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ with $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$.

- Binary Classification aims to minimize the number of mistakes in the dataset:

$$L(\omega) = \sum_{i=1}^{n} I(y_i \cdot \langle x_i, \omega \rangle < 0).$$

$$L(\omega) = \sum_{i=1}^{n} \sigma(y_i \cdot \langle x_i, \omega \rangle). \quad \text{(Non-robust)}$$

When $\sigma(t) = \log_2(1 + 2^{-t})$, we get logistic regression.
When $\sigma(t) = \max(1 - t, 0)$, we get SVM.



However, **Convex objective functions are sensitive to outliers.**

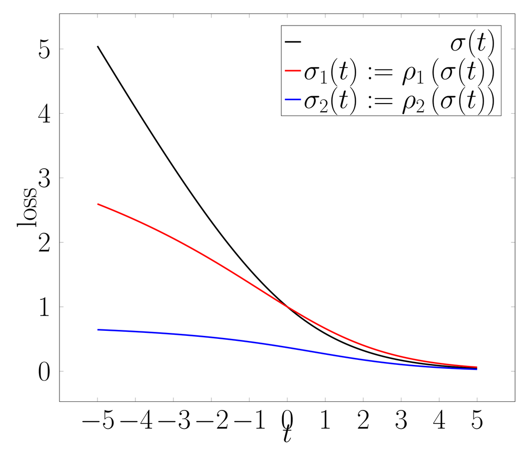
- Using following transformations,

$$\rho_1(t) = \log_2(t+1), \quad \rho_2(t) := 1 - \frac{1}{\log_2(t+2)},$$

we can **bend** the loss functions to get:

$$L_1(\omega) = \sum_{i=1}^{n} \rho_1(\sigma(y_i \cdot \langle x_i, \omega \rangle)), \quad \text{(Robust Type I)}$$

$$L_2(\omega) = \sum_{i=1}^{n} \rho_2(\sigma(y_i \cdot \langle x_i, \omega \rangle)). \quad \text{(Robust Type II)}$$

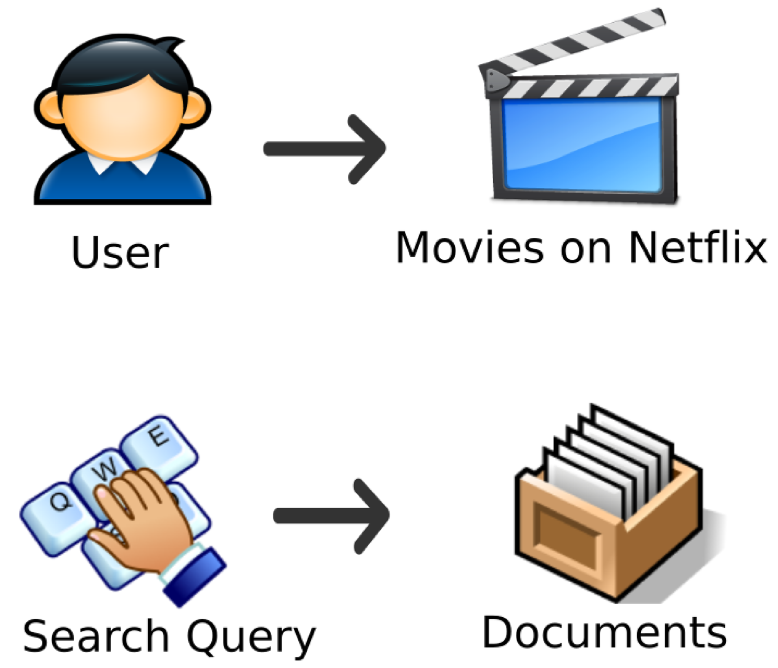


- As $t \to \infty$, **Type I loss function** goes to ∞ at a much slower rate
- Even if $t \to \infty$, **Type II loss function** does not go to ∞.
- **Type II loss function** has stronger statistical guarantees.
- **Type I loss function** is easier to optimize, since its gradient does not vanish.

Learning to Rank

Notations:

- $\mathcal{X}$ = set of users, $\mathcal{Y}$ = set of items, r_{xy} = rating user x gave to item y
- $\phi(x, y) \in \mathbb{R}^d$: extracted feature between x and y, $\omega \in \mathbb{R}^d$: model parameter
- $f_\omega(x, y) := \langle \phi(x, y), \omega \rangle$: the score model assigns to item y for user x



Rank of an item y for user x can be defined as:

$$\text{rank}_\omega(x, y) = \sum_{y' \in \mathcal{Y}_x, y' \neq y} I(f_\omega(x, y) - f_\omega(x, y') < 0).$$

Using this, **objective function for ranking** can be expressed as:

$$L(\omega) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}_x} r_{xy} \sum_{y' \in \mathcal{Y}_x, y' \neq y} \sigma(f_\omega(x, y) - f_\omega(x, y')).$$

Discounted Cumulative Gain (DCG):

$$\text{DCG}(\omega) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}_x} \frac{r_{xy}}{\log_2(\text{rank}_\omega(x, y) + 2)},$$

Gain of an item degrades logarithmically based on its rank

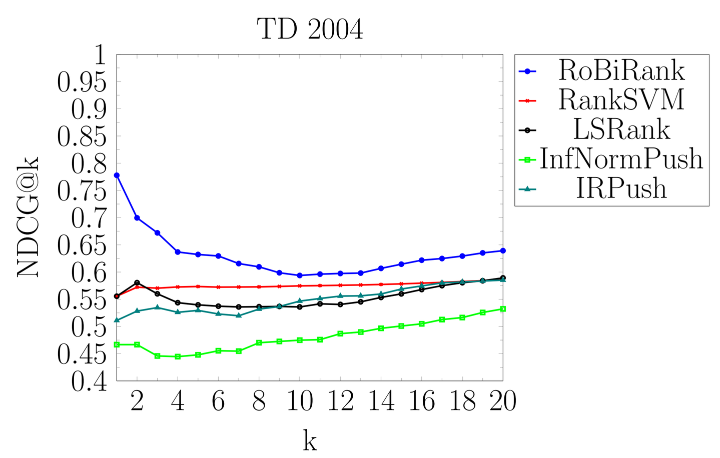
It turns out, **Maximizing DCG $\Leftrightarrow$ Minimizing Robust version of $L(\omega)$**

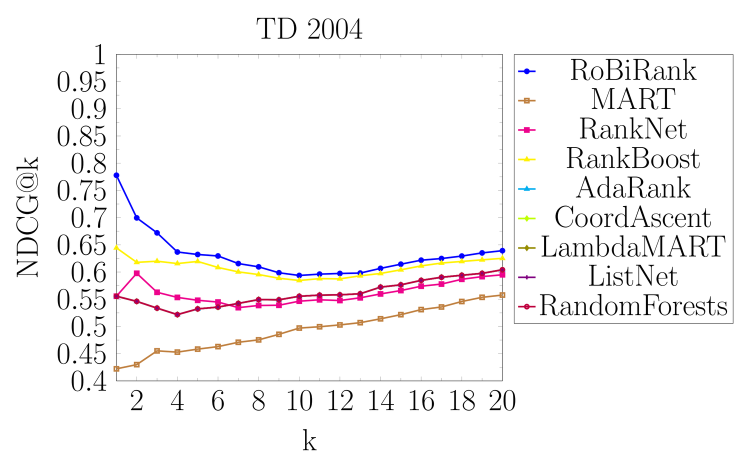
$$L_2(\omega) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}_x} r_{xy} \cdot \rho_2 \left(\sum_{y' \in \mathcal{Y}_x, y' \neq y} \sigma(f_\omega(x, y) - f_\omega(x, y')) \right). \quad \text{(Robust Type II)}$$

To avoid the vanishing gradient problem, our proposed method - RoBiRank, optimizes:

$$L_1(\omega) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}_x} r_{xy} \cdot \rho_1 \left(\sum_{y' \in \mathcal{Y}_x, y' \neq y} \sigma(f_\omega(x, y) - f_\omega(x, y')) \right). \quad \text{(Robust Type I)}$$

- **Results on small-medium datasets:**





RoBiRank shows better performance at the top as expected

Latent Collaborative Retrieval

- **When the size of the data, especially $\mathcal{Y}$ is large,**
  - Generating features $\phi(x, y)$ for all x and y is challenging
  - Computing $\sum_{y' \in \mathcal{Y}_x, y' \neq y} \sigma(f_\omega(x, y) - f_\omega(x, y'))$ is expensive
  - The data usually consists of implicit feedback: $r_{xy} = 0$ for most (x, y).
- To avoid the feature engineering burden, let
  - user parameter: $U_1, U_2, \ldots, U_n \in \mathbb{R}^d$
  - item parameter: $V_1, V_2, \ldots, V_m \in \mathbb{R}^d$
  - score: $f_\omega(x, y) := \langle U_x, V_y \rangle$,

  as in matrix factorization. The objective function becomes

$$\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}_x} r_{xy} \cdot \rho_1 \left(\sum_{y' \in \mathcal{Y}_x, y' \neq y} \sigma(\langle U_x, V_y \rangle - \langle U_x, V_{y'} \rangle) \right).$$

- To avoid calculating the summation over $\mathcal{Y}$, using the following property of $\rho_1(\cdot)$,

$$\rho_1(t) = \log_2(t+1) \leq -\log_2 \xi + \frac{\xi \cdot (t+1) - 1}{\log 2}, \quad \text{(for any } \xi > 0)$$

we **linearize** the objective function:

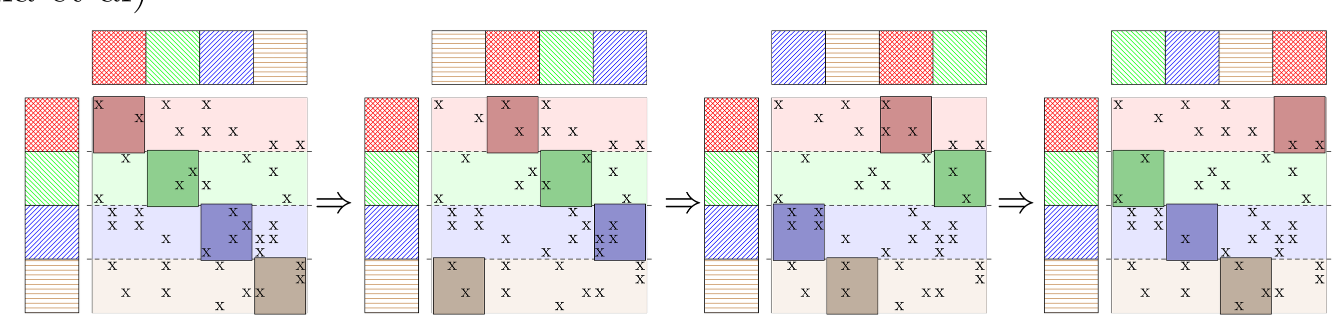
$$\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}_x} r_{xy} \cdot \left[-\log_2 \xi_{xy} + \frac{\xi_{xy} \cdot \left(\sum_{y' \neq y} \sigma(\langle U_x, V_y \rangle - \langle U_x, V_{y'} \rangle) + 1 \right) - 1}{\log 2} \right],$$

by introducing ξ_{xy} for each x, y with $r_{xy} \neq 0$.

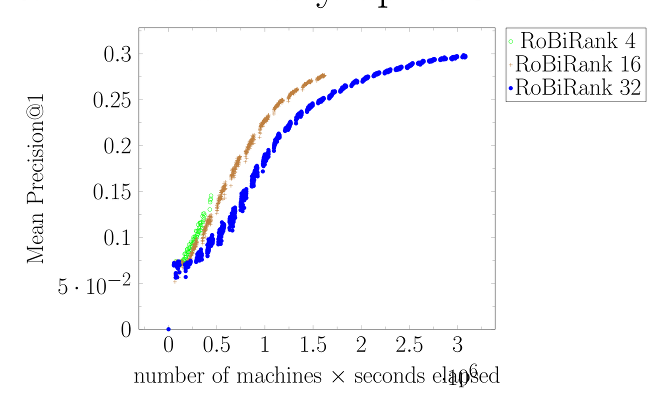
- If we uniformly sample (x, y, y') from $\{(x, y, y') : r_{xy} \neq 0\}$, we obtain an **unbiased estimator**, which allows us to take **stochastic gradient** with convergence guarantees.

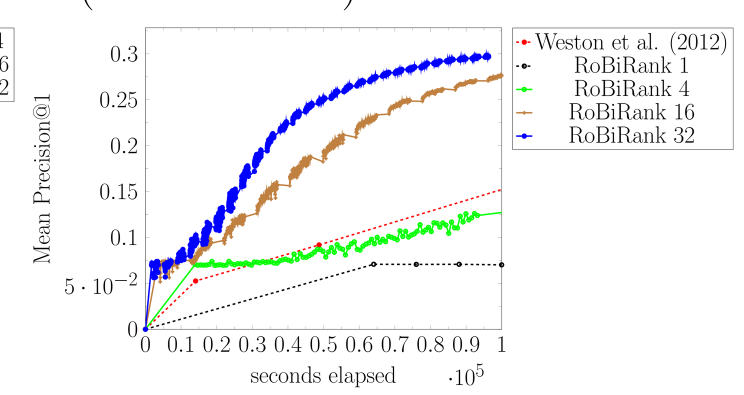
Parallelization

- User parameters and item parameters are partitioned into multiple machines
- User parameters always stay, item parameters are exchanged after each epoch
- Within each epoch, SGD updates are taken within accessible region (Stratified SGD of Gemula et al)



- RoBiRank scales nicely up to 32 machines (16 cores each)





Paper

Ranking via Robust Binary Classification, Hyokun Yun, Parameswaran Raman, S.V.N.Vishwanathan, (NIPS 2014)