Ranking via Robust Binary Classification

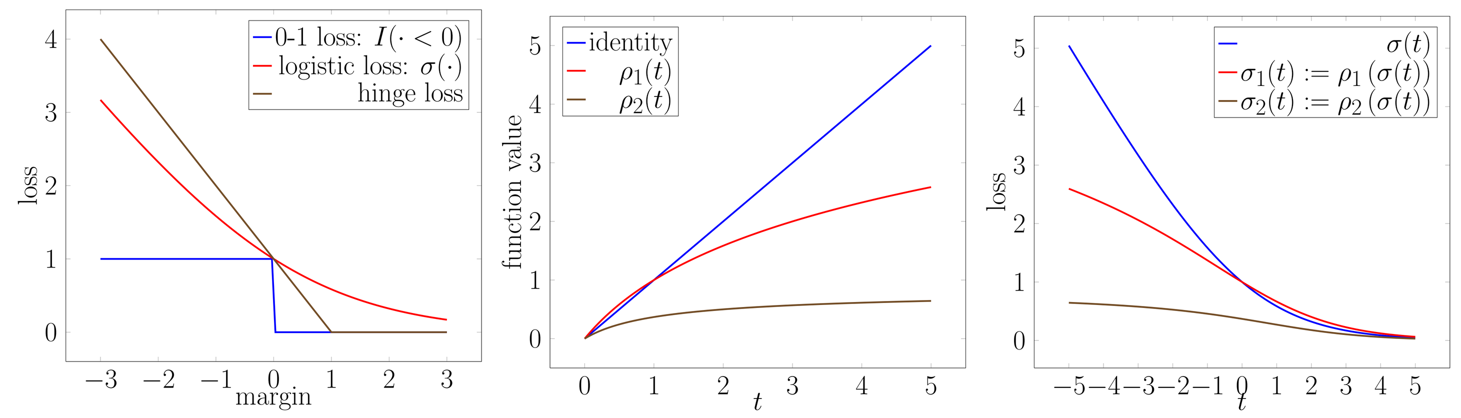
Hyokun Yun[1], Parameswaran Raman[2], S.V.N. Vishwanathan[1,2]

Amazon[1], University of California Santa Cruz[2]

Abstract

- We show that learning to rank can be viewed as a generalization of robust classification.
- Motivated by this observation, we propose RoBiRank, which is a non-convex bound of (N)DCG.
- Although non-convex, it consists of Type-I loss functions [1] and thus amenably optimized.
- When applied to latent collaborative retrieval (matrix factorization with ranking loss), the algorithm can be efficiently parallelized:
- Our algorithm shows competitive performance on latent collaborative retrieval of Million Song Dataset (MSD), which requires to model $386,133 \times 49,824,519$ pairwise interactions.

Robust Classification



- Suppose $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ with $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$.
- Ideally, we would like to optimize the number of mistakes:

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

but since it is discrete, we bound each indicator by a continuous loss function:

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

  - When $\sigma(t) := \log_2\left(1 + 2^{-t}\right)$, we get logistic regression.
  - When $\sigma(t) := \max 1 - t, 0$, we get SVM.
- 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 *warp* loss functions to get:

$$\overline{L}_1(\omega) := \sum_{i=1}^{n} \rho_1\left(\sigma(y_i \cdot \langle x_i, \omega \rangle)\right), \quad \text{(Robust Type I)}$$

$$\overline{L}_2(\omega) := \sum_{i=1}^{n} \rho_2\left(\sigma(y_i \cdot \langle x_i, \omega \rangle)\right). \quad \text{(Robust Type II)}$$

  - As $t \to \infty$, Type I loss function $\rho_1(\sigma(-t))$ goes to ∞ in much slower rate than $\sigma(-t)$ does.
  - Even if $t \to \infty$, Type II loss function $\rho_1(\sigma(-t))$ does not go to ∞.
  - Type II loss function has stronger statistical guarantees.
  - Type I loss function is easier to optimize, since the gradient does not vanish.

Learning to Rank

- Notations
  - $\mathcal{X} := \{x_1, x_2, \ldots, x_n\}$: set of users
  - $\mathcal{Y} := \{y_1, y_2, \ldots, y_m\}$: set of items
  - s_{xy}: score user x assigns to item y
  - $\phi(x, y) \in \mathbb{R}^d$: extracted feature between x and y.
  - $\omega \in \mathbb{R}^d$: model parameter
  - $f_\omega(x, y) := \langle \phi(x, y), \omega \rangle$: score model assigns on item y for user x
  - $\text{rank}_\omega(x, y)$: rank of item y for user x. Note that

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

- Simple objective function for ranking would be [2]:

$$\begin{aligned}\min_\omega L(\omega) &:= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}_x} s_{xy} \cdot \text{rank}_\omega(x, y), \\ &= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}_x} s_{xy} \sum_{y' \in \mathcal{Y}_x, y' \neq y} I\left(f_\omega(x, y) - f_\omega(x, y') < 0\right),\end{aligned}$$

and again, we can bound each indicator by a continuous loss:

$$\min_\omega \overline{L}(\omega) := \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}_x} s_{xy} \sum_{y' \in \mathcal{Y}_x, y' \neq y} \sigma\left(f_\omega(x, y) - f_\omega(x, y') < 0\right).$$

- Discounted Cumulative Gain (DCG):

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

Maximization of DCG is equivalent to:

$$\min_\omega \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}_x} s_{xy} \cdot \left\{1 - \frac{1}{\log_2\left(\text{rank}_\omega(x, y) + 2\right)}\right\}$$

$$\Leftrightarrow \min_\omega \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}_x} s_{xy} \cdot \left\{1 - \frac{1}{\log_2\left(\sum_{y' \in \mathcal{Y}_x, y' \neq y} I\left(f_\omega(x, y) - f_\omega(x, y') < 0\right) + 2\right)}\right\}$$

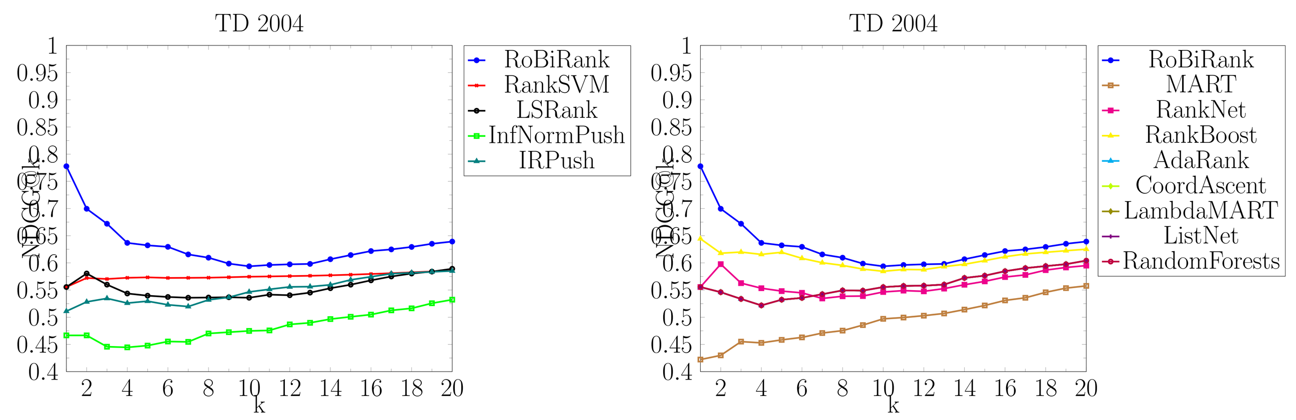
$$\Leftrightarrow \min_\omega \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}_x} s_{xy} \cdot \rho_2\left(\sum_{y' \in \mathcal{Y}_x, y' \neq y} I\left(f_\omega(x, y) - f_\omega(x, y') < 0\right)\right).$$

Its continous bound would be:

$$\overline{L}_2(\omega) := \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}_x} s_{xy} \cdot \rho_2\left(\sum_{y' \in \mathcal{Y}_x, y' \neq y} \sigma\left(f_\omega(x, y) - f_\omega(x, y')\right)\right). \quad \text{(Robust Type II)}$$

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

$$\overline{L}_1(\omega) := \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}_x} s_{xy} \cdot \rho_1\left(\sum_{y' \in \mathcal{Y}_x, y' \neq y} \sigma\left(f_\omega(x, y) - f_\omega(x, y')\right)\right). \quad \text{(Robust Type I)}$$



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 $\sum_{y' \in \mathcal{Y}_x, y' \neq y} \sigma\left(f_\omega(x, y) - f_\omega(x, y')\right)$ is expensive
  - Usually consists of implicit feedback: $s_{xy} = 0$ 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 factorzation [3]. The objective function becomes

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

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

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

we *linearize* the objective function:

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

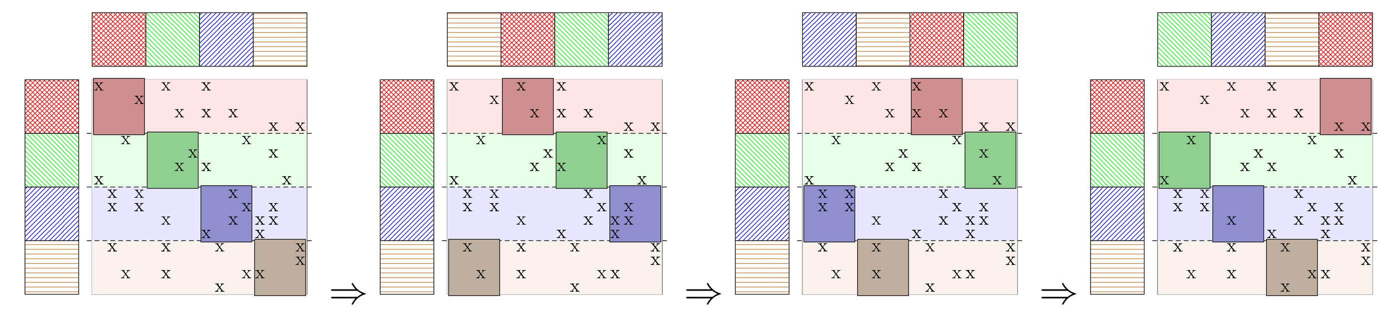
by introducing ξ_{xy} for each x, y with $s_{xy} \neq 0$.
- If we uniformly sample (x, y, y') from $\{(x, y, y') : s_{xy} \neq 0\}$,

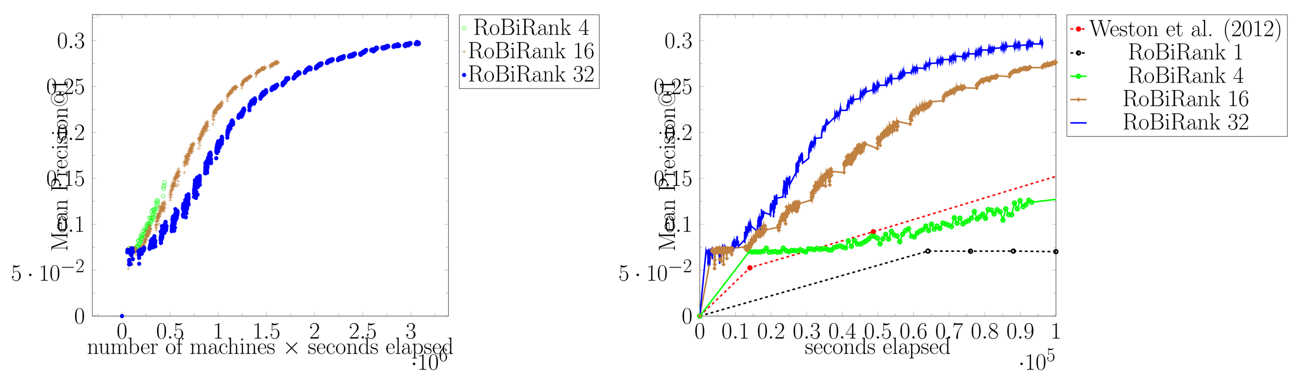
$$s_{xy} \cdot \left[\frac{-\log_2 \xi_{xy} + \frac{\xi_{xy} - 1}{\log 2}}{|\mathcal{Y}| - 1} + \xi_{xy} \cdot \sigma\left(\langle U_x, V_y \rangle - \langle U_x, V_{y'} \rangle\right)\right],$$

is an unbiased estimator, which allows us to take guaranteed stochastic gradient.

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 [4])





References

[1] N. Ding. Statistical Machine Learning in T-Exponential Family of Distributions. (Ph.D Thesis)

[2] D. Buffoni, P. Gallinari, N. Usunier, and C. Calauzenes. Learning scoring functions with order-preserving losses and standardized supervision completion. (ICML 2011)

[3] J. Weston, C. Wang, R. Weiss, and A. Berenzweig. Latent collaborative retrieval. (ICML 2012)

[4] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis. Large-scale matrix factorization with distributed stochas- tic gradient descent. (KDD 2011)