WMRB: Learning to Rank in a Scalable Batch Training Approach

Kuan Liu
Information Sciences Institute
University of Southern California
kuanl@usc.edu

Prem Natarajan
Information Sciences Institute
University of Southern California
pnataraj@isi.edu

ABSTRACT

We propose a new learning to rank algorithm, named Weighted Margin-Rank Batch loss (WMRB), to extend the popular Weighted Approximate-Rank Pairwise loss (WARP). WMRB uses a new rank estimator and an efficient batch training algorithm. The approach allows more accurate item rank approximation and explicit utilization of parallel computation to accelerate training. In three item recommendation tasks, WMRB consistently outperforms WARP and other baselines. Moreover, WMRB shows clear time efficiency advantages as data scale increases.

KEYWORDS

Learning to Rank, Batch Training, Scalability

1 INTRODUCTION

Rank based learning algorithms have been widely applied to recommendation problems. One prominent example is Weighted Approximate-Rank Pairwise loss [2]. It achieves a high precision on the *top* of a predicted ranked list instead of an *averaged* high precision over the entire list. However, it is not scalable to large item set in practice due to its intrinsic online learning fashion. In this work, we address the limitation by proposing a novel algorithm and empirically demonstrate its advantages in both accuracy and time efficiency.

2 BACKGROUND

Notation. Let x denote a user, y an item, and $\mathbf{Y}$ the entire item set. $\mathbf{y}_x$ denotes items interacted by user x. $\bar{\mathbf{y}}_x \equiv \mathbf{Y} \setminus \mathbf{y}_x$ is the irrelevant item set. We omit subscript x when there is no ambiguity. $f_y(x)$ denotes the model score. The rank of item y is defined as

$$r_y = rank_y(f, x, \mathbf{y}) = \sum_{\bar{y} \in \bar{\mathbf{y}}} \mathbf{I}[f_y(x) \leq f_{\bar{y}}(x)], \quad (1)$$

where $\mathbf{I}$ is the indicator function. Finally, $|t|_+ \equiv max(t, 0), t \in \mathbb{R}$.

Weighted Approximate-Rank Pairwise loss (WARP) develops an *online* approximation of item ranks. Its critical component is **an iterative sampling approximation procedure**: For each user-item pair (x, y) , sample $y' \in \mathbf{Y}$ uniformly with replacement until $1 + f_{y'}(x) < f_y(x)$ is violated. It estimates item ranks by

$$r_y \approx rank_y^{warp}(f, x, \mathbf{y}) = \lfloor \frac{|\mathbf{Y}| - 1}{N} \rfloor \quad (2)$$

where N is the sampling times to find the violating example. It then incurs a rank-sensitive loss as in Order Weighted Average [3], i.e.,

Figure 1: Standard deviations (relative values) of two types of rank estimators at different item ranks. Simulation is done with item set size N=100,000. 'online' uses estimator (2) and 'sampled-batch q' uses (4) where $q = |\mathbf{Z}|/|\mathbf{Y}|$.
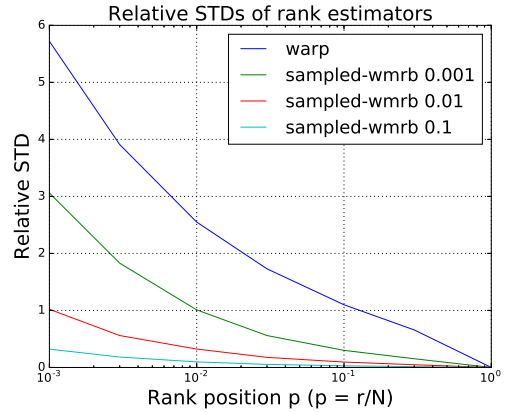
$$\Phi^{owa}(r_y) = \sum_{j=1}^{r_y} \alpha_j \quad \alpha_1 \geq \alpha_2 \geq .. \geq 0, \quad (3)$$

where the non-increasing α series control the sensitivity to ranks.

3 WEIGHTED MARGIN-RANK BATCH LOSS

Limitations of WARP. We first point out several limitations of WARP. 1) **Rank estimator** in (2) is not only biased[1] but also with **large variance**. As simulated in Fig.1, online estimation (blue) has a large relative variance, especially when items have high ranks (small p). 2) **Low updating frequency** results in prolonged training time in practice. This is because it can take a large number of sampling iterations to find a violating item, especially after the beginning stage of training. 3) **Intrinsic sequential manner** of WARP prevents full utilization of available parallel computation (e.g. GPUs), making it hard to train large or flexible models.

3.1 Proposed Approach

We address the limitations by combining a rank-sensitive loss and batch training. We first define (sampled) **margin rank** as

$$rank_y^{wmrb}(f, x, \mathbf{y}) = \frac{|\mathbf{Y}|}{|\mathbf{Z}|} \sum_{y' \in \mathbf{Z}} |1 - f_y(x) + f_{y'}(x)|_+ \mathbf{I}(y' \in \bar{\mathbf{y}}), \quad (4)$$

where $\mathbf{Z}$ is a subset of $\mathbf{Y}$ randomly sampled (without replacement).

---

[1]Suppose an item has a rank r in a population N. Let $p = r/N$. Expectation of the estimator in (2) is approximately $p + \sum_{k=2}^{N} \frac{1}{k} p(1-p)^{k-1} > p$. It overestimates the rank seriously when r is small.

Table 1: Recommendation accuracy comparisons (in %). Best results are in bold (e.g., 12.6). WMRB outperforms pairwise based methods as well as batch based method CE.

Datasets		XING			Yelp			ML-20m		
Metrics		P@5	R@30	NDCG@30	P@5	R@30	NDCG@30	P@5	R@30	NDCG@30
-	POP	0.5	2.7	1.3	0.3	0.9	0.5	6.2	10.0	8.5
Pairwise	WARP	2.6	8.3	5.6	1.3	4.4	2.5	9.8	14.2	13.4
	A-WARP	2.6	11.6	6.7	1.3	4.3	2.5	10.1	13.3	13.5
Batch	CE	2.5	12.3	6.5	1.4	4.5	2.6	9.6	14.3	13.2
	WMRB	**3.0**	**12.6**	**7.2**	**1.5**	**5.1**	**2.9**	**10.2**	**14.6**	**13.9**

Table 2: Dataset statistics. U: users; I: items; S: interactions.

Data	$|U|$	$|I|$	$|S_{\text{train}}|$	$|S_{\text{test}}|$
XING	1,500,000	327,002	2,338,766	484,237
Yelp	1,029,433	144,073	1,917,218	213,460
ML-20m	138,493	27,278	7,899,901	2,039,972

While *margin rank* defined in (4) is not the rank in (1), it characterizes overall score violation of irrelevant items. The margin loss is often used to approximate the indicator function. Moreover, (4) can be readily computed in a batch manner—Model scores between a user and a batch of items $f_{y'}(x) \forall y' \in \mathbf{Y}$ are first computed; The margin losses of violating items are then summed up.

We then design a **rank-sensitive loss function**. Note the *margin rank* $r_y = rank_y^{wmrb}(f, x, \mathbf{y})$ is a non-negative real number rather than an integer as in (3). We define a differentiable loss function to incur "rank weighted" loss as follows:

$$L^{wmrb}(x,y) = \Phi^{wmrb}(r_y) = \log(r_y + 1). \quad (5)$$

By noting $\Phi''(r) = -\frac{1}{(1+r)^2} < 0$, the loss is more sensitive with small r, thus mimicking the property as in (3).

Compared to WARP. , WMRB replaces the sequential sampling procedure with batch computations. It results in a different rank approximation and loss function. Per user-item pair, WMRB involves more computation and is compensated with easy utilization of parallel computing. WMRB updates model parameters much more frequently than WARP – which only updates the parameters of one user-item after many sampling.

WMRB has an unbiased estimator of margin rank. Its different sampling scheme results in smaller variances. Simulation in Fig.1 shows sampled-wmrb has much smaller variance than *warp* as long as $|\mathbf{Z}|/|\mathbf{Y}|$ is not too small.

4 RESULTS

We validate WMRB on three datasets: XING[2], Yelp[3], and MovieLens-20m[4]. The tasks are to recommend to users job posts, Yelp business, and movies, respectively. We assess the quality of recommendation by comparing models' recommendation to ground truth interactions split from the datasets. We report recommendation accuracies under metrics Precsion@5, Recall@30, and NDCG@30 as well as training time. The datasets statistics are listed in Table 2.

We compare WMRB to different methods. POP recommends items purely based on popularity. WARP and A-WARP are implemented in LightFM [1]. A-WARP differs from vanilla WARP by incorporating available attributes. CE uses Cross-Entropy loss

Table 3: Dataset complexities and training time comparisons. (T_{epoch}: average epoch time; T_{conv}: total training time.) With increasing data scales, WMRB shows time efficiency advantages over pairwise based implementation.

Datasets	# of Param.	# of Attr.	LightFM		WMRB	
			T_{epoch}	T_{conv}	T_{epoch}	T_{conv}
ML-20m	4.6M	11	**7m**	**1.2h**	22m	3.3 h
Yelp	9.3M	19	10m	5.0 h	**9m**	**3.9 h**
XING	12.1M	33	94m	31.2h	**24m**	**20.7h**

function and is a batch training based algorithm implemented by ourselves. CE and WMRB incorporate attributes as in A-WARP.

Accuracy. Table 1 reports accuracy comparisons of different models. We highlight two observations. First, WMRB consistently outperforms WARP and A-WARP. For instance, the relative improvements are 8.6%, 18.6%, and 9.8% on Recall@30. Second, with both being batch based methods, WMRB wins over CE clearly, indicating the effectiveness of the rank-sensitive loss.

Time efficiency. Table 3 reports dataset complexity and training time of different models. To measure complexity, we report the total number of model parameters and the average number of attributes per user-item pair. While we make an effort to speed up each method,[5] we are most interested in how running time changes with different data scales given fixed models and configurations.

From Table 3, WMRB is slower than LightFM on ML-20m. It catches up quickly on Yelp, where data scale increases. On XING, which has the largest model size and complexity, WMRB is much faster. The trend clearly shows scalability advantages of batch training based WMRB.

5 CONCLUSION

In this work, we propose a simple yet effective approach to scale up learning to rank algorithm. It implements the idea of ordered weighted average loss in a batch training algorithm. Our preliminary empirical study shows its effectiveness.

REFERENCES
[1] Maciej Kula. 2015. Metadata Embeddings for User and Item Cold-start Recommendations. *arXiv preprint arXiv:1507.08439* (2015). http://arxiv.org/abs/1507.08439
[2] Jason Weston, Samy Bengio, and Nicolas Usunier. 2010. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning* 81, 1 (2010), 21–35.
[3] Ronald R Yager. 1988. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on systems, Man, and Cybernetics* 18, 1 (1988), 183–190.

---

[2]http://2016.recsyschallenge.com/
[3]https://www.yelp.com/dataset_challenge. Downloaded in Feb 17.
[4]https://grouplens.org/datasets/movielens/20m/

---

[5]WMRB are implemented based on Tensorflow and run on a single GPU (GeForce GTX TITAN X). LightFM runs asynchronous SGD on Cython with 5 cores.(Intel Xeon CPU 3.30GHz)