Greedy Optimized Multileaving for Personalization

Kojiro Iizuka
Gunosy Inc.
Tokyo, Japan
kojiro.iizuka@gunosy.com

Takeshi Yoneda
Gunosy Inc.
Tokyo, Japan
takeshi.yoneda@gunosy.com

Yoshifumi Seki
Gunosy Inc.
Tokyo, Japan
yoshifumi.seki@gunosy.com

ABSTRACT

Personalization plays an important role in many services. To evaluate personalized rankings, online evaluation, such as A/B testing, is widely used today. Recently, *multileaving* has been found to be an efficient method for evaluating rankings in information retrieval fields. This paper describes the first attempt to optimize the multileaving method for personalization settings. We clarify the challenges of applying this method to personalized rankings. Then, to solve these challenges, we propose greedy optimized multileaving (GOM) with a new *credit feedback function*. The empirical results showed that GOM was stable for increasing ranking lengths and the number of rankers. We implemented GOM on our actual news recommender systems, and compared its online performance. The results showed that GOM evaluated the personalized rankings precisely, with significantly smaller sample sizes ($< 1/10$) than A/B testing.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Retrieval effectiveness**.

KEYWORDS

Interleaving and multileaving; evaluation of personalized ranking

1 INTRODUCTION

Personalization plays an important role in various web services, such as e-commerce, streaming, and news [4, 5, 7]. A/B testing has been widely adopted to evaluate the effectiveness of personalization algorithms. For a precise evaluation, however, A/B testing requires a large number of users; the more the number of algorithms for A/B testing increases, the more users are needed.

Recently, multileaving has been found to be an efficient method for evaluating rankings in information retrieval fields [3]. Interleaving evaluates two rankings with a small sample size, while

multileaving extends the evaluation to three or more rankings. However, studies that examine the application of multileaving for personalized rankings are limited [1].

This paper describes our first attempt to optimize multileaving for personalized rankings. The paper's contributions are as follows:

- **New Problems**: Clarifies the challenges of applying the multileaving method to personalized rankings;
- **Algorithm**: Proposes the greedy optimized multileaving (GOM) method with the *personalization credit function* to solve these problems; and
- **Stability and Sensitivity**: Confirms the stability and sensitivity of GOM throughout offline and online experiments.

We achieved high sensitivity with low computation cost on the personalized rankings, which differed for each user and each time. In addition, we ensured stability over the number of rankers and ranking length. These results show that the proposed method can evaluate many algorithms or hyperparameters efficiently in practical environments, such as Social Network Service (SNS), news, and Consumer to Consumer (CtoC) market services.

2 RELATED WORK AND CHALLENGES

2.1 Multileaving Method

Multileaving is a method that evaluates multiple rankings using user click feedback [10]. The typical steps of a multileaving evaluation are as follows.

The first step is to input multiple ranking sets $I = \{I_1, I_2, \ldots, I_n\}$. These input rankings are generated from *rankers*. The second step is to generate output ranking sets $O = \{O_1, O_2, \ldots, O_m\}$. The third step is to show each output ranking O_k to users with probability p_k. The fourth step is to aggregate the user click *credit* and evaluate which input ranking is better. Basically, the ranking is evaluated as better when the sum of credit is higher.

Each multileaving method consists of three components: (1) a way to construct O from I, (2) probability p_k for each output ranking O_k, and (3) a credit function δ. In this paper, we represent the i-th item of ranking I_j and O_j as $I_{j,i}$ and $O_{j,i}$; the user click credit as $\delta(O_{k,i}, I_j)$; and the length of O_k as l. In the next section, we describe two established multileaving methods, team draft multileaving (TDM) [10] and optimized multileaving (OM) [8], in detail.

2.2 Team Draft Multileaving

TDM is a method that selects an input ranking randomly and adds an item to the output ranking that was not previously chosen [9]. This process is then repeated until the multileaved ranking is of sufficient length. In the credit aggregation of TDM, rankers are credited for each click on an item drawn from the corresponding ranker; however, because rankers are credited only for clicks on

items drawn from the corresponding ranker, TDM tends to be unstable over increasing ranker sizes [2].

2.3 Optimized Multileaving

OM generates the output rankings, and solves an optimization problem formulated by *sensitivity* and *bias* [8]. The output ranking's sensitivity discriminates the differences in effectiveness between input rankings; meanwhile, the bias of the output rankings measures the differences between the expected credits of the input rankings for random clicks [6, 10]. In OM, the sensitivity that constrains the bias is maximized. By solving this optimization problem, OM achieves more sensitivity than TDM. In this paper, similar to [6, 10], we use insensitivity instead of sensitivity. Insensitivity (σ) is defined by the credit sum and the mean of credit μ_k. $\sigma_k^2 = \sum_{j=1}^n \left(\sum_{i=1}^l f(i)\delta(O_{k,i}, I_j) - \mu_k \right)^2$, where $f(i)$ is the probability with which a user clicks on the i-th item, and the mean of credit $\mu_k = \frac{1}{n} \sum_{j=1}^n \sum_{i=1}^l f(i)\delta(O_{k,i}, I_j)$. The formulation of OM is as follows [6]:

$$\min_{p_k} \quad \alpha \sum_{r=1}^l \lambda_r + \sum_{k=1}^m p_k \sigma_k^2$$

subject to

$$(\forall r, j, j')| \sum_{k=1}^m p_k \sum_{i=1}^r \delta(O_{k,i}, I_j) - \sum_{k=1}^m p_k \sum_{i=1}^r \delta(O_{k,i}, I_{j'})| \le \lambda_r$$

$$\sum_{k=1}^m p_k = 1, \quad 0 \le p_k \le 1 \quad (k = 1, \dots, m).$$

$$(1)$$

We follow the original work [10], and use $f(i) = 1/i$. The bias term is added to the objective function, and is adjusted with hyperparameter α. α is the only hyperparameter in this formulation. λ_r is the maximum difference of the expected credits in any input rankings pairs, and represents the bias. If λ_r is close to zero, the bias becomes small.

In previous studies, the click credit function was defined by the input ranking's position [6, 8, 10]. For example,

$$\delta(O_{k,i}, I_j) = \frac{1}{rank(O_{k,i}, I_j)}, \quad (2)$$

where $rank(O_{k,i}, I_j)$ represents the position of item $O_{k,i}$ in ranking I_j. If there is no item $O_{k,i}$ in ranking I_j, then the credit value is $1/(|I_j| + 1)$. We call this credit function *inverse credit*. By definition, the deeper the click position, the weaker the credit. Another well-known definition is $\delta(O_{k,i}, I_j) = -rank(O_{k,i}, I_j)$ if $O_{k,i} \in I_j$; otherwise, $-(|I_j| + 1)$. In this case, the credit difference between the top and bottom items is too big, compared to (top item, top item) or (bottom item, bottom item). This difference induces noise for the credit evaluation. The empirical results, described in Section 4, indicated that when credit above is used, OM tended to become unstable as the ranking length increased.

2.4 Challenges

The challenges of applying the multileaving method to personalization are as follows:

C1: Achieving high sensitivity with low computation cost for the rankings, which differed for each user and each time; and

C2: Ensuring stability over the number of rankers and the ranking length.

OM maximizes sensitivity by assuming that the same rankings are inputted multiple times. In personalization, however, input rankings differ for each user. Furthermore, input rankings vary according to time stamps in environments where new items rapidly increase, such as SNS, news, and CtoC market services. Therefore, the ranking differs for each user and each time. In these cases, OM cannot be pre-calculated, and requires low computation cost to show multileaved output rankings to users in real-time.

Unstability over the number of rankers is known to occur in TDM [2, 10]. This is a serious problem for personalization, because there are usually many algorithm candidates and hyperparameters. In addition, the empirical results showed that OM was unstable over the ranking length when ordinary click credit is used. This is critical because many services have long ranking length to show as many personalized items as possible. Thus, for personalization, we should ensure stability over the number of rankers and the ranking length.

3 GREEDY OPTIMIZED MULTILEAVING

3.1 Formulation

To solve the first challenge, we propose a new formulation using the characteristic of personalized ranking. Personalized ranking is shown to the user a few times, because it differs for each user and each time in the situation when the number of new items increase rapidly. In other words, the problem in OM for personalization is to select a output ranking O_k from overall output ranking O. Therefore, we can consider ranking output probability as a one-hot vector, $p_k = 1$, if O_k is selected; otherwise, $p_{-k} = 0$. Equation (1) can be simplified as follows:

$$\arg \min_k \quad \alpha \sum_{r=1}^l \lambda_r + \sigma_k^2$$

subject to

$$(\forall r, j, j')| \sum_{i=1}^r \delta(O_{k,i}, I_j) - \sum_{i=1}^r \delta(O_{k,i}, I_{j'})| \le \lambda_r.$$

To solve this problem, we can use the greedy strategy. In this paper, we call this formulation greedy optimized multileaving (GOM). GOM has a low computation cost, because it does not need to solve a linear problem. This strategy is fast enough to compute in real-time in production recommender systems. Also, our implementation is publicly available [1].

3.2 Assigning Credit

To solve the second challenge, we propose a new definition for the credit function:

$$\delta(O_{k,i}, I_j) = -|\{j'|rank(O_{k,i}, I_{j'}) \le rank(O_{k,i}, I_j)\}|, \quad (3)$$

---

[1] https://github.com/mathetake/intergo

If there is no item $O_{k,i}$ in ranking I_j, then the credit value is $-(|I_j|+1)$. We call this credit personalization credit.

This definition is interpreted as considering a mutual interaction with input rankings, and gives credit to multiple rankings per click. Because relative ranking orders are used instead of the rankings' absolute position, the credits are calculated without position noise. For example, we set $I_1 = [1,2,\ldots,99,100,101,102]$, $I_2 = [1,2,\ldots,99,101,102,100]$, $I_3 = [1,2,\ldots,99,102,100,101]$, and $O_k = [1,2,\ldots,99,102,101,100]$. When 101 is clicked, the personalization credit values are $\delta(101,I_1) = -2$, $\delta(101,I_2) = -1$, and $\delta(101,I_3) = -3$. Conversely, the inverse credits are $\delta(101,I_1) = 1/101 = 0.0099$, $\delta(101,I_2) = 1/100 = 0.01$, and $\delta(101,I_3) = 1/102 = 0.0098$. Each absolute inverse credit value is much smaller and closer than personalization credits.

4 EXPERIMENTS

4.1 Offline Experiment Settings

In the offline experiment, we simulated user clicks, and evaluated several methods, which are compared below:

- TDM: Described in Section 2;
- GOM-I: GOM, using the inverse credit (2); and
- GOM-P: GOM, using the personalization credit (3).

These experiments assumed a practical environment that requires a low computation cost to generate rankings in real-time; therefore, we did not use OM for the performance comparisons. We used TDM for the performance comparison described in Section 4 because TDM has been examined in online settings[6].

Algorithm 1: User click simulation in personalized setting

input: the number of rankers n, ranking length l
$win = 0$
for $i = 1, \ldots, numeval$ **do**
    Select ranking index r randomly from 1 to n
    for $k = 1, \ldots, n$ **do** $credit[k] = 0$;
    for $j = 1, \ldots, numclick$ **do**
        $InitialRanking = $ generateRankingRandomly(l)
        for $k = 1, \ldots, n$ **do** $I_k = $ Shuffle($InitialRanking$);
        Get MultileavedRanking O from I
        Select one *item* from I_r at the top $x\%$ position randomly
        Click *item* in O and update sum of *credit* for all I
    end
    win += $|\{k|credit[k] > credit[r]\}|$
end
$accuracy = win/(numeval * (n-1))$
return $accuracy$;

The simulation steps are shown in Algorithm 1. We fixed the constant values $numeval = 100$, $numclick = 100$, number of output rankings = 10, and click bias probability $x = 80\%$. We evaluated the accuracy over the number of rankers 2, 3, ...20 when the ranking length was fixed at 10. We also evaluated the accuracy of ranker lengths 5, 15, ..., 195 when the number of rankers was fixed at 3.

Next, we evaluated insensitivity and bias. Insensitivity σ_k was divided by the square of the average credit μ_k^2, because the insensitivity is proportional to it.
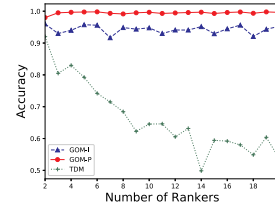

Figure 1: Accuracy versus the number of rankers for the fixed ranking length fixed at 10 using the random click simulation (averaged over 100 runs).
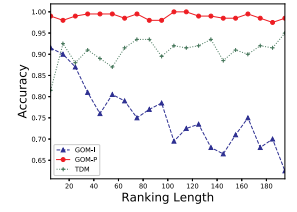
Figure 2: Accuracy versus the ranking lengths for the fixed number of rankers fixed at 3 using the random click simulation (averaged over 100 runs).
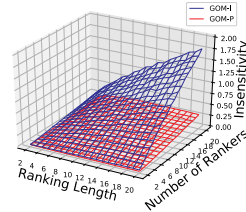

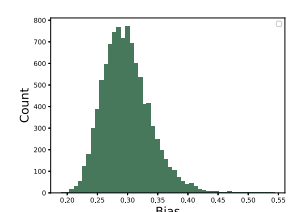**Figure 3: Insensitivity versus the number of rankers and the ranking lengths (averaged over 100 runs) .**

Figure 4: Bias distribution on generating GOM-P rankings 10,000 times.

4.2 Offline Experiment Results and Discussion

Figure 1 shows that GOM-P and GOM-I were more accurate compared to TDM. When the number of rankers increased, TDM's accuracy decreased. TDM credit caused this inaccuracy.

Figure 2 shows that the GOM-P and TDM methods had higher accuracy compared to GOM-I. When the ranking length increased, GOM-I's accuracy decreased. The noise of the inverse credit at the ranking's deep position of the ranking induced this inaccuracy. In contrast, TDM and GOM-P were stable over the ranking lengths.

Figure 3 shows the insensitivity for the number of rankers and the ranking length. GOM-P was sensitive compared to GOM-I in these cases; therefore, personalization credit achieved high sensitivity over the number of rankers and ranking lengths. This sensitivity resulted in the higher accuracy of GOM-P.

Figure 4 shows the bias distribution of GOM-P, which appears to be a normal distribution. The ideal bias distribution is that all biases are the same value at some point. The standard deviation of GOM-P was 0.039422, and the mean was 0.298570.

Interestingly, we found that hyperparameter α did not affect the accuracy, insensitivity, or standard deviation of bias. This means that we can set $\alpha = 0$. Then, to get a multileaved ranking, we only have to minimize the insensitivity in GOM; no parameter tuning is needed.

Table 1: Differences between the sum of credits. The values of GOM-P and TDM were all positive, while some values of GOM-I (written in blue) were negative. The GOM-P and TDM's results were consistent with previous CTR results of A/B testing (algo-A < algo-B < algo-C < algo-D < algo-E), but GOM-I was inconsistent. This means that the inverse credit function which was often used in previous studies is not appropriate for long ranking.

	GOM-I				GOM-P				TDM			
	algo-A	algo-B	algo-C	algo-D	algo-A	algo-B	algo-C	algo-D	algo-A	algo-B	algo-C	algo-D
algo-B	-14,962				21,111				615			
algo-C	-21,642	-6,680			36,561	15,450			1,259	644		
algo-D	-24,179	-9,217	-2,537		44,484	23,373	7,923		1,812	1,197	553	
algo-E	-5,246	9,716	16,396	18,933	52,597	31,486	16,036	8,113	2,117	1,502	858	305

4.3 Online Experiment Settings

We conducted an online experiment on Gunosy,[2] one of the most popular news applications in Japan. On this service, we recommend personalized news articles using a user click log [11]. The topics of the news articles are broad, ranging from entertainment to political opinions.

We prepared five personalization algorithms, and evaluated the effectiveness for each multileaving method. According to previous A/B testing, we knew the effectiveness of the algorithms as algo-A < algo-B < algo-C < algo-D < algo-E. In this A/B testing, we used the click-through rate (CTR) as the effectiveness of the personalization algorithms. CTR is a metric that divides article clicks by article impressions.

The online experiment was carried out for one week. We presented multileaved rankings for a particular portion of user requests, and received 10,826,923 article impressions. The deepest ranking length was up to 120.

4.4 Online Experiment Results and Discussion

Table 1 shows the credit differences between the evaluated algorithms in GOM-P, GOM-I, and TDM. The value 21, 111 for GOM-P indicates the credit difference between algo-B and algo-A. The differences of sum credit for GOM-P and TDM were all positive values, which means that the GOM-P and TDM's results were consistent with previous CTR results. In contrast, some values of GOM-I written in blue were negative, and inconsistent with previous outcomes. In this online experiment setting, the number of rankers was five, and the ranking length was longer than 100. According to the offline experiments, this small number of rankers caused the consistent TDM results, while the long ranking length caused the inconsistent GOM-I results.

Figure 5 shows that the multileaving methods converged much faster than A/B testing (< 1/10). These p-values were calculated using bootstrap subsampling (sampling size = 50). The x-axis represents user number N, and the y-axis represents the p-value averaged over all pairs of the algorithms. The multileaving p-value was from paired t-tests. In the A/B testing, $N/2$ users were assigned per algorithm, and the p-value came from unpaired t-tests.

One of the reasons for the slow convergence of A/B testing was group bias. The users assigned to algo-C in the A/B testing tended to be inactive. The multileaving method did not suffer from this group bias, because different algorithms were examined in the same user groups.
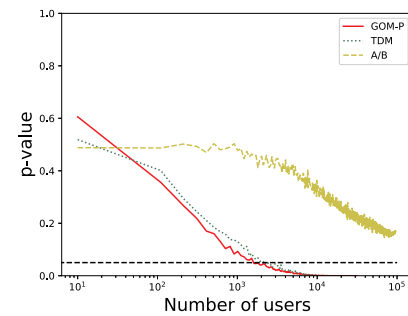

Figure 5: P-values versus the number of users. The multileaving methods converged much faster than A/B testing (< 1/10).

5 CONCLUSION

In this paper, we proposed a new multileaving method for personalized rankings that requires a high degree of freshness, many hyperparameters, and a long ranking length for a rich user experience. To the best of our knowledge, few theoretical multileaving studies have been conducted for these personalized settings.

We clarified the challenges of applying this method for personalized rankings. We then formalized this personalized multileaving problem, and introduced a new credit feedback function. We also conducted an offline experiment, which showed that the proposed method was stable, regardless of the number of rankers or ranking length. Finally, we conducted an online experiment in the large-scale recommender system. We confirmed that the proposed method could evaluate rankings precisely using a significantly smaller sample size than A/B testing.

In future work, we will conduct further offline click simulations for various types of click bias. In addition, we will try to apply GOM's feedback schema to speed up online learning for ranking. We will also investigate whether the GOM results agree with specific measures, such as Normalized Discounted Cumulative Gain (NDCG).

---

[2]https://gunosy.com

REFERENCES

[1] Juliette Aurisset, Michael Ramm, and Joshua Parks. 2017. Innovating Faster on Personalization Algorithms at Netflix Using Interleaving. https://medium.com/netflix-techblog/interleaving-in-online-experiments-at-netflix-a04ee392ec55

[2] Brian Brost, Cox J. Ingemar, Yevgeny Seldin, and Christina Lioma. 2016. An Improved Multileaving Algorithm for Online Ranker Evaluation. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[3] Olivier Chapelle, Thorsten Joachims, Filip Radlinski, and Yisong Yue. 2012. Large-scale Validation and Analysis of Interleaved Search Evaluation. *ACM transactions on information systems* 30, 1 (2012).

[4] Abhinandan Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google News Personalization: Scalable Online Collaborative Filtering. In *Proceedings of the 16th International Conference on World Wide Web*. ACM, New York, NY, USA.

[5] Mihajlo Grbovic and Haibin Cheng. 2018. Real-time Personalization using Embeddings for Search Ranking at Airbnb. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[6] Tomohiro Manabe, Akiomi Nishida, Makoto P. Kato, Takehiro Yamamoto, and Sumio Fujita. 2017. A Comparative Live Evaluation of Multileaving Methods on a Commercial cQA Search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[7] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. 2017. Embedding-based news recommendation for millions of users. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA.

[8] Filip Radlinski and Nick Craswell. 2013. Optimized Interleaving for Online Retrieval Evaluation. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*.

[9] Filip Radlinski, Madhu Kurup, and Thorsten Joachims. 2008. How Does Click-through Data Reflect Retrieval Quality?. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*.

[10] Anne Schuth, Floor Sietsma, Shimon Whiteson, Damien Lefortier, and Maarten de Rijke. 2014. Multileaved Comparisons for Fast Online Evaluation. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*.

[11] Takeshi Yoneda, Shunsuke Kozawa, Keisuke Osone, Yukinori Koide, Yosuke Abe, and Yoshifumi Seki. 2019. Algorithm and Architecture of Immediate Personalized News Recommendation. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI)*.