Online Ranking Combination

Erzsébet Frigó
frigo.erzsebet@sztaki.hu
Institute for Computer Science and Control (MTA SZTAKI)
Budapest, Hungary

Levente Kocsis
kocsis@sztaki.hu
Institute for Computer Science and Control (MTA SZTAKI)
Budapest, Hungary

ABSTRACT

As a task of high importance for recommender systems, we consider the problem of learning the convex combination of ranking algorithms by online machine learning. In the case of two base rankers, we show that the exponentially weighted combination achieves near optimal performance. However, the number of required points to be evaluated may be prohibitive with more base models in a real application. We propose a gradient based stochastic optimization algorithm that uses finite differences. Our new algorithm achieves similar empirical performance for two base rankers, while scaling well with an increased number of models. In our experiments with five real-world recommendation data sets, we show that the combination offers significant improvement over previously known stochastic optimization techniques. Our algorithm is the first effective stochastic optimization method for combining ranked recommendation lists by online machine learning.

CCS CONCEPTS

• **Information systems** → *Collaborative filtering*; • **Theory of computation** → *Online learning algorithms.*

KEYWORDS

ranking; combination; RFDSA

ACM Reference Format:
Erzsébet Frigó and Levente Kocsis. 2019. Online Ranking Combination. In *Thirteenth ACM Conference on Recommender Systems (RecSys '19), September 16–20, 2019, Copenhagen, Denmark.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3298689.3346993

1 INTRODUCTION

A milestone in the research of recommendation algorithms, the Netflix Prize Competition [4] had high impact on research directions. The target of the contest was based on the one to five star ratings given by users, with one part of the data used for model training and the other for evaluation. As an impact of the competition, tasks now termed batch rating prediction were dominating research results. However, real systems differ not just in that the user feedback is implicit, but also in that they process data streams where users request one or a few items at a time and get exposed to new information that may change their needs and taste when they return to the service next time. Furthermore, an online trained model may change and return completely different lists for the same user even for interactions very close in time.

The difficulty of evaluating streaming recommenders was first mentioned in [18], although the authors evaluated models by offline training and testing split. Ideas for online evaluation metrics appeared first in [21, 22, 29]. In online or prequential evaluation [9], which has grown in popularity, the ranking measure is computed from a sequence of examples. For each example in the sequence, the recommender system provides a top-k list of items to the active user. The list is evaluated against typically a single relevant item that the user interacted with. Then, the user-item interaction is added to the previously available data, and the recommender system is able to update its model.

Recommender systems often rely on an ensemble of base ranking algorithms. For instance, in the Netflix prize competition, considerable effort went into choosing the algorithms to the blend and combining them [28]. In an online scenario, the environment for a combination algorithm is non-stationary: not only the user preferences and item popularities, but also the base ranking models change in time. Therefore, the combination of the base algorithms also needs to be updated. While it is infeasible to update the parameters of the combination with the computationally intensive blending approaches used in batch settings, convex combination of the base models often lead to satisfying results. In summary, we consider online convex combination algorithms for (implicit feedback) recommenders under prequential evaluation.

From the machine learning point of view, the main difficulty of combining ranked recommendation lists is that the typical ranking measures, such as NDCG [12], are not continuous, making their optimization a difficult task. In this paper, we compare and identify the numerical issues of two strategies to optimize for non-continuous rewards. The first approach uses exponentially weighted forecasters, which explore the weight space globally and do not rely on the existence of a gradient of the reward function. The second class of methods uses gradient descent to maximize the reward.

Exponentially weighted algorithms (EWA) [7] optimize ranking combination weights by exploring the weight space globally. EWA was shown to be close to optimal for Lipschitz-continuous environments [19]. We will show that EWA is able to optimize ranking combination as well, under certain assumption (see Proposition 4.1). However, the number of combinations that needs to be evaluated to fulfill the assumption grows exponentially with the number of base rankers. Therefore, it is not practical in a real application if more base rankers are employed.

To be able to handle a larger number of base rankers, we turn our attention to the second approach, local optimization by gradient based methods. In particular, we start with the Resilient Simultaneous Perturbation Stochastic Approximation (RSPSA) algorithm [15], which was used for optimizing model parameters in games. While RSPSA was shown to cope with non-continuous rewards, it is non-trivial whether it can cope with ranking functions as well. Indeed, we observe empirically that RSPSA does not scale well for ranking prediction. The reason for this is that ranking functions have many flat regions with respect to individual combination weights.

Our method, Resilient Finite Difference Stochastic Approximation (RFDSA+), is the first effective stochastic optimization method for combining ranked lists. To improve the scalability properties of RSPSA, we switch from simultaneous perturbation to finite differences to identify flat regions with respect to a given weight. In this way, we eliminate the noise of the perturbation of other weights and concentrate always only on optimizing a single weight at a time. We show empirically that RFDSA+ achieves near optimal performance when two base rankers are combined, and scales well with the number of base rankers.

The article is organized as follows: after discussing the related research in Section 2, we formalize our framework in Section 3. Exponentially weighted algorithms are discussed in Section 4, where in Proposition 4.1 we show the theoretical guarantee of EWA. Gradient based algorithms are discussed in Section 5. Our proposed algorithm RFDSA+ is described in Section 6. Empirical evaluation highlighting the strength of RFDSA+ is provided in Section 7. Some conclusions and discussion of future research close the paper in Section 8.

2 RELATED RESEARCH

Research on incremental recommender algorithms with prequential evaluation scenario has gained popularity in recent years. There are several papers that use prequential evaluation [3, 5, 13, 21, 22, 31], however, only [22] considers the issue of combining multiple base rankers. The latter will be discussed in more detail in Section 5.1, and evaluated empirically in Section 7.

Ranking combination has received considerable attention during the Netflix prize competition, when the approach of [28] was essential for the winning entry. In the batch setting, one of the later approaches that can be adapted naturally to an online scenario is [6]. The authors use exponentially weighted forecaster, and use the cumulative loss of each base algorithm to compute its score in the convex combination. One can notice that any arbitrary linear shift of the scores of a base algorithm would leave its cumulative loss unchanged, but it would affect the base algorithm contribution to the mix. Therefore, the algorithm seems somewhat less sound, nevertheless, may still perform reasonably well on some practical instances. We will describe the algorithm more formally in Section 4.3, and evaluate (for implicit feedback problems) empirically in Section 7.

In the online setting, ranking combination was proposed by [25, 30] using dueling bandits. Their approach assumes that the loss functions are convex and stationary. Neither assumption seems reasonable for most ranking measures in a real application. There are several algorithms in the literature of online learning that can be considered for combining ranking models. [2] considered a two-point approximation of the gradient for convex functions. The ranking measures are not convex, nevertheless, the algorithm is similar to SPSA [27] that have been applied to optimizing non-convex functions as well. We will discuss the algorithm in Section 5.2. The exponentially weighted algorithm was applied to optimize (non-convex) Lipschitz-continuous functions [19] and it has $O(\sqrt{T})$ guarantees in full-information setting, where T is the length of the episode. Full information setting would imply, however, evaluating a prohibitively large number of points when the number of base rankers is slightly larger. There are bandit variants as well [14] that evaluate only one point per iteration, however, they scale badly on error. The regret bound for the continuum-armed bandits is $O(T^{(N+1)/(N+2)})$ [14], where N is the dimensionality of the problem. The exponentially weighted algorithms will be considered in Section 4.

Finally, there are a large number of stochastic approximation algorithms that can, in principle, be applied to online ranking combination. Unfortunately, neither of them is straightforward to use for ranking functions such as NDCG that are not continuous, and even a smoothed cumulative ranking reward function can be non-convex as well. In most games, the reward is also non-continuous, for example, 1/0 for win/loss, or a discrete number of points (or money) that can be won in a card game. The algorithm RSPSA [15] was proposed for (offline) optimization of some parameter of a poker playing program. Our proposed algorithm, RFDSA+, builds on the idea of RSPSA but considers one weight at a time, to remedy the problems of past algorithms in handling flat areas in ranking functions.

3 PROBLEM SETUP

We consider the online combination of the ranked list of multiple base recommender algorithms. As soon as the base algorithms give a prediction, we have to apply and potentially re-learn the combination weight on the fly. In contrast to the typical batch learning tasks where we can for example perform grid search by using a large amount of past training data, closer to a real recommender system operation, in our task, we process the recommendation requests and the feedback as a sequence in time. Compared to batch learning, the advantage of the online methods is that they can adapt faster to concept drifts [8] that can rearrange the relative strength of the different base models.

Both batch and prequential evaluation rely on a set of recorded user-item interactions. For batch evaluation, one splits the data in a training and a test set, and trains the algorithms on the former and tests on the latter. Conversely, for prequential evaluation, we test algorithms sequentially on each data point, and potentially use all preceding data points for training. Since often, the user selects a single item only, we will consider implicit feedback evaluation metrics with only one relevant item, but the evaluation can easily be generalized for the case when the user takes multiple choices or when the feedback is explicit. Prequential evaluation is closer to a real application, since in practice, user interaction occurs sequentially. Algorithms can also exploit the most recent data. It is true for both evaluation methods that recommendation is made before revealing the choice of the user, that is a given user-item interaction is processed independently of what the system recommends.

Given a chronologically ordered data set with T records, prequential evaluation is an episode with T rounds. In each round t, we take the following steps.

(1) We observe the next user-item pair from the data set, and set the active user accordingly.
(2) We query the recommender system for a top-K recommendation for the active user.
(3) We evaluate the output recommendation list against the single relevant item j_t that the user interacted with.
(4) Finally, we reveal the relevant item j_t to the recommender system, and allow to update the model using the additional user-item pair.

In the context of convex combination algorithms, we consider N base ranking algorithms, and the ith base algorithm is denoted by $\mathcal{A}_i$. In each round $t = 1, \ldots, T$, first, each base algorithm $\mathcal{A}_i$ assigns a score x_{tij} to each item j. After that, the convex combination algorithm assigns the weight θ_{ti} to each algorithm $\mathcal{A}_i$. The weights form an N-dimensional vector $\boldsymbol{\theta}_t = (\theta_{t1}, \ldots, \theta_{tN})$. The parameter space is $\boldsymbol{\theta}_t \in \Theta = \mathbb{R}^N_{0+}$. The combined score of item j in round t is $x_{tj} = \sum_{i=1}^N \theta_{ti} x_{tij}$. The top lists are generated by sorting the items by the combined scores in descending order.

After the active user's preferred item is revealed, the combination algorithm collects the reward r_t, which depends on the top list generated and on the user's choice. With an abuse of notation, we will denote the reward of a base ranker $\mathcal{A}_i$ by r_{ti}, the reward corresponding to a weight assignment $\boldsymbol{\theta}$ by $r_t(\boldsymbol{\theta})$, and the reward obtained by a combination algorithm C by $r_t(C)$. The cumulative reward collected up to round t is $R_t = \sum_{\tau=1}^t r_\tau$. We let R_{ti}, $R_t(\boldsymbol{\theta})$, and $R_t(C)$ denote the cumulative reward corresponding to a base ranker, a weight vector, and a combination algorithm.

There are several choices of ranking measures. A popular choice, which we use in our experiments, is NDCG@K [12]. In prequential evaluation, we assume the worst scenario that there is only one item with non-zero label in each round t, namely j_t. The NDCG@K of a permutation π_t of the items reduces to

$$r_t = \text{NDCG@K}(\pi_t) = \begin{cases} 1/\log_2(rank_{\pi_t}(j_t)+1) & \text{if } rank_{\pi_t}(j_t) \le K, \\ 0 & \text{otherwise,} \end{cases}$$

as there is always exactly one relevant item and hence the ideal DCG is equal to one.

4 EXPONENTIALLY WEIGHTED BASELINE ALGORITHMS

The first set of baseline rank combination algorithms in this section rely on the exponentially weighted forecaster [7]. They explore the weight space globally, without relying on the existence of a gradient of the reward function. In Proposition 4.1, we will also show that in the case of two base recommenders, the exponentially weighted combination achieves near optimal performance.

4.1 ExpA

The simplest choice to deal with multiple base rankers is to use the exponentially weighted forecaster on the rankers. Accordingly, the combination algorithm, denoted by ExpA, selects base ranker $\mathcal{A}_i$ in round t with probability

$$p_{ti} = \frac{e^{-\eta_t \sum_{\tau=1}^{t-1} r_{\tau i}}}{\sum_{j=1}^N e^{-\eta_t \sum_{\tau=1}^{t-1} r_{\tau j}}}. \quad (1)$$

Selecting base ranker $\mathcal{A}_i$ in round t means setting $\theta_{ti} = 1$ and $\theta_{tj} = 0$ for $j \neq i$. The algorithm is guaranteed to achieve a cumulative reward that is not worse than the cumulative reward of the best base rankers by an additive $O(\sqrt{T})$ term in expectation [7].

4.2 ExpW

While ExpA can locate the best base ranker, we can hope that a convex combination of the rankers can achieve a better performance than any single ranker. We choose a finite set of points $P \subset \Theta$ and apply the exponentially weighted forecaster to P to choose the weight combinations $\boldsymbol{\theta}_t \in P$ to play. If an appropriately large number of points are chosen, and the cumulative reward function $R_T(\boldsymbol{\theta})$ (as a function of $\boldsymbol{\theta}$) is sufficiently smooth, then the algorithm, denoted by ExpW, will achieve a cumulative reward that is close to that of the optimal convex combination. The following proposition formalizes this statement.

Proposition 4.1. *Let $P \subset \Theta$ be a finite set such that*

$$\mathrm{E}\left[\max_{\boldsymbol{\theta} \in \Theta} R_T(\boldsymbol{\theta}) - \max_{p \in P} R_T(p)\right] \le \sqrt{T}. \quad (2)$$

Then the regret of the exponentially weighted forecaster applied on P is bounded by

$$\mathrm{E}\left[\max_{\boldsymbol{\theta} \in \Theta} R_T(\boldsymbol{\theta}) - R_T(\text{ExpW})\right] \le \tilde{O}\left(\sqrt{T}\right). \quad (3)$$

The proof follows by putting together the regret bound of the exponentially weighted forecaster and inequality (2).

For a sufficiently large T, function $R_T(\boldsymbol{\theta})$ is fairly smooth in practice, as observed in Section 7.3. For two base rankers, the parameter space can be represented by a one dimensional simplex, i.e. a section. Then, a uniform grid with $O(\sqrt{T})$ gridpoints can be sufficient, if the cumulative function acts like a Lipschitz function on the grid points. This latter condition will often be true (see also Figure 1). However, with more base rankers, the number of points required for a Lipschitz-like cumulative function is $\Omega(T^{(N-1)/2})$. Since ExpW needs to evaluate the reward in each point, the number of evaluations scales exponentially with the number of base rankers.

4.3 ExpAW

In [6], the authors proposed an algorithm that can be regarded as a mix of ExpA and ExpW. The algorithm, denoted here by ExpAW, relies on the cumulative performance of the base rankers (as ExpA), but it is used as the weight of the base ranker, instead of using it as selection probability. The weight of base ranker $\mathcal{A}_i$ in round t is

$$\theta_{ti} = \frac{e^{-\eta_t \sum_{\tau=1}^{t-1} r_{\tau i}}}{\sum_{j=1}^N e^{-\eta_t \sum_{\tau=1}^{t-1} r_{\tau j}}}. \quad (4)$$

It is easy to see that reward of a base ranker does not change if the scores of rankers are scaled by some factor. However, the scaling will affect the reward of the combination algorithm in an arbitrary way. Nevertheless, with a reasonable normalization, the algorithm may still lead to a decent performance, and it is less likely to be affected by an increase in the number of base rankers.

5 GRADIENT BASELINE ALGORITHMS

In this section, we present the second set of baseline rank combination methods, which are guided by the gradient of the reward function. In the first subsection, we describe algorithm (SGD) that computes the gradient of a surrogate to the reward. The next two algorithms (SPSA and RSPSA) are stochastic approximation algorithms that approximate the gradient by finite differences. We mention that our new algorithm RFDSA+ builds on RSPSA, extending it to deal with the difficulties of flat regions in the ranking functions.

5.1 SGD

We call our first combination algorithm SGD, since it uses stochastic gradient descent for the mean squared error (MSE) as a surrogate to the reward. The target for the current item is set to 1, and the targets for a set of randomly sampled negative items is set to 0. After seeing a user-item pair, a stochastic gradient step is taken to minimize the MSE between the ranking score (x_{tj}) and the target value. The algorithm was used by [23] for matrix factorization and by [22] for online combination.

We do not expect the algorithm to have difficulty with a large number of base rankers. However, minimizing the surrogate loss may not result in a sufficiently good optimization of the original reward function.

5.2 SPSA

The gradient of most ranking functions with respect to the combination weights is typically zero in most points where it exists. However, if we average over more time steps, it starts to 'smooth out'. It still cannot be computed in a closed form, but it can be approximated by finite differences. For online optimization of convex functions, [2] suggested the gradient to be approximated by simultaneous perturbation, with an online gradient step taken in the approximated direction. For non-convex optimization, a similar algorithm is known as Simultaneous Perturbation Stochastic Approximation (SPSA) [27]. The approximated gradient g_{ti} is given by

$$g_{ti} = (r_t(\boldsymbol{\theta}_t + c_t\boldsymbol{\Delta}_t) - r_t(\boldsymbol{\theta}_t - c_t\boldsymbol{\Delta}_t))/(c_t\Delta_{ti}),$$

where c_t is an appropriately decreasing sequence,

$$\boldsymbol{\Delta}_t = (\Delta_{t1}, \dots, \Delta_{tN}),$$

and Δ_{ti} are ± 1 valued unbiased Bernoulli random variables.

5.3 RSPSA

In SPSA, especially with non-smooth functions, the difficulty lies in choosing the appropriate perturbation. The sum of ranking reward functions is a step function. If the perturbation size is too small, we might stuck on a plateau and can not find the right direction. If the perturbation is too large, we miss local optima. The appropriate perturbation step size might differ depending on the coordinate and time.

The RSPSA algorithm was proposed in [15] for games, which also have a discrete reward (e.g., 1 for win, 0 for loss). The algorithm combines the simultaneous perturbation approximation with the Resilient Backpropagation (RPROP) [11] update rule. In RPROP, we assign a distinct step size to each weight. Informally, if the direction of the gradient changes, then the step size is decreased. Otherwise, the step size is increased. The weight update depends only on the sign of the gradient, and the step size determines how much the weight changes. In RSPSA, the perturbation size for each weight is connected to the step size, solving the above mentioned difficulty. The RPROP update rule is designed for batch update, and therefore, in our setting, we use mini batches to collect the gradients before an update.

6 OUR METHOD: RFDSA+

We designed our new method by observing the behavior of RSPSA for ranking combination. One of the strengths of the RPROP update rule is that it is increasing the update steps on a large plateau, and taking larger steps in the directions of the gradient. Ranking functions as function of a combination weight consist of constant intervals. However if the perturbation is sufficiently large, the averaged gradient estimate will be non-zero. If the step size for a weight is small in a flat area, then it should be increased in order to escape the flat area, but also in order to be able to estimate the right direction. In other words, the weight needs a sufficiently large perturbation to be able to influence the ranking function.

However, we observed that in RSPSA, the estimated direction changes often in the flat area, and the step size in fact decreases. To illustrate the problem, consider a ranking function that is completely flat in the direction of some but not all coordinates in the neighborhood of the current $\boldsymbol{\theta}_t$. In this case, the direction of the estimated gradient of the 'flat' coordinates becomes an unbiased Bernoulli variable, since even if the ranking function is completely flat with respect to coordinate i, the numerator of g_{ti} will still be non-zero because of the non-flat coordinates. However, the numerator will be independent of the randomly chosen direction of Δ_{ti}, and hence g_{ti} will simply mirror the random variable Δ_{ti}.

To remedy the problem of flat regions, we switch from simultaneous perturbation to finite differences in order to identify that the ranking function is flat with respect to the weight in question. Note that by perturbing just one weight, we eliminate the noise coming from the perturbation of the other weights. If we detect a flat region, then we increase the step size.

The pseudocode of the RFDSA+ is provided in Algorithm 1. The key differences to RSPSA are switching from simultaneous perturbation to finite differences (line 7–8), and handling the flat regions (line 22–23). The RPROP update is given by line 11–28.

The algorithm has four parameters: the mini-batch size B, the initial step size δ_0, and the step size adjustment variables η^+ and η^-. For noise functions, typical values are $\eta^+ = 1.1$ and $\eta^- = 0.85$ [15]. The initial value of the step size has minimal influence, since it is quickly adjusted; it is set to $\delta_0 = 0.1$. The size of the mini batch will be chosen 1,000 in the experiments, the same as for SPSA and RSPSA. The length of an episode T, and the number of the base rankers N is determined by the problem.

The key variables of the RFDSA+ algorithm are the step sizes δ_i, corresponding to each weight θ_i. The auxiliary variables s_i store the previous weight update and are used for identifying a change in the direction of the partial derivatives. During a mini batch, the negative partial derivatives are collected in the variables g_i.

The RFDSA+ algorithm starts with an initialization phase in line 1–4. After every user interaction, at time t, the partial derivatives are computed as follows. For each base ranker i, we perturb its weight by twice the corresponding step size (line 7). The coupling factor 2 is standard for RSPSA [15], but slightly different values can be used as well. We use one-sided positive perturbation in the description of the algorithm. Using one-sided perturbation halves the number of evaluations needed. With one-sided perturbation, it is more natural to choose the direction randomly ($\pm 2\delta_i$) valued Bernoulli random variable. The current description was chosen for brevity. The partial derivatives g_i are updated in line 8, using the finite difference estimator.

At the end of each mini batch, the weights θ_i and the step sizes δ_i are updated according to the RPROP rule [11] in lines 11–28, independently for each component i. The auxiliary variable h detects a change of direction in the partial derivative. If there is no change (lines 13–15), the step size is increased, and the weight θ_i will be updated in the direction of the derivatives with the amount determined by the step size. If there is change in the direction, then the step size is decreased, and the weight is left unchanged. The weight will be updated after the next mini batch (line 20). The key modification that deals with flat regions in the partial derivatives is shown in lines 22–23. Accordingly, the step size is increased if the partial derivative is 0 during the mini batch. Detecting the flat region is made possible by using finite difference estimation, instead of simultaneous perturbation. The actual weight update is shown in line 25.

7 EXPERIMENTS

In this section, first we empirically investigate how well the combination algorithms perform for two base rankers, compared to the optimal (static) combination. Then we analyze how the combination algorithms scale when a larger number of base rankers are available.

7.1 Data sets

All data sets consists of time-ordered sequence of user-item pairs. Only the first occurrence of a user-item pair is included. The task at a certain point of time is to rank the available items for the current user. After a top list is provided by a particular algorithm, a reward is obtained using $NDCG@100$ as ranking measure (see Section 3). In our case, there is only one item with non-zero label (the one from the current user-item pair). Following the evaluation step, the item is revealed to the base rankers and the combination algorithm, allowing them to update their model.

In these experiments, we use three data sets from the Amazon collection (CDs and Vinyl; Movies and TV; Electronics [20]), the 10M MovieLens data set[1], and a twitter data set where the items are defined by the hashtags used in tweets.

[1]http://grouplens.org/datasets/movielens/

```
1  for i = 1 to N do
2      $\theta_i \leftarrow 1/N$; $g_i \leftarrow 0$
3      $s_i \leftarrow 0$; $\delta_i \leftarrow \delta_0$
4  end
5  for t = 1 to T do
6      for i = 1 to N do
7          $\theta^+ \leftarrow \theta$; $\theta^+_i \leftarrow \theta_i + 2\delta_i$
8          $g_i \leftarrow g_i + (r_t(\theta^+) - r_t(\theta))/(2\delta_i)$
9      end
10     if $t \mod B = 0$ then
11         for i = 1 to N do
12             $h \leftarrow g_i s_i$
13             if $h > 0$ then
14                 $\delta_i \leftarrow \eta^+ \delta_i$
15                 $s_i \leftarrow sign(g_i)\delta_i$
16             else if $h < 0$ then
17                 $\delta_i \leftarrow \eta^- \delta_i$
18                 $s_i \leftarrow 0$
19             else
20                 $s_i \leftarrow sign(g_i)\delta_i$
21             end
22             if $g_i = 0$ then
23                 $\delta_i \leftarrow \eta^+ \delta_i$
24             else
25                 $\theta_i \leftarrow \theta_i + s_i$
26             end
27             $g_i \leftarrow 0$
28         end
29     end
30 end
```

Algorithm 1: RFDSA+

7.2 Base rankers

We rely on two basic classes of collaborative filtering models: item based nearest neighbor (ITEM2ITEM) [26] and matrix factorization [1]. These two classes of methods represent the most successful and most popular collaborative filtering algorithms[2] [17, 24]. In addition to the two techniques, we also include temporal popularity (denoted POP), which records how many times an item was visited in the preceding time window.

For ITEM2ITEM, we use a time-decayed item-to-item similarity function, the model being updated every day. When computing the score for an item, we consider the similarity to all items previously visited by the user. Thus, this algorithm also incorporates the recent history.

We include four matrix factorization variants: online matrix factorization (OMF) [23], online asymmetric matrix factorization (OAMF) [16], batch matrix factorization (MF), and (batch) implicit alternating least squares (IALS) [10]. All variants use latent factors with ten dimensions. The online variants update once after

[2]For particular data sets, there may be superior algorithms, especially in batch settings. The two main base rankers considered are representatives of two main approaches to collaborative filtering, and have natural incremental versions. None of the combination algorithms exploit the particular base rankers, thus replacing the base rankers is straightforward.

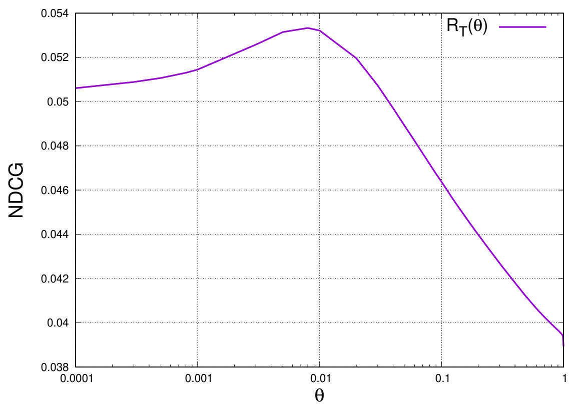

Figure 1: Reward with various combination coefficients (θ) for the combination of OMF **and** ITEM2ITEM **on the Amazon-CD set. In the figure, θ denotes the normalized weight of the** OMF **base ranker. The normalized weight for** ITEM2ITEM **is $1 - \theta$. $R_T(0) = 0.03434$.**

every user-item pair. The batch variants retrain their models after every 100,000 time steps, using a required number of iterations. We use stochastic gradient descent for OMF, OAMF and MF with the current item from the data set designated as positive item, and additional negative items sampled randomly [23].

The parameters of the base rankers are optimized for each data set. In the combination, the scores of the base rankers are normalized by the standard deviation.

7.3 Combination of two models

We show our results for combining the two base models OMF and ITEM2ITEM. We let θ denote the weight of OMF in the convex combination. The average cumulative reward, depending on θ, is shown for the Amazon-CD data set in Figure 1. Interestingly, the optimum is reached for a combination that puts heavy weigh to ITEM2ITEM, even though OMF alone performs better than ITEM2ITEM.

The average cumulative reward of the combination algorithms is shown in Figure 2. The peculiar shape in the first three years is due to the low amount of data collected and the more significant changes in the data distribution. We observe the relative order of the base algorithms changes over time: at first OMF is better, then ITEM2ITEM, and then OMF again. This shows that selecting an algorithm on partial data, and using only that algorithm later is a poor choice. EXPA follows the better base algorithm, being slightly worse than that due to exploration. EXPW[3] achieves a performance that equals to the best static convex combination (cf. Figure 1). EXPAW is on par with EXPW in the beginning, but its performance deteriorates later. This is natural, since it is choosing a larger weight for OMF due to the superior performance of OMF, despite that the actual optimum is to assign a large weight to ITEM2ITEM, as seen in Figure 1. SGD has similar performance to EXPW, giving also a larger weight to OMF. This is possibly because SGD and OMF optimize the same surrogate loss function.

[3]For EXPW, the set of points P consisted of a uniform grid with 100 points.

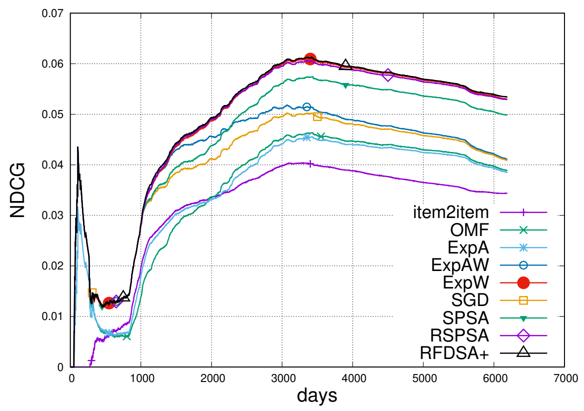

Figure 2: Average cumulative NDCG of the ranking algorithms on the Amazon-CD set.

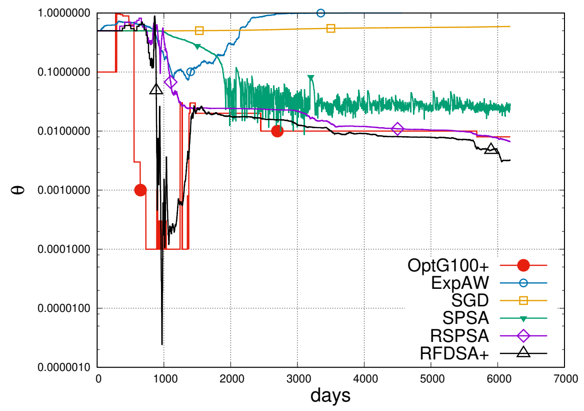

Figure 3: The weight assignment of the ranking algorithms on the Amazon-CD set. OPTG100+ **corresponds to the optimal weight assignment over 100 uniform grid points, with a few additional points chosen near the presumed optimum. In the figure, θ denotes the normalized weight of the** OMF **base ranker. The normalized weight for** ITEM2ITEM **is $1 - \theta$.**

The weight assignment of the combination algorithms is shown in Figure 3. The figure includes additionally an optimal static weight assignment, i.e. $\theta_t = \text{argmax}_{\theta \in P} R_t(\theta)$. By analyzing the weight assignment of the three combination algorithms that optimize NDCG directly (SPSA, RSPSA and RFDSA+), we observe that all give ITEM2ITEM a large weight, although the weights for SPSA are further away from the optimum. Consequently, we notice in Figure 2 that the three algorithms perform well, RSPSA and RFDSA+ matching the optimal performance of EXPW.

7.4 Scaling

We analyze the scaling of the combination algorithms in two ways: (1) by including an increasing number of OMF base rankers (differing only in the random initialization) next to ITEM2ITEM, and (2) by including all six base rankers in the mix.

Table 1: Combination of six base rankers on five data sets. The average NDCG of the base rankers is shown at the top of table, while the average NDCG of the combination algorithms at the bottom.

Algorithm	Amazon-CD	Amazon-Movies	Amazon-Electro	MovieLens	Twitter
ITEM2ITEM	0.0343	0.0350	0.0156	0.1445	0.0221
OMF	0.0389	0.0440	0.0222	0.1357	0.3528
POP	0.0628	0.0663	0.0347	0.0857	0.3486
OAMF	0.0318	0.0320	0.0160	0.1717	0.3118
MF	0.0052	0.0086	0.0056	0.0051	0.0055
IALS	0.0046	0.0075	0.0060	0.0053	0.0054
EXPA	0.0628	0.0663	0.0347	0.1717	0.3486
EXPAW	0.0628	0.0664	0.0347	0.1717	0.3486
SGD	0.0640	0.0674	0.0353	0.1568	0.3563
SPSA	0.0696	0.0692	0.0349	0.1678	0.3683
RSPSA	0.0640	0.0670	0.0396	0.1435	0.4468
RFDSA+	0.0880	0.0882	0.0452	0.1879	0.4601

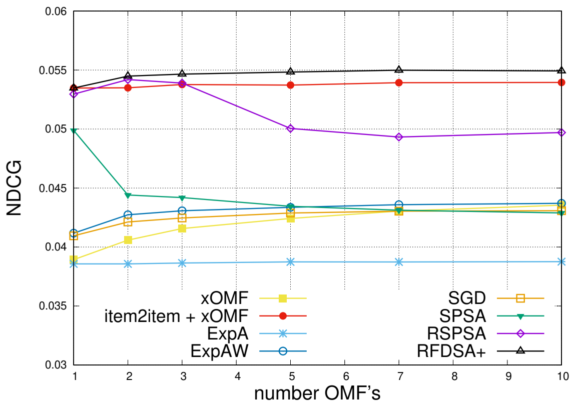

Figure 4: Average NDCG of the ranking algorithms on the Amazon-CD set with varying number of OMFs. The combination includes one ITEM2ITEM and one to ten OMF base rankers. In the case of xOMF there is only one OMF, but the dimension of the latent factors is increased from 10 to the range of 10–100.

In the first case, assuming that the various OMF models achieve similar performance, one expects that the optimal weight for ITEM2ITEM stays relatively the same, with the weight of one OMF from the previous section divided among the multiple instances. The difficulty here is that the proper weight assignment (for ITEM2ITEM) needs to be found in a space with larger dimensionality. For larger dimensions, placing grid points that cover the parameter space sufficiently would require exponential number of evaluations, thus we do not include EXPW in this experiment. The performance of the other combination algorithms is shown in Figure 4.

We observe that the ranking performance of RFDSA+ is not dropping as the number of OMFs increases. It is even able to use the slight variation in the OMFs to increase the performance slightly. The performance of SPSA and RSPSA deteriorates significantly as more OMFs are included in the mix. EXPA, EXPAW and SGD all cope well with the increased dimension, but their performance is much weaker overall than that of RFDSA+. The relative invariance of EXPA underlines that the individual OMF rankers achieve similar performance (we checked that the variance of their NDCG score is indeed very small). We added two further baselines to the figure: xOMF is a variant of OMF with increased latent vector dimension, and ITEM2ITEM+xOMF, a combination using RFDSA+. We observe that the individual performance of an online factor model increases with the dimension of the latent vectors. However, in combination with ITEM2ITEM, it is better to use many smaller models than one big one, assuming that they are combined with an algorithm such as RFDSA+ that scales well. Results on the other data sets are similar and omitted due to space limitations.

Next, we show the performance of the combination algorithms when all the six base rankers are used in Table 1. First, we notice that the individual performance of the batch base rankers (MF and IALS) is poor for all data sets. The performance of the other base rankers vary, depending on the data set. Regarding the performance of the combination algorithms, we can draw somewhat similar conclusion as for Figure 4: RFDSA+ has significantly better performance for all data sets compared to other combination algorithms. We also note that the improvement in performance over the best individual base ranker is considerable for all data sets. EXPA achieves approximately the performance of the best individual ranker. EXPAW and SGD cope reasonably well with more base rankers, but their performance is not exceeding by much the performance of the base ranker. SPSA and RSPSA (which were performing well for two base rankers) are not performing particularly well when a larger number of models are included in the mix.

8 CONCLUSIONS

In this paper, we have considered the task of learning the online convex combination of base recommender algorithms by stochastic optimization. For the case of two base rankers, we have shown that the class of exponential weighted algorithms attains close to optimal performance. However, the algorithm cannot be applied in real application with a larger number of base rankers, because of the exponential number of evaluations needed. To remedy the scaling problem, we have proposed a new algorithm RFDSA+. The algorithm uses finite differences to estimate the gradient of the ranking reward, and the RPROP update rule to adjust the combination weights. The update rule was modified in order to deal with flat regions that often appear in ranking functions. The new algorithm is shown empirically to perform close to optimum for two base rankers, and scale well if the number of models is increased by homogeneous base rankers or varied ones. We observed that by applying the RFDSA+ combination algorithm a considerable improvement in ranking performance can be obtained over the base rankers.

REFERENCES

[1] Jacob Abernethy, Kevin Canini, John Langford, and Alex Simma. 2007. Online collaborative filtering. *University of California at Berkeley, Tech. Rep* (2007).

[2] Alekh Agarwal, Ofer Dekel, and Lin Xiao. 2010. Optimal Algorithms for Online Convex Optimization with Multi-Point Bandit Feedback.. In *COLT*. Citeseer, 28–40.

[3] Marie Al-Ghossein, Pierre-Alexandre Murena, Talel Abdessalem, Anthony Barré, and Antoine Cornuéjols. 2018. Adaptive collaborative topic modeling for online recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 338–346.

[4] James Bennett, Stan Lanning, et al. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, Vol. 2007. New York, NY, USA., 35.

[5] Robin Burke. 2010. Evaluating the dynamic properties of recommendation algorithms. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 225–228.

[6] Róbert Busa-Fekete, Balázs Kégl, Tamás Éltető, and György Szarvas. 2011. Ranking by calibrated AdaBoost. In *Proceedings of the Learning to Rank Challenge*. 37–48.

[7] Nicolo Cesa-Bianchi and Gábor Lugosi. 2006. *Prediction, learning, and games*. Cambridge university press.

[8] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. 2004. Learning with drift detection. In *Brazilian symposium on artificial intelligence*. Springer, 286–295.

[9] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. 2009. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 329–338.

[10] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets.. In *ICDM*, Vol. 8. Citeseer, 263–272.

[11] Christian Igel and Michael Hüsken. 2000. Improving the Rprop Learning Algorithm. In *Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000)*, H. Bothe and R. Rojas (Eds.). ICSC Academic Press, 115–121. citeseer.ist.psu.edu/igel00improving.html

[12] Kalervo Järvelin and Jaana Kekäläinen. 2000. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 41–48.

[13] Michael Jugovac, Dietmar Jannach, and Mozhgan Karimi. 2018. Streamingrec: a framework for benchmarking stream-based news recommenders. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 269–273.

[14] Robert D Kleinberg. 2005. Nearly tight bounds for the continuum-armed bandit problem. In *Advances in Neural Information Processing Systems*. 697–704.

[15] Levente Kocsis and Csaba Szepesvári. 2006. Universal parameter optimisation in games based on SPSA. *Machine learning* 63, 3 (2006), 249–286.

[16] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–434.

[17] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[18] Neal Lathia, Stephen Hailes, and Licia Capra. 2009. Temporal collaborative filtering with adaptive neighbourhoods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 796–797.

[19] Odalric-Ambrym Maillard and Rémi Munos. 2010. Online learning in adversarial lipschitz environments. *Machine Learning and Knowledge Discovery in Databases* (2010), 305–320.

[20] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 43–52.

[21] Róbert Pálovics and András A Benczúr. 2015. Temporal influence over the Last.fm social network. *Social Network Analysis and Mining* 5, 1 (2015), 4.

[22] Róbert Pálovics, András A Benczúr, Levente Kocsis, Tamás Kiss, and Erzsébet Frigó. 2014. Exploiting temporal influence in online recommendation. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 273–280.

[23] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 502–511.

[24] I Pilászy, A Serény, G Dózsa, B Hidasi, A Sári, and J Gub. 2015. Neighbor methods vs. matrix factorizationcase studies of real-life recommendations. In *LSRS Workshop at ACM RecSys*.

[25] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*. ACM, 784–791.

[26] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. ACM, 285–295.

[27] J. C. Spall. 1992. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Trans. Automat. Control* 37 (1992), 332–341.

[28] Andreas Töscher, Michael Jahrer, and Robert M Bell. 2009. The bigchaos solution to the netflix grand prize. *Netflix prize documentation* (2009), 1–52.

[29] João Vinagre, Alípio Mário Jorge, and João Gama. 2014. Evaluation of recommender systems in streaming environments. In *Workshop on 'Recommender Systems Evaluation: Dimensions and Design' (REDD 2014), held in conjunction with RecSys 2014*.

[30] Yisong Yue and Thorsten Joachims. 2009. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 1201–1208.

[31] Daniel Zoller, Stephan Doerfel, Christian Pölitz, and Andreas Hotho. 2017. Leveraging User-Interactions for Time-Aware Tag Recommendations.. In *RecTemp@ RecSys*. 9–15.