User Modeling and User-Adapted Interaction (2022) 32:649–683
https://doi.org/10.1007/s11257-021-09306-7

Online convex combination of ranking models

Erzsébet Frigó[1] · **Levente Kocsis**[1]

Received: 5 March 2021 / Accepted in revised form: 24 September 2021 / Published online: 6 November 2021

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. First, we propose a stochastic optimization algorithm that uses finite differences. Our new algorithm achieves close to optimal 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. The proposed algorithm is the first effective stochastic optimization method for combining ranked recommendation lists by online machine learning. Secondly, we propose an exponentially weighted algorithm based on a grid over the space of combination weights. We show that the algorithm has near-optimal worst-case performance bound. The bound provides the first theoretical guarantee for non-convex bandits using limited number of evaluations under very general conditions.

Keywords Ranking combination · Online learning · Black-box optimization

1 Introduction

Recommender systems are ubiquitous in our online existence. A large number of algorithms have been developed over the years, some of them are specific to particular domains, others are more general. Selecting the right algorithm for a particular domain is not an easy task. Instead of selecting a single algorithm, recommender systems often rely on an ensemble of base ranking algorithms. The influence of combination algorithms was highlighted for batch rating prediction in the Netflix Prize competition (Balcan et al. 2007), when the approach of Töscher et al. (2009)

✉ Levente Kocsis
kocsis@sztaki.hu

Erzsébet Frigó
frigob@sztaki.hu

[1] Institute for Computer Science and Control (SZTAKI), Eötvös Loránd Research Network (ELKH), Kende u. 13–17, Budapest, Hungary

was essential for the winning entry. Beyond competitions, ensembles are widely used in industrial applications as well (Amatriain and Agarwal 2016).

A milestone in the research of recommendation algorithms, the Netflix Prize 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.

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 leads to satisfying results. In summary, we consider online convex combination algorithms under implicit feedback.

From the machine learning point of view, the main difficulty of combining ranked recommendation lists is that the typical ranking measures, such as NDCG (Järvelin and Kekäläinen 2000), are not continuous, making their optimization a difficult task. Optimizing non-continuous functions is handled most often by using a continuous surrogate function, explicitly or implicitly. The optimization then proceeds by using some form of gradient descent with respect to the surrogate function. An example for this approach is the algorithm proposed in Pálovics et al. (2014) for online ranking combination. A potential drawback in this case is that optimal weight vector for the surrogate function can be very different from the one that is optimal for the original ranking measure.

Black-box optimization algorithms (Conn et al. 2009) do not need the compact form of the gradient and can optimize a function directly. These algorithms have difficulty in optimizing functions with a large number of parameters (as it would be the case for a factor model with millions of parameters), but they can provide an effective alternative, when the number of parameters is moderate (as in the case of convex combination). In this paper, we propose two black-box optimization strategies for non-continuous ranking measures.

The first strategy is a local optimization algorithm that uses finite differences to approximate the gradient. We build on the resilient simultaneous perturbation stochastic approximation (RSPSA) algorithm (Kocsis and Szepesvári 2006), 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. 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, and modify the update rule to

deal with flat regions. Our proposed method, Resilient Finite Difference Stochastic Approximation (RFDSA+), is shown to be the first effective stochastic optimization method for combining ranked lists. We show empirically that RFDSA+ achieves near optimal performance when two base rankers are combined, and scales well with the number of base rankers.

While the effectiveness of the RFDSA+ algorithm on real-word data sets is encouraging, it has no theoretical guarantees. Moreover, in the presence of multiple optima, the algorithm may converge to an inferior local optimum. The second strategy deals with these two issues. It relies on exponentially weighted algorithms (EWA) (Cesa-Bianchi and Lugosi 2006) that explore the space of combination weights globally. EWA was shown to be close to optimal for Lipschitz-continuous environments (Maillard and Munos 2010). We will show that EWA is able to optimize ranking combination as well, under certain assumption (see Property 1). However, the number of combinations that needs to be evaluated grows exponentially with the number of base rankers. For the practical case of limited number of evaluations, we propose a new algorithm, limited advice on grids (LAG). We show that LAG has sub-linear regret (i.e. loss of performance compared to an optimal static combination weight). We note that we provide here the first theoretical guarantee under multiple, but limited number of evaluations for non-convex functions on continuous domains. The regret bound is proved to be log-optimal. We show on synthetic data that LAG can deal well with multiple optima. However, on real-word data sets that appear to be unimodal, RFDSA+ performs significantly better.

The article is organized as follows: after discussing the related research in Sect. 2, we formalize our framework in Sect. 3. The proposed RFDSA+ algorithm is described in Sect. 4. Exponentially weighted algorithms are discussed in Sect. 5, including the theoretical guarantees for LAG. Empirical evaluation highlighting the strengths of RFDSA+ and LAG is provided in Sect. 6. Some conclusions and discussion of future research close the paper in Sect. 7.

2 Related research

Research on incremental recommender algorithms with online or prequential evaluation (Gama et al. 2009) scenario has gained popularity in recent years. There are several papers that use prequential evaluation (Al-Ghossein et al. 2018; Jugovac et al. 2018; Zoller et al. 2017; Burke 2010; Pálovics et al. 2014; Pálovics and Benczúr 2015); however, only (Pálovics et al. 2014) considers the issue of combining multiple base rankers. The latter will be discussed in more detail in Sect. 6.1.4, and evaluated empirically in Sect. 6.

Ranking combination has received considerable attention during the Netflix Prize competition, when the approach of Töscher et al. (2009) 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 described in Busa-Fekete et al. (2011). The authors use the cumulative loss of each base algorithm weighted exponentially 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 Sect. 6.1.2, and evaluate it (for implicit feedback problems) empirically in Sect. 6.3.

Model combination methods have been studied extensively for classification as well (Kuncheva 2014). In the case of batch evaluation, complex combination models are often used. Conversely, a survey of more than sixty ensemble learning algorithms (Gomes et al. 2017) showed that flat structures (such as convex combination) are more frequent for streaming data. For classification, ensembles of simple homogeneous models with dynamic cardinality are more frequent. There are heterogeneous ensembles as well, in which case the cardinality is fixed (see Gomes et al. 2017, and the references therein). Recommender systems, on the other hand, mostly rely on ensembles of complex heterogeneous ranking models, and the cardinality is fixed.

In an online setting, ranking combination was proposed by Radlinski et al. (2008), Yue and Joachims (2009) 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. Agarwal et al. (2010) considered a two-point approximation of the gradient for convex functions. The ranking measures are not convex, nevertheless, the algorithm is similar to SPSA (Spall 1992) that have been applied to optimizing non-convex functions as well. We will discuss the algorithm in Sect. 4.

An exponentially weighted algorithm was applied to optimize (non-convex) Lipschitz-continuous functions (Maillard and Munos 2010) with an $O(\sqrt{T})$ bound on the regret in full-information setting, where T is the length of the episode. While we rely on a grid structure over the weight space for our exponentially weighted algorithms, it is possible to use interval trees as suggested in Cohen-Addad and Kanade (2017). The authors proposed full-information algorithm for online optimization of (smoothed) piecewise constant functions that is close to the optimization of ranking measures. The algorithm discussed in the 1-dimensional case is computationally efficient if the discontinuities in the function can be easily determined. The authors propose some extensions for higher dimensions for the case when the separation surfaces are orthogonal. The orthogonality condition is not valid for the convex combination of (ranking) models, however. A similar class of functions was discussed in Balcan et al. (2019). Full information setting implies evaluating a prohibitively large number of points, when the number of base rankers is slightly larger. There are bandit variants that evaluate only one point per iteration, however, they scale badly on error. Using a grid-based variant, similar to ours, (Kleinberg 2005) proves a regret bound of $O(T^{(d+1)/(d+2)})$, where d is the dimensionality of the problem. For stationary/stochastic continuous bandits hierarchical partitioning seems to offer better empirical performance (Bubeck et al. 2011; Grill et al. 2015), but the theoretical guarantees remain the same in the worst case. Stronger guarantees exist when the function class constrained to be convex (Hazan and Levy 2014; Hazan and Li 2016). We are not aware of any theoretical guarantees for non-convex functions under

multiple, but limited number of evaluations. The exponentially weighted algorithms will be considered in Sect. 5.

We argued in Sect. 1 that given the moderate dimensionality of the convex combination, black-box optimization algorithms can be good candidates to optimize combination weights of ranking models. There are certain properties of the environment that affect adversely the applicability of these algorithms, including the non-continuity of the ranking measures, the non-stationarity of the online scenario, and a fair amount noise in the evaluation. Stochastic approximation (SA) algorithms can deal well with the noise and the non-stationarity. We discuss one of the best known SA algorithms, Simultaneous Perturbation Stochastic Approximation (SPSA) in Sect. 4, and we evaluate it empirically in Sect. 6. The problem for algorithms such as SPSA is that ranking functions such as NDCG that are not continuous. In most games, the reward is also non-continuous, for instance, 1/0 for win/loss, or a discrete number of points (or money) that can be won in a card game. The algorithm RSPSA (Kocsis and Szepesvári 2006) was proposed for (offline) optimization of some parameters of a poker playing program. Our proposed algorithm, RFDSA+, builds on the idea of RSPSA, but identifies the flat area for each weight by switching to estimation by finite differences. The problem of flat regions in the ranking measures is discussed in more depth in Sect. 4, and illustrated empirically in Sect. 6.2. There are a large number of alternative black-box optimization algorithms (see e.g., Conn et al. 2009; Larson et al. 2019). Of these, the covariance matrix adaptation evolution strategy (CMA-ES) was proven to be effective on a wide range of test benches with larger dimensional spaces (Hansen et al. 2010). In Sect. 6, we evaluate empirically a variant (Igel et al. 2006), which was shown to work well in dynamic environments (Au and Leung 2012).

3 Problem setup

In our task, we want to combine the ranked lists of multiple base recommender algorithms by monitoring the performance of the base recommenders and adaptively changing the weights on the fly. As soon as the base algorithms give a prediction, we have to apply and potentially re-learn the combination weight by a data streaming, online machine learning method.

The difficulty of evaluating streaming recommenders was first mentioned in Lathia et al. (2009), although the authors evaluated models by offline training and testing split. Ideas for online evaluation metrics appeared first in Pálovics and Benczúr (2015), Vinagre et al. (2014), Pálovics et al. (2014). In the following, we use prequential evaluation (Gama et al. 2009), which has grown in popularity in recent years. 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. 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.

For batch and prequential evaluation, we have to assume that the preference of the user is independent of the recommendation of the system. This assumption is necessary for testing on a data collected independently, usually before the recommender algorithms of the experiment are developed. In contrast, papers using the cascade model (Craswell et al. 2008) in a bandit scenario (see e.g., Kveton et al. 2015) assume that the choice of the user is strongly determined by the recommendation made. Offline testing is more difficult in this case, since the collected data needs to cover all possible recommendations. For an online retailer or hash tag recommendation (see Sect. 6.3.1), the user can select items regardless of the recommendation. In these domains, batch or prequential evaluations are reasonable. Conversely, cascade models could be more appropriate for search engines.

The protocol for prequential evaluation with implicit feedback is as follows. 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, and the recommender collects reward r_t.
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 $t = 1, \dots, T$, 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 $\boldsymbol{\theta}_t = (\theta_{t1}, \dots, \theta_{tN})$. 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 $\mathcal{C}$ by $r_t(\mathcal{C})$. 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 $R_t(\mathcal{C})$ 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 (Järvelin and Kekäläinen 2000). In prequential evaluation, we assume the scenario that there is only one item with nonzero 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. Another ranking measure used in the experiments is the mean reciprocal rank (*MRR* Voorhees and Tice 1999). The MRR@K of a permutation π_t is

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

4 Resilient finite difference stochastic approximation (RFDSA+)

In this section, we present our main algorithm, RFDSA+. First, we provide a motivation for the construction of the algorithm, followed by its full description.

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, the ranking function starts to "smooth out". The gradient still cannot be computed in a closed form, but it can be approximated by finite differences. For online optimization of convex functions, (Agarwal et al. 2010) 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) (Spall 1992). The approximated gradient g_{ti} for weight component θ_{ti} is given by

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

where c_t is an appropriately chosen, decreasing sequence,

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

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

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

The RSPSA algorithm was proposed in Kocsis and Szepesvári (2006) 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) (Igel and Hüsken 2000) update rule.

In RPROP, we assign a distinct step size to each weight. The gradient update of the weights depends only on the sign of the gradient, and the step size determines how much the weight changes. The step sizes are also adjusted during the updating process. Considering each coordinate separately, the step size is updated based on

the changes of the sign of the corresponding coordinate of the gradient. If the direction of the gradient changes, i.e., the sign switches, then the step size is decreased. Otherwise, it is increased.

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.

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 increases the update steps on a large plateau, taking larger steps in the directions of the gradient. Ranking measures as function of a combination weight consist of constant intervals. However, if the perturbation is sufficiently large, the averaged gradient estimate will be nonzero. Therefore, 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.

We observed that in RSPSA, the estimated direction changes often in a flat area. In accordance to the RPROP rule, the change in the direction results in a decrease of the step size, which is contrary to the desired behavior outlined above. 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 sign of the coordinates of the estimated gradient corresponding to the 'flat' coordinates becomes an unbiased Bernoulli variable. The reason behind this behavior is that even if the ranking function is completely flat with respect to coordinate i, the numerator of g_{ti} will still be nonzero because of the non-flat coordinates. However, the value of the numerator will be independent of the randomly chosen direction of Δ_{ti}, that still appears in the denominator, 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. When the gradient is estimated by finite differences, only one weight is perturbed at a time, the other weights remain unchanged. Therefore, 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 (lines 7–9), and handling the flat regions (lines 23–24). The RPROP update is given by lines 12–29.

```
1  for i = 1 to N do
2  |  θ_i ← 1/N; g_i ← 0
3  |  s_i ← 0; δ_i ← δ_0
4  end
5  for t = 1 to T do
6  |  for i = 1 to N do
7  |  |  sample Δ_i = ±1
8  |  |  θ^+ ← θ; θ_i^+ ← θ_i + 2δ_i Δ_i
9  |  |  g_i ← g_i + (r_t(θ^+) − r_t(θ))/(2δ_i Δ_i)
10 |  end
11 |  if t mod B = 0 then
12 |  |  for i = 1 to N do
13 |  |  |  h ← g_i s_i
14 |  |  |  if h > 0 then
15 |  |  |  |  δ_i ← η^+ δ_i
16 |  |  |  |  s_i ← sign(g_i)δ_i
17 |  |  |  else if h < 0 then
18 |  |  |  |  δ_i ← η^− δ_i
19 |  |  |  |  s_i ← 0
20 |  |  |  else
21 |  |  |  |  s_i ← sign(g_i)δ_i
22 |  |  |  end
23 |  |  |  if g_i = 0 then
24 |  |  |  |  δ_i ← η^+ δ_i
25 |  |  |  else
26 |  |  |  |  θ_i ← θ_i + s_i
27 |  |  |  end
28 |  |  |  g_i ← 0
29 |  |  end
30 |  end
31 end
```

Algorithm 1: RFDSA+

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 noisy functions, typical values are $\eta^+ = 1.1$ and $\eta^- = 0.85$ (Kocsis and Szepesvári 2006). 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 1000 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 partial derivatives are collected in the variables g_i.

The RFDSA+ algorithm starts with an initialization phase in lines 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 8). The coupling factor 2 is standard for RSPSA (Kocsis and Szepesvári 2006), 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. The partial derivatives g_i are updated in line 9, 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 (Igel and Hüsken 2000) in lines 12–29, independently for each component i. The auxiliary variable h detects the direction change of the partial derivative. If there is no change (lines 14–16), 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 (lines 17–19), then the step size is decreased, and the weight is left unchanged. The weight will be updated after the next mini batch (line 21). The key modification that deals with flat regions in the partial derivatives is shown in lines 23–24. 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 weights are modified in line 26. As discussed previously, s_i depends only on the sign, and not the size of the estimated gradient.

We note that while that differences between the RSPSA and the RFDSA+ algorithms are seemingly small, the ability to handle better the flat areas proves to be significant in the empirical performance, as discussed in Sect. 6.

5 Grid based exponentially weighted algorithms

The algorithms presented in this section explore the weight space globally, without relying on the existence of a gradient of the reward function. We define a grid structure over the weight space, and we explore the grid points using some variant of the exponentially weighted forecaster (Cesa-Bianchi and Lugosi 2006).

In Sect. 5.1, we consider an algorithm for the case when there is no limit on the number of evaluations per round. Since a very large number of evaluations is not feasible in a practical application, we propose an exponentially weighted algorithm with limited number of evaluations in Sect. 5.2. The theoretical properties of the exponentially weighted algorithms are analyzed in Sect. 5.3. As an important theoretical contribution, we show that the proposed algorithm for the limited case has near-optimal worst-case performance.

5.1 ExpW

Exponentially weighted algorithm for Lipschitz-continuous functions in the full-information case was proposed in e.g., (Maillard and Munos 2010). They propose a simple variant that relies on a uniform grid over parameter space, and use the exponentially

weighted forecaster to select the next parameter vector, and update the estimated value for each grid point.

While our ranking measures are non-continuous, we will provide a weaker Lipschitz-like condition in Sect. 5.3, Eq. 5, that is sufficient for the good performance of the algorithm. The condition is validated empirically in Sect. 6.3.3.

In the algorithm denoted ExpW, we choose a subset Q from the parameter space Θ. We consider here a more general definition of the set Q, but in practice, we will use a uniform grid, described in more detail later. In each round t, we select a weight vector $\mathbf{q} \in Q$, according to a probability distribution $p_t(\mathbf{q})$, defined as

$$p_t(\mathbf{q}) = \frac{\exp\left(\eta \sum_{\tau=1}^{t-1} r_\tau(\mathbf{q})\right)}{\sum_{\mathbf{q}' \in Q} \exp\left(\eta \sum_{\tau=1}^{t-1} r_\tau(\mathbf{q}')\right)}, \tag{1}$$

where η, called the learning rate, is a parameter specified in Sect. 5.3. After the user feedback, we evaluate the ranking performance r_t for all weight vectors $\mathbf{q}$.

We will show that this algorithm has close to optimal performance guarantees, but the number of evaluations per round is high, especially in the presence of several base rankers.

5.2 LAG

In order to decrease the number of points that need to be evaluated in each round, we can apply the limited advice exponentially weighted forecaster (Seldin et al. 2014) on a similar set Q. We denote this algorithm by LAG. We limit the number of evaluated weight vectors in each round by M. The point of Q to play is chosen using a random distribution, similarly to ExpW. However, as not all weight combinations of Q are evaluated, we need to estimate the reward values. As in Seldin et al. (2014), the estimator $\hat{r}_t(\mathbf{q})$ of $r_t(\mathbf{q})$ is defined by

$$\hat{r}_t(\mathbf{q}) = \frac{r_t(\mathbf{q})}{p_t(\mathbf{q}) + (1 - p_t(\mathbf{q}))\frac{M-1}{|Q|-1}} \mathbb{I}_t(\mathbf{q}),$$

where $\mathbb{I}_t(\mathbf{q})$ is the indicator of $\mathbf{q}$ being evaluated in round t.

We select the weight vector $\mathbf{q} \in Q$ in round t by sampling from the probability distribution p_t:

$$p_t(\mathbf{q}) = \frac{\exp\left(\eta_t \sum_{\tau=1}^{t-1} \hat{r}_t(\mathbf{q})\right)}{\sum_{\mathbf{q}' \in Q} \exp\left(-\eta_t \sum_{\tau=1}^{t-1} \hat{r}_t(\mathbf{q}')\right)}.$$

We evaluate the selected vector and sample uniformly further $M - 1$ weight vectors without replacement from Q that are also evaluated. We note that the estimator $\hat{r}_t(\mathbf{q})$ is unbiased, however, it has high variance when M is small. Moreover, when all vectors in Q are evaluated ($M = |Q|$), the LAG algorithm is identical to ExpW.

The pseudocode of LAG is provided in Algorithm 2. In the pseudocode, we use the variable $w(\mathbf{q})$ to store $\exp(\eta_t \sum_{\tau=1}^{t-1} \hat{r}_t(\mathbf{q}))$. The probability of being evaluated is denoted by $p'(\mathbf{q})$. The algorithm has three inputs: the limit on the number of evaluations per round, M, which is a design parameter, the set Q and the learning rate η, which will be chosen in the next subsection.

Input: Q, M, η
for $\mathbf{q} \in Q$ **do**
  $w(\mathbf{q}) \leftarrow 1$;
end
for $t = 1$ **to** T **do**
  Draw $\mathbf{q}_t$ according to $p(\mathbf{q}) = w(\mathbf{q}) / \sum_{\mathbf{q}' \in Q} w(\mathbf{q}')$.
  Construct recommendation using $\mathbf{q}_t$, and observe user feedback.
  Sample uniformly further $M - 1$ weight vectors without replacement from Q, and evaluate them.
  for *all evaluated* $\mathbf{q}$ **do**
    $p'(\mathbf{q}) \leftarrow p(\mathbf{q}) + (1 - p(\mathbf{q}))(M - 1)/(|Q| - 1)$
    $w(\mathbf{q}) \leftarrow w(\mathbf{q}) \exp(\eta r_t(\mathbf{q}) / p'(\mathbf{q}))$
  end
end

Algorithm 2: LAG

5.3 Theoretical properties

In online learning, performance bounds are derived for the regret of the algorithm (Cesa-Bianchi and Lugosi 2006). Informally, the regret is the performance loss from not knowing the best weight vector. Formally, it is the difference between the reward of the best constant weight vector $\boldsymbol{\theta} \in \Theta$ and the reward collected by the algorithm:

$$\mathfrak{R}_T(\mathcal{C}) = \max_{\boldsymbol{\theta} \in \Theta} R_T(\boldsymbol{\theta}) - R_T(\mathcal{C}). \tag{2}$$

Comparing the reward of the combination algorithm to the best constant gridpoint mirrors the expectation that the performance of the algorithm depends on the environment. If all weight vectors pay a low reward, then we do not expect a high reward from the combination algorithm either. If a low regret can be guaranteed, that means the performance of the algorithm is always close to the optimal convex combination. Regret bounds deal with worst-case regret, since they bound the maximum expected regret over all possible environments.

In the following, we first provide an upper bound for ExpW. This will require an assumption on the cumulative ranking measure. We show that this bound is log-optimal. Then, we provide regret upper and lower bounds for LAG, under the same assumption. The proofs are provided in the "Appendix".

5.3.1 Full information

In case of ExpW, if an appropriately large Q is chosen, and the cumulative reward function $R_T(\boldsymbol{\theta})$ is sufficiently smooth, then the expected regret is guaranteed to be low. The following proposition formalizes this statement.

Proposition 1 *Let $Q \subset \Theta$ be a finite set such that for some constant $c \in \mathbb{R}$*

$$\mathrm{E}\left[\max_{\boldsymbol{\theta}\in\Theta} R_T(\boldsymbol{\theta}) - \max_{\mathbf{q}\in Q} R_T(\mathbf{q})\right] \leq c\sqrt{T}. \tag{3}$$

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

$$\mathrm{E}\left[\max_{\boldsymbol{\theta}\in\Theta} R_T(\boldsymbol{\theta}) - R_T(ExpW)\right] \leq \sqrt{\frac{T}{2}\ln|Q|} + c\sqrt{T} \tag{4}$$

that can be achieved by setting $\eta = \sqrt{2\frac{\ln|Q|}{T}}$.

The goal of the condition in inequality (3) is to ensure that the optimum is not on an isolated spike of the parameter space, making it practically impossible to be found by any algorithm. For a sufficiently large T, while function $R_T(\boldsymbol{\theta})$ still consists of small steps, it is fairly smooth in practice, as observed in Sect. 6.3.3. The following property is a generalization of the Lipschitz-property for such functions that lets us find appropriate subsets.

Property 1 (Lipschitz-like) *Define a d-dimensional κ-grid as $\{0 \leq i/\kappa \leq 1 \mid i \in \mathbb{Z}_{0+}\}^d$. The function R_T is Lipschitz-like on $\Phi = [0,1]^d$ if there is a $\lambda \in \mathbb{R}$ such that for all $\kappa \in \mathbb{R}_{0+}$,*

$$\mathrm{E}[\max_{\boldsymbol{\theta}\in\Phi} R_T(\boldsymbol{\theta}) - \max_{\mathbf{q}\in\mathcal{G}_\kappa} R_T(\mathbf{q})] \leq \frac{\lambda T\sqrt{d}}{\kappa}, \tag{5}$$

where $\mathcal{G}_\kappa$ is the κ-grid.

The parameter κ of the grid describes the density of the grid. The distance of closest points in a κ-grid is $1/\kappa$. There are $\lfloor \kappa + 1 \rfloor$ points along each edge of the parameter space, and the grid consists of $\lfloor \kappa + 1 \rfloor^d$ points.

Note that without loss of generality, in the case of ranking combination, we can assume that the sum of the weights is 1. Applying this restriction, we can decrease the dimensionality of the problem by 1, thus in our case the dimensionality is $d = N - 1$. For instance, in the case of two base rankers, the weight of the first ranker will be $\theta_1 \in [0,1]$, and the second will be $\theta_2 = 1 - \theta_1$.

If R_T is Lipschitz-like, then a $\sqrt{T}$-grid satisfies the condition of Proposition 1 for $c = \lambda\sqrt{d}$. As a corollary, applying ExpW on this grid results in an $\tilde{O}(\sqrt{T})$ regret bound. This is formalized in the next proposition.

Proposition 2 *If R_T is Lipschitz-like with the constant λ, then the regret of the exponentially weighted forecaster applied on a $\sqrt{T}$-grid is bounded by*

$$\mathrm{E}\left[\max_{\theta\in\Theta} R_T(\theta) - R_T(ExpW)\right] \leq \left(\frac{1}{\sqrt{2}} + \lambda\right)\sqrt{Td\ln T}. \tag{6}$$

To show that under these conditions no stronger algorithms may exist, we prove a matching lower bound.

Proposition 3 *For the combination problem on Θ, the expected regret is lower bounded by*

$$\inf \sup \mathrm{E}[\mathfrak{R}_T] \geq 0.03\sqrt{T}, \tag{7}$$

where the infimum is over all playing strategies and the supremum is over all oblivious environments satisfying Property 1 for $\lambda = 1$.

If we compare the lower bound to the upper bound of ExpW for functions that satisfy Property 1, we obtain that the performance of ExpW is optimal up to logarithmic terms. Since functions that satisfy Property 1 also satisfy the condition of Eq. 3 by taking the grid as the subset Q, the proposition provides a lower bound for Proposition 1 as well.

For two base rankers, the parameter space can be represented by 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 Fig. 5). 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.

5.3.2 Limited number of evaluations

We have seen that the ExpW algorithm has strong regret guarantees, but at the price of high number of evaluations. In practice, the number of evaluations are limited for computational reasons. We now turn our attention to the case when the number of evaluations are limited, upper bounding the regret of the LAG algorithm. Decreasing the number of points allowed to be evaluated provides a trade-off between complexity and efficiency, as stated by the following proposition.

Proposition 4 *Let $Q \subset \Theta$ be a finite set such that for some constants $c, c' \in \mathbb{R}$*

$$\mathrm{E}\left[\max_{\theta\in\Theta} R_T(\theta) - \max_{\mathbf{q}\in Q} R_T(\mathbf{q})\right] \leq c\frac{T}{(TM)^{\frac{1}{d+2}}}, \tag{8}$$

and $|Q| < c'(TM)^{\frac{d}{d+2}}$.

Then the expected regret of the limited advice exponentially weighted forecaster applied on Q for any $M \leq |Q|$ is bounded by

$$\mathrm{E}\left[\max_{\boldsymbol{\theta}\in\Theta} R_T(\boldsymbol{\theta}) - R_T(\mathrm{LAG})\right] \leq \left(c + 2\sqrt{c' \ln |Q|}\right) \frac{T^{\frac{d+1}{d+2}}}{M^{\frac{1}{d+2}}} \tag{9}$$

that can be achieved by setting $\eta_t = \sqrt{\frac{M \ln |Q|}{t|Q|}}$.

The bound clearly shows the trade-off between the number of points evaluated and the regret. Assuming the Lipschitz-like property, and setting $c = \lambda\sqrt{d}$, $c' = 2$ in Proposition 4, we obtain the first main theoretical result.

Theorem 1 *If R_T is Lipschitz-like with the constant λ, then the regret of the limited advice exponentially weighted forecaster applied on a $(TM)^{\frac{1}{d+2}}$-grid is bounded by*

$$\mathrm{E}\left[\max_{\boldsymbol{\theta}\in\Theta} R_T(\boldsymbol{\theta}) - R_T(\mathrm{LAG})\right] \leq \left(\lambda\sqrt{d} + 2\sqrt{2\ln(2TM)}\right) \frac{T^{\frac{d+1}{d+2}}}{M^{\frac{1}{d+2}}} \tag{10}$$

We note that if all points of Q can be evaluated, then the bound with a $\sqrt{T}$ -grid is similar to the bound on ExpW stated in Proposition 2.

To show that LAG is optimal, we need to show that in an environment that satisfies our conditions, no algorithm can perform better. This is stated in the following theorem.

Theorem 2 *For the combination problem on $\Phi = [0,1]^d$ with $M \leq T^{\phi}$ observations per round and $\frac{3}{4}\log T \leq T^{\frac{d-2\phi}{d+2}}$, the expected regret is lower bounded by*

$$\inf \sup \mathrm{E}[\mathfrak{R}_T] \geq 0.03 \frac{T^{(d+1)/(d+2)}}{M^{1/(d+2)}}, \tag{11}$$

where the infimum is over all playing strategies and the supremum is over all oblivious environments satisfying Property 1 for $\lambda = 3$.

Note that for any fixed $\phi < \frac{d}{2}$, the condition of the theorem is satisfied for large enough T. The case of $\phi \geq \frac{d}{2}$ is not relevant because the full information lower bound in Proposition 3 provides a stronger bound. This is not surprising since ExpW needs to evaluate $T^{\frac{d}{2}}$ weight combinations only for optimal performance.

We observe that the worst-case regret of LAG is much higher than that of ExpW. Nevertheless, it is also clear that the bound for limited number of evaluations is improvable by at most logarithmic factors. We note that Theorem 1 provides the first sub-linear regret bound for non-convex continuous domains under limited number of evaluations. The conditions of this bound are also very general and non-restrictive.

6 Experiments

In this section, we evaluate empirically the proposed combination algorithms. First, we describe the baseline combination algorithms. Then, we illustrate the strengths of our algorithms using synthetic data in Sect. 6.2. Experiments on real data are provided in Sect. 6.3.

6.1 Baselines

For baselines, we use two exponentially weighted algorithms, four stochastic approximation variants similar to RFDSA+, a stochastic gradient algorithm that uses a surrogate function, and the state-of-the-art black-box optimization algorithm CMA-ES, described in the following.

6.1.1 ExpA

The simplest choice to deal with multiple base rankers is to select the base ranker that appears the best. In a non-stationary environment this can be best achieved by the exponentially weighted forecaster. Accordingly, the combination algorithm, denoted by ExpA, selects base ranker $\mathcal{A}_i$ in round t with probability

$$p_{ti} = \frac{e^{-\eta_t \sum_{\tau=1}^{t-1} r_{\tau i}}}{\sum_{j=1}^{N} e^{-\eta_t \sum_{\tau=1}^{t-1} r_{\tau j}}}. \tag{12}$$

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 (Cesa-Bianchi and Lugosi 2006).

6.1.2 ExpAW

In Busa-Fekete et al. (2011), 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

$$\theta_{ti} = \frac{e^{-\eta_t \sum_{\tau=1}^{t-1} r_{\tau i}}}{\sum_{j=1}^{N} e^{-\eta_t \sum_{\tau=1}^{t-1} r_{\tau j}}}. \tag{13}$$

It is easy to see that the reward of a base ranker does not change if the scores of the 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.

6.1.3 Stochastic approximation algorithms: SPSA, RSPSA, RSPSA+ and RFDSA

The SPSA and RSPSA algorithms were described in Sect. 4. RFDSA denotes the algorithm that uses finite differences for the estimation of the gradient, and RPROP for updating the combination vector, but it is not increasing the step size in flat regions. RSPSA+ extends RSPSA with the step size increase for flat regions, but the partial derivatives are estimated using simultaneous perturbation (as in the original RSPSA formulation). These two hybrid choices are designed to elicit the individual effect of the two elements when switching from RSPSA to RFDSA+.

6.1.4 SGD

While we advocate in this paper a direct optimization of the ranking measure, most classification or ranking combination algorithms optimize a surrogate of the evaluation measure by stochastic gradient descent. The surrogate function can appear explicitly or implicitly in the formulation of the algorithm. For ranking it is usual to consider the current item as a positive instance, and some randomly sampled items as negative instances. In this case the target for the positive item is set to 1, and for negative items 0. After seeing a user-item pair, a stochastic gradient step is taken to minimize the mean squared error between the ranking score (x_{ij}) and the target value of the positive item and of the selected negative items. The algorithm was used by Pan et al. (2008) for matrix factorization and by Pálovics et al. (2014) for online combination. In the following, we refer to this algorithm by SGD.

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.

6.1.5 CMAES

Covariance matrix adaptation evolution strategy (CMA-ES) is a black-box optimization algorithm that was proven to be effective on a wide range of test benches with larger dimensional spaces (Hansen et al. 2010). For mutation, CMA-ES samples from a Gaussian distribution that depends on adaptive covariance matrices. The selection strategy can be elitist or non-elitist. In a non-elitist strategy, the current population is replaced by their best offspring. In an elitist strategy, parents are preserved if they are sufficiently good. Since, in our online learning scenario, the goal is not only to find the best weight vector (as it is in an optimization scenario), but also to exploit the current best solution, we feel that the more aggressive, elitist selection is appropriate. We use an elitist selection variant (Igel et al. 2006), which was shown to work well in dynamic environments (Au and Leung 2012). The variant is referred generally as CMA-ES($1 + \lambda$), where 1 denotes the number of parents kept, and λ is the size of the candidate population. We will set $\lambda = N$, which means that the algorithm will use the same number of evaluations as RFDSA+. The fitness of the parent

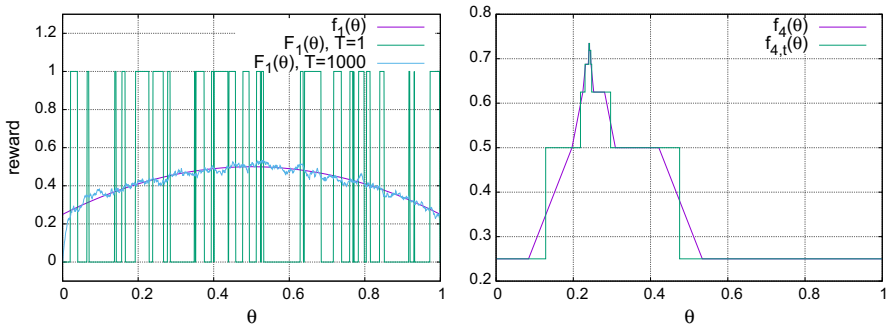

Fig. 1 Synthetic test functions F_1 (left) and F_4 (right)

is reevaluated when the offspring population is evaluated. This is necessary to avoid persisting with an individual that got 'lucky' by having an evaluation at a time when the base rankers had high success rate. The evaluation is performed in batches of 5000 rounds. This seemed empirically a good choice to reduce sufficiently the noise of the evaluation, while still allowing the search to progress. In the following, we will refer to the algorithm by CMAES.

6.2 Synthetic data

In this section, we illustrate the strength and weaknesses of the combination algorithms using synthetic functions. The advantage of these functions is that we can control their shape, eliciting particular characteristics. The optimal weight vectors are known for these functions. Therefore, we show the performance of the algorithms as average cumulative regret, $(\max_{\theta\in\Theta} R_t(\theta) - R_t(\mathcal{C}))/t$.

6.2.1 Test functions

The functions used in this set of experiments do not assume an underlying ranking, but they do preserve certain properties of ranking functions. We note that all combination algorithms discussed, except SGD, rely only on the ranking value of a particular combination vector, and do not use explicitly the item scores assigned by base ranking algorithms. All functions are stationary and stochastic. At each time step t, the random functions F_t are constructed as binary-valued (0,1) segments. The functions are defined on the unit cube $[0, 1]^d$, and if a combination algorithms selects a vector outside of the cube, then the value corresponding to its projection on the cube is returned.

The first function sequence, F_1, relies on a one-dimensional quadratic function $f_1(\theta) = 0.5 - (\theta - 0.5)^2$. At each time step, the function is divided in 100 segments, with the delimiting points chosen uniformly. The value of the segment j is a 0-1 valued Bernoulli variable with the expectation given by $(f_1(\theta_{j_1}) + f_1(\theta_{j_2}))/2$, where θ_{j_1} and θ_{j_2} are the extremities of the segment. In Fig. 1, left we show the function f_1, along with a random instance $F_{1,t}$ and the average of $F_{1,t}$ taken over 1000 time steps.

We observe, that the function is fairly noisy even after the averaging. The expectation of $F1$ is concave, therefore combination algorithms that approximate the gradient (in particular *SPSA*) are expected to perform well.

The second function sequence, F_2, is a d-dimensional version of the previous function. At each time step, each coordinate is divided in 100 intervals, resulting in 100^d cubes. As previously, the value of a cube is a Bernoulli variable with expectation given by $\sum_{i=1}^{d}(f_1(\theta_{i,j1}) + f_1(\theta_{i,j2}))/2d$.

The third function sequence, F_3, relies on a function constructed by the union of two quadratic functions

$$f_3(\theta) = \begin{cases} 0.5 - 0.8(\theta - 0.25)^2 - 1.2 \cdot 0.25^2, & \theta < 0.5 \\ 0.5 - 0.8 \cdot 0.25^2 - 1.2(\theta - 0.75)^2, & \theta \geq 0.5 \end{cases}$$

The function has a lower (local) maximum at $\theta = 0.25$ and a higher (global) maximum at $\theta = 0.75$. At each time step, 100 segments are constructed in a similar way as for F_1. Since the function has two separate maxima, gradient approximation algorithms can get stuck in the local maximum.

The fourth function sequence, F_4, relies on a one-dimensional function specially constructed to have several flat regions. The construction is similar to the function used in the lower bound in Kleinberg (2005). The interval is split in two, with one chosen as the better one and the other the sub-optimal one. The better sub-interval is split recursively. The elevation of the better sub-interval is also halved after a split. The precise shape of the function is shown in Fig. 1, right. The segments are chosen again randomly, but only one discontinuity is chosen per each oblique interval. The expected value of each segment is the value of the constant interval that is included in the segment. A sample 'segmentation' is also shown in the figure as $f_{4,t}(\theta)$. Similarly to previous functions, the value observed for a segment is a Bernoulli 0–1 valued variable with the expected value described above.

The fifth function sequence, F_5, relies on a two-dimensional extension of the previous function. The segments for the first coordinate is constructed as for F_4. 100 segments are chosen for the second coordinate, with extremities chosen uniformly. The expectation for the second coordinate is constant, and it is only present to add noise when simultaneous perturbation is preferred. When the value of the function is queried at a certain time, it will return the same value for queries that have the second coordinate the same and the first coordinate fall on the same flat region. It can result in different values if the second coordinate is different.

The sequence of functions enlisted above are similar to ranking functions in that ranking functions are also made of discrete segments with fixed values (although, not necessarily binary). The stationary condition makes it easier to define the optima, and to keep the desired characteristics of the functions.

6.2.2 Convergence of LAG

In the first set of experiment with synthetic data, we validate empirically the theoretical results from Sect. 5.3 concerning the convergence of the LAG algorithm. We use the d-dimensional quadratic function sequences F_2, with $d = 1, 2, 3$. The regret

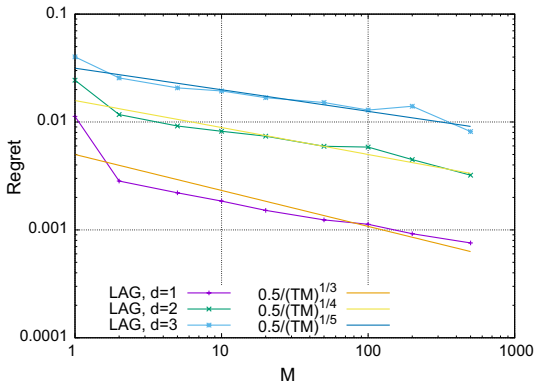

Fig. 2 The average regret of LAG samples per round limited to M on the d-dimensional test function F_2 after $T = 1,000,000$ rounds. Each data point is averaged over 1000 runs

of LAG after $T = 1,000,000$ rounds is plotted in Fig. 2 with varying number of sample points per round (M). We observe that the empirical results mirror fairly well the theoretical bound of $O(1/(TM)^{d+2})$.

The main discrepancy between the theoretical bound and the empirical perform is at $M = 1$. The discrepancy can be understood by comparing the base bandit algorithm used in LAG (Seldin et al. 2014) with $M = 1$ to non-stochastic bandit algorithms with explicit exploration, such as Exp3.P (Auer et al. 2002) or with implicit exploration (Neu 2015). The lack of exploration term in the limited advice algorithm leads to a higher variance of the estimation of the reward, since the selection probabilities (p_{ti}) can be very small. The higher variance is an obstacle for deriving high probability bounds, but not as much for deriving bounds on the expected regret. It also appears to affect negatively the empirical performance, especially when the number of arms (i.e. grid points) are large such as in our case. The problem is mitigated when M is larger, since the selection probability is mixed in the estimation with uniform distribution using a coefficient linear in $M - 1$ (see Algorithm 2, line 10).

6.2.3 Performance of black-box optimization algorithms

The next set of experiments illustrate the performance of black-box optimization algorithms, focusing on the properties of the reward functions such as concavity, multi-modality and flatness.

First, we test algorithms on F_1, which is concave and unimodal, and the results are shown in Fig. 3, left. LAG with a moderate number of samples per round ($M = 10$) is doing well for this function. This is not surprising since it is a one-dimensional function with a small Lipschitz constant near the optimum. SPSA is doing also well for this function, which is natural since for similar algorithms there are known $O(\sqrt{T})$ regret bounds on concave reward functions[1] (Shamir 2017). RSPSA and

[1] In fact, the problem discussed in Shamir (2017) was the minimization of convex functions, which is the reciprocal problem to the maximization of concave reward functions.

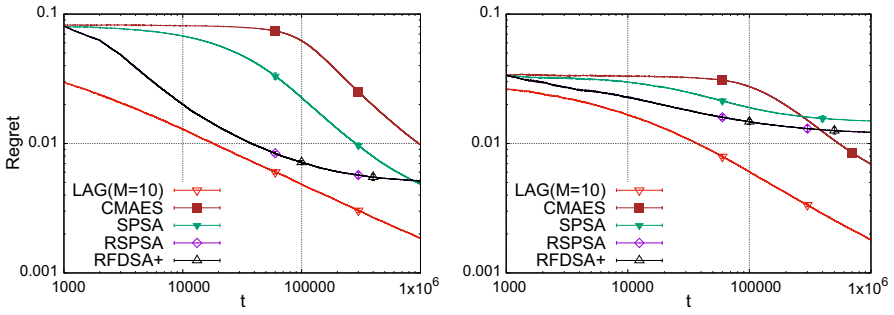

Fig. 3 The time-varying regret of black-box optimization algorithms on the one-dimensional quadratic test functions F_1 (left) and F_3 (right). Each data point is averaged over 1000 runs, and 99% confidence intervals are shown. Note that most of the intervals are very small

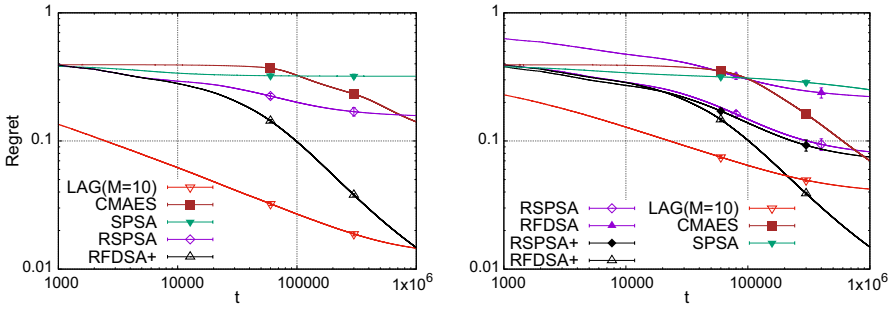

Fig. 4 The time-varying regret of black-box optimization algorithms on the test functions with flat regions F_4 (left) and F_5 (right). Each data point is averaged over 1000 runs, and 99% confidence intervals are shown

RFDSA+ perform identically, and converge faster than SPSA although they have a slightly higher regret after T=1,000,000 rounds. CMAES has the weakest result.

Next, we use F_3, which is multi-modal, and the results are being shown in Fig. 3, right. The Lipschitz constant near the optimum is similar for F_1 and F_2, and the difference between the expected reward at the two maxima is sufficiently large. Consequently, LAG is performing similarly well as in the previous experiment. SPSA and RSPSA/RFDSA+ are much more affected. These algorithms will often converge to the weaker local maximum. Interestingly, CMAES is able to cope better with multiple optima. While CMAES and the SA algorithms are all local search algorithms the candidate selection in CMAES has a longer tail compared to the perturbation size of the SA algorithms. This leads to an advantage for CMAES, since it is more likely to escape local optima.

Finally, we test the effect of flatness extension in RFDSA+. For the 1-dimensional function F_4, we observe in Fig. 4, left that RFDSA+ significantly outperforms RSPSA. RFDSA+ deals with the flatness of the function by switching to finite-difference estimation and increasing the step-size when flatness is detected. The effect

of the two mechanism is tested in Fig. 4, right with the two-dimensional function F_5. We observe that switching to finite-difference estimation only (RFDSA) deteriorates the performance. Dealing with the flatness using simultaneous perturbation (RSPSA+) has some advantage over RSPSA, but a very small one. The two mechanisms combined in RFDSA+ lead to a significant increase in performance. Of the remaining algorithms, SPSA is unable to deal with the non-concavity of the function, especially with the flatness, and performs poorly. CMAES starts slowly, but overtakes slightly RSPSA after nearly 1,000,000 steps. The Lipschitz constant near the optimum of F_4 (and consequently, F_5) is fairly large, therefore, the performance of LAG is not as strong anymore, especially for the two-dimensional function.

6.3 Real data

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. We can not compute analytically the optimal static weight assignment on real data, and therefore, we show the performance as the average cumulative reward $(R_t(C)/t)$.

6.3.1 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 Sect. 3). In our case, there is only one item with nonzero 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 McAuley et al. 2015), the 10M MovieLens data set,[2] and a Twitter data set, where the items are defined by the hashtags used in tweets (Pálovics et al. 2017).

6.3.2 Base rankers

We rely on two basic classes of collaborative filtering models: item-based nearest neighbor (ITEM2ITEM) (Sarwar et al. 2001) and matrix factorization (Abernethy et al. 2007). These two classes of methods represent the most successful and most popular collaborative filtering algorithms[3] Koren et al. (2009), Pilászy et al. (2015). In

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

[3] 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.

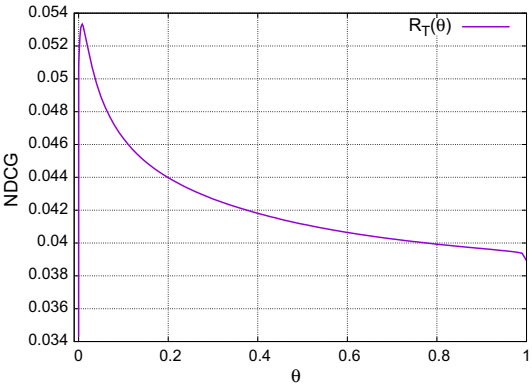

Fig. 5 Reward with various combination coefficients (θ) for the combination of OMF and ITEM2ITEM on the Amazon-CD set. In the figure, θ denotes the normalized weight of the OMF base ranker. The normalized weight for ITEM2ITEM is $1 - \theta$

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) (Pan et al. 2008), online asymmetric matrix factorization (OAMF) (Koren 2008), batch matrix factorization (MF), and (batch) implicit alternating least squares (IALS) (Hu et al. 2008). All variants use latent factors with ten dimensions. The online variants update once after 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 (Pan et al. 2008).

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.

6.3.3 Combination of two models

We start with the combination of two base models OMF and ITEM2ITEM. We let θ denote the weight of OMF in the convex combination. The average cumulative reward at the end of the episode (R_T/T), depending on θ, is shown for the Amazon-CD data set in Fig. 5. Interestingly, the optimum is reached for a combination that puts heavy weigh to ITEM2ITEM, even though OMF alone performs better than ITEM2ITEM.

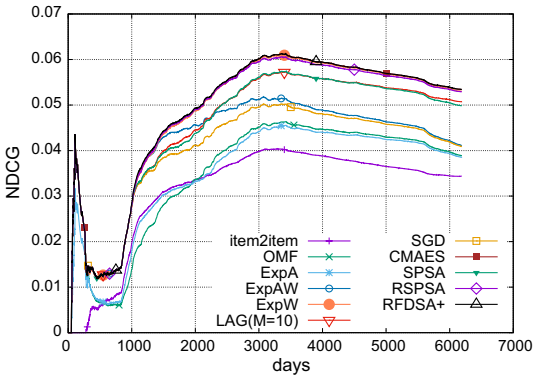

Fig. 6 Average cumulative NDCG of the ranking algorithms on the Amazon-CD set

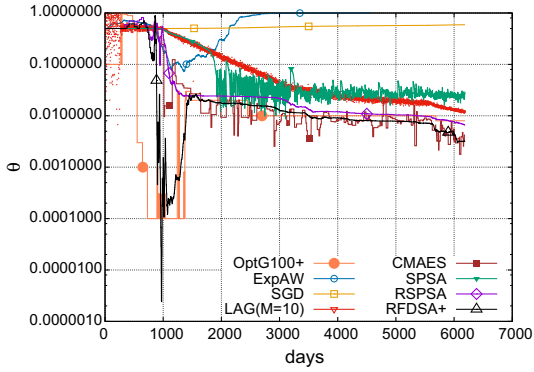

Fig. 7 The weight assignment of the ranking algorithms on the Amazon-CD set. OptG100+ corresponds to the optimal weight assignment over 100 uniform grid points, with a few additional points chosen near the presumed optimum. In the figure, θ denotes the normalized weight of the OMF base ranker. The normalized weight for item2item is $1 - \theta$

The average cumulative reward of the combination algorithms is shown in Fig. 6. 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 item-2item, 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[4] achieves a performance that equals to the best static convex combination (cf. Fig. 5). 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 Fig. 5. SGD has similar performance to ExpAW, giving also a larger weight to OMF. This is possibly because SGD and OMF optimize the same surrogate loss function. RSPSA, RFDSA+ and CMAES obtain near optimal performance. SPSA performs somewhat weaker than the other SA algorithms, and LAG with $M = 10$

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

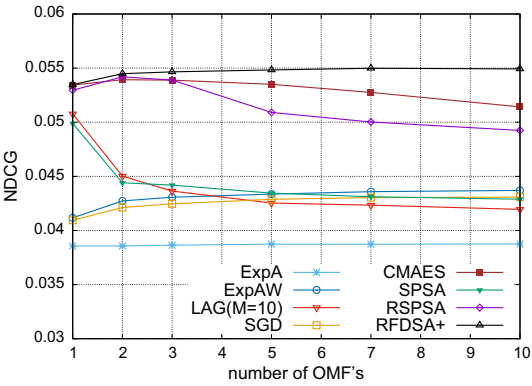

Fig. 8 Average NDCG of the ranking algorithms on the Amazon-CD set with varying number of OMFs. The combination includes one ITEM2ITEM and one to ten OMF base rankers. In the case of xOMF there is only one OMF, but the dimension of the latent factors is increased from 10 to the range of 10–100

has similar performance to SPSA. We note that the reward has a fairly large Lipschitz constant near the optimum (cf. Fig. 5), thus the weaker performance of LAG is not surprising.

The weight assignment of the combination algorithms is shown in Fig. 7. The figure includes additionally an optimal static weight assignment, i.e. $\theta_t = \text{argmax}_{\theta \in \Theta} R_t(\theta)$. By analyzing the weight assignment of the five combination algorithms that optimize NDCG directly (SPSA, RSPSA, RFDSA+, LAG and CMAES), we observe that all give ITEM2ITEM a large weight, although the weights for SPSA and LAG are further away from the optimum. RSPSA, RFDSA+, and CMAES follow closely the optimal weight assignment, and therefore, match the optimal performance of EXPW.

6.3.4 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.

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 higher dimensional space. 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 Fig. 8.

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. CMAES copes better with the increase in dimension, but the performance also drops somewhat. LAG is perhaps the algorithm most impacted by an increase of the number of base rankers, which is in accordance with the theoretical results. EXPA, EXPAW and SGD all cope

Table 1 Combination of six base rankers on five data sets

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
SGD	0.0640	0.0674	0.0353	0.1568	0.3563
EXPA	0.0628	0.0663	0.0347	0.1717	0.3486
EXPAW	0.0628	0.0664	0.0347	0.1717	0.3486
LAG (M=10)	0.0643	0.0670	0.0349	0.1701	0.3950
CMAES	0.0738	0.0745	0.0390	0.1788	0.4599
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**

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

The best performance for each data set is shown in bold

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. Results on the other data sets are similar and omitted for brevity.

In Table 1, we show the performance of the combination algorithms when all the six base rankers are used. 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 a somewhat similar conclusion as for Fig. 8: 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, RSPSA and LAG (which were performing well for two base rankers) are not performing particularly well when a larger number of models are included in the mix. CMAES is the only other algorithm that significantly improves on the base rankers performances, but it is still considerably weaker than RFDSA+ except on the Twitter data set for which the disadvantage is small.

To validate the ranking performance, we evaluate the algorithms with *MRR*@100 (see Sect. 3), shown in Table 2. The results are fairly similar to those

Table 2 Combination of six base rankers on five data sets

Algorithm	Amazon-CD	Amazon-Movies	Amazon-Electro	MovieLens	Twitter
ITEM2ITEM	0.0232	0.0248	0.0115	0.0837	0.0165
OMF	0.0199	0.0213	0.0101	0.0639	0.2732
POP	0.0357	0.0353	0.0160	0.0337	0.2410
OAMF	0.0164	0.0152	0.0072	0.0935	0.2324
MF	0.0019	0.0032	0.0023	0.0011	0.0024
IALS	0.0015	0.0026	0.0026	0.0014	0.0024
SGD	0.0361	0.0359	0.0163	0.0854	0.2468
EXPA	0.0357	0.0353	0.0160	0.0935	0.2410
EXPAW	0.0357	0.0354	0.0160	0.0935	0.2410
LAG (M=10)	0.0364	0.0356	0.0161	0.0930	0.2930
CMAES	0.0426	0.0401	0.0183	0.0989	**0.3677**
SPSA	0.0402	0.0371	0.0161	0.0899	0.2611
RSPSA	0.0361	0.0356	0.0187	0.0799	0.3502
RFDSA+	**0.0526**	**0.0510**	**0.0235**	**0.1062**	0.3662

The MRR of the base rankers is shown at the top of table, while the MRR of the combination algorithms at the bottom

The best performance for each data set is shown in bold

with *NDCG*. Therefore, we may conclude that the combination algorithms that optimize NDCG directly yield ranking performances that are strong when evaluated with other (similar) ranking measures.

6.3.5 Time complexity

For all algorithms considered, the time intensive components are (1) computing the item scores x_{tij} for each model, and (2) computing the ranking measure $r_t(\boldsymbol{\theta})$ for each parameter vector that needs to be evaluated. Both are linear in the number of items (or log-linear in the case of producing an ordered list for $r_t(\boldsymbol{\theta})$). The first component depends on the complexity of the base models. In our experiments, we included base models that are fast, which is a natural choice for online recommendation. The second component depends on the number of vectors that are evaluated in each round.

For the combinations that included six base models in the previous section (see Table 1), the second component needed at most half the time needed for the first component, but in most cases it needed a significantly smaller fraction than that. Since the first component is necessary for any combination algorithm, we may conclude that the computational overhead of the considered algorithms is moderate. This would not be the case for EXPW that evaluates a much larger number of parameter vectors. The speed of recommendation was around 50 recommendations per seconds, depending on the data set.

Based on these results, we conclude that any combination algorithm included in Table 1 can be used in an industrial application without considerable time penalty (even when the base ranking models are fast). If the base models are somewhat slower (e.g., deep networks), it could be possible to use LAG with a higher number of evaluations. However, such a choice depends on the specific application.

7 Conclusions

In this paper, we considered the task of learning the online convex combination of base recommender algorithms by stochastic optimization. We argued the potential strengths of black-box optimization for this task.

We proposed a new stochastic approximation 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 empirically shown 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.

We also proposed a new exponentially weighted algorithm with limited number of evaluation based on a grid structure, LAG. The LAG algorithm can deal well with multiple optima, but does not scale well with a larger number of base ranking models. We proved upper bounds on the regret of the algorithm, which was shown to be log-optimal. We derived the first theoretical results in the literature for online non-convex bandits with limited number of evaluations.

The relatively weak empirical performance of LAG is connected to the coarseness of the grid, which does not allow the algorithm to explore the weight vectors close to the optimum. Using a finer grid, with significantly higher number of gridpoints, would prevent LAG to explore and differentiate the value at each gridpoint. A possible remedy would be to use the more hierarchical grid constructions proposed in Bubeck et al. (2011), Grill et al. (2015). These algorithms, however, were designed for stochastic/stationary bandits, and adapting them to a non-stationarity setting is left for future research.

In our experiments, RFDSA+ was highly effective for online ranking combination. It will be interesting to evaluate the algorithm in related domains, including online combination of heterogeneous classification models. RFDSA+ extends RSPSA, which was applied successfully to optimize parameters of game programs. Since, games have discrete evaluation (e.g., 0/1 for win/loss), flat regions could also occur. Therefore, it is reasonable to assume, that RFDSA+ may improve the parameter optimization performance in domains like games.

A Proof of regret upper bounds

The proofs of the regret upper bounds rely on the regret bound of the original exponentially weighted algorithm, with an added term resulting from the discretization error of the grid. First, we prove the regret bound of ExpW (Proposition 1), and then, the regret bound of LAG (Proposition 4).

A.1 Proof of Proposition 1

Recall that the regret bound of the exponentially weighted forecaster on a K-armed game[5] with T rounds (Cesa-Bianchi and Lugosi 2006) is

$$\mathrm{E}[\mathfrak{R}_T] \leq \sqrt{\frac{T}{2} \ln K}. \tag{14}$$

Consider the difference between the cumulative reward of ExpW and the best constant point $q \in Q$. This difference is equivalent to the regret of the exponentially weighted forecaster algorithm in a $|Q|$-armed full-information game, hence it can be bounded as

$$\mathrm{E}\left[\max_{q \in Q} R_T(q) - R_T(\mathrm{ExpW})\right] \leq \sqrt{\frac{T}{2} \ln |Q|} \tag{15}$$

as we saw it in (14).

Summing the condition of the theorem, Eqs. (3) and (15), we get

$$\mathrm{E}\left[\max_{\theta \in \Theta} R_T(\theta) - R_T(\mathrm{ExpW})\right] \leq c\sqrt{T} + \sqrt{\frac{T}{2} \ln |Q|},$$

concluding our proof.

A.2 Proof of Proposition 4

Recall that the regret bound of the advice efficient exponentially weighted forecaster limited to evaluating M arms in a K-armed game with T rounds (Seldin et al. 2014) is

$$\mathrm{E}[\mathfrak{R}_T] \leq 2\sqrt{\frac{K}{M} T \ln K}. \tag{16}$$

Consider the difference between the cumulative reward of LAG and the best constant point $q \in Q$. This difference is equivalent to the regret of the advice efficient

[5] In the "Appendix", in an abuse of notation, K will denote the number arms in line with the bandit terminology, and not the length of the recommended list. For the set Q we will have $|Q| = K$.

exponentially weighted forecaster algorithm in a $|Q|$-armed limited-information game, hence it can be bounded as

$$\mathrm{E}\left[\max_{q\in Q} R_T(q) - R_T(\mathrm{LAG})\right] \leq 2\sqrt{\frac{|Q|}{M} T \ln |Q|}. \tag{17}$$

as we saw it in (16).

Summing the condition of the theorem, Eqs. (8) and (17), we get

$$\mathrm{E}\left[\max_{\theta\in\Theta} R_T(\theta) - R_T(\mathrm{LAG})\right] \leq c\frac{T}{(TM)^{\frac{1}{d+2}}} + 2\sqrt{\frac{|Q|}{M} T \ln |Q|} \leq \left(c + 2\sqrt{c' \ln |Q|}\right)\frac{T^{\frac{d+1}{d+2}}}{M^{\frac{1}{d+2}}}$$

concluding our proof.

B Proof of lower bounds

The proofs of the lower bounds rely on the proof of lower bound for a K-armed bandit with limited feedback (Seldin et al. 2014). In particular, the proof of Theorem 2 in Seldin et al. (2014) defines $K + 1$ games. In the hth game, arm h is the designated arm. The reward of the designated arm on round t is 1 with probability of $(1 + \epsilon)/2$ and 0 otherwise. For the other arms $h' \neq h$, the reward is 1 with probability of $(1 - \epsilon)/2$ and 0 otherwise. In the $\emptyset$th game, all arms behave like non-designated arms, i.e., in each round, all arms pay 1 with probability of $(1 - \epsilon)/2$ and 0 otherwise. Before the start of the episode, the environment selects randomly one of the games ($h > 0$). It is shown in Seldin et al. (2014) that the probability of being able to differentiate the hth game from the $\emptyset$th game is very small, depending on ϵ, K, M, and T. If the hth game is not detected, then no algorithm can do better than the uniform selection policy corresponding to the $\emptyset$th game. In that case, the forecaster will suffer a regret of ϵT. The tuning of ϵ is determined by a trade-off: if ϵ is too high, then it is easy to differentiate between the games; if ϵ is too low, then the regret of uniform policy is small. The proof sets $\epsilon = \frac{\sqrt{3}}{8}\sqrt{\frac{K}{MT}}$. With that value, the minimax regret bound obtained in Seldin et al. (2014) is

$$\inf \sup \mathfrak{R}_T \geq 0.03\sqrt{\frac{K}{M} T}. \tag{18}$$

In our lower bound proofs, we divide the weight space in independent boxes that correspond to arms of a bandit. We rely on the above lower bound, but we have to show that for a particular ϵ, all games defined above conform with Property 1. We start with the easier, full-information case (Proposition 3), and then follow up with the more important result of for the limited number of evaluations (Theorem 2).

B.1 Proof of Proposition 3

We map the finite armed problem and reward sequences defined in Theorem 2 of Seldin et al. (2014) into a reward function series on the domain of $\Phi = [0,1]^d$. Any game played in our framework on these reward functions has a correspondent game played in the finite armed framework, resulting the same regret. Therefore the expected minimax regret in our framework will be higher or equal than the expected minimax regret in the initial finite armed environment.

In the full-information case the reward of all arms are revealed, i.e. $K = M$. We choose $K = T^{d/3}$ to get the desired bound by using Eq. (18). The transformation and the mapping is the following. Let $\kappa = \lceil K^{\frac{1}{d}} \rceil$. We split $[0,1]^d$ into κ^d subcubes of equal size. We assign a subcube to each arm. If $K < \kappa^d$, we assign the remaining unassigned subcubes to the last arm. In each round, each subcube has a constant value, that is the same as the reward of the arm assigned to it.

To prove that the reward function series belonging to each game satisfies Eq. (5) for $\lambda = 1$, we will show that

$$\mathrm{E}\left[\max_{\theta\in\Theta} R_T(\theta)\right] - \mathrm{E}\left[\max_{k\in\mathcal{G}_{\tilde{\kappa}}} R_T(k)\right] \leq \frac{\lambda T\sqrt{d}}{\tilde{\kappa}}$$

for all $\tilde{\kappa} \in \mathbb{R}_+$, where $G_{\tilde{\kappa}}$ is a d-dimensional $\tilde{\kappa}$-grid. If $\kappa \leq \tilde{\kappa}$, the assumption is clearly satisfied because at least one gridpoint will fall into the subcube that corresponds to the optimal arm and the left hand side will be 0. Otherwise $\tilde{\kappa} < \kappa$ and the right hand side can be lower bounded by $\frac{\lambda T\sqrt{d}}{\kappa}$.

Considering the second term of the left hand side, observe that the expected maximum is larger or equal compared to the expected value of any gridpoint. Thus, we can lower bound the second term by $\frac{1-\epsilon}{2}T = \frac{1+\epsilon}{2}T - \epsilon T$.

$$\left(\mathrm{E}\left[\max_{\theta\in\Theta} R_T(\theta)\right] - \frac{1+\epsilon}{2}T\right) + \epsilon T \leq \frac{\lambda T\sqrt{d}}{\kappa}.$$

Consider now the first two terms, closed into the bracket. $\mathrm{E}\left[\max_{\theta\in\Theta} R_T(\theta)\right]$ equals to the cumulative reward of the best arm in the K armed game mapped. The second term, $\frac{1+\epsilon}{2}T$ is the expected cumulative reward of the best arm, thus this is greater or equal to the expected cumulative reward of any forecaster applied on the mentioned problem, including the exponentially weighted forecaster. Their difference is less or equal to the regret of the exponentially weighted forecaster. That lets us upper bound the first two term by the regret bound of the exponentially weighted forecaster,

$$\left(\mathrm{E}\left[\max_{\theta\in\Theta} R_T(\theta)\right] - \frac{1+\epsilon}{2}T\right) \leq \sqrt{\frac{T}{2}\log K}.$$

Now we need to show that

$$\sqrt{\frac{T}{2}\log K} + \epsilon T \leq \frac{\lambda T\sqrt{d}}{\kappa}.$$

Using $(\kappa-1)^d < K$, we lower bound the right hand side,

$$\sqrt{\frac{T}{2}\log K} + \epsilon T \leq \frac{\lambda T\sqrt{d}}{K^{\frac{1}{d}}+1}. \tag{19}$$

Recalling that $\lambda = 1$, $K = M = T^{\frac{d}{3}}$ and $\epsilon = \frac{\sqrt{3}}{8}\sqrt{\frac{K}{MT}} = \frac{\sqrt{3}}{8\sqrt{T}}$, we get

$$\sqrt{\frac{\log T}{6}} + \frac{\sqrt{3}}{8d} \leq \frac{\sqrt{T}}{T^{\frac{1}{3}}+1}$$

that is satisfied for any $d \geq 1$ and $T \geq 1$.

B.2 Proof of Theorem 2

The structure of the proof for the limited information case is similar to the proof of Proposition 3. However, now we choose $K = M^{\frac{d}{d+2}}T^{\frac{d}{d+2}}$ and $\lambda = 3$.

The computation is identical until Eq. (19),

$$\sqrt{\frac{T}{2}\log K} + \epsilon T \leq \frac{\lambda T\sqrt{d}}{K^{\frac{1}{d}}+1}.$$

Starting from here, we insert $K = M^{\frac{d}{d+2}}T^{\frac{d}{d+2}}$ and $\epsilon = \frac{\sqrt{3}}{8}\sqrt{\frac{K}{MT}} = \frac{\sqrt{3}}{8}(TM)^{\frac{-1}{d+2}}$, getting

$$\sqrt{\frac{T}{2}\frac{d}{d+2}(\log T + \log M)} + \frac{\sqrt{3}}{8}(TM)^{\frac{-1}{d+2}}T \leq \frac{\lambda T\sqrt{d}}{(TM)^{\frac{1}{d+2}}+1}.$$

Using that $\frac{\sqrt{3}}{8}(TM)^{\frac{-1}{d+2}}T < \frac{T\sqrt{d}}{(TM)^{\frac{1}{d+2}}+1}$, we get

$$\sqrt{\frac{T}{2}\frac{d}{d+2}(\log T + \log M)} \leq \frac{(\lambda-1)T\sqrt{d}}{(TM)^{\frac{1}{d+2}}+1}.$$

Using that $M \leq T^{\phi}$ and $\lambda = 3$, we get

$$\sqrt{\frac{1+\phi}{2(d+2)}\log T} \leq \frac{2\sqrt{T}}{T^{\frac{1+\phi}{d+2}}+1}.$$

We can assume that $\phi < \frac{d}{2}$ because otherwise the this theorem is weaker than Proposition 3 that is already proved in Sect. B.1. Using that and simplifying, we get

$$\sqrt{\frac{\frac{1}{d}+\frac{1}{2}}{2(1+\frac{2}{d})}\log T} \leq \frac{\sqrt{T}}{T^{\frac{1+\phi}{d+2}}}.$$

After simplification and bounding we get

$$\frac{3}{4}\log T \leq T^{\frac{d-2\phi}{d+2}},$$

which is the condition of the theorem.

Acknowledgements The research was supported by the Ministry of Innovation and Technology NRDI Office within the framework of the Hungarian Artificial Intelligence National Laboratory Program.

Funding Open access funding provided by ELKH Institute for Computer Science and Control.

References

Abernethy, J., Canini, K., Langford, J., Simma, A.: Online Collaborative Filtering. University of California at Berkeley, Technical Report (2007)

Agarwal, A., Dekel, O., Xiao, L.: Optimal algorithms for online convex optimization with multi-point bandit feedback. In: COLT, pp. 28–40 (2010)

Al-Ghossein, M., Murena, P.A., Abdessalem, T., Barré, A., Cornuéjols, A.: Adaptive collaborative topic modeling for online recommendation. In: Proceedings of the 12th ACM Conference on Recommender Systems, pp. 338–346. ACM (2018)

Amatriain, X., Agarwal, D.: Tutorial: lessons learned from building real-life recommender systems. In: Proceedings of the 10th ACM Conference on Recommender Systems, p. 433 (2016)

Au, C.K., Leung, H.F.: An empirical comparison of CMA-ES in dynamic environments. In: International Conference on Parallel Problem Solving from Nature, pp. 529–538. Springer (2012)

Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The nonstochastic multiarmed bandit problem. SIAM J. Comput. **32**(1), 48–77 (2002)

Balcan, M.F., Dick, T., Sharma, D.: Online optimization of piecewise Lipschitz functions in changing environments (2019). arXiv:1907.09137

Bennett, J., Lanning, S., et al.: The Netflix prize. In: Proceedings of KDD Cup and Workshop, vol. 2007, p. 35. New York, NY, USA (2007)

Bubeck, S., Munos, R., Stoltz, G., Szepesvári, C.: X-armed bandits. J. Mach. Learn. Res. **12**(5) (2011)

Burke, R.: Evaluating the dynamic properties of recommendation algorithms. In: Proceedings of the fourth ACM Conference on Recommender Systems, pp. 225–228. ACM (2010)

Busa-Fekete, R., Kégl, B., Éltető, T., Szarvas, G.: Ranking by calibrated adaboost. In: Proceedings of the Learning to Rank Challenge, pp. 37–48 (2011)

Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning, and Games. Cambridge University Press, Cambridge (2006)

Cohen-Addad, V., Kanade, V.: Online optimization of smoothed piecewise constant functions. In: Artificial Intelligence and Statistics, pp. 412–420. PMLR (2017)

Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-Free Optimization. SIAM, Philadelphia (2009)

Craswell, N., Zoeter, O., Taylor, M., Ramsey, B.: An experimental comparison of click position-bias models. In: Proceedings of the 2008 International Conference on Web Search and Data Mining, pp. 87–94 (2008)
Gama, J., Sebastião, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 329–338. ACM (2009)
Gomes, H.M., Barddal, J.P., Enembreck, F., Bifet, A.: A survey on ensemble learning for data stream classification. ACM Comput. Surv. (CSUR) **50**(2), 1–36 (2017)
Grill, J.B., Valko, M., Munos, R.: Black-box optimization of noisy functions with unknown smoothness. Adv. Neural Inf. Process. Syst. **28**, 667–675 (2015)
Hansen, N., Auger, A., Ros, R., Finck, S., Pošík, P.: Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In: Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation, pp. 1689–1696 (2010)
Hazan, E., Levy, K.: Bandit convex optimization: towards tight bounds. Adv. Neural Inf. Process. Syst. **27**, 784–792 (2014)
Hazan, E., Li, Y.: An optimal algorithm for bandit convex optimization (2016). arXiv:1603.04350
Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: 2008 Eighth IEEE International Conference on Data Mining, pp. 263–272 (2008)
Igel, C., Hüsken, M.: Improving the Rprop learning algorithm. In: Bothe, H., Rojas,, R. (eds.) Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000), pp. 115–121. ICSC Academic Press (2000)
Igel, C., Suttorp, T., Hansen, N.: A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 453–460 (2006)
Järvelin, K., Kekäläinen, J.: 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, pp. 41–48. ACM (2000)
Jugovac, M., Jannach, D., Karimi, M.: Streamingrec: a framework for benchmarking stream-based news recommenders. In: Proceedings of the 12th ACM Conference on Recommender Systems, pp. 269–273. ACM (2018)
Kleinberg, R.D.: Nearly tight bounds for the continuum-armed bandit problem. In: Advances in Neural Information Processing Systems, pp. 697–704 (2005)
Kocsis, L., Szepesvári, C.: Universal parameter optimisation in games based on SPSA. Mach. Learn. **63**(3), 249–286 (2006)
Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 426–434. ACM (2008)
Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley, New York (2014)
Kveton, B., Szepesvari, C., Wen, Z., Ashkan, A.: Cascading bandits: learning to rank in the cascade model. In: International Conference on Machine Learning, pp. 767–776 (2015)
Larson, J., Menickelly, M., Wild, S.M.: Derivative-free optimization methods (2019). arXiv:1904.11585
Lathia, N., Hailes, S., Capra, L.: Temporal collaborative filtering with adaptive neighbourhoods. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 796–797. ACM (2009)
Maillard, O.A., Munos, R.: Online learning in adversarial lipschitz environments. In: Machine Learning and Knowledge Discovery in Databases, pp. 305–320 (2010)
McAuley, J., Targett, C., Shi, Q., Den Hengel, Van, A.: Image-based recommendations on styles and substitutes. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 43–52. ACM (2015)
Neu, G.: Explore no more: improved high-probability regret bounds for non-stochastic bandits. Adv. Neural Inf. Process. Syst. **28**, 3168–3176 (2015)
Pálovics, R., Benczúr, A.A.: Temporal influence over the Last.fm social network. Soc. Netw. Anal. Min. **5**(1), 4 (2015)

Pálovics, R., Benczúr, A.A., Kocsis, L., Kiss, T., Frigó, E.:ACM, : Exploiting temporal influence in online recommendation. In: Proceedings of the 8th ACM Conference on Recommender Systems, pp. 273–280. ACM (2014)
Pálovics, R., Szalai, P., Pap, J., Frigó, E., Kocsis, L., Benczúr, A.A.: Location-aware online learning for top-k recommendation. Pervasive Mob. Comput. **38**, 490–504 (2017)
Pan, R., Zhou, Y., Cao, B., Liu, N.N., Lukose, R., Scholz, M., Yang, Q.: One-class collaborative filtering. In: Eighth IEEE International Conference on Data Mining, 2008. ICDM'08, pp. 502–511. IEEE (2008)
Pilászy, I., Serény, A., Dózsa, G., Hidasi, B., Sári, A., Gub, J.: Neighbor methods vs. matrix factorizationcase studies of real-life recommendations. In: LSRS Workshop at ACM RecSys (2015)
Radlinski, F., Kleinberg, R., Joachims, T.: Learning diverse rankings with multi-armed bandits. In: Proceedings of the 25th International Conference on Machine Learning, pp. 784–791. ACM (2008)
Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, pp. 285–295. ACM (2001)
Seldin, Y., Bartlett, P., Crammer, K., Abbasi-Yadkori, Y.: Prediction with limited advice and multiarmed bandits with paid observations. In: ICML, pp. 280–287 (2014)
Shamir, O.: An optimal algorithm for bandit and zero-order convex optimization with two-point feedback. J. Mach. Learn. Res. **18**(1), 1703–1713 (2017)
Spall, J.C.: Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE Trans. Autom. Control **37**, 332–341 (1992)
Töscher, A., Jahrer, M., Bell, R.M.: The BigChaos solution to the Netflix grand prize. Netflix prize documentation, pp. 1–52. (2009)
Vinagre, J., Jorge, A.M., Gama, J.: Evaluation of recommender systems in streaming environments. In: Workshop on 'Recommender Systems Evaluation: Dimensions and Design' (REDD 2014), held in conjunction with RecSys 2014 (2014)
Voorhees, E.M., Tice, D.M.: The TREC-8 question answering track report. In: TREC, vol. 99, pp. 77–82 (1999)
Yue, Y., Joachims, T.: Interactively optimizing information retrieval systems as a dueling bandits problem. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 1201–1208. ACM (2009)
Zoller, D., Doerfel, S., Pölitz, C., Hotho, A.: Leveraging user-interactions for time-aware tag recommendations. In: RecTemp@ RecSys, pp. 9–15 (2017)

Erzsébet Frigó is a Ph.D. Candidate at Institute for Computer Science and Control (SZTAKI), Eötvös Loránd Research Network (ELKH). She has an M.Sc. in Computer Science Engineering from Budapest University of Technology and Economics (BME). Her research focuses on recommender systems and online machine learning.

Levente Kocsis is a senior researcher at Institute for Computer Science and Control (SZTAKI), Eötvös Loránd Research Network (ELKH). He has a Ph.D. in Computer Science from Maastricht University. His research interests are in the fields of recommender systems, online machine learning, stochastic optimization and games.