Combinatorial Online Prediction via Metarounding

Fujita, Takahiro
Department of Informatics, Kyushu University

Hatano, Kohei
Department of Informatics, Kyushu University

Takimoto, Eiji
Department of Informatics, Kyushu University

https://hdl.handle.net/2324/1786656

Combinatorial Online Prediction via Metarounding

Takahiro Fujita[1], Kohei Hatano[1], and Eiji Takimoto[1]
{takahiro.fujita, hatano,eiji}@inf.kyushu-u.ac.jp[1]

Department of Informatics, Kyushu University

Abstract. We consider online prediction problems of combinatorial concepts. Examples of such concepts include *s-t* paths, permutations, truth assignments, set covers, and so on. The goal of the online prediction algorithm is to compete with the best fixed combinatorial concept in hindsight. A generic approach to this problem is to design an online prediction algorithm using the corresponding offline (approximation) algorithm as an oracle. The current state-of-the art method, however, is not efficient enough. In this paper we propose a more efficient online prediction algorithm when the offline approximation algorithm has a guarantee of the integrality gap.

1 Introduction

Online prediction problems of combinatorial concepts arise in many situations such as routing, ranking and scheduling. Examples of such combinatorial concepts include, *s-t* paths, set covers, permutations and so on. In a combinatorial online prediction problem, we assume a finite set $\mathcal{C} \subseteq \mathbb{R}^n$ of combinatorial concepts, where each combinatorial concept is represented as a vector in $\mathbb{R}^n$. Then we consider the following protocol between the player and the adversary: For each trial $t = 1, ..., T$, (i) the player predicts $\boldsymbol{c}_t \in \mathcal{C}$, (ii) the adversary returns a loss vector $\boldsymbol{\ell}_t \in \mathcal{L} \subseteq [0,1]^n$, and (iii) the player incurs loss $\boldsymbol{c}_t \cdot \boldsymbol{\ell}_t$. Typical goal of the player is to minimize the regret $\sum_{t=1}^{T} \boldsymbol{c}_t \cdot \boldsymbol{\ell}_t - \min_{\boldsymbol{c} \in \mathcal{C}} \sum_{t=1}^{T} \boldsymbol{c} \cdot \boldsymbol{\ell}_t$.

A straightforward approach for combinatorial online prediction problems is to apply Hedge [8]. Given a set of experts (i.e., prediction strategies or algorithms), Hedge is guaranteed to predict almost as well as the best expert in hindsight. So, with each combinatorial concept $\boldsymbol{c} \in \mathcal{C}$ as an expert, Hedge can achieve a good regret bound [1]. This approach, however, is inefficient in general since there are exponentially many concepts in the class $\mathcal{C}$ and thus Hedge takes exponential time at each trial.

There are many results on efficient combinatorial online prediction algorithms for individual concepts such as *k*-sets [21], permutations [11, 23, 24], spanning trees [6] and so on. There are some work on classes of combinatorial concepts [6, 15, 19].

---

[1] Note that Hedge is suboptimal in some cases (see [3] for the details).

A generic alternative approach for combinatorial online prediction is to convert offline approximation algorithms for combinatorial optimization to online prediction algorithms. More precisely, we assume an offline α-approximation algorithm A for the linear optimization problem over $\mathcal{C}$. The algorithm is supposed to take a loss vector $\ell \in [0, 1]^n$ as input and outputs $c \in \mathcal{C}$ such that $c \cdot \ell \leq \alpha \min_{c' \in \mathcal{C}} c' \cdot \ell$. We assume that the player can use the offline algorithm A as an oracle, and each call of the oracle can be done in a unit time.

The goal of the player is to minimize the α-regret:

$$\sum_{t=1}^{T} c_t \cdot \ell_t - \alpha \min_{c \in \mathcal{C}} \sum_{t=1}^{T} c \cdot \ell_t.$$

The α-regret measures the difference between cumulative loss of the player and that of an α-approximate fixed concept in hindsight, which can be computed by the approximation algorithm A.

There are two main previous researches on the player's strategy for combinatorial online prediction problems where approximation algorithms are available. First, Kalai and Vempala proposed Follow the perturbed leader (FPL [14]). FPL uses an exact offline optimization algorithm (i.e., $\alpha = 1$). The 1-regret bound of FPL is $O(\sqrt{T})$ and its running time per trial is $O(n)$. FPL also works with α-approximation algorithms, however, its α-regret bound becomes $O(\alpha^T \sqrt{T})$ in general.

Kakade et al. proposed a different strategy using α-approximation algorithms, which achieves $O(\alpha \sqrt{T})$ α-regret bound [13]. The running time of the algorithm is $O(poly(n)T)$. Unfortunately, the time complexity at each trial depends on T, which is not desirable in practice.

In this paper, we consider a slightly stronger assumption on the offline approximation algorithms. We assume that the player has access to the following approximation algorithm A:

Assumption 1 *Given $\ell \in [0, 1]^n$ as input, A outputs $c \in \mathcal{C}$ such that $c \cdot \ell \leq \alpha \min_{x \in \mathcal{P}} x \cdot \ell$ for some $\alpha > 1$, where $\mathcal{P}$ is a convex set such that $\mathcal{P} \supset \mathcal{C}$, and linear optimization over $\mathcal{P}$ can be done in polynomial time in n.*

This assumption is motivated by the fact that many combinatorial optimization problems have LP relaxation schemes. Then, our main result is stated as follows:

Theorem 1 *Under Assumption 1, there exists a strategy of the player whose α-regret bound is $O((\alpha + \varepsilon)\sqrt{T})$ and its running time is polynomial in n and $1/\varepsilon$ for any $\varepsilon > 0$.*

The key notion of our result is *metarounding*, a robust version of rounding for relaxation-based approximation schema. Originally, metarounding was proposed by Carr and Vempala for approximately solving the multicast congestion problem [4]. We will show that metarounding is quite suitable for online prediction of combinatorial concepts as well.

One of our technical contribution is a new construction of metarounding using a *boosting* algorithms. Boosting [16] is a technique to construct highly accurate

hypothesis by combining moderately accurate base hypotheses, which apparently seems nothing to do with metarounding. But, in fact, robustifying rounding and boosting hypotheses share a common structure and they are formulated as similar optimization problems.

Our algorithm is adaptive in the sense that it does not require the explicit knowledge on the approximation ratio α, which is advantageous in practice. Our preliminary experiments show that our algorithm indeed obtains better approximation ratios than theoretically guaranteed and its running time is much faster than a method based on previous work of Carr and Vempala.

2   Preliminaries

Let $\Phi : \Gamma \to \mathbb{R}$ be a strictly convex function defined on a closed convex set $\Gamma \subseteq \mathbb{R}^n$. The Bregman divergence Δ_Φ with respect to Φ is defined as $\Delta_\Phi(\boldsymbol{p}, \boldsymbol{q}) = \Phi(\boldsymbol{p}) - \Phi(\boldsymbol{q}) - \nabla\Phi(\boldsymbol{q}) \cdot (\boldsymbol{p} - \boldsymbol{q})$.

The *unnormalized relative entropy* $\Delta(\boldsymbol{p}, \boldsymbol{q})$ from $\boldsymbol{q} \in \mathbb{R}^n_+$ to $\boldsymbol{p} \in \mathbb{R}^n_+$ is defined as

$$\Delta(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^n p_i \ln \frac{p_i}{q_i} + \sum_{i=1}^n q_i - \sum_{i=1}^n p_i.$$

It is known that $\Delta(\boldsymbol{p}, \boldsymbol{q}) \geq 0$ and $\Delta(\boldsymbol{p}, \boldsymbol{q}) = 0$ if and only if $\boldsymbol{p} = \boldsymbol{q}$. Unnormalized relative entropy is not symmetric in general, i.e., $\Delta(\boldsymbol{p}, \boldsymbol{q}) \neq \Delta(\boldsymbol{q}, \boldsymbol{p})$ for some $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^n_+$. In fact, unnormalized relative entropy is a special case of Bregman divergence [5].

3   Main Structure

In this section, we describe the main structure of our algorithms for combinatorial online prediction. Let $\mathcal{P} \subset \mathbb{R}^n_+$ such that (i) $\mathcal{P}$ is convex, (ii) $\mathcal{P}$ contains $\mathcal{C}$, and (iii) linear optimization over $\mathcal{P}$ can be done in polynomial time in n. For example, $\mathcal{P}$ might be represented a set of $poly(n)$ linear constraints. Then linear programming over $\mathcal{P}$ can be solved in polynomial time, say, by using interior point methods. Also, $\mathcal{P}$ might be described as linear constraints and semidefinite constraints. Then linear optimization over $\mathcal{P}$ belongs to semidefinite programs, which are solvable efficiently.

Our algorithms consist of two components, an online prediction algorithm and a metarounding algorithm. At each trial t, our online prediction algorithm predict $\boldsymbol{x}_t$ not in $\mathcal{C}$, but in a "relaxed" space $\mathcal{P}$. Then the metarounding algorithm chooses $\boldsymbol{c}_t \in \mathcal{C}$ by "rounding" $\boldsymbol{x}_t$, where, roughly speaking, $\boldsymbol{c}_t$ is close to $\boldsymbol{x}_t$. Then, the regret of the algorithm would be close to that of online prediction algorithms over $\mathcal{P}$. We give a formal definition of metarounding algorithms as follows.

Definition 1 *An algorithm A is a metarounding algorithm if A is given $\boldsymbol{x} \in \mathcal{P}$ as input and outputs $\boldsymbol{c} \in \mathcal{C}$ such that for any $\boldsymbol{\ell} \in \mathcal{L} \subseteq [0,1]^n$, $E[\boldsymbol{c} \cdot \boldsymbol{\ell}] \leq \alpha \boldsymbol{x} \cdot \boldsymbol{\ell}$, where the expectation is taken w.r.t. the internal randomness of the algorithm A.*

The notion of metarounding was first proposed by Carr and Vempala [4]. Given an α-approximation algorithm with relaxation, which requires $\boldsymbol{\ell} \in \mathcal{L} = [0,1]^n$ and outputs $\boldsymbol{c} \cdot \boldsymbol{\ell} \le \alpha \min_{\boldsymbol{x} \in \mathcal{P}} \boldsymbol{x} \cdot \boldsymbol{\ell}$, they show how to construct a metarounding algorithm by using the approximation algorithm as an oracle. We will propose more efficient methods to construct a metarounding algorithm from the approximation algorithm.

A related notion is proposed by Kalai and Vempala [14]. An algorithm has α-pointwise approximation property if, given any $\boldsymbol{\ell} \in \mathcal{L}$, the algorithm outputs $\boldsymbol{c} \in \mathcal{C}$ such that $c_i \le \alpha c_i^*$ for $i = 1, \dots, n$, where $\boldsymbol{c}^* = \arg \min_{\boldsymbol{c}' \in \mathcal{C}} \boldsymbol{c}' \cdot \boldsymbol{\ell}$. Kalai and Vempala showed that FPL with approximation algorithm with this property can achieve good α-regret bounds [14]. In particular, when $\mathcal{L} = [0,1]^n$, the notion of metarounding turns out to be equivalent with α-pointwise approximation property in some sense (shown in Proposition 4). But, in general cases where $\mathcal{L} \subset [0,1]^n$, both notions seem to be incomparable with each other. Further, as we will show, the notion of metarounding is applicable more widely than the point-wise approximation property.

We show that any online prediction algorithm A whose prediction space is $\mathcal{P}$ can be combined with a metarounding algorithm. At each trial t, the combined algorithm gets the prediction $\boldsymbol{p}_t$ of the precition algorithm A, gives the prediction $\boldsymbol{p}_t$ to the metarounding algorithm as an input, and get the combinatrial concept $\boldsymbol{c}_t$, which is used as the prediction of the combined algorithm.

Proposition 2 *Suppose that there exists an online prediction algorithm A whose prediction $\boldsymbol{p}_t$ at each trial t belongs to $\mathcal{P}$ and its 1-regret w.r.t. $\mathcal{P}$ is bounded by Reg_A. Then, the α-regret of A combined with a metarounding algorithm w.r.t. the concept class $\mathcal{C}$ is at most αReg_A.*

Proof. Assume that the algorithm A with a metarounding algorithm predicts $\boldsymbol{c}_t$ at each trial t. Then,

$$\sum_{t=1}^{T} \boldsymbol{c}_t \cdot \boldsymbol{\ell}_t \le \alpha \sum_{t=1}^{T} \boldsymbol{p}_t \cdot \boldsymbol{\ell}_t \text{ (by definition of metarounding)}$$

$$\le \alpha \min_{\boldsymbol{p} \in \mathcal{P}} \sum_{t=1}^{T} \boldsymbol{p} \cdot \boldsymbol{\ell}_t + \alpha Reg_A \text{ (by definition of algorithm } A)$$

$$\le \alpha \min_{\boldsymbol{c} \in \mathcal{C}} \sum_{t=1}^{T} \boldsymbol{c} \cdot \boldsymbol{\ell}_t + \alpha Reg_A \text{ (since } \mathcal{C} \subseteq \mathcal{P}).$$

$\square$

As corollaries, combined with a metarounding algorithm, Follow the Regularized Leader(FTRL, e.g., [10]) and FPL achieve good α-regret bounds. In particular, FTRL generalizes popular algorithms such as Hedge [8] and OGD [25], respectively. The details of the algorithms are shown in Algorithm 2 and 1, respectively.

Corollary 3 *Let* $\lambda = \max_{t, \boldsymbol{x} \in \mathcal{P}} \boldsymbol{\ell}_t^T [\nabla^2 \Phi(\boldsymbol{x})]^{-1} \boldsymbol{\ell}_t$ *and* $D = \max_{\boldsymbol{x} \in \mathcal{P}} \Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}_1)$. *Then, with an appropriate choice of* η, *the* α*-regret of* $FTRL(\eta)$ *with Metarounding is* $O(\alpha \sqrt{\lambda D T})$.

Corollary 4 *Let* $D \geq |\boldsymbol{c} - \boldsymbol{c}'|_1$ *for all* $\boldsymbol{c}, \boldsymbol{c}' \in C$. *Then, with an appropriate choice of* η, *the* α*-regret of* $FPL(\eta)$ *with Metarounding is* $O(\alpha D \sqrt{nT})$.

Algorithm 1 FTRL(η) with Metarounding

---

1. Let $\boldsymbol{x}_1 \in \mathcal{P}$ be any initial point.
2. For $t = 1, \ldots, T$
   (a) Run the metarounding with $\boldsymbol{x}_t$ and get $\boldsymbol{c}_t \in \mathcal{C}$.
   (b) Predict $\boldsymbol{c}_t$ and incur loss $\boldsymbol{c} \cdot \boldsymbol{\ell}_t$.
   (c) Update $\boldsymbol{x}_{t+1/2} = \arg\min_{\boldsymbol{x} \in \mathcal{P}} \eta \boldsymbol{x} \cdot \sum_{j=1}^{t} \boldsymbol{\ell}_j + \Phi(\boldsymbol{x})$.
   (d) (Projection) Let $\boldsymbol{x}_{t+1} = \arg\min_{\boldsymbol{x} \in \mathcal{P}} \Delta_\Phi(\boldsymbol{x}, \boldsymbol{x}_{t+1/2})$.

Algorithm 2 FPL(η) with Metarounding

---

1. Let $\boldsymbol{\ell}_0 = \boldsymbol{0}$.
2. For $t = 1, \ldots, T$
   (a) Solve the linear program over $\mathcal{P}$: $\boldsymbol{x}_t = \arg\min_{\boldsymbol{x} \in \mathcal{P}} \boldsymbol{x} \cdot \left(\sum_{j=0}^{t-1} \boldsymbol{\ell}_j + \boldsymbol{p}_t \right)$, where $\boldsymbol{p}_t$ is a uniform random vector in $[0, 1/\eta]^n$.
   (b) Run the metarounding algorithm with $\boldsymbol{x}_t$ and get $\boldsymbol{c}_t \in \mathcal{C}$.
   (c) Predict $\boldsymbol{c}_t \in \mathcal{C}$ and incur the loss $\boldsymbol{c}_t \cdot \boldsymbol{\ell}_t$.

---

4 Examples and Applications of Metarounding

We show some examples and applications of metarounding.

4.1 Online Set Cover

Let S be a finite set and $U \subseteq 2^S$ be a fixed set of the subsets of S. A cover is a subset U' of U that satisfies $\bigcup_{u \in U'} u = S$. The online set cover problem is stated as follows: For each trial t, (i) the player predicts a cover $U_t \subseteq U$, (ii) the adversary returns weights $\boldsymbol{\ell}_t \in [0, 1]^U$, and (iii) the player incurs loss $\sum_{u \in U_t} \ell_t(u)$. The problem is a combinatorial online prediction problem with the concept class $\mathcal{C} = \{ \boldsymbol{c} \in \{0, 1\}^U \mid \{u \in U \mid c(u) = 1\} \text{ is a cover} \}$ and the loss vector space $\mathcal{L} = [0, 1]^U$.

The corresponding offline optimization problem is the weighted minimum set cover problem. The problem has an $O(\log |S|)$ approximation algorithm using an LP-relaxation and a randomized metarounding [18]. The relaxed space $\mathcal{P} \subseteq [0,1]^U$ is described as the set of feasible solutions $\boldsymbol{x} \in [0,1]^U$ that satisfies the following linear constraints: $\sum_{u \in U: s \in u} x(u) \geq 1$ ($\forall s \in S$). The metarounding is simple: Given a feasible solution $\boldsymbol{x} \in \mathcal{P}$ as input, set $c(u) = 1$ with probability $x(u)$ and $c(u) = 0$ with probability $1 - x(u)$. It is shown that the metarounding has approximation factor $O(\log |S|)$. That is, for any loss vector $\boldsymbol{\ell} \in [0,1]^U$, it holds that $E[\boldsymbol{c} \cdot \boldsymbol{\ell}] = O(\log |S|)\boldsymbol{x} \cdot \boldsymbol{\ell}$.

4.2 Online MAX-SAT

Let $\{C_1, C_2, \ldots, C_m\}$ be a fixed set of disjunctive clauses over a set of k Boolean variables. Each clause C_i is a disjunction of some literals, where a literal is either a Boolean variable or its negation. The online MAX-SAT problem is the following: For each trial t, (i) the player predicts an assignment $\boldsymbol{a}_t \in \{0,1\}^k$ to the variables, (ii) the adversary returns weights $\boldsymbol{\ell}_t \in [0,1]^m$ for the clauses, and (iii) the player gets reward defined by the sum of weights $\ell_{t,i}$ over the clauses C_i satisfied by $\boldsymbol{a}_t$. The problem is the *reward version* of a combinatorial online prediction problem with the concept class $\mathcal{C}$ and *reward* vector space $\mathcal{L}$ as described below. The class $\mathcal{C}$ consists of vectors in $\{0,1\}^n$ for $n = k + m$ such that the first k bit vector $\boldsymbol{a}$ represents the truth assignment and the last m bit vector $\boldsymbol{b}$ represents the truth values of the clauses for the assignment $\boldsymbol{a}$. That is, for each $1 \leq i \leq m$, $b_i = 1$ if and only if C_i is satisfied by $\boldsymbol{a}$. Note that the last m bits $\boldsymbol{b}$ are determined by the first k bits $\boldsymbol{a}$. The reward space $\mathcal{L}$ consists of vectors $0^k \boldsymbol{\ell}$ where the first k bits are 0 and $\boldsymbol{\ell} \in [0,1]^m$ represents the weights. So, the dot product of a concept $\boldsymbol{c} = \boldsymbol{ab}$ and a reward $0^k \boldsymbol{\ell}$ becomes $\boldsymbol{b} \cdot \boldsymbol{\ell}$, which is the reward of the truth assignment $\boldsymbol{a}$ for the weights $\boldsymbol{\ell}$, as required.

The corresponding offline optimization problem is the weighted MAX-SAT problem. The problem has an $3/4$ approximation algorithm using an LP-relaxation and a randomized metarounding [9]. The relaxed space $\mathcal{P} \subseteq [0,1]^n$ is described as the set of feasible solutions $\boldsymbol{x} = \boldsymbol{yz} \in [0,1]^n$ that satisfies the following linear constraints: $\sum_{j \in S_i^+} y_j + \sum_{j \in S_i^-} (1 - y_j) \geq z_i$ $(1 \leq \forall i \leq m)$, where S_i^+ (S_i^-, resp.) denotes the set of Boolean variables occurring nonnegated (negated, resp.) in C_i. The metarounding only computes the first k bit vector $\boldsymbol{a}$ from a relaxed solution $\boldsymbol{y} \in [0,1]^k$ in the following way: Let d be the flip of a fair coin. If $d = 0$, then choose $\boldsymbol{a}$ from $\{0,1\}^n$ uniformly at random, and if $d = 1$, then for each $1 \leq j \leq k$, set $a_j = 1$ with probability y_j and set $a_j = 0$ with probability $1 - y_j$. Note again that the last m bit vector $\boldsymbol{b}$ is determined by $\boldsymbol{a}$. It is shown that the metarounding has approximation factor $3/4$. That is, for any weight vector $\boldsymbol{\ell} \in [0,1]^m$, it holds that $E[\boldsymbol{b} \cdot \boldsymbol{\ell}] \geq (3/4)\boldsymbol{y} \cdot \boldsymbol{\ell}$.

4.3 Other Examples and Applications

Other applications include the rank aggregation problem for which metarounding algorithms exist [1,2]. An online version of the problem was investigated in [24].

Interestingly enough, in this case, $\mathcal{L} \neq [0,1]^n$, for which the pointwise property does not hold.

As far as we know, all rounding methods used in relaxation-based approximation algorithms are metarounding as well. In general, however, there might be concept classes $\mathcal{C}$ for which metarounding algorithms are not known and only α-approximation algorithms are available. Also, one might prefer approximation algorithms with better approximation ratios to metarounding algorithms. In fact, in our experiments, we show a modification of a set covering algorithm, which is not known to metarounding but has better approximation ratio. In the next section, we show a construction of metarounding using α-approximation algorithms.

5   Construction of Metarounding Algorithms

In this section, we describe our metarounding algorithms for $\mathcal{L} = [0.1]^n$ using an α-approximation algorithm with the relaxation scheme in Assumption 1 as an oracle. Recall that the approximation algorithm, given $\boldsymbol{\ell} \in \mathcal{L}$, is supposed to output $\boldsymbol{c} \in \mathcal{C}$ such that $\boldsymbol{c} \cdot \boldsymbol{\ell} \leq \alpha \min_{\boldsymbol{x} \in \mathcal{P}} \boldsymbol{x} \cdot \boldsymbol{\ell}$. The following characterization of metarounding for $\mathcal{L} = [0,1]^n$ is useful.

Proposition 5 *Suppose that $\mathcal{L} = [0,1]^n$. A is a metarounding algorithm if and only if given input $\boldsymbol{x} \in \mathcal{P}$, A outputs $\boldsymbol{c} \in \mathcal{C}$ such $E[c_i] \leq \alpha x_i$ for each $i = 1, \ldots, n$.*

Proof. Suppose that A is a metarounding algorithm. Then, for $\boldsymbol{\ell} \in [0,1]^n$ such that $\ell_i = 1$ for some i and $\ell_j = 0$ for $j \neq i$, it must be that $E[c_i] \leq \alpha x_i$. On the other hand, If $E[c_i] \leq \alpha x_i$, it trivially follows that A is a metarounding algorithm.   □

Due to Proposition 5, the problem of constructing a metarounding algorithm is reduced to finding a convex combination $\boldsymbol{\lambda}$ over $\mathcal{C}$ such that $\sum_{\boldsymbol{c} \in \mathcal{C}} \lambda_{\boldsymbol{c}} c_i \leq \alpha x_i$ $(i = 1, \ldots, n)$. That is, by choosing a combinatorial concept $\boldsymbol{c} \in \mathcal{C}$ randomly according to the convex combination $\boldsymbol{\lambda}$, we get that $E[\boldsymbol{c} \cdot \boldsymbol{\ell}] \leq \alpha \boldsymbol{x} \cdot \boldsymbol{\ell}$ for any $\boldsymbol{\ell} \in [0,1]^n$. Note that the size of $\mathcal{C}$ is exponentially large w.r.t. n in general. Therefore, a naive linear programming formulation over $\mathcal{C}$ to find the convex combination λ would take exponential time.

As noted in the previous section, the first metarounding algorithm was proposed by Carr and Vempala [4]. They formulate a linear program over $\mathcal{C}$ (so the size of the problem could be exponential!). Yet, surprisingly, they showed the problem is solvable by the ellipsoid method (see, e.g, [17]) in polynomial time.

Their theoretical result is quite beautiful, however, might not be practical in the following reasons. First, the ellipsoid method is often much slower in practice, compared to the simplex method or interior point methods. The number of iterations of the ellipsoid method is $O(n^2 \ln \frac{R}{\varepsilon})$, where R is the radius of the initial ellipsoid which contains the feasible region and ε is a precision parameter [2]. Its time complexity per iteration is $O(n^2)$, under the assumption that the

---

[2] In addition, to achieve this bound, we need to allow the ellipsoid method to violate feasible constraints by amount of ε.

running time of the approximation algorithm is constant. The $O(n^4)$ dependence of its time complexity makes the algorithm impractical.

Second, the ellipsoid method requires the knowledge about R which is sometimes not attainable in advance. Setting sufficiently large R would work, but could result in unnecessary computation. A more detailed treatment of R and ε for rational linear programs is found in, e.g., Schrijver's book [17].

5.1   Our formulation

Our formulation is a modification of the original formulation by Carr and Vempala with an additional advantage. The advantage is that our formulation does not require the knowledge of the approximation ratio α of the rounding algorithm. This property is beneficial since, as is often the case, theoretical bounds of the approximation ratio are loose. In other words, our formulation can take advantage of such situations. Our formulation is in problem (1). By linear programming duality, the equivalent dual problem turns out to be problem (2).

$$
\begin{aligned}
\min_{\boldsymbol{\lambda},\beta} \ & \beta \quad\quad\quad\quad\quad\quad\quad (1)\\
\text{s.t.} \ & \sum_{\boldsymbol{c}\in\mathcal{C}} \lambda_{\boldsymbol{c}} c_i \leq \beta x_i \ (i=1,\dots,n),\\
& \sum_{\boldsymbol{c}\in\mathcal{C}} \lambda_{\boldsymbol{c}} \geq 1,\\
& \lambda_{\boldsymbol{c}} \geq 0 \ (\boldsymbol{c}\in\mathcal{C}).
\end{aligned}
$$

$$
\begin{aligned}
\max_{\boldsymbol{\ell},\gamma} \ & \gamma \quad\quad\quad\quad\quad\quad\quad (2)\\
\text{s.t.} \ & \boldsymbol{c}\cdot\boldsymbol{\ell} \geq \gamma \ (\boldsymbol{c}\in\mathcal{C}),\\
& \boldsymbol{\ell}\cdot\boldsymbol{x} \leq 1,\\
& \boldsymbol{\ell} \geq \boldsymbol{0}, \ \gamma \geq 0.
\end{aligned}
$$

Lemma 1 *Suppose that there exists a rounding algorithm which, given input $\boldsymbol{x} \in \mathcal{P}$, outputs $\boldsymbol{c} \in \mathcal{C}$ s.t. $\boldsymbol{c}\cdot\boldsymbol{\ell} \leq \alpha\boldsymbol{x}\cdot\boldsymbol{\ell}$ for any $\boldsymbol{\ell} \in [0,1]^n$. Then, the optimum of problem (1) is at most α.*

Proof. We prove this lemma by contradiction. Let $(\boldsymbol{\lambda}^*,\beta^*)$ and $(\boldsymbol{\ell}^*,\gamma^*)$ be optimal solutions of problems (1) and (2), respectively. Note that, by duality of linear programs, $\beta^* = \gamma^*$. Suppose that $\beta^* = \gamma^* > \alpha$. Then, by using the rounding algorithm with input $\boldsymbol{\ell}^*$, the algorithm outputs some $\boldsymbol{c} \in \mathcal{C}$ such that $\boldsymbol{c}\cdot\boldsymbol{\ell}^* \leq \alpha\boldsymbol{x}\cdot\boldsymbol{\ell}^* \leq \alpha$. On the other hand, this violates the constraint that $\boldsymbol{c}\cdot\boldsymbol{\ell}^* \geq \gamma^* > \alpha$. So, it contradicts the assumption that $(\boldsymbol{\ell}^*,\gamma^*)$ is an optimal solution.   □

5.2   Metarounding by Boosting

Now we are ready to describe our algorithm for constructing metarounding. The dual problem (2), roughly speaking, can be viewed as the problem of finding a "difficult" loss vector $\boldsymbol{\ell}$ such that $\boldsymbol{c}\cdot\boldsymbol{\ell}$ is large for each $\boldsymbol{c} \in \mathcal{C}$ under some constraints. Here we have access to an α-approximation algorithm with relaxation.Our key observation is that the problem has a similar structure as boosting (e.g., [16]): The problem of boosting is to find a "difficult" distribution over data

such for which any base hypotheses (weak hypotheses) have low weighted accuracy, where the booster has access to weak learner which produce a hypothesis with reasonably small error w.r.t. a given distribution over data.

In fact, we will prove that a boosting-type algorithm works for constructing a metarounding algorithm. Our algorithm, Metarounding by Boosting (MBB), is based on the boosting algorithm SoftBoost [20]. Note that a straightforward application of SoftBoost does not work for our metarounding problem. SoftBoost is applicable for an entropy-maximizing problem of probability distributions with constraints, while our problem deals with non-negative vectors with constraints.

For the time being, we assume that we know an upper bound L on L_ε, defined as $L_\varepsilon = \max\{\|\boldsymbol{\ell}\|_1 \mid \boldsymbol{\ell}$ is an optimal solution of the dual problem (2) over $\mathcal{C}' \subseteq \mathcal{C}$ whose solution is at most $\alpha + \varepsilon.$ $\}$. L_ε is the constant determined by $\mathcal{P}$, $\mathcal{C}$ and ε. Later, we will explain how to remove this assumption.

The description of MBB is given in Algorithm 3. MBB works in iterations. At each iteration k, MBB solves a modified subproblem of the dual problem (2), which is a convex optimization problem. The objective is unnormalized relative entropy from the initial vector $\boldsymbol{\ell}_1$. Note that, by definition of unnormalized relative entropy, any feasible solution satisfies $\boldsymbol{\ell} \geq \mathbf{0}$. So, we can remove the positivity constraint.

Algorithm 3 Metarounding by Boosting (MBB)

Input: $\boldsymbol{x} \in \mathcal{P}$, $L > 0$.

1. Let $\boldsymbol{\ell}_1 = \frac{1}{n}\mathbf{1}$ and let $\mathcal{C}_1 = \emptyset$.
2. For $k = 1, \ldots,$
    (a) Run the approximation algorithm A with input $\boldsymbol{\ell}_k$ and get $\boldsymbol{c}_k \in \mathcal{C}$. Let $\mathcal{C}_{k+1} = \mathcal{C}_k \cup \{\boldsymbol{c}_k\}$ and let $\hat{\gamma}_{k+1} = \max_{j=1,\ldots,k} \boldsymbol{c}_k \cdot \boldsymbol{\ell}_k + \varepsilon$.
    (b) Update $\boldsymbol{\ell}_{k+1}$ as

$$\boldsymbol{\ell}_{k+1} = \arg\min_{\boldsymbol{\ell}} \Delta(\boldsymbol{\ell}, \boldsymbol{\ell}_1) \tag{3}$$

$$\text{sub.to } \boldsymbol{c} \cdot \boldsymbol{\ell} \geq \hat{\gamma}_{t+1} \ (\boldsymbol{c} \in \mathcal{C}_{t+1}), \ \boldsymbol{\ell} \cdot \boldsymbol{x} \leq 1, \ \boldsymbol{\ell} \cdot \mathbf{1} \leq L.$$

    (c) **If** problem (3) is infeasible, let $K = k + 1$ and break;
3. Solve problem (2) for the reduced set $\mathcal{C}_K$ and output its Lagrange multipliers $\boldsymbol{\lambda}$.

Lemma 2 *For each $k = 1, \ldots, K - 1$, $\Delta(\boldsymbol{\ell}_{k+1}, \boldsymbol{\ell}_1) - \Delta(\boldsymbol{\ell}_k, \boldsymbol{\ell}_1) \geq \Delta(\boldsymbol{\ell}_{k+1}, \boldsymbol{\ell}_k)$.*

Proof. Let $\mathcal{D}_k$ be the feasible set in problem (3) at k-th iteration. Observe that $\hat{\gamma}_k$ is non-decreasing because of max function. So, we have $\mathcal{D}_{k+1} \subseteq \mathcal{D}_k$. Now, by Generalized Pythagorean Theorem for Bregman divergences (see, e.g., [5]), $\Delta(\boldsymbol{\ell}_{k+1}, \boldsymbol{\ell}_1) \geq \Delta(\boldsymbol{\ell}_k, \boldsymbol{\ell}_1) + \Delta(\boldsymbol{\ell}_{k+1}, \boldsymbol{\ell}_k)$. □

Let $C = \max\{\|\boldsymbol{c}\|_\infty \mid \boldsymbol{c} \in \mathcal{C}\}$. Then the following lemma holds.

Lemma 3 $\Delta(\boldsymbol{\ell}_{t+1}, \boldsymbol{\ell}_t) \geq \frac{\varepsilon^2}{8LC^2}$.

Proof. Let $\bar{c}_{k,i} = c_{k,i}/C$ and $L_k = \sum_{i=1}^n \ell_{k,i}$. Then

$$\Delta(\boldsymbol{\ell}_{k+1}, \boldsymbol{\ell}_k) = \sum_i \ell_{k+1,i} \ln \frac{\ell_{k+1,i}}{\ell_{k,i}} - L_{k+1} + L_k.$$

By decomposing $\ell_{j,i} = \ell_{j,i}\bar{c}_i + \ell_{j,i}(1 - \bar{c}_i)$ for $j = k, k+1$, and and then applying Log-Sum inequality (e.g., [7]), the r. h. s. of the inequality above is lower bounded by

$$p \ln \frac{p}{q} + (P - p) \ln \frac{P - p}{Q - q} - P + Q,$$

where $p = \bar{\boldsymbol{c}} \cdot \boldsymbol{\ell}_{t+1}$ and $q = \bar{\boldsymbol{c}} \cdot \boldsymbol{\ell}_t$, $P = L_{t+1}$, and $Q = L_t$, respectively. Then,

$$p \ln \frac{p}{q} + (P - p) \ln \frac{P - p}{Q - q} - P + Q = p \ln \frac{p/P}{q/Q} \frac{P}{Q} + (P - p) \ln \frac{1 - p/P}{1 - q/Q} \frac{P}{Q}$$
$$= P\Delta_2(p/P, q/Q) + P \ln \frac{P}{Q},$$

where Δ_2 is called binary relative entropy, i.e., $\Delta_2(a, b) = a \ln \frac{a}{b} + (1 - a) \ln \frac{1-a}{1-b}$. By Pinsker's inequality, $\Delta_2(a, b) \geq (a - b)^2/2$ (see,e.g., [20]). Then, by Proposition 8 in Appendix, the first term of the lower bound is further bounded below as

$$P \ln \frac{P}{Q} - P + Q + \frac{P}{2} \left(\frac{p}{P} - \frac{q}{Q} \right)^2 \geq \frac{(P - Q)^2}{2 \max\{P, Q\}} + \frac{P}{2} \left(\frac{p}{P} - \frac{q}{Q} \right)^2. \quad (4)$$

Then we consider two cases. Suppose that (i) $\frac{P}{Q}q \geq q + \frac{\varepsilon}{2C}$. This assumption is equivalent to the condition that $P \geq Q + \frac{\varepsilon Q}{2qC}$. So, the first term in (4) is bounded below by

$$\frac{1}{2 \max\{P, Q\}} \left(\frac{\varepsilon Q}{2qC} \right)^2 \geq \frac{\varepsilon^2}{8LC^2},$$

where in the last inequality holds since $q \leq Q$ and $P, Q \leq L$.

Otherwise, it holds that (ii) $\frac{P}{Q}q \leq q + \frac{\varepsilon}{2C}$. This condition implies that $\frac{q}{Q} + \frac{\varepsilon}{2PC} \leq \frac{p}{P}$. So, the second term is at least $\frac{P}{2} \left(\frac{\varepsilon}{2PC} \right)^2 = \frac{\varepsilon^2}{8PC^2} \geq \frac{\varepsilon^2}{8LC^2}$. $\square$

Proposition 6 *For each $k = 1, \ldots, K - 1$, $\Delta(\boldsymbol{\ell}_k, \boldsymbol{\ell}_1) \leq O(L \ln Ln)$.*

Proof.

$$\Delta(\boldsymbol{\ell}, \boldsymbol{\ell}_1) = \sum_i^n \ell_i \ln \frac{\ell_i}{L} \frac{L}{1/n} - L + 1 \leq \sum_i^n \ell_i \ln \frac{\ell_i}{L} + L \ln Ln + 1 \leq L \ln Ln + 1.$$

where the inequalities hold since $\sum_i \ell_i \leq L$ and $\ln \frac{\ell_i}{L} \leq 0$. $\square$

Theorem 7 *1. Given a point $\boldsymbol{x} \in \mathcal{P}$, MBB outputs a convex combination $\boldsymbol{\lambda}$ over $\mathcal{C}$ such that*

$$\sum_{\boldsymbol{c} \in \mathcal{C}} \lambda_{\boldsymbol{c}} c_i \leq (\alpha + \varepsilon) x_i \ (i = 1, \ldots, n).$$

2. MBB terminates after

$$K \leq \frac{8L^2 C^2 \ln Ln}{\varepsilon^2} + 2$$

iterations.

Proof. (i) The algorithm ensures that the problem (3) over $\mathcal{C}_K$ is infeasible. So, if we solve the dual problem (2) with the restricted set $\mathcal{C}_K$, its solution $(\boldsymbol{\ell}_K^*, \gamma_K^*)$ must satisfy that $\gamma_K^* \leq \hat{\gamma}_K$. Note that, by the property of the approximation algorithm,

$$\hat{\gamma}_K = \max_{k=1,\ldots,K} \boldsymbol{c}_k \cdot \boldsymbol{\ell}_k + \varepsilon \leq \alpha \min_{\boldsymbol{x}' \in \mathcal{P}} \boldsymbol{x}' \cdot \boldsymbol{\ell}_{k^*} + \varepsilon \leq \alpha \boldsymbol{x} \cdot \boldsymbol{\ell}_{k^*} + \varepsilon \leq \alpha + \varepsilon,$$

where $k^* = \arg\max_{k=1,\ldots,K} \boldsymbol{c}_k \cdot \boldsymbol{\ell}_k$. Finally, the corresponding primal problem over $\mathcal{C}_K$ has an optimal solution $(\boldsymbol{\lambda}_K^*, \beta_K^*)$ such that $\beta_K^* = \gamma_K^*$ by duality, which completes the proof of the first claim.

(ii)By Lemma 2 for $k = 1, \ldots, K - 2$ and summing them up, we have

$$\Delta(\boldsymbol{\ell}_{K-1}, \boldsymbol{\ell}_1) - \Delta(\boldsymbol{\ell}_2, \boldsymbol{\ell}_1) \geq \sum_{k=1}^{K-2} \left(\Delta(\boldsymbol{\ell}_{k+1}, \boldsymbol{\ell}_k) \right). \tag{5}$$

By Lemma 3, the right hand side of (5) is bounded as

$$\sum_{k=1}^{K-2} \left(\Delta(\boldsymbol{\ell}_{k+1}, \boldsymbol{\ell}_k) \right) \geq (K - 2) \frac{\varepsilon^2}{8LC^2}. \tag{6}$$

Combining Proposition 6 and inequalities (5) and (6), we have

$$(K - 2) \frac{\varepsilon^2}{8LC^2} \leq \Delta(\boldsymbol{\ell}_{K-1}, \boldsymbol{\ell}_1) - \Delta(\boldsymbol{\ell}_2, \boldsymbol{\ell}_1) \leq \Delta(\boldsymbol{\ell}_{K-1}, \boldsymbol{\ell}_1),$$

where the last inequality holds since the unnormalized relative entropy is non-negative. Rearranging this inequality, we complete the proof. □

How to remove the assumption on L So far, we are assuming that $L_\varepsilon \leq L$ for some $L > 1$ which is known. We can remove this assumption by a simple doubling method. Let $L_m = 2^{m-1}$. At each trial $m = 1, \ldots,$ we run MBB with $L = L_m$. Then, we check if the 1-norm of the dual solution of problem (2) over C_K is less than L_m. If the 1-norm is strictly less than L_m, we are done. Otherwise, we let $L = L_{m+1}$ and try this process again. It can be easily verified that the total number of iteration is still $O(L_\varepsilon^2 C^2 \ln(L_\varepsilon n)/\varepsilon^2)$.

Time complexity of MBB per iteration is that of convex and linear programs with n variables and $O(\ln n/\varepsilon^2)$ linear constraints, which are solved in polynomial time in n and $1/\varepsilon$. Later, we show that MBB is much faster than the metarounding based on the ellipsoid method in the next section.

6   Experiments

We compare performances of two metarounding algorithms, that of the ellipsoid method [4] and MBB on artificial data. Our experiment is performed on a server with four cores of Intel Xeon CPU X5560 2.80GHz and a memory of 198G bytes. We implement programs using Matlab with Optimization Toolbox. We solve the convex program involved in MBB by sequential quadratic programming.

We generate an artificial data set of set cover instances in the following way. The data set consists of m items and n sets of items. We first add random perturbation to the instances by setting so that for each instance i and each set j, the set j includes the instance i with probability p. Then we fix k "relevant" sets which covers whole the instances. For each instance i, we randomly choose a set among k relevant sets, so that the set covers the instance i. In our experiments, we set $m = 100$, $k = 0.2n$, $p = 0.2$, $n = 10, 50, 100, 200, 500, 1000$, respectively.

We use the simple LP-relaxation based set covering algorithm using deterministic rounding by Hochbaum [12, 22]. The algorithm has a f-approximation guarantee for the LP solution, where $f = \max_{i=1,\dots,m} f_i$ and f_i is the number of sets covering the instance i. The algorithm works as follows. First, the set covering problem is formulated as an integer program. Then, the algorithm solves the LP-relaxation of the problem. Finally, the algorithm rounds the solution $\boldsymbol{p}$ of LP and get the integer solution $\boldsymbol{x}$ by setting $x_i = 1$ if and only if $p_i \geq 1/f$. Note that , one can show that this rounding process is indeed a metarounding. So, we also consider a modification of the algorithm which does not seem to be a metarounding. Our modification is simple. After obtaining the integer solution $\boldsymbol{x}$, we sort each element x_i in the descending order of its associated loss ℓ_i and get $\tilde{\boldsymbol{x}}$. Then, for each $j = 1, \dots, n$, we remove the set j from $\tilde{\boldsymbol{x}}$ as long as the modified vector still represents a set cover.

We generate an internal point $\boldsymbol{x} \in \mathcal{P}$ in the following way. For a random cost vector $\boldsymbol{\ell} \in [0,1]^n$, we solve the offline set cover problem by using the set covering algorithm above and get a cover $\boldsymbol{c} \in \mathcal{C}$. We repeat this process for 20 iterations and get the average vector of obtained covers.

Given an internal point $\boldsymbol{x} \in \mathcal{P}$, we run metarounding algorithms. For the ellipsoid method, we set $R = n^2$ and $\varepsilon = 0.01$. For MBB, we set $\varepsilon = 0.01$ as well.

Fig. 1 shows the computation times (left) and numbers of iterations (right) of metarounding algorithms when we increase the number n of sets. As can be seen in Fig. 1, MBB runs about 10^2 or 10^3 times faster than the ellipsoid method. Further, since MBB runs in much fewer iterations, MBB tends to produce much more sparse convex combination of concepts than the ellipsoid method.

Then, we compare the actual approximation ratio β obtained by metarounding algorithms. For the same data sets, we plot the approximation ratios in Fig. 2. In addition, we also plot the approximation ratio obtained by MBB with our modified set covering algorithm. MBB achieves better approximation ratios than the ellipsoid method. By using the modified algorithm, MBB further gains better ratios than MBB with Hochbaum's original algorithm. Therefore MBB can take advantage of the situation where actual the approximation ratio of the algorithm is better than theoretically guaranteed.
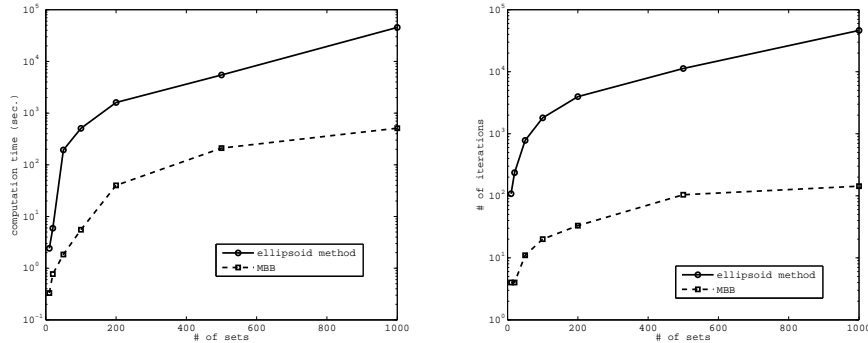
Fig. 1. Computation times (left, CPU time in seconds) and numbers of iterations (right) of metarounding algorithms.

7   Conclusion

In this paper, we propose algorithms for online combinatorial prediction using metarounding algorithms. Also, we show an efficient construction method of metarounding algorithms using a relaxation-based approximation algorithm as an oracle. Our algorithm is adaptive in the sense that it does not require the explicit knowledge on the approximation ratio of the approximation algorithm . Also, computation time of our algorithms at each trial do not depend on T, unlike previous methods.

Acknowledgements

References

1. N. Ailon. Aggregation of Partial Rankings, p-Ratings and Top-m Lists. *Algorithmica*, 57(2):284–300, 2008.
2. N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. *Journal of the ACM*, 55(5), 2008.
3. J.-Y. Audibert, S. Bubeck, and G. Lugosi. Minimax Policies for Combinatorial Prediction Games. In *Proceedings of the 24th Annual Conference on Learning Theory (COLT 2011)*, pages 107–132, 2011.
4. D. R. Carr and S. Vempala. Randomized metarounding. *Random Structure and Algorithms*, 20(1):343–352, 2002.
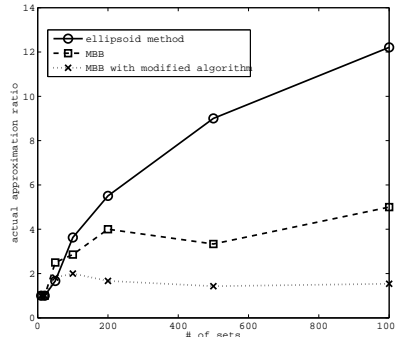
Fig. 2. Actual approximation ratios obtained by metarounding algorithms, the ellipsoid method and MBB with Hochbaum's original set covering algorithm and MBB with our modified set covering algorithm.

5. N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games.* Cambridge University Press, 2006.
6. N. Cesa-Bianchi and G. Lugosi. Combinatorial Bandits. In *Proceedings of the 22nd Conference on Learning Theory (COLT 2009)*, 2009.
7. T. Cover and J. Thomas. *Elements of Information Theory.* Wiley, 1991.
8. Y. Freund and R. E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
9. M. Goemans and D. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7:656–666, 1994.
10. E. Hazan. The convex optimization approach to regret minimization. In Suvrit Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*, chapter 10, pages 287–304. MIT Press, 2011.
11. D. P. Helmbold and M. K. Warmuth. Learning Permutations with Exponential Weights. *Journal of Machine Learning Research*, 10:1705–1736, 2009.
12. D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11:555–556, 1982.
13. S. Kakade, A. T. Kalai, and L. Ligett. Playing games with approximation algorithms. *SIAM Journal on Computing*, 39(3):1018–1106, 2009.
14. A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.
15. W. M. Koolen, M. K. Warmuth, and J. Kivinen. Hedging Structured Concepts. In *Proceedings of the 23rd Conference on Learning Theory (COLT 2010)*, pages 93–105, 2010.
16. R. E. Schapire and Y. Freund. *Boosting: Foundation and Algorithms.* MIT Press, 2012.
17. A. Schrijver. *Theory of linear and integer programming.* Wiley, 1998.
18. A. Srinivasan. Improved approximations of packing and covering problems. In *27th ACM Symposium on the Theory of Computing*, pages 268–276, 1995.

19. D. Suehiro, K. Hatano, S. Kijima, E. Takimoto, and K. Nagano. Online Prediction under Submodular Constraints. In *Proceedings of 23th Annual Conference on Learning Theory (ALT 2012)*, pages 260–274, 2012.
20. M. Warmuth, K. Glocer, and G. Rätsch. Boosting Algorithms for Maximizing the Soft Margin. In *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, pages 1585–1592, 2007.
21. M. K. Warmuth and D. Kuzmin. Randomized Online PCA Algorithms with Regret Bounds that are Logarithmic in the Dimension. *Journal of Machine Learning Research*, 9:2287–2320, 2008.
22. D. P. Williamson and D. B. Shmoys. *The Design of Approximation ALgorithms*. Cambridge University Press, 2011.
23. S. Yasutake, K. Hatano, S. Kijima, E. Takimoto, and M. Takeda. Online Linear Optimization over Permutations. In *Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC 2011)*, pages 534–543, 2011.
24. S. Yasutake, K. Hatano, E. Takimoto, and M. Takeda. Online Rank Aggregation. In *Proceedings of 4th Asian Conference on Machine Learning (ACML2012)*, pages 539–553, 2012.
25. M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pages 928–936, 2003.

Appendix

Proposition 8 *For $A, B \geq 0$,*

$$A \ln \frac{A}{B} - A + B \geq \frac{1}{2 \max\{A, B\}} (A - B)^2.$$

Proof. Let $f(x) = x \ln \frac{x}{B} - x + B$. By using Taylor expansion at $x = B$, there exists some B' such that

$$f(x) = f(B) + f'(B)(x - B) + \frac{f''(B')}{2}(x - B)^2 = \frac{1}{2B'}(x - B)^2,$$

where B' satisfies $x < B' < B$ or $B < B' < x$. Since $B' \leq \max\{x, B\}$, we have

$$f(x) \geq \frac{1}{2 \max\{x, B\}}(x - B)^2.$$

By letting $x = A$, we complete the proof.