Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computer and System Sciences 71 (2005) 291–307
JOURNAL OF COMPUTER AND SYSTEM SCIENCES
www.elsevier.com/locate/jcss

Efficient algorithms for online decision problems☆

Adam Kalai[a,*], Santosh Vempala[b]

[a] *Department of Computer Science, Toyota Technological Institute, 1427 E. 60th St., Chicago, IL 60637, USA*
[b] *Massachusetts Institute of Technology, MA, USA*

Received 20 February 2004; received in revised form 1 October 2004

Available online 20 December 2004

Abstract

In an online decision problem, one makes a sequence of decisions without knowledge of the future. Each period, one pays a cost based on the decision and observed state. We give a simple approach for doing nearly as well as the best single decision, where the best is chosen with the benefit of hindsight. A natural idea is to follow the leader, i.e. each period choose the decision which has done best so far. We show that by slightly perturbing the totals and then choosing the best decision, the expected performance is nearly as good as the best decision in hindsight. Our approach, which is very much like Hannan's original game-theoretic approach from the 1950s, yields guarantees competitive with the more modern exponential weighting algorithms like Weighted Majority.

More importantly, these follow-the-leader style algorithms extend naturally to a large class of structured online problems for which the exponential algorithms are inefficient.

Keywords: Online algorithms; Hannan's algorithm; Optimization; Decision theory

1. Introduction

In an online decision problem, one has to make a sequence of decisions without knowledge of the future. One version of this problem is the case with n experts (corresponding to decisions). Each period, we pick one expert and then observe the *cost* $\in [0, 1]$ for each expert. Our cost is that of the chosen expert.

☆ An extended abstract of this paper appeared at COLT 2003 [16].

* Corresponding author.
E-mail addresses: kalai@tti-c.org (A. Kalai)
URLs: http://people.cs.uchicago.edu/∼kalai, http://www-math.mit.edu/∼vempala.

0022-0000/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcss.2004.10.016

Our goal is to ensure that our total cost is not much larger than the minimum total cost of any expert. This is a version of the *predicting from expert advice* problem. [1] Exponential weighting schemes for this problem have been discovered and rediscovered in many areas [12]. Even in learning, there are too many results to mention (for a survey, see [4]).

The following different approach can also be used. We add a random perturbation to the total cost so far of each expert e each period, and then choose the expert of minimal cost.

- Follow the perturbed leading expert: On each period $t = 1, 2, \ldots,$
  1. For each expert $e \in \{1, 2, \ldots, n\}$, pick $p_t[e] \geqslant 0$ from exp. distribution $d\mu(x) = \varepsilon e^{-\varepsilon x}$.
  2. Choose expert with minimal $c_t[e] - p_t[e]$, where $c_t[e] =$ total cost of expert e so far.

The above algorithm is quite similar to Hannan's original algorithm [2] [14] (which gave additive bounds). Following the perturbed leader gives small regret relative to the best expert,

$$E[cost] \leqslant (1+\varepsilon)(\text{min cost in hindsight}) + \frac{O(\log n)}{\varepsilon}. \tag{1}$$

While the algorithm and guarantees are similar to randomized versions of Weighted Majority, the algorithm can be efficiently generalized to a large class of problems. This problem is discussed in more detail in Section 2.

Next consider the more structured problem of *online shortest paths* [28], where one has a directed graph and a fixed pair of nodes (s, t). Each period, one has to pick a path from s to t, and then the times on all the edges are revealed. The per-period cost is the sum of the times on the edges of the chosen path.

With bounded times, one can ignore the structure in this problem and view it as an expert problem where each path is an independent expert. While the number of paths may be exponential in the size of the graph, the above bound only depends logarithmically on the number of experts. However, the runtime of an experts algorithm for this problem would be exponential in the size of the problem.

As is common for such problems with nice structure, a clever and efficient algorithm has been designed for this problem [28]. Their approach was to mimic the distribution over paths that would be chosen by the exponential algorithm, but with efficient implicit calculations. Similar algorithms have been designed for several other problems [15,28,27,11,6].

Surprisingly, the natural generalization of following the perturbed leading expert can be applied to all these problems and more, efficiently. In the case of shortest paths,

- Follow the perturbed leading path: On each period $t = 1, 2, \ldots,$
  1. For each edge e, pick $p_t[e]$ randomly from an exponential distribution. (See FPL* in the next section for exact parameters.)
  2. Use the shortest path in the graph with weights $c_t[e] + p_t[e]$ on edge e, where $c_t[e] =$ total time on edge e so far.

As a corollary of Theorem 1.1, with m edges and n nodes

$$E[time] \leqslant (1+\varepsilon)(\text{best time in hindsight}) + \frac{O(mn\ \log n)}{\varepsilon}.$$

[1] A small difference is that we are required to pick a single expert, rather than a weighting on experts.

[2] We are grateful to Sergiu Hart for the pointer to Hannan's algorithm.

As is standard, "best time in hindsight" refers to the minimum total time spent, if one had to use the same path each period, and we are assuming all edge times are between 0 and 1. This is similar to the aforementioned bounds of Takimoto and Warmuth [28].

Before discussing further applications, we describe the general model and theorems that are proven.

1.1. Linear generalization and results

We consider a linear generalization in which we, the decision maker, must make a series of decisions $d_1, d_2, \ldots$, each from a possibly infinite set $\mathcal{D} \subset \mathbb{R}^n$. After the tth decision d_t is made, we observe the state $s_t \in \mathcal{S} \subset \mathbb{R}^n$. There is a cost of $d \cdot s$ for making decision d in state s, so our cost is $\sum d_t \cdot s_t$.

The expert problem can be mapped into this setting as follows: n is the number of experts, the state each period is the observed vector of costs, and choosing expert i corresponds to the decision vector d with a 1 in position i and 0 everywhere else.

For the path problem, n is now the number of edges, the state each period is the vector of observed costs (one per edge), and a decision to take a path corresponds to a $\{0, 1\}$-vector with 1's in the positions of edges that are on the path.

Thus, our goal is to have a total cost $\sum d_t \cdot s_t$ not far from $\min_{d\in\mathcal{D}} \sum d \cdot s_t$, the cost of the best offline decision, if one had to choose a single decision in hindsight. (It is impossible, in general, to be competitive with the best dynamic strategy that may change decisions each period. Such a comparison leads to large regret.) Let M be a function that computes the best single decision in hindsight, $\arg\min_{d\in\mathcal{D}} \sum d \cdot s_t$. Because costs are additive, it suffices to consider M as a function of total state vectors, $M : \mathbb{R}^n \to \mathcal{D}$,

$$M(s) = \arg\min_{d\in\mathcal{D}} d \cdot s.$$

In the case of experts, M simply finds an expert of minimum cost given the total cost vectors so far. In the case of paths, M finds the shortest path in the graph with weights which are the total times on each edge. (Note, for ease of analysis, we are not distinguishing between actual decisions, i.e. experts or paths, and their representation in $\mathbb{R}^n$.)

We will give several more examples that can be mapped into this linear model. On the surface, it resembles a convex optimization problem, however, instead of requiring $\mathcal{D}$ to be convex, we only assume that the optimizer M can be computed efficiently. [3]

Given such a linear problem of dimension n, and given a black-box algorithm for computing M, we can give an online algorithm whose cost is near the minimum offline cost,

$$\text{min-cost}_T = \min_{d\in\mathcal{D}} \sum_1^T d \cdot s_t = M(s_1 + s_2 + \cdots + s_T) \cdot (s_1 + s_2 + \cdots + s_T).$$

The additive and multiplicative versions of Follow the Perturbed Leader (FPL) are as follows.

- **FPL**(ε): On each period t,
  1. Choose p_t uniformly at random from the cube $\left[0, \frac{1}{\varepsilon}\right]^n$.
  2. Use $M(s_1 + \ldots + s_{t-1} + p_t)$.

[3] This is not a restrictive assumption because efficient $(1+\varepsilon)$ online computation implies efficient $(1+\varepsilon)$ offline approximation of M by standard techniques [21]. What we show is the converse: how to use efficient offline algorithms for the online problem.

- **FPL*** (ε): On each period t,
  1. Choose p_t at random according to the density $d\mu(x) \propto e^{-\varepsilon|x|_1}$. (Independently for each coordinate, choose $\pm(r/\varepsilon)$ for r from a standard exponential distribution.)
  2. Use $M(s_1 + \ldots + s_{t-1} + p_t)$.

Motivation for these algorithms can be seen in a simple two-expert example. Suppose the cost sequence was $(0, \frac{1}{2})$ followed by alternating costs of $(1, 0)$ and $(0, 1)$. Then, following the leader (without perturbations) always incurs a cost of 1, while each expert incurs a cost of about $t/2$ over t periods. With n experts, the situation is even worse—*any* deterministic algorithm can be forced to have a cost of t over t periods (each time only the selected expert incurs a cost of 1) while the best expert has a cost of at most t/n. By adding perturbations, the algorithm becomes less predictable, one the one hand. On the other hand, it takes longer to adapt to a setting where one expert is clearly better than others. This tradeoff is captured by the following theorem, stated in terms of the following parameters.[4] Here the L^1 norm of a vector $x \in \mathbb{R}^n$ is $|x|_1 = \sum_1^n |x_i|$.

$$\begin{aligned}\text{(diameter)} D &\geqslant |d - d'|_1, \quad \text{for all } d, d' \in \mathcal{D},\\ R &\geqslant |d \cdot s|, \quad \text{for all } d \in \mathcal{D}, s \in \mathcal{S},\\ A &\geqslant |s|_1, \quad \text{for all } s \in \mathcal{S}.\end{aligned}$$

Theorem 1.1. *Let $s_1, s_2, \ldots, s_T \in S$ be a state sequence.* (*a*) *Running FPL with parameter $\varepsilon \leqslant 1$ gives,*

$$E[cost of FPL(\varepsilon)] \leqslant min\text{-}cost_T + \varepsilon RAT + \frac{D}{\varepsilon},$$

(*b*) *For nonnegative $\mathcal{D}, \mathcal{S} \subset \mathbb{R}^n_+$, FPL* gives,*

$$E[cost of FPL*(\varepsilon/2A)] \leqslant (1+\varepsilon) min\text{-}cost_T + \frac{4AD(1+\ln n)}{\varepsilon}.$$

Of course, it makes sense to state the bounds in terms of the minimizing values of ε, as long as T or min-cost$_T$ are known in advance, giving

$$\begin{aligned}E[\text{cost of FPL}(\sqrt{D/RAT})] &\leqslant \text{min-cost}_T + 2\sqrt{DRAT},\\ E[\text{cost of FPL*}(\varepsilon_1)] &\leqslant \text{min-cost}_T + 4\sqrt{(\text{min-cost}_T)AD(1+\ln n)} + 4AD(1+\ln n),\end{aligned}$$

where $\varepsilon_1 = \min(1/2A, \sqrt{D(1+\ln n)/A(\text{min-cost}_T)})$. Even if they are not known in advance, simple ε-halving tricks can be used to get nearly the same guarantees.

1.2. Further applications and algorithms

For the *tree update problem*, it seems complicated to efficiently implement the weighted majority style algorithms, and no efficient $(1+\varepsilon)$-algorithms were known. This problem is a classic online problem [26] introduced by Sleator and Tarjan with Splay Trees, around the same time as they introduced the list update problem [25]. In the tree update problem, one maintains a binary search tree over n items in

[4] Note that the parameters need only hold for "reasonable" decisions that an optimal offline decision might actually make, e.g. $D \geqslant |M(s) - M(s')|_1 \forall s, s'$ would suffice (we do not need to consider the cost of paths that visit a node twice).

the face of an unknown sequence of accesses to these items. For each access, i.e. lookup, the cost is the number of comparisons necessary to find the item, which is equal to its depth in the tree.

One could use FPL for this problem as well. This would maintain frequency counts for each item in the tree, and then before each access it would find the best tree given these frequencies plus perturbations (which can be computed in $O(n^2)$ time using dynamic programming). But doing so much computation and so many tree rotations, just to prepare for a lookup, would be taking the online analysis model to an absurd extreme. Instead, we give a way to achieve the same effect with little computation and few updates to the tree:

- Follow the lazy leading tree (N):
  1. For $1 \leqslant i \leqslant n$, let $s_i := 0$ and choose v_i randomly from $\{1, 2, \ldots, N\}$.
  2. Start with the best tree as if there were v_i accesses to node i.
  3. After each access, set a to be the accessed item, and:
  (a) $s_a := s_a + 1$.
  (b) If $s_a \geqslant v_a$ then
     i. $v_a := v_a + N$.
     ii. Change trees to the best tree as if there were v_i accesses to node i.

Over T accesses, for $N = \sqrt{T/n}$, one gets the following *static* bounds [5] as a corollary of Lemma 1.2 and Theorem 1.1,

$$E[\text{cost of lazy trees}] \leqslant (\text{cost of best tree}) + 2n\sqrt{nT}.$$

Because any algorithm must pay at least 1 per acccess, the above additive regret bound is even stronger than a multiplicative $(1+\varepsilon)$-competitive bound, i.e. $T \leqslant (\text{cost of best tree})$. In contrast, Splay Trees have a guarantee of $3\log_2 3 \times (\text{cost of best tree})$ plus an additive term, but they have other desirable properties. This algorithm has what Blum et. al. call *strong static optimality* [6]. For the simpler *list update problem*, they presented both implicit exponential and follow the perturbed leader types of algorithms. Theirs was the original motivation for our work, and they were also unaware of the similarity to Hannan's algorithm.

The key point here is that step (ii) is executed with probability at most $1/N$, so one expects to update only $\sqrt{nT}$ times over T accesses. Thus the computational costs and movement costs, which he have thus far ignored, are small. Corresponding to FPL and FPL*, which call the black-box M once each period, we give general lazy algorithms Follow the Lazy Leader, FLL and FLL*, that have exactly the same performance guarantees, but only call the black box with probability εA each period, and thus are extremely efficient. Since εT is typically $O(\sqrt{T})$ (ignoring n), this means that on a sequence of length T we only need to do $O(\sqrt{T})$ updates. This is especially important if there is a *movement cost* to change trees. [6] In our case, this cost becomes negligible. The slight disadvantage of the lazy algorithms is that they only work against an adversary that is oblivious to their random choices.

Lemma 1.2. *For any fixed sequence of states $s_1, s_2, \ldots$, $FPL(\varepsilon)$ and $FLL(\varepsilon)$ (also FPL* and FLL*) have identical expectations on each period t. However, the probability of $FLL(\varepsilon)$ (or $FLL*(\varepsilon)$) performing an update is at most εA.*

[5] We do not give dynamic guarantees and our results do not apply to the dynamic optimality conjecture [26].

[6] Similar issues have been addressed in the exponential algorithm literature, however without regard to efficiency.

The *Adaptive Huffman coding* problem [19] is not normally considered as an online algorithm. But it fits naturally into the framework. There, one wants to choose a prefix tree for each symbol in a message, "on the fly" without knowledge of the sequence of symbols in advance. The cost is the length of the encoding of the symbol, i.e. again its depth in the tree. Adaptive Huffman coding is exactly the follow-the-leader algorithm applied to this problem. For a compression problem, however, it is natural to be concerned about sequences of alternating 0s and 1s. Adaptive Huffman coding does not give $(1+\varepsilon)$ guarantees. If the encoder and decoder have a shared random (or pseudorandom) sequence, then they can apply FPL or FLL as well. The details are similar to the tree update problem.

Efficient $(1+\varepsilon)$ algorithms have been designed for *online pruning* of decision trees, decision graphs, and their variants [15,27]. Not surprisingly, FPL* and FLL* will apply.

1.2.1. Online approximation algorithms

An interesting case that does not fit our model is the set of problems where no known efficient algorithm for offline optimality exists. In these cases, we cannot hope to get online $(1+\varepsilon)$ optimality, but it is natural to hope that an efficient α-approximation algorithm could be turned into an efficient online $(1+\varepsilon)\alpha$-competitive algorithm. In general, all we can show is a $(1+\varepsilon)\alpha^T$-competitive algorithm, which is only interesting for α close to 1 (which can be found for many problems such as Euclidean Traveling Salesman Problem [1]).

A sample problem would be an *online max-cut* problem: we have a multigraph and we must choose a cut. The score of a cut is the number of edges crossing the cut (we refer to score instead of cost for maximization problems). In the online version of this linear maximization problem, [7] one edge is added at a time. Without knowledge of the next edge, we must choose a cut, and receive a score of 1 if the edge crosses the cut and 0 otherwise.

In Section 5, we show that our algorithm can be used with approximation algorithms with a certain property, which we call *pointwise approximate*. Some examples include the max-cut algorithm of [13] and the classification algorithm of [18].

A general conversion from offline approximation algorithms to online approximation algorithms would be very interesting.

1.2.2. Online linear optimization

The focus of earlier work [16] was the general problem of online linear optimization. Independently, Zinkevich has introduced an elegant deterministic algorithm for the more general online convex optimization problem [31]. His algorithm is well-suited for convex problems but not for the discrete problems which we focus on here. A natural extension of FPL to a convex set $\mathcal{D}$ would be Follow the Expected Leader (FEL):

- **FEL**(ε, m): On each period t,
  1. Choose $p_t^1, p_t^2, \ldots, p_t^m$ independently and uniformly at random from the cube $\left[0, \frac{1}{\varepsilon}\right]^n$.
  2. Use $\frac{1}{m}\sum_{i=1}^{m} M(s_1 + \cdots + s_{t-1} + p_t^i)$.

[7] To view max-cut as a linear optimization problem, consider a coordinate for each pair of vertices (u, v) . The objective vector c at each coordinate is the number of edges between u and v, and a cut is represented by a $\{0,1\}$ vector with 1s in the coordinates where u and v are on different sides.

For this algorithm, we are assuming that the set of possible decisions is convex so that we may take the average of several decisions. In this case, the expected guarantees can be converted into high-probability guarantees. Formulated another way, FEL applies to the following problem.

Online linear optimization: Given a feasible convex set $\mathcal{D} \subset \mathbb{R}^n$, and a sequence of objective vectors $s_1, s_2, \ldots \in \mathbb{R}^n$, choose a sequence of points $d_1, d_2, \ldots \in \mathcal{D}$ that minimizes $\sum_{t=1}^{T} d_t \cdot s_t$. When choosing d_t, only $s_1, s_2, \ldots s_{t-1}$ are known.

A typical example of such a problem would be a factory that is able to produce a variety of objects (such as chairs and tables), with a convex set of feasible production vectors. Each period, we must decide on how many of each object to produce, and afterwards we are informed of the profit vector. Our goal is to have profit nearly as large as the profit of the best single production vector, if we had to use the same production vector each period.

By linearity of expectation, the expected performance of FEL is equal to the expected performance of FPL. However, as m gets larger, the algorithm becomes more and more deterministic, and the expected guarantees can be converted to high-probability guarantees that hold with larger and larger probabilities.

We refer the reader to [16,31] for a more in-depth study of this problem.

2. Experts problem

We would like to apply our algorithm to the predicting from expert advice problem, where one has to choose a particular expert each period. Here, it would seem that $D = 1$ and $A = n$. This is unfortunate because we need $A = 1$ to get the standard bounds. For the multiplicative case, we can fix this problem by observing that the worst case for our algorithm (and in fact most algorithms) is when each period only one expert incurs cost.[8] Thus we may as well imagine that $A = 1$, and we get the standard $(1+\varepsilon) \times$ (best expert) $+ O(\log n/\varepsilon)$ bounds of Weighted Majority.

To get slightly better bounds, and more importantly, better intuition, one can use the following analysis approach. This is an alternative analysis that applies to many problems, but does not have the full generality of the approach used in the remainder of the paper. First, imagine the algorithm with no perturbations, i.e. $p_1 = p_2 = \cdots = 0$. We can bound its performance in terms of the cost of the best expert, i.e. the leader at the end, and the number of times the leader (so far) changed during the execution:

$$\text{cost of following the leader} \leqslant \text{cost of final leader} + \#\text{ times leader changed}. \tag{2}$$

To see this, note that each time the leader does not change, that means that the cost we incur is the same as the amount min-cost increases by. Each time the leader does change, our cost can increase by at most 1.

Let us now return to the case with perturbations. Without loss of generality, we assume that the perturbations from period to period are the same, i.e. $p_1 = p_2 = \cdots = p_t$. From linearity of expectation, this will not change our expected performance. Equivalently, we pretend that rather than perturbations, we have a period 0 with cost vector $-p_1$. Now, when we refer to the leader, we are including the pretend

[8] Imagine comparing two scenarios, one with one period $s_1 = (a, b)$ and the second with two periods $s_1 = (a, 0)$ and $s_2 = (0, b)$ It is not difficult to see that our cost in the second scenario is larger, because we have more weight on the second expert after the first period. Nevertheless, the cost of the best expert in both scenarios is the same.

period 0 perturbations. We argue that the leader changes infrequently. In particular,

$$E_{p_1}[\#\text{ changes of leader}] \leqslant \varepsilon E_{p_1}[\text{cost of FPL}]. \tag{3}$$

To see this, fix a particular period. Expert i is the leader if and only if the perturbation $p_1[i]$ of expert i is sufficiently large. In particular, i is the leader iff $p_1[i] \geqslant v$ for some value v, which depends on the total cost of the experts and the perturbations of the other experts. Whatever v is, we can bound the probability that i remains leader. If i incurs cost c, then i certainly remains leader if $p_1[i] > v + c$, because this means i was already a leader by more than c.

The exponential density from which $p_1[i]$ is chosen, namely $\varepsilon e^{-\varepsilon x}$, has the following property:

$$\begin{aligned} P[p_1[i] > v + c \mid p_1[i] \geqslant v] &= \frac{\int_{v+c}^{\infty} \varepsilon e^{-\varepsilon x} dx}{\int_{v}^{\infty} \varepsilon e^{-\varepsilon x} dx} \\ &= e^{-\varepsilon c} \\ &\geqslant 1 - \varepsilon c. \end{aligned}$$

In other words, given that expert i is leader, the probability it does not remail leader is at most εc. On the other hand, given that expert i is leader, the cost is c. Therefore, the probability of changing leader is at most ε times the expected cost. Summing over periods establishes (3). Applying (2) to the modified sequence, and using (3) gives:

$$E[\text{cost of FPL}] \leqslant \text{cost of final leader} + \varepsilon(\text{cost of FPL}).$$

However, the cost of the final leader is not exactly the same as the cost of the best expert, because we have added perturbations. This makes sense, because there must be a cost to adding perturbations. Say the truly best expert was expert b. Like any fixed expert, it has expected perturbation $E[[p_1[b]] = 1/\varepsilon$. Say the final leader is expert j. Then

$$\text{cost of final leader} \leqslant \text{min-cost}_T + p_1[j] - p_1[b].$$

In other words $p_1[j] - p_1[b]$ is an upper-bound on how much we could have deceived ourselves. But $E[p_1[j]] \leqslant E[\max_i p_1[i]] \leqslant (1 + \ln n)/\varepsilon$. In a moment, we will argue this last inequality. But, taking it for granted, this gives a final bound of

$$E[\text{cost of FPL}](1 - \varepsilon) \leqslant \text{min-cost}_T + \frac{\ln n}{\varepsilon}.$$

These bounds are comparable, and in the worst case, only slightly larger by a constant in front of $\ln n$ term than the bounds for randomized weighted majority.

More importantly, the analysis also offers one explanation of the source of the tradeoff between the $(1 + \varepsilon)$ and $1/\varepsilon$ terms. The more initial randomness, the less likely any sequence is to make us switch (less predictable). However, the more randomness we add, the more we are deceiving ourselves.

Another interesting point that comes from this analysis is the use of fresh randomness each period. In terms of expectation, for any fixed cost sequence, it does not matter whether we use fresh randomness or not. However, if we did not use fresh randomess, i.e. $p_1 = p_2 = \cdots$, an *adaptive* adversary that can choose cost vectors based on our previous decisions (but not on our private coin flips) could figure out what our perturbations p_1 were and give us large regret. Rerandomizing each period makes our algorithm have low regret against adaptive adversaries as well.

Finally, it remains to show that the expected maximum perturbation is at most $(1 + \log n)/\varepsilon$. To see this, note by scaling that it is $1/\varepsilon$ times the expected maximum of n standard exponential distributions with mean 1. Note that the expectation of a nonnegative random variable X is $E[X] = \int_0^\infty \Pr[X \geqslant x]\,dx$. Consider $x_1, x_2, \ldots, x_n$, each drawn independently from the standard exponential distribution e^{-x}. The expected maximum is

$$\begin{aligned}\int_0^\infty \Pr[\max(x_1, \ldots, x_n) \geqslant x]\,dx &\leqslant \int_0^{\ln n} \Pr[\max(x_1, \ldots, x_n) \geqslant x]\,dx + \int_{\ln n}^\infty ne^{-x}\,dx.\\ &\leqslant \ln(n) + 1.\end{aligned}$$

This implies that for scaled exponential distributions, the expected maximum is at most $(1 + \ln n)/\varepsilon$.

3. Additive analysis

We first analyze FPL, proving Theorem 1.1 (a). Hindsight gives us an analysis that is vastly simpler than Hannan's. For succinctness, we use the notational shortcut

$$s_{1:t} = s_1 + s_2 + \cdots + s_t.$$

We will now bound the expected cost of FPL on any particular sequence of states.

The idea is to first analyze a version of the algorithm where we use $M(s_{1:t})$ on period t (instead of $M(s_{1:t-1})$). Of course, this is only a hypothetical algorithm since we do not know s_t in advance. But, as we show, this "be the leader" algorithm has no regret. The point of adding randomness is that it makes following the leader not that different than being the leader. The more randomness we add, the closer they are (and the smaller the εRAT term). However, there is a cost to adding randomness. Namely, a large amount of randomness may make a worse choice seem better. This accounts for the D/ε term. The analysis is relatively straightforward.

First, we see by induction on T that using $M(s_{1:t})$ on day t gives 0 regret,

$$\sum_{t=1}^{T} M(s_{1:t}) \cdot s_t \leqslant M(s_{1:T}) \cdot s_{1:T}. \tag{4}$$

For $T = 1$, it is trivial. For the induction step from T to $T + 1$,

$$\begin{aligned}\sum_{t=1}^{T+1} M(s_{1:t}) \cdot s_t &\leqslant M(s_{1:T}) \cdot s_{1:T} + M(s_{1:T+1}) \cdot s_{T+1}\\ &\leqslant M(s_{1:T+1}) \cdot s_{1:T} + M(s_{1:T+1}) \cdot s_{T+1}\\ &= M(s_{1:T+1}) \cdot s_{1:T+1}.\end{aligned}$$

Eq. (4) shows that if one used $M(s_{1:t})$ on period t, one would have no regret. Essentially, this means that the hypothetical "be the leader" algorithm would have no regret. Now consider adding perturbations. We first show that perturbations do not hurt too much.

Lemma 3.1. *For any state sequence $s_1, s_2, \ldots$, any $T > 0$, and any vectors $p_0 = 0, p_1, p_2, \ldots, p_T \in \mathbb{R}^n$,*

$$\sum_{t=1}^{T} M(s_{1:t} + p_t) \cdot s_t \leqslant M(s_{1:T}) \cdot s_{1:T} + D \sum_{t=1}^{T} |p_t - p_{t-1}|_\infty.$$

Proof. Pretend the cost vector s_t on period t was actually $s_t + p_t - p_{t-1}$. Then the cumulative $s_{1:t}$ would actually be $s_{1:t} + p_t$, by telescoping. Making these substitutions in (4) gives

$$\begin{aligned}
\sum_{t=1}^{T} M(s_{1:t} + p_t) \cdot (s_t + p_t - p_{t-1}) &\leqslant M(s_{1:T} + p_T) \cdot (s_{1:T} + p_T) \\
&\leqslant M(s_{1:T}) \cdot (s_{1:T} + p_T) \\
&= M(s_{1:T}) \cdot s_{1:T} + \sum_{t=1}^{T} M(s_{1:T}) \cdot (p_t - p_{t-1}).
\end{aligned}$$

$$\begin{aligned}
\sum_{t=1}^{T} M(s_{1:t} + p_t) \cdot s_t \leqslant M(s_{1:T}) \cdot s_{1:T} + \sum_{t=1}^{T} (M(s_{1:T}) - M(s_{1:t} + p_t)) \\
\cdot (p_t - p_{t-1})
\end{aligned}$$

Recall that $D \geqslant |d - d'|_1$ for any decision vectors d, d'. Also note that $u \cdot v \leqslant |u|_1 |v|_\infty$. □

Proof of Theorem 1.1. (a) In terms of expected performance, it wouldn't matter whether we chose a new p_t each day or whether $p_t = p_1$ for all $t > 1$. Applying Lemma 3.1 to the latter scenario gives,

$$\sum_{t=1}^{T} M(s_t + p_1) \cdot s_t \leqslant M(s_{1:T}) \cdot s_{1:T} + D|p_1|_\infty \leqslant M(s_{1:T}) \cdot s_{1:T} + \frac{D}{\varepsilon}. \tag{5}$$

Thus, it just remains to show that the expected difference between using $M(s_{1:t-1} + p_t)$ instead of $M(s_{1:t} + p_t)$ on each period t is at most $\varepsilon A R$.

Key idea: we notice that the *distributions* over $s_{1:t-1} + p_t$ and $s_{1:t} + p_t$ are similar. In particular, they are both distributions over cubes. If the cubes were identical, i.e. $s_{1:t-1} = s_{1:t}$, then $E[M(s_{1:t-1} + p_t) \cdot s_t] = E[M(s_{1:t} + p_t) \cdot s_t]$. If they overlap on a fraction f of their volume, then we could say,

$$E[M(s_{1:t-1} + p_t) \cdot s_t] \leqslant E[M(s_{1:t} + p_t) \cdot s_t] + (1 - f)R$$

This is because on the fraction that they overlap, the expectation is identical, and on the fraction that they do not overlap, one can only be R larger, by the definition of R. By Lemma 3.2 following this proof, $1 - f \leqslant \varepsilon |s_t|_1 \leqslant \varepsilon A$. □

Lemma 3.2. *For any $v \in \mathbb{R}^n$, the cubes $\left[0, \frac{1}{\varepsilon}\right]^n$ and $v + \left[0, \frac{1}{\varepsilon}\right]^n$ overlap in at least a $(1 - \varepsilon|v|_1)$ fraction.*

Proof. Take a random point $x \in \left[0, \frac{1}{\varepsilon}\right]^n$. If $x \notin v + \left[0, \frac{1}{\varepsilon}\right]^n$, then for some i, $x_i \notin v_i + [0, \frac{1}{\varepsilon}]$, which happens with probability at most $\varepsilon|v_i|$ for any particular i. By the union bound, we are done. □

If we know T in advance, it makes sense to use a setting of ε which minimizes the guarantees from Theorem 1.1. As mentioned, we can get bounds nearly as good, without such knowledge, by standard ε-halving techniques. Alternatively, we can follow Hannan's lead and use gradually increasing perturbations:

- **Hannan**(δ): On each period t,
  1. Choose p_t uniformly at random from the cube $\left[0, \frac{\sqrt{t}}{\delta}\right]^n$.
  2. Use $M(s_{1:t-1} + p_t)$.

Using a similar argument, it is straightforward to show:

Theorem 3.3. *For any state sequence $s_1, s_2, \ldots$, after any number of periods $T > 0$,*

$$E[cost\,of\,Hannan(\delta)] \leqslant M(s_{1:T}) \cdot s_{1:T} + 2\delta RA\sqrt{T} + \frac{D\sqrt{T}}{\delta}.$$

Proof. WLOG we may choose $p_t = (\sqrt{t})p_1$ because $\frac{p_t}{\sqrt{t}}$ is identically distributed, for all t, and we are only bounding the expectation. Applying Lemma 3.1 to this scenario gives,

$$\sum_{t=1}^{T} M(s_{1:t} + \sqrt{t}\,p_1) \cdot s_t \leqslant M(s_{1:T}) \cdot s_{1:T} + D|p_1|_\infty \sum_{t=1}^{T} (\sqrt{t} - \sqrt{t-1}).$$

The last term is at most $D(1/\delta)\sqrt{T}$.

Now, $M(s_{1:t-1} + p_t)$ and $M(s_{1:t} + p_t)$ are distributions over cubes of side $\sqrt{t}/\delta$. By Lemma 3.2, they overlap in a fraction that is at least $1 - |s_t|_1\delta/\sqrt{t} \geqslant 1 - A\delta/\sqrt{t}$. On this fraction, their expectation is identical so,

$$E[(M(s_{1:t-1} + p_t) - M(s_{1:t} + p_t)) \cdot s_t] \leqslant \frac{\delta RA}{\sqrt{t}}.$$

Thus we have shown,

$$E\left[\sum_{t=1}^{T} M(s_{1:t-1} + p_t) \cdot s_t\right] \leqslant M(s_{1:T}) \cdot s_{1:T} + \frac{D\sqrt{T}}{\delta} + \sum_{t=1}^{T} \frac{\delta RA}{\sqrt{t}}.$$

Finally, straightforward induction shows $\sum_{t=1}^{T} \frac{1}{\sqrt{t}} \leqslant 2\sqrt{T}$. □

3.1. Follow the lazy leader

Here, we introduce an algorithm called Follow the Lazy Leader or FLL, with the following properties:

- FLL is equivalent to FPL in terms of expected cost.
- FLL rarely calls the oracle M.
- FLL rarely changes decision from one period to the next.

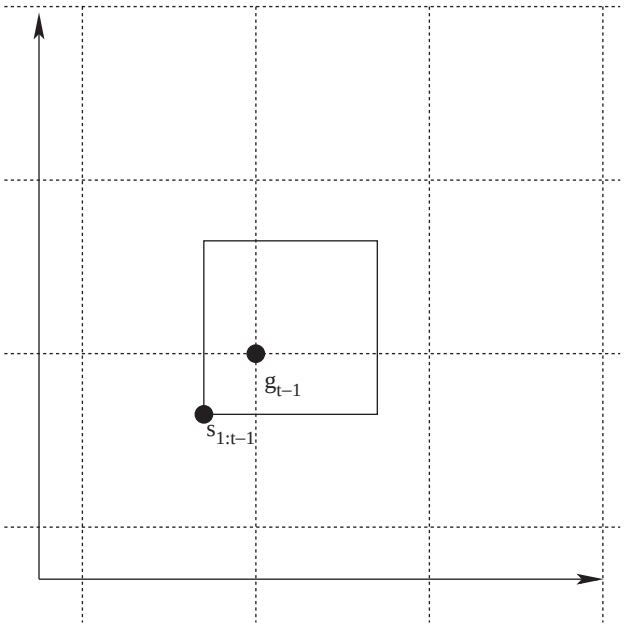

Fig. 1. The perturbed point $s_{1:t-1} + p_t$ is uniformly random over a cube of side $1/\varepsilon$ with vertex at $s_{1:t-1}$. One way to do this is to choose a random grid of spacing $1/\varepsilon$ and take the unique grid point in this cube. By using the same grid each period, the selected point moves rarely (for sufficiently large $1/\varepsilon$).

If calling the oracle is a computationally expensive operation or if there is a cost to switching between different decisions, then this is a desirable property. For example, to find the best binary search tree in hindsight on n items takes time $O(n^2)$, and it would be ridiculous to do this between every access to the tree.

The trick is to take advantage of the fact that we can correlate our perturbations from one period to the next—this will not change the expected totals. We will choose the perturbations so that $s_{1:t-1} + p_t = s_{1:t} + p_{t+1}$ as often as possible, as shown in Fig. 1. When this is the case, we do not need to call $M(s_{1:t} + p_{t+1})$ as we will get the same result.

- **FLL**(ε):
  1. Once, at the beginning, choose $p \in \left[0, \frac{1}{\varepsilon}\right]^n$ uniformly, determining a grid $G = \{p + \frac{1}{\varepsilon} z | z \in \mathbb{Z}^n\}$.
  2. On period t, use $M(g_{t-1})$, where g_{t-1} is the unique point in $G \cap \left(s_{1:t-1} + [0, \frac{1}{\varepsilon})^n\right)$. (Clearly if $g_t = g_{t-1}$, then there is no need to re-evaluate $M(g_t) = M(g_{t-1})$.)

It is not difficult to see that the point g_{t-1} is uniformly distributed over $s_{1:t-1} + [0, \frac{1}{\varepsilon})^n$, like FPL. Thus, in expectation, FPL(ε) and FLL(ε) behave identically on any single period, for any fixed sequence of states. Furthermore, since often $g_{t-1} = g_t$, rarely does a decision need to be changed or even computed. To be more formal:

Proof of Lemma 1.2 (FLL case). FLL(ε) chooses a uniformly random grid of spacing $1/\varepsilon$. There will be exactly one grid point inside $s_{t-1} + [0, \frac{1}{\varepsilon})^n$, and by symmetry, it is uniformly distributed over

that set. Thus we see that the grid point g_{t-1} will be distributed exactly like FPL(ε), uniform over $s_{1:t-1} + \left[0, \frac{1}{\varepsilon}\right]^n$.

Now, $g_{t-1} \neq g_t$ iff the grid point in $s_{1:t-1} + \left[0, \frac{1}{\varepsilon}\right]^n$, which we know is uniform over this set, is not in $s_{1:t} + \left[0, \frac{1}{\varepsilon}\right]^n$. By Lemma 3.2, we know this happens with probability at most $\varepsilon|s_t|_1$. □

4. Competitive analysis

The competitive theorems are similar. The restriction we make is that decision and state vectors are non-negative, i.e. $\mathcal{D}, \mathcal{S} \subset \mathbb{R}^n_+$.

Proof of Theorem 1.1. (b) WLOG, we may assume $p_t = p_1$ for all $t > 1$, because this does not change the expectation. As before, by Lemma 3.1,

$$\sum_{t=1}^{T} M(s_{1:t} + p_1) \cdot s_t \leqslant M(s_{1:T}) \cdot s_{1:T} + D|p_1|_\infty.$$

At the end of Section 2, it was shown that the expected maximum of n exponential distributions with mean ε is at most $(1 + \ln n)/\varepsilon$, i.e. $|p_1|_\infty \leqslant (1 + \ln n)/\varepsilon$. Furthermore, we claim that

$$E[M(s_{1:t-1} + p_1) \cdot s_t] \leqslant e^{\varepsilon A} E[M(s_{1:t} + p_1) \cdot s_t]. \tag{6}$$

To see this, again notice that the distributions over $s_{1:t-1} + p_1$ and $s_{1:t} + p_1$ are similar. In particular,

$$\begin{aligned} E[M(s_{1:t-1} + p_1) \cdot s_t] &= \int_{x \in \mathbb{R}^n} M(s_{1:t-1} + x) \cdot s_t \, d\mu(x) \\ &= \int_{y \in \mathbb{R}^n} M(s_{1:t} + y) \cdot s_t \, d\mu(y + s_t) \\ &= \int_{y \in \mathbb{R}^n} (M(s_{1:t} + y) \cdot s_t) \, e^{-\varepsilon(|y+s_t|_1 - |y|_1)} \, d\mu(y). \end{aligned} \tag{7}$$

Finally, $-\varepsilon(|y + s_t|_1 - |y|_1) \leqslant \varepsilon|s_t|_1 \leqslant \varepsilon A$ by the triangle inequality. This establishes (6). For $\varepsilon \leqslant 1/A$, $e^{\varepsilon A} \leqslant 1 + 2\varepsilon A$. Finally, combining the above gives,

$$E[\text{cost of FPL}^*(\varepsilon)] \leqslant (1 + 2\varepsilon A)\left(\text{min-cost}_T + \frac{D(1 + \ln n)}{\varepsilon}\right).$$

Evaluating FPL$^*(\varepsilon/2A)$ and using the fact that $\varepsilon \leqslant 1$ gives the theorem. □

Remark 1. The careful reader will have observed that we did not require any positive perturbations. Since s_t is always nonnegative, for Eq. (7), the theorem would hold if we choose only negative perturbations. The reason we use a symmetric distribution is only out of convenience—to be compatible with our FLL* algorithm, for which we do not know how to design an asymmetric version.

Remark 2. A small technical difficulty arises in that for these multiplicative algorithms, $s_{1:t-1} + p_t$ may have negative components, especially for small t. For some problems, like the online path problem, this can

cause difficulty because there may be negative cycles in the graph. (Coincidentally, Takimoto and Warmuth make the assumption that the graph has no cycles whatsoever [28].) A less-restrictive approach to solving this problem in general is to add large fixed pretend costs at the beginning, i.e. $s_0 = (M, M, \ldots, M)$. For a sufficiently large M, with high probability all of the components of $s_{0:t-1} + p_t$ will be non-negative. Furthermore, one can show that these costs do not have too large an effect. A more elegant solution for the path problem is given by Awerbuch and Mansour [3].

A lazy version of the multiplicative algorithm can be defined as well:

- **FLL*(ε)**:
  1. Choose p_1 at random according to the density $d\mu(x) \propto e^{-\varepsilon|x|_1}$.
  2. On each period t, use $M(s_{1:t-1} + p_t)$.
  3. Update
  (a) With probability $\min\left(1, \frac{d\mu(p_t - s_t)}{d\mu(p_t)}\right)$, set $p_{t+1} = p_t - s_t$ (so that $s_{1:t} + p_{t+1} = s_{1:t-1} + p_t$).
  (b) Otherwise, set $p_{t+1} := -p_t$.

  In expectation, this algorithm is equivalent to FPL*.

Proof of Lemma 1.2 (FLL* case). We first argue by induction on t that the distribution of p_t for FLL*(ε) has the same density $d\mu(x) \propto e^{-\varepsilon|x|_1}$. (In fact, this holds for any center-symmetric $d\mu$.) For $t = 1$ this is trivial. For $t + 1$, the density at x is

$$d\mu(x + s_t) \min\left\{1, \frac{d\mu(x)}{d\mu(x + s_t)}\right\} + d\mu(-x)\left(1 - \min\left\{1, \frac{d\mu(-x - s_t)}{d\mu(-x)}\right\}\right). \tag{8}$$

This is because we can reach $p_{t+1} = x$ by either being at $p_t = x + s_t$ or $p_t = -x$. Observing that $d\mu(-x) = d\mu(x)$,

$$\begin{aligned} d\mu(x + s_t) \min\left\{1, \frac{d\mu(x)}{d\mu(x + s_t)}\right\} &= \min\{d\mu(x + s_t), d\mu(x)\} \\ &= d\mu(-x) \min\left\{1, \frac{d\mu(-x - s_t)}{d\mu(-x)}\right\}. \end{aligned}$$

Thus, (8) is equal to $d\mu(x)$.

Finally, the probability of switching is at most

$$\begin{aligned} 1 - \frac{d\mu(p_t + s_t)}{d\mu(p_t)} &= 1 - e^{-\varepsilon(|p_t + s_t|_1 - |p_t|_1)} \\ &\leqslant 1 - e^{-\varepsilon|s_t|_1} \\ &\leqslant \varepsilon|s_t|_1 \\ &\leqslant \varepsilon A. \qquad \square \end{aligned}$$

Again, the above shows that the oracle need be called very rarely—only when $s_{1:t-1} + p_t$ changes.

5. Approximation algorithms

We have seen that the online version of linear optimization can be solved using an optimal offline algorithm. In particular, when the offline optimization problem can be solved exactly in polynomial-time,

so can the online version. In this section, we consider the situation when the algorithm for the offline optimization problem is only guaranteed to find an *approximate* optimum.

We could apply our online algorithms here, with the change that instead of calling an exact optimization oracle M, we have access to an approximation algorithm A. We say that A achieves an α-approximation if, on any input, the cost of the solution it finds is at most α times the minimum solution for a minimization problem.

The difficulty in the analysis is Eq. (4). In the case of an approximation, we can only say

$$\sum_{t=1}^{T} A(s_{1:t}) \cdot s_t \leqslant \alpha^T M(s_{1:T}) \cdot s_{1:T}.$$

For problems with a FPTAS (see [29]), we can use say $\varepsilon/4$ instead of ε in FPL* and an $\alpha = (1 + \frac{\varepsilon}{6T})$ approximation, because the result would be $(1 + \frac{\varepsilon}{3})(1 + \frac{\varepsilon}{6T})^T \leqslant 1 + \varepsilon$ competitive.

For approximation algorithms with larger α, another type which can be used is the following:

Definition 1. An approximation algorithm A for a linear minimization problem on variables $x^1, \ldots, x^n$, is said to achieve an α point-wise approximation to M, if on any input instance x, the solution it finds, $A(x)$, has the property that $E[A(x)^i] < \alpha M(x)^i$ for all i.

The definition for maximization problems is analogous. Several algorithms have point-wise guarantees, e.g. the max-cut algorithm of [13], the metric labeling algorithm of [18], etc.

For any sequence of states, $s_1, s_2, \ldots, s_t$, it is easy to see that

$$\sum_{t=1}^{T} A(s_t) \cdot s_t \leqslant \alpha \sum_{t=1}^{T} M(s_t) \cdot s_t.$$

Thus following the perturbed leader with a pointwise approximation algorithm costs at most α times as much as the (inefficient) exact online version, i.e. the competitive ratio goes up by a factor of α.

Other examples of approximation algorithms with pointwise guarantees include the randomized vertex ordering algorithms of [20,10,24,9].

6. Conclusions and open problems

For many problems, exponential weighting schemes such as the weighted majority provide inefficient online algorithms that perform almost as well as the offline analogs. Hannan's approach can be generalized to get efficient algorithms for linear problems whose offline optimization can be done efficiently.

This separation of the online optimization problem into its online and offline components seems helpful. In many cases, the guarantees of this approach may be slightly worse than custom-designed algorithms for problems (the additive term may be slightly larger). However, we believe that this separation at least highlights where the difficulty of a problem enters. For example, an online shortest-path algorithm [28] must be sophisticated enough at least to solve the offline shortest path problem.

Furthermore, the simplicity of the "follow the leader" approach sheds some light on the static online framework. The worst-case framework makes it problematic to simply follow the leader, which is a

natural, justifiable approach that works in other models. Adding randomness simply makes the analysis work, and is necessary only in the worst case kind of sequence where the leader changes often. (Such a sequence may be plausible in some scenarios, such as compressing the sequence 0101….)

As one can see, there are several ways to extend the algorithm. Recently, Awerbuch and Kleinberg [2], and Blum et al. [23] have extended the algorithm to the *bandit* case of the generalization, where only the cost of the chosen decision is revealed. Surprisingly, given only this limited feedback, they can still guarantee asymptotically low regret. Their challenge is to nicely deal with the exploration/exploitation tradeoff.

Other variations include tracking (following the best decision that may change a few times). We have also considered using the L_2 norm rather than the L_1 norm [16]. It is not clear to us how to generalize to other loss functions than the one used here.

Finally, while these algorithms are fairly general, there are of course many problems for which they cannot be used. It would be great to generalize FPL to nonlinear problems such as portfolio prediction [8]. For this kind of problem, it is not sufficient to maintain additive summary statistics.

Acknowledgments

We would like to thank Avrim Blum, Bobby Kleinberg, Danny Sleator, and the anonymous referees for their helpful comments.

References

[1] S. Arora, Polynomial time approximation schemes for Euclidean TSP and other geometric problems, J. ACM 45 (1998) 753–782.
[2] B. Awerbuch, R. Kleinberg, Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches, in: Proceedings of the 36th ACM Symposium on Theory of Computing, 2004, pp. 45–53.
[3] B. Awerbuch, Y. Mansour, Adapting to a reliable network path, in: Proceedings of the 21st ACM Symposium on Principles of Distributed Computing, 2003, pp. 360–367.
[4] A. Blum, On-line algorithms in machine learning, Technical Report CMU-CS-97-163, Carnegie Mellon University, 1997.
[6] A. Blum, S. Chawla, A. Kalai, Static optimality and dynamic search optimality in lists and trees, Algorithmica 36 (3) (2003) 249–260.
[8] T. Cover, Universal portfolios, Math. Finance 1 (1991) 1–29.
[9] J. Dunagan, S. Vempala, On Euclidean embeddings and bandwidth minimization, in: Proceedings of the Fifth International Symposium on Randomization and Approximation techniques in Computer Science, 2001, pp. 229–240.
[10] U. Feige, Approximating the bandwidth via volume respecting embeddings, in: Proceedings of the 30th ACM Symposium on the Theory of Computing, 1998, pp. 90–99.
[11] Y. Freund, R. Schapire, Y. Singer, M. Warmuth, Using and combining predictors that specialize, in: Proceedings of the 29th Annual ACM Symposium on the Theory of Computing, 1997, pp. 334–343.
[12] D. Foster, R. Vohra, Regret in the on-line decision problem, Games Econom. Behav. 29 (1999) 1084–1090.
[13] M. Goemans, D. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, J. ACM 42 (1995) 1115–1145.
[14] J. Hannan, Approximation to Bayes risk in repeated plays, in: M. Dresher, A. Tucker, P. Wolfe (Eds.), Contributions to the Theory of Games, vol. 3, Princeton University Press, Princeton, 1957, pp. 97–139.
[15] D. Helmbold, R. Schapire, Predicting nearly as well as the best pruning of a decision tree, Mach. Learning 27 (1) (1997) 51–68.
[16] A. Kalai, S. Vempala, Geometric algorithms for online optimization, MIT Technical Report MIT-LCS-TR-861, 2002.

[18] J. Kleinberg, E. Tardos, Approximation algorithms for classification Problems with Pair-wise relationships: Metric labeling and Markov random fields, in: Proceedings of 39th Foundations of Computer Science, 1999, pp. 14–23.
[19] D. Knuth, Dynamic Huffman coding, J. Algorithms 2 (1985) 163–180.
[20] N. Linial, E. London, Y. Rabinovich, The geometry of graphs and some of its algorithmic applications, Combinatorica 15 (2) (1995) 215–245.
[21] N. Littlestone, From on-line to batch learning, in: Proceedings of the Second Annual Workshop on Computational Learning Theory, 1989, pp. 269–284.
[23] B. McMahan, A. Blum, Online geometric optimization in the bandit setting against an adaptive adversary, in: Proceedings of the 17th Annual Conference on Learning Theory, 2004, pp. 109–123.
[24] S. Rao, Small distortion and volume preserving embeddings for planar and Euclidean metrics, in: Proceedings of Symposium on Computational Geometry, 1999, pp. 300–306.
[25] Daniel Sleator, Robert Tarjan, Amortized efficiency of list update and paging rules, Comm. ACM 28 (1985) 202–208.
[26] D. Sleator, R. Tarjan, Self-adjusting binary search trees, J. ACM 32 (1985) 652–686.
[27] E. Takimoto, M. Warmuth, Predicting nearly as well as the best pruning of a planar decision graph, Theoret. Comput. Sci. 288 (2) (2002) 217–235.
[28] E. Takimoto, M. Warmuth, Path kernels and multiplicative updates, J. Mach. Learning Res. 4 (5) (2003) 773–818.
[29] V. Vazirani, Approximation Algorithms, Springer, Berlin, 2001.
[31] M. Zinkevich, Online convex programming and generalized infinitesimal gradient ascent, in: Proceedings of the 20th International Conference on Machine Learning, 2003, pp. 928–936.