Efficient Algorithms for Online Decision Problems

Adam Kalai[1] and Santosh Vempala[1]

Massachusetts Institute of Technology, 77 Mass. Ave.
Cambridge, MA 02139, USA
{akalai, vempala}@math.mit.edu

Abstract. In an online decision problem, one makes a sequence of decisions without knowledge of the future. Tools from learning such as Weighted Majority and its many variants [13, 18, 4] demonstrate that online algorithms can perform nearly as well as the best single decision chosen in hindsight, even when there are exponentially many possible decisions. However, the naive application of these algorithms is *inefficient* for such large problems. For some problems with nice structure, specialized efficient solutions have been developed [10, 16, 17, 6, 3].
We show that a very simple idea, used in Hannan's seminal 1957 paper [9], gives *efficient* solutions to all of these problems. Essentially, in each period, one chooses the decision that worked best in the past. To guarantee low regret, it is necessary to add randomness. Surprisingly, this simple approach gives additive ϵ regret per period, efficiently. We present a simple general analysis and several extensions, including a $(1+\epsilon)$-competitive algorithm as well as a lazy one that rarely switches between decisions.

1 Introduction

In an online decision problem, one has to make a sequence of decisions without knowledge of the future. Exponential weighting schemes for these problems have been discovered and rediscovered in may areas [7]. Even in learning, there are too many results to mention (for a survey, see [1]). We show that Hannan's original idea[1] of doing what worked best against the past (with perturbed totals) gives efficient and simple algorithms for online decision problems. We extend his algorithm to get multiplicative $(1+\epsilon)$ guarantees as well as algorithms that do few updates. Fortunately, the same algorithm and analysis apply to many such problems, including some open problems. Let us begin with examples.

Experts problem. There are n experts, each of which incurs a cost between 0 and 1 each period. Each period, we have to pick a single expert, and then we incur the same cost as that expert. After we pick the expert, the costs of all experts are revealed. The goal is to have a total cost not much larger than the minimum total cost of any expert.

- Follow the perturbed leading expert: On each period $t = 1, 2, \ldots,$

[1] We are grateful to Sergiu Hart for the pointer to Hannan's algorithm; we regret that we were unaware of it in an earlier version of this paper [11].

1. For each expert e, pick $p_t[e] \geq 0$ randomly from the exponential distribution $d\mu(x) = \epsilon e^{-\epsilon x}$.
2. Choose the expert with smallest $c[e] - p_t[e]$, where $c[e]$ is the total cost of expert e so far.

As we discuss later, one can show that on any period,

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

The above algorithm and guarantees are similar to the randomized version of Weighted Majority. We present this application just as motivation, for those familiar with Weighted Majority.[2]

Online shortest path. [16] One has a directed graph with n nodes and m edges, 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. The standard solution to this type of problem would be to view each *path* as an expert, and use an algorithm such as Weighted Majority. The difficulty is that there may be exponentially many paths, and so this is inefficient. Fortunately, by following the perturbed leader, we can now take advantage of the structure of the problem and only add randomness to the edges individually rather than considering each path seperately:

- Follow the perturbed leading path: On each period $t = 1, 2, \ldots$,
  1. For each edge e, pick $p_t[e] \in \mathbb{R}$ randomly from an exponential distribution. (See Section 2.4 for the exact parameters.)
  2. Use the shortest path in the graph with weight $s[e] + p_t[e]$ on edge e, where $s[e]$ is the total time on edge e so far.

As a corollary of Theorem 2,

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

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 bounds of Takimoto and Warmuth [16], and their specialized algorithm is also efficient. For the next problem, no efficient near optimal algorithm was previously known.

Tree update problem. This problem is a classic online problem [15] introduced by Sleator and Tarjan with Splay Trees, around the same time as they introduced the list update problem [14]. In the tree update problem, one maintains a binary search tree over n items in 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.

[2] The natural idea of following the leader without randomness fails on the following example. Imagine just two experts whose cost sequence is $(0, \frac{1}{2}), (1, 0), (0, 1), (1, 0), (0, 1), \ldots$. The leader, i.e. best expert so far, always happens to be the one that costs 1 next. So, following the leader will cost about t, while staying with either expert by itself will cost about $t/2$ (worse example with $n > 2$).

One could use "follow the perturbed leader" mentioned above 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 $\Theta(n^2)$ using dynamic programming). But doing so much computation and so many tree rotations, just to prepare for a lookup seems a little bit ridiculous. Instead, we give a way to achieve the same effect with little computation and few updates to the tree:

- Follow the lazy tree(N):
  1. For $1 \leq i \leq n$, 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 $s_a \geq v_a$ 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[3] as a corollary of Theorem 1,

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

Because any algorithm must pay at least 1 per acccess, the above additive regret bound is even stronger than a multiplicative $(1+\epsilon)$-competitive bound, i.e. $T \leq$ (cost of best tree). In contrast, Splay Trees have a guarantee of $3\log_2 3 \times$(cost of best tree) plus an additive term, but they have other desirable properties. Standard tricks can be used if T is not known in advance.

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. This algorithm has what Blum et. al. call *strong static optimality* [3]. They also presented a follow the perturbed leader type of algorithm for the easier list update problem. Theirs was the original motivation for our work, and they were also unaware of the similarity to Hannan's algorithm.

1.1 Generalization

What is important about the above problems is that in each problem there are a small number of summary features, e.g. the total time on each edge and the total number of accesses to each item. The cost of any decision, e.g. path or tree, is *additive* in terms of these summary feature. That is, the cost of a decision over two sequences is the cost when we add the two sets of summary features. The computational savings comes from adding randomness only to each summary feature, because

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

there are typically a small number of summary features compared to the exponentially (or even infinitely many) possible decisions.

We give (expected) $(1+\epsilon)$-competitive bounds for any such additive problem. We also give guarantees on having (expected) ϵ fraction of periods where any calculation or changes of decision are necessary. It is remarkable that Hannan in 1957 [9], in addition to inventing the problem, came up with an algorithm that has efficiency properties better than those of many modern algorithms. While he wasn't concerned with efficiency and only studied additive regret, the algorithm we call "follow the perturbed leader" for additive regret is essentially his algorithm. Perhaps the reason his particular algorithm hasn't often been revisited is because his analysis is quite complex. In fact, we don't understand it, and we present simple proofs of all our results. However, his is inspirational paper may be worth mining for other good ideas.

We discuss various extensions, such as to the case when you can only approximate the leader and to the case when the set of decisions is a convex set, where you can average decisions. We discuss several applications, including online shortest path, the tree update problem, online decision tree pruning, online linear programming, and adaptive Huffman coding. For the tree update problem and adaptive Huffman coding, no efficient $(1+\epsilon)$-optimal algorithms were known. We hope the technique will be useful for other natural problems.

The focus of an earlier version of this paper [11] was the general problem of online linear optimization, which we describe later. Independently, Zinkevich has introduced an elegant deterministic algorithm for the more general online convex optimization problem [19]. His algorithm is well-suited for convex problems but not for the discrete problems which we focus on here.

2 Algorithms

In this section we define the model for online decision problems and describe Hannan's basic algorithm which leads to a bound on the (additive) regret. Then we describe an extension that changes decisions rarely, and another extension which has a (multiplicative) competitive bound.

2.1 The model

A decision maker must make a series of decisions from a possibly infinite set $\mathcal{D}$. After each decision is made, a *state* is revealed. We assume that each state can be represented by a nonnegative vector $s \in \mathbb{R}_+^n$ from some set of feasible states $S \subset \mathbb{R}_+^n$. In the online path example, this would be the vector with a component for each edge indicating the time on that edge. In the online tree problem, it would simply be the vector e_i (the vector of all 0s with a 1 in the ith position) for an access to item number i. Moreover we assume there is a non-negative cost for each decision and state, represented by a function $c : \mathcal{D} \times S \to \mathbb{R}_+$. We assume this function is *additive* and extend it to $\mathcal{D} \times \mathbb{R}^n$. By additive, we mean that the cost for a single decision d on a sequence of states $s_1, s_2, \ldots, s_t$ can be computed by

$$c(d, s_1) + c(d, s_2) + \ldots + c(d, s_t) = c(d, s_1 + s_2 + \ldots + s_t).$$

Here the sum $s_1 + \ldots + s_t$ are the summary features of the problem. It follows that each decision d can be equated to a point in $d \in \mathbb{R}^n$, where the ith coordinate is $c(d, e_i)$. For example, for a path, this would be a $\{0,1\}$ vector with a 1 in every position corresponding to an edge in the path. For a tree, this would be the vector where the ith component is the depth of i in the tree. For notational ease, we make no distinction between a decision and its corresponding vector, so that $\mathcal{D} \subset \mathbb{R}^n$. Thus the cost of a decision on the sequence $s_1, s_2, \ldots, s_t$ can be written as

$$c(d, s_1) + c(d, s_2) + \ldots + c(d, s_t) = d \cdot (s_1 + s_2 + \ldots + s_t).$$

For further succinctness, we use the notational shortcut

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

In order to avoid searching over all decisions, we need to assume that there is some oracle M that can tell us, for any summary vector $s \in \mathbb{R}^n$, what the best decision would have been.

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

In the path problem, this could be implemented by the shortest path algorithm applied to the total times on each edge. In the tree problem, this could be implemented by a dynamic programming algorithm that takes as input the total number of times each item was accessed. For a state sequence $s_1, s_2, \ldots, s_t$, the minimum achievable *offline* static cost is $M(s_{1:t}) \cdot s_{1:t}$.

Finally, we need to bound the costs and other vectors. Suppose,

$$\begin{aligned} c(d, s) &\in [0, R], \text{ for all } d \in \mathcal{D}, s \in S \\ |s|_1 &\le A, \text{ for all } s \in S \\ |d - d'|_1 &\le D, \text{ for all } d, d' \in \mathcal{D} \end{aligned}$$

Here $|x|_1 = |x_1| + |x_2| + \ldots + |x_n|$ is the L_1 norm of $x \in \mathbb{R}^n$. The parameter D, the L_1 diameter of the set of decisions, is a geometric parameter that we use to bound the performance. In the above bounds, we actually only need to consider "reasonable" decisions, i.e. decisions that are the possible output of the oracle $M(s)$ for some $s \in \mathbb{R}^n$. For example, one need not consider paths that visit a node more than once. Also, R need only be an upper bound on the difference between the cost of two decisions on a single period.

The L_1 norm may seem arbitrary, but it is quite natural for many discrete applications as we will presently see. In [11], we also discuss the L_2 norm.

2.2 Follow the perturbed leader

In this section, we describe Hannan's algorithm in the context of an additive online decision problem. Recall that $s_{1:t} = s_1 + s_2 + \ldots + s_t$.

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

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

Theorem 1. *Let $s_1, s_2, \ldots \in \mathbb{R}^n$ be any state sequence. For any $T > 0$, the expected cost of FPL(ϵ) on the first T periods is bounded by,*

$$E[\text{cost of } FPL(\epsilon)] \leq M(s_{1:T}) \cdot s_{1:T} + \epsilon RAT + \frac{D}{\epsilon},$$

where D, R and A are defined in Section 2.1.

We defer all proofs to the appendix. 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 don't 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.
If T is known in advance, $\epsilon = \sqrt{D/(RAT)}$ minimizes the above bound giving $2\sqrt{DRAT}$ regret,

$$E[\text{cost of FPL}(\sqrt{D/RAT})] \leq M(s_{1:t}) \cdot s_{1:t} + 2\sqrt{DRAT}$$

Without advance knowledge of T, using a standard ϵ-halving technique, where every so often you restart with a smaller ϵ, you can achieve slightly worse bounds. Hannan gives a more elegent solution, where he slowly increased the size of the perturbation:

- **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 $M(s_{1:t-1} + p_t)$.

In the appendix, we will show the following (similar to what Hannan showed) but our proof is significantly simpler:

$$E[\text{cost of Hannan}(\delta = \sqrt{D/2AR})] \leq M(s_{1:T}) \cdot s_{1:T} + 2\sqrt{2DRAT}$$

Thus one loses only a factor of $\sqrt{2}$ for not knowing T, setting $\delta = \sqrt{D/(2AR)}$,

2.3 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.

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 $s_{1:t-1} + p_t = s_{1:t} + p_{t+1}$, or in terms of FLL, $g_{t-1} = g_t$, as often as possible.

- **FLL**(ϵ):
  1. Once, at the beginning, choose $p \in \left[0, \frac{1}{\epsilon}\right]^n$ uniformly, determining a grid $G = \{p + \frac{1}{\epsilon} z | z \in \mathbb{Z}^n\}$.
  2. On period t, use $M(g_{t-1})$, where g_{t-1} is the unique point in $G \cap \left(s_{1:t-1} + [0, \frac{1}{\epsilon})^n\right)$. (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 $s_{1:t-1} + [0, \frac{1}{\epsilon})^n$, like FPL. Thus FPL($\epsilon$) and FLL($\epsilon$) 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 precise,

Lemma 1. *For any fixed sequence of states $s_1, s_2, \ldots$, FPL(ϵ) and FLL(ϵ) behave identically on each period t, i.e. the distribution over g_{t-1} for FLL(ϵ) is identical to the distribution over $s_{1:t-1} + p_t$ for FPL(ϵ). Also, for FLL(ϵ), $\Pr[g_{t-1} \neq g_t] \leq \epsilon |s_t|_1 \leq \epsilon A$.*

The main *disadvantage* of the FLL algorithm is that it is predictable – its choices in different rounds are very related. This means that an adaptive adversary can force the algorithm to have large regret. This is true of any algorithm that uses very little randomness.

2.4 Competitive versions of FLL and FPL

In this section, we give algorithms that are nearly optimal in a multiplicative sense.

- **FPL***(ϵ): On each period t,
  1. Choose p_t at random according to the density $d\mu(x) \propto e^{-\epsilon|x|_1}$. (This can be done by, for each coordinate, choosing $r \geq 0$ according to the standard exponential density e^{-r} and setting the ith coordinate of p_t to $\pm(r/\epsilon)$.)
  2. Use $M(s_{1:t-1} + p_t)$.
- **FLL***(ϵ):
  1. Choose p_1 at random according to the density $d\mu(x) \propto e^{-\epsilon|x|_1}$.

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 $p_{t+1} := -p_t$.

Lemma 2. *On any period t, FPL*(ϵ) and FLL*(ϵ) behave identically. In other words, the distribution over $s_{t-1} + p_t$ for both is the same. Also, for FLL*(ϵ), $\Pr[s_{1:t-1} + p_t \neq s_{1:t} + p_{t+1}] \leq \epsilon |s_t|_1 \leq \epsilon A$.*

Again, the above shows that the oracle need be called very rarely – only when $s_{1:t-1} + p_t$ changes. The main theorem here is:

Theorem 2. *For any state sequence $s_1, s_2, \ldots, s_t$ and any $0 \leq \epsilon \leq 1/A$, the cost of FPL*(ϵ) is bounded by,*

$$E[\text{cost of FPL*}(\epsilon)] \leq (1 + \epsilon 2A) M(s_{1:T}) \cdot s_{1:T} + \frac{D(1 + \log n)}{\epsilon}$$

Using $\epsilon' = \epsilon/2A$, one can get $(1+\epsilon)$ guarantees with an $O(AD \log(n)/\epsilon)$ additive term. 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 [16].) 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.

2.5 Follow the expected leader and online linear optimization

A simple extension is possible for convex sets, which we call 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 $\left[0, \frac{1}{\epsilon}\right]^n$.
  2. Use $\frac{1}{m} \sum_{i=1}^m M(s_{1:t-1} + p_t^i)$.

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.*

Since linear optimization is a general framework, this models many online problems. The parameters D and A now take on special geometric

significance as D is the diameter of the feasible set and A is a bound on the objective vectors. Other extensions for this problem are to allow for negative objective vectors and costs, in the additive regret case.
In [11], we discussed this problem in greater detail. Independently, Zinkevich has introduced a deterministic algorithm for an online convex optimization problem [19], which is more general than the problem described in this section. His nice algorithm is well-suited for convex problems like the one mentioned in this section but not for the discrete problems which we focused on earlier.

3 Applications

We have already discussed the online shortest paths and binary search tree applications. For these problems, it is just a matter of calculating the parameters. For example, in the tree problem, $R = n$, $A = 1$, and $D = n^2$ will suffice as upper bounds. In the online path problem, $A = m$ and $D = 2n$ will suffice, assuming times are in $[0, 1]$, and of course we only need to consider paths of length at most n.
The Adaptive Huffman coding problem [12] 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 such a problem, however, it is natural to be concerned about sequences of alternating 0s and 1s. Adaptive Huffman coding does not give $(1+\epsilon)$ 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 $(1+\epsilon)$ algorithms have been designed for online pruning of decision trees, decision graphs, and their variants [10, 17]. Not surprisingly, FPL* and FLL* will apply.

3.1 Predicting from expert advice

We would like to apply our algorithm to the predicting from expert advice problem [4], 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[4]. Thus we may as well imagine that $A = 1$, and we

[4] 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.

get the standard $(1+\epsilon) \times (\text{best expert}) + O(\log n/\epsilon)$ bounds of Weighted Majority.
In fact, one can give a very simple direct analysis of FPL* for the experts problem in less than a page (which we cannot afford). In summary, one first observes that if the leader never changes, then following the leader gives no regret. Second, one observes that the total additive regret is at most the number of times the leader has changed. Third, one checks that by adding randomness with an exponential distribution, the probability that the leader changes during any period is very small, in particular at most ϵ times the expected cost of the algorithm during that period. Finally, one checks that adding randomness of that magnitude doesn't hurt too much.

4 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 original idea leads to an efficient algorithm, with similar guarantees, provided that the offline problem can be solved efficiently.
This separation of the adaptive 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 [16] 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. One natural variation is tracking (following the best decision that may change a few times). Another variation would be a bandit version, but it may be challenging to come up with a good model of revealed information.
A problem with our approach is that one needs to solve the optimization problem exactly in order for our theorems to work. For some problems, only an approximate solution is possible. If, for any ϵ, there is an efficient algorithm for finding a $1+\epsilon$ optimal solution, i.e. one of cost at most $1+\epsilon$ times the cost of the best static offline solution, then the given approach can still be made to work. However, an interesting problem is to extend it to the case where only worse approximations are known.
In [11], we point out that some approximation algorithms have a pointwise guarantee which allows our analysis to work. Such problems include the max-cut algorithm of [8] and others. We find the online max-cut

problem particularly natural: edges are added, one at a time, to a multi-graph. At each time period, we must choose a cut, and receive a score of 1 if the added edge crosses the cut and 0 otherwise.

Finally, while Hannan's algorithm and our variants are quite 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 [5]. For this kind of problem, it is not sufficient to maintain additive summary statistics.

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

References

1. A. Blum. On-line algorithms in machine learning. Technical Report CMU-CS-97-163, Carnegie Mellon University, 1997
2. Avrim Blum and Carl Burch. On-line learning and the metrical task system problem. *Machine Learning*, 39(1):35–58, April 2000.
3. Avrim Blum, Shuchi Chawla, and Adam Kalai. Static Optimality and Dynamic Search Optimality in Lists and Trees. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '02), 2002.*
4. N. Cesa-Bianchi, Y. Freund, D. Haussler, D. Helmbold, R. Schapire, and M. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427-485, 1997.
5. Thomas Cover. Universal Portfolios. In *Math. Finance* **1**, 1-29, 1991.
6. Y. Freund, R. Schapire, Y. Singer, and M. Warmuth. Using and combining predictors that specialize. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pp. 334–343, 1997.
7. D. Foster and R. Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, vol.29, pp.1084-1090, 1999.
8. M. Goemans and D. Williamson, "Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming," J. ACM, 42, 1115–1145, 1995.
9. J. Hannan. Approximation to Bayes risk in repeated plays. In M. Dresher, A. Tucker, and P. Wolfe, editors, *Contributions to the Theory of Games, volume 3*, pages 97-139. Princeton University Press, 1957.
10. D. Helmbold and R. Schapire. Predicting nearly as well as the best pruning of a decision tree. *Machine Learning*, 27(1):51-68, 1997.
11. A. Kalai and S. Vempala. Geometric algorithms for online optimization. MIT Technical report MIT-LCS-TR-861, 2002.
12. D. Knuth. Dynamic Huffman Coding. *J. Algorithms*, 2:163-180, 1985.
13. N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.
14. Daniel Sleator and Robert Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202-208, 1985.
15. Daniel Sleator and Robert Tarjan. Self-Adjusting Binary Search Trees. *Journal of the ACM* 32:652-686, 1985.

16. E. Takimoto and M. Warmuth. Path Kernels and Multiplicative Updates. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pp. 74-89, 2002.
17. E. Takimoto and M. Warmuth. Predicting Nearly as Well as the Best Pruning of a Planar Decision Graph. *Theoretical Computer Science*, 288(2): 217-235, 2002.
18. V. Vovk. Aggregating strategies. In *Proc. 3rd Ann. Workshop on Computational Learning Theory*, pp. 371–383, 1990.
19. M. Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. CMU Technical Report CMU-CS-03-110, 2003.

A Analysis

The motivation for following the perturbed leader can be seen in the simple two-expert example given earlier. In that example, we would have no regret had we stayed with the same expert the whole time, and low regret so long as we didn't switch often. Now, imagine adding a pretend day 0, on which each expert had a random cost, chosen from some large range $[0, 1/\epsilon]$. As long as the two numbers are sufficiently far apart, follow the leader will stick with one expert the whole time. If these two numbers happen to be within 1 of each other, follow the leader will again alternate. This is the intuition behind the $(1+\epsilon)$ term. However, there is a penalty for adding randomness. The more randomness one adds, the more periods it will take to compensate for a false bias if one of the paths is actually better than the other, which leads to the additive term.
In the above motivation, we assumed that the randomness was chosen once in advance. By linearity of expectation, the expectation does not change if randomness is chosen anew each period. Furthermore, choosing new randomness protects against an adaptive adversary that can see one's previous decisions (but not one's random coins).
Here we present the proofs of the guarantees of the various algorithms.

A.1 FPL

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

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

For $T = 1$, it is trivial. For the induction step from $T-1$ to T, notice that

$$M(s_{1:T-1}) \cdot s_{1:T-1} \leq M(s_{1:T}) \cdot s_{1:T-1},$$

by definition of M. Combining this with $s_{1:T} = s_T + s_{1:T-1}$ completes the induction step.
Equation (1) 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.

Next we prove Theorem 1 of Section 2.2. We first show that perturbations don't hurt too much. Conceptually, it is easiest to bound this cost when $p_t = p_{t-1}$, because this is as if there was a pretend 0th period on which these perturbations actually happened. In general, we get:

Lemma 3. *For and state sequence $s_1, s_2, \ldots$, any $T > 0$, and any vectors $p_0 = 0$, $p_1, p_2, \ldots, p_t \in \mathbb{R}^n$,*

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

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 (1) gives,

$$\sum_{t=1}^{T} M(s_{1:t} + p_t) \cdot (s_t + p_t - p_{t-1}) \leq M(s_{1:T} + p_T) \cdot (s_{1:T} + p_T)$$
$$\leq M(s_{1:T}) \cdot (s_{1:T} + p_T)$$

By un-telescoping, we can rewrite this as,

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

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

Proof (Proof of Theorem 1). 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 to the latter scenario gives,

$$E\left[\sum_{t=1}^{T} M(s_t + p_1) \cdot s_t\right] \leq M(s_{1:T}) \cdot s_{1:T} + D|p_1|_\infty \leq M(s_{1:T}) \cdot s_{1:T} + \frac{D}{\epsilon} \tag{2}$$

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 ϵAR. **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,

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

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 4 following this proof, $1 - f \leq \epsilon |s_t|_1 \leq \epsilon A$.

Lemma 4. *For any $v \in \mathbb{R}^n$, the cubes $\left[0, \frac{1}{\epsilon}\right]^n$ and $v + \left[0, \frac{1}{\epsilon}\right]^n$ overlap in at least a $(1 - \epsilon|v|_1)$ fraction.*

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

A.2 Hannan

The following theorem was used at the end of Section 2.2:

Theorem 3. *For any state sequence $s_1, s_2, \ldots$, after any number of periods $T > 0$,*

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

Proof. WLOG we may choose $p_t = (\sqrt{t})p_1$, because all the p_t are identically distributed, and we are only bounding the expectation. Applying Lemma 3 to this scenario gives,

$$E\left[\sum_{t=1}^{T} M(s_{1:t} + \sqrt{t}p_1) \cdot s_t\right] \leq M(s_{1:T}) \cdot s_{1:T} + D|p_1|_\infty \sum_{t=1}^{T} (\sqrt{j} - \sqrt{j-1})$$

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 4, they overlap in a fraction that is at least $1 - |s_t|_1\delta/\sqrt{t} \geq 1 - A\delta/\sqrt{t}$. On this fraction, their expectation is identical so,

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

Thus we have shown,

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

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

A.3 FLL

Proof (Proof of Lemma 1). FLL(ϵ) chooses a uniformly random grid of spacing $1/\epsilon$. There will be exactly one grid point inside $s_{t-1} + [0, \frac{1}{\epsilon})^n$, 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 $s_{1:t-1} + \left[0, \frac{1}{\epsilon}\right]^n$.
Now, $g_{t-1} \neq g_t$ iff the grid point in $s_{1:t-1} + \left[0, \frac{1}{\epsilon}\right]^n$, which we know is uniform over this set, is not in $s_{1:t} + \left[0, \frac{1}{\epsilon}\right]^n$. By Lemma 4, we know this is at most $\epsilon|s_t|_1$.

A.4 FPL* and FLL*

Proof (Proof of Theorem 2). WLOG, we may assume $p_t = p_1$ for all $t > 1$, because this does not change the expectation. As before, by Lemma 1,

$$E\left[\sum_{t=1}^{T} M(s_{1:t} + p_1) \cdot s_t\right] \leq M(s_{1:T}) \cdot s_{1:T} + D|p_1|_\infty$$

In expectation, the magnitude of each coordinate of p_1 is $1/\epsilon$, as it is a number from the exponential distribution scaled by a $1/\epsilon$ factor. To bound the expected maximum, note that the expectation of a non-negative random variable X is $E[X] = \int_0^\infty \Pr[X \geq x]dx$. Consider $x_1, x_2, \ldots, x_n$, each drawn independently from the exponential distribution e^{-x}. The expected maximum is

$$\int_0^\infty \Pr[\max(x_1, x_2, \ldots, x_n) \geq x]dx \leq \left(\int_{\log n}^\infty ne^{-x}dx\right) + \log n = 1 + \log n$$

Furthermore, we claim that

$$E[M(s_{1:t-1} + p_1) \cdot s_t] \leq e^{\epsilon A} E[M(s_{1:t} + p_1) \cdot s_t] \tag{3}$$

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

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

Finally, $-\epsilon(|y+s_t|_1 - |y|_1) \leq \epsilon|s_t|_1 \leq \epsilon A$ by the triangle inequality. This establishes (3). For $\epsilon \leq 1/A$, $e^{\epsilon A} \leq 1 + 2\epsilon A$.

Proof (Proof of Lemma 2). We first argue by induction on t that the distribution over p_t for FLL*(ϵ) has the same density $d\mu(x) \propto e^{-\epsilon|x|_1}$. (In fact, this holds for any center-symmetric $d\mu$.) For $t = 1$ this is trivial. For $t+1$, the density at x is

$$d\mu(x + s_t)\min\{1, \frac{d\mu(x)}{d\mu(x + s_t)}\} + d\mu(-x)(1 - \min\{1, \frac{d\mu(-x - s_t)}{d\mu(-x)}\}) \tag{4}$$

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 $d\mu(-x) = d\mu(x)$,

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

Thus, (4) is equal to $d\mu(x)$.
Finally, the probability of switching is at most

$$\begin{aligned} 1 - \frac{d\mu(p_t + s_t)}{d\mu(p_t)} &= 1 - e^{-\epsilon(|p_t+s_t|_1 - |p_t|_1)} \\ &\leq 1 - e^{-\epsilon|s_t|_1} \\ &\leq \epsilon|s_t|_1 \end{aligned}$$