JMLR: Workshop and Conference Proceedings vol 40:1–19, 2015

Achieving All with No Parameters: AdaNormalHedge

Haipeng Luo HAIPENGL@CS.PRINCETON.EDU
Department of Computer Science, Princeton University

Robert E. Schapire SCHAPIRE@CS.PRINCETON.EDU
Microsoft Research, New York City and Department of Computer Science, Princeton University

Abstract

We study the classic online learning problem of predicting with expert advice, and propose a truly parameter-free and adaptive algorithm that achieves several objectives simultaneously without using any prior information. The main component of this work is an improved version of the NormalHedge.DT algorithm (Luo and Schapire, 2014), called AdaNormalHedge. On one hand, this new algorithm ensures small regret when the competitor has small loss and almost constant regret when the losses are stochastic. On the other hand, the algorithm is able to compete with any convex combination of the experts simultaneously, with a regret in terms of the relative entropy of the prior and the competitor. This resolves an open problem proposed by Chaudhuri et al. (2009) and Chernov and Vovk (2010). Moreover, we extend the results to the sleeping expert setting and provide two applications to illustrate the power of AdaNormalHedge: 1) competing with time-varying unknown competitors and 2) predicting almost as well as the best pruning tree. Our results on these applications significantly improve previous work from different aspects, and a special case of the first application resolves another open problem proposed by Warmuth and Koolen (2014) on whether one can simultaneously achieve optimal shifting regret for both adversarial and stochastic losses.

Keywords: expert algorithm, NormalHedge, adaptivity, unknown competitors, time-varying competitors, first order bounds, sleeping expert, adaptive regret, shifting regret

1. Introduction

The problem of predicting with expert advice was first pioneered by Littlestone and Warmuth (1994); Freund and Schapire (1997); Cesa-Bianchi et al. (1997); Vovk (1998) and others two decades ago. Roughly speaking, in this problem, a player needs to decide a distribution over a set of experts on each round, and then an adversary decides and reveals the loss for each expert. The player's loss for this round is the expected loss of the experts with respect to the distribution that he chose, and his goal is to have a total loss that is not much worse than any single expert, or more generally, any fixed and unknown convex combination of experts.

Beyond this classic goal, various more difficult objectives for this problem were studied in recent years, such as: learning with unknown number of experts and competing with all but the top small fraction of experts (Chaudhuri et al., 2009; Chernov and Vovk, 2010); competing with a sequence of different combinations of the experts (Herbster and Warmuth, 2001; Cesa-Bianchi et al., 2012); learning with experts who provide confidence-rated advice (Blum and Mansour, 2007); and achieving much smaller regret when the problem is "easy" while still ensuring worst-case robustness (de Rooij et al., 2014; Van Erven et al., 2014; Gaillard et al., 2014). Different algorithms were proposed separately to solve these problems to some extent. In this work, we essentially provide *one*

single parameter-free algorithm that achieves all these goals with absolutely no prior information and significantly improved results in some cases.

Our algorithm is a variant of Chaudhuri et al. (2009)'s NormalHedge algorithm, and more specifically is an improved version of NormalHedge.DT (Luo and Schapire, 2014). We call it Adaptive NormalHedge (or AdaNormalHedge for short). NormalHedge and NormalHedge.DT provide guarantees for the so-called ϵ-quantile regret simultaneously for any ϵ, which essentially corresponds to competing with a uniform distribution over the top ϵ-fraction of experts. Our new algorithm improves NormalHedge.DT from two aspects (Section 3):

1. AdaNormalHedge can compete with not just the competitor of the specific form mentioned above, but indeed any unknown fixed competitor simultaneously, with a regret in terms of the relative entropy between the competitor and the player's prior belief of the experts.

2. AdaNormalHedge ensures a new regret bound in terms of the *cumulative magnitude of the instantaneous regrets*, which is always at most the bound for NormalHedge.DT (or NormalHedge). Moreover, the power of this new form of regret is almost the same as the second order bound introduced in a recent work by Gaillard et al. (2014). Specifically, it implies 1) a small regret when the loss of the competitor is small and 2) an almost constant regret when the losses are generated randomly with a gap in expectation.

Our results resolve the open problem asked in Chaudhuri et al. (2009) and Chernov and Vovk (2010) on whether a better ϵ-quantile regret in terms of the loss of the expert instead of the horizon can be achieved. In fact, our results are even better and more general.

AdaNormalHedge is a simple and truly parameter-free algorithm. Indeed, it does not even need to know the number of experts in some sense. To illustrate this idea, in Section 4 we extend the algorithm and results to a setting where experts provide confidence-rated advice (Blum and Mansour, 2007). We then focus on a special case of this setting called the sleeping expert problem (Blum, 1997; Freund et al., 1997), where the number of "awake" experts is dynamically changing and the total number of underlying experts is indeed unknown. AdaNormalHedge is thus a very suitable algorithm for this problem. To show the power of all the abovementioned properties of AdaNormalHedge, we study the following two examples of the sleeping expert problem and use AdaNormalHedge to significantly improve previous work.

The first example is adaptive regret, that is, regret on any time interval, introduced by Hazan and Seshadhri (2007). This can be reduced to a sleeping expert problem by adding a new copy of each original expert on each round (Freund et al., 1997; Koolen et al., 2012). Thus, the total number of sleeping experts is not fixed. When some information on this interval is known (such as the length, the loss of the competitor on this interval, etc), several algorithms achieve optimal regret (Hazan and Seshadhri, 2007; Cesa-Bianchi et al., 2012). However, when no prior information is available, all previous work gives suboptimal bounds. We apply AdaNormalHedge to this problem. The resulting algorithm, which we called AdaNormalHedge.TV, enjoys the optimal adaptive regret in not only the adversarial case but also the stochastic case due to the properties of AdaNormalHedge.

We then extend the results to the problem of tracking the best experts where the player needs to compete with the best partition of the whole process and the best experts on each of these partitions (Herbster and Warmuth, 1995; Bousquet and Warmuth, 2003). This resolves one of the open problems in Warmuth and Koolen (2014) on whether a single algorithm can achieve optimal shifting regret for both adversarial and stochastic losses. Note that although recent work by Sani et al.

(2014) also solves this open problem in some sense, their method requires knowing the number of partitions and other information ahead of time and also gives a worse bound for stochastic losses, while AdaNormalHedge.TV is completely parameter-free and gives optimal bounds.

We finally consider the most general case where the competitor varies over time with no constraints, which subsumes the previous two examples (adaptive regret and shifting regret). This problem was introduced in Herbster and Warmuth (2001) and later generalized by Cesa-Bianchi et al. (2012). Their algorithm (fixed share) also requires knowing some information on the sequence of competitors to optimally tune parameters. We avoid this issue by showing that while this problem seems more general and difficult, it is in fact *equivalent to its special case*: achieving adaptive regret. This equivalence theorem is independent of the concrete algorithms and may be of independent interest. Applying this result, we show that without any parameter tuning, AdaNormalHedge.TV automatically achieves a bound comparable to the one achieved by the optimally tuned fixed share algorithm when competing with time-varying competitors.

Concrete results and detailed comparisons on this first example can be found in Section 5. To sum up, AdaNormalHedge.TV is an algorithm that is simultaneously adaptive in the number of experts, the competitors and the way the losses are generated.

The second example we provide is predicting almost as well as the best pruning tree (Helmbold and Schapire, 1997), which was also shown to be reducible to a sleeping expert problem (Freund et al., 1997). Previous work either only considered the log loss setting, or assumed prior information on the best pruning tree is known. Using AdaNormalHedge, we again provide better or comparable bounds without knowing any prior information. In fact, due to the adaptivity of AdaNormalHedge in the number of experts, our regret bound depends on the total number of distinct traversed edges so far, instead of the total number of edges of the decision tree as in Freund et al. (1997) which could be exponentially larger. Concrete comparisons can be found in Section 6.

Related work. While competing with any unknown competitor simultaneously is relatively easy in the log loss setting (Littlestone and Warmuth, 1994; Adamskiy et al., 2012; Koolen et al., 2012), it is much harder in the bounded loss setting studied here. The well-known exponential weights algorithm gives the optimal results only when the learning rate is optimally tuned in terms of the competitor (Freund and Schapire, 1999). Chernov and Vovk (2010) also studied ϵ-quantile regret, but no concrete algorithm was provided. Several work considers competing with unknown competitors in a different unconstrained linear optimization setting (Streeter and Mcmahan, 2012; Orabona, 2013; McMahan and Orabona, 2014; Orabona, 2014). Jadbabaie et al. (2015) studied general adaptive online learning algorithms against time-varying competitors, but with different and incomparable measurement of the hardness of the problem. As far as we know, none of the existing algorithms enjoys all the nice properties discussed in this work at the same time as our algorithms do.

2. The Expert Problem and NormalHedge.DT

In the expert problem, on each round $t = 1, \dots, T$: the player first chooses a distribution $\mathbf{p}_t$ over N experts, then the adversary decides each expert's loss $\ell_{t,i} \in [0, 1]$, and reveals these losses to the player. At the end of this round, the player suffers the weighted average loss $\hat{\ell}_t = \mathbf{p}_t \cdot \boldsymbol{\ell}_t$ with $\boldsymbol{\ell}_t = (\ell_{t,1}, \dots, \ell_{t,N})$. We denote the *instantaneous regret* to expert i on round t by $r_{t,i} = \hat{\ell}_t - \ell_{t,i}$, the cumulative regret by $R_{t,i} = \sum_{\tau=1}^{t} r_{\tau,i}$, and the cumulative loss by $L_{t,i} = \sum_{\tau=1}^{t} \ell_{\tau,i}$. Throughout the paper, a bold letter denotes a vector with N corresponding coordinates. For example, $\mathbf{r}_t$, $\mathbf{R}_t$ and $\mathbf{L}_t$ represent $(r_{t,1}, \dots, r_{t,N})$, $(R_{t,1}, \dots, R_{t,N})$ and $(L_{t,1}, \dots, L_{t,N})$ respectively.

Usually, the goal of the player is to minimize the regret to the best expert, that is, $\max_i R_{T,i}$. Here we consider a more general case where the player wants to minimize the regret to an arbitrary convex combination of experts: $R_T(\mathbf{u}) = \sum_{t=1}^T \mathbf{u} \cdot \mathbf{r}_t$ where the competitor $\mathbf{u}$ is a fixed unknown distribution over the experts. In other words, this regret measures the difference between the player's loss and the loss that he would have suffered if he used a constant strategy $\mathbf{u}$ all the time. Clearly, $R_T(\mathbf{u})$ can be written as $\mathbf{u} \cdot \mathbf{R}_T$ and can then be upper bounded appropriately by a bound on each $R_{T,i}$ (for example, $\max_i R_{T,i}$). However, our goal is to get a better and more refined bound on $R_T(\mathbf{u})$ that depends on $\mathbf{u}$. More importantly, we aim to achieve this without knowing the competitor $\mathbf{u}$ ahead of time. When it is clear from the context, we drop the subscript T in $R_T(\mathbf{u})$.

In fact, in Section 5, we will consider an even more general notion of regret introduced in Herbster and Warmuth (2001), where we allow the competitor to vary over time and to have different scales. Specifically, let $\mathbf{u}_1, \dots, \mathbf{u}_T$ be T different vectors with N nonnegative coordinates (denoted by $\mathbf{u}_{1:T}$). Then the regret of the player to this sequence of competitors is $R(\mathbf{u}_{1:T}) = \sum_{t=1}^T \mathbf{u}_t \cdot \mathbf{r}_t$. If all these competitors are distributions (which they are not required to be), then this regret captures a very natural and general concept of comparing the player's strategy to any other strategy. Again, we are interested in developing low-regret algorithms that do not need to know any information of this sequence of competitors beforehand.

We briefly describe a recent algorithm for the expert problem, NormalHedge.DT (Luo and Schapire, 2014) (a variant of NormalHedge (Chaudhuri et al., 2009)), before we introduce our new improved variants. On round t, NormalHedge.DT sets $p_{t,i} \propto \exp\left(\frac{[R_{t-1,i}+1]_+^2}{3t}\right) - \exp\left(\frac{[R_{t-1,i}-1]_+^2}{3t}\right)$, where $[x]_+ = \max\{0, x\}$. Let $\epsilon \in (0, 1]$ and competitor $\mathbf{u}_\epsilon^*$ be a distribution that puts all the mass on the $\lceil N\epsilon \rceil$-th best expert, that is, the one that ranks $\lceil N\epsilon \rceil$ among all experts according to their total loss $L_{T,i}$ from the smallest to the largest. Then the regret guarantee for NormalHedge.DT states $R(\mathbf{u}_\epsilon^*) \le O\left(\sqrt{T \ln\left(\frac{\ln T}{\epsilon}\right)}\right)$ simultaneously for all ϵ, which means the algorithm suffers at most this amount of regret for all but an ϵ fraction of the experts. Note that this bound does not depend on N at all. This is the first concrete algorithm with this kind of adaptive property (the original NormalHedge (Chaudhuri et al., 2009) still has a weak dependence on N). In fact, as we will show later, one can even extend the results to any competitor $\mathbf{u}$. Moreover, we will improve NormalHedge.DT so that it has a much smaller regret when the problem is "easy" in some sense.

Notation. We use $[N]$ to denote the set $\{1, \dots, N\}$, Δ_N to denote the simplex of all distributions over $[N]$, and $\mathrm{RE}(\cdot \,||\, \cdot)$ to denote the relative entropy between two distributions, Also define $\tilde{L}_{t,i} = \sum_{\tau=1}^t [\ell_{\tau,i} - \hat{\ell}_\tau]_+$. Many bounds in this work will be in terms of $\tilde{L}_{T,i}$, which is always at most $L_{T,i}$ since trivially $[\ell_{t,i} - \hat{\ell}_t]_+ \le \ell_{t,i}$. We consider "log log" terms to be nearly constant, and use $\hat{O}()$ notation to hide these terms. Indeed, as pointed out by Chernov and Vovk (2010), $\ln \ln x$ is smaller than 4 even when x is as large as the age of the universe expressed in microseconds ($\approx 4.3 \times 10^{17}$).

3. A New Algorithm: AdaNormalHedge

We start by writing NormalHedge.DT in a general form. We define *potential function* $\Phi(R, C) = \exp\left(\frac{[R]_+^2}{3C}\right)$ with $\Phi(0, 0)$ defined to be 1, and also a weight function with respect to this potential: $w(R, C) = \frac{1}{2}\left(\Phi(R+1, C+1) - \Phi(R-1, C+1)\right)$. Then the prediction of NormalHedge.DT is simply to set $p_{t,i}$ to be proportional to $w(R_{t-1,i}, C_{t-1})$ where $C_t = t$ for all t. Note that C_t is closely related to the regret. In fact, the regret is roughly of order $\sqrt{C_T}$ (ignoring the log term). Therefore,

Algorithm 1 AdaNormalHedge

Input: A prior distribution $\mathbf{q} \in \Delta_N$ over experts (uniform if no prior available).
Initialize: $\forall i \in [N], R_{0,i} = 0, C_{0,i} = 0$.
for $t = 1$ **to** T **do**
  Predict[1] $p_{t,i} \propto q_i w(R_{t-1,i}, C_{t-1,i})$.
  Adversary reveals loss vector $\boldsymbol{\ell}_t$ and player suffers loss $\hat{\ell}_t = \mathbf{p}_t \cdot \boldsymbol{\ell}_t$.
  Set $\forall i \in [N], r_{t,i} = \hat{\ell}_t - \ell_{t,i}, R_{t,i} = R_{t-1,i} + r_{t,i}, C_{t,i} = C_{t-1,i} + |r_{t,i}|$.
end for

in order to get an expert-wise and more refined bound, we replace C_t by $C_{t,i}$ for each expert so that it captures some useful information for each expert i. There are several possible choices for $C_{t,i}$ (discussed at the end of Appendix A), but for now we focus on the one used in our new algorithm: $C_{t,i} = \sum_{\tau=1}^{t} |r_{\tau,i}|$, that is, the cumulative magnitude of the instantaneous regrets up to time t. We call this algorithm AdaNormalHedge and summarize it in Algorithm 1. Note that we even allow the player to have a prior distribution $\mathbf{q}$ over the experts, which will be useful in some applications as we will see in Section 5. The theoretical guarantee of AdaNormalHedge is stated below.

Theorem 1 *The regret of AdaNormalHedge to any competitor $\mathbf{u} \in \Delta_N$ is bounded as follows:*

$$R(\mathbf{u}) \le \sqrt{3(\mathbf{u} \cdot \mathbf{C}_T)\left(\mathrm{RE}(\mathbf{u} \,||\, \mathbf{q}) + \ln B + \ln(1 + \ln N)\right)} = \hat{O}\left(\sqrt{(\mathbf{u} \cdot \mathbf{C}_T)\mathrm{RE}(\mathbf{u} \,||\, \mathbf{q})}\right), \quad (1)$$

where $\mathbf{C}_T = (C_{T,1}, \ldots, C_{T,N})$, $B = 1 + \frac{3}{2}\sum_i q_i \left(1 + \ln(1 + C_{T,i})\right) \le \frac{5}{2} + \frac{3}{2}\ln(1+T)$. Moreover, if $\mathbf{u}$ is a uniform distribution over a subset of $[N]$, then the regret can be improved to

$$R(\mathbf{u}) \le \sqrt{3(\mathbf{u} \cdot \mathbf{C}_T)\left(\mathrm{RE}(\mathbf{u} \,||\, \mathbf{q}) + \ln B + 1\right)}. \quad (2)$$

Before we prove this theorem (see sketch at the end of this section and complete proof in Appendix A), we discuss some implications of the regret bounds and why they are interesting. First of all, the relative entropy term $\mathrm{RE}(\mathbf{u} \,||\, \mathbf{q})$ captures how close the player's prior is to the competitor. A bound in terms of $\mathrm{RE}(\mathbf{u} \,||\, \mathbf{q})$ can be obtained, for example, using the classic exponential weights algorithm but requires carefully tuning the learning rate as a function of $\mathbf{u}$. Without knowing $\mathbf{u}$, as far as we know, AdaNormalHedge is the only algorithm that can achieve this.[2]

On the other hand, if $\mathbf{q}$ is a uniform distribution, then using bound (2) and the fact $C_{T,i} \le T$, we get an ϵ-quantile regret bound similar to the one of NormalHedge.DT: $R(\mathbf{u}^*_\epsilon) \le R(\mathbf{u}_{S_\epsilon}) \le \sqrt{3T\left(\ln\left(\frac{1}{\epsilon}\right) + \ln B + 1\right)}$ where $\mathbf{u}_{S_\epsilon}$ is uniform over the top $\lceil N\epsilon \rceil$ experts in terms of their total loss $L_{T,i}$.

However, the power of a bound in terms of $\mathbf{C}_T$ is far more than this. Gaillard et al. (2014) introduced a new second order bound that implies much smaller regret when the problem is easy. It turns out that our seemingly weaker first order bound is also enough to get the exact same results! We state these implications in the following theorem which is essentially a restatement of Theorems 9 and 11 of Gaillard et al. (2014) with weaker conditions.

Theorem 2 *Suppose an expert algorithm guarantees $R(\mathbf{u}) \le \sqrt{(\mathbf{u} \cdot \mathbf{C}_T)A(\mathbf{u})}$ where $A(\mathbf{u})$ is some function of $\mathbf{u}$. Then it also satisfies the following:*

[1] If $\mathbf{p}_{t,i} \propto 0$ happens for all i, predict arbitrarily.

[2] In fact, one can also derive similar bounds for NormalHedge and NormalHedge.DT using our analysis. See discussion at the end of Appendix A.

1. *Recall* $L_{T,i} = \sum_{t=1}^{T} \ell_{t,i}$ *and* $\tilde{L}_{T,i} = \sum_{t=1}^{T} [\ell_{t,i} - \hat{\ell}_t]_+$. *We have*

$$R(\mathbf{u}) \leq \sqrt{2(\mathbf{u} \cdot \tilde{\mathbf{L}}_T) A(\mathbf{u})} + A(\mathbf{u}) \leq \sqrt{2(\mathbf{u} \cdot \mathbf{L}_T) A(\mathbf{u})} + A(\mathbf{u}).$$

2. *Suppose the loss vector* $\boldsymbol{\ell}_t$*'s are independent random variables and there exists an* i^* *and some* $\alpha \in (0, 1]$ *such that* $\mathbb{E}[\ell_{t,i} - \ell_{t,i^*}] \geq \alpha$ *for any* t *and* $i \neq i^*$. *Let* $\mathbf{e}_{i^*}$ *be a distribution that puts all the mass on expert* i^*. *Then we have* $\mathbb{E}[R_{T,i^*}] \leq \frac{A(\mathbf{e}_{i^*})}{\alpha}$, *and with probability at least* $1 - \delta$, $R_{T,i^*} \leq \hat{O}\left(\frac{A(\mathbf{e}_{i^*})}{\alpha} + \frac{1}{\alpha}\sqrt{A(\mathbf{e}_{i^*}) \ln \frac{1}{\delta}}\right)$.

The proof of Theorem 2 is based on the same idea as in Gaillard et al. (2014), and is included in Appendix B for completeness. For AdaNormalHedge, the term $A(\mathbf{u})$ is $3(\mathrm{RE}(\mathbf{u} \,||\, \mathbf{q}) + \ln B + \ln(1 + \ln N))$ in general (or smaller for special $\mathbf{u}$ as stated in Theorem 1). Applying Theorem 2 we have $R(\mathbf{u}) = \hat{O}\left(\sqrt{(\mathbf{u} \cdot \tilde{L}_T)\mathrm{RE}(\mathbf{u} \,||\, \mathbf{q})}\right)$.[3] Specifically, if $\mathbf{q}$ is uniform and assuming without loss of generality that $L_{T,1} \leq \cdots \leq L_{T,N}$, then by a similar argument, we have for AdaNormalHedge, $R(\mathbf{u}_\epsilon^*) \leq \hat{O}\left(\sqrt{\tilde{L}_{T,\lceil N\epsilon \rceil} \ln\left(\frac{1}{\epsilon}\right)}\right)$ for any ϵ. This answers the open question (in the affirmative) asked by Chaudhuri et al. (2009) and Chernov and Vovk (2010) on whether an improvement for small loss can be obtained for ϵ-quantile regret without knowing ϵ.

On the other hand, when we are in a stochastic setting as stated in Theorem 2, AdaNormalHedge ensures $R_{T,i^*} \leq \hat{O}\left(\frac{1}{\alpha} \ln\left(\frac{1}{q_{i^*}}\right)\right)$ in expectation (or with high probability with an extra confidence term), which does not grow with T. Therefore, the new regret bound in terms of $\mathbf{C}_T$ actually leads to significant improvements compared to NormalHedge.DT.

Comparison to Adapt-ML-Prod (Gaillard et al., 2014). Adapt-ML-Prod enjoys a second order bound in terms of $\sum_{t=1}^{T} r_{t,i}^2$, which is always at most the term $\sum_{t=1}^{T} |r_{t,i}|$ appeared in our bounds.[4] However, on one hand, as discussed above, these two bounds have the same improvements when the problem is easy in several senses; on the other hand, Adapt-ML-Prod does not provide a bound in terms of $\mathrm{RE}(\mathbf{u} \,||\, \mathbf{q})$ for an unknown $\mathbf{u}$. In fact, as discussed at the end of Section A.3 of Gaillard et al. (2014), Adapt-ML-Prod cannot improve by exploiting a good prior $\mathbf{q}$ (or at least its current analysis cannot). Specifically, while the regret for AdaNormalHedge does not have an explicit dependence on N and is much smaller when the prior $\mathbf{q}$ is close to the competitor $\mathbf{u}$, the regret for Adapt-ML-Prod always has a $\ln N$ multiplicative term for $\sum_{t=1}^{T} r_{t,i}^2$, which means even a good prior results in the same regret as a uniform prior! More advantages of AdaNormalHedge over Adapt-ML-Prod will be discussed in concrete examples in the following sections.

Proof sketch of Theorem 1. The main idea is to show that the weighted sum of potentials does not increase much on each round. In fact, we show that the final potential $\sum_{i=1}^{N} q_i \Phi(R_{T,i}, C_{T,i})$ is exactly bounded by B (defined in Theorem 1). From this, assuming without loss of generality that $q_1 \Phi(R_{T,1}, C_{T,1}) \geq \cdots \geq q_N \Phi(R_{T,N}, C_{T,N})$, we have $q_i \Phi(R_{T,i}, C_{T,i}) \leq \frac{B}{i}$ for all i, which, by solving for $R_{T,i}$, gives $R_{T,i} \leq \sqrt{3C_{T,i} \ln\left(\frac{B}{iq_i}\right)}$. Multiplying both sides by u_i, summing over N and applying the Cauchy-Schwarz inequality, we arrive at $R(\mathbf{u}) \leq \sqrt{3(\mathbf{u} \cdot \mathbf{C}_T)(D(\mathbf{u} \,||\, \mathbf{q}) + \ln B)}$,

[3] We see $O(\mathrm{RE}(\mathbf{u} \,||\, \mathbf{q}))$ as a minor term and hide it in the big O notation, which is not completely rigorous but will ease the presentation. Same thing happens for other first order bounds in this work.

[4] We briefly discuss the difficulty of getting a similar second order bound for our algorithm at the end of Appendix A.

where we define $D(\mathbf{u} \,||\, \mathbf{q}) = \sum_{i=1}^{N} u_i \ln\left(\frac{1}{iq_i}\right)$. It remains to show that $D(\mathbf{u} \,||\, \mathbf{q})$ and $\mathrm{RE}(\mathbf{u} \,||\, \mathbf{q})$ are close by standard analysis and Stirling's formula.

4. Confidence-rated Advice and Sleeping Experts

In this section, we generalize AdaNormalHedge to deal with experts that make confidence-rated advice, a setting that subsumes many interesting applications as studied by Blum (1997) and Freund et al. (1997). In this general setting, on each round t, each expert first reports its *confidence* $\mathcal{I}_{t,i} \in [0, 1]$ for the current task. The player then predicts $\mathbf{p}_t$ as usual with an extra yet natural restriction that if $\mathcal{I}_{t,i} = 0$ then $p_{t,i} = 0$. That is, the player has to ignore those experts who abstain from making advice (by reporting zero confidence). After that, the loss $\ell_{t,i}$ for those experts who did not abstain (i.e. $\mathcal{I}_{t,i} \neq 0$) are revealed and the player still suffers loss $\hat{\ell}_t = \mathbf{p}_t \cdot \boldsymbol{\ell}_t$. We redefine the instantaneous regret $r_{t,i}$ to be $\mathcal{I}_{t,i}(\hat{\ell}_t - \ell_{t,i})$, that is, the difference between the loss of the player and expert i weighted by the confidence. The goal of the player is, as before, to minimize cumulative regret to any competitor $\mathbf{u}$: $R(\mathbf{u}) = \sum_{t \le T} \mathbf{u} \cdot \mathbf{r}_t$. Clearly, the classic expert problem that we have studied in previous sections is just a special case of this general setting with $\mathcal{I}_{t,i} = 1$ for all t and i.

Moreover, with this general form of $r_{t,i}$, AdaNormalHedge can be used to deal with this general setting with only one simple change of scaling the weights by the confidence:

$$p_{t,i} \propto q_i \mathcal{I}_{t,i} w(R_{t-1,i}, C_{t-1,i}), \tag{3}$$

where $R_{t,i}$ and $C_{t,i}$ is still defined to be $\sum_{\tau=1}^{t} r_{\tau,i}$ and $\sum_{\tau=1}^{t} |r_{\tau,i}|$ respectively. The constraint $\mathcal{I}_{t,i} = 0 \Rightarrow p_{t,i} = 0$ is clearly satisfied. In fact, Algorithm 1 can be seen as a special case of this general form of AdaNormalHedge with $I_{t,i} \equiv 1$. Furthermore, the regret bounds in Theorem 1 still hold without any changes, which are summarized below (proof deferred to Appendix A).

Theorem 3 *For the confidence-rated expert problem, regret bounds* (1) *and* (2) *still hold for general AdaNormalHedge (Eq.* (3)*).*

Previously, Gaillard et al. (2014) studied a general reduction from an expert algorithm to a confidence-rated expert algorithm. Applying those results here gives the exact same algorithm and regret guarantee mentioned above. However, we point out that the general reduction is not always applicable. Specifically, it is invalid if there is an unknown number of experts in the confidence-rated setting (explained more in the next paragraph) while the expert algorithm in the standard setting requires knowing the number of experts as a parameter. This is indeed the case for most algorithms (including Adapt-ML-Prod and even the original NormalHedge by Chaudhuri et al. (2009)). AdaNormalHedge naturally avoids this problem since it does not depend on N at all.

Sleeping Experts. We are especially interested in the case when $\mathcal{I}_{t,i} \in \{0, 1\}$, also called the specialist/sleeping expert problem where $\mathcal{I}_{t,i} = 0$ means that expert i is "asleep" for round t and not making any advice. This is a natural setting where the total number of experts is unknown ahead of time. Indeed, the number of awake experts can be dynamically changing over time. An expert that has never appeared before should be thought of as being asleep for all previous rounds.

AdaNormalHedge is a very suitable algorithm to deal with this case due to its independence of the total number of experts. If an expert i appears for the first time on round t, then by definition it will naturally start with $R_{t-1,i} = 0$ and $C_{t-1,i} = 0$. Although we state the prior q as a distribution,

which seems to require knowing the total number of experts, it is not an issue algorithmically since $\mathbf{q}$ is only used to scale the unnormalized weights (Eq. (3)). For example, if we want $\mathbf{q}$ to be a uniform distribution over N experts where N is unknown beforehand, then to run AdaNormalHedge we can simply treat q_i in Eq. (3) to be 1 for all i, which clearly will not change the behavior of the algorithm anyway. In this case, if we let N_t denote the total number of distinct experts that have been seen up to time t and the competitor $\mathbf{u}$ concentrates on any of these experts, then the relative entropy term in the regret (up to time t) will be $\ln N_t$ (instead of $\ln N$), which is changing over time.

Using the adaptivity of AdaNormalHedge in both the number of experts and the competitor, we provide improved results for two instances of the sleeping expert problem in the next two sections.

5. Time-Varying Competitors

In this section, we study a more challenging goal of competing with time-varying competitors in the standard expert setting (that is, each expert is always awake and again $r_{t,i} = \hat{\ell}_t - \ell_{t,i}$), which turns out to be reducible to a sleeping expert problem. Results for this section are summarized in Table 1.

5.1. Special Cases: Adaptive Regret and Tracking the Best Expert

We start from a special case: adaptive regret, introduced by Hazan and Seshadhri (2007) to better capture changing environments. Formally, consider any time interval $t = t_1, \dots, t_2$, and let $R_{[t_1,t_2],i} = \sum_{t=t_1}^{t_2} r_{t,i}$ be the regret to expert i on this interval (similarly define $L_{[t_1,t_2],i} = \sum_{t=t_1}^{t_2} \ell_{t,i}$ and $C_{[t_1,t_2],i} = \sum_{t=t_1}^{t_2} |r_{t,i}|$). The goal of the player is to obtain relatively small regret on any interval. Freund et al. (1997) essentially introduced a way to reduce this problem to a sleeping expert problem, which was later improved by Adamskiy et al. (2012). Specifically, for every pair of time t and expert i, we create a sleeping expert, denoted by (t, i), who is only awake after (and including) round t and since then suffers the same loss as the original expert i. So we have Nt sleeping experts in total on round t. The prediction $p_{t,i}$ is set to be the sum of all the weights of sleeping expert (τ, i) $(\tau = 1, \dots, t)$. It is clear that doing this ensures that the cumulative regret up to time t_2 with respect to sleeping expert (t_1, i) is exactly $R_{[t_1,t_2],i}$ in the original problem.

This is a sleeping expert problem for which AdaNormalHedge is very suitable, since the number of sleeping experts keeps increasing and the total number of experts is in fact unknown if the horizon T is unknown. Theorem 3 implies that the resulting algorithm gives the following adaptive regret:

$$R_{[t_1,t_2],i} = \hat{O}\left(\sqrt{\left(\sum_{t=t_1}^{t_2} |r_{t,i}|\right) \ln\left(\frac{1}{q_{(t_1,i)}}\right)}\right) = \hat{O}\left(\sqrt{\left(\sum_{t=t_1}^{t_2} |r_{t,i}|\right) \ln\left(Nt_1\right)}\right),$$

where $\mathbf{q}$ is a prior over the Nt_2 experts and the last step is by setting the prior to be $q_{(t,i)} \propto 1/t^2$ for all t and i.[5] This prior is better than a simple uniform distribution which leads to a term $\ln(Nt_2)$ instead of $\ln(Nt_1)$. We call this algorithm AdaNormalHedge.TV.[6] To be concrete, on round t AdaNormalHedge.TV predicts $p_{t,i} \propto \sum_{\tau=1}^{t} \frac{1}{\tau^2} w\left(R_{[\tau,t-1],i}, C_{[\tau,t-1],i}\right)$.

Again, Theorem 2 can be applied to get a more interpretable bound $\hat{O}\left(\sqrt{\tilde{L}_{[t_1,t_2],i} \ln\left(Nt_1\right)}\right)$ where $\tilde{L}_{[t_1,t_2],i} = \sum_{t=t_1}^{t_2} [\ell_{t,i} - \hat{\ell}_t]_+ \le L_{[t_1,t_2],i}$, and a much smaller bound $\hat{O}\left(\frac{\ln(Nt_1)}{\alpha}\right)$ if the losses are stochastic on interval $[t_1, t_2]$ in the sense stated in Theorem 2.

[5] Note that as discussed before, the fact that t_2 is unknown and thus $\mathbf{q}$ is unknown does not affect the algorithm.

[6] "TV" stands for "time-varying".

One drawback of AdaNormalHedge.TV is that its time complexity per round is $O(Nt)$ and the overall space is $O(NT)$. However, the data streaming technique used in Hazan and Seshadhri (2007) can be directly applied here to reduce the time and space complexity to $O(N \ln t)$ and $O(N \ln T)$ respectively, with only an extra multiplicative $O(\sqrt{\ln(t_2 - t_1)})$ factor in the regret.

Tracking the best expert. In fact, AdaNormalHedge.TV is a solution for one of the open problems proposed by Warmuth and Koolen (2014). Adaptive regret immediately implies the so-called K-shifting regret for the problem of tracking the best expert in a changing environment. Formally, define the K-shifting regret $R_{K\text{-Shift}}$ to be $\max \sum_{k=1}^{K} R_{[t_{k-1}+1,t_k],i_k}$ where the max is taken over all $i_1, \cdots, i_K \in [N]$ and $0 = t_0 < t_1 < \cdots < t_K = T$. In other words, the player is competing with the best K-partition of the whole game and the best expert on each of these partitions. Let $L^*_{K\text{-Shift}} = \min \sum_{k=1}^{K} L_{[t_{k-1}+1,t_k],i_k}$ be the total loss of such best partition (that is, the min is taken over the same space), and similarly define $\tilde{L}^*_{K\text{-Shift}} = \min \sum_{k=1}^{K} \tilde{L}_{[t_{k-1}+1,t_k],i_k} \le L^*_{K\text{-Shift}}$. Since essentially $R_{K\text{-Shift}}$ is just the sum of K adaptive regrets, using the bounds discussed above and the Cauchy-Schwarz inequality, we conclude that AdaNormalHedge.TV ensures $R_{K\text{-Shift}} = \hat{O}\left(\sqrt{K\tilde{L}^*_{K\text{-Shift}} \ln(NT)}\right)$. Also, if the loss vectors are generated randomly on these K intervals, each satisfying the condition stated in Theorem 2, then the regret is $R_{K\text{-Shift}} = \hat{O}\left(\frac{K \ln(NT)}{\alpha}\right)$ in expectation (high probability bound is similar). These bounds are optimal up to logarithmic factors (Hazan and Seshadhri, 2007). This is exactly what was asked in Warmuth and Koolen (2014): whether there is an algorithm that can do optimally for both adversarial and stochastic losses in the problem of tracking the best expert. AdaNormalHedge.TV achieves this goal without knowing K or any other information, while the solution provided by Sani et al. (2014) needs to know K , $L^*_{K\text{-Shift}}$ and α to get the same adversarial bound and a worse stochastic bound of order $O(1/\alpha^2)$.

Comparison to previous work. For adaptive regret, the FLH algorithm by Hazan and Seshadhri (2007) treats any standard expert algorithm as a sleeping expert, and has an additive term $O(\sqrt{t_2} \ln t_2)$ in addition to the base algorithm's regret (when no prior information is available), which adds up to a large $O(K\sqrt{T} \ln T)$ term for K-shifting regret. Due to this extra additive regret, FLH also does not enjoy first order bounds nor small regret in the stochastic setting, even if the base algorithm that it builds on provides these guarantees. On the other hand, FLH was proposed to achieve adaptive regret for any general online convex optimization problem. We point out that using AdaNormalHedge as the master algorithm in their framework will give similar improvements as discussed here.

Adapt-ML-Prod is not directly applicable here for the corresponding sleeping expert problem since the total number of experts is unknown.

Another well-studied algorithm for this problem is "fixed share". Several works on fixed share for the simpler "log loss" setting were studied before (Herbster and Warmuth, 1998; Bousquet and Warmuth, 2003; Adamskiy et al., 2012; Koolen et al., 2012). Cesa-Bianchi et al. (2012) studied a generalized fixed share algorithm for the bounded loss setting considered here. When $t_2 - t_1$ and $L_{[t_1,t_2],i}$ are known, their algorithm ensures $R_{[t_1,t_2],i} = O\left(\sqrt{L_{[t_1,t_2],i} \ln(N(t_2 - t_1))}\right)$ for adaptive regret, and when K, T and $L^*_{K\text{-Shift}}$ are known, they have $R_{K\text{-Shift}} = O\left(\sqrt{KL^*_{K\text{-Shift}} \ln(NT/K)}\right)$. No better result is provided for the stochastic setting. More importantly, when no prior information is known, which is the case in practice, the best results one can extract from their analysis are $R_{[t_1,t_2],i} = O(\sqrt{t_2 \ln(Nt_2)})$ and $R_{K\text{-Shift}} = O(K\sqrt{T \ln(NT)})$, which are much worse than our results.

Table 1: Comparison of different algorithms for time-varying competitors[7]

Algorithm	$R_{[t_1,t_2],i}$	$R_{K\text{-Shift}}$	$R(\mathbf{u}_{1:T})$
AdaNormalHedge.TV (this work)	$\sqrt{\tilde{L}_{[t_1,t_2],i}\ln(Nt_1)}$	$\sqrt{K\tilde{L}^*_{K\text{-Shift}}\ln(NT)}$	$\sqrt{V(\mathbf{u}_{1:T})\tilde{L}(\mathbf{u}_{1:T})\ln(NT)}$
FLH[8] (Hazan and Seshadhri, 2007)	$\sqrt{t_2}\ln t_2$	$K\sqrt{T}\ln T$	unknown
Fixed Share (Cesa-Bianchi et al., 2012)	$\sqrt{t_2\ln(Nt_2)}$	$K\sqrt{T\ln(NT)}$	unknown

5.2. General Time-Varying Competitors

We finally discuss the most general goal: compete with different $\mathbf{u}_t$ on different rounds. Recall $R(\mathbf{u}_{1:T}) = \sum_{t=1}^T \mathbf{u}_t \cdot \mathbf{r}_t$ where $\mathbf{u}_t \in \mathbb{R}_+^N$ for all t (note that $\mathbf{u}_t$ does not even have to be a distribution). Clearly, adaptive regret and K-shifting regret are special cases of this general notion. Intuitively, how large this regret is should be closely related to how much the competitor's sequence $\mathbf{u}_{1:T}$ varies. Cesa-Bianchi et al. (2012) introduced a distance measurement to capture this variation: $V(\mathbf{u}_{1:T}) = \sum_{t=1}^T \sum_{i=1}^N [u_{t,i} - u_{t-1,i}]_+$ where we define $u_{0,i} = 0$ for all i. Also let $\|\mathbf{u}_{1:T}\| = \sum_{t=1}^T \|\mathbf{u}_t\|_1$ and $L(\mathbf{u}_{1:T}) = \sum_{t=1}^T \mathbf{u}_t \cdot \boldsymbol{\ell}_t$. Fixed share is shown to ensure the following regret (Cesa-Bianchi et al., 2012): $R(\mathbf{u}_{1:T}) = O\big(\sqrt{V(\mathbf{u}_{1:T})L(\mathbf{u}_{1:T})\ln\left(N\|\mathbf{u}_{1:T}\|/V(\mathbf{u}_{1:T})\right)}\big)$ when $V(\mathbf{u}_{1:T}), L(\mathbf{u}_{1:T})$ and $\|\mathbf{u}_{1:T}\|$ are known. No result was provided otherwise.[9] Here, we show that our parameter-free algorithm AdaNormalHedge.TV actually achieves almost the same bound without knowing any information beforehand. Moreover, while the results in Cesa-Bianchi et al. (2012) are specific for the fixed share algorithm, we prove the following results which are independent of the concrete algorithms and may be of independent interest.

Theorem 4 *Suppose an expert algorithm ensures $R_{[t_1,t_2],i} \le \sqrt{A\sum_{t=t_1}^{t_2} z_{t,i}}$ for any t_1, t_2 and i, where $z_{t,i} \ge 0$ can be anything depending on t and i (e.g. $|r_{t,i}|$, $[\ell_{t,i} - \hat{\ell}_t]_+$, $\ell_{t,i}$ or constant 1), and A is a term independent of t_1, t_2 and i. Then this algorithm also ensures*

$$R(\mathbf{u}_{1:T}) \le \sqrt{AV(\mathbf{u}_{1:T})\sum_{t=1}^T \mathbf{u}_t \cdot \mathbf{z}_t}.$$

Specially, for AdaNormalHedge.TV, plugging $A = \hat{O}(\ln(NT))$ and $z_{t,i} = [\ell_{t,i} - \hat{\ell}_t]_+$ gives $R(\mathbf{u}_{1:T}) = \hat{O}\Big(\sqrt{V(\mathbf{u}_{1:T})\tilde{L}(\mathbf{u}_{1:T})\ln(NT)}\Big)$, where $\tilde{L}(\mathbf{u}_{1:T}) = \sum_{t=1}^T \sum_{i=1}^N u_{t,i}[\ell_{t,i} - \hat{\ell}_t]_+$.

The key idea of the proof is to rewrite $R(\mathbf{u}_{1:T})$ as a weighted sum of several adaptive regrets in an optimal way (see Appendix C for the complete proof). This theorem tells us that while playing with time-varying competitors seems to be a harder problem, it is in fact not any harder than its special case: achieving adaptive regret on any interval. Although the result is independent of the algorithms, one still cannot derive bounds on $R(\mathbf{u}_{1:T})$ for FLH or fixed share based on their adaptive regret bounds, because when no prior information is available, the bounds on $R_{[t_1,t_2],i}$ for these algorithms are of order $O(\sqrt{t_2})$ instead of $O(\sqrt{t_2 - t_1})$, which is not good enough. We refer the reader to Table 1 for a summary of this section.

[7] For fair comparison, we only consider the case when no prior information (e.g. $t_1, t_2, V(\mathbf{u}_{1:T})$ etc) is known.

[8] The choice of the base algorithm does not matter since the dominating term in the regret comes from FLH itself.

[9] Although in Section 7.3 of Cesa-Bianchi et al. (2012), the authors mentioned online tuning technique for the parameters, it only works for special cases (e.g. adaptive regret).

6. Competing with the Best Pruning Tree

We now turn to our second application on predicting almost as well as the best pruning tree within a template tree. This problem was studied in the context of online learning by Helmbold and Schapire (1997) using the approach of Willems et al. (1993, 1995). It is also called the tree expert problem in Cesa-Bianchi and Lugosi (2006). Freund et al. (1997) proposed a generic reduction from a tree expert problem to a sleeping expert problem. Using this reduction with our new algorithm, we provide much better results compared to previous work (summarized in Table 2).

Specifically, consider a setting where on each round t, the predictor has to make a decision y_t from some convex set $\mathcal{Y}$ given some side information x_t, and then a convex loss function $f_t : \mathcal{Y} \to [0,1]$ is revealed and the player suffers loss $f_t(y_t)$. The predictor is given a *template tree* $\mathcal{G}$ to consult. Starting from the root, each node of $\mathcal{G}$ performs a test on x_t to decide which of its child should perform the next test, until a leaf is reached. In addition to a test, each node (except the root) also makes a prediction based on x_t. A *pruning tree* $\mathcal{P}$ is a tree induced by replacing zero or more nodes (and associated subtrees) of $\mathcal{G}$ by leaves. The prediction of a pruning tree $\mathcal{P}$ given x_t, denoted by $\mathcal{P}(x_t)$, is naturally defined as the prediction of the leaf that x_t reaches by traversing $\mathcal{P}$. The player's goal is thus to predict almost as well as the best pruning tree in hindsight, that is, to minimize $R_{\mathcal{G}} = \sum_{t=1}^T f_t(y_t) - \min_{\mathcal{P}} \sum_{t=1}^T f_t(\mathcal{P}(x_t))$.

The idea of the reduction introduced by Freund et al. (1997) is to view each edge of $\mathcal{G}$ as a sleeping expert (indexed by e), who is awake only when traversed by x_t, and in that case predicts $y_{t,e}$, the same prediction as the child node that it connects to. The predictor runs a sleeping expert algorithm with loss $\ell_{t,e} = f_t(y_{t,e})$, and eventually predicts $y_t = \sum_{e \in E} p_{t,e} y_{t,e}$ where E denotes the set of edges of $\mathcal{G}$ and $p_{t,e}$ ($e \in E$) is the output of the expert algorithm; thus by convexity of f_t, we have $f_t(y_t) \le \sum_{e \in E} p_{t,e} f_t(y_{t,e}) = \hat{\ell}_t$. Note that we only care about the predictions of those awake experts since otherwise $p_{t,e}$ is required to be zero. Now let $\mathcal{P}^*$ be one of the best pruning trees, that is, $\mathcal{P}^* \in \arg\min_{\mathcal{P}} \sum_{t=1}^T f_t(\mathcal{P}(x_t))$, and m be the number of leaves of $\mathcal{P}^*$. In the expert problem, we will set the competitor $\mathbf{u}$ to be a uniform distribution over the m terminal edges (that is, the ones connecting the leaves) of $\mathcal{P}^*$, and the prior $\mathbf{q}$ to be a uniform distribution over all the edges. Since on each round, one and only one of those m experts is awake, and its prediction is exactly $\mathcal{P}^*(x_t)$, we have $R(\mathbf{u}) = \sum_{t=1}^T \frac{1}{m}(\hat{\ell}_t - f_t(\mathcal{P}^*(x_t)))$, and therefore $R_{\mathcal{G}} \le mR(\mathbf{u})$.

It remains to pick a concrete sleeping algorithm to apply. There are two reasons that make AdaNormalHedge very suitable for this problem. First, since m is clearly unknown ahead of time, we are competing with an unknown competitor $\mathbf{u}$, which is exactly what AdaNormalHedge can deal with. Second, the number of awake experts is dynamically changing, and as discussed before, in this case AdaNormalHedge enjoys a regret bound that is adaptive in the the number of experts seen so far. Formally, recall the notation N_t, which in this case represents the total number of distinct traversed edges up to round t. Then by Theorem 3, we have

$$R_{\mathcal{G}} = \hat{O}(m\sqrt{(\mathbf{u} \cdot \mathbf{C}_T)\mathrm{RE}(\mathbf{u} \,||\, \mathbf{q})}) = \hat{O}\left(\sqrt{m\left(\sum_{t=1}^T |\hat{\ell}_t - f_t(\mathcal{P}^*(x_t))|\right)\ln\left(\frac{N_T}{m}\right)}\right),$$

which, by Theorem 2, implies $R_{\mathcal{G}} = \hat{O}\left(\sqrt{m\tilde{L}^* \ln(N_T/m)}\right)$ where $\tilde{L}^* = \sum_{t=1}^T [f_t(\mathcal{P}^*(x_t)) - \hat{\ell}_t]_+$, which is at most the total loss of the best pruning tree $L^* = \sum_{t=1}^T f_t(\mathcal{P}^*(x_t))$. Moreover, the algorithm is efficient: the overall space requirement is $O(N_T)$, and the running time on round t is $O(\|x_t\|_{\mathcal{G}})$ where we use $\|x_t\|_{\mathcal{G}}$ to denote the number of edges that x_t traverses.

Table 2: Comparison of different algorithms for the tree expert problem

Algorithm	$R_{\mathcal{G}}$	Time (per round)	Space (overall)	Need to know L^* and m?
AdaNormalHedge (this work)	$\hat{O}\left(\sqrt{m\tilde{L}^* \ln\left(\frac{N_T}{m}\right)}\right)$	$O\left(\|x_t\|_{\mathcal{G}}\right)$	$O\left(N_T\right)$	No
Adapt-ML-Prod (Gaillard et al., 2014)	$\hat{O}\left(\sqrt{m\tilde{L}^* \ln N}\right)$	$O\left(\|x_t\|_{\mathcal{G}}\right)$	$O\left(N_T\right)$	No
Exponential Weights (Helmbold and Schapire, 1997)	$O\left(\sqrt{mL^*}\right)$	$O\left(d\|x_t\|_{\mathcal{G}}\right)$	$O\left(N_T\right)$	Yes

Comparison to other solutions. The work by Freund et al. (1997) considers a variant of the exponential weights algorithm in a "log loss" setting, and is not directly applicable here (specifically it is not clear how to tune the learning rate appropriately). A better choice is the Adapt-ML-Prod algorithm by Gaillard et al. (2014) (the version for the sleeping expert problem). However, there is still one issue for this algorithm: it does not give a bound in terms of $\mathrm{RE}(\mathbf{u} \,||\, \mathbf{q})$ for an unknown $\mathbf{u}$. So to get a bound on $R(\mathbf{u})$, the best thing to do is to use the definition $R(\mathbf{u}) = \mathbf{u} \cdot \mathbf{R}_T$ and a bound on each $R_{T,i}$. In short, one can verify that Adapt-ML-Prod ensures regret $R(\mathbf{u}) = \hat{O}\left(\sqrt{m\tilde{L}^* \ln N}\right)$ where $N = |E|$ is the total number of edges/experts. We emphasize that N can be much larger than N_T when the tree is huge. Indeed, while N_T is at most T times the depth of $\mathcal{G}$, N could be exponentially large in the depth. The running time and space of Adapt-ML-Prod for this problem, however, is the same as AdaNormalHedge.

We finally compare with a totally different approach (Helmbold and Schapire, 1997), where one simply treats each pruning tree as an expert and run the exponential weights algorithm. Clearly the number of experts is exponentially large, and thus the running time and space are unacceptable by a naive implementation. This issue is avoided by using a clever dynamic programming technique. If L^* and m are known ahead of time, then the regret for this algorithm is $O(\sqrt{mL^*})$ by tuning the learning rate optimally. As discussed in Freund et al. (1997), the linear dependence on m in this bound is much better than the one of the form $O\left(\sqrt{mL^* \ln N}\right)$, which, in the worst case, is linear in m. This was considered as the main drawback of using the sleeping expert approach. However, the bound for AdaNormalHedge is $\hat{O}\left(\sqrt{m\tilde{L}^* \ln(N_T/m)}\right)$, which is much smaller as discussed previously and in fact comparable to $O(\sqrt{mL^*})$. More importantly, L^* and m are unknown in practice. In this case, no sublinear regret is known for this dynamic programming approach, since it relies heavily on the fact that the algorithm is using a fixed learning rate and thus the usual time-varying learning rate methods cannot be applied here. Therefore, although theoretically this approach gives small regret, it is not a practical method. The running time is also slightly worse than the sleeping expert approach. For simplicity, suppose every internal node has d children. Then the time complexity per round is $O(d\|x\|_{\mathcal{G}})$. The overall space requirement is $O(N_t)$, the same as other approaches. Again, see Table 2 for a summary of this section.

Finally, as mentioned in Freund et al. (1997), the sleeping expert approach can be easily generalized to predicting with a decision graph. In that case, AdaNormalHedge still enjoys all the improvements discussed in this section (details omitted).

Acknowledgements. The authors thank Yoav Freund for helpful discussions and the anonymous reviewers for their comments.

References

Dmitry Adamskiy, Wouter M Koolen, Alexey Chernov, and Vladimir Vovk. A closer look at adaptive regret. In *Algorithmic Learning Theory*, pages 290–304, 2012.

Avrim Blum. Empirical support for Winnow and Weighted-Majority algorithms: Results on a calendar scheduling domain. *Machine Learning*, 26(1):5–23, 1997.

Avrim Blum and Yishay Mansour. From external to internal regret. *Journal of Machine Learning Research*, 8:1307–1324, 2007.

Olivier Bousquet and Manfred K. Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3:363–396, 2003.

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, May 1997.

Nicolò Cesa-Bianchi, Pierre Gaillard, Gábor Lugosi, and Gilles Stoltz. Mirror descent meets fixed share (and feels no regret). In *Advances in Neural Information Processing Systems 25*, 2012.

Kamalika Chaudhuri, Yoav Freund, and Daniel Hsu. A parameter-free hedging algorithm. In *Advances in Neural Information Processing Systems 22*, 2009.

Alexey Chernov and Vladimir Vovk. Prediction with advice of unknown number of experts. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, 2010.

Steven de Rooij, Tim van Erven, Peter D. Grünwald, and Wouter M. Koolen. Follow the leader if you can, hedge if you must. *Journal of Machine Learning Research*, 15:1281–1316, 2014.

Yoav Freund and Robert 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, August 1997.

Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999.

Yoav Freund, Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. Using and combining predictors that specialize. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 334–343, 1997.

Pierre Gaillard, Gilles Stoltz, and Tim Van Erven. A second-order bound with excess losses. In *Proceedings of the 27th Annual Conference on Learning Theory*, 2014.

Elad Hazan and C. Seshadhri. Adaptive algorithms for online decision problems. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 14, 2007.

David P. Helmbold and Robert E. Schapire. Predicting nearly as well as the best pruning of a decision tree. *Machine Learning*, 27(1):51–68, April 1997.

Mark Herbster and Manfred Warmuth. Tracking the best expert. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 286–294, 1995.

Mark Herbster and Manfred Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998.

Mark Herbster and Manfred K Warmuth. Tracking the best linear predictor. *The Journal of Machine Learning Research*, 1:281–309, 2001.

Ali Jadbabaie, Alexander Rakhlin, Shahin Shahrampour, and Karthik Sridharan. Online optimization: Competing with dynamic comparators. In *The 18th International Conference on Artificial Intelligence and Statistics*, 2015.

Wouter M Koolen, Dmitry Adamskiy, and Manfred K Warmuth. Putting bayes to sleep. In *Advances in Neural Information Processing Systems 25*, pages 135–143, 2012.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

Haipeng Luo and Robert E. Schapire. A Drifting-Games Analysis for Online Learning and Applications to Boosting. In *Advances in Neural Information Processing Systems 27*, 2014.

H Brendan McMahan and Francesco Orabona. Unconstrained online linear learning in hilbert spaces: Minimax algorithms and normal approximations. In *Proceedings of the 27th Annual Conference on Learning Theory*, 2014.

Francesco Orabona. Dimension-free exponentiated gradient. In *Advances in Neural Information Processing Systems 26*, pages 1806–1814, 2013.

Francesco Orabona. Simultaneous model selection and optimization through parameter-free stochastic learning. In *Advances in Neural Information Processing Systems 27*, 2014.

Amir Sani, Gergely Neu, and Alessandro Lazaric. Exploiting easy data in online optimization. In *Advances in Neural Information Processing Systems 27*, 2014.

Robert E. Schapire. Drifting games. *Machine Learning*, 43(3):265–291, June 2001.

Matthew Streeter and Brendan Mcmahan. No-regret algorithms for unconstrained online convex optimization. In *Advances in Neural Information Processing Systems 25*, pages 2402–2410, 2012.

Tim Van Erven, Wojciech Kotlowski, and Manfred K Warmuth. Follow the leader with dropout perturbations. In *Proceedings of the 27th Annual Conference on Learning Theory*, 2014.

V. G. Vovk. A game of prediction with expert advice. *Journal of Computer and System Sciences*, 56(2):153–173, April 1998.

Manfred K. Warmuth and Wouter M. Koolen. Open problem: Shifting experts on easy data. In *Proceedings of the 27th Annual Conference on Learning Theory*, 2014.

Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. Context tree weighting: A sequential universal source coding procedure for FSMX sources. In *Proceedings 1993 IEEE International Symposium on Information Theory*, page 59, 1993.

Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. The context tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.

Appendix A. Complete proofs of Theorem 1 and 3

The analysis of NormaHedge.DT is based on the idea of converting the expert problem into a drifting game (Schapire, 2001; Luo and Schapire, 2014). Here, we extract and simplify the key idea of their proof and also improve it to form our analysis. We need the following two lemmas. The first one is an improved version of Lemma 2 of Luo and Schapire (2014).

Lemma 5 *For any $R \in \mathbb{R}, C \geq 0$ and $r \in [-1, 1]$, we have*

$$\Phi(R + r, C + |r|) \leq \Phi(R, C) + w(R, C)r + \frac{3|r|}{2(C + 1)}.$$

Proof We first argue that $\Phi(R + r, C + |r|)$, as a function of r, is piecewise-convex on $[-1, 0]$ and $[0, 1]$. Since the value of the function is 1 when $R + r < 0$ and is at least 1 otherwise. It suffices to only consider the case when $R+r \geq 0$. On the interval $[0, 1]$, we can rewrite the exponent (ignoring the constant $\frac{1}{3}$) as:

$$\frac{(R + r)^2}{C + r} = (C + r) + \frac{(R - C)^2}{C + r} + 2(R - C),$$

which is convex in r. Combining with the fact that "if $g(x)$ is convex then $\exp(g(x))$ is also convex" proves that $\Phi(R+r, C+|r|)$ is convex on $[0, 1]$. Similarly when $r \in [-1, 0]$, rewriting the exponent as

$$\frac{(R + r)^2}{C - r} = (C - r) + \frac{(R + C)^2}{C - r} - 2(R + C)$$

completes the argument.

Now define function $f(r) = \Phi(R+r, C+|r|)-w(R, C)r$. Since $f(r)$ is clearly also piecewise-convex on $[-1, 0]$ and $[0, 1]$, we know that the curve of $f(r)$ is below the segment connecting points $(-1, f(-1))$ and $(0, f(0))$ on $[-1, 0]$, and also below the segment connecting points $(1, f(1))$ and $(0, f(0))$ on $[0, 1]$. This can be mathematically expressed as:

$$f(r) \leq \max\{f(0) + (f(0) - f(-1))r, f(0) + (f(1) - f(0))r\} = f(0) + (f(1) - f(0))|r|,$$

where we use the fact $f(-1) = f(1)$. Now by Lemma 2 of Luo and Schapire (2014), we have

$$f(1) - f(0) = \tfrac{1}{2}\left(\Phi(R + 1, C + 1) + \Phi(R - 1, C + 1)\right) - \Phi(R, C) \leq \tfrac{1}{2}\left(\exp\left(\tfrac{4}{3(C+1)}\right) - 1\right),$$

which is at most $\frac{e^{\frac{4}{3}}-1}{2(C+1)}$ since C is nonnegative and $e^x - 1 \leq \frac{e^a-1}{a}x$ for any $x \in [0, a]$. Noting that $e^{\frac{4}{3}} - 1 \leq 3$ completes the proof. ■

The second lemma makes use of Lemma 5 to show that the weighted sum of potentials does not increase much and thus the final potential is relatively small.

Lemma 6 *AdaNormalHedge ensures $\sum_{i=1}^N q_i \Phi(R_{T,i}, C_{T,i}) \leq B = 1+\frac{3}{2}\sum_{i=1}^N q_i \left(1 + \ln(1 + C_{T,i})\right)$.*

Proof First note that since AdaNormalHedge predicts $p_{t,i} \propto q_i w(R_{t-1,i}, C_{t-1,i})$, we have

$$\textstyle\sum_{i=1}^N q_i w(R_{t-1,i}, C_{t-1,i}) r_{t,i} = 0. \tag{4}$$

Now applying Lemma 5 with $R = R_{t-1,i}, C = C_{t-1,i}$ and $r = r_{t,i}$, multiplying the inequality by q_i on both sides and summing over i gives $\sum_{i=1}^N q_i\Phi(R_{t,i}, C_{t,i}) \le \sum_{i=1}^N q_i\Phi(R_{t-1,i}, C_{t-1,i}) + \frac{3}{2}\sum_{i=1}^N \frac{q_i|r_{t,i}|}{C_{t-1,i}+1}$. We then sum over $t \in [T]$ and telescope to show $\sum_{i=1}^N q_i\Phi(R_{T,i}, C_{T,i}) \le 1 + \frac{3}{2}\sum_{i=1}^N q_i \sum_{t=1}^T \frac{|r_{t,i}|}{C_{t-1,i}+1}$. Finally applying Lemma 14 of Gaillard et al. (2014) to show $\sum_{t=1}^T \frac{|r_{t,i}|}{C_{t-1,i}+1} \le 1 + \ln(1 + C_{T,i})$ completes the proof. ■

We are now ready to prove Theorem 1 and Theorem 3.

Proof (of Theorem 1) Assume $q_1\Phi(R_{T,1}, C_{T,1}) \ge \cdots \ge q_N\Phi(R_{T,N}, C_{T,N})$ without loss of generality. Then by Lemma 6, it must be true that $q_i\Phi(R_{T,i}, C_{T,i}) \le \frac{B}{i}$ for all i, which, by solving for $R_{T,i}$, gives $R_{T,i} \le \sqrt{3C_{T,i}\ln\left(\frac{B}{iq_i}\right)}$. Multiplying both sides by u_i, summing over N and applying the Cauchy-Schwarz inequality, we arrive at $R(\mathbf{u}) \le \sum_{i=1}^N \sqrt{3u_iC_{T,i}\cdot u_i\ln\left(\frac{B}{iq_i}\right)} \le \sqrt{3(\mathbf{u}\cdot\mathbf{C}_T)(D(\mathbf{u}\,||\,\mathbf{q}) + \ln B)}$, where we define $D(\mathbf{u}\,||\,\mathbf{q}) = \sum_{i=1}^N u_i\ln\left(\frac{1}{iq_i}\right)$. It remains to show that $D(\mathbf{u}\,||\,\mathbf{q})$ and $\mathrm{RE}(\mathbf{u}\,||\,\mathbf{q})$ are close. Indeed, we have $D(\mathbf{u}\,||\,\mathbf{q}) - \mathrm{RE}(\mathbf{u}\,||\,\mathbf{q}) = \sum_{i=1}^N u_i\ln\left(\frac{1}{iu_i}\right)$, which, by standard analysis, can be shown to reach its maximum when $u_i \propto \frac{1}{i}$ and the maximum value is $\ln\sum_i \frac{1}{i} \le \ln(1+\ln N)$. This completes the proof for Eq. (1).

Finally, when $\mathbf{u}$ is in the special form as described in Theorem 1, we have $D(\mathbf{u}\,||\,\mathbf{q}) - \mathrm{RE}(\mathbf{u}\,||\,\mathbf{q}) = \ln|S| + \frac{1}{|S|}\sum_{i\in S}\ln\left(\frac{1}{i}\right) \le \ln|S| - \frac{1}{|S|}\ln(|S|!)$. By Stirling's formula $x! \ge \sqrt{2\pi x}\left(\frac{x}{e}\right)^x$, we arrive at $D(\mathbf{u}\,||\,\mathbf{q}) - \mathrm{RE}(\mathbf{u}\,||\,\mathbf{q}) \le 1 - (\ln\sqrt{2\pi|S|})/|S| \le 1$, proving Eq. (2). ■

Proof (of Theorem 3) It suffices to point out that $r_{t,i}$ is still in the interval $[-1, 1]$ and Eq. (4) in the proof of Lemma 6 still holds by the new prediction rule Eq. (3). The entire proof for Theorem 1 applies here exactly. ■

The algorithm and the proof can be generalized to $C_{t,i} = \sum_{\tau=1}^t |r_{\tau,i}|^d$ for any $d \in [0, 1]$. Indeed, the only extra work is to prove the convexity of $\Phi(R + r, C + |r|^d)$. When $d = 0$, we recover NormalHedge.DT exactly and get a bound on $R(\mathbf{u})$ for any $\mathbf{u}$ (in terms of $\sqrt{T}$), instead of just $R(\mathbf{u}^*_\epsilon)$ as in the original work. It is clear that $d = 1$ gives the smallest bound, which is why we use it in AdaNormalHedge. The ideal choice, however, should be $d = 2$ so that a second order bound similar to the one of Gaillard et al. (2014) can be obtained. Unfortunately, the function $\Phi(R + r, C + r^2)$ turns out to not always be piecewise-convex, which breaks our analysis. Whether $d = 2$ gives a low-regret algorithm and how to analyze it remain an open question.

Appendix B. Proof of Theorem 2

Proof For the first result, the key observation is $\mathbf{u}\cdot\mathbf{C}_T = R(\mathbf{u}) + 2\mathbf{u}\cdot\tilde{\mathbf{L}}_T$. We only consider the case when $R(\mathbf{u}) \ge 0$ since otherwise the statement is trivial. By the condition we thus have $R(\mathbf{u})^2 \le (R(\mathbf{u}) + 2\mathbf{u}\cdot\tilde{\mathbf{L}}_T)A(\mathbf{u})$, which by solving for $R(\mathbf{u})$ gives

$$R(\mathbf{u}) \le \tfrac{1}{2}(A(\mathbf{u}) + \sqrt{A(\mathbf{u})^2 + 8(\mathbf{u}\cdot\tilde{\mathbf{L}}_T)A(\mathbf{u})}) \le \sqrt{2(\mathbf{u}\cdot\tilde{\mathbf{L}}_T)A(\mathbf{u})} + A(\mathbf{u}),$$

proving the bound we want.

For the second result, let $\mathbb{E}_t$ denote the expectation conditioning on all the randomness up to round t. So by the condition, we have $\mathbb{E}_t[r_{t,i^*}] = \sum_{i=1}^N p_{t,i}\mathbb{E}_t[\ell_{t,i} - \ell_{t,i^*}] \geq \alpha(1 - p_{t,i^*})$, and thus $\mathbb{E}[R_{T,i^*}] \geq \alpha S$ where we define $S = \sum_{t=1}^T \mathbb{E}[1 - p_{t,i^*}]$. On the other hand, by convexity we also have $|r_{t,i^*}| \leq \sum_{i=1}^N p_{t,i}|\ell_{t,i} - \ell_{t,i^*}| \leq 1 - p_{t,i^*}$ and thus $\mathbb{E}[R_{T,i^*}] \leq \mathbb{E}[\sqrt{A(\mathbf{e}_{i^*})C_{T,i^*}}] \leq \sqrt{A(\mathbf{e}_{i^*})S}$ by the concavity of the square root function. Combining the above two statements gives $S \leq \frac{A(\mathbf{e}_{i^*})}{\alpha^2}$, and plugging this back shows $\mathbb{E}[R_{T,i^*}] \leq \frac{A(\mathbf{e}_{i^*})}{\alpha}$. The high probability statement follows from the exact same argument of Gaillard et al. (2014) using a martingale concentration lemma. ■

Appendix C. Proof of Theorem 4

Below we use $[s, t]$ to denote the set $\{s, s+1, \ldots, t-1, t\}$ for $1 \leq s \leq t \leq T$ and $s, t \in \mathbb{N}^+$.

Proof We first fix an expert i and consider the regret to this expert $\sum_{t=1}^T u_{t,i} r_{t,i}$. Let $a_j \geq 0$ and $U_j = [s_j, t_j]$ for $j = 1, \ldots, M$ be M positive numbers and M corresponding time intervals such that $u_{t,i} = \sum_{j=1}^M a_j \mathbf{1}\{t \in U_j\}$. Note that this is always possible, with a trivial choice being $a_j = u_{t,j}$, $s_j = t_j = j$ and $M = T$; we will however need a more sophisticated construction specified later. By the adaptive regret guarantee, we have

$$\sum_{t=1}^T u_{t,i} r_{t,i} = \sum_{j=1}^M a_j \sum_{t=1}^T \mathbf{1}\{t \in U_j\} r_{t,i} = \sum_{j=1}^M a_j R_{[s_j,t_j],i} \leq \sum_{j=1}^M a_j \sqrt{A \sum_{t=s_j}^{t_j} z_{t,i}},$$

which, by the Cauchy-Schwarz inequality, is at most

$$\sqrt{\sum_{j=1}^M a_j} \cdot \sqrt{A \sum_{j=1}^M a_j \sum_{t=s_j}^{t_j} z_{t,i}} = \sqrt{\sum_{j=1}^M a_j} \cdot \sqrt{A \sum_{j=1}^M a_j \sum_{t=1}^T \mathbf{1}\{t \in U_j\} z_{t,i}} = \sqrt{\sum_{j=1}^M a_j} \cdot \sqrt{A \sum_{t=1}^T u_{t,i} z_{t,i}}.$$

Therefore, we need a construction of a_j and U_j such that $\sum_{j=1}^M a_j$ is minimized. This is addressed in Lemma 7 below which shows that there is in fact always a (optimal) construction such that $\sum_{j=1}^M a_j$ is exactly $\sum_{t=1}^T [u_{t,i} - u_{t-1,i}]_+$. Now summing the resulting bound over all experts and applying the Cauchy-Schwarz inequality again proves the theorem. ■

Lemma 7 *Let $v_1, \ldots, v_T$ be T nonnegative numbers and $h(\{v_1, \ldots, v_T\}) = \min \sum_{j=1}^M a_j$ where the minimum is taken over the set of all possible choices of $M \in \mathbb{N}^+, a_j > 0, U_j = [s_j, t_j]$ with $1 \leq s_j \leq t_j \leq T, s_j, t_j \in \mathbb{N}^+$ $(j = 1, \ldots, M)$ such that $v_t = \sum_{j=1}^M a_j \mathbf{1}\{t \in U_j\}$ for all t. Then with v_0 defined to be 0 we have*

$$h(\{v_1, \ldots, v_T\}) = \sum_{t=1}^T [v_t - v_{t-1}]_+.$$

Proof We prove the lemma by induction on T. The base case $T = 1$ is trivial. Now assume the lemma is true for any number of v's smaller than T. Suppose $a_j, U_j = [s_j, t_j]$ $(j = 1, \ldots, M)$ is an

optimal construction. Let $t^* \in \arg\min_t v_t$. Without loss of generality assume t^* belongs and only belongs to $U_1, \dots, U_k$ for some k. By the definition of h, we must have

$$h(\{v_1, \dots, v_T\}) = v_{t^*} + h(\{v_1 - w_1, \dots, v_{t^*-1} - w_{t^*-1}\}) + h(\{v_{t^*+1} - w_{t^*+1}, \dots, v_T - w_T\}),$$

where we define $w_t = \sum_{j=1}^k a_j \mathbf{1}\{t \in U_j\}$ and $h(\emptyset) = 0$. (Note that we also use the fact that $v_{t^*} = \min_t v_t$ so that $v_t - w_t \geq v_t - v_{t^*}$ is always nonnegative as required.) Now the key idea is to show that "extending" each U_j $(j \in [k])$ to the entire time interval $[1, T]$ does not increase the objective value in some sense. First consider extending U_1. By the inductive assumption, we have

$$\begin{aligned}
&h(\{v_1 - w_1 - a_1, \dots, v_{s_1-1} - w_{s_1-1} - a_1, v_{s_1} - w_{s_1}, \dots, v_{t^*-1} - w_{t^*-1}\} \\
&= (v_1 - w_1 - a_1) + \left(\sum_{t=2}^{s_1-1} [(v_t - w_t - a_1) - (v_{t-1} - w_{t-1} - a_1)]_+ \right) \\
&\quad + [(v_{s_1} - w_{s_1}) - (v_{s_1-1} - w_{s_1-1} - a_1)]_+ + \left(\sum_{t=s_1+1}^{t^*-1} [(v_t - w_t) - (v_{t-1} - w_{t-1})]_+ \right) \\
&= h(\{v_1 - w_1, \dots, v_{t^*-1} - w_{t^*-1}\}) + [(v_{s_1} - w_{s_1}) - (v_{s_1-1} - w_{s_1-1} - a_1)]_+ \\
&\quad - ([(v_{s_1} - w_{s_1}) - (v_{s_1-1} - w_{s_1-1})]_+ + a_1) \\
&\leq h(\{v_1 - w_1, \dots, v_{t^*-1} - w_{t^*-1}\}),
\end{aligned}$$

where the inequality follows from the fact $[b + c]_+ \leq [b]_+ + c$. Similarly, we also have

$$\begin{aligned}
&h(\{v_{t^*+1} - w_{t^*+1}, \dots, v_{t_1} - w_{t_1}, v_{t_1+1} - w_{t_1+1} - a_1, \dots, v_T - w_T - a_1\} \\
&\leq h(\{v_{t^*+1} - w_{t^*+1}, \dots, v_T - w_T\}).
\end{aligned}$$

By extending $U_2, \dots, U_k$ one by one in a similar way, we arrive at

$$h(\{v_1, \dots, v_T\}) \geq v_{t^*} + h(\{v_1 - v_{t^*}, \dots, v_{t^*-1} - v_{t^*}\}) + h(\{v_{t^*+1} - v_{t^*}, \dots, v_T - v_{t^*}\}).$$

Notice that the right hand side of the above inequality admits a valid construction for $\{v_1, \dots, v_T\}$: an interval $U = [1, T]$ with weight $a = v_{t^*}$, together with the optimal constructions for $\{v_1 - v_{t^*}, \dots, v_{t^*-1} - v_{t^*}\}$ and $\{v_{t^*+1} - v_{t^*}, \dots, v_T - v_{t^*}\}$. Therefore, by the optimality of h, the above inequality must be an equality. By using the inductive assumption again, we thus have

$$\begin{aligned}
h(\{v_1, \dots, v_T\}) &= v_{t^*} + h(\{v_1 - v_{t^*}, \dots, v_{t^*-1} - v_{t^*}\}) + h(\{v_{t^*+1} - v_{t^*}, \dots, v_T - v_{t^*}\}) \\
&= v_{t^*} + \left(v_1 - v_{t^*} + \sum_{t=2}^{t^*-1} [v_t - v_{t-1}]_+ \right) + \left(v_{t^*+1} - v_{t^*} + \sum_{t=t^*+2}^{T} [v_t - v_{t-1}]_+ \right) \\
&= \left(\sum_{t=1}^{t^*-1} [v_t - v_{t-1}]_+ \right) + \left(\sum_{t=t^*+1}^{T} [v_t - v_{t-1}]_+ \right) \\
&= \sum_{t=1}^{T} [v_t - v_{t-1}]_+,
\end{aligned}$$

where the last step follows from $[v_{t^*} - v_{t^*-1}]_+ = 0$ by the definition of t^*. This completes the proof. ■