No-Regret Learning on Possibly Infinite Sets of Experts

CIS 625 Project Report

Kishor Jothimurugan and Caleb Stanford

June 2018

Abstract

The standard framework of no-regret learning consists of n experts who incur (arbitrary and independent) losses at every time step, and the goal is to minimize the cumulative expected loss w.r.t. the best expert in hindsight by choosing a distribution over experts at each step. In this setting n is assumed to be finite and small. We look at some works that deal with a large, potentially infinite number of experts.

1 Introduction

Any framework for online / no-regret learning consists of three basic components. First, we require a *set of experts*: this may be a hypothesis class $\mathcal{H}$, in which case it has some concrete structure (which may be exploited by the learning algorithm), or the experts may be left purely abstract with no additional assumptions. Second, we require an *environment*: the environment assigns, to each expert at every time step, a real-valued loss in the interval $[0, 1]$. In the most general case, there are no assumptions about this loss vector, and in particular, it may be chosen adversarially. Some restricted cases are: (i) the case where experts are hypotheses, where we assume that the loss equals the error of that hypothesis on a new instance (thus, correlated hypotheses have correlated losses); (ii) the "realizable" case, where we assume that there exists an expert for which the loss is always 0. Third and finally, we require a *learning algorithm* which, given the losses so far, and sometimes also given some "advice" from the experts, picks a distribution over experts before receiving the loss vector at each step.

Practically speaking, we should have two goals with this framework. (1) The main goal is to minimize *regret*, which is the difference between the total loss of the algorithm, and the total loss of the best expert, after T steps. (2) The second goal (perhaps just as important) is to do so using a reasonable amount of time at each time step. As discussed in class, the biggest problem with classical algorithms (i.e. *randomized weighted majority* or *hedge*, both of which we may refer to generically as *Weighted Majority*) is that they achieve (1) well but fail goal (2) if the number of experts is large. Specifically for T timesteps and N experts we get $O(\sqrt{T \log N})$ regret, much better than the maximum T, but we require $O(N)$ time at each step (this is the time required even to read the entire loss vector). If N is large, $\log N$ may contribute to the regret, but the $O(N)$ time at each step will quickly become the bigger practical bottleneck. For this reason, the classical algorithms are effective only for small N. Moreover, if N is infinite, the algorithms will not work at all.

In this project, we therefore look at two works which extend the results of no-regret learning to possibly infinite classes of experts. We are interested in results which are as general as possible, and we also discuss the complexity of the algorithms, to address goal (2). We chose the following papers:

- Mehryar Mohri and Scott Yang. Competing with automata-based expert sequences. In *Intl. Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.
- Shai Ben-David, Dávid Pál, and Shai Shalev-Shwartz. Agnostic online learning. In *Conference on Learning Theory (COLT)*, 2009.

At face value, the two papers are very different, so we begin in §2 by suggesting a common framework under which the no-regret learning problem with infinitely many experts can be studied. We show that learning is impossible in this framework with no assumptions, and that both papers correspond to particular restrictions of the common framework. In §3, we look at the setting of *expert sequences*: instead of competing against a finite set of experts, we compete against a restricted, infinite subset of all *sequences* of experts so far. So, the "set of experts" here will be the set of sequences of experts. (Note that although the set of expert sequences is infinite, the number of expert sequences up to a finite number of rounds T remains finite, but potentially very large.) In §4, we look at the setting where the set of experts is a possibly infinite hypothesis class, but which has finite "Littlestone dimension". The results on Littlestone dimension go back further to the realizable case, but we were more interested in this paper because it studies the agnostic setting, which is more general and practical. Finally, in §5 we discuss some future directions that could be explored (either through further investigation of the literature, or through new research). This includes our idea for how the main results from these two papers could be combined.

2 A Common Framework for Infinitely Many Experts

The basic online learning framework, where the set of experts is kept abstract and the environment chooses losses arbitrarily, is heavily skewed in the environment's favor when the set of experts is infinite. In particular, whatever the algorithm does, it must output a distribution over experts, which must only cover a finite subset of the experts. Since the environment sees this distribution at each timestep, it can assign loss 1 to all experts which are included in the distribution, and 0 to all others. Since the set of experts is infinite, there always exists an expert which has not been included in any distribution so far, so the regret will be exactly T: the maximum amount possible.

In order to make the no-regret learning problem feasible, we need a basic restriction on the environment. In particular, we can allow some structure to the experts — henceforth, *hypotheses* $\mathcal{H}$ — that the environment is required to respect. We suggest that *at each step, the hypotheses should be partitioned into finitely many classes, such that the loss is required to be the same within each class.* If $\mathcal{Y}$ is the finite set of classes, then each $h \in \mathcal{H}$ chooses a class $y \in \mathcal{Y}$ at each round (possibly depending on an input data point $\mathbf{x}$ provided by the environment). Then, the learning algorithm chooses a distribution over $\mathcal{Y}$ and the environment chooses a loss vector over $\mathcal{Y}$. We get the following definition:

Definition 2.1 (Finite-Partition Online Learning Framework)**.** An *finite-partition online learning framework* consists of $(L, E, \mathcal{X}, \mathcal{Y}, \mathcal{H})$, where:

- $\mathcal{X}$ is a (possibly infinite) *input set*, representing information that the hypotheses may use to make their prediction.
- $\mathcal{Y}$ is a finite *output set*, representing the possible choices at each round.
- $\mathcal{H}$ is a (possibly infinite) *hypothesis class.* Each hypothesis is a function

  $$h : X \times \{1, 2, \ldots, T\} \to \mathcal{Y},$$

  for some finite number of rounds T. We let $\mathcal{H}_T$ be the set of hypotheses for T rounds, and we require that $\mathcal{H}_T$ is nonempty for every $T \geq 1$.
- E is the *environment* and L is the *learner.* On round t for $t = 1, 2, \ldots, T$, E provides an input $\mathbf{x}_t \in X$ to L. Then, L responds with a distribution p_t over the output set Y. We assume here that L is deterministic. Following L's response, the environment provides a vector of losses $l_t : Y \to \{0, 1\}$. The learner thus experiences the loss $p_t \cdot l_t$ (dot product), and each hypothesis $h \in \mathcal{H}$ experiences a loss $l_t(h(\mathbf{x}_t, t))$.

The *regret* of the framework is the loss of L minus the loss of the best hypothesis in hindsight:

$$\mathrm{Reg}_T(L, E, \mathcal{X}, \mathcal{Y}, \mathcal{H}) = \sup_{h \in \mathcal{H}_T} \sum_{t=1}^{T} \Big((p_t \cdot l_t) - l_t(h(\mathbf{x}_t, t)) \Big).$$

The framework is more versatile than it may appear. It covers the classical case with finitely many experts and no assumptions about the loss vector: to get n experts, take $|\mathcal{Y}| = n$, and hypotheses $\mathcal{H}_T = \{h_y \mid y \in \mathcal{Y}\}$, where $h_y(\mathbf{x}, t) = y$ for all $\mathbf{x}, t$. However, the following theorem shows that good bounds on the regret are not possible without considering specific cases for $\mathcal{H}$:

Theorem 2.2. Let $\mathcal{X}$ be arbitrary and $|\mathcal{Y}| = n \geq 2$, and take $\mathcal{H}_T = (\mathcal{X} \times \{1, 2, \ldots, T\}) \to \mathcal{Y}$ (the largest class of hypotheses possible). Then there exists an adversarial environment E such that for every learner L and number of steps T, $\mathrm{Reg}_T(L, E, \mathcal{X}, \mathcal{Y}, \mathcal{H}) \geq \frac{n-1}{n} T$.

Proof. This is a variant of the lower bound argument from class. E works as follows: at round t, look at the distribution p_t and pick $y_t = \arg\min_y(p_t(y))$ (the output on which L puts the least weight). Assign loss 0 for y_t and 1 for all other $y \in \mathcal{Y}$. Because $\mathcal{H}_T = (\mathcal{X} \times \{1, 2, \ldots, T\}) \to \mathcal{Y}$, there exists $h \in \mathcal{H}_T$ with 0 loss on every round. But L puts at most $\frac{1}{n}$ weight on y_t by choice of y_t, so it gets loss at least $\frac{n-1}{n}$ at each round. $\square$

The remainder of this report studies the following question: for which triples $(\mathcal{X}, \mathcal{Y}, \mathcal{H})$ is learning possible? The next two sections consider restrictions on $(\mathcal{X}, \mathcal{Y}, \mathcal{H})$ under which sub-linear regret bounds can be derived (better than the linear regret of Theorem 2.2). In §3, $\mathcal{H}$ consists of a subset of all sequences of outputs $\mathcal{Y}^*$, or "expert sequences", where the subset is given by the accepting strings of a finite automaton on input alphabet $\mathcal{Y}$. So, $\mathcal{X}$ is ignored, but the output of each hypothesis depends on the time t. The bounds given are nontrivial only if the automaton only accepts sub-exponentially many strings. In §4, we consider $\mathcal{H}$ to be any set of hypotheses $\mathcal{X} \to \mathcal{Y}$ with finite Littlestone dimension. So, the time $t \in \mathbb{N}$ is ignored, and the output of each hypothesis depends only on the input $\mathbf{x}$.

3 Automata-Based Expert Sequences

Mohri and Yang are motivated by work that has been done on *tracking the best expert* [4]. Recall that the definition of regret implies that in online learning, we measure our success only relative to the best success among the N experts we have chosen. If it happens that none of experts perform well, algorithms like Weighted Majority have good regret, but still don't perform impressively in practice. In fact, this problem occurs even for "simple" failure cases involving some clear pattern of shifts in the best expert over time: for example, (1) the best expert alternates between two candidates on odd and even rounds, or (2) the best expert switches k times over the course of the T rounds, for some small constant k.

These failure cases can be prevented if we generalize away from a finite set of experts, instead to an infinite set of *expert sequences* which capture all the patterns that we want to perform well against. The authors provide a learning algorithm and regret bound for any class of expert sequences that satisfies: (i) the set of sequences is a regular language, and (ii) the number of sequences of length T is sufficiently small (polynomial, or even sub-exponential in T is enough). In particular, this covers the two failure cases mentioned in the previous paragraph (alternating between experts or k-shifting regret); see examples in §3.2. To capture (i), we will assume that the set of expert sequences is the language of an unambiguous nondeterministic finite automaton (UFA). The running time of the main algorithm will then depend linearly on the number of states in this automaton.

Actually, the paper defines and studies a more general notion of regret called "weighted regret", for which the expert sequences are given by a weighted finite automaton (WFA).

For the purposes of this report, we have adapted the results to the simpler case of "unweighted regret" which requires only that we have an unambiguous NFA (the weights of the WFA do not matter), and which fits into our common framework from the previous section. Because we restrict to this simpler case, the theorems and algorithm here may not immediately resemble those in the paper. We briefly discuss the weighted case, and other extensions covered in the paper, in §3.4.

3.1 Expert sequences and regret

An expert sequence is just a string over a finite alphabet. We may refer to expert sequences as "meta-experts" or "hypotheses", but to avoid ambiguity, "expert" always means one of the finitely many.

Definition 3.1 (Expert sequence)**.** Let Σ be a finite alphabet, called the set of experts. An *expert sequence* is a string $w \in \Sigma^*$. If T is the length of w, then w defines a corresponding hypothesis $h_w : \mathcal{X} \times \{1, 2, \ldots, T\} \to \mathcal{Y}$, where $h_w(\mathbf{x}, t) = w[t]$ (the tth character of w).

Definition 3.2 (Regret against a set of expert sequences)**.** Let $A \subseteq \Sigma^*$ be a set of expert sequences which contains at least one string of each length. In the learning framework of §2, we take $\mathcal{X} = \{1\}$, $\mathcal{Y} = \Sigma$, and $\mathcal{H} = \{h_w \mid w \in A\}$. Thus, for a learner L, environment E, and number of rounds T, the *regret against* A is

$$\begin{aligned}\mathrm{Reg}_T(L, E, A) &:= \mathrm{Reg}_T(L, E, \{1\}, \Sigma, \{h_w \mid w \in A\}) \\ &= \sup_{w \in A} \sum_{t=1}^{T} \Big((p_t \cdot l_t) - l_t(w[t]) \Big).\end{aligned}$$

In the paper, A is considered to be regular (accepted by some UFA), and this is called the *unweighted regret* (to contrast with *weighted regret* which involves a weighted automaton). But even if A is not regular, it is possible to get the same good regret bound with a naive algorithm, *if* we ignore the complexity of the algorithm.

Theorem 3.3 (Naive, inefficient algorithm)**.** Let $A \subseteq \Sigma^*$ be computable, and let K_T be the number of strings in A of length T. Then there exists a learning algorithm L such that for any number of rounds T and for any environment E,

$$\mathrm{Reg}_T(L, E, \mathcal{A}) \leq \sqrt{\tfrac{1}{2}\, T \log K_T}.$$

The running time of L on T rounds depends on how long it takes to compute the set of strings in A of length T. If this is $f(T)$, then the running time is $f(T) + O(T \cdot K_T)$.

Proof. L is simply (randomized) weighted majority, but on K_T meta-experts. We precompute the K_T expert sequences and keep track of weights for all K_T meta-experts. When the environment provides a loss vector, we convert it to a loss vector over all K_T meta-experts and update accordingly. When we wish to output a distribution over K_T meta-experts, we collapse it to a distribution just over the experts of the current round.

In carrying out this procedure, it is as if we are doing the usual (randomized) weighted majority on K_T experts, but the environment is choosing loss vectors to respect the structure of the meta-experts. Thus, the regret guarantee $\sqrt{\frac{1}{2}T\ \log(N)}$ holds, with N replaced by K_T. Similarly the running time is $O(T \cdot K_T)$ instead of $O(T \cdot N)$ □

The upshot is that it's possible to get sub-linear regret as long as $\sqrt{\frac{1}{2}T \log K_T} = o(T)$, which holds iff K_T is sub-exponential: that is, for all $a > 1$, for all sufficiently large T, $K_T \leq a^T$. On the other hand, the running time of the naive algorithm makes it impractical. We thus turn to the case where A is regular, given by a finite automaton.

3.2 Regret against an unambiguous NFA and main results

Definition 3.4. An *unambiguous finite automaton (UFA)* is a nondeterministic finite automaton (NFA) $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ which satisfies the property that for any accepted string, there is at most one accepting path. We allow multiple starting states and multiple final states, but we assume there are no epsilon transitions.

Every DFA is also a UFA, but UFAs can be more succinct. For any UFA $\mathcal{A}$ over Σ, the language $A = L(\mathcal{A})$ is a set of expert sequences. The main result here is a learning algorithm which achieves good regret (the same regret as the naive algorithm), but does so with efficient running time (linear in T and linear in the size of the UFA). The proof of the theorem consists in providing the algorithm L, called Automata Weighted Majority, which we describe in the next section.

Definition 3.5 (Regret against a UFA)**.** Let $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ be a UFA. The regret against $\mathcal{A}$ is the regret against the set of expert sequences it accepts: $\mathrm{Reg}_T(L, E, \mathcal{A}) := \mathrm{Reg}_T(L, E, L(A))$.

Theorem 3.6. Let $\mathcal{A}$ be any UFA. Let K_T be the number of strings in $L(\mathcal{A})$ of length T. Then there exists a learning algorithm L such that for any number of rounds T and for any environment E,

$$\mathrm{Reg}_T(L, E, \mathcal{A}) \leq \sqrt{\tfrac{1}{2}\ T\ \log K_T}.$$

Additionally, the running time of L on T rounds is $O(|\mathcal{A}| \cdot n \cdot T)$, where $n = |\Sigma|$ is the number of experts.

We can use Theorem 3.6 to investigate the regret bounds and corresponding running time for some special cases, including k-shifting and repeating patterns of period k (the two examples mentioned at the start of this section).

Example 3.7 (k-shifting regret)**.** Let $\textsc{Shift}_k \subseteq \Sigma^*$ be the set of expert sequences in which the expert switches at most k times. For example, if $\Sigma = \{0,1\}$ then $\textsc{Shift}_k = (0^* \cup 1^*)^{k+1}$. We can design an UFA $\mathcal{A}$ for this language with $c \cdot k \cdot n$ states and transitions, where c is a constant and $n = |\Sigma|$, by keeping track of the current expert and the number of switches so far. $\mathcal{A}$ will be unambiguous because the indices of the switches in any accepted string are unique. The number of strings of length T accepted is $K_T = 2 \cdot \sum_{i=0}^{k} n^{i+1}\binom{T-1}{i} \leq 2n^{k+1}T^k$. (This is a rough bound; for a much more precise bound on this sum, see [6].) Therefore, by Theorem 3.6, L attains regret

$$\mathrm{Reg}_T(L, E, \mathcal{A}) \leq \sqrt{\tfrac{1}{2}\, T \log(2n^{k+1}T^k)} \leq \sqrt{\tfrac{1}{2}\, T \log(n^{2k}T^{2k}} = \sqrt{k\, T \log(nT)},$$

with running time proportional to $kn^2 \cdot T$.

Compared to existing algorithms in the literature designed specifically for k-shifting regret (such as [4]), as far as we can tell, this is roughly an equivalent regret bound. Some k-shifting regret algorithms are surveyed in [7] (lecture notes for an online learning course), including the naive, inefficient algorithm which gets a regret bound $\sqrt{2\, k\, T \log(nT)}$ (this is basically Theorem 3.3), and a smarter solution with regret $\sqrt{2\, k\, T \log(nT/k)}$. The smarter solution also gets running time $O(n \cdot T)$, which is better than we have above.

Example 3.8 (Periodic expert sequences)**.** Let $\textsc{Periodic}_k \subseteq \Sigma^*$ be the set of expert sequences which are periodic of period k: for example, if $\Sigma = \{0,1\}$ and $k = 2$, then $\textsc{Periodic}_k = 0^* \cup 1^* \cup (01)^* \cup (01)^*0 \cup (10)^* \cup (10)^*1$. We can design a UFA $\mathcal{A}$ for this language with a separate cycle of states for each possible pattern, so there are $n^k \cdot k$ states and transitions. The number of strings of length T accepted is $K_T \leq n^k$. By Theorem 3.6, L attains regret

$$\mathrm{Reg}_T(L, E, \mathcal{A}) \leq \sqrt{\tfrac{1}{2}\, T\, \log(n^k)} = \sqrt{\tfrac{1}{2}\, k\, T \log n},$$

but here the running time is somewhat worse: proportional to $kn^{k+1} \cdot T$. If $k = 2$, the example where we wish to detect if the best expert alternates between two candidates, this is already cubic in the number of experts.

Example 3.9 (Finite unions)**.** Regular languages are closed under union, so we can protect against any finite union of failure cases simultaneously. For example, we can make L perform well against periodic sequences of length $1, 2$, or 3 *as well as* k-shifts with $k = 4$. Suppose we have UFAs $\mathcal{A}_1, \ldots, \mathcal{A}_m$ with $K_{T,1}, \ldots, K_{T,m}$ strings accepted of length T, respectively, and assume further that their languages are disjoint (this is necessary to preserve unambiguity in the following construction). Then we can use the standard NFA construction to get a UFA $\mathcal{A}$ for the union, which accepts $K_T \leq K_{T,1} + \cdots + K_{T,m}$ strings and has size $|\mathcal{A}| = |\mathcal{A}_1| + \cdots + |\mathcal{A}_m|$. Therefore, L attains regret

$$\mathrm{Reg}_T(L, E, \mathcal{A}) \leq \sqrt{\tfrac{1}{2} T \log(K_{T,1} + \cdots + K_{T,m})},$$

with running time $n \cdot (|\mathcal{A}_1| + \cdots + |\mathcal{A}_m|) \cdot T$. Roughly speaking, the running time scales additively, but the regret scales sub-additively (because of the log, it's only slightly larger than the largest regret term of all the $\mathcal{A}_i$).

3.3 The Automata Weighted Majority algorithm

We present the algorithm which proves Theorem 3.6. Our strategy is to *mimic exactly* the behavior of the naive, inefficient algorithm (Theorem 3.3), but without explicitly keeping track of weights for all K_T different experts, as K_T could be much larger than T. So, we have to find a clever way to store all of these weights simultaneously and concisely.

For this, we take a dynamic programming approach (the paper constructs a large weighted automaton instead of a table, but it is the same idea). Let $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ be the UFA, and let $H = L(\mathcal{A}) \cap \Sigma^T$ be the set of expert sequences (meta-experts) of length T that are considered, so that $|H| = K_T$. We assume that $\mathcal{A}$ is "trim", meaning that all states are reachable from an initial state and can reach a final state. All UFAs can be trimmed just by removing all non-reachable states; this is a standard initial step in many automaton constructions, and it guarantees by unambiguity that there are no two paths in the automaton from one state to another labeled with the same input string. Recall that in the naive algorithm, we wish to maintain a weight $u_t[w]$ for each timestep t and each meta-expert $w \in H$, where the weight is initialized to $u_0[w] = 1$ and multiplied by $e^{-\eta l}$ whenever the meta-expert experiences loss l. At time t, we wish to collapse meta-experts based on the current character in the sequence: $v_t[a] = \sum_{w \in H : w[t] = a} u_t[w]$. At timestep $t+1$, we output the distribution over Σ given by normalizing from the weights (or potentials) $v_t[a]$. In the Automata Weighted Majority algorithm, we only compute v_t at each step, not u_t.

To accomplish this, we build two tables α and β (*forward* and *backward* weights, respectively). For $t \in \{0, 1, 2, \ldots, T\}$ and $q \in Q$, let $\alpha[t, q]$ be the sum over all paths of length t ending at state q of the weight such a sequence of experts would have in the algorithm at time t: that is, if the string is $a_1 a_2 \ldots, a_t \in \Sigma^t$, then the weight would be $e^{-\eta \sum_{i=1}^{t} l_i[a_i]}$, and $\alpha[t, q]$ is the sum over all such sequences. All such sequences will be prefixes of actual meta-experts in H. Let $\beta[t, q]$ be the *number* of paths of length $T - t$ from state q to any final state (unlike $\alpha[t, q]$, this is independent of the losses l_t). For sets of states P, P' and a string w, denote by $\mathsf{path}(P, w, P')$ the statement that there is a path from a state in P to a state in P' labeled by the string w. Then the following calculations show how to express

Algorithm 1 Automata Weighted Majority

```
Initialize:
    set $\beta[T,q] := 1$ for $q \in F$ and $\beta[T,q] := 0$ for $q \notin F$
    for $t = T-1, T-2, ..., 2, 1, 0$ do
        for all $q \in Q$, set $\beta[t,q] := \sum_{a\in\Sigma} \sum_{q' : q' \in \Delta(q,a)} \beta[t+1, q']$
    end for
    compute $K_T := \sum_{q \in I} \beta[0,q]$
    set $\eta := \sqrt{\frac{\log K_T}{T/8}}$    (optimal $\eta$ to obtain $\sqrt{\frac{1}{2} T \log K_T}$ regret bound)
    set $\alpha[0,q] := 1$ for $q \in I$ and $\alpha[0,q] := 0$ for $q \notin I$
for $t = 0, 1, 2, ..., T-2, T-1$ do
    compute $v_t[a] := \sum_{q\in Q} \alpha[t,q] \cdot \beta[t,q]$
    predict distribution $v_t$ over $\Sigma$
    receive loss $l_{t+1}$ over $\Sigma$
    for all $q \in Q$, set $\alpha[t+1, q'] := \sum_{a\in\Sigma} \sum_{q : q' \in \Delta(q,a)} \alpha[t,q] \cdot e^{-\eta l_{t+1}[a]}$
end for
```

$v_t[a]$ in terms of α and β:

$$\begin{aligned}
v_t[a] = \sum_{\substack{w \in H \\ w[t]=a}} u_t[w] &= \sum_{\substack{w \in H \\ w[t]=a}} \prod_{i=1}^{t} e^{-\eta l_i(w[t])} \\
&= \sum_{q\in Q} \sum_{\substack{w\in H \\ w[t]=a \\ \mathsf{path}(I, w[1:t], q) \\ \mathsf{path}(q, w[t+1:T], F)}} \prod_{i=1}^{t} e^{-\eta l_i(w[t])} \\
&= \sum_{q\in Q} \sum_{\substack{w_1 \in \Sigma^t \\ \mathsf{path}(I, w_1, q)}} \sum_{\substack{w_2 \in \Sigma^{T-t} \\ \mathsf{path}(q, w_2, F)}} \prod_{i=1}^{t} e^{-\eta l_i(w_1[t])} \\
&= \sum_{q\in Q} \left[\left(\sum_{\substack{w_1 \in \Sigma^t \\ \mathsf{path}(I, w_1, q)}} \prod_{i=1}^{t} e^{-\eta l_i(w_1[t])} \right) \cdot \left(\sum_{\substack{w_2 \in \Sigma^{T-t} \\ \mathsf{path}(q, w_2, F)}} 1 \right) \right] \\
&= \sum_{q\in Q} \alpha[t,q] \cdot \beta[t,q].
\end{aligned}$$

The algorithm can thus compute v_t by calculating α and β. Row $t+1$ of α can be calculated from row t of α and the losses at round t, and row t of β can be calculated from row $t+1$

of β. Thus, we can compute β to initialize (offline) and compute α online. To compute the next row of α or β requires summing over all transitions in the automaton and all input characters, so the total time complexity of the algorithm is $T \cdot |\Sigma| \cdot |\mathcal{A}|$. Note, however, that it should be possible to save the factor of Σ depending on how the automaton is represented; in particular, this should be possible if we represent a separate transition for each character and each pair of states. This completes the analysis of Automata Weighted Majority and the proof of Theorem 3.6.

3.4 Weighted case and further generalizations

The paper defines a notion of *weighted regret*: this is like the regret, but there is a distribution over expert sequences, and expert sequences which are more likely are considered more significant in the regret. Formally, we add a *penalty term* to the total loss of each expert sequence: if $w \in \Sigma^*$ is an expert sequence with probability $p(w)$, the authors take the penalty $-\log p(w) \in [0, \infty]$. This means that if $p(w)$ is small the expert sequence will likely drop out of consideration, and the learning algorithm only needs to compete with expert sequences which are sufficiently likely. This is useful for example because we can allow sequences with an arbitrary number of shifts instead of just k-shifts regret, as long as each additional shift decreases the likelihood of the sequence.

We can specify the distribution p over expert sequences with a weighted finite automaton (where the weights of the automaton are real numbers, which can be thought of as probabilities). The Automata Weighted Majority algorithm can take the weights into account, and enjoys an upper bound on weighted regret. For somewhat unclear reasons, the authors also assume the weighted automaton is deterministic, but only when describing the algorithm. This may indeed be necessary in the weighted case (unlike the unweighted case where we can have a UFA, which need not be a DFA).

All of this is really just the first half of the paper. In the second half, the authors consider two extensions. First, they consider approximation algorithms for Automata Weighted Majority: instead of keeping the entire T-by-$|Q|$ array of weights in memory, in some cases this can be approximated by a smaller weighted automaton, but at some cost to the regret bound. In particular, the additional loss added to the regret bound is given by the ∞-Rényi divergence, which measures the "distance" between the distributions of one weighted automaton and another approximating it. Therefore, the goal is to find an approximating automaton which minimizes this divergence, while being as small as possible. Second, they consider an extension to the sleeping expert setting — at each step, some of the base experts may abstain, and the regret goal becomes relaxed so that we do not have to compete with the "sleeping" experts. The interested reader should start by reading the original paper of Freund et al. [3].

3.5 Discussion

From a practical perspective, the regret bounds and computational complexity of Automata Weighted Majority look very good. Additionally, the framework of expert sequences — even when restricted to regular sets of expert sequences — appears very general and could be widely applicable. The work is completely "agnostic" in the sense that we have not assumed anything about the losses obtained from the environment. Indeed, it seems especially crucial that we remain agnostic when we are dealing with *expert sequences*, because one of the main motivations for expert sequences is that the best expert may change over time!

However, a lingering concern with Automata Weighted Majority is that it requires knowing the number of rounds T beforehand. This appears fairly important to the algorithm: we need to construct a table with T rows, and we cannot simply construct it "on the fly" whenever we need the next row because we need to initialize it from the bottom upwards. In a practical setting, especially any *online* setting, it is unlikely that we know the exact number of data points we will see before the algorithm begins. (In contrast, although Weighted Majority and related algorithms assume knowledge of T beforehand, this assumption can be relaxed with the "doubling trick".)

One way to address this concern would be to assume that, instead of knowing T beforehand, we just know some upper bound $T' \geq T$. It seems like the algorithm could be modified to allow this. However, it would still require constructing the table of size T' up front, so there would be a trade-off between taking T' large enough and not spending too much time and memory on initialization. Or, perhaps it is possible to adapt the doubling trick to this setting, if we are okay with constructing a new table of size T every time we double T.

4 Agnostic Online Learning

We now look at another online learning framework that fits the general framework of §2. We consider a restriction of the standard no-regret learning in which losses encountered by the experts are not entirely arbitrary but depend only on the the labels they predict on a particular input. More formally, let $\mathcal{X}$ denote the input space and $\mathcal{H}$ be a hypothesis class, i.e., a class of functions from $\mathcal{X}$ to $\{0, 1\}$. At each time point, the adversary/nature decides on a pair $(\mathbf{x}, y) \in \mathcal{X} \times \{0, 1\}$ and reveals $\mathbf{x}$ to the learner. The learner picks a hypothesis $h \in \mathcal{H}$ and predicts $h(\mathbf{x})$. The learner incurs a loss of 1 if $h(\mathbf{x}) \neq y$ and 0 otherwise. Here we view each hypothesis in our class as an expert. The goal of the learner is to minimize the (expected) regret over T time steps. The following theorem shows that there is no deterministic algorithm that achieves a sub-linear regret bound.

Theorem 4.1. Let $\mathcal{X}$ be a singleton and $\mathcal{H}$ consist of the two constant functions from $\mathcal{X}$ to $\{0, 1\}$. Then, any deterministic online learning algorithm that learns from $\mathcal{H}$ has worst case regret of at least $T/2$.

Proof. At each step i the adversary knows what the learner will predict based on its previous inputs. The adversary simply picks the opposite label. This causes the learner to make T mistakes. Out of the two hypothesis in $\mathcal{H}$, one hypothesis has error at most $T/2$ showing that regret is at least $T/2$. $\square$

The above result shows that we need randomization to achieve good regret bounds. In fact, it is possible even when the hypothesis class is *infinite* as long as the Littlestone dimension of $\mathcal{H}$ (denoted $\mathrm{Ldim}(\mathcal{H})$) is bounded. In the learning framework of §2, we have $\mathcal{Y} = \{0, 1\}$, and $\mathcal{H}$ is a class of hypotheses that do not depend on the time component. Here the environment is also restricted to output only those loss vectors in exactly one entry is 0 but it is not an important assumption and can be eliminated easily.

4.1 Littlestone's dimension and the realizable case

Littlestone's dimension (Ldim) [5] is a measure of complexity of the hypothesis class similar to VC Dimension that is more suited in the setting of online learning. To define Ldim, we need a few concepts. An instance-labeled tree of depth d is a complete (ordered) binary tree of height d where each node that is not a leaf is labeled by an element from $\mathcal{X}$. A root to leaf path on this tree, $n_1, ..., n_{d+1}$, describes a sequence of instance-label pairs, $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_d, y_d)$, where $\mathbf{x}_i$ is the label of the node n_i and y_i is 0 if the node n_{i+1} is the left child of n_i and y_i is 1 otherwise.

Definition 4.2 (Shattered Tree)**.** An instance-labeled tree is said to be shattered by a hypothesis class $\mathcal{H}$ if for every sequence of instance-label pairs $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_d, y_d)$ described by root-to-leaf path of the tree, there is a hypothesis $h \in \mathcal{H}$ such that $h(\mathbf{x}_i) = y_i$ for all $i \in \{1, ..., d\}$.

Definition 4.3 (Littlestone's dimension)**.** The Littlestone's dimension of a hypothesis class $\mathcal{H}$ is the largest d such that there is a instance-labeled tree of depth d that is shattered by $\mathcal{H}$.

The definition of Ldim looks very similar to VCdim except for the fact that we now have shattered trees in place of shattered set of points. If we insist that all nodes at any particular depth are labeled by the same instance, this definition yields VC dimension. Hence it is easy to see that the following holds:

$$\mathrm{VCdim}(\mathcal{H}) \leq \mathrm{Ldim}(\mathcal{H})$$

For finite $\mathcal{H}$, we have $\mathrm{Ldim}(\mathcal{H}) \leq \log(|\mathcal{H}|)$ as each leaf of a shattered tree corresponds to a different hypothesis in $\mathcal{H}$.

Example 4.4 (Parity Functions)**.** Let P_n denote the class of parity functions over $\{0,1\}^n$. The above inequalities imply that $\text{Ldim}(P_n) = n$.

Littlestone's dimension exactly captures the minimum possible number of mistakes that an online learning algorithm makes in the realizable case, i.e, the case in which adversary only gives instance-label pairs that are of the form $(\mathbf{x}, c(\mathbf{x}))$ for some c that is fixed for all T rounds. This leads to the following theorem:

Theorem 4.5 (Littlestone 1988)**.** The worst case number of mistakes of any deterministic algorithm A is at least $\text{Ldim}(\mathcal{H})$. There is a deterministic algorithm that achieves a mistake bound of $\text{Ldim}(\mathcal{H})$.

The lower bound is easy to see because the shattered tree of depth $\text{Ldim}(\mathcal{H})$ provides a strategy for the adversary to force the learner to make $\text{Ldim}(\mathcal{H})$ mistakes. The result also easily extends to randomized algorithms for which we get a lower bound of $\text{Ldim}(\mathcal{H})/2$. The generic algorithm that gives the upper bound is called the Standard Optimal Algorithm (SOA) which is described below:

Algorithm 2 Standard Optimal Algorithm (SOA)

```
initialize: $V_0 = \mathcal{H}$
for $t = 1, 2, ...$ do
    receive $\mathbf{x}_t$
    for $r \in \{0,1\}$ define $V_t^r = \{h \in V_{t-1} \mid h(\mathbf{x}_t) = r\}$
    predict $\hat{y}_t = \text{argmax}_r \text{Ldim}(V_t^r)$
    receive label $y_t$
    update $V_t \leftarrow V_t^{y_t}$
end for
```

It is easy to bound the mistakes made by the above algorithm. As an example, let us look at parity functions. In the case of parity functions, the above algorithm can be implemented efficiently. To store V_t, we just need to store the labels of a linearly independent subset of the examples seen (which spans all the examples).

4.2 Non-realizable case

Here, we do not make any assumption on the adversary's instance-label pairs. The authors of [1] provide a way to get low regret in this case if $\text{Ldim}(\mathcal{H})$ is finite. They make use of the following result about learning with experts:

Theorem 4.6. There is an online learning algorithm that uses N experts and achieves an expected regret bound of $\sqrt{(1/2)\log(N)T}$ where T is the number of rounds.

This is the standard learning with experts algorithm. We cannot use this directly to solve our problem because the number of experts, $|\mathcal{H}|$, can be infinite. But we can argue that in the case of finite Ldim, the number of "behaviours" of the hypothesis class on any list of T instances is small and hence giving us a small number of experts to track. This approach is similar to the approach used to bound the generalization error of an hypothesis that "fits" the data in the general PAC learning model, provided the VC dimension of the hypothesis class is finite.

To formalize the above idea, let us consider experts of a certain form. For each $L \leq \mathrm{Ldim}(\mathcal{H})$ and each sequence $1 \leq i_1 < i_2 < ... < i_L \leq T$, define an expert, $\texttt{Expert}(i_1, ..., i_L)$. We define the prediction of such an expert on a sequence of T instances, $\mathbf{x}_1, ..., \mathbf{x}_T$, as follows: Run the Standard Optimal Algorithm on the sequence, and in round t, instead of receiving y_t, define $y_t = \hat{y}_t$ if $t \notin \{i_1, ..., i_L\}$ and $y_t = \neg\hat{y}_t$ if $t = i_j$ for some j. The sequence $y_1, ..., y_T$ are the labels predicted by $\texttt{Expert}(i_1, ..., i_L)$ on $\mathbf{x}_1, ..., \mathbf{x}_T$.

Lemma 4.7. For any sequence of instances $\mathbf{x}_1, ..., \mathbf{x}_T$ and $h \in \mathcal{H}$, there is an $L \leq \mathrm{Ldim}(\mathcal{H})$ and a sequence $1 \leq i_1 < i_2 < ... < i_L \leq T$ such that the prediction of $\texttt{Expert}(i_1, ..., i_L)$ on the sequence $\mathbf{x}_1, ..., \mathbf{x}_T$ is the same as $h(\mathbf{x}_1), ..., h(\mathbf{x}_T)$.

Proof. Run SOA on the sequence $(\mathbf{x}_1, h(x_1)), ..., (\mathbf{x}_T, h(\mathbf{x}_T))$ and define $i_1, ..., i_L$ to be the sequence of rounds in which the algorithm made a mistake. By the upper bound on the mistakes made by SOA, $L \leq \mathrm{Ldim}(\mathcal{H})$. It is easy to see that the resulting expert behaves the same as h on the given sequence of instances. $\square$

Therefore, now we can use Theorem 3 to get good a regret bound. We run the expert learning algorithm but only track the experts of the form $\texttt{Expert}(i_1, ..., i_L)$ where $L \leq \mathrm{Ldim}(\mathcal{H})$. By Lemma 4, this is the same as tracking all hypotheses in $\mathcal{H}$. The number of experts considered is at most $T^{\mathrm{Ldim}(\mathcal{H})}$ thereby giving a (expected) regret bound of $\sqrt{(1/2)\mathrm{Ldim}(\mathcal{H})T\log(T)}$.

The paper [1] also gives a lower bound of $\Omega(\sqrt{\mathrm{Ldim}(\mathcal{H})T})$ for the expected regret (that is, an adversarial environment E which can attain this regret against any learner L).

4.3 Discussion

The main issue that has not been addressed is the complexity of maintaining V_i and computing Ldim. It turns out that this is a hard problem. But the authors argue that for specific hypothesis classes it might be feasible. One such hypothesis class is the class of parity functions over $\{0,1\}^n$ in which case V_i can be represented as a list of linearly independent instances along with their labels. Ldim of such a subclass is the difference between n and the number of linearly independent points in the list.

Although this work potentially provides a way to get good regret bounds for infinite number of experts, many natural infinite hypothesis classes such as linear threshold functions have infinite Ldim even though the VC dimension is finite, proving the impossibility of learning such classes in the online setting. This makes us wonder whether or not the adversarial model for online learning is the right one. It makes sense to consider having a prior distribution over the input space and design algorithms that have low regret with high probability.

5 Questions and Future Directions

Combining expert sequences with Littlestone dimension. The two papers address two different challenges associated with the general framework of §2. One paper deals with the time component (expert sequences) while the other deals with infinite number of experts/hypotheses $\mathcal{X} \to \mathcal{Y}$. We can combine the two results directly as follows: Given a subset of sequences of hypotheses from a hypothesis class with finite Littlestone dimension, we can use the approach from §4 to convert the problem to the case of sequences of finite experts by reducing the hypothesis class $\mathcal{H}$ to a finite class Σ that exhibits all behaviours of $\mathcal{H}$ on input sequences of length T. We can now adapt the solution from §3. A better way to generalize would be to adapt the definition of Littlestone dimension to work for hypotheses defined as functions from $\mathcal{X} \times \{1, ..., T\} \to \mathcal{Y}$. This requires two developments in the definition. Firstly, we can extend the definition to work for $|\mathcal{Y}| > 2$. Then we can incorporate the time component into the definition by placing ordering constraints on the labels of the nodes in a shattered tree.

Generalizing to more sets of expert sequences. In the work on automata-based expert sequences, the set of expert sequences $A \subseteq \Sigma^*$ is given by an unambiguous NFA (or a weighted automaton) $\mathcal{A}$. In particular, no-regret learning is possible if (i) A is regular and (ii) A has sub-exponentially many strings of length T. Can (i) and/or (ii) be relaxed? From the perspective of formal language theory, what is the precise class of languages A such that no-regret learning is possible (with sub-linear regret and an appropriate requirement on running time)? Mohri and Yang state that their results "can be straightforwardly extended to... complex formal language families such as (probabilistic) context-free languages over expert sequences." But is this the largest possible class?

As a starting point, we might want to formally characterize the class of languages satisfying (i) and (ii) in the first place — these might be termed "sparse" regular languages because they accept a decreasing fraction of strings as the length increases. A passing remark in [2] is that a language is sparse iff it is a finite union of languages of the form $u_1 v_1^* u_2 v_2^* \ldots u_n v_n^*$, where $u_i, v_i \in \Sigma^*$. That is, it is a finite union of finite concatenations of possibly iterated strings. An alternate characterization might involve Linear Recurrence Sequences (LRS), since the number of strings of length T in any regular language is given by an LRS (see [9] for a brief introduction). The right requirement on the LRS might be that all eigenvalues of its matrix have absolute value at most 1. At any rate, the goal would be to relax the formal characterization while preserving efficient no-regret learnability.

Generalizing to other automaton models. We have seen in Theorem 3.6 that the complexity of Automata Weighted Majority depends linearly on the size of the UFA $\mathcal{A}$. So even if we do not generalize beyond languages satisfying (i) and (ii) above, it is worth asking if this complexity can be improved if the automaton model is changed. In particular, for automaton models that differ in conciseness from UFAs, can algorithms be given which match or improve the complexity of Automata Weighted Majority (for the same regular language)? For instance, this question could be investigated for DFAs, for NFAs, for alternating finite automata, and for symbolically represented NFAs.

Relaxing the adversarial model of the environment. As discussed in §4.3, the adversarial model of the environment makes online, no-regret learning for certain hypothesis classes impossible. It thus may be interesting to get a better model of the environment which makes some reasonable simplifying assumptions. Any such model would involve distributions over the input and a relaxed requirement on learning (e.g. works well with high probability). However, we would still want to avoid the i.i.d. assumption, which is quite strong. It remains to be seen if such models lead us to interesting problems rather than just reducing to PAC learning.

References

[1] Shai Ben-David, Dávid Pál, and Shai Shalev-Shwartz. Agnostic online learning. In *Conference on Learning Theory (COLT)*, 2009.

[2] Mário JJ Branco and Jean-Éric Pin. Equations defining the polynomial closure of a lattice of regular languages. In *International Colloquium on Automata, Languages, and Programming*, pages 115–126. Springer, 2009.

[3] 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 Theory of computing*, pages 334–343. ACM, 1997.

[4] Mark Herbster and Manfred K Warmuth. Tracking the best expert. *Machine learning*, 32(2):151–178, 1998.

[5] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.

[6] Michael Lugo. Sum of the first k binomial coefficients for fixed n. MathOverflow. URL: https://mathoverflow.net/q/17236 (version: 2017-10-01).

[7] Shie Mannor and Shai Shalev-Shwartz. Online learning with shifting and drifting experts. Course notes, Advanced Course in Machine Learning, The Hebrew University of Jerusalem, 2010. URL: https://www.cs.huji.ac.il/~shais/Lecture5.pdf.

[8] Mehryar Mohri and Scott Yang. Competing with automata-based expert sequences. In *Intl. Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

[9] Joël Ouaknine and James Worrell. On linear recurrence sequences and loop termination. *ACM SIGLOG News*, 2(2):4–13, 2015.