14
Online Learning: Experts and Bandits

In this set of chapters, we consider a basic problem in online algorithms and online learning: how to dynamically choose from among a set of "experts" in a way that compares favorably to any fixed expert. Both this abstract problem, and the techniques behind the solution, are important parts of the algorithm designer's toolkit.

14.1 The Mistake-Bound Model

Suppose there are N experts who make predictions about a certain event every day—for example, whether it rains today or not, or whether the stock market goes up or not. Let U be the set of possible choices. The process in the experts setting goes as follows:

The term *expert* just refers to a person who has an opinion, and does not reflect whether they are good or bad at the prediction task at hand.

1. At the beginning of each time step t, each expert makes a prediction. Let $\mathcal{E}^t \in U^N$ be the vector of predictions.

2. The algorithm makes a prediction a^t, and simultaneously, the actual outcome o^t is revealed.

Note the order of events: the experts predictions come first, then the algorithm chooses an expert at the same time as the reality being revealed.

The goal is to minimize the number of mistakes, i.e., the number of times our prediction a^t differs from the outcome o^t.

Suppose we have 8 experts, and $\mathcal{E}^t = (0, 1, 0, 0, 0, 1, 1, 0)$. If we follow the third expert and predict $a^t = 0$, but the actual outcome is $o^t = 1$, we make a mistake; if we would have picked the second expert, we would have been correct.

Fact 14.1. There exists an algorithm that makes at most $\lceil \log_2 N \rceil$ mistakes, if there is a perfect expert.

Proof. The algorithm just considers all the experts who have made no mistakes so far, and predicts what the majority of them predict. Note that every time we make a mistake, the number of experts who have not been wrong yet reduces by a factor of 2 or more. (And when we do not make a mistake, this number does not increase.) Since there is at least one perfect expert, we can make at most $\lceil \log_2 N \rceil$ mistakes. □

Show that any algorithm must make at least $\lceil \log_2 N \rceil$ mistakes in the worst case.

Fact 14.2. There is an algorithm that, on any sequence, makes at most $M \leq m^*(\lceil \log_2 N \rceil + 1) + \lceil \log_2 N \rceil$ mistakes, where m^* is the number of mistakes made by the best of these experts on this sequence.

Proof. Think of time as being divided into "epochs". In each epoch, we proceed as in the perfect expert scenario as in Fact 14.1: we keep track of all experts who have not yet made a mistake in that epoch, and predict the majority opinion. The set of experts halves (at least) with every mistake the algorithm makes. When the set becomes empty, we end the epoch, and start a new epoch with all the N experts.

Note that in each epoch, every expert makes at least one mistake. Therefore the number of completed epochs is at most m^*. Moreover, we make at most $\lceil \log_2 N \rceil + 1$ mistakes in each completed epoch, and at most $\lceil \log_2 N \rceil$ mistakes the last epoch, giving the result. □

However, this algorithm is very harsh and very myopic. Firstly, it penalizes even a single mistake by immediately discarding the expert. But then, at the end of an epoch, it wipes the slate clean and forgets the past performance of the experts. Maybe we should be gentler, but have a better memory?

14.2 The Weighted Majority Algorithm

This algorithm, due to Littlestone and Warmuth, is remarkable for its simplicity. We assign a weight w_i to each expert $i \in [N]$. Let $w_i^{(t)}$ denote the weight of expert i at the beginning of round t. Initially, all weights are 1, i.e., $w_i^{(1)} = 1$.

1. In round t, predict according to the weighted majority of experts. In other words, choose the outcome that maximizes the sum of weights of experts that predicted it. I.e.,

$$a^t \leftarrow \arg\max_{u \in U} \sum_{i:\text{expert } i \text{ predicts } u} w_i^{(t)}.$$

We break ties arbitrarily, say, by picking the first of the options that achieve the maximum.

2. Upon seeing the outcome, set

$$w_i^{(t+1)} = w_i^{(t)} \cdot \begin{cases} 1 & \text{if } i \text{ was correct} \\ \frac{1}{2} & \text{if } i \text{ was incorrect} \end{cases}.$$

Theorem 14.3. *For any sequence of predictions, the number of mistakes made by the weighted majority algorithm (WM) is at most*

$$2.41(m_i + \log_2 N),$$

where m_i is the number of mistakes made by expert i.

Proof. The proof uses a potential-function argument. Let

$$\Phi^t := \sum_{i \in [N]} w_i^{(t)}.$$

Note that

1. $\Phi_1 = N$, since the weights start off at 1,
2. $\Phi_{t+1} \leq \Phi_t$ for all t, and
3. if Algorithm WM makes a mistake in round t, the sum of weights of the wrong experts is higher than the sum of the weights of the correct experts, so

$$\begin{aligned} \Phi^{t+1} &= \sum_{i \text{ wrong}} w_i^{(t+1)} + \sum_{i \text{ correct}} w_i^{(t+1)} \\ &= \frac{1}{2} \sum_{i \text{ wrong}} w_i^{(t)} + \sum_{i \text{ correct}} w_i^{(t)} \\ &= \Phi^t - \frac{1}{2} \sum_{i \text{ wrong}} w_i^{(t)} \\ &\leq \frac{3}{4} \Phi^t \end{aligned}$$

If after T rounds, expert i has made m_i mistakes and WM has made M mistakes, then

$$\left(\frac{1}{2}\right)^{m_i} = w_i^{(T+1)} \leq \Phi^{T+1} \leq \Phi^1 \left(\frac{3}{4}\right)^M = N \left(\frac{3}{4}\right)^M.$$

Now taking logs, and rearranging,

$$M \leq \frac{m_i + \log_2 N}{\log_2 \frac{4}{3}} \leq 2.41(m_i + \log_2 N). \qquad \square$$

In other words, if the best of the N experts on this sequence was wrong m^* times, we would be wrong at most $2.41(m^* + \log_2 n)$ times. Note that we are much better on the multiplier in front of the m^* term than Fact 14.2 was, at the expense of being slightly worse on the multiplier in front of the $\log_2 N$ term.

We cannot hope to compare ourselves to the best way of dynamically choosing experts to follow. This result says that at least we do not much worse to the best static policy of choosing an expert—in fact, choosing the best expert in hindsight—and sticking with them. We'll improve our performance soon, but all our results will still compare to the best static policy for now.

14.2.1 A Gentler Penalization

Instead of penalizing each wrong expert by a factor of 1/2, we could penalize the experts by a factor of $(1 - \varepsilon)$. This allows us to trade off the multipliers on the m^* term and the logarithmic term.

Theorem 14.4. *For $\varepsilon \in (0, 1/2)$, penalizing each incorrect expert by a factor of $(1 - \varepsilon)$ guarantees that the number of mistakes made by MW is at most*

$$2(1 + \varepsilon)m_i + O\left(\frac{\log N}{\varepsilon}\right).$$

Proof. Using an analysis identical to Theorem 14.3, we get that $\Phi^{t+1} \leq (1 - \frac{\varepsilon}{2})\Phi^t$ and therefore

$$(1-\varepsilon)^{m_i} \leq \Phi^{T+1} \leq \Phi^1 \Big(1 - \frac{\varepsilon}{2}\Big)^M = N\Big(1 - \frac{\varepsilon}{2}\Big)^M \leq N \exp\big(-\varepsilon M/2\big).$$

Now taking logs, and simplifying,

$$\begin{aligned} M &\leq \frac{-m_i \log(1-\varepsilon) + \ln N}{\varepsilon/2} \\ &\leq 2\,\frac{m_i(\varepsilon+\varepsilon^2)}{\varepsilon} + O\left(\frac{\log N}{\varepsilon}\right), \end{aligned}$$

because $-\ln(1-\varepsilon) = \varepsilon + \frac{\varepsilon^2}{2} + \frac{\varepsilon^3}{3} + \cdots \leq \varepsilon + \varepsilon^2$ for $\varepsilon \in [0,1]$. □

This shows that we can make our mistakes bound as close to $2m^*$ as we want, but this approach seems to have this inherent loss of a factor of 2. In fact, no deterministic strategy can do better than a factor of 2, as we show next.

Proposition 14.5. *No deterministic algorithm $\mathcal{A}$ can do better than a factor of* 2*, compared to the best expert.*

Proof. Note that if the algorithm is deterministic, its predictions are completely determined by the sequence seen thus far (and hence can also be computed by the adversary). Consider a scenario with two experts A,B, the first always predicts 1 and the second always predicts 0. Since $\mathcal{A}$ is deterministic, an adversary can fix the outcomes such that $\mathcal{A}$'s predictions are always wrong. Hence at least one of A and B will have an error rate of $\leq$ 1/2, while $\mathcal{A}$'s error rate will be 1. □

14.3 Randomized Weighted Majority

Consider the proof of Proposition 14.5, but applied to the WM algorithm: the algorithm alternates between predicting 0 and 1, whereas the actual outcome is the opposite. The weights of the two experts remain approximately the same, but because we are deterministic, we choose the wrong one. What if we interpret the weights being equal as a signal that we should choose one of the two options with equal probability?

This is the idea behind the *Randomized Weighted Majority* algorithm (RMW) of Littlestone and Warmuth: the weights evolve in exactly the same way as in Theorem 14.4, but now the prediction at each time is drawn randomly proportional to the current weights of the experts. I.e., instead of Step 1 in that algorithm, we do the following:

$$\Pr[\text{action } u \text{ is picked}] = \frac{\sum_{i:\text{expert } i \text{ predicts } u} w_i^{(t)}}{\sum_i w_i^{(t)}}.$$

Note that the update of the weights proceeds exactly the same as previously.

Theorem 14.6. *Fix $\varepsilon \leq 1/2$. For any fixed sequence of predictions, the expected number of mistakes made by randomized weighted majority (RWM) is at most*

$$\mathbb{E}[M] \leq (1+\varepsilon)m_i + O\left(\frac{\log N}{\varepsilon}\right)$$

The quantity $\varepsilon m_i + O(\frac{\log N}{\varepsilon})$ gap between the algorithm's performance and that of the best expert is called the *regret* with respect to expert i.

Proof. The proof is an analysis of the weight evolution that is more careful than in Theorem 14.4. Again, the potential is $\Phi^t = \sum_i w_i^{(t)}$. Define

$$F^t := \frac{\sum_{i \text{ incorrect}} w_i^{(t)}}{\sum_i w_i^{(t)}}$$

to be the fraction of weight on incorrect experts at time t. Note that

$$\mathbb{E}[M] = \sum_{t \in [T]} F_t.$$

Indeed, we make a mistake at step t precisely with the probability F_t, since the adversary does not see our random choice when deciding on the actual outcome a^t. By our re-weighting rules,

$$\Phi^{t+1} = \Phi^t\left((1-F_t) + F_t(1-\varepsilon)\right) = \Phi^t(1-\varepsilon F_t)$$

Bounding the size of the potential after T steps,

$$(1-\varepsilon)^{m_i} \leq \Phi^{T+1} = \Phi^1 \prod_{t=1}^{T}(1-\varepsilon F_t) \leq Ne^{-\varepsilon \sum F_t} = Ne^{-\varepsilon \mathbb{E}[M]}$$

Now taking logs, we get $m_i \ln(1-\varepsilon) \leq \ln N - \varepsilon \mathbb{E}[M]$, using the approximation $-\log(1-\varepsilon) \leq \varepsilon + \varepsilon^2$ gives us

$$\mathbb{E}[M] \leq m_i(1+\varepsilon) + \frac{\ln N}{\varepsilon}. \qquad \square$$

14.3.1 Classifying Adversaries for Randomized Algorithms

In the above analysis, it was important that the actual random outcome was independent of the prediction of the algorithm. Let us formalize the power of the adversary:

Oblivious Adversary. Constructs entire sequence $\mathcal{E}^1, o^1, \mathcal{E}^2, o^2, \cdots$ upfront.

Adaptive Adversary. Sees the previous choices of the algorithm, but must choose o^t independently of our actual prediction a^t in round t. Hence, o^t can be a function of $\mathcal{E}^1, o^1, \ldots, \mathcal{E}^{t-1}, o^{t-1}, \mathcal{E}^t$, as well as of $a^1, \ldots, a^{t-1}$, but not of a^t.

The adversaries are equivalent on deterministic algorithms, because such an algorithm always outputs the same prediction and the oblivious adversary could have calculated a^t in advance when creating $\mathcal{E}^{t+1}$. They may be different for randomized algorithms. However, it turns out that RWM works in both models, because our predictions do not affect the weight updates and hence the future.

14.4 The Hedge Algorithm, and a Change in Perspective

Let's broaden the setting slightly, and consider the following *dot-product* game. In each round,

Define the *probability simplex* as

$$\Delta_N := \{x \in [0,1]^N \mid \sum_i x_i = 1\}.$$

1. The algorithm produces a vector of probabilities
$$p^t = (p_1^t, p_2^t, \cdots, p_N^t) \in \Delta_N.$$
2. The adversary produces
$$\ell^t = (\ell_1^t, \ell_2^t, \cdots, \ell_N^t) \in [-1,1]^N.$$
3. The loss of the algorithm in this round is $\langle \ell^t, p^t \rangle$.

This equivalence between randomized and fractional algorithms is a common theme in algorithm design, especially in approximation and online algorithms.

We can move between this "fractional" model where we play a point in the probability simplex Δ_N, and the randomized model of the previous section (with an adaptive adversary), where we must play a single expert (which is a vertex of the simplex Δ_N. Indeed, setting ℓ^t to be a vector of 0s and 1s can capture whether an expert is correct or not, and we can set

$$p_i^t = \Pr[\text{algorithm plays expert } i \text{ at time } t]$$

to deduce that

$$\Pr[\text{mistake at time } t] = \left\langle \ell^t, p^t \right\rangle.$$

14.4.1 The Hedge Algorithm

The Hedge algorithm starts with weights $w_i^1 = 1$ for all experts i. In each round t, it defines $p^t \in \Delta_N$ using:

$$p_i^t \leftarrow \frac{w_i^t}{\sum_j w_j^t}, \tag{14.1}$$

and updates weights as follows:

$$w_i^{t+1} \leftarrow w_i^t \cdot \exp(-\varepsilon \ell_i^t). \tag{14.2}$$

Theorem 14.7. *Consider a fixed $\varepsilon \leq 1/2$. For any sequences of loss vectors in $[-1,1]^N$ and for all indices $i \in [N]$, the Hedge algorithm guarantees:*

$$\sum_{i=1}^{T} \langle p^t, \ell^t \rangle \leq \sum_{t=1}^{T} \ell_i^t + \varepsilon T + \frac{\ln N}{\varepsilon}$$

Proof. As in previous proofs, let $\Phi^t = \sum_j w_j^t$, so that $\Phi^1 = N$, and

$$\begin{aligned}
\Phi^{t+1} &= \sum w_i^{t+1} = \sum_i w_i^t e^{-\varepsilon \ell_i^t} \\
&\leq \sum_i w_i^t \left(1 - \varepsilon \ell_i^t + \varepsilon^2 \left(\ell_i^t\right)^2\right)\right) && \text{(using } e^x \leq 1 + x + x^2 \ \forall x \in [-1, 1]) \\
&\leq \sum w_i^t (1 + \varepsilon^2) - \varepsilon \sum w_i^t \ell_i^t && \text{(because } |\ell_i^t| \leq 1) \\
&= (1 + \varepsilon^2)\Phi^t - \varepsilon \Phi^t \ \langle p^t, \ell^t \rangle && \text{(because } w_i^t = p_i^t \cdot \Phi^t) \\
&= \Phi^t \ \left(1 + \varepsilon^2 - \varepsilon \langle p^t, \ell^t \rangle\right) \\
&\leq \Phi^t \ e^{\varepsilon^2 - \varepsilon \langle p^t, \ell^t \rangle} && \text{(using } 1 + x \leq e^x)
\end{aligned}$$

Again, comparing to the final weight of the i^{th} coordinate,

$$e^{-\varepsilon \sum \ell_i^t} = w_i^{(t+1)} \leq \Phi^{T+1} \leq \Phi^1 \ e^{\varepsilon^2 T - \varepsilon \sum \langle p^t, \ell_i^t \rangle};$$

now using $\Phi^1 = N$ and taking logs proves the claim. □

Moreover, choosing $\varepsilon = \sqrt{\frac{\ln N}{T}}$ gives $\varepsilon T + \frac{\ln N}{\varepsilon} = 2\sqrt{T \ln N}$, and the regret term is *concave and sublinear in time T*. This suggests that the further we run the algorithm, the quicker the average regret goes to zero, which suggests the algorithm is in some sense "learning".

14.4.2 Two Useful Corollaries

The following corollary will be useful in many contexts: it just flips Theorem 14.7 on its head, and shows that the average regret is small after sufficiently many steps.

Corollary 14.8. *For $T \geq \frac{4 \log N}{\varepsilon^2}$, the average loss of the Hedge algorithm is*

$$\begin{aligned}
\frac{1}{T} \sum_t \langle p^t, \ell^t \rangle &\leq \min_i \ \frac{1}{T} \sum_t \ell_i^t + \varepsilon \\
&= \min_{p^* \in \Delta_N} \ \frac{1}{T} \sum_t \left\langle \ell^t, p^* \right\rangle + \varepsilon.
\end{aligned}$$

The viewpoint of the last expression is useful, since it indicates that the dynamic strategy given by Hedge for the dot-product game is comparable (in the sense of having tiny regret) against any fixed strategy p^* in the probability simplex.

Finally, we state a further corollary that is useful in future lectures. It can be proved by running Corollary 14.8 with losses $\ell^t = -g^t / \rho$.

Corollary 14.9 (Average Gain)**.** *Let $\rho \geq 1$ and $\varepsilon \in (0, 1/2)$. For any sequence of gain vectors $g^1, \ldots, g^T \in [-\rho, \rho]^N$ with $T \geq \frac{4\rho^2 \ln N}{\varepsilon^2}$, the gains version of the Hedge algorithm produces probability vectors $p^t \in \Delta_N$ such that*

$$\frac{1}{T} \sum_{t=1}^{T} \left\langle g^t, p^t \right\rangle \geq \max_{i \in [N]} \ \frac{1}{T} \sum_{t=1}^{T} \left\langle g^t, e_i \right\rangle - \varepsilon.$$

In passing we mention that if the gains or losses lie in the range $[-\gamma, \rho]$, then we can get an asymmetric guarantee of $T \geq \frac{4\gamma\rho \ln N}{\varepsilon^2}$.

14.5 Optional: The Bandit Setting

The model of experts or the dot-product problem is often called the *full-information* model, because the algorithm gets to see the entire loss vector ℓ^t at each step. (Recall that we view the entries of the probability vector p^t played by the algorithm as the probability of playing each of the actions, and hence $\langle \ell^t, p^t \rangle$ is just the expected loss incurred by the algorithm. Now we consider a different model, where the algorithm only gets to see the loss of the action it plays. Specifically, in each round,

1. The algorithm again produces a vector of probabilities

$$p^t = (p_1^t, p_2^t, \cdots, p_N^t) \in \Delta_N.$$

   It then chooses an action $a^t \in [N]$ with these marginal probabilities.

2. *In parallel*, the adversary produces

$$\ell^t = (\ell_1^t, \ell_2^t, \cdots, \ell_N^t) \in [-1, 1]^N.$$

   However, now the algorithm only gets to see the loss $\ell_{a^t}^t$ corresponding to the action chosen by the algorithm, and not the entire loss vector.

This limited-information setting is called the *bandit* setting.

The name comes from the analysis of slot machines, which are affectionately known as "one-armed bandits".

14.5.1 The Exp3 Algorithm

Surprisingly, we can obtain algorithms for the bandit setting from algorithms for the experts setting, by simply "hallucinating" the cost vector, using an idea called *importance sampling*. This causes the parameters to degrade, however.

Indeed, consider the following algorithm: we run an instance $\mathcal{A}$ of the RWM algorithm, which is in the full information model. So at each timestep,

1. $\mathcal{A}$ produces a probability vector $p^t \in \Delta_N$.
2. We choose an expert $I^t \in [N]$, where

$$\Pr[I^t = i] = q_i^t := \gamma \cdot \frac{1}{N} + (1 - \gamma) \cdot p_i^t.$$

   I.e., with probability γ we pick a uniformly random expert, else we follow the suggestion given by p^t.

3. We get back the loss value $\ell^t_{I^t}$ for this chosen expert.

4. We construct an "estimated loss" $\hat{\ell}^t \in [0,1]^N$ by setting

$$\tilde{\ell}^t_j = \begin{cases} \frac{\ell^t_j}{q^t_j} & \text{if } j = I^t \\ 0 & \text{if } j \neq I^t \end{cases}.$$

We now feed $\tilde{\ell}^t$ to the RWM instance $\mathcal{A}$, and go back to Step 1.

We now show this algorithm achieves low regret. The first observation is that the estimated loss vector is an unbiased estimate of the actual loss, just because of the way we reweighted the answer by the inverse of the probability of picking it. Indeed,

$$\mathbb{E}[\tilde{\ell}^t_i] = \frac{\ell^t_i}{q^t_i} \cdot q^t_i + 0 \cdot (1 - q^t_i) = \ell^t_i. \tag{14.3}$$

Since each true loss value lies in $[-1, 1]$, and each probability value is at least γ/N, the absolute value of each entry in the $\tilde{\ell}$ vectors is at most N/γ. Now, since we run RWM on these estimated loss vectors belonging to $[0, N/\gamma]^N$, we know that

$$\sum_t \langle p^t, \tilde{\ell}^t \rangle \leq \sum_t \tilde{\ell}^t_i + \frac{N}{\gamma}\left(\varepsilon T + \frac{\log N}{\varepsilon}\right).$$

Taking expectations over both sides, and using (14.3),

$$\sum_t \langle p^t, \ell^t \rangle \leq \sum_t \ell^t_i + \frac{N}{\gamma}\left(\varepsilon T + \frac{\log N}{\varepsilon}\right).$$

However, the LHS is not our real loss, since we chose I^t according to q^t and not p^t. This means our expected total loss is really

$$\begin{aligned} \sum_t \langle q^t, \ell^t \rangle &= (1-\gamma) \sum_t \langle p^t, \ell^t \rangle + \frac{\gamma}{N} \sum_t \langle \mathbb{1}, \ell^t \rangle \\ &\leq \sum_t \ell^t_i + \frac{N}{\gamma}\left(\varepsilon T + \frac{\log N}{\varepsilon}\right) + \gamma T. \end{aligned}$$

Now choosing $\varepsilon = \sqrt{\frac{\log N}{T}}$ and $\gamma = \sqrt{N}\left(\frac{\log N}{T}\right)^{1/4}$ gives us a regret of $\approx N^{1/2}T^{3/4}$. The interesting fact here is that the regret is again sub-linear in T, the number of timesteps: this means that as $T \to \infty$, the per-step regret tends to zero.

The dependence on N, the number of experts/options, is now polynomial, instead of being logarithmic as in the full-information case. This is necessary: there is a lower bound of $\Omega(\sqrt{NT})$ in the bandit setting. And indeed, the Exp3 algorithm itself achieves a near-optimal regret bound of $O(\sqrt{NT \log N})$; we can show this by using a finer analysis of Hedge that makes more careful approximations. We defer these improvements for now, and instead give an application of this bandit setting to a problem in item pricing.

14.5.2 Item Pricing via Bandits

To be added in.