Contents

1 Experts and bandits 3
  1.1 Expert advice ........................................... 3
    1.1.1 The setting ....................................... 3
    1.1.2 If one expert is perfect ..................... 4
    1.1.3 If one expert is nearly perfect .......... 5
    1.1.4 Regret ........................................... 6
    1.1.5 Randomization .................................... 7
  1.2 Linear loss game on the simplex ........... 8
    1.2.1 The setting ..................................... 8
    1.2.2 Hedge .......................................... 9
    1.2.3 Entropy maximization ......................... 11
  1.3 Prediction with partial feedback .......... 12
    1.3.1 The setting .................................... 12
    1.3.2 Inverse probability weighting ............. 12
    1.3.3 Hedging over actions with bandit feedback .. 15
    1.3.4 Hedging over experts with bandit feedback .. 17
  1.4 Bibliographic references .................... 18
About these notes

These are lecture notes for the “Interactive Learning” seminar course taught in Fall 2017 at Columbia University as COMS 6998-4. They are being written and revised throughout the course. Borrowing a line from Sasha Rakhlin:

These lecture notes are constantly evolving, so if your version says $x < y$ today, it might say $x > y$ tomorrow.
Chapter 1

Experts and bandits

1.1 Expert advice

1.1.1 The setting

Consider a scenario where an agent (the “learner”) must repeatedly make a decision on some important matter, and each decision has some quantifiable consequence that ought to be optimized.

Formally, suppose there are several rounds of this decision-making game. In each round, the learner must choose between two actions, \{±1\}. The opponent (the “environment” or “Nature”) picks one of the actions for that round to be correct, and the other action is a mistake. The goal is to minimize the number of mistakes made over the rounds of the game.

So far, this setting is impossibly abstract. Whatever actions the learner chooses, they could all be correct, or they could all be mistakes, . . .

Let’s make things more tractable. Suppose that before choosing the action in each round, the learner receives a recommended action from each of \(N\) experts. Why should this help? The hope is that some (maybe just one!) of the experts make good recommendations over the rounds of the game—we will formalize this shortly. If this is the case, the learner can then try to just do (almost) as well as the best of these experts. The protocol is given in Algorithm 1.

**Algorithm 1** Protocol for online decision-making using expert advice (binary actions)

1: \textbf{for} \(t = 1, 2, \ldots\) \textbf{ do}
2: \quad Receive experts’ actions: \(b_{t,i} \in \{±1\}\) for \(i \in [N]\).
3: \quad Learner chooses action: \(a_t \in \{±1\}\).
4: \quad Learner receives correct action: \(y_t \in \{±1\}\).
5: \textbf{end for}

We use the notation \([n] := \{1, \ldots, n\}\) to denote the first \(n\) positive integers.
1.1.2 If one expert is perfect . . .

Suppose we are promised that one of the experts will always recommend the correct action in all rounds. We would like to design an algorithm for the learner that minimizes the number of mistakes.

The following algorithm keeps track of which experts are perfect-so-far, and chooses \( a_t \) to agree with any such expert. This is called the Consistent Expert algorithm.

**Algorithm 2 Consistent Expert algorithm**

1: Let \( V_0 := [N] \).

2: for \( t = 1, 2, \ldots \) do

3: Receive experts’ actions: \( b_t, i \in \{ \pm 1 \} \) for \( i \in [N] \).

4: Choose action: pick any \( i \in V_{t-1} \), and set \( a_t := b_t, i \).

5: Receive correct action: \( y_t \in \{ \pm 1 \} \).

6: Update: \( V_t := \{ i \in V_{t-1} : b_t, i = y_t \} \).

7: end for

Observe that by round \( t \), each “surviving” expert in \( V_{t-1} \) has predicted the correct action perfectly in the first \( t - 1 \) rounds.

**Theorem 1.1 (Consistent Expert).** If one of the \( N \) experts makes no mistakes, then a learner using Algorithm 2 (Consistent Expert) makes at most \( N - 1 \) mistakes.

Can we do better? Here is one way: choose \( a_t \) to agree with the majority of perfect-so-far experts. This is called the Halving algorithm.

**Algorithm 3 Halving algorithm**

1: Let \( V_0 := [N] \).

2: for \( t = 1, 2, \ldots \) do

3: Receive experts’ actions: \( b_t, i \in \{ \pm 1 \} \) for \( i \in [N] \).

4: Choose action: \( a_t := \text{sign} \left( \sum_{i \in V_{t-1}} b_t, i \right) \).

5: Receive correct action: \( y_t \in \{ \pm 1 \} \).

6: Update: \( V_t := \{ i \in V_{t-1} : b_t, i = y_t \} \).

7: end for

In Algorithm 3, we use the notation \( \text{sign}(z) := +1 \) if \( z > 0 \), and \( \text{sign}(z) := -1 \) if \( z \leq 0 \).

Whenever the learner makes a mistake, the number of perfect-so-far experts decreases by at least a factor of two. This can happen at most \( \log_2(N) \) times before the number of perfect-so-far experts is one. Thus we have proven the following.

**Theorem 1.2 (Halving).** If one of the \( N \) experts makes no mistakes, then a learner using Algorithm 3 (Halving) makes at most \( \log_2(N) \) mistakes.

This is exponentially better than the Consistent Expert algorithm!
1.1.3 If one expert is nearly perfect . . .

We now generalize to the case where we only assume that there is an expert that makes at most $K$ mistakes, for some non-negative integer $K$. Neither of the previous algorithms make sense anymore, because it is possible that there are no perfect-so-far experts even after the first round. So we shouldn’t throw out experts that make just a single mistake.

A simple idea is to use a weighted majority over all the experts (instead of a simple majority over the perfect-so-far experts). Instead of throwing out an expert who makes a mistake, we simply cut its weight in half. Therefore, the weight of an expert is small if it makes many mistakes. This is the Weighted Majority algorithm.

**Algorithm 4** Weighted Majority algorithm

1: Let $w_{1,i} := 1$ for $i \in [N]$.
2: for $t = 1, 2, \ldots$ do
3: Receive experts’ actions: $b_{t,i} \in \{\pm 1\}$ for $i \in [N]$.
4: Choose action: $a_t := \text{sign} \left( \sum_{i \in [N]} w_{t,i} \cdot b_{t,i} \right)$.
5: Receive correct action: $y_t \in \{\pm 1\}$.
6: Update: for each $i \in [N]$, $w_{t+1,i} := w_{t,i}/2$ if $b_{t,i} \neq y_t$, and $w_{t+1,i} := w_{t,i}$ otherwise.
7: end for

**Theorem 1.3 (Weighted Majority).** If one of the $N$ experts makes at most $K$ mistakes, then a learner Algorithm 4 (Weighted Majority) makes at most $(K + \log_2(N))/\log_2(4/3)$ mistakes.

**Proof.** Let $Z_t := \sum_{i=1}^N w_{t,i}$ denote the total weight of all experts at the start of round $t$ (so, e.g., $Z_1 = N$). Suppose the learner makes a mistake on round $t$. Let $\alpha_t \geq 1/2$ denote the fraction of the total weight corresponding to experts $i$ that choose $b_{t,i} = a_t$. Half of that weight will be cut by the learner by the end of round $t$. So the remaining weight $Z_{t+1}$ in such a round satisfies

$$Z_{t+1} = (1 - \alpha_t)Z_t + \frac{1}{2} \cdot \alpha_t Z_t \leq \frac{3}{4}Z_t.$$

So if a total of $M$ mistakes have been made through round $t$, then

$$Z_{t+1} \leq \left(\frac{3}{4}\right)^M Z_1 = \left(\frac{3}{4}\right)^M N.$$

On the other hand, if there is an expert $i$ who makes at most $K$ mistakes after $t$ rounds, then its weight satisfies $w_{t+1,i} \geq 2^{-K}$. The total weight after $t$ rounds must be at least the weight of this expert:

$$Z_{t+1} \geq 2^{-K}.$$

Therefore

$$2^{-K} \leq \left(\frac{3}{4}\right)^M N.$$
CHAPTER 1. EXPERTS AND BANDITS

Taking logarithm of both sides and rearranging gives

\[ M \leq \frac{K + \log_2(N)}{\log_2(4/3)}, \]

which finishes the proof. \( \square \)

1.1.4 Regret

As the number of rounds \( T \) grows to infinity, it may be unreasonable to assume that there is an expert that makes at most a constant number of mistakes (independent of \( T \)). Perhaps a more realistic assumption is that there is an expert with a low rate of making mistakes, in the sense that the number of mistakes after \( T \) rounds is at most \( \rho T \), for some fraction \( \rho \).

Under this relaxed assumption, the Weighted Majority algorithm guarantees the learner makes at most

\[ \frac{\rho T + \log_2(N)}{\log_2(4/3)} \leq 2.41(\rho T + \log_2(N)) \]

mistakes after \( T \) rounds. Note that this bound is not interesting if \( \rho \geq \log_2(4/3)/2 \approx 0.21 \), because in that case, the bound is larger than \( T/2 \) (which is what one gets, in expectation, by guessing uniformly at random . . . ).

We would instead like an algorithm that performs almost as well as the best expert, with a mistake bound of the form

\[ M_T - \min_{i \in [N]} M_{T,i} \leq o(T), \]

where \( M_T \) is the number of mistakes of the learner after \( T \) rounds, \( M_{T,i} \) is the number of mistakes of expert \( i \) after \( T \) rounds. We call the left-hand side quantity the regret of the learner to the best expert after \( T \) rounds.

If such a sublinear regret bound is possible, and the best expert makes at most \( \rho T \) mistakes after \( T \) rounds, then the learner makes at most \( (\rho + o(1))T \) mistakes after \( T \) rounds. This would mean that the learner’s performance approaches that of the best expert as \( T \) increases to infinity.

Unfortunately, a sublinear regret bound is impossible in general.

**Theorem 1.4 (Impossibility of sublinear regret).** Let \( b_{t,1} = -1 \) and \( b_{t,2} = +1 \) for all rounds \( t \). For any learning algorithm, there is a sequence \( y_1, \ldots, y_T \in \{\pm1\} \) such that the regret of the learner is at least \( T/2 \).

**Proof.** For any \( y_1, \ldots, y_T \in \{\pm1\} \), the better of the two experts makes at most \( T/2 \) mistakes. Therefore there is some sequence for which the learner makes \( T \) mistakes while the best expert makes only \( T/2 \) mistakes. \( \square \)
Exercise 1.1. Show how to modify Algorithm 4 so that its mistake bound is

$$2K + O\left(\sqrt{K \log(N) + \log(N)}\right)$$

whenever there is an expert that makes at most $K$ mistakes.

### 1.1.5 Randomization

It turns out we can get around this lower bound by allowing randomization and only considering the number of mistakes made by the learner in expectation. Here, we are limiting the power of the opponent and assuming that their decisions are made before learner’s random coins are tossed. In fact, we shall simply think of the sequence of correct and incorrect actions as being set once and for all before the start of the game.

We modify the Weighted Majority algorithm to use randomness in a natural way. This is the Randomized Weighted Majority algorithm.

**Algorithm 5** Randomized Weighted Majority algorithm

1: Parameter: $\eta > 0$.
2: $w_{1,i} := 1$ for $i \in [N]$.
3: **for** $t = 1, 2, \ldots$ **do**
4: Receive experts’ actions: $b_{t,i} \in \{\pm 1\}$ for $i \in [N]$.
5: Choose action: randomly draw $i_t \sim (w_{t,1}, \ldots, w_{t,N})/Z_t$, and set $a_t := b_{t,i_t}$.
6: Receive correct action: $y_t \in \{\pm 1\}$.
7: Update: for each $i \in [N]$, $w_{t+1,i} := e^{-\eta}w_{t,i}$ if $b_{t,i} \neq y_t$, and $w_{t+1,i} := w_{t,i}$ otherwise.
8: **end for**

There are two differences in Randomized Weighted Majority relative to Weighted Majority. First, we have replaced the value of 1/2 in the update step with an arbitrary fraction $e^{-\eta}$. This is simply because there is nothing special about the factor of 1/2, so we may as well pick the $\eta$ that gives the best guarantee—think of it as a “tuning parameter”.

Second, we now choose the action by randomly picking an expert with probability proportional to the weight of the expert, and then following the action of the chosen expert; we then only count the expected number of mistakes. The randomization and averaging are crucial: without them, the learner would be subject to the previous impossibility result. Essentially, they provide a way to transform a difficult discrete problem into an easier continuous problem. Let $p_t := (w_{t,1}, \ldots, w_{t,N})/Z_t$ denote the normalized vector of weights on experts in round $t$ (where $Z_t := w_{t,1} + \cdots + w_{t,N}$). Then the probability that

---

1Tuning parameters can usually be set to optimize some criterion (such as a regret bound), but often these settings are overly conservative. Therefore, it is desirable to have algorithms that do not have tuning parameters.
Randomized Weighted Majority makes a mistake—i.e., the fraction of weight on mistaken experts—is a linear function of $w_t$:

$$\Pr(b_t, i \neq y_t) = \sum_{i=1}^{N} \Pr(i_t = i) \cdot 1_{\{b_t, i \neq y_t\}}$$

$$= \sum_{i=1}^{N} \frac{w_t, i}{Z_t} \cdot 1_{\{b_t, i \neq y_t\}}$$

$$= \langle \ell_t, p_t \rangle,$$

where $\ell_t := (1_{\{b_t,1 \neq y_t\}}, \ldots, 1_{\{b_t,N \neq y_t\}})$ is the vector that indicates whether each expert makes a mistake in round $t$.\footnote{We use the notation $\langle u, v \rangle := \sum_{i=1}^{N} u_i v_i$ for the standard inner product between vectors $u, v \in \mathbb{R}^N$; and we use $1_{[P]}$ for the indicator of a predicate $P$, which takes value 1 if $P$ is true, and value 0 if $P$ is false.} The expected number of mistakes over all $T$ rounds is

$$\mathbb{E}(M_T) = \sum_{t=1}^{T} \Pr(a_t \neq y_t) = \sum_{t=1}^{T} \langle \ell_t, p_t \rangle.$$

It turns out with a suitable setting of the parameter $\eta$, the expected regret of Randomized Weighted Majority is sublinear in the number of rounds.

### 1.2 Linear loss game on the simplex

#### 1.2.1 The setting

The Randomized Weighted Majority algorithm turns out to be a special case of an algorithm for a more general problem. In this problem, again there are $N$ experts. In each round, the learner chooses a probability distribution $p_t = (p_{t,1}, \ldots, p_{t,N})$ over the $N$ experts; after, each expert incurs a non-negative loss $\ell_{t,i}$, and the learner incurs the weighted loss $\sum_{i=1}^{N} p_{t,i} \ell_{t,i} = \langle \ell_t, p_t \rangle$ where $\ell_t := (\ell_{t,1}, \ldots, \ell_{t,N})$. We call this the linear loss game on the (probability) simplex. The protocol is given in Algorithm 6.

**Algorithm 6** Protocol for the linear loss game on the simplex

1: for $t = 1, 2, \ldots$ do
2: Learner chooses probability vector: $p_t = (p_{t,1}, \ldots, p_{t,N}) \in \Delta^{N-1}$.
3: Learner receives loss vector: $\ell_t = (\ell_{t,1}, \ldots, \ell_{t,N}) \in \mathbb{R}_+^N$.
4: Learner incurs loss: $\langle \ell_t, p_t \rangle$.
5: end for

The following notations are used above:

$$\mathbb{R}_+^N := \left\{(p_1, \ldots, p_N) \in \mathbb{R}^N : p_i \geq 0 \forall i \in [N] \right\},$$

$$\Delta^{N-1} := \left\{(p_1, \ldots, p_N) \in \mathbb{R}_+^N : p_1 + \cdots + p_N = 1 \right\}.$$
The total loss of expert $i$ after $T$ rounds is

$$L_{T,i} := \sum_{t=1}^{T} \ell_{t,i},$$

and the total loss of the learner after $T$ rounds is

$$L_T := \sum_{t=1}^{T} \langle \ell_t, p_t \rangle.$$

The goal of the learner is to minimize the regret after $T$ rounds:

$$L_T - \min_{i \in [N]} L_{T,i}.$$

**Exercise 1.2.** Let $L_{T,q} := \sum_{i=1}^{T} q_i L_{T,i}$ be the total loss for a fixed probability distribution $q \in \Delta^{N-1}$. Show that

$$\min_{i \in [N]} L_{T,i} = \min_{q \in \Delta^{N-1}} L_{T,q}.$$

### 1.2.2 Hedge

The algorithm that generalizes the Randomized Weighted Majority algorithm is called the Hedge algorithm.

**Algorithm 7 Hedge algorithm**

**Require:** $\eta > 0$.

1. Let $w_1 = (w_{1,1}, \ldots, w_{1,N}) := (1, \ldots, 1)$.
2. for $t = 1, 2, \ldots$ do
3. Choose probability vector: $p_t = w_t / Z_t \in \Delta^{N-1}$, where $Z_t := \sum_{i=1}^{N} w_{t,i}$.
4. Receive loss vector: $\ell_t = (\ell_{t,1}, \ldots, \ell_{t,N}) \in \mathbb{R}_+^N$.
5. Update: $w_{t+1,i} := w_{t,i} \exp(-\eta \ell_{t,i})$ for all $i \in [N]$.
6. end for

**Theorem 1.5 (Hedge).** For any sequence of loss vectors $\ell_1, \ldots, \ell_T \in \mathbb{R}_+^N$, the total loss $L_T$ incurred by a learner using Algorithm 7 (Hedge) with parameter $\eta > 0$ after $T$ rounds satisfies

$$L_T - L_{T,i} = \sum_{t=1}^{T} \langle \ell_t, p_t \rangle - \sum_{t=1}^{T} \ell_{t,i} \leq \frac{\ln(N)}{\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \langle \ell^2_t, p_t \rangle, \quad i \in [N]$$

where $\ell^2_t := (\ell^2_{t,1}, \ldots, \ell^2_{t,N})$. 
The proof is very similar to the analysis of the Weighted Majority algorithm. The main idea is to bound the total weight $Z_{T+1}$ after $T$ rounds from above and below. The total weight shrinks from round to round as the different experts incur losses, and this amount can be related to the loss incurred by the learner. On the other hand, the total weight $Z_{T+1}$ can also be related to the total loss of any single expert after all $T$ rounds.

Proof. Below, we use the following approximations of the exponential:

$$\exp(z) \leq 1 + z + z^2/2, \quad z \leq 0; \quad (1.1)$$

$$\exp(z) \geq 1 + z, \quad z \in \mathbb{R}. \quad (1.2)$$

We consider the relative change in total weight after round $t$:

$$\frac{Z_{t+1}}{Z_t} = \sum_{i=1}^{N} \frac{w_{t+1,i}}{Z_t} = \sum_{i=1}^{N} p_{t,i} \exp(-\eta \ell_{i,t})$$

$$\leq \sum_{i=1}^{N} p_{t,i} \left(1 - \eta \ell_{i,t} + \frac{\eta^2}{2} \ell_{i,t}^2\right)$$

$$= 1 - \eta \langle \ell_t, p_t \rangle + \frac{\eta^2}{2} \langle \ell_t^2, p_t \rangle$$

$$\leq \exp \left(-\eta \langle \ell_t, p_t \rangle + \frac{\eta^2}{2} \langle \ell_t^2, p_t \rangle \right).$$

Above, the first inequality uses Equation (1.1), and the second inequality uses Equation (1.2). Therefore,

$$\ln \left(\frac{Z_{t+1}}{Z_t} \right) \leq -\eta \langle \ell_t, p_t \rangle + \frac{\eta^2}{2} \langle \ell_t^2, p_t \rangle. \quad (1.3)$$

Summing up Equation (1.3) over all $t = 1, \ldots, T$,

$$\ln(Z_{T+1}) - \ln(Z_1) = \ln(Z_{T+1}) - \ln(N)$$

$$\leq -\eta \sum_{t=1}^{T} \langle \ell_t, p_t \rangle + \frac{\eta^2}{2} \sum_{t=1}^{T} \langle \ell_t^2, p_t \rangle.$$

On the other hand, for any $i \in [N],$

$$\ln(Z_{T+1}) \geq \ln(w_{T+1,i}) = -\eta \sum_{t=1}^{T} \ell_{t,i}.$$  

Therefore, for any $i \in [N],$

$$\eta \sum_{t=1}^{T} \langle \ell_t, p_t \rangle \leq \eta \sum_{t=1}^{T} \ell_{t,i} + \ln(N) + \frac{\eta^2}{2} \sum_{t=1}^{T} \langle \ell_t^2, p_t \rangle,$$

which concludes the proof. □
Exercise 1.3. Show that if \( \ell_{t,i} \in [0, 1] \) for each \( i \) and \( t \), then there is some setting of \( \eta \) (in terms of \( N \) and \( T \)) that guarantees the regret of the learner using Algorithm \( \mathbf{7} \) after \( T \) rounds satisfies
\[
L_T - \min_{i \in [N]} L_{T,i} \leq \sqrt{2T \ln(N)}.
\]

Exercise 1.4. Deduce a bound on the expected regret for a learner using Algorithm \( \mathbf{5} \) (Randomized Weighted Majority) after \( T \) rounds for the setting from Section 1.1.1.

Exercise 1.5. Suppose it is promised that \( \min_{i \in [N]} L_{T,i} \leq L^* \). Show that if \( \ell_{t,i} \in [0, 1] \) for each \( i \) and \( t \), then there is some setting of \( \eta \) (in terms of \( N \) and \( L^* \)) that guarantees the regret of the learner using Algorithm \( \mathbf{7} \) after \( T \) rounds satisfies
\[
L_T - \min_{i \in [N]} L_{T,i} \leq O\left( \sqrt{L^* \log(N) + \log(N)} \right).
\]

1.2.3 Entropy maximization

The probability vector chosen by Hedge in round \( t \) turns out to be the solution to the following optimization problem:
\[
\min_{p \in \Delta^{N-1}} \sum_{s=1}^{t-1} \langle \ell_s, p \rangle - \frac{1}{\eta} \sum_{i=1}^{N} p_i \ln \frac{1}{p_i}.
\]

The objective balances between minimizing the total loss in previous rounds using \( p \) and maximizing the entropy of \( p \). This motivates the following alternative proof of Theorem 1.5.

Alternative proof of Theorem 1.5. The proof uses the fact that the excess total loss \( L_T - L_{T,q} \) compared to any fixed \( q \in \Delta^{N-1} \) can be related to the decreases in relative entropy \( \text{RE}(q, p_t) - \text{RE}(q, p_{t+1}) \) to \( q \) from round to round, where
\[
\text{RE}(q, p) := \sum_{i=1}^{N} q_i \ln \frac{q_i}{p_i}.
\]

Indeed,
\[
\text{RE}(q, p_{t+1}) - \text{RE}(q, p_t) = \sum_{i=1}^{N} q_i \ln \frac{p_{t+1,i}}{p_{t,i}}
\]
\[
= \eta \sum_{i=1}^{N} q_i \ell_{t,i} + \ln \left( \sum_{i=1}^{N} p_{t,i} \exp(-\eta \ell_{t,i}) \right)
\]
\[
\leq \eta \left( \langle \ell_t, q \rangle - \langle \ell_t, p_t \rangle \right) + \frac{\eta^2}{2} \langle \ell_t^2, p_t \rangle.
\]

This shows that in each round, if the learner incurs a significantly greater loss than the fixed distribution \( q \), then the update of the learner’s distribution moves it “closer” to \( q \) in
the sense of relative entropy. Now, summing this up from for \( t = 1, \ldots, T \) gives

\[
\text{RE}(q, p_{T+1}) - \text{RE}(q, p_1) \leq \eta \left( L_{T,a} - L_T \right) + \frac{\eta^2}{2} \sum_{t=1}^{T} \langle t^2, p_t \rangle.
\]

The claim follows because \( \text{RE}(q, p_1) \leq \ln(N) \) and \( \text{RE}(q, p_{T+1}) \geq 0 \). \( \square \)

## 1.3 Prediction with partial feedback

### 1.3.1 The setting

We return to the original online decision-making problem. The setting from Section 1.1.1 considers just two actions, but it is natural to generalize to the case where there are \( K \geq 2 \) possible actions \( \{1, \ldots, K\} \). However, we also need to consider the nature of the feedback. Is the learner told the “correct” action? What if there are multiple correct actions? What if each action has a different “cost” associated with it in each round? The possibilities are seemingly endless.

We focus on a specific but canonical case here: every action \( a \in \{1, \ldots, K\} \) incurs some cost \( c_t(a) \geq 0 \) in round \( t \), but the learner only observes the value of the loss for the chosen action \( a_t \). For simplicity, we begin with the version of the problem where there are no experts offering their advice. This is called the “multi-armed bandits” problem or online decision-making with “bandit feedback” (for historical reasons). The protocol is given in Algorithm 8.

**Algorithm 8** Protocol for online decision-making with bandit feedback

1: for \( t = 1, 2, \ldots \) do
2: Learner chooses action: \( a_t \in \{1, \ldots, K\} \).
3: Learner receives cost for chosen action: \( c_t(a_t) \geq 0 \).
4: end for

The total loss of picking a single action \( a \in \{K\} \) in all \( T \) rounds is \( L_{T,a} := \sum_{t=1}^{T} c_t(a) \), and the total loss of the learner after \( T \) rounds is \( L_T := \sum_{t=1}^{T} c_t(a_t) \). The regret of the learner after \( T \) rounds is

\[
L_T - \min_{a \in \{K\}} L_{T,a}.
\]

We would like an algorithm that guarantees low regret in expectation.

### 1.3.2 Inverse probability weighting

A simple approach to the multi-armed bandits problem is to try to reduce it back to the “full-information” setting of the linear loss game on the simplex from Section 1.2.1, where the entire loss vector is revealed after each round.
We first consider a simpler problem. Suppose we only want to know the average cost of a particular action, say, action 1, over $T$ rounds. How can we do this? Easy: pick $a_t = 1$ in each round; we are then guaranteed to observe $c_t(1)$ in every round.

Now suppose we only want to know the average costs of two particular actions, say, actions 1 and 2. You can’t just pick $a_t = 1$, since then we won’t see $c_t(2)$. However, if we let a fair coin toss determine whether we pick either action 1 or action 2, then we have a (50%, 50%) chance of seeing $c_t(1)$ and $c_t(2)$. The observed cost $c_t(a_t)$ is, in expectation,

$$\mathbb{E}(c_t(a_t)) = \frac{c_t(1) + c_t(2)}{2}. $$

Over $T$ rounds, we have

$$\mathbb{E}\left(\frac{1}{T} \sum_{t=1}^{T} c_t(a_t)\right) = \frac{1}{T} \sum_{t=1}^{T} \frac{c_t(1) + c_t(2)}{2},$$

and by the law of large numbers, $\sum_{t=1}^{T} c_t(a_t)/T$ will be close to this expected value with high probability as $T$ becomes large. This is interesting—it at least has some information about both actions—but it is not quite what we wanted.\footnote{One option that sounds plausible is to use action 1 in the first $T/2$ rounds, and action 2 in the last $T/2$ rounds. But this is reasonable only if

$$\frac{2}{T} \sum_{t=1}^{T/2} c_t(1) = \frac{2}{T} \sum_{t=T/2+1}^{T} c_t(1),$$

and an analogous statement holds for action 2. Such assumptions can be avoided using randomness.}

Here is a simple trick for isolating the individual costs. Consider the following estimators of $c_t(1)$ and $c_t(2)$:

$$\hat{c}_t(1) := \frac{1_{a_t=1}}{1/2} c_t(a_t);$$

$$\hat{c}_t(2) := \frac{1_{a_t=2}}{1/2} c_t(a_t).$$

Observe that one of these estimators is guaranteed to be zero (since we cannot have $a_t = 1$ and $a_t = 2$ both be true). If $a_t = 1$, then $\hat{c}_t(1)$ is twice the observed cost while $\hat{c}_t(2) = 0$ (and vice versa if $a_t = 2$). However, in expectation, both estimates are equal to the costs of the respective actions:

$$\mathbb{E}(\hat{c}_t(1)) = \Pr(a_t = 1) \cdot \frac{1_{a_t=1}}{1/2} c_t(1) + \Pr(a_t = 2) \cdot \frac{1_{a_t=2}}{1/2} c_t(2) = c_t(1);$$

$$\mathbb{E}(\hat{c}_t(2)) = \Pr(a_t = 1) \cdot \frac{1_{a_t=2}}{1/2} c_t(1) + \Pr(a_t = 2) \cdot \frac{1_{a_t=1}}{1/2} c_t(2) = c_t(2).$$
Again, over $T$ rounds, we have

$$
\mathbb{E}\left( \frac{1}{T} \sum_{t=1}^{T} \hat{c}_t(1) \right) = \frac{1}{T} \sum_{t=1}^{T} c_t(1)
$$

and by the law of large numbers, $\sum_{t=1}^{T} \hat{c}_t(1)/T$ will be close to this expected value with high probability as $T$ becomes large; an analogous statement holds for action 2.

More generally, suppose we fix a probability distribution $p_t := (p_t(1), \ldots, p_t(K)) \in \Delta^{K-1}$ over the $K$ actions, and then randomly draw $a_t \sim p_t$. For each particular action $a \in [K]$, define the estimator

$$
\hat{c}_t(a) := \frac{1_{\{a_t = a\}}}{p_t(a)} c_t(a_t).
$$

(1.4)

Again, in expectation, for all actions $a \in [K]$,

$$
\mathbb{E}(\hat{c}_t(a)) = \sum_{a' = 1}^{K} \Pr(a_t = a') \cdot \frac{1_{\{a' = a\}}}{p_t(a)} c_t(a')
\begin{align*}
  &= \Pr(a_t = a) \cdot \frac{1_{\{a = a\}}}{p_t(a)} c_t(a) \\
  &= c_t(a).
\end{align*}
$$

Therefore, the vector of cost estimates $\hat{c}_t$ is an unbiased estimator of the vector of (true) costs $c_t$. This technique is called inverse probability weighting.

We see that randomization provides a simple means to enable counterfactual inference: for any $a \in [K]$, we are able to estimate (in an unbiased manner) the loss that we would have incurred if we had chosen action $a$.

Although unbiasedness is a useful property of an estimator, it is not the only property that is relevant in applications. In particular, besides the mean of $\hat{c}_t(a)$, we may also care about, say, higher-order moments $\hat{c}_t(a)$, or other properties of its distribution. For instance, the variance of $\hat{c}_t(a)$ is

$$
\text{var}(\hat{c}_t(a)) = \mathbb{E}(\hat{c}_t(a)^2) - \mathbb{E}(\hat{c}_t(a))^2
\begin{align*}
  &= \sum_{a' = 1}^{K} \Pr(a_t = a') \cdot \frac{1_{\{a' = a\}}^2}{p_t(a)^2} c_t(a')^2 - c_t(a)^2 \\
  &= \Pr(a_t = a) \cdot \frac{1_{\{a = a\}}}{p_t(a)^2} c_t(a)^2 - c_t(a)^2 \\
  &= \left( \frac{1}{p_t(a)} - 1 \right) c_t(a)^2.
\end{align*}
$$

This can be large if $p_t(a)$ is small. This means that the estimates $\hat{c}_t$ may only be reliable if the probabilities $p_t$ are not too close to zero.
There is a trade-off involved in the choice of the probability distribution \( p_t \). The most “balanced” choice \( p_t \) is simply the uniform distribution: all actions are tried equally often, and the losses in each round are estimated with reasonably small variance (\( \text{var}(\hat{c}_t(a)) \leq K - 1 \) for all \( a \in [K] \) if \( c_t(a) \in [0, 1] \)). However, choosing actions in this way may yield poor performance (i.e., high regret); it is instead preferred to put lower probability on actions that seem to incur high loss, and higher probability on actions that seem to incur low loss. This is called the “exploration vs. exploitation” dilemma.

### 1.3.3 Hedging over actions with bandit feedback

A natural reduction from the bandit feedback setting to the full-information setting is to simply run an algorithm (like Hedge) designed for the full-information setting using the cost estimates \( \hat{c}_t \) described above in Equation (1.4) as the loss vectors \( \ell_t \). This is exactly the approach taken by the Exp3 algorithm (short for Exponential-weight algorithm for Exploration and Exploitation).

**Algorithm 9** Exp3 algorithm

**Require:** \( \eta > 0 \).

1. Let \( w_1 = (w_{1,1}, \ldots, w_{1,K}) := (1, \ldots, 1) \).
2. for \( t = 1, 2, \ldots \) do
3. Form vector \( p_t := w_t/Z_t \in \Delta^{K-1} \), where \( Z_t := \sum_{a=1}^K w_{t,a} \).
4. Choose action: randomly draw \( a_t \sim p_t \).
5. Receive cost for chosen action: \( c_t \geq 0 \).
6. Form cost estimates: \( \hat{c}_t(a) = 1_{\{a_t = a\}} c_t(a_t)/p_t(a) \) for all \( a \in [K] \).
7. Update: \( w_{t+1,a} := w_{t,a} \exp(-\eta \hat{c}_t(a)) \) for all \( a \in [K] \).
8. end for

**Theorem 1.6** (Exp3). For any sequence of cost vectors \( c_1, \ldots, c_T \in \mathbb{R}_+^K \), the total loss \( L_T := \sum_{t=1}^T c_t(a_t) \) incurred by a learner using Algorithm 9 (Exp3) with parameter \( \eta > 0 \) after \( T \) rounds satisfies

\[
\mathbb{E}(L_T) \leq L_{T,\eta} + \frac{\ln(K)}{\eta} + \frac{\eta}{2} \sum_{a'=1}^K c_t(a')^2, \quad a \in [K].
\]

**Proof.** Let \( \mathbb{E}_t \) denote the conditional expectation operator given all information through the first \( t - 1 \) rounds. We use the following properties about the estimated costs from Section 1.3.2:

\[
\mathbb{E}(\hat{c}_t(a)) = c_t(a);
\]

\[
\mathbb{E}_t(\hat{c}_t(a)^2) = \frac{c_t(a)^2}{p_t(a)}.
\]
Observe that the evolution of the weights $w_t$ in Exp3 is the same as that of Hedge using loss vectors $\ell_t = \hat{c}_t$. Therefore, we have from Theorem 1.5 that, for any $a \in [K]$,

$$\sum_{t=1}^{T} \langle \hat{c}_t, p_t \rangle \leq \sum_{t=1}^{T} \hat{c}_t(a) + \frac{\ln(K)}{\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \langle \hat{c}_t^2, p_t \rangle. \quad (1.5)$$

Now we take expectations of both sides of Equation (1.5). First, observe that each term on the left-hand side of Equation (1.5) has conditional expectation

$$E_t(\langle \hat{c}_t, p_t \rangle) = \sum_{a=1}^{K} p_t(a) E_t(\hat{c}_t(a)) = \sum_{a=1}^{K} p_t(a) c_t(a) = E(c_t(a)).$$

This shows that the left-hand side of Equation (1.5) is the expected loss of the learner $E(L_T)$.

Next, the first summation on the right-hand side of Equation (1.5) has expectation

$$\sum_{t=1}^{T} E(\hat{c}_t(a)) = \sum_{t=1}^{T} c_t(a) = L_{T,a}.$$

Finally, each term in the second summation on the right-hand side of Equation (1.5) has expectation

$$E(\langle \hat{c}_t^2, p_t \rangle) = \sum_{a'=1}^{K} E(p_t(a') \hat{c}_t(a')^2)$$
$$= \sum_{a'=1}^{K} E(p_t(a') E(\hat{c}_t(a')^2))$$
$$= \sum_{a'=1}^{K} E \left( \frac{p_t(a') c_t(a')^2}{p_t(a')} \right)$$
$$= \sum_{a'=1}^{K} c_t(a')^2.$$

So, the inequality relating the expectations of each side of Equation (1.5) finally becomes

$$E(L_T) \leq L_{T,a} + \frac{\ln(K)}{\eta} + \frac{\eta}{2} \sum_{a'=1}^{K} c_t(a')^2,$$

which completes the proof.  

Exercise 1.6. Show that if $c_t(a) \in [0, 1]$ for each $a$ and $t$, then there is some setting of $\eta$ (in terms of $K$ and $T$) that guarantees the expected regret of the learner using Algorithm 9 after $T$ rounds satisfies

$$E(L_T) - \min_{a \in [K]} L_{T,a} \leq \sqrt{2KT \ln(K)}. \quad (\text{It turns out the expected regret bound of Exp3 is suboptimal. There is a different algorithm with expected regret bound } O(\sqrt{KT}), \text{ and this is optimal.})$$
1.3.4 Hedging over experts with bandit feedback

Now we bring experts back into the picture. The protocol is given in Algorithm 10.

**Algorithm 10 Protocol for online decision-making with bandit feedback and expert advice**

1: for \( t = 1, 2, \ldots \) do
2: Learner receives experts’ actions: \( b_{t,i} \in [K] \) for \( i \in [N] \).
3: Learner chooses action: \( a_t \in [K] \).
4: Learner receives cost for chosen action: \( c_t(a_t) \geq 0 \).
5: end for

The total loss of expert \( i \in [N] \) in all \( T \) rounds is \( L_{T,i} := \sum_{t=1}^{T} c_t(b_{t,i}) \), and the total loss of the learner after \( T \) rounds is \( L_T := \sum_{t=1}^{T} c_t(a_t) \). Hence, the regret of the learner after \( T \) rounds is

\[
L_T - \min_{i \in [K]} L_{T,i}.
\]

We would like an algorithm that guarantees low regret in expectation. Note that this is a very different notion of regret than what was considered in Section 1.3.1 and Section 1.3.3. This is because we are comparing the performance of the learner to that of the best expert, rather than the best action. This is an important difference because there may not be any single action that has low cost over all \( T \) rounds. However, there may be an expert who is good at picking different actions in different rounds. This setting is often called the contextual bandit setting, because the experts can be viewed as making use of context information in each round to recommend actions, which is natural in real-world applications.

The main idea of the following algorithm is similar to that of Exp3. We use a full-information algorithm (Hedge) with estimated costs for the experts. In this case, we need to estimate the cost incurred by an expert \( i \) in round \( t \): since expert \( i \) recommends action \( b_{t,i} \in [K] \) in round \( t \), we use

\[
\hat{c}_{t}(b_{t,i}) = \frac{1_{\{a_t = b_{t,i}\}}}{p_t(a)} c_t(a_t)
\]

just as in Equation (1.4). The vector of cost estimates for experts can then be used to define weights over experts, which in turn are used to define a probability distribution over actions. The resulting algorithm is called Exp4 (short for Exponential-weight algorithm for Exploration and Exploitation using Expert advice).

**Theorem 1.7 (Exp4).** For any sequence of cost vectors \( c_1, \ldots, c_T \in \mathbb{R}_+^K \), the total loss \( L_T := \sum_{t=1}^{T} c_t(a_t) \) incurred by a learner using Algorithm 10 (Exp4) with parameter \( \eta > 0 \) after \( T \) rounds satisfies

\[
\mathbb{E}(L_T) \leq L_{T,i} + \frac{\ln(N)}{\eta} + \frac{\eta}{2} \sum_{a=1}^{K} c_t(a)^2, \quad i \in [N].
\]
Algorithm 11 Exp4 algorithm

Require: $\eta > 0$.

1: Let $w_1 = (w_{1,1}, \ldots, w_{1,N}) := (1, \ldots, 1)$.

2: for $t = 1, 2, \ldots$ do

3: Receive experts’ actions: $b_{t,i} \in [K]$ for $i \in [N]$.

4: Form vector $p_t \in \Delta^{K-1}$ given by $p_t(a) := \sum_{i=1}^{N} w_{t,i} 1_{\{b_{t,i} = a\}} / \sum_{i=1}^{N} w_{t,i}$. 

5: Choose action: randomly draw $a_t \sim p_t$.

6: Receive cost for chosen action: $c_t(a_t) \geq 0$.

7: Form cost estimates: $\hat{c}_t(a) = 1_{\{a = a_t\}} c_t(a_t) / p_t(a)$ for all $a \in [K]$.

8: Update: $w_{t+1,i} := w_{t,i} \exp(-\eta \hat{c}_t(b_{t,i}))$ for all $i \in [N]$.

9: end for

The proof of Theorem 1.7 is very similar to that of Theorem 1.6.

Exercise 1.7. Prove Theorem 1.7.

Exercise 1.8. Show that if $c_t(a) \in [0, 1]$ for each $a$ and $t$, then there is some setting of $\eta$ (in terms of $K$, $N$, and $T$) that guarantees the expected regret of the learner Algorithm 11 satisfies

$$\mathbb{E}(L_T) - \min_{i \in [N]} L_{T,i} \leq \sqrt{2KT \ln(N)}.$$

1.4 Bibliographic references

The Weighted Majority and Randomized Weighted Majority algorithms are due to Littlestone and Warmuth (1994). The impossibility of sublinear regret is due to Cover (1965). The Hedge algorithm is due to Freund and Schapire (1997), and the analysis based on relative entropy is from Freund and Schapire (1999). The importance weighting trick is attributed to Horvitz and Thompson (1952). The Exp3 and Exp4 algorithms are due to Auer et al. (2002).
Bibliography


