1 Prediction with expert advice

Algorithm 1 Protocol for online decision-making with expert advice

1: for \( t = 1 \) to \( T \) do
2: Receive experts’ recommendations: \( b_{t,i} \in \{\pm 1\} \) for \( i \in [N] \).
3: Choose action: \( a_t \in \{\pm 1\} \).
4: Receive feedback: \( y_t \in \{\pm 1\} \) is the correct action.
5: end for

Recall that the “Follow the weighted majority” algorithm has the following guarantee. Letting \( M_T \) denote the number of mistakes made by the learner in all \( T \) rounds, and \( M_{T,i} \) denote the number of mistakes made by expert \( i \in [N] \) in all \( T \) rounds, we showed

\[
M_T \leq 2.41 \min_{i \in [N]} M_{T,i} + O(\log(N)).
\]

1.1 Regret

The leading constant of 2.41 above makes the bound useless when the best expert makes a mistake around 21% of the time. Can it be improved? Ideally, it would simply be equal to one, so that we obtain a bound on the relative mistake difference \( M_T - M_{T,i} \), for any \( i \in [N] \), of \( O(\log(N)) \). We call this relative difference the regret (of the learner) to expert \( i \), and call

\[
\mathcal{R}_T := M_T - \min_{i \in [N]} M_{T,i}
\]

the regret (to the best expert). Even a bound sublinear in \( T \) would be interesting, as it would imply that the average regret tends to zero as \( T \to \infty \).

Unfortunately, it is not generally possible to get a sublinear regret bound.

Theorem 1. Let \( b_{t,1} = -1 \) and \( b_{t,2} = +1 \) for all \( t \leq T \). For any learning algorithm \( A \), there is a sequence \( y_1, y_2, \ldots, y_T \in \{\pm 1\} \) such that \( \mathcal{R}_T \geq T/2 \).

Proof. For any \( y_1, y_2, \ldots, y_T \in \{\pm 1\} \), the better of the two experts makes at most \( T/2 \) mistakes. Therefore there is some sequence for which \( A \) makes \( T \) mistakes while the best expert makes only \( T/2 \) mistakes. \( \square \)
1.2 Randomized weighted majority

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 your adversary and assuming that her decisions are made before your 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 first round.

How does randomization help? For the construction in the proof of Theorem 1, we can simply pick actions in \{\pm 1\} uniformly at random to get a bound of \(T/2\) on the expected number of mistakes for any sequence of \(y_1, \ldots, y_T\) (so the expected regret is zero).

More generally, we may use a simple variant of the “Follow the weighted majority” algorithm.

**Algorithm 2** Randomized Weighted Majority (RWM)

**input** \(\varepsilon \in (0, 1)\).

1. Let \(w_{1,i} := 1\) for \(i \in [N]\).
2. for \(t = 1\) to \(T\) do
3. Receive experts’ recommendations: \(b_{t,i} \in \{\pm 1\}\) for \(i \in [N]\).
4. Choose action: randomly draw \(a_t \sim p_t := (p_t, -1, p_t, 1)\), where
   \[
   p_{t,a} \propto \sum_{i \in [N]: b_{t,i} = a} w_{t,i}, \quad a \in \{\pm 1\}.
   \]
5. Receive feedback: \(y_t \in \{\pm 1\}\) is the correct action.
6. Update:
   \[
   w_{t+1,i} := \begin{cases} 
   (1 - \varepsilon)w_{t,i} & \text{if } b_{t,i} \neq y_t; \\
   w_{t,i} & \text{otherwise}.
   \end{cases}
   \]
7. end for

There are two differences in RWM relative to “Follow the weighted majority”. First, we have replaced the value of \(1/2\) in the update step with an arbitrary fraction \(1 - \varepsilon\). This is simply because there is nothing special about the factor of \(1/2\), so we may as well pick the \(\varepsilon\) that gives the best guarantee—think of it as a “tuning parameter”.

Second, we now choose the action randomly with probability proportional to the total weight of experts choosing each action. Equivalently, we pick an expert \(i_t \in [N]\) at random with probability proportional to \(w_{t,i}\), and the simply set \(a_t := b_{t,i_t}\); we then only count the expected number of
mistakes. The randomization and averaging are crucial: without them, the learner would be subject to the lower bound in Theorem 1. Essentially, they provide a way to transform a difficult discrete problem into an easier continuous problem. Suppose \( \tilde{\mathbf{w}}_t := \left( w_{t,1}, \ldots, w_{t,N} \right)/Z_t \) is the normalized vector of weights on experts (and \( Z_t := w_{t,1} + \cdots + w_{t,N} \)), and the correct action is \( y_t \in \{\pm 1\} \). Then the probability that RWM makes a mistake—i.e., the fraction of weight on mistaken experts—is a linear function of \( \tilde{\mathbf{w}}_t \):

\[
\Pr(b_{t,i} \neq y_t) = \sum_{i=1}^{N} \Pr(i_t = i) \mathbb{1}\{b_{t,i} \neq y_t\} \\
= \sum_{i=1}^{N} \frac{w_{t,i}}{Z_t} \mathbb{1}\{b_{t,i} \neq y_t\} \\
= \langle \ell_t, \tilde{\mathbf{w}}_t \rangle
\]

where \( \ell_t := (\mathbb{1}\{b_{t,1} \neq y_t\}, \ldots, \mathbb{1}\{b_{t,N} \neq y_t\}) \in \{0, 1\}^N \) is the vector indicating which experts make a mistake. In this notation, the expected number of mistakes by RWM in all \( T \) rounds is

\[
\mathbb{E}(M_T) = \sum_{t=1}^{T} \Pr(a_t \neq y_t) = \sum_{t=1}^{T} \Pr(b_{t,i} \neq y_t) = \sum_{t=1}^{T} \langle \ell_t, \tilde{\mathbf{w}}_t \rangle. \tag{1}
\]

Instead of giving an analysis of RWM, we’ll move on to look at a more general online learning problem where the loss vector \( \ell_t := (\ell_{t,1}, \ldots, \ell_{t,N}) \) can be any vector in the non-negative orthant \( \mathbb{R}_N^+ := \{ \mathbf{x} \in \mathbb{R}^N : x_i \geq 0 \forall i \in [N] \} \) of \( \mathbb{R}^N \), and the goal is to choose probability vectors \( \tilde{\mathbf{w}}_t \) to minimize the right-hand side of (1).

1.3 The linear loss game on the simplex

Consider the following online learning protocol, which we call the linear loss game on the probability simplex \( \Delta^{N-1} \subset \mathbb{R}^N \).

<table>
<thead>
<tr>
<th>Algorithm 3</th>
<th>Protocol for linear loss game on the simplex</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>for ( t = 1 ) to ( T ) do</td>
</tr>
<tr>
<td></td>
<td>Choose vector: ( \mathbf{x}_t \in \Delta^{N-1} ).</td>
</tr>
<tr>
<td></td>
<td>Receive feedback: ( \ell_t \in \mathbb{R}_+^N ).</td>
</tr>
<tr>
<td></td>
<td>Incur loss: ( \langle \ell_t, \mathbf{x}_t \rangle ).</td>
</tr>
<tr>
<td>5:</td>
<td>end for</td>
</tr>
</tbody>
</table>

3
In this setting, we forego the notions of experts and actions, and simply require the learner to pick a probability vector in $\Delta^{N-1}$ in each round. One can think of the learner as hedging over $N$ different options. The loss vector $\ell_t \in \mathbb{R}^N_+$ determines the loss incurred for any $x \in \Delta^{N-1}$, specifically $\langle \ell_t, x \rangle \in \mathbb{R}_+$. Therefore, the loss incurred by the learner in round $t$ is $\langle \ell_t, x_t \rangle$, and the total loss over all $T$ rounds is $L_T := \sum_{t=1}^T \langle \ell_t, x_t \rangle$. We shall compare the total loss of the learner to the total loss incurred by any fixed vector $u \in \Delta^{N-1}$ over all $T$ rounds, $L_T, u := \sum_{t=1}^T \langle \ell_t, u \rangle$. Clearly, the minimum of $L_T, u$ over $u \in \Delta^{N-1}$ is achieved at some vertex of $\Delta^{N-1}$, so we write the total loss incurred by the vertex $e_i$ as $L_{T,i}$. The regret of the learner to the best vector $u \in \Delta^{N-1}$ in hindsight is $R_T := L_T - \min_{u \in \Delta^{N-1}} L_T, u$.

The generalization of RWM is the following algorithm, called “Hedge”.

**Algorithm 4 Hedge**

1: Let $w_{1,i} := 1$ for $i \in [N]$.
2: for $t = 1$ to $T$ do
3: Choose vector $x_t := \frac{w_t}{Z_t} \in \Delta^{N-1}$, where $Z_t := \sum_{i=1}^N w_{t,i}$.
4: Receive feedback: $\ell_t \in \mathbb{R}^N_+$.
5: Incur loss: $\langle \ell_t, x_t \rangle$.
6: Update: $w_{t+1,i} := w_{t,i} \exp(-\eta \ell_{t,i})$, $i \in [N]$.
7: end for

**Theorem 2.** Pick any sequence of loss vectors $\ell_1, \ell_2, \ldots, \ell_T \in \mathbb{R}^N_+$. The total loss incurred by a learner using Algorithm 4 with parameter $\eta > 0$ is bounded as

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

for any $i \in [N]$, where $\ell_t^2 := (\ell_{t,1}^2, \ell_{t,2}^2, \ldots, \ell_{t,N}^2)$.

**Remark 1.** If $\ell_t \in [0, 1]^N$ for all $t \leq T$, then a different analysis gives the bound, for any $i \in [N]$,

$$\sum_{t=1}^T \langle \ell_t, x_t \rangle \leq \frac{1}{1-e^{-\eta}} \left( \eta \sum_{t=1}^T \ell_{t,i} + \ln(N) \right).$$

However, the form of the bound in (2) will be useful in later applications.
The proof of Theorem 2 is very similar to the analysis of the “Follow 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 options 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 option after all \( T \) rounds.

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

\[
\exp(z) \leq 1 + z + z^2/2 \quad \text{for all } z \leq 0; \\
\exp(z) \geq 1 + z \quad \text{for all } z \in \mathbb{R}.
\]

We consider the relative change in total weight after round \( t \):

\[
\frac{Z_{t+1}}{Z_t} = \frac{\sum_{i=1}^{N} w_{t+1,i}}{Z_t} = \frac{\sum_{i=1}^{N} x_{t,i} \exp(-\eta \ell_{t,i})}{Z_t} \leq \sum_{i=1}^{N} x_{t,i} \left( 1 - \eta \ell_{t,i} + \frac{\eta^2}{2} \ell_{t,i}^2 \right) \]

\[
= 1 - \eta \langle \ell_t, x_t \rangle + \frac{\eta^2}{2} \langle \ell_t^2, x_t \rangle \\
\leq \exp\left( -\eta \langle \ell_t, x_t \rangle + \frac{\eta^2}{2} \langle \ell_t^2, x_t \rangle \right),
\]

so

\[
\ln\left( \frac{Z_{t+1}}{Z_t} \right) = \ln\left( \sum_{i=1}^{N} x_{t,i} \exp(-\eta \ell_{t,i}) \right) \leq -\eta \langle \ell_t, x_t \rangle + \frac{\eta^2}{2} \langle \ell_t^2, x_t \rangle. \tag{4}
\]

Summing up the bound (4) over all \( t \),

\[
\ln(Z_{T+1}) - \ln(Z_1) = \ln(Z_{T+1}) - \ln(N) \leq -\eta \sum_{t=1}^{T} \langle \ell_t, x_t \rangle + \frac{\eta^2}{2} \sum_{t=1}^{T} \langle \ell_t^2, x_t \rangle.
\]

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 \( i \in [N] \),

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

\( \square \)
Whenever $\ell_t \in [0,1]^N$, we have $\ell^2_{t,i} \leq 1$; in this case, it is easy to check that using $\eta := \sqrt{2 \ln(N)/T}$ guarantees the sublinear bound
\[ \mathcal{R}_T \leq \sqrt{2T \ln(N)}. \]

A more clever choice of $\eta$ can give a somewhat better bound, as the following corollary shows.

**Corollary 1.** Pick any sequence of loss vectors $\ell_1, \ell_2, \ldots, \ell_T \in [0,1]^N$. If $L_* \geq \min_{i \in [N]} L_{T,i}$ and
\[ \eta := 2\left(\sqrt{\frac{2\ln(N)}{L_*}} \left(1 + \frac{2\ln(N)}{L_*}\right) - \frac{2\ln(N)}{L_*}\right) > 0, \]
then the regret of learner using Algorithm 2 with parameter $\eta$ is bounded as
\[ \mathcal{R}_T \leq 2\sqrt{2L_* \ln(N)} + 8 \ln(N). \]

**Remark 2.** Based on the bound from (3), one can see that there is a slightly different choice of $\eta > 0$ that yields the bound
\[ \mathcal{R}_T \leq \sqrt{2L_* \ln(N)} + \ln(N). \]

**Proof.** The main observation is that $\ell^2_{t,i} \leq \ell_{t,i}$ since $\ell_{t,i} \in [0,1]$, so the final term in (2) is bounded above by $(\eta/2)L_T$. Therefore, if $\eta \in (0,2)$, we have
\[ \mathcal{R}_T \leq \left((1 - \eta/2)^{-1} - 1\right) \min_{i \in [N]} L_{T,i} + (1 - \eta/2)^{-1} \eta^{-1} \ln(N) \]
\[ \leq \left((1 - \eta/2)^{-1} - 1\right) L_* + (1 - \eta/2)^{-1} \eta^{-1} \ln(N). \]
The chosen value of $\eta$ minimizes this latter bound on $\mathcal{R}_T$. 

The choice of $\eta$ in Corollary 1 requires an upper bound on $\min_{i \in [N]} L_{T,i}$. A good bound can be found adaptively using a guess-and-check approach commonly known as the “doubling trick”.

We now give another proof of Theorem 2 based on relative entropy divergence (also called Kullback-Leibler divergence). For any probability vectors $p, q \in \Delta^{N-1}$, the **relative entropy divergence** of $q$ from $p$ is
\[ \text{RE}(p\|q) := \sum_{i=1}^{N} p_i \ln \frac{p_i}{q_i}. \]

It can be checked this quantity is non-negative and is zero if and only if $p = q$. However, it is not symmetric—i.e., $\text{RE}(p\|q) \neq \text{RE}(q\|p)$ in general—nor does it satisfy the triangle inequality (and hence it is called a divergence rather than a distance or metric).
Alternative proof of Theorem 2. Fix any $u \in \Delta^{N-1}$. Recall that $x_{t+1,i} \propto x_{t,i} \exp(-\eta \ell_{t,i})$ for any $t \leq T$. Therefore

$$
\text{RE}(u \| x_{t+1}) - \text{RE}(u \| x_t) = \sum_{i=1}^{N} u_i \ln \left( \frac{x_{t,i}}{x_{t+1,i}} \right)
= \sum_{i=1}^{N} u_i \ln \left( \frac{x_{t,i}}{x_{t,i} \exp(-\eta \ell_{t,i}) / \sum_{j=1}^{N} x_{t,j} \exp(-\eta \ell_{t,j})} \right)
= \eta \sum_{i=1}^{N} u_i \ell_{t,i} + \ln \left( \sum_{i=1}^{N} x_{t,i} \exp(-\eta \ell_{t,i}) \right)
\leq \eta \langle \ell_t, u \rangle - \eta \langle \ell_t, x_t \rangle + \frac{\eta^2}{2} \langle \ell_t^2, x_t \rangle \quad (5)
$$

where the inequality follows from the original proof of Theorem 2. Summing up the bound (5) over all $t$,

$$
\text{RE}(u \| x_{T+1}) - \text{RE}(u \| x_1) \leq \eta \sum_{t=1}^{T} \langle \ell_t, u \rangle - \eta \sum_{t=1}^{T} \langle \ell_t, x_t \rangle + \frac{\eta^2}{2} \sum_{t=1}^{T} \langle \ell_t^2, x_t \rangle.
$$

After some rearranging, the claim follows because RE($u \| x_1$) $\leq \ln(N)$ and RE($u \| x_{T+1}$) $\geq 0$. \qed

In the case that $\ell_t \in [0,1]^N$ (so that $\ell_{t,i}^2 \leq \ell_{t,i}$), the bound in (5) further simplifies to

$$
\text{RE}(u \| x_{t+1}) \leq \text{RE}(u \| x_t) - \eta \left( (1 - \eta/2) \langle \ell_t, x_t \rangle - \langle \ell_t, u \rangle \right).
$$

For $\eta \in (0, 2)$, this says that if the loss of $x_t$ is much larger than the loss of $u$ in round $t$, then the learner’s next probability vector $x_{t+1}$ is closer to $u$ (than $x_t$ is) in terms of relative entropy divergence.

The exponential weights used by Hedge can be seen to generalize the Halving algorithm and (Randomized) Weighted Majority. In that sense, the weights seem rather natural. However, it is not obvious why relative entropy should be involved in the analysis. Is there an interpretation of the exponential weights used by Hedge that makes a transparent connection to relative entropy?

**Lemma 1.** The probability vector $x_{t+1}$ in Algorithm 4 is the solution to the optimization problem

$$
\min_{x \in \Delta^{N-1}} \{ \eta \langle \ell_t, x \rangle + \text{RE}(x \| x_t) \}. \quad (6)
$$

7
Proof. The proof uses the fact that the optimization problem (6) is a convex optimization problem over the probability simplex in which strong duality holds. We explain what this entails without getting into too many details.

Define $\mathcal{L}(x, \lambda)$ be the Lagrangian function associated with the optimization problem (6):

$$\mathcal{L}(x, \lambda) := \eta \langle \ell_t, x \rangle + \text{RE}(x\|x_t) + \lambda \left( \sum_{i=1}^{N} x_i - 1 \right).$$

The Lagrangian permits the removal of the “sums to one” constraint on $x$ (i.e., the constraint $\sum_{i=1}^{N} x_i = 1$ implicit in “$x \in \Delta^{N-1}$”). The term added to the original objective function can be non-zero whenever $x$ violates the constraint. If $x$ violates the “sums to one” constraint, then maximizing the Lagrangian $\mathcal{L}(x, \lambda)$ over $\lambda \in \mathbb{R}$ yields an arbitrarily large value. Therefore, we conclude that

$$\min_{x \in \Delta^{N-1}} \eta \langle \ell_t, x \rangle + \text{RE}(x\|x_t) = \min_{x \in \mathbb{R}^N_+} \max_{\lambda \in \mathbb{R}} \mathcal{L}(x, \lambda).$$

Note that if we switch the order of the min and max on the right-hand side, then the value cannot increase—it may only decrease:

$$\min_{x \in \mathbb{R}^N_+} \max_{\lambda \in \mathbb{R}} \mathcal{L}(x, \lambda) \geq \max_{\lambda \in \mathbb{R}} \min_{x \in \mathbb{R}^N_+} \mathcal{L}(x, \lambda).$$

This is a property known as weak duality, which always holds. The fact that the optimization problem (6) satisfies strong duality implies that the value actually does not change at all when the order of the min and max are swapped:

$$\min_{x \in \mathbb{R}^N_+} \max_{\lambda \in \mathbb{R}} \mathcal{L}(x, \lambda) = \max_{\lambda \in \mathbb{R}} \min_{x \in \mathbb{R}^N_+} \mathcal{L}(x, \lambda).$$

It turns out we can analytically determine the minimum of $\mathcal{L}(x, \lambda)$ over $x \in \mathbb{R}^N_+$ (in terms of any fixed $\lambda \in \mathbb{R}$) from first-order optimality conditions. For any $i \in [N]$, the partial derivative of $\mathcal{L}(x, \lambda)$ with respect to $x_i$ is

$$\frac{\partial \mathcal{L}(x, \lambda)}{\partial x_i} = \eta \ell_{t,i} + 1 + \ln \frac{x_i}{x_{t,i}} + \lambda.$$ 

Letting $Z_\lambda := \exp(1 + \lambda) > 0$, we see that the partial derivatives for all $i \in [N]$ are zero when

$$x_i = x_{\lambda,i} := \frac{x_{t,i} \exp(-\eta \ell_{t,i})}{Z_\lambda}, \quad \forall i \in [N].$$
Therefore, since $x_\lambda := (x_{\lambda,1}, x_{\lambda,2}, \ldots, x_{\lambda,N}) \in \mathbb{R}_+^N$ (in fact, it is in the strictly positive orthant $\mathbb{R}_+^N := \{x \in \mathbb{R}^N : x_i > 0 \forall i \in [N]\}$ whenever $x_t$ is), it is the unique solution to the minimization of $L(x, \lambda)$ over $x \in \mathbb{R}_+^N$. This is useful because we now know that the optimal solution to the optimization problem \((6)\) must have the form $x_\lambda$ for the value of $\lambda \in \mathbb{R}$ that solves the maximization problem

$$\max_{\lambda \in \mathbb{R}} L(x_\lambda, \lambda) = \max_{\lambda \in \mathbb{R}} - \left( \frac{1}{Z_\lambda} \sum_{j=1}^{N} x_{t,j} \exp(-\eta \ell_{t,j}) + \lambda \right).$$

Again, the solution can be determined analytically from first-order optimality conditions. The derivative of $L(x_\lambda, \lambda)$ with respect to $\lambda$ is equal to zero when

$$Z_\lambda = \exp(1 + \lambda) = \sum_{j=1}^{N} x_{t,j} \exp(-\eta \ell_{t,j}).$$

Therefore the optimal solution $x \in \Delta^{N-1}$ to the optimization problem in \((6)\) is given by

$$x_i = \frac{x_{t,i} \exp(-\eta \ell_{t,i})}{\sum_{j=1}^{N} x_{t,j} \exp(-\eta \ell_{t,j})}$$

for each $i \in [N]$, which is exactly $x_{t+1}$ as given in Algorithm 4.

From Lemma 1 it is easy to see that $x_{t+1}$ can equivalently be obtained as the solution to

$$\min_{x \in \Delta^{N-1}} \left\{ \eta \sum_{\tau=1}^{t} (\ell_{\tau}, x) + \operatorname{RE}(x \| x_1) \right\}.$$

### 2 Prediction with partial feedback

So far, we have considered online decision-making with just two possible actions; the feedback provided is which of the two actions is the “correct” one. The generalization to $K \geq 2$ actions (say, $\{1, 2, \ldots, K\}$) is rather straightforward, provided that the feedback is which of the $K$ actions is the “correct” one. We can simply run Algorithm 4 with the loss vector $\ell_t := (\mathbb{I}\{b_{t,1} \neq y_t\}, \mathbb{I}\{b_{t,2} \neq y_t\}, \ldots, \mathbb{I}\{b_{t,N} \neq y_t\})$, where $y_t \in [K]$ is the correct action.

However, in many applications, the feedback provided may be limited to whether or not the chosen action was correct or not. That is, if the
chosen action is $a_t$, then the feedback provided is not $y_t$ itself, but rather $\mathbb{I}\{a_t \neq y_t\}$. For instance, an internet company may have several ads that can be potentially shown to a website viewer, but can only select one to display. The feedback provided is whether or not the viewer clicks on the chosen ad; the company does not know whether or not the viewer would have clicked on some other ad. More generally, if there is a loss associated with each of $K$ possible actions, then the only feedback provided may be the loss of the chosen action, and not any of other other actions. We call this partial feedback, or more specifically (and for historical reasons), bandit feedback.

For simplicity, we’ll first just consider a simple setting of online decision-making with bandit feedback that ignores expert advice. However, we shall generalize the prediction setting from before to allow every action to incur an arbitrary non-negative loss in a given round; in round $t$, the losses will be encoded in a loss vector $c_t := (c_t(1), c_t(2), \ldots, c_t(K)) \in \mathbb{R}^K$. This added flexibility allows us to model settings in which, say, multiple actions may be correct (or partially correct, etc.). We will consider the total loss of picking any single action $a \in [K]$ in all $T$ rounds, $L_{T,a} := \sum_{t=1}^{T} c_t(a)$, and also the total loss of the learner $L_T := \sum_{t=1}^{T} c_t(a_t)$. We are interested in learning algorithms that guarantee low expected regret to the best action:

$$\mathbb{E}(R_T) := \mathbb{E}(L_T) - \min_{a \in [K]} L_{T,a}.$$ 

**Algorithm 5** Protocol for online decision-making with bandit feedback

1. for $t = 1$ to $T$ do
2.   Choose action: $a_t \in [K]$.
3. end for

2.1 The importance weighting trick

A simple approach to online decision-making with bandit feedback is to try to reduce it back to the full-information setting (like the setting in Algorithm 3), where the entire loss vector is revealed after each round.

First, suppose you only have to care about the loss of a particular action. For concreteness, say it is action 1. How can you do this? Easy: pick $a_t = 1$; you are then guaranteed to observe $c_t(1)$.
Now, suppose you only have to care about the losses of two particular actions, say, actions 1 and 2. You can’t just pick $a_t = 1$, since then you won’t see $c_t(2)$. However, if you let a fair coin toss determine whether you pick either action 1 and action 2, then you have a 50%-50% chance of seeing $c_t(1)$ and $c_t(2)$. The observed loss $c_t(a_t)$ is, in expectation,

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

the simple average of the losses of actions 1 and 2. This may be useful to know, but it may not be as good as knowing the individual losses themselves.

Here is a simple trick to isolate the individual losses. Consider the following estimators of $c_t(1)$ and $c_t(2)$:

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

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

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 loss while $\hat{c}_t(2) = 0$ (and vice versa if $a_t = 2$). However, in expectation, both estimates are equal to the losses of the respective actions:

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

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

More generally, suppose a priori you fix a probability distribution $p_t := (p_t(1), p_t(2), \ldots, p_t(K)) \in \Delta^{K-1}$ over the $K$ actions, and then randomly draw $a_t \sim p_t$. For each particular action $a \in [K]$, define the estimator

$$\hat{c}_t(a) := \left( \frac{1}{2} \mathbb{1}\{a_t = a\} \middle/ p_t(a) \right) c_t(a_t).$$

(7)
Again, in expectation, for all actions $a \in [K]$,

$$
\mathbb{E}(\hat{c}_t(a)) = \sum_{a'=1}^{K} \Pr(a_t = a') \cdot \frac{1}{p_t(a)} \cdot 1\{a' = a\} c_t(a') \\
= \Pr(a_t = a) \cdot \frac{1}{p_t(a)} \cdot 1\{a = a\} c_t(a) \quad \text{(all but the $a$-th term are zero)} \\
= c_t(a). \quad (8)
$$

Therefore, the vector of loss estimates $\hat{c}_t$ is an unbiased estimator of the vector of (true) losses $c_t$.

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

$$
\text{var}(\hat{c}_t(a)) = \mathbb{E}(\hat{c}_t(a)^2) - \mathbb{E}(\hat{c}_t(a))^2 \\
= \sum_{a'=1}^{K} \Pr(a_t = a') \cdot \frac{1}{p_t(a)} \cdot 1\{a' = a\}^2 c_t(a')^2 - c_t(a)^2 \\
= \Pr(a_t = a) \cdot \frac{1}{p_t(a)^2} c_t(a)^2 - c_t(a)^2 \\
= \left( \frac{1}{p_t(a)} - 1 \right) c_t(a)^2. \quad (9)
$$

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.
2.2 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 (something like Algorithm 2 or Algorithm 4) designed for the full-information setting with the loss estimates $\hat{\ell}_t$ described above in (7). This is exactly the approach taken by Algorithm 6, which is called “Exp3” (short for “Exponential-weight algorithm for Exploration and Exploitation”).

**Algorithm 6 Exp3**

**input** $\eta > 0$.

1. Let $w_1(a) := 1$ for $a \in [K]$.

2. for $t = 1$ to $T$ do

3. Choose action: randomly draw $a_t \sim p_t \in \Delta^{K-1}$, where $p_t(a) := w_t(a)/Z_t$ and $Z_t := \sum_{a=1}^{K} w_t(a)$.

4. Receive feedback: action $a_t$ incurs loss $c_t(a_t) \in \mathbb{R}_+$. 

5. Update: for each $a \in [K]$, 

   

   $$w_{t+1}(a) := w_t(a) \exp(-\eta \hat{c}_t(a))$$

   

   $$= \begin{cases} w_t(a) & \text{if } a \neq a_t; \\
   w_t(a) \exp(-\eta c_t(a_t)/p_t(a_t)) & \text{if } a = a_t.
   \end{cases}$$

6. end for

**Theorem 3.** Pick any sequence of loss vectors $c_1, c_2, \ldots, c_T \in \mathbb{R}_+^K$. The expected total loss incurred by a learner using Algorithm 6 with parameter $\eta > 0$ is bounded as

$$\mathbb{E} \left( \sum_{t=1}^{T} c_t(a_t) \right) \leq \sum_{t=1}^{T} c_t(a) + \frac{\ln(K)}{\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \sum_{a'=1}^{K} c_t(a')^2$$

for any $a \in [K]$.

**Proof.** Below we use the following facts about the estimated loss vectors from (8) and (9):

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

$$\mathbb{E}(\hat{c}_t(a)^2 | p_t) = \frac{c_t(a)^2}{p_t(a)}.$$ 

Above, the second expectation is conditioned on $p_t$; note that $p_t$ is itself a random quantity depending on the randomness in the first $t-1$ rounds.
Observe that the evolution of the weights $w_t(a)$ in Algorithm 6 is the same as that of Algorithm 4 using the estimated loss vectors $\hat{c}_t \in \mathbb{R}^K_+$. Therefore, by Theorem 2, we have for any $a \in [K]$,

$$\sum_{t=1}^{T} \langle \hat{c}_t, p_t \rangle \leq \sum_{t=1}^{T} \hat{c}_t(a) + \frac{\ln(K)}{\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \langle \hat{c}_t^2, p_t \rangle$$  \hspace{1cm} (10)

(above, the vectors $p_t$ and $\hat{c}_t$ play the roles of the $x_t$ and $\ell_t$ used in Algorithm 4). Now we take expectation of both sides with respect to the random choices of Algorithm 6. First, observe that each term on the left-hand side of (10) has conditional expectation

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

Next, each term in the first summation on the right-hand side of (10) has expectation

$$\mathbb{E}(\hat{c}_t(a)) = c_t(a).$$

Finally, each term in the second summation on the right-hand side of (10) has conditional expectation

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

Taking more expectations and plugging back into (10) proves the claim. \qed

2.3 Hedging over experts with bandit feedback

Now we bring experts back into the picture.

**Algorithm 7** Protocol for online decision-making with bandit feedback and expert advice

1. for $t = 1$ to $T$ do
2. Receive experts’ recommendations: $b_{t,i} \in [K]$ for $i \in [N]$.
3. Choose action: $a_t \in [K]$.
4. Receive feedback: loss of chosen action $c_t(a_t) \in \mathbb{R}_+$.
5. end for
We will consider the total loss of any single expert \( i \in [N] \) in all \( T \) rounds, 
\[
L_{T,i} := \sum_{t=1}^{T} c_t(b_{t,i}),
\]
and also the total loss of the learner \( L_T := \sum_{t=1}^{T} c_t(a_t) \).
We are interested learning algorithms that guarantee low expected regret to the best expert:
\[
\mathbb{E}(R_T) := \mathbb{E}(L_T) - \min_{i \in [N]} L_{T,i}.
\]
Note that this is a very different notion of regret than what was considered in Theorem 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 loss 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 machine learning applications.

The main idea of the following algorithm is the same as that of Algorithm 6. We run a full-information algorithm with estimated loss vectors for the experts, rather than the actions. In this case, we need to estimate the loss incurred by an expert \( i \) in round \( t \): since expert \( i \) recommends action \( b_{t,i} \in [K] \) in round \( t \), the estimate we use is
\[
\hat{\ell}_{t,i} := \hat{c}_t(b_{t,i}) = \frac{1\{a_t = b_{t,i}\}}{p_t(a)} c_t(a_t)
\]
just as in (7). The estimated loss vector for experts \( \hat{\ell}_t \in \mathbb{R}^N_+ \) can then be used to define weights over experts, which in turn are used to define a probability distribution over actions. The resulting algorithm, Algorithm 6 is called “Exp4” (short for “Exponential-weight algorithm for Exploration and Exploitation using Expert advice”).
Algorithm 8 Exp4

input $\eta > 0$.
1: Let $w_{1,i} := 1$ for $i \in [N]$.
2: for $t = 1$ to $T$ do
3: Receive experts’ recommendations: $b_{t,i} \in [K]$ for $i \in [N]$.
4: Choose action: randomly draw $a_t \sim p_t \in \Delta^{K-1}$, where $p_t(a) := \sum_{i=1}^N w_{t,i} 1\{b_{t,i} = a\}/Z_t$ for each $a \in [K]$, and $Z_t := \sum_{i=1}^N w_{t,i}$.
5: Receive feedback: action $a_t$ incurs loss $c_t(a_t) \in \mathbb{R}^+$. 
6: Update: for each $i \in [N]$,

$$w_{t+1,i} := w_{t,i} \exp(-\eta \hat{\ell}_{t,i})$$

$$= \begin{cases} w_{t,i} & \text{if } b_{t,i} \neq a_t; \\ w_{t,i} \exp(-\eta c_t(a_t)/p_t(a_t)) & \text{if } b_{t,i} = a_t. \end{cases}$$

7: end for

Theorem 4. Pick any sequence of loss vectors $c_1, c_2, \ldots, c_T \in \mathbb{R}^K_+$. The expected total loss incurred by a learner using Algorithm 8 with parameter $\eta > 0$ is bounded as

$$\mathbb{E}\left(\sum_{t=1}^T \ell_t(a_t)\right) \leq \sum_{t=1}^T \ell_{t,i} + \frac{\ln(N)}{\eta} + \frac{\eta}{2} \sum_{t=1}^T \sum_{a=1}^K \ell_t(a)^2$$

for any $i \in [N]$.

Proof. The proof is very similar to that of Theorem 3. Let $x_{t,i} := w_{t,i}/Z_t$ for each $i \in [N]$. Choosing a random action $a_t \sim p_t$ is the same as choosing a random expert $i_t \sim x_t := (x_{t,1}, x_{t,2}, \ldots, x_{t,N}) \in \Delta^{N-1}$ and then setting $a_t := b_{t,i_t}$. The evolution of the weights $w_{t,i}$ in Algorithm 8 is the same as that of Algorithm 4 using the estimated loss vectors $\hat{\ell}_t \in \mathbb{R}^N$ for experts. Therefore, by Theorem 2 we have for any $i \in [N]$,

$$\sum_{t=1}^T \langle \hat{\ell}_t, x_t \rangle \leq \sum_{t=1}^T \hat{\ell}_{t,i} + \frac{\ln(N)}{\eta} + \frac{\eta}{2} \sum_{t=1}^T \langle \hat{\ell}_t^2, x_t \rangle. \quad (11)$$

Again, we take expectations of both sides. The following calculations are
essentially the same as in the proof of Theorem 3:

\[
\mathbb{E}(\langle \hat{\ell}_t, x_t \rangle | x_t) = \sum_{i=1}^{N} x_{t,i} c_t(b_{t,i}) = \sum_{a=1}^{K} \sum_{i=1}^{N} x_{t,i} \mathbb{1}\{b_{t,i} = a\} c_t(a) = \mathbb{E}(c_t(a_t) | x_t);
\]

\[
\mathbb{E}(\hat{\ell}_{t,i}) = c_t(b_{t,i});
\]

\[
\mathbb{E}(\langle \hat{\ell}_{t,i}^2, x_t \rangle | x_t) = \sum_{i=1}^{N} x_{t,i} \frac{c_t(b_{t,i})^2}{p_t(b_{t,i})} = \sum_{a=1}^{K} \sum_{i=1}^{N} x_{t,i} \frac{\mathbb{1}\{b_{t,i} = a\} c_t(a)^2}{p_t(a)} = \sum_{a=1}^{K} c_t(a)^2.
\]

Taking more expectations and plugging back into (11) proves the claim.

Again, if all of the (true) losses are in the range \([0, 1]\), then using \(\eta := \sqrt{2 \ln(N)/(KT)}\) guarantees the sublinear expected regret bound (to the best expert)

\[
\mathbb{E}(R_T) \leq \sqrt{2KT \ln(N)}.
\]

### 3 Matrix games and the minimax theorem

Let \(M \in [0, 1]^{m \times n}\) be a payoff matrix for a game between two players, the “row player” and the “column player”. If the row player picks row \(i \in [m]\), and simultaneously, the column player picks row \(j \in [n]\), then the row player suffers a loss of \(M_{i,j}\). See Table 1 for a simple example.

<table>
<thead>
<tr>
<th></th>
<th>kick to keeper’s left</th>
<th>kick to keeper’s right</th>
</tr>
</thead>
<tbody>
<tr>
<td>dive left</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>dive right</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1: Simple penalty kick game between the row player (keeper) and the column player (kicker). A more accurate payoff matrix would allow other moves (e.g., stay put, kick straight) and also account for the probability of a goal given the moves of the keeper and kicker.

We allow the players to use mixed strategies: that is, they can choose probability distributions \(p \in \Delta^{m-1}\) and \(q \in \Delta^{n-1}\) over their possible actions \([m]\) and \([n]\). After the strategies \(p\) and \(q\) are fixed, the row player randomly draws \(I \sim p\), and simultaneously, the column players randomly draws \(J \sim q\); the expected loss suffered by the row player is

\[
\mathbb{E} M_{t,j} = p^\top M q.
\]
Suppose the column player’s objective is to maximize the loss of the row player (i.e., the game is zero-sum). If the row player uses a strategy \( p \), then the column player will use the strategy \( q \) that maximizes \( p^\top Mq \). (The column players gets to see the row player’s strategy before choosing a strategy.) In this case, the expected loss suffered by the row player who uses \( p \) is

\[
\max_{q \in \Delta^{n-1}} p^\top Mq.
\]

The row player should therefore use a minimax strategy \( p_\star \), for which

\[
\max_{q \in \Delta^{n-1}} p_\star^\top Mq = \min_{p \in \Delta^{m-1}} \max_{q \in \Delta^{n-1}} p^\top Mq =: v_\star.
\]

This \( \min \max \) value \( v_\star \) is called the value of the game. A remarkable theorem attributed to von Neumann states that the order of the \( \min \) and \( \max \) can be swapped without changing the value: that is,

\[
\max_{p \in \Delta^{m-1}} \min_{q \in \Delta^{n-1}} p^\top Mq = \min_{q \in \Delta^{n-1}} \max_{p \in \Delta^{m-1}} p^\top Mq.
\]

We’ll see a proof of this shortly.

### 3.1 Repeated games

Suppose the same game with payoff matrix \( M \) is played \( T \) times. In each round \( t \), the row and column players pick mixed strategies \( p_t \) and \( q_t \), and actions \( I_t \) and \( J_t \) are drawn just as in the previous setup. At the end of round \( t \), the row player observes the expected losses of all actions \( i \in [m] \) (i.e., the vector \( Mq_t \)). The goal of the row player is to minimize the expected loss, averaged over the \( T \) rounds:

\[
\frac{1}{T} \sum_{t=1}^{T} \mathbb{E} M_{I_t,J_t} = \frac{1}{T} \sum_{t=1}^{T} p_t^\top M q_t.
\]

**Algorithm 9** Protocol for a repeated game with payoff matrix \( M 

1: for \( t = 1 \) to \( T \) do
2:  Row player picks \( p_t \in \Delta^{m-1} \).
3:  Column player picks \( q_t \in \Delta^{n-1} \).
4:  Simultaneously, draw \( I_t \sim p_t \) and \( J_t \sim q_t \), so loss of row player is \( M_{I_t,J_t} \), and the expected loss is \( p_t^\top M q_t \).
5:  Row player observes \( Mq_t \).
6: end for

18
This protocol is the exactly the linear loss game on the simplex with \( \ell_t := M q_t \). Therefore, if the row player uses Algorithm 4 (with parameter \( \eta = \sqrt{2 \ln(m)/T} \)) to choose the strategies \( p_t \) (i.e., the \( x_t \) in Algorithm 4) in every round, the expected loss, averaged over all \( T \) rounds, is

\[
\frac{1}{T} \sum_{t=1}^{T} p_t^\top M q_t \leq \min_{p \in \Delta^{m-1}} \frac{1}{T} \sum_{t=1}^{T} p^\top M q_t + \sqrt{\frac{2 \ln(m)}{T}}. \tag{12}
\]

In particular,

\[
\frac{1}{T} \sum_{t=1}^{T} p_t^\top M q_t \leq \frac{1}{T} \sum_{t=1}^{T} p_*^\top M q_t + \sqrt{\frac{2 \ln(m)}{T}}
\]

where \( p_* \) is a minimax strategy for the row player. Moreover, since \( p_*^\top M q_t \leq \max_{q \in \Delta^{n-1}} p_*^\top M q = v_* \), it follows that

\[
\frac{1}{T} \sum_{t=1}^{T} p_t^\top M q_t \leq v_* + \sqrt{\frac{2 \ln(m)}{T}}.
\]

This says that the row player never does much worse (on average, over many rounds \( T \)) that the (one-shot) minimax value of the game.

### 3.2 A proof of von Neumann’s minimax theorem

We’ll just prove the following lemma.

**Lemma 2** ((Half of) von Neumann’s minimax theorem).

\[
\min_{p \in \Delta^{m-1}} \max_{q \in \Delta^{n-1}} p^\top M q \leq \max_{q \in \Delta^{n-1}} \min_{p \in \Delta^{m-1}} p^\top M q.
\]

**Proof.** Consider the repeated game as described in the previous section. Suppose the column player always chooses \( q_t \) in response to the row player so that

\[
p_t^\top M q_t = \max_{q \in \Delta^{n-1}} p_t^\top M q.
\]

Furthermore, assume that the row player uses Algorithm 4 to choose the \( p_t \).
Let \( \bar{p} := (1/T) \sum_{t=1}^{T} p_t \) and \( \bar{q} := (1/T) \sum_{t=1}^{T} q_t \). Then
\[
\min_{p \in \Delta^{m-1}} \max_{q \in \Delta^{n-1}} p^\top Mq \leq \max_{q \in \Delta^{n-1}} \bar{p}^\top Mq
\]
\[
= \max_{q \in \Delta^{n-1}} \frac{1}{T} \sum_{t=1}^{T} p_t^\top Mq
\]
\[
\leq \frac{1}{T} \sum_{t=1}^{T} \max_{q \in \Delta^{n-1}} p_t^\top Mq
\]
\[
= \frac{1}{T} \sum_{t=1}^{T} p_t^\top Mq_t
\]
\[
\leq \min_{p \in \Delta^{m-1}} \frac{1}{T} \sum_{t=1}^{T} p_t^\top Mq_t + \sqrt{\frac{2 \ln(m)}{T}}
\]
\[
= \min_{p \in \Delta^{m-1}} p^\top Mq + \sqrt{\frac{2 \ln(m)}{T}}
\]
\[
\leq \max_{q \in \Delta^{n-1}} \min_{p \in \Delta^{m-1}} p^\top Mq + \sqrt{\frac{2 \ln(m)}{T}}.
\]

The third inequality follows from (12). Now taking \( T \to \infty \) gives
\[
\min_{p \in \Delta^{m-1}} \max_{q \in \Delta^{n-1}} p^\top Mq \leq \max_{q \in \Delta^{n-1}} \min_{p \in \Delta^{m-1}} p^\top Mq.
\]

Bibliographic notes

The lower bound in Theorem 1 is due to Cover (1965). The Randomized Weighted Majority algorithm is due to Littlestone and Warmuth (1994). The Hedge algorithm is due to Freund and Schapire (1997), and their original analysis gives the bound in (3). The analysis based on relative entropy is from Freund and Schapire (1999), as are the application to repeated games and the proof of von Neumann’s minimax theorem. The importance weighting trick is attributed to Horvitz and Thompson (1952). The Exp3 and Exp4 algorithms are due to Auer et al. (2002); here, we have only presented somewhat simplified versions of them for sake of easier analysis.

References


