Chapter 7

Online Learning

So far in the book has considered *batch* learning. Batch learning starts with some training data, analyzes it, and then "ships the result of the analysis into the world" (see Figure 7.1). "The result" can be a fixed classifier h, a distribution over classifiers ρ, or anything else, the important point is that it does not change from the moment the selection procedure is over. It takes no new information into account. This is also the reason why we had to assume that new samples come from the same distribution as the samples in the training set, because the classifier was not designed to adapt.

Online learning is a learning framework, where data collection, analysis, and application of inferred knowledge are in a perpetual loop, see Figure 7.1. Examples of problems, which fit into this framework include:

- Investment in the stock market.
- Online advertizing and personalization.
- Online routing.
- Games.
- Robotics.
- And so on ...

The recurrent nature of online learning problems makes them closely related to repeated games. They also borrow some of the terminology from the game theory, including calling the problems *games* and every "Act - Observe - Analyze" cycle a *game round*. In general, we may need online learning in the following scenarios:

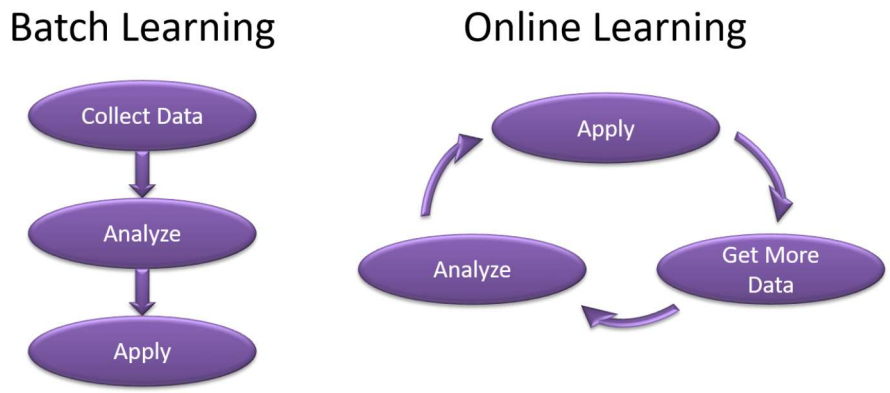

Figure 7.1: **Online learning vs. batch.**

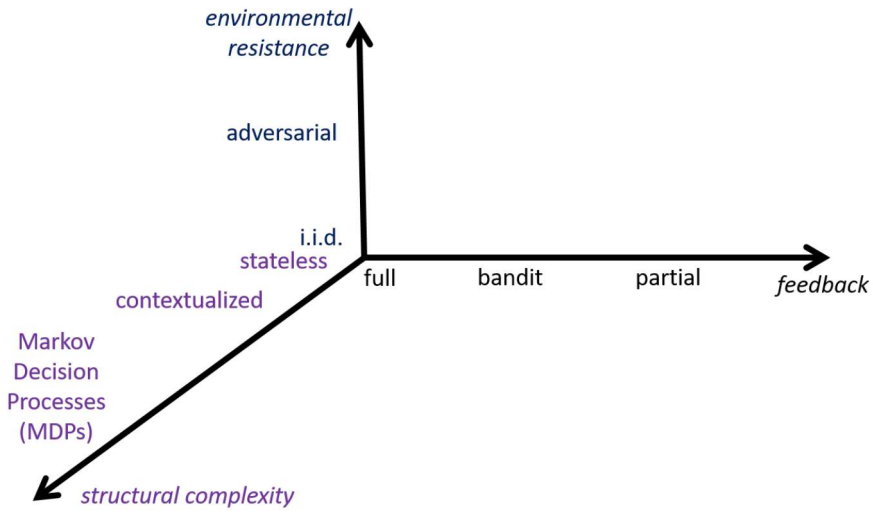

Figure 7.2: **The Space of Online Learning Problems.**

- Interactive learning: situations, where an algorithm continuously gets new information and taking it into account may improve the quality of future actions. For example, interaction with Internet users falls under this category – it makes sense to adapt to user behavior as the algorithm proceeds from one user query to the next.

- Adversarial or game-theoretic settings, where we cannot assume that "the future behaves similarly to the past". For example, in spam filtering we cannot assume that new spam messages are generated from the same distribution as the old ones. Or, in playing chess we cannot assume that the moves of the opponent are sampled i.i.d..

- Intelligent data collection. To cite Thompson (1933), "*there can be no objection to the use of data, however meagre, as a guide to action required before more can be collected*". Thompson was one of the pioneers of online learning and the theory of adaptive medical trials. In the context of adaptive medical trials the message is that it makes sense to look into the outcome of completed treatments before deciding on further treatments, as opposed to the more classical approach of A/B testing, where the size of treatment and control groups are decided upon before experimentation begins.[1]

As with many other problems in computer science, having loops (as in Figure 7.1) makes things much more challenging, but also much richer and more fun to work on. For example, online learning allows to treat adversarial environments, which is impossible to do in the batch setting.

7.1 The Space of Online Learning Problems

Online learning problems are characterized by three major parameters:

1. The amount of *feedback* that the algorithm receives in every round of interaction with the environment.

2. The *environmental resistance* to the algorithm.

[1] My other favourite quote on the topic is by Robbins (1952): "Until recently, statistical theory has been restricted to the design and analysis of sampling experiments in which the size and composition of the samples are completely determined before the experimentation begins. The reasons for this are partly historical, dating back to the time when the statistician was consulted, if at all, only after the experiment was over, and partly intrinsic in the mathematical difficulty of working with anything but a fixed number of independent random variables. A major advance now appears to be in the making with the creation of a theory of the *sequential design* of experiments, in which the size and composition of the samples are not fixed in advance but are functions of the observations themselves."

3. The *structural complexity* of a problem.

Jointly they define *the space of online learning problems*, see Figure 7.2. It is not really a space, but a convenient way to organize the material and get initial orientation in the zoo of online learning settings. We discuss the three axes of the space with some examples below.

Feedback

Feedback refers to the amount of information that the algorithm receives in every round of interaction with the environment. The most basic forms of feedback are *full information* and *limited* (better known as *bandit*[2]) feedback.

A classical example of a full information game is investment in the stock market. In every round of this game an algorithm distributes wealth over a set of stocks and the next day it observes the rates of all the stocks, which constitutes full information. With full information the algorithm can evaluate the quality of its own investment strategy, as well as any alternative investment strategy.

A classical example of a bandit feedback game are medical treatments. An algorithm has a set of *actions* (in this case treatments), but it can only apply one treatment to a given patient. The algorithm observes the outcome of the applied treatment, but not of the other treatments, resulting in limited feedback. With limited feedback the algorithm only observes the quality of the selected strategy, but it gets no direct access to the quality of alternative strategies that could have been selected. Limited feedback leads to the *exploration-exploitation trade-off*, which is the trade-mark signature of online learning. The essence of the exploration-exploitation trade-off is that in order to estimate the quality of actions the algorithm has to try them out (to explore). If it explores too little, it risks missing some good actions and end up performing suboptimally. However, exploration has a cost, because trying out suboptimal actions for too long is also undesirable. The goal is to balance exploration (trying new actions) with exploitation, which is taking actions, which are currently believed to be the optimal ones. The "Act-Observe-Analyze" cycle comes into play here, because unlike in batch learning the training set is not given, but is built by the algorithm for itself: if it does not try an action, it gets no data from it.

There are many other problems that fall within the bandit feedback framework, with another popular example being online advertising. A simplistic way of modeling online advertising is by assuming that there is a pool of advertisements, but on every round of the game it is only allowed to show one advertisement to a user. Since the advertiser only observes feedback for the advertisement that has been presented, the problem can be formulated as an online learning problem with bandit feedback.

There are other feedback models, which we will only touch briefly. In the bandit feedback model the algorithm observes a noisy estimate of the quality of selected action, for example, whether an advertisement was clicked or not. In *partial* feedback model studied under *partial monitoring* the feedback has some relation to the action, but not necessarily its quality. For example, in dynamic pricing the seller only observes whether a proposed price was above or below the value of a product for a buyer, but not the value itself (the maximal price the buyer would be ready to pay for the product). Bandit feedback is a special case of partial feedback, where the observation is the value. Another example is *dueling bandit* feedback, where the feedback is a relative preference over a pair of items rather than the absolute value of the items. For example, an answer to the question "Do you prefer fish or chicken?" is an example of dueling bandit feedback. Dueling bandit feedback model is used in information retrieval systems, since humans are much better in providing relative preferences rather than absolute utility values.

Environmental Resistance

Environmental resistance is concerned with how much the environment resists to the algorithm. Two classical examples are i.i.d. (a.k.a. *stochastic*) and *adversarial* environments. An example of an i.i.d. environment is the weather. It has a high degree of uncertainty, but it does not play against the algorithm. Another example of an i.i.d. environment are outcomes of medical treatments. Here also there is uncertainty in the outcomes, but the patients are not playing against the algorithm. An example of an adversarial environment is spam filtering. Here the spammers are deliberately changing distribution of the spam messages in order to outplay the spam filtering algorithm. Another classical example of an

[2] "The name derives from an imagined slot machine .... (Ordinary slot machines with one arm are one-armed bandits, since in the long run they are as effective as human bandits in separating the victim from his money.)" (Lai and Robbins, 1985)

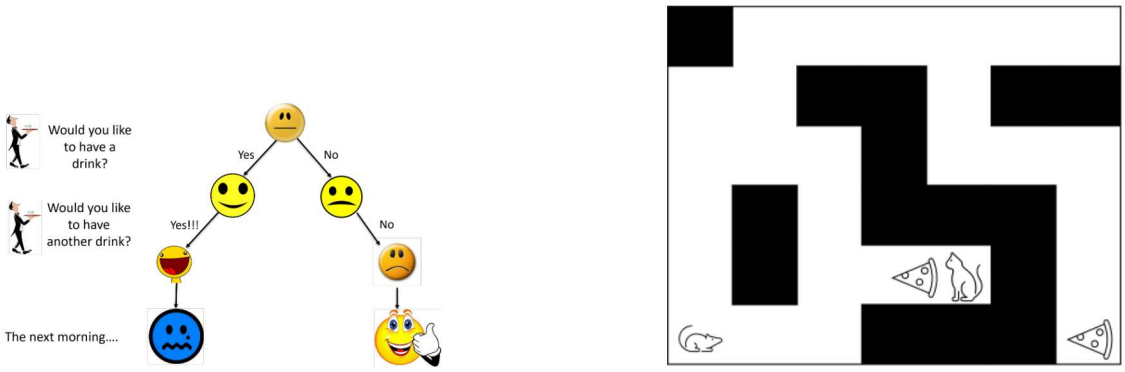

Figure 7.3: **Planning.** Even when the immediate outcomes are known, long-term goals require planning.

adversarial environment is the stock market. Even though the stock market does not play directly against an individual investor (assuming the investments are small), it is not stationary, because if there would be regularity in the market it would be exploited by other investors and would be gone.

The environment may also be collaborative, for example, when several agents are jointly solving a common task. Yet another example are slowly changing environments, where the parameters of a distribution are slowly changing with time.

Structural Complexity

In structural complexity we distinguish between *stateless* problems, *contextualized* problems (or problems with state), and *Markov decision processes*. In stateless problems actions are taken without taking any additional information except the history of the outcomes into account. In contextualized problems in every round of the game the algorithm observes a context (or state) and takes an action within the observed context. An example of a context is a medical record of a patient or, in the advertising example, it could be the parameters of the user that opened a web page.

Markov decision processes (MDPs) are concerned with processes with evolving state. The difference between contextualized problems and Markov decision processes is that in the former the actions of the algorithm do not influence the next state, whereas in the latter they do. For example, subsequent treatments of the same patient are changing his or her state and, therefore, depend on each other. In contrast, in subsequent treatments of different patients treatment of one patient does not influence the state of the next patient and, thus, can be modeled as a contextualized problem.

Markov decision processes are studied within the field of *reinforcement learning* (RL). There is no clear cut distinction between online learning and reinforcement learning, and one could be seen as a subfield of the other, or the other way around. But as a rule of thumb, problems involving evolution of states, such as Markov decision processes, are part of reinforcement learning, and problems that do not involve evolution of states are part of online learning.

When actions impact the state of the environment and the agent, they may have long-term consequences and, therefore, require *planning*. For example, getting to the "safe" slice of pizza in Figure 7.3 requires the mouse to plan a sequence of actions. Even if the immediate outcomes of every action in every state are known (there is no noise in execution of motor commands), there is still computational work to be done to infer the optimal action in each state. This is in contrast to online learning, where if the outcome of every action is known (e.g., the outcome of every treatment), inferring the best action is trivial. In many situations planning is combined with uncertainty estimation, for example, if the floor is slippery, the mouse might need to infer the relation between its actions and transitions between states. To summarize, online learning is primarily focused on uncertainty estimation, whereas reinforcement learning is focused on uncertainty estimation and planning, and the latter may be interesting and non-trivial even in absence of the former.

Online Learning	Reinforcement Learning
Uncertainty Estimation	Uncertainty Estimation
—	Planning

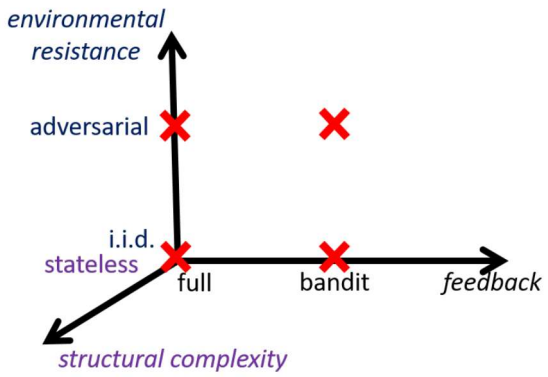

Figure 7.4: **The four basic online learning problems.**

There are many other online learning problems, which do not fit directly into Figure 7.2, but can still be discussed in terms of feedback, environmental resistance, and structural complexity. For example, in *combinatorial bandits* the goal is to select a set of actions, potentially with some constraints, and the quality of the set is evaluated jointly. An instance of a combinatorial bandit problem is selection of a path in a graph, such as communication or transport network. In this case an action can be decomposed into sub-actions corresponding to selection of edges in the graph. The goal is to minimize the length of a path, which may correspond to the delay between the source and the target nodes. Various forms of feedback can be considered, including bandit feedback, where the total length of the path is observed; semi-bandit feedback, where the length of each of the selected edges is observed; cascading bandit feedback, where the lengths of the edges are observed in a sequence until a terminating node (e.g., a server that is down) or the target is reached; or a full information feedback, where the length of all edges is observed.

Summary

To summarize, online and reinforcement learning introduce three new elements that we have not seen in batch learning: (1) incomplete feedback and *exploration* to deal with it, (2) the ability to handle *adversarial environments*, and (3) *planning*. And there is an infinite world of novel problem formulations that can be modeled in online and reinforcement learning.

In the following sections we consider in detail a number of the most basic online learning problems, and key tools for dealing with uncertainty and addressing the exploration-exploitation trade-off, and for handling i.i.d. and adversarial environments. (The topic of planning is left outside of the scope of the book.)

7.2 A General Basic Setup

We start with four most basic games in online learning, depicted by the red crosses in Figure 7.4: stateless i.i.d. full information, stateless i.i.d. adversarial, stateless i.i.d. bandit, and stateless adversarial bandit. The first setting is very easy and is studied in Exercise 7.2. The stateless i.i.d. adversarial setting is known as *prediction with expert advice*, and the two bandit settings are known as *stochastic multiarmed bandits* and *adversarial multiarmed bandits*, respectively. We first provide a general setup that encompasses all the four problems, and then specialize it to each of them.

We are given a $K \times \infty$ matrix of losses $\ell_{t,a}$, where $t \in \{1, 2, \dots\}$ and $a \in \{1, \dots, K\}$ and $\ell_{t,a} \in [0, 1]$.

$$\begin{array}{rcccccc} & \ell_{1,1}, & \ell_{2,1}, & \cdots & \ell_{t,1}, & \cdots \\ & \vdots & \vdots & \cdots & \vdots & \cdots \\ \textit{Losses} & \ell_{1,a}, & \ell_{2,a}, & \cdots & \ell_{t,a}, & \cdots \\ & \vdots & \vdots & \cdots & \vdots & \cdots \\ & \ell_{1,K}, & \ell_{2,K}, & \cdots & \ell_{t,K}, & \cdots \end{array}$$

$$\xrightarrow[\textit{time}]{}$$

The matrix is fixed before the game starts according to a protocol defined below, but not revealed to the algorithm. The game proceeds in the following way.

Game Protocol

For $t = 1, 2, \dots$:

1. Pick a row A_t
2. Suffer ℓ_{t,A_t}
3. Observe ... [the observations are defined below]

Definition of the four games There are two basic ways to generate the matrix of losses and two basic ways to define the observations, which jointly make up the four games, as summarized in the table below.

The first way to generate the matrix is to sample $\ell_{t,a}$-s independently, so that the mean of the losses in each row is fixed, $\mathbb{E}[\ell_{t,a}] = \mu(a)$. The second is to generate $\ell_{t,a}$-s arbitrarily. The second model of generation of losses is known as an *oblivious adversary*, since the generation happens before the game starts and does not take actions of the algorithm into account.[3]

The first way to define the observations is to reveal the full column $\ell_{t,1}, \dots, \ell_{t,K}$ after the algorithm has played the row A_t. The second is to reveal only the selected entry ℓ_{t,A_t}.

Jointly the two ways of generating the matrix of losses and the two ways of defining the observations define the four variants of the game.

Matrix generation \ Observations	Observe $\ell_{t,1}, \dots, \ell_{t,K}$	Observe ℓ_{t,A_t}
$\ell_{t,a}$-s are sampled i.i.d. with $\mathbb{E}[\ell_{t,a}] = \mu(a)$	I.I.D. Prediction with expert advice	Stochastic multiarmed bandits
$\ell_{t,a}$ are selected arbitrarily (by an adversary)	Prediction with expert advice (adversarial)	Adversarial multiarmed bandits

Performance Measure The goal of the algorithm is to play so that the loss it suffers will not be significantly larger than the loss of the best row in hindsight. There are several ways to formalize this goal. The basic performance measure is the *regret* defined by

$$R_T = \sum_{t=1}^{T} \ell_{t,A_t} - \min_a \sum_{t=1}^{T} \ell_{t,a}.$$

In adversarial problems we analyze the *expected regret*[4] defined by

$$\mathbb{E}[R_T] = \mathbb{E}\left[\sum_{t=1}^{T} \ell_{t,A_t}\right] - \mathbb{E}\left[\min_a \sum_{t=1}^{T} \ell_{t,a}\right].$$

[3] It is also possible to consider an *adaptive adversary*, which generates losses as the game proceeds and takes past actions of the algorithm into account. We do not discuss this model in the book.

[4] It is also possible to analyze the regret, but we do not do it here.

If the sequence of losses is deterministic, we can remove the second expectation and obtain

$$\mathbb{E}\left[R_T\right] = \mathbb{E}\left[\sum_{t=1}^{T} \ell_{t,A_t}\right] - \min_a \sum_{t=1}^{T} \ell_{t,a}.$$

In stochastic problems we analyze the *pseudo regret* defined by

$$\bar{R}_T = \mathbb{E}\left[\sum_{t=1}^{T} \ell_{t,A_t}\right] - \min_a \mathbb{E}\left[\sum_{t=1}^{T} \ell_{t,a}\right] = \mathbb{E}\left[\sum_{t=1}^{T} \ell_{t,A_t}\right] - T \min_a \mu(a).$$

Note that since for random variables X and Y we have $\mathbb{E}\left[\min\left\{X, Y\right\}\right] \leq \min\left\{\mathbb{E}\left[X\right], \mathbb{E}\left[Y\right]\right\}$ [it is recommended to verify this identity], we have $\bar{R}_T \leq \mathbb{E}\left[R_T\right]$.

The different notions of regret Let us briefly discuss the relations and differences between regret, expected regret, and pseudo-regret. First, we note that in the oblivious adversarial setting the losses are considered to be selected deterministically and, therefore, the expectation in the second term can be dropped, resulting in $\mathbb{E}\left[\min_a \sum_{t=1}^{T} \ell_{t,a}\right] = \min_a \mathbb{E}\left[\sum_{t=1}^{T} \ell_{t,a}\right] = \min_a \sum_{t=1}^{T} \ell_{t,a}$. Thus, in the oblivious adversarial setting the notions of expected regret and pseudo-regret coincide. (This is not true for the adaptive setting, but we do not delve into it here.) The difference between regret and expected/pseudo-regret in the adversarial setting is thus only in the first term – whether we want to obtain guarantees on the expected performance of an algorithm, $\mathbb{E}\left[\sum_{t=1}^{T} \ell_{t,A_t}\right]$, or individual roll-outs of its play, $\sum_{t=1}^{T} \ell_{t,A_t}$. Both are valid targets, we focus on the expected performance because it is a tiny bit easier.

In the stochastic setting $\mathbb{E}\left[\min_a \sum_{t=1}^{T} \ell_{t,a}\right] \leq \min_a \mathbb{E}\left[\sum_{t=1}^{T} \ell_{t,a}\right] = T \min_a \mu(a)$, and so the expected regret and the pseudo-regret are not the same. In pseudo-regret the performance baseline is the expected performance of a best action, $T \min_a \mu(a)$, defined by $\mu(a)$, whereas in expected regret it is the expected best roll-out of any action. Imagine that there are K actions and the loss of every action at every round is a Bernoulli random variable with bias $\frac{1}{2}$. Then $\mu(a) = \frac{1}{2}$ for all a and the pseudo-regret baseline is $\frac{1}{2}T$. And for any algorithm $\mathbb{E}\left[\sum_{t=1}^{T} \ell_{t,A_t}\right] = \frac{1}{2}T$, thus $\bar{R}_T = 0$. However, $\mathbb{E}\left[\min_a \sum_{t=1}^{T} \ell_{t,a}\right] \leq \frac{1}{2}T$, because the baseline (the competitor of the algorithm) is allowed to select the best out of K roll-outs of T Bernoulli random variables with bias $\frac{1}{2}$. In fact, $\mathbb{E}\left[\min_a \sum_{t=1}^{T} \ell_{t,a}\right] \approx \frac{1}{2}T - \sqrt{\frac{1}{2}T \ln K}$ (see Section 7.4.1), leading to $\mathbb{E}\left[R_T\right] \approx \sqrt{\frac{1}{2}T \ln K}$. Even though the loss of any algorithm cannot be smaller than $\frac{1}{2}T$ in expectation, and the loss of any fixed row is also $\frac{1}{2}T$ in expectation, the expected regret grows as $\sqrt{\frac{1}{2}T \ln K}$, because the competitor has the advantage of being allowed to make K tries of sampling T Bernoulli losses and selecting the best, whereas the algorithm gets just one try. Thus, comparing the performance of an algorithm to the best row in expectation rather than the expected best roll-out is considered more reasonable. We will be able to derive bounds on pseudo-regret in the stochastic setting that grow at the rate of $\ln T$, whereas the fluctuations of $\sum_{t=1}^{T} \ell_{t,a}$ and $\sum_{t=1}^{T} \ell_{t,A_t}$ can be as large as $\sqrt{T}$, and so the best possible bounds on the regret and the expected regret in the stochastic setting are of order $\sqrt{T}$.

Explanation of the Names In the complete definition of prediction with expert advice game, in every round of the game the player gets an advice from K experts and then takes an action, which may be a function of the advice. The player suffers a loss depending on the action taken, and the experts suffer losses depending on their advice. Hence the name, prediction with expert advice. If we restrict the actions of the player to following the advice of a single expert, then from the perspective of the playing strategy the actual advice does not matter and it is only the loss that defines the strategy. We consider the restricted setting, because it allows to highlight the relation to multiarmed bandits.

The name multiarmed bandits comes from the analogy with slot machines, which are one-armed bandits. In this game the actions are the "arms" of a slot machine.

Losses vs. Rewards In some games it is more natural to consider rewards (also called gains), rather than losses. In fact, in the literature on stochastic problems it is more popular to work with rewards, whereas in the literature on adversarial problems it is more popular to work with losses. There is a simple transformation $r = 1 - \ell$, which brings a losses game into a gains game and the other way around. Interestingly, in the adversarial setting working with losses leads to tighter and simpler results (see Exercise 7.13). In the stochastic setting the choice does not matter.

7.3 I.I.D. (stochastic) Multiarmed Bandits

In this section we consider a multiarmed bandit game, where the outcomes (the sequence of losses) are generated i.i.d. with fixed, but unknown means. In this game there is no difference between working with losses or rewards, and since most of the literature is based on games with rewards we are going to use rewards in order to be consistent. The treatment of losses is identical - see Seldin (2015).

Notations We are given a $K \times \infty$ matrix of rewards (or gains) $r_{t,a}$, where $t \in \{1, 2, \dots\}$ and $a \in \{1, \dots, K\}$.

$$
\begin{array}{cc}
\multirow{6}{*}{\textit{Action rewards}} & \begin{array}{ccccc}
r_{1,1}, & r_{2,1}, & \cdots & r_{t,1}, & \cdots \\
\vdots & \vdots & \cdots & \vdots & \cdots \\
r_{1,a}, & r_{2,a}, & \cdots & r_{t,a}, & \cdots \\
\vdots & \vdots & \cdots & \vdots & \cdots \\
r_{1,K}, & r_{2,K}, & \cdots & r_{t,K}, & \cdots
\end{array} \\
& \longrightarrow \\
& \textit{time}
\end{array}
$$

We assume that $r_{t,a}$-s are in $[0, 1]$ and that they are generated independently, so that $\mathbb{E}[r_{t,a}] = \mu(a)$. We use $\mu^* = \max_a \mu(a)$ to denote the expected reward of an optimal action and $\Delta(a) = \mu^* - \mu(a)$ to denote the *suboptimality gap* (or simply the *gap*) of action a. The suboptimality gap $\Delta(a)$ measures by how much, in expectation, playing action a is worse than playing the optimal action. We use $a^* \in \arg\max_a \mu(a)$ to denote *a best action* (note that there may be more than one best action, in such case we let a^* be any of them).

Game Definition

For $t = 1, 2, \dots$:

1. Pick a row A_t

2. Observe & accumulate r_{t,A_t}

Performance Measure Let $N_t(a)$ denote the number of times action a was played up to round t. We measure the performance using the pseudo regret and we rewrite it in the following way (note that since we are working with rewards, we subtract the expected reward of the algorithm from the expected

reward of a best arm, whereas for losses it was the other way around)

$$\begin{aligned}
\bar{R}_T &= \max_a \mathbb{E}\left[\sum_{t=1}^{T} r_{t,a}\right] - \mathbb{E}\left[\sum_{t=1}^{T} r_{t,A_t}\right] \\
&= T\mu^* - \mathbb{E}\left[\sum_{t=1}^{T} r_{t,A_t}\right] \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\mu^* - r_{t,A_t}\right] \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\mathbb{E}\left[\mu^* - r_{t,A_t} | A_t\right]\right] \qquad (7.1) \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\mu^* - \mu(A_t)\right] \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\Delta(A_t)\right] \\
&= \sum_a \Delta(a)\mathbb{E}\left[N_T(a)\right].
\end{aligned}$$

In step (7.1) we note that $\mathbb{E}[r_{t,A_t}]$ is an expectation over two random variables, the selection of A_t, which is based on the history of the game, and the draw of r_{t,A_t}, for which $\mathbb{E}[r_{t,A_t}|A_t] = \mu(A_t)$. We have $\mathbb{E}[r_{t,A_t}] = \mathbb{E}[\mathbb{E}[r_{t,A_t}|A_t]]$, where the inner expectation is with respect to the draw of r_{t,A_t} and the outer expectation is with respect to the draw of A_t.

Note that in the i.i.d. setting the reward of an algorithm is compared to the expected reward of the best action in expectation, $\max_a \mathbb{E}\left[\sum_{t=1}^{T} r_{t,a}\right] = T \max_a \mu(a)$, whereas in the adversarial setting the reward of an algorithm is compared to the reward of the best action in hindsight, $\max_a \sum_{t=1}^{T} r_{t,a}$.

Exploration-exploitation trade-off: A simple approach

I.I.D. multiarmed bandits is the simplest problem, where we face the exploration-exploitation trade-off. In general, the goal is to play a best arm in all the rounds, but since the identity of the best arm is unknown, it has to be identified first. In order to identify a best arm we need to explore all the arms. However, rounds used for exploration of suboptimal arms increase the regret, because every time we play a suboptmal arm a, we pay $\Delta(a)$ in the regret. The total regret is $\bar{R} = \sum_a \Delta(a)\mathbb{E}[N_T(a)]$, where $\mathbb{E}[N_T(a)]$ is the expected number of times a suboptimal action a was played. At the same time, saving too much on exploration may lead to confusion between a best and a suboptimal arm, which may eventually lead to even higher regret if we start exploiting a wrong arm.

So let us make a first attempt at quantifying this trade-off. Assume that we have just two actions, and we know the suboptimality gap Δ, so that $\mu(a) = \mu(a^*) - \Delta$, but we do not know which of the two actions is the better one, so we need to figure it out. Assume that we know the time horizon T we are going to play the game. A possible approach is to start with εT exploration rounds, where we play each of the two arms $\frac{1}{2}\varepsilon T$ times, followed by $(1-\varepsilon)T$ exploitation rounds, where we play the arm that yielded the highest reward by the end of the exploration phase. What should be the length of the exploration phase εT and what will be the pseudo-regret of this playing strategy?

Let $\delta(\varepsilon)$ denote the probability that we misidentify the best arm at the end of the exploration phase, namely, due to statistical fluctuations the suboptimal arm a happens to yield a higher reward than the optimal arm a^*. The pseudo regret can be bounded by:

$$\bar{R}_T \leq \underbrace{\frac{1}{2}\Delta\varepsilon T}_{\text{exploration}} + \underbrace{\delta(\varepsilon)\Delta(1-\varepsilon)T}_{\text{exploitation}} \leq \frac{1}{2}\Delta\varepsilon T + \delta(\varepsilon)\Delta T = \left(\frac{1}{2}\varepsilon + \delta(\varepsilon)\right)\Delta T,$$

where the first term is a bound on the pseudo regret during the exploration phase and the second term is a bound on the pseudo regret during the exploitation phase in case we select a wrong arm at the end of the exploration phase. Now what is $\delta(\varepsilon)$? Let $\hat{\mu}_t(a)$ denote the empirical mean of observed rewards of arm a up to round t. For the exploitation phase it is natural to select the arm that maximizes $\hat{\mu}_{\varepsilon T}(a)$ at the end of the exploration phase. Therefore:

$$\begin{aligned}\delta(\varepsilon) &= \mathbb{P}(\hat{\mu}_{\varepsilon T}(a) \geq \hat{\mu}_{\varepsilon T}(a^*)) \\ &\leq \mathbb{P}\left(\hat{\mu}_{\varepsilon T}(a) \geq \mu(a) + \frac{1}{2}\Delta\right) + \mathbb{P}\left(\hat{\mu}_{\varepsilon T}(a^*) \leq \mu^* - \frac{1}{2}\Delta\right) \\ &\leq 2e^{-2\frac{\varepsilon T}{2}\left(\frac{1}{2}\Delta\right)^2} = 2e^{-\varepsilon T \Delta^2/4},\end{aligned}$$

where the last line is by Hoeffding's inequality. By substituting this back into the regret bound we obtain:

$$\bar{R}_T \leq \left(\frac{1}{2}\varepsilon + 2e^{-\varepsilon T\Delta^2/4}\right)\Delta T.$$

In order to minimize $\frac{1}{2}\varepsilon + 2e^{-\varepsilon T\Delta^2/4}$ we take a derivative and equate it to zero, which leads to $\varepsilon = \frac{\ln(T\Delta^2)}{T\Delta^2/4}$. It is easy to check that the second derivative is positive, confirming that this is the minimum. Note that ε must be non-negative, so strictly speaking we have $\varepsilon = \max\left\{0, \frac{\ln(T\Delta^2)}{T\Delta^2/4}\right\}$. If we substitute this back into the regret bound we obtain:

$$\bar{R}_T \leq \min\left\{\Delta T, \left(\frac{2\ln(T\Delta^2)}{T\Delta^2} + 2e^{-\ln(T\Delta^2)}\right)\Delta T\right\} = \min\left\{\Delta T, \frac{2\ln(T\Delta^2)}{\Delta} + \frac{2}{\Delta}\right\}.$$

Note that the number of exploration rounds is $\varepsilon T = \max\left\{0, \frac{\ln(T\Delta^2)}{\Delta^2/4}\right\}$.

Pay attention that the regret bound has an inverse dependence on Δ, meaning that it gets *larger* as Δ gets *smaller*. Although intuitively when Δ is small we do not care that much about playing a suboptimal action as opposed to the case when Δ is large, problems with small Δ are actually harder and lead to larger regret. The reason is that the number of rounds that it takes to identify the best action (the number of exploration steps εT) grows with $1/\Delta^2$. Even though in each exploration round we only suffer a regret of Δ, the fact that the number of exploration rounds grows with $1/\Delta^2$ makes problems with small Δ harder and makes the regret grow at the rate of $1/\Delta$. However, note that if the time horizon T is very small in relation to $1/\Delta^2$, then there is not enough time to identify the best action, and the ΔT term dominates the minimum in the regret bound, see Exercise 7.4 for further details.

The above approach has three drawbacks: (1) it assumes knowledge of the time horizon T, (2) it assumes knowledge of the gap Δ, and (3) if we would generalize it to more than one arm, the length of the exploration phase would depend on the smallest gap, even if there are many arms with larger gap that are much easier to eliminate. The following approach resolves all the three problems.

The Upper Confidence Bound (UCB) algorithm

We present the UCB1 algorithm of Auer et al. (2002a).[5]

Algorithm 3 UCB1 (Auer et al., 2002a)

Initialization: Play each action once.
for $t = K+1, K+2, \ldots$ **do**
  Play $A_t = \arg\max_a \hat{\mu}_{t-1}(a) + \sqrt{\frac{3\ln t}{2N_{t-1}(a)}}$.
end for

The expression $U_t(a) = \hat{\mu}_{t-1}(a) + \sqrt{\frac{3\ln t}{2N_{t-1}(a)}}$ is called an *upper confidence bound*. Why? Because $U_t(a)$ upper bounds $\mu(a)$ with high probability. UCB approach follows the *optimism in the face of uncertainty principle*. That is, we take an optimistic estimate of the reward of every arm by taking the upper limit of the confidence bound. UCB1 algorithm has the following regret guarantee.

[5]See Exercise 7.5 for an improved parametrization and analysis.

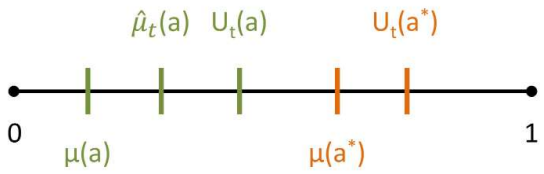

Figure 7.5: Illustration for UCB analysis.

Theorem 7.1. *For any time T the regret of UCB1 satisfies:*

$$\bar{R}_T \leq 6 \sum_{a:\Delta(a)>0} \frac{\ln T}{\Delta(a)} + \left(1 + \frac{\pi^2}{3}\right) \sum_a \Delta(a).$$

Proof. For the analysis it is convenient to have the following picture in mind – see Figure 7.5. A suboptimal arm may be played if $U_t(a) \geq U_t(a^*)$. Our goal is to show that this does not happen too often. The analysis is based on the following three points, which bound the corresponding distances in Figure 7.5.

1. We show that $U_t(a^*) > \mu(a^*)$ for almost all rounds. A bit more precisely, let $F(a^*)$ be the number of rounds when $U_t(a^*) \leq \mu(a^*)$, then $\mathbb{E}[F(a^*)] \leq \frac{\pi^2}{6}$.

2. In a similar way, we show that $\hat{\mu}_t(a) < \mu(a) + \sqrt{\frac{3 \ln t}{2N_t(a)}}$ for almost all rounds. A bit more precisely, let $F(a)$ be the number of rounds when $\hat{\mu}_t(a) \geq \mu(a) + \sqrt{\frac{3 \ln t}{2N_t(a)}}$, then $\mathbb{E}[F(a)] \leq \frac{\pi^2}{6}$. (Note that $L_t(a) = \hat{\mu}_t(a) - \sqrt{\frac{3 \ln t}{2N_t(a)}}$ is a lower confidence bound for $\mu(a)$, meaning that with high probability $L_t(a) < \mu(a)$.)

3. When Point 2 holds we have that $U_t(a) = \hat{\mu}_{t-1}(a) + \sqrt{\frac{3 \ln t}{2N_{t-1}(a)}} \leq \mu(a) + 2\sqrt{\frac{3 \ln t}{2N_{t-1}(a)}} = \mu(a^*) - \Delta(a) + 2\sqrt{\frac{3 \ln t}{2N_{t-1}(a)}}$.

Let us fix time horizon T and analyze what happens by time T (note that the algorithm does not depend on T). We have that for most rounds $t \leq T$:

$$U_t(a) < \mu(a^*) - \Delta(a) + \sqrt{\frac{6 \ln t}{N_{t-1}(a)}} \leq \mu(a^*) - \Delta(a) + \sqrt{\frac{6 \ln T}{N_{t-1}(a)}},$$
$$U_t(a^*) > \mu(a^*).$$

Thus, we can play a suboptimal action a only in the following cases:

- Either $\sqrt{\frac{6 \ln T}{N_{t-1}(a)}} \geq \Delta(a)$, which means that $N_{t-1}(a) \leq \frac{6 \ln T}{\Delta(a)^2}$. (As long as a suboptimal arm has not been played $\frac{6 \ln T}{\Delta(a)^2}$ times, its confidence interval is not tight enough to reliably disambiguate it from the best arm.)
- Or one of the confidence intervals in Points 1 or 2 has failed.

In other words, after a suboptimal action a has been played for $\left\lceil \frac{6 \ln T}{\Delta(a)^2} \right\rceil$ rounds, it can only be played again only if one of the confidence intervals fails. Therefore,

$$\mathbb{E}[N_T(a)] \leq \left\lceil \frac{6 \ln T}{\Delta(a)^2} \right\rceil + \mathbb{E}[F(a^*)] + \mathbb{E}[F(a)] \leq \frac{6 \ln T}{\Delta(a)^2} + 1 + \frac{\pi^2}{3}$$

and since $\bar{R}_T(a) = \sum_a \Delta(a) \mathbb{E}[N_T(a)]$ the result follows.

To complete the proof it is left to prove Points 1 and 2. We prove Point 1, the proof of Point 2 is identical. We start by looking at

$$\mathbb{P}(U_t(a^*) \le \mu(a^*)) = \mathbb{P}\left(\hat{\mu}_{t-1}(a^*) + \sqrt{\frac{3\ln t}{2N_{t-1}(a^*)}} \le \mu(a^*)\right) = \mathbb{P}\left(\mu(a^*) - \hat{\mu}_{t-1}(a^*) \ge \sqrt{\frac{3\ln t}{2N_{t-1}(a^*)}}\right).$$

The delicate point is that $N_{t-1}(a^*)$ is a random variable dependent on $\hat{\mu}_{t-1}(a^*)$, and thus we cannot apply Hoeffding's inequality directly. Instead, we introduce a series of random variables $X_1, X_2, \dots$, such that X_i-s have the same distribution as r_{t,a^*}-s. Let $\bar{\mu}_s = \frac{1}{s}\sum_{i=1}^{s} X_i$ be the average of the first s elements of the sequence. Then we have:

$$\begin{aligned}
\mathbb{P}\left(\mu(a^*) - \hat{\mu}_{t-1}(a^*) \ge \sqrt{\frac{3\ln t}{2N_{t-1}(a^*)}}\right) &\le \mathbb{P}\left(\exists s \in \{1,\dots,t\} : \mu(a^*) - \bar{\mu}_s \ge \sqrt{\frac{3\ln t}{2s}}\right) \\
&\le \sum_{s=1}^{t} \mathbb{P}\left(\mu(a^*) - \bar{\mu}_s \ge \sqrt{\frac{3\ln t}{2s}}\right) \\
&\le \sum_{s=1}^{t} \frac{1}{t^3} = \frac{1}{t^2},
\end{aligned}$$

where in the first line we decouple $\hat{\mu}_t(a^*)$-s from $N_t(a^*)$-s via the use of $\bar{\mu}_s$-s, and in the last line we apply Hoeffding's inequality (Theorem 3.5). Note that $3\ln t = \ln t^3$ corresponds to $\ln\frac{1}{\delta}$ in Hoeffding's inequality, and thus $\delta = \frac{1}{t^3}$. Finally, we have:

$$\mathbb{E}\left[F(a^*)\right] = \sum_{t=1}^{\infty} \mathbb{P}\left(\mu(a^*) - \hat{\mu}_{t-1}(a^*) \ge \sqrt{\frac{3\ln t}{2N_{t-1}(a^*)}}\right) \le \sum_{t=1}^{\infty} \frac{1}{t^2} = \frac{\pi^2}{6}.$$

□

We refer the reader to Exercise 7.5 for an improved parametrization and a tighter regret bound for the UCB1 algorithm.

7.4 Prediction with Expert Advice

Now we turn our attention to stateless adversarial game with full information feedback.

Notations We are given a $K \times \infty$ matrix of expert losses $\ell_{t,a}$, where $t \in \{1, 2, \dots\}$ and $a \in \{1, \dots, K\}$.

$$\begin{array}{c}
\\
\textit{Expert Losses} \\
\\
\\
\\
\\
\end{array}
\begin{array}{cccccc}
\ell_{1,1}, & \ell_{2,1}, & \cdots & \ell_{t,1}, & \cdots \\
\vdots & \vdots & \cdots & \vdots & \cdots \\
\ell_{1,a}, & \ell_{2,a}, & \cdots & \ell_{t,a}, & \cdots \\
\vdots & \vdots & \cdots & \vdots & \cdots \\
\ell_{1,K}, & \ell_{2,K}, & \cdots & \ell_{t,K}, & \cdots \\
\multicolumn{5}{c}{\longrightarrow} \\
\multicolumn{5}{c}{\textit{time}}
\end{array}$$

Game Definition

For $t = 1, 2, \dots$:

1. Pick a row A_t
2. Observe the column $\ell_{t,1}, \dots, \ell_{t,K}$ & suffer ℓ_{t,A_t}