LeanBandits
A Lean package for bandit algorithms

Rémy Degenne

July 21, 2025

Chapter 1

Introduction

A bandit algorithm sequentially chooses actions and then observes rewards, whose distribution depends on the action chosen. The algorithm does not know the distribution of the rewards and sees only a reward from the chosen action at any given time. A key part of the interaction is that the algorithm can choose the next action based on all the previous actions and rewards. The goal of the algorithm is typically to maximize the cumulative reward over time. The researcher studying bandit algorithms is interested in the performance of the algorithm, which is measured by the "regret" R_T after choosing T actions, that is the difference between the cumulative reward of always playing the best action and the cumulative reward of the algorithm. A theoretical guarantee will be of the form $\mathbb{E}[R_T] \leq f(T)$ for some function f. Here the expectation is taken over the randomness of the algorithm and the rewards. In parallel to the theoretical study, the researcher may also be interested in the practical performance of the algorithm, which is usually measured by the average regret over several runs of the algorithm with rewards sampled from standard probability distributions.

From that description, we highlight three key components of research work on bandit algorithms:

1. A bandit algorithm is both a subject of theoretical study and a practical tool, that we should be able to implement and run,

2. the bandit model defines a probability space, on which we want to take expectations, and the theoretical study deals with random variables on that space using tools like concentration inequalities,

3. for the experimental part, we need to be able to sample rewards from a range of probability distributions.

Chapter 2

Stochastic multi-armed bandits

2.1 Bandit model and probability space

Definition 1 (Bandit)**.** The interaction of an algorithm with a stochastic bandit with arms in $\mathcal{A}$ (a measurable space) and real rewards is described by the following data:

- $\nu : \mathcal{A} \rightsquigarrow \mathbb{R}$, a Markov kernel, conditional distribution of the rewards given the arm pulled,
- for all $t \in \mathbb{N}$, a policy $\pi_t : (\mathcal{A} \times \mathbb{R})^{t+1} \rightsquigarrow \mathcal{A}$, a Markov kernel which gives the distribution of the arm to pull at time $t+1$ given the history of previous pulls and rewards,
- $P_0 \in \mathcal{P}(\mathcal{A})$, a probability measure that gives the distribution of the first arm to pull.

Definition 2 (Bandit probability space)**.** By an application of the Ionescu-Tulcea theorem, a bandit $\mathcal{B} = (\nu, \pi, P_0)$ defines a probability distribution on the space $\Omega := (\mathcal{A} \times \mathbb{R})^{\mathbb{N}}$, the space of infinite sequences of arms and rewards. We denote that distribution by $\mathbb{P}$. TODO: explain how the probability distribution is constructed.

Definition 3 (Arms, rewards and history)**.** For $t \in \mathbb{N}$, we denote by A_t the arm pulled at time t and by X_t the reward received at time t. Formally, these are measurable functions on $\Omega = (\mathcal{A} \times \mathbb{R})^{\mathbb{N}}$, defined by $A_t(\omega) = \omega_{t,1}$ and $X_t(\omega) = \omega_{t,2}$. We denote by $H_t \in (\mathcal{A} \times \mathbb{R})^{t+1}$ the history of pulls and rewards up to and including time t, that is $H_t = ((A_0, X_0), \dots, (A_t, X_t))$. Formally, $H_t(\omega) = (\omega_0, \dots, \omega_t)$.

Lemma 4. *The conditional distribution of the reward X_t given the arm A_t in the bandit probability space $(\Omega, \mathbb{P})$ is $\nu(A_t)$.*

Proof. □

Lemma 5. *The law of the arm A_0 in the bandit probability space $(\Omega, \mathbb{P})$ is P_0.*

Proof. □

Lemma 6. *The conditional distribution of the arm A_{t+1} given the history H_t in the bandit probability space $(\Omega, \mathbb{P})$ is $\pi_t(H_t)$.*

Proof. □

2.2 Regret and other bandit quantities

Definition 7. For an arm $a \in \mathcal{A}$, we denote by μ_a the mean of the rewards for that arm, that is $\mu_a = \nu(a)[X]$. We denote by μ^* the mean of the best arm, that is $\mu^* = \max_{a \in \mathcal{A}} \mu_a$.

Definition 8 (Regret)**.** The regret R_T of a sequence of arms $A_0, \dots, A_{T-1}$ after T pulls is the difference between the cumulative reward of always playing the best arm and the cumulative reward of the sequence:

$$R_T = T\mu^* - \sum_{t=0}^{T-1} \mu_{A_t} .$$

Definition 9. For an arm $a \in \mathcal{A}$, its gap is defined as the difference between the mean of the best arm and the mean of that arm: $\Delta_a = \mu^* - \mu_a$.

Definition 10. For an arm $a \in \mathcal{A}$ and a time $t \in \mathbb{N}$, we denote by $N_{t,a}$ the number of times that arm a has been pulled before time t, that is $N_{t,a} = \sum_{s=0}^{t-1} \mathbb{I}\{A_s = a\}$.

Lemma 11. *For $\mathcal{A}$ finite, the regret R_T can be expressed as a sum over the arms and their gaps:*

$$R_T = \sum_{a \in \mathcal{A}} N_{T,a} \Delta_a .$$

Proof.

$$\begin{aligned} R_T = T\mu^* - \sum_{t=0}^{T-1} \mu_{A_t} &= T\mu^* - \sum_{a \in \mathcal{A}} \sum_{t=0}^{T-1} \mathbb{I}\{A_t = a\}\mu_a \\ &= T\mu^* - \sum_{a \in \mathcal{A}} N_{T,a}\mu_a \\ &= \sum_{a \in \mathcal{A}} N_{T,a}\mu^* - \sum_{a \in \mathcal{A}} N_{T,a}\mu_a \\ &= \sum_{a \in \mathcal{A}} N_{T,a}\Delta_a . \end{aligned}$$

□

2.3 Alternative model

The description of the bandit model above considers that at time t, a reward X_t is generated, depending on the arm A_t pulled at that time. An alternative way to talk about that process is to imagine that there is a stream of rewards from each arm, and that the algorithm sees the first, then second, etc. reward from the arms at it pulls them. We introduce definitions to talk about the n^{th} reward of an arm, and the time at which that reward is pulled.

Definition 12. For an arm $a \in \mathcal{A}$ and a time $n \in \mathbb{N}$, we denote by $T_{n,a}$ the time at which arm a was pulled for the n-th time, that is $T_{n,a} = \min\{s \in \mathbb{N} \mid N_{s+1,a} = n\}$. Note that $T_{n,a}$ can be infinite if the arm is not pulled n times.

Definition 13. For $a \in \mathcal{A}$ and $n \in \mathbb{N}$, let $Z_{n,a} \sim \nu(a)$, independent of everything else. We define $Y_{n,a} = X_{T_{n,a}} \mathbb{I}\{T_{n,a} < \infty\} + Z_{n,a} \mathbb{I}\{T_{n,a} = \infty\}$, the reward received when pulling arm a for the n-th time if that time is finite, and equal to $Z_{n,a}$ otherwise.

TODO: that definition requires changing the probability space to $\Omega \times \mathbb{R}^{\mathbb{N} \times \mathcal{A}}$.

Lemma 14. *$T_{N_{t,a},a} \leq t - 1 < \infty$ for all $t \in \mathbb{N}$ and $a \in \mathcal{A}$.*

Proof. By definition, $T_{N_{t,a},a} = \min\{s \in \mathbb{N} \mid N_{s+1,a} = N_{t,a}\} \leq t - 1 < \infty$. □

Lemma 15. *$T_{N_{t+1,A_t},A_t} = t$ for all $t \in \mathbb{N}$.*

Proof. □

Lemma 16. *$Y_{N_{t+1,A_t},A_t} = X_t$ for all $t \in \mathbb{N}$ and $a \in \mathcal{A}$.*

Proof. By Lemma 15, we have $T_{N_{t+1,A_t},A_t} = t < \infty$, so $Y_{N_{t+1,A_t},A_t} = X_{T_{N_{t+1,A_t},A_t}} = X_t$. □

Lemma 17.

$$\sum_{n=1}^{N_{t,a}} Y_{n,a} = \sum_{s=0}^{t-1} \mathbb{I}\{A_s = a\} X_s \,.$$

Proof. □

Chapter 3

Concentration inequalities

Chapter 4

Bandit algorithms

4.1 Explore-Then-Commit

Note: times start at 0 to be consistent with Lean.

Note: we will describe the algorithm by writing $A_t = ...$, but our formal bandit model needs a policy π_t that gives the distribution of the arm to pull. What me mean is that π_t is a Dirac distribution at that arm.

Definition 18 (Explore-Then-Commit algorithm)**.** The Explore-Then-Commit (ETC) algorithm with parameter $m \in \mathbb{N}$ is defined as follows:

1. for $t < Km$, $A_t = t \mod K$ (pull each arm m times),
2. compute $\hat{A}_m^* = \arg\max_{a \in [K]} \hat{\mu}_a$, where $\hat{\mu}_a = \frac{1}{m} \sum_{t=0}^{Km-1} \mathbb{I}(A_t = a) X_t$ is the empirical mean of the rewards for arm a,
3. for $t \geq Km$, $A_t = \hat{A}_m^*$ (pull the empirical best arm).

Lemma 19. *For the Explore-Then-Commit algorithm with parameter m, for any arm $a \in [K]$ and any time $t \geq Km$, we have*

$$N_{t,a} = m + (t - Km)\mathbb{I}\{\hat{A}_m^* = a\} .$$

Proof. □

Theorem 20. *Suppose that $\nu(a)$ is 1-sub-Gaussian for all arms $a \in [K]$. Then for the Explore-Then-Commit algorithm with parameter m, the expected regret after T pulls with $T \geq Km$ is bounded by*

$$\mathbb{E}[R_T] \leq m \sum_{a=1}^{K} \Delta_a + (T - Km) \sum_{a=1}^{K} \Delta_a \exp\left(-\frac{m\Delta_a^2}{4}\right) .$$

Proof. By Lemma 11, we have $\mathbb{E}[R_T] = \sum_{a=1}^{K} \mathbb{E}\left[N_{T,a}\right] \Delta_a$. It thus suffices to bound $\mathbb{E}[N_{T,a}]$ for each arm a with $\Delta_a > 0$. It suffices to prove that

$$\mathbb{E}[N_{T,a}] \leq m + (T - Km) \exp\left(-\frac{m\Delta_a^2}{4}\right) .$$

By definition of the Explore-Then-Commit algorithm (or by Lemma 19),

$$N_{T,a} = m + (T - Km)\mathbb{I}\{\hat{A}^*_m = a\} .$$

It thus suffices to prove the inequality $\mathbb{P}(\hat{A}^*_m = a) \leq \exp\left(-\frac{m\Delta_a^2}{4}\right)$ for $\Delta_a > 0$.

$$\begin{aligned}\mathbb{P}(\hat{A}^*_m = a) &\leq \mathbb{P}(\hat{\mu}_a \geq \hat{\mu}_{a^*}) \\ &= \mathbb{P}\left(\frac{1}{m}\sum_{t=0}^{Km-1} \mathbb{I}(A_t = a)X_t \geq \frac{1}{m}\sum_{t=0}^{Km-1} \mathbb{I}(A_t = a^*)X_t\right) .\end{aligned}$$

TODO: here we need to state in a rigorous way that the empirical means are means of m i.i.d. samples $Y_{a,i}$ and $Y_{a^*,i}$ from the distributions $\nu(a)$ and $\nu(a^*)$.

$$\begin{aligned}&= \mathbb{P}\left(\frac{1}{m}\sum_{i=1}^{m} Y_{a,i} \geq \frac{1}{m}\sum_{i=1}^{m} Y_{a^*,i}\right) \\ &= \mathbb{P}\left(\frac{1}{m}\sum_{i=1}^{m} (Y_{a,i} - Y_{a^*,i} + \Delta_a) \geq \Delta_a\right) .\end{aligned}$$

$Y_{a,i} - Y_{a^*,i} + \Delta_a = (Y_{a,i} - \mu_a) - (Y_{a^*,i} - \mu_{a^*})$ has mean 0 and is 2-sub-Gaussian, since $Y_{a,i} - \mu_a$ and $Y_{a^*,i} - \mu_{a^*}$ are 1-sub-Gaussian and independent. By Hoeffding's inequality, we have

$$\mathbb{P}\left(\frac{1}{m}\sum_{i=1}^{m} (Y_{a,i} - Y_{a^*,i} + \Delta_a) \geq \Delta_a\right) \leq \exp\left(-\frac{m\Delta_a^2}{4}\right) .$$

This concludes the proof. □

4.2 UCB

Chapter 5

Practical Algorithms

The algorithms presented in the previous chapters are theoretical algorithms, in the very basic sense that they perform operations on real numbers to decide what they will sample, and those numbers are not representable in a computer. In practice, we need to use floating point numbers (or bounded integers or rationals), both for the algorithm and for the rewards that are sampled from the environment.

We will want to perform experiments about the algorithms for which we have theoretical guarantees, and that means using an implementation of the algorithms that can actually run on a computer. However if we use `float` in the specification of the algorithms, we won't be able to prove anything, because those numbers lack any good algebraic properties.

Our strategy to resolve that conflict is to describe first the algorithm as a function of real numbers, and then automatically generate a version of the algorithm that uses floating point numbers, by using a translation tactic (similar to `to_additive`). The result is that we have only one hand-written algorithm, and even though the theorems don't directly apply to the floating point version, we are assured that the only difference between the experimental and theoretical algorithms is the result of the translation tactic. This is in stark contrast with the situation in the literature, where the theoretical algorithm is specified in pseudo-code in LaTeX and the experiments are done on a separately written implementation, typically in Python. The experimental version may or may not be also implementing a series of tricks or optimizations of constants.

TODO: describe the translation tactic. It should basically be like `to_additive`.

Chapter 6

Sampling