FAKULTÄT FÜR WIRTSCHAFTSINFORMATIK
UND WIRTSCHAFTSMATHEMATIK

UNIVERSITÄT MANNHEIM

Prof. Dr. Leif Döring
Daniel Schmidt, Benedikt Wille

Reinforcement Learning

1. Exercise Sheet

16.02.2026

For this assignment, knowledge from lectures 1 to 4 is assumed.

1. The Regret - Part 1

Recall Definitions 1.2.1 and 1.2.6 from the lecture. Suppose ν is a bandit model and $(\pi_t)_{t=1,\dots,n}$ a learning strategy. Then the regret is defined by

$$R_n(\pi) := nQ_* - \mathbb{E}_\pi\left[\sum_{t=1}^n X_t\right], \quad n \in \mathbb{N},$$

where $Q_* := \int_{-\infty}^{\infty} x P_{a_*}(dx)$ the expected reward of the best arm $a_* = \text{argmax}_a Q_a$.

a) Suppose a two-armed bandit with $Q_1 = 1$ and $Q_2 = -1$ and a learning strategy π given by

$$\pi_t = \begin{cases} \delta_1, & t \text{ even}, \\ \delta_2, & t \text{ odd}. \end{cases}$$

Calculate the regret $R_n(\pi)$.

b) Define a stochastic bandit and a learning strategy such that the regret is 5 for all $n \geq 5$.

c) Show for all learning strategies π that $R_n(\pi) \geq 0$ and $\limsup_{n\to\infty} \frac{R_n(\pi)}{n} < \infty$.

d) Let $R_n(\pi) = 0$. Suppose that the best arm is unique. Prove that π is deterministic, i.e. all π_t are almost surely constant and only chose the best arm.

e) Suppose ν is a 1-subgaussian bandit model with k arms and $km \leq n$, then consider the Explore then Commit algorithm and recall the regret bound:

$$R_n \leq \underbrace{m \sum_{a\in\mathcal{A}} \Delta_a}_{\text{exploration}} + \underbrace{(n-mk) \sum_{a\in\mathcal{A}} \Delta_a \exp\Big(-\frac{m\Delta_a^2}{4}\Big)}_{\text{exploitation}}.$$

Assume now $k = 2$, such that $\Delta_1 = 0$ and $\Delta_2 = \Delta$ then we get

$$R_n \leq m\Delta + (n-m2)\Delta \exp\Big(-\frac{m\Delta^2}{4}\Big) \leq m\Delta + n\Delta \exp\Big(-\frac{m\Delta^2}{4}\Big).$$

Show that this upper bound is minimized for $m = \max\Big\{1, \Big\lceil \frac{4}{\Delta^2} \log(\frac{n\Delta^2}{4}) \Big\rceil\Big\}$.

2. The Regret - Part 2

Show the following two claims.

a) If the failure probability do not decay to zero then the regret grows linearly.

b) If the failure probability $\tau_n(\pi)$ behaves like $\frac{1}{n}$, then the regret behaves like $\sum_{a \in \mathcal{A}} \Delta_a \log(n)$ with constants that depend on the concrete bandit model.
Hint: Recall from basic analysis that $\int_1^n \frac{1}{x} dx = \log(n)$ *and how to relate sums and integrals for monotone integrands.*

3. Sub-Gaussian random variables

Recall Definition 1.3.3. of a σ-sub-Gaussian random variable X.

a) Show that every σ-sub-Gaussian random variable satisfies $\mathbb{V}[X] \leq \sigma^2$.

b) Suppose X is σ-sub-Gaussian. Prove that cX is $|c|\sigma$-sub-Gaussian.

c) Show that $X_1 + X_2$ is $\sqrt{\sigma_1^2 + \sigma_2^2}$-sub-Gaussian if X_1 and X_2 are independent σ_1-sub-Gaussian and σ_2-sub-Gaussian random variables.

d) Show that a Bernoulli-variable is $\frac{1}{2}$-sub-Gaussian.

e) Show that every centered bounded random variable, say bounded below by a and above by b is $\frac{(b-a)}{2}$-sub-Gaussian.

4. Regret Bound

Recall the upper bound on the regret for ETC in the case of two arms from exercise 1. Show that

$$R_n(\pi) \leq \Delta + C\sqrt{n}$$

for some model-free constant C so that, in particular, $R_n(\pi) \leq 1 + C\sqrt{n}$ for all bandit models with regret bound $\Delta \leq 1$ (for instance for Bernoulli bandits).
Hint: First show that Equation (1.2) in the lecture notes is true and then use the same trick as in the proof of Theorem 1.3.9.

5. Advanced: ϵ-greedy Regret

Let π be the learning strategy that first explores each arm once and then continuous according to ϵ-greedy for some $\epsilon \in (0, 1)$ fixed. Furthermore, assume that ν is a 1-sub-gaussian bandit model. Show that the regret grows linearly:

$$\lim_{n \to \infty} \frac{R_n(\pi)}{n} = \frac{\epsilon}{K} \sum_{a \in \mathcal{A}} \Delta_a$$

Coding Assignments

Submissions

Each exercise sheet will include coding tasks. Over the course of the semester, you must submit three of these tasks. The required tasks will be marked accordingly, with a submission deadline set two weeks after the sheet is uploaded.

You can work in groups of up to three people. Submissions will be graded as pass or fail. To be eligible for the exam, you must pass all three required submissions. If a submission is graded as fail, your group will have an additional week to make improvements and resubmit the code. Submissions should be sent via email. To ensure reproducibility, you should clearly state the Python version and the packages you used. Additionally, your code should be well-commented and parameter names should be sensible. The code should be organized neatly using the tools we showed you in the first exercise session. You can get a refresher on these tools using the guide on our website.

Working with Python

The tasks should be completed using Python. If you need a Python refresher, we recommend this page. We generally recommend using VSCode and pyenv (which is also available for Windows). If you run into setup issues, feel free to send us an email or ask us after the exercise session — we're happy to help!

You are free to organize your code as you like—whether in a single Python file, Jupyter Notebooks, or a structured GitHub repository. However, we strongly recommend keeping your work well-organized. As we progress through the material, it will become harder to keep track of everything without a structured approach.

Implementation Tips

The assignments will include implementation tips that you can follow but are not required to. In the beginning, especially when implementing bandits, the suggested structure might seem overly complex. However, this structure will be essential later when we move into Tabular Reinforcement Learning. It can be beneficial to follow this approach from the start.

6. *Programming task: Bandits

The aim of this task is to implement several kinds of stochastic bandits. You should be able to play Gaussian and Bernoulli bandits (i.e. the arms are normally/bernoulli distributed). It should be possible to set the means of the arms manually or randomly. For the gaussian random means case they should be independently standard normally distributed and for the bernoulli random means case the parameter p should be independently uniformly distributed between 0 and 1. Additionally, implement a mode where you specify a certain reward gap Δ and after drawing the random means fix the mean of the highest arm and replace the other means in descending order by values decreasing by the reward gap. In this case, after the arm with the highest mean

μ^* was detected, the mean of the k-th biggest arm gets replaced by $\mu^* - k\Delta$, where in the case of Bernoulli bandits you can set all means that would be negative to zero.

Hint: You can achieve this by implementing a class that takes the number of arms, the distribution mode, a (possibly empty) list of means, and arguments for determining the means as inputs. The class should have a function that allows to perform an action (drawing of a certain arm) that returns a value according to the distribution of the arm.

7. *Programming task: ETC Algorithm

The aim of this task is to implement the ETC Algorithm. To this end, write a function or class with a function that executes one step of the algorithm.

Hint: You can achieve this by implementing a class that takes the bandit and number of exploration rounds as inputs. You will need to keep track of the amount of played rounds internally. The class should have a function performing one step of the algorithm that returns the chosen arm and the reward obtained during the step.

8. *Programming task: A first experiment

Simulate a 10-armed Gaussian bandit with random means and play the bandit $n = 10000$ times with the explore-then-commit learning strategy for different choices of m. Average your results over $N = 1000$ iterations of the experiment and plot the following:

(a) the regrets over time,

(b) the correct action rates over time, i.e. the averaged amount of times the algorithm played the best arm at each timestep,

(c) the estimations of the means of the different arms plotted against the actual means of the arms over time, and

(d) the average probabilities of choosing each of the arms over time.

Use the regret bound from Theorem 1.3.2 to find an optimal choice of m that minimizes the regret. If you cannot find an analytic form for the optimal m, use a numerical method to determine the optimal m. Compare the results to the ones with your arbitrary choices of m.

To this end, you should write a function that executes N runs of a given algorithm and aggregates the necessary metrics for the plots as well as the plot functions. Although you will not need it for this exercise yet, please log the variances of the necessary metrics as well.

Hint: You need not overcomplicate things at this point. Think about which metrics you need for the plots, how and when to extract that information while training the algorithm, and how to store the data accordingly. To keep things in order you might consider separating the tasks of executing one run of the algorithm and computing the averages and variances across the runs into different functions. Do not get too obsessed with „pretty plotting"- as long as you have plots that you can reasonably interpret it will be sufficient for this course.

The solution to the theoretical exercises will be discussed in the exercise class in B2 on February 23, 2026, at Mathelounge in B6 B301.