Online and Reinforcement Learning

2023-2024

Home Assignment 6

Christian Igel **Yevgeny Seldin**
Department of Computer Science
University of Copenhagen

The deadline for this assignment is **20 March 2024, 21:00**. You must submit your *individual* solution electronically via the Absalon home page.

A solution consists of:

- A PDF file with detailed answers to the questions, which may include graphs and tables if needed. Do *not* include your full source code in the PDF file, only selected lines if you are asked to do so.
- A .zip file with all your solution source code with comments about the major steps involved in each question (see below). Source code must be submitted in the original file format, not as PDF. The programming language of the course is Python.

Important Remarks:

- IMPORTANT: Do NOT zip the PDF file, since zipped files cannot be opened in *SpeedGrader*. Zipped PDF submissions will not be graded.
- Your PDF report should be self-sufficient. I.e., it should be possible to grade it without opening the .zip file. We do not guarantee opening the .zip file when grading.
- Your code should be structured such that there is one main file (or one main file per question) that we can run to reproduce all the results presented in your report. This main file can, if you like, call other files with functions, classes, etc.
- Handwritten solutions will not be accepted.

1 Deep Q-Learning (50 points)

We consider the *Lunar Lander* environment from the Gymnasium framework, see `https://gymnasium.farama.org/environments/box2d/lunar_lander/` for details of the RL task. You may need to install Gymnasium via `pip install gymnasium[box2d]` and perhaps some other packages.

The notebook `LunarLanderDeepAssignment.ipynb` implements a simple deep Q-learning except for one line. If you have problems with the visualization, you can comment out the corresponding lines (try to get it running, it is fun).

1.1 Neural architecture

The policy network definition contains the lines:

```
    def forward(self, x_input):
        x = F.tanh(self.fc1(x_input))
```

```
        x = F.tanh(self.fc2(x))
        x = torch.cat((x_input, x), dim=1)
        x = self.output_layer(x)
        return x
```

1.1.1 Structure (4 points)

What is `x = torch.cat((x_input, x), dim=1)` doing?

1.1.2 Activation function (8 points)

Why has the tanh activation function been chosen instead of, say, the standard logistic function?

1.2 Adding the Q-learning (16 points)

Add the missing line computing the variable `targets_tensor` setting the actual Q-learning targets. The line should be added to the notebook after the comment line `# Compute targets`.

After that, the notebook should reliably find a policy solving the task with a return larger that 200 (of course, there might be bad trials that do not find a sufficiently good solution).

Report the line in the report and *briefly* explain what it is doing. State the equation it implements using the notation introduced in the lecture.

The notebook saves a plot of the learning process in the file `deepQ.pdf`. Show a successful learning process in the report by including the corresponding plot (or a beautified version of it).

1.3 Epsilon (4 points)

Explain what the line

```
    explore_p = explore_stop + (explore_start - explore_stop)*np.exp(-decay_rate*ep)
```

is doing.

1.4 Gather (8 points)

Explain what the line

```
    Q_tensor = torch.gather(output_tensor, 1, actions_tensor.unsqueeze(-1)).squeeze()}
```

is doing and why it is necessary (it is for a data structure/programming reason, not a theoretical one).

1.5 No grad (10 points)

Explain what the line

```
    with torch.no_grad():
```

is doing and why it is – from the lecture's conceptual point of view – necessary.

1.6 Programming (optional, 0 points)

Any suggestions how to make the programming more elegant/efficient? Send your suggestions to igel@diku.dk.

2 A tighter analysis of the Hedge algorithm (20 points)

1. Apply Hoeffding's lemma (Lemma 2.6 in Yevgeny's lecture notes) in order to derive a better parametrization and a tighter bound for the expected regret of the Hedge algorithm. [Do not confuse Hoeffding's lemma with Hoeffding's inequality!] Guidance:

   (a) Traverse the analysis of the Hedge algorithm that we did in class. There will be a place where you will have to bound expectation of an exponent of a function ($\sum_a p_t(a)e^{-\eta X_{t,a}}$). Instead of going the way we did, apply Hoeffding's lemma.

   (b) Find the value of η that minimizes the new bound. (You should get $\eta = \sqrt{\frac{8 \ln K}{T}}$ - please, prove this formally.)

   (c) At the end you should obtain $\mathbb{E}[R_T] \leq \sqrt{\frac{1}{2} T \ln K}$. (I.e., you will get an improvement by a factor of 2 compared to what we did in class.)

   Remark: Note that the regret upper bound matches the lower bound up to the constants. This is an extremely rare case.

2. Explain why the same approach cannot be used to tighten the regret bound of the EXP3 algorithm.

3 Empirical evaluation of algorithms for adversarial environments (5 points)

Is it possible to evaluate experimentally the quality of algorithms for adversarial environments? If yes, how would you design such an experiment? If no, explain why it is not possible.

Hint: Think what kind of experiments can certify that an algorithm for an adversarial environment is good and what kind of experiments can certify that the algorithm is bad? How easy or hard is it to construct the corresponding experiments?

4 Empirical comparison of UCB1 and EXP3 algorithms (25 points)

Implement and compare the performance of UCB1 with improved parametrization derived in the earlier home assignment, and EXP3 in the i.i.d. multiarmed bandit setting. For EXP3 take time-varying $\eta_t = \sqrt{\frac{\ln K}{tK}}$. You can use the following settings:

- Time horizon $T = 10000$.
- Take a single best arm with bias $\mu^* = 0.5$.
- Take $K - 1$ suboptimal arms for $K = 2, 4, 8, 16$.
- For suboptimal arms take $\mu = \mu^* - \frac{1}{4}$, $\mu = \mu^* - \frac{1}{8}$, $\mu = \mu^* - \frac{1}{16}$ (3 different experiments with all suboptimal arms being equally bad).

Make 20 repetitions of each experiment and for each experiment plot the average empirical pseudo regret (over the 20 repetitions) as a function of time and the average empirical pseudo regret + one standard deviation (over the 20 repetitions). The empirical pseudo regret is defined as $\hat{R}_T = \sum_{t=1}^{T} \Delta(A_t)$.

Important: do not forget to add a legend and labels to the axes.

Break UCB1 Now design an adversarial sequence for which you expect UCB1 to suffer linear regret. Consider UCB1 with random tie breaking. Explain how you design the sequence and execute UCB1 and EXP3 on it and report the observations (in a form of a plot). Even though the sequence is deterministic, you should make several repetitions (say, 20), because there is internal randomness in the algorithms. (If you fail to break UCB1 with randomized tie breaking, break UCB1 with deterministic tie breaking, we will not deduct points for this.)

You are welcome to try other settings (not for submission). For example, check what happens when μ^* is close to 1 or 0. Or what happens when the best arm is not static, but switches between rounds. Even though you can break UCB1, it is actually pretty robust, unless you design adversarial sequences that exploit the knowledge of the algorithm.

---

5 [Optional, not for submission] The doubling trick

Consider the following forecasting strategy (the "doubling trick"): time is divided into periods $(2^m, \ldots, 2^{m+1}-1)$, where $m = 0, 1, 2, \ldots$. (In other words, the periods are $(1), (2,3), (4, \ldots, 7), (8, \ldots, 15), \ldots$.) In period $(2^m, \ldots, 2^{m+1}-1)$ the strategy uses the optimized Hedge forecaster (from the previous question) initialized at time 2^m with parameter $\eta_m = \sqrt{\frac{8 \ln K}{2^m}}$. Thus, the Hedge forecaster is reset at each time instance that is an integer power of 2 and restarted with a new value of η. By the analysis of optimized Hedge we know that with $\eta_m = \sqrt{\frac{8 \ln K}{2^m}}$ its expected regret within the period $(2^m, \ldots, 2^{m+1}-1)$ is bounded by $\sqrt{\frac{1}{2} 2^m \ln K}$.

1. Prove that for any $T = 2^m - 1$ the overall expected regret (considering the time period $(1, \ldots, T)$) of this forecasting strategy satisfies

$$\mathbb{E}[R_T] \leq \frac{1}{\sqrt{2}-1} \sqrt{\frac{1}{2} T \ln K}.$$

   (Hint: at some point in the proof you will have to sum up a geometric series.)

2. Prove that for any arbitrary time T the expected regret of this forecasting strategy satisfies

$$\mathbb{E}[R_T] \leq \frac{\sqrt{2}}{\sqrt{2}-1} \sqrt{\frac{1}{2} T \ln K}.$$

Remark: The expected regret of "anytime" Hedge with $\eta_t = 2\sqrt{\frac{\ln K}{t}}$ satisfies $\mathbb{E}[R_T] \leq \sqrt{T \ln K}$ for any T. For comparison, $\frac{1}{\sqrt{2}(\sqrt{2}-1)} \approx 1.7$ and $\frac{1}{\sqrt{2}-1} \approx 2.4$. Thus, anytime Hedge is both more elegant and more efficient than Hedge with the doubling trick.

6 [Optional, not for submission] Rewards vs. losses

The original EXP3 algorithm for multiarmed bandits was designed for the game with rewards rather than losses (see Auer et al. (2002, Page 6)). In the game with rewards we have an infinite matrix of rewards $\{r_t^a\}_{a \in \{1,\ldots,K\}, t \geq 1}$, where $r_t^a \in [0,1]$. On each round of the game the algorithm plays an action A_t and accumulates and observed reward $r_t^{A_t}$. The remaining rewards r_t^a for $a \neq A_t$ remain unobserved.

On the one hand, we can easily convert rewards into losses by taking $\ell_t^a = 1 - r_t^a$ and apply the EXP3 algorithm we saw in class. On the other hand, a bit surprisingly, working directly with rewards (as Auer et al. did) turns to be more cumbersome and less efficient. The high-level reason is that the games with rewards and losses have a different dynamics. In the rewards game when an action is played its

relative quality (expressed by the cumulative reward) increases. Therefore, we need explicit exploration to make sure that we do not get locked on a suboptimal action. In the losses game when an action is played its relative quality (expressed by the cumulative loss) decreases. Therefore, we never get locked on any particular action and exploration happens automatically without the need to add it explicitly (sometimes this is called implicit exploration). The low-level reason when it comes down to the analysis of the algorithm is that it is easier to upper bound the exponent of x for negative x as opposed to positive x.

The original EXP3 algorithm for the rewards game looks as follows, where the most important difference with the algorithm for the losses game is highlighted in red and two additional minor differences are highlighted in blue (we explicitly emphasize that the sign in the exponent changes from "-" to "+"). $\tilde{R}_t$ is used to denote cumulative importance-weighted rewards.

Algorithm 1 The EXP3 Algorithm for the game with rewards and fixed time horizon

1: $\forall a : \tilde{R}_0(a) = 0$
2: **for** $t = 1, \dots, T$ **do**
3: $\quad \forall a : \; p_t(a) = (1-\eta)\frac{e^{+\eta \tilde{R}_{t-1}(a)}}{\sum_{a'} e^{+\eta \tilde{R}_{t-1}(a')}} + \frac{\eta}{K}$
4: $\quad$ Sample A_t according to p_t and play it
5: $\quad$ Observe r_{t,A_t}
6: $\quad \forall a : \; \tilde{r}_{t,a} = \frac{r_{t,a}\mathbb{1}(A_t=a)}{p_t(a)} = \begin{cases} \frac{r_{t,a}}{p_t(a)}, & \text{if } A_t = a \\ 0, & \text{otherwise} \end{cases}$
7: $\quad \forall a : \; \tilde{R}_t(a) = \tilde{R}_{t-1}(a) + \tilde{r}_{t,a}$
8: **end for**

1. Explain why the analysis of the EXP3 algorithm for the rewards game without the addition of explicit exploration $\frac{\eta}{K}$ in Line 3 of the algorithm would not work. More specifically - if you would try to follow the lines of the analysis of EXP3 with losses, at which specific point you would get stuck and why?

2. How the addition of explicit exploration term $\frac{\eta}{K}$ in Line 3 of the algorithm allows the analysis to go through? (You can check the analysis of the algorithm in Auer et al. (2002, Page 7).)

3. By how much the expected regret guarantee for EXP3 with rewards is weaker than the expected regret guarantee for EXP3 with losses? (Check Auer et al. (2002, Corollary 3.2) and assume that g takes its worst-case value, which is T.)

4. You are mostly welcome to experiment and see whether theoretical analysis reflects the performance in practice. I.i.d. experiments will be the easiest to construct, but you are also welcome to try adversarial settings.

References

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal of Computing*, 32(1), 2002.