Online and Reinforcement Learning

2023-2024

Home Assignment 7

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

The deadline for this assignment is **3 April 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 PPO (25 points)

Let us consider the Proximal Policy Optimization (PPO) algorithm in the variant presented in the lecture and described on the slides "Deep Reinforcement Learning".

1.1 Return expressed as advantage over another policy (5 points)

The expected return of a policy π' can be expressed in terms of its advantage over another policy π and $J(\pi)$:

$$J(\pi') = J(\pi) + \mathbb{E}\left\{ \sum_{t=0}^{\infty} \gamma^t \underbrace{A^{\pi}(s_t, a_t)}_{\text{advantage of following } \pi' \text{ instead of } \pi} \,\middle|\, s_0, \pi' \right\}$$

Provide a proof (with intermediate steps) using the notation from the lecture.

1.2 Clipping (10 points)

PPO uses the clipping

$$\min\left(\frac{\pi'(a_t,s_t)}{\pi(a_t,s_t)}\hat{A}^{\pi}(s_t,a_t), \text{clip}\left(\frac{\pi'(a_t,s_t)}{\pi(a_t,s_t)}, 1-\epsilon, 1+\epsilon\right)\hat{A}^{\pi}(s_t,a_t)\right)$$

with $\text{clip}(x,l,u) = \min(\max(x,l),u)$.

In the lecture, it has been stated that the gradient-based update of the PPO-Clip objective mentioned above updates the policy "if $\frac{\pi'(a_t,s_t)}{\pi(a_t,s_t)} \in [1-\epsilon, 1+\epsilon]$ or if the update leads to getting closer to this range". The first condition is trivial. Assume that the first condition does not hold. What is is meant by "if the gradient direction does not point away from the interval"? Formalize and prove this condition (assuming the first condition is not met, i.e., $\frac{\pi'(a_t,s_t)}{\pi(a_t,s_t)} \notin [1-\epsilon, 1+\epsilon]$).

1.3 Pi prime in PPO (10 points)

The policy generating the experience, see procedure "Gather experience" on the slides, uses the policy π' with parameters $\boldsymbol{\theta'}$, where the probability of an action a_t^e taken in a state s_t^e (in episode e at step t) is stored in p_t^e. The PPO update, as described on the slide "PPO-Clip optimization", considers the expression $\frac{\pi'(a_t^e,s_t^e)}{p_t^e}$. Why is this expression not always one?

2 `MBIE` and Prior Knowledge on Support Sets (25 points)

In this exercise, we study a variant of `MBIE` that leverages prior knowledge to reduce exploration. Specifically, we are interested in incorporating some prior knowledge on the transition graph of the MDP into the algorithm. For a given (s,a) pair, let $\mathcal{K}_{s,a}$ denote the *support set* of the distribution $P(\cdot|s,a)$:

$$\mathcal{K}_{s,a} := \Big\{x \in \mathcal{S} : P(x|s,a) > 0\Big\}.$$

In words, $\mathcal{K}_{s,a}$ is the set of all possible next-states when choosing action a in state s. For example, in RiverSwim, $\mathcal{K}_{s_2,\mathsf{left}} = \{s_1\}$ whereas $\mathcal{K}_{s_2,\mathsf{right}} = \{s_1, s_2, s_3\}$. In the general setting of RL, not only P but also its associated support sets $\mathcal{K}_{s,a}, s \in \mathcal{S}, a \in \mathcal{A}$ are assumed unknown to the agent. We are interested in studying an intermediate setting, where P is unknown, but the associated support sets are known through, e.g., some domain knowledge. Intuitively, this corresponds to knowing the transition graph of the MDP, but without knowing the actual probabilities.

Assume that support set $\mathcal{K}_{s,a}$ for each pair (s,a) is provided to the agent. Then, the agent can leverage this prior knowledge to *rule out* some candidate models in $\mathcal{M}_t$, the high-probability set of plausible models maintained by `MBIE` at time t. (For example, in RiverSwim, the agent would know that a model that would include a next-state of s_4 under (s_2, right) is certainly wrong as it contradicts with the prior knowledge $\mathcal{K}_{s_2,\mathsf{right}}$; such a model could therefore be shave off from $\mathcal{M}_t$.)

(i) Argue how the confidence set for P could be modified to incorporate the prior knowledge on the support sets. Write down the precise mathematical form of the revised confidence sets.

(ii) Modify `MBIE` using this confidence set proposed in Part (i). Let us call this algorithm `MBIE-supp`. Implement `MBIE-supp` (i.e., modify `HA5_Q5.py`), and examine it in the 5-state Ergodic RiverSwim using the same parameters as in Exercise 5 of Home Assignment 5. Plot $n(t)$ versus time t (for a single run) under both `MBIE` and `MBIE-supp`.

(iii) *(Optional, not for submission.)* What if only some support sets are known a priori?

3 Offline Evaluation of Bandit Algorithms (50 points)

1. Evaluation of algorithms for online learning with limited feedback in real life (as opposed to simulations) is a challenging topic. The straightforward way is to implement an algorithm, execute it "live", and measure the performance, but most often this is undesirable. Give two reasons why.

2. There are two alternative offline evaluation methods: importance-sampling and rejection sampling. In both cases we have to know the distribution that was used for collecting the data. Note that we only observe the reward (or loss) for an action taken by the algorithm when the action matches the action of the logging policy. In the importance-sampling approach we reweigh the reward by inverse probability of the action being taken by the logging policy when there is a match and assign zero reward otherwise. The rejection sampling approach requires that the logging policy samples all actions uniformly at random. At the evaluation phase rejection sampling scrolls through the log of events until the first case where the action of the logging policy matches the action of the evaluated policy. The corresponding reward is assigned and all events that were scrolled over are discarded. You can read more about the rejection sampling approach in Li et al. (2011). The importance-sampling approach is more versatile, because it does not require a uniform logging policy. With importance-sampling it is possible to take data collected by an existing policy and evaluate new policies, as long as the logging distribution is known and strictly positive for all actions.

   The Theoretical Part of the Task Our theoretical aim is to modify the UCB1 and EXP3 algorithms to be able to apply them to logged data using the importance-sampling approach. For simplicity, we assume that the logging policy used uniform sampling. Put attention that importance weighting in offline evaluation based on uniform sampling (the one you are asked to analyse) changes the range of the rewards from $[0, 1]$ to $[0, K]$, where K is the number of actions. Recall that the original versions of UCB1 and EXP3 assumed that the rewards are bounded in the $[0, 1]$ interval. Your task is to modify the two algorithms accordingly. Put attention that the variance of the importance weighted estimates is "small" (of order K rather than of order K^2) and if you do the analysis carefully you should be able to exploit it in the modified EXP3, but not in UCB1.

   (a) Modify the UCB1 algorithm with improved parametrization from the earlier home assignment to work with importance-weighted losses generated by a logging policy based on uniform sampling. Provide a pseudo-code of the modified algorithm (at the same level as the UCB1 pseudo-code in the lecture notes) and all the necessary calculations supporting your modification, including a pseudo-regret bound. You do not need to redo the full derivation, it is sufficient to highlight the key points where you make changes and how they affect the regret bound, assuming you do it clearly.

   (b) Briefly explain why you are unable to exploit the small variance in the modified UCB1.

   (c) Modify the EXP3 algorithm from the lecture notes (with fixed time horizon T) to work with importance-weighted losses generated by a logging policy based on uniform sampling. Provide a pseudo-code of the modified algorithm (at the same level as the EXP3 pseudo-code in the lecture notes) and all the necessary calculations supporting your modification, including an expected regret bound. As already mentioned, with a careful analysis you should be able to exploit the small variance of importance-weighted losses.

   (d) Anytime modification: In order to transform the fixed-horizon EXP3 from the previous task to an anytime EXP3 (an EXP3 that assumes no knowledge of the time horizon) you should replace the time horizon T in the learning rate by the running time t and reduce the learning rate by a factor of $\sqrt{2}$. The regret bound of anytime EXP3 is larger by a factor of $\sqrt{2}$ compared to the regret bound of EXP3 tuned for a specific T. In return, the bound holds for all t and not just for one specific time T the algorithm was tuned for. All you need to do for this point is to write the new learning rate and the new regret bound, you do not need to prove anything. You can find more details on the derivation in (Bubeck and Cesa-Bianchi, 2012), if you want. In the experiments you should use the anytime version of the algorithm and the anytime expected regret bound.

The Practical Part of the Task (0 points, not for submission - you will be asked implement and run the two algorithms in Assignment 8, after we discuss your solution of the theoretical part, but you can go ahead with it, if you want) Now you should evaluate the modified UCB1 algorithm and the modified anytime EXP3 algorithm on the data.

In this question you will work with a preprocessed subset of R6B Yahoo! Front Page Today Module User Click Log Dataset[1]. The data is given in `data_preprocessed_features` file as space-separated numbers. There are 701682 rows in the file. Each row has the following format:

(a) First comes the ID of the action that was taken (the ID of an article displayed to a user). The subset has 16 possible actions, indexed from 0 to 15, corresponding to 16 articles.

(b) Then comes the click indicator (0 = no click = no reward; 1 = click = reward). You may notice that the clicks are very sparse.

(c) And then you have 10 binary features for the user, which you can ignore. (Optionally, you can try to use the features to improve the selection strategy, but this is not for submission.)

You are given that the actions were selected uniformly at random and you should work with importance-weighted approach in this question.

In the following we refer to the quality of arms by their cumulative reward at the final time step $T = 701682$. Provide plots as described in the next two points for the following subsets of arms:

i. All arms.

ii. Extract rounds with the best and the worst arm (according to the reward at T) and repeat the experiment with just these two arms. Put attention that after the extraction you can assume that you make offline evaluation with just two arms that were sampled uniformly at random [out of two arms]. The time horizon will get smaller.

iii. The same with the best and two worst arms.

iv. The best and three worst arms.

v. The best, the median, and the worst arm. (Since the number of arms is even, there are two median arms, the "upper" and the "lower" median; you can pick any of the two.)

(d) Provide one plot per each setup described above, where you report the *regret* of EXP3 and UCB1 as a function of time t. For each of the algorithms you should make 10 repetitions and report the mean and the mean $\pm$ one standard deviation over the repetitions. Put attention that the regret at running time t should be computed against the action that is the best at time t, not the one that is the best at the final time T!

(e) The same plot, where you add the expected regret bound for EXP3 and the regret of a random strategy, which picks actions uniformly at random. (We leave you to think why we are not asking to provide a bound for UCB1.)

(f) Discussion of the results.

Optional, not for submission (for those who have taken "Machine Learning B" course): since the mean rewards are close to zero and have small variance, algorithms based on the kl-inequality, such as kl-UCB (Cappé et al., 2013), or algorithms that are able to exploit small variance, for example, by using the Unexpected Bernstein inequality, are expected to perform better than Hoeffding-based UCB1.

References

Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5, 2012.

[1] `https://webscope.sandbox.yahoo.com/catalog.php?datatype=r`

Olivier Cappé, Aurélien Garivier, Odalric-Ambrym Maillard, Rémi Munos, and Gilles Stoltz. Kullback–Leibler upper confidence bounds for optimal sequential allocation. *The Annals of Statistics*, 41 (3), 2013.

Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the ACM international conference on Web Search and Data Mining*, 2011.