Online and Reinforcement Learning

2023-2024

Home Assignment 8

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

The deadline for this assignment is **14 April 2024, 22:00**. 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 Offline Evaluation of Bandit Algorithms (40 points)

1. In Home Assignment 5 you were asked to modify the UCB1 and EXP3 algorithms to work with importance-weighted data for offline evaluation. Please, write down a pseudo-code of the two modified algorithms (at the same level of detail as the pseudo-codes in the lecture notes). For EXP3, please, write down an anytime version.
2. Now you should evaluate the modified UCB1 algorithm and the modified anytime EXP3 algorithm on real 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:
   - 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.

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

- Then comes the click indicator (0 = no click = no reward; 1 = click = reward). You may notice that the clicks are very sparse.
- 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 points (2b) and (2c) below 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.)

(a) In the pdf report provide a printout of your Python implementation of the modified UCB1 and EXP3 functions. (Just the two functions, not the whole Python code, the whole code goes into the zip file.)

(b) 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!

(c) 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.)

(d) 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.

2 Comparison between UCRL2 and UCRL2-L (35 points)

The goal of this exercise is to perform an empirical evaluation of `UCRL2` (Jaksch et al., 2010) and `UCRL2-L` (see slides), and compare their regret. The original algorithm `UCRL2` differs from `UCRL2-L` only in the choice of confidence sets. `UCRL2` relies on the following confidence sets for P and R: For all $(s,a) \in \mathcal{S} \times \mathcal{A}$,

$$C_{s,a}^{\texttt{UCRL2}} = \left\{ R' \in [0,1] : |\widehat{R}_t(s,a) - R'| \leq \sqrt{\frac{3.5}{N_t(s,a)} \log\left(\frac{2SAt}{\delta}\right)} \right\},$$

$$C_{s,a}'^{\texttt{UCRL2}} = \left\{ P' \in \Delta(\mathcal{S}) : \|\widehat{P}_t(\cdot|s,a) - P'\|_1 \leq \sqrt{\frac{14S}{N_t(s,a)} \log\left(\frac{2At}{\delta}\right)} \right\}.$$

Those used in `UCRL2-L` are defined in the slides and an implementation of it is provided in the accompanied Python file `UCRL2_L.py`.

(i) Implement `UCRL2`. (You can directly modify `UCRL2_L.py`.)

(ii) Examine both `UCRL2` and `UCRL2-L` on a 6-state RiverSwim. In the experiment, set the initial state to the left-most state s_0, and set time horizon $T = 3 \times 10^5$. As for failure probability δ, set: $\delta_{\texttt{UCRL2}} = 0.05$ and $\delta_{\texttt{UCRL2-L}} = \frac{1}{4} \times 0.05 = 0.0125$ —this is to ensure a fair comparison between the two algorithms.

We are interested in the following definition of regret, as implemented in the file:

$$\mathfrak{R}(T) = \sum_{t=1}^{T} (g^\star - r_t).$$

Report the empirical regret curves under both algorithms, averaged over 50 independent runs, and report the corresponding 95% confidence intervals. (Recall that for a set of i.i.d. random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, we define the 95% confidence interval by

$$\Big[\texttt{mean}(\mathbf{X}) - 1.96\frac{\texttt{std}(\mathbf{X})}{\sqrt{n}}, \texttt{mean}(\mathbf{X}) + 1.96\frac{\texttt{std}(\mathbf{X})}{\sqrt{n}}\Big],$$

where `mean` and `std` denote the mean and the standard deviation, respectively.)

(iii) Plot the *empirical gain* $\frac{1}{t}\sum_{t'=1}^{t} r_t$ under each algorithm. In the plot, also add the horizontal line corresponding to the optimal gain $g^\star$.

(iv) Report the average number of episodes initiated by both algorithms.

(v) Using the results obtained in (ii)-(iv), discuss the differences between `UCRL2-L` and `UCRL2`.

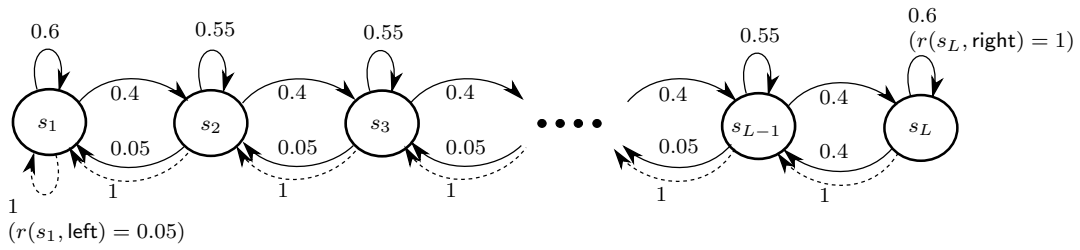

Figure 1: The L-state RiverSwim MDP

3 Leveraging Prior Knowledge on the Graph of MDP to Reduce Regret (25 points)

In this exercise, we study a variant of `UCRL2-L` that leverages prior knowledge to reduce exploration, and hence, regret. (A similar investigation was done for `MBIE` in HA7.)

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 `UCRL2-L` 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 shaved 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.

Then modify `UCRL2-L` using this new confidence set. Let us call this algorithm `UCRL2-L-supp`. Implement `UCRL2-L-supp` (i.e., modify `UCRL2_L.py`), and examine it in the 6-state Ergodic RiverSwim (Figure 2). In the experiment, set the initial state to the left-most state s_0, and set $T = 4 \times 10^5$ and $\delta = 0.05$. Report the empirical regret, averaged over 50 independent runs, along with the corresponding 95% confidence intervals.

(ii) Repeat the experiment in Part (i) for `UCRL2-L` (i.e., without prior knowledge) and discuss the results.

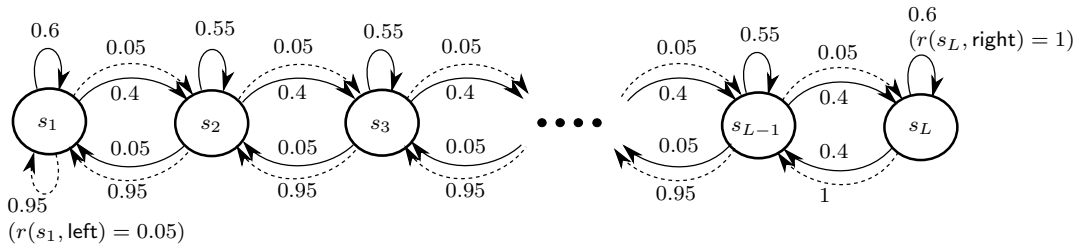

Figure 2: The L-state Ergodic RiverSwim MDP

Further, we are interested in studying another intermediate setting where P is unknown, but **cardinality** of the associated support sets are known. Hence, the agent knows the *number* of possible next-states under various state-action pairs. Intuitively, this corresponds to knowing *the degree of various nodes* in the transition graph of the MDP, but without knowing the actual probabilities or the elements of $\mathcal{K}_{s,a}$.

(iii) Now assume that $|\mathcal{K}_{s,a}|$ for each pair (s, a) is known a priori. Derive a variant of `UCRL2-L`, which we call `UCRL2-L-SuppSize`, that can use this prior knowledge. Implement `UCRL2-L-SuppSize` and examine it in 6-state Ergodic RiverSwim using the same setting as above. Report the empirical regret, averaged over 50 independent runs, along with the corresponding 95% confidence intervals. Compare the result with those of Parts (i)-(ii) above.

References

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.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11, 2010.