CS 4180/5180: Reinforcement Learning and Sequential Decision Making (Fall 2024) Lawson L.S. Wong
Northeastern University Due March 1, 2024

# Exercise 6: Temporal-Difference Learning

Please remember the following policies:

- Exercises are due at **11:59 PM** Boston time (ET).
- **Submissions should be made electronically on Canvas.**
  **Please ensure that your solutions for both the written and programming parts are present.**
  **Upload all files in a single submission, keeping the files individual (do not zip them).**
  You can make as many submissions as you wish, but only the latest one will be considered, and late days will be computed based on the latest submission.
- Each exercise may be handed in up to two days late (24-hour period), penalized by 5% per day.
  Submissions later than this will not be accepted. There is no limit on the total number of late days used over the course of the semester.
- Written solutions may be handwritten or typeset. For the former, please ensure handwriting is legible. If you write your answers on paper and submit images of them, that is fine, but please put and order them correctly in a single .pdf file. One way to do this is putting them in a Word document and saving as a PDF file.
- Programming solutions should be in Python 3, either as `.py` or `.ipynb` files.
  For the latter (notebook format), please export the output as a PDF file and submit that together. To do so, first export the notebook as an HTML file (File: Save and Export Notebook as: HTML), then open the HTML file in a brower, and finally print it as a PDF file.
- You are welcome to discuss these problems with other students in the class, but you must understand and write up the solution **and code** yourself, *and* indicate who you discussed with (if any). If you are collaborating with a large language model (LLM), acknowledge this and include your entire interaction with this system. We strongly encourage you to formulate your own answers first before consulting other students / LLMs.
- Typically, each assignment contains questions designated [CS 5180 only.], which are required for CS 5180 students. Students in CS 4180 can complete these questions for extra credit, for the same point values.
  Occasionally, there will also be [Extra credit.] questions, which can be completed by everyone. Point values for these questions depend on the depth of the extra-credit investigation.
- Contact the teaching staff if there are *extenuating* circumstances.

1. **1 point.** (RL2e 6.11, 6.12) *Q-learning vs. SARSA.*
   **Written:**
   (a) Why is Q-learning considered an off-policy control method?
   (b) Suppose action selection is greedy. Is Q-learning then exactly the same algorithm as SARSA? Will they make exactly the same action selections and weight updates?

2. **1 points.** (RL2e 7.3) *Larger random-walk task.*
   **Code/plot/written:** Read and understand Example 7.1.
   Why do you think a larger random walk task (19 states instead of 5) was used in Example 7.1?
   Would a smaller walk (fewer states) have shifted the advantage to a different value of $n$?
   How about the change in left-side outcome from 0 to $-1$ made in the larger walk?
   Do you think that made any difference in the best value of $n$?
   Use your implementation from Ex5 Q3 to provide empirical evidence for your answers.

3. **5 points.** (RL2e 6.9, 6.10) *Windy gridworld.*
**Code/plot:** In this question, you will implement several TD-learning methods and apply them to the windy gridworld in Example 6.5.

   (a) Implement the windy gridworld domain. Read the description in Example 6.5 carefully to find all details.

   (b) Implement the following methods, to be applied to windy gridworld:

   - On-policy Monte-Carlo control (for $\varepsilon$-soft policies) – consider using your code from Ex4
   - SARSA (on-policy TD control)
   - Expected SARSA
   - $n$-step SARSA (choose an appropriate $n$, e.g., $n = 4$)
   - Q-learning (off-policy TD control)
   - *Optional*: Dynamic programming (to provide an upper bound)

   To compare each method, generate line plots similar to that shown in Example 6.5 (do not generate the inset figure of the gridworld). Make sure you understand the axes in the plot, which is not the same as before (why is it different?).
   As in previous exercises, perform at least 10 trials, and show the average performance with confidence bands (1.96× standard error).
   If you implement the optional DP solution, use it to generate and plot an upper bound on performance.
   *Note*: You may adjust hyperparameters for each method as necessary; for SARSA, use the values provided in the example ($\varepsilon = 0.1, \alpha = 0.5$) so that you can reproduce the plot in the textbook.

   For the following parts, apply at least two of the above TD methods to solve them.

   (c) *Windy gridworld with King's moves*: Re-solve the windy gridworld assuming eight possible actions, including the diagonal moves, rather than four. How much better can you do with the extra actions?
   Can you do even better by including a ninth action that causes no movement at all other than that caused by the wind?

   (d) *Stochastic wind*: Re-solve the windy gridworld task with King's moves, assuming that the effect of the wind, if there is any, is stochastic, sometimes varying by 1 from the mean values given for each column. That is, a third of the time you move exactly according to these values, as you did above, but also a third of the time you move one cell above that, and another third of the time you move one cell below that. For example, if you are one cell to the right of the goal and you move left, then one-third of the time you move one cell above the goal, one-third of the time you move two cells above the goal, and one-third of the time you move to the goal.

4. **[CS 5180 only.] 2 points.** *Bias-variance trade-off.*
In lecture, we discussed that Monte-Carlo methods are unbiased but typically high-variance, whereas TD methods trade off bias to obtain lower-variance estimates. We will investigate this claim empirically in this question, from the perspective of prediction.

The overall experimental setup is as follows.

- We will continue with the deterministic (original) windy gridworld domain.
- A fixed policy $\pi$ will be specified.
- A certain number of "training" episodes $N \in \{1, 10, 50\}$ will be collected.
- Each method being investigated (TD(0), $n$-step TD, Monte-Carlo prediction) will estimate the state-value function based on the $N$ training episodes.
- We then evaluate the distribution of learning targets each method experiences at a specified state $S$.
To do so, an additional 100 "evaluation" episodes will be generated. Instead of using these to perform further updates to the state-value function, we will instead evaluate the distribution of learning targets $V(S)$ will effectively experience based on the "evaluation" episodes. For example, TD(0) will experience a set of 100 $\{R + V(S')\}$ targets, whereas Monte-Carlo will experience a set of 100 $\{G\}$ targets.

Note that in practice you should pre-collect both the training and evaluation episodes for efficiency and to ensure consistency while comparing between different methods.

(a) **Code/plot:** Perform the above experiment for the specified methods and training episodes $N$.
Use a near-optimal stochastic policy $\pi$ (e.g., found by SARSA or other methods in Q4).
Perform the evaluation for the start state (indicated 'S' in Example 6.5).
For $n$-step TD, use $n = 4$, but you may consider trying more values of $n$ as well.
Plot the histogram of learning targets experienced in the evaluation episodes for each combination of $N$ and method (i.e., at least 9 histograms total).
*Optional*: Use dynamic programming or any other appropriate method to compute the true value of $v_\pi(s)$ for comparison purposes, and add this to your plots as well.

(b) **Written:** Describe what you observe from your histograms.
Comment on what they may show about the bias-variance trade-off between the different methods, and how it may depend on the amount of training that has already occurred.

(c) **[Extra credit.]** If we considered the scenario of control (i.e., we would use on-policy action-value methods, iteratively update the policy during training, and use it to generate the next training episode), would that change the results, and how? Implement this to computationally test your hypothesis.