# ECE433/COS435 Reinforcement Learning
# Assignment 1: Value Functions
# Spring 2026

**Fill me in**

Your name here.

February 17, 2026

## Collaborators

**Fill me in**

Please fill in the names and NetIDs of your collaborators in this section. Make sure you each submit separately, but feel free to work together and submit the same answers as long as you put your names together here.

## Instructions

Writeups should be typesetted in Latex and submitted as PDF. **Please submit the asked-for snippet and answer in the solutions box as part of your writeup. Attach code as a .zip file *with the exact structure as we've distributed the zip*, we will run the code against standardized unit tests.**

**Grading.** The problem set will graded out of 50 points, and the coding assignment will also be graded out of 50 points, making the total score 100 points.

## Question 0 (0 points). Feedback

How many hours did you spend on this homework? Please fill in the solution after you've done all the questions.

**Solution**

Your solution here...

# Problem Set (50 points)

## Question 1: Discount Factor and Optimal Policy: Battery Management MDP (10 points)

A delivery robot has battery level $e \in \{0, 1, 2, 3, 4, 5\}$. The episode starts at $e = 2$. There are two actions:

- **Work** (deliver one package): if $e > 0$, you receive reward $+1$ and transition to $e' = e-1$.
- **Recharge**: if $e < 5$, you receive reward 0 and transition to $e' = e + 1$.

Absorbing states:

- If the battery reaches $e = 0$, the robot shuts down ($e = 0$ is an absorbing state, looping back to itself with reward 0).
- If the robot reaches full charge $e = 5$ (i.e., on the transition $4 \to 5$), it receives an additional bonus reward $+12$ and then loops back on itself indefinitely with reward 0 after that.

Assume an infinite-horizon discounted objective with discount factor $\gamma \in [0, 1)$.

(a) Consider two strategies starting from $e = 2$:

(a) Strategy A: *Work until shutdown.*

(b) Strategy B: *Recharge until full charge* (and collect the bonus).

Compute the discounted return of each strategy and derive an inequality in $\gamma$ characterizing when Strategy B yields higher return than Strategy A. Show your steps in detail.

Solution

Your solution here...

(c) In 1–2 sentences, explain how increasing $\gamma$ changes what the agent "cares about" in this MDP, and how that change is reflected in the optimal action at $e = 2$.

Solution

Your solution here...

## Question 2: Bias Variance Tradeoffs for Expected SARSA (15 points)

Consider two TD update targets for a state-action pair $(s, a)$:

- **Sarsa:** $\hat{v}_t = r_t + \gamma\, Q_t(s_{t+1},\, a_{t+1})$, where $a_{t+1} \sim \pi_t(\cdot \mid s_{t+1})$

- **Expected Sarsa:** $v_t = r_t + \gamma \sum_{a'} \pi_t(a' \mid s_{t+1})\, Q_t(s_{t+1}, a')$

Assume both methods share the same reward $r_t$ and next state $s_{t+1}$, and that—conditional on $s_{t+1}$—the only source of randomness distinguishing the two targets is the action selection $a_{t+1}$. Let $T_{sa}^{s'} = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ denote the transition probability, and write $\pi_{s'a'}$ as shorthand for $\pi_t(a' \mid s')$.

**(a)** Show that both update targets have the same expected value. That is, show

$$\mathbb{E}\{\hat{v}_t\} = \mathbb{E}\{v_t\}.$$

What does this tell you about the relative *bias* of the two methods (under the same policy $\pi$)?

Solution

Your solution here...

**(b)** Using the identity $\mathrm{Var}(X) = \mathbb{E}\{X^2\} - (\mathbb{E}\{X\})^2$, write out expressions for the variance of each update target. Since the two targets share the same mean, argue that comparing their variances reduces to comparing $\mathbb{E}\{X_t^2\}$ for each.

Solution

Your solution here...

**(c)** After expanding and simplifying, you should find that the difference in variance, $\mathrm{Var}(\hat{v}_t) - \mathrm{Var}(v_t)$, equals

$$\gamma^2 \sum_{s'} T_{sa}^{s'} \left( \sum_{a'} \pi_{s'a'} \big(Q_t(s', a')\big)^2 \;-\; \Big(\sum_{a'} \pi_{s'a'}\, Q_t(s', a')\Big)^2 \right).$$

Recall that for non-negative weights $w_i \geq 0$ with $\sum_i w_i = 1$, the quantity

$$\sum_i w_i\, x_i^2 \;-\; \Big(\sum_i w_i\, x_i\Big)^2$$

is the *weighted variance* of the values $x_i$. Using this fact, provide an intuitive explanation for *when* Expected Sarsa will have meaningfully lower variance than Sarsa, and when the two methods will behave similarly.

Solution

Your solution here...

**(d)** Based on your analysis, conjecture about the relative online performance of Sarsa versus Expected Sarsa in two settings:

(i) A problem with a near-greedy optimal policy where all $Q$-values at each state are similar.

(ii) A problem where $Q$-values vary widely across actions and the behavior policy has high stochasticity (e.g., large $\epsilon$ in $\epsilon$-greedy).

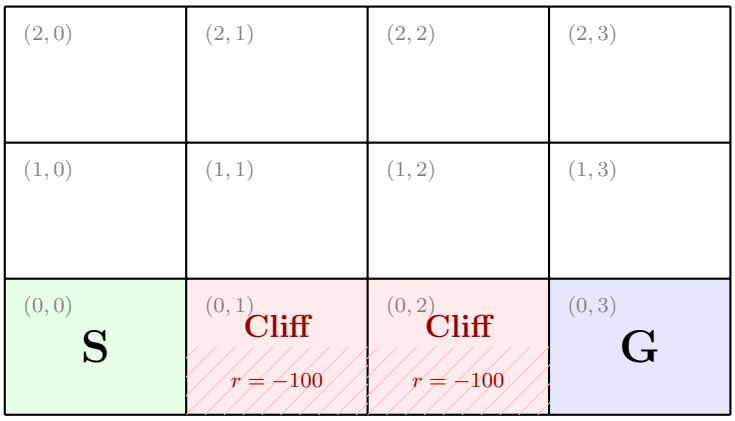
Briefly justify your reasoning.

Solution

Your solution here...

## Question 3: Thinking through a SARSA, Expected SARSA, and Q-Learning Update (15 points)

Consider a simplified $3 \times 4$ CliffWalking grid. The agent starts at $S$ (bottom-left), the goal is $G$ (bottom-right), and the bottom-middle cells are cliff. Actions are $\mathcal{A} = \{U, D, L, R\}$. Stepping off the grid leaves the agent in place.




All other moves: $r = -1$

Cliff: $r = -100$, reset to $S$

Goal: $r = 0$, episode ends

Off-grid: stay in place

The rewards and transitions are annotated in the diagram above. Transitions are deterministic.

Use $\gamma = 1$, learning rate $\alpha = 0.5$, and the $\epsilon$-greedy behavior policy with $\epsilon = 0.5$ and $|\mathcal{A}| = 4$.

**Initial $Q$-values.** Rather than starting from $Q \equiv 0$, suppose the agent has already done some learning. The non-zero entries are:

| State | $Q(\cdot, U)$ | $Q(\cdot, D)$ | $Q(\cdot, L)$ | $Q(\cdot, R)$ |
|---|---|---|---|---|
| $S = (2, 0)$ | $-3$ | 0 | 0 | 0 |
| $(1, 0)$ | 0 | $-8$ | 0 | $-2$ |
| $(1, 1)$ | 0 | $-100$ | $-2$ | $-1$ |

All entries not shown are 0.

**Trajectory.** The agent executes the following short trajectory before falling off the cliff:

$$S \xrightarrow{U} (1,0) \xrightarrow{R} (1,1) \xrightarrow{D} \text{Cliff}$$

with rewards $-1$, $-1$, $-100$ respectively. *Important:* Use ONLY this trajectory for updates in all three parts of this question, assuming updates in order of steps. Remember, you are learning from experience, this is the only experience you have access to. We assume an end of episode after falling off of the cliff, so no actions or rewards after that (expected value is 0).

**Important:** For SARSA, recall that $a_{t+1}$ is the *next action in the trajectory.* At each step, the next action is the one shown on the next arrow.

**(a)** Perform the **SARSA** updates for each of the three transitions. Show the updated $Q$-value after each step.

Solution

Your solution here... Make sure to show Q(S, U), Q((1,0), R), and Q((1,1), D) after each update.

**(b)** Starting from the *same* initial $Q$-values (not the values from part (a)), perform the **Q-Learning** updates for the same trajectory. Show each update and identify where the updates differ from SARSA.

Solution

Your solution here... Make sure to show Q(S, U), Q((1,0), R), and Q((1,1), D) after each update.

**(c)** Again starting from the *same* initial $Q$-values, perform the **Expected SARSA** updates. Show each update and identify where the updates differ from SARSA and Q-Learning.

Solution

Your solution here... Make sure to show Q(S, U), Q((1,0), R), and Q((1,1), D) after each update.

**(d)** Compare the updated values of $Q(S, U)$ and $Q((1, 0), R)$ across the three algorithms. In 2–3 sentences, explain intuitively why they differ, and relate this to the on-policy vs. off-policy distinction.

Solution

Your solution here...

## Question 4: Bellman Residual (10 points)

In the lecture, we introduced the (optimal) Bellman operator for an infinite horizon MDP with discount factor $\gamma$ and transition $p$:

$$(\mathbb{B}V)(s) = \max_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V(s') \right\},$$

and the Bellman operator with respect to a certain policy $\pi$:

$$(\mathbb{B}^\pi V)(s) = \sum_{a \in \mathcal{A}} r(s,a)\pi(a|s) + \gamma \sum_{s' \in \mathcal{S}, a \in \mathcal{A}} p(s'|s,a)\pi(a|s)V(s').$$

We denote the optimal policy by $\pi^*$ and the optimal value function by $V^*$. As we know from the lecture, learning $V^*$ is equivalent to solving the following Bellman equation:

$$V(s) - \mathbb{B}V(s) = 0, \forall s \in \mathcal{S}.$$

For an arbitrary function $V : \mathcal{S} \to \mathbb{R}$, define the Bellman residual to be $(\mathbb{B}V - V)$, and its magnitude by $\|\mathbb{B}V - V\|_\infty$. Recall that for a vector $x = (x_i)_{i \in [d]}$, $\|\cdot\|_\infty$ is defined by $\max_{i \in [d]} |x_i|$. As we will see through the course, this **Bellman residual is an important component of several important RL algorithms such as the Deep Q-network.**

## Question 4.a

Prove the following statements for an arbitrary $V : \mathcal{S} \to \mathbb{R}$ (not necessarily a value function for any policy):

$$\|V - V^\pi\|_\infty \leq \frac{\|V - \mathbb{B}^\pi V\|_\infty}{1-\gamma}, \text{ for any policy } \pi$$
$$\|V - V^*\|_\infty \leq \frac{\|V - \mathbb{B}V\|_\infty}{1-\gamma}.$$

(Hint: use Bellman equation to expand $V^\pi$, then apply triangle inequality.)

Solution

Your solution here...

## Question 4.b

Now let's assume that $\pi$ is the greedy policy extracted from $V$, and assume $\|V - \mathbb{B}V\|_\infty \leq \epsilon$. Prove the following inequality by utilizing the results in Question 4.a:

$$V^* - V^\pi \leq \frac{2\epsilon}{1-\gamma}.$$

This shows that as long as the Bellman residual of $V$ is small, then the policy learned from $V$ will not be too bad.

**Solution**

Your solution here...

# Coding Problems (50 points)

In this question you will implement tabular RL algorithms on two MDPs (typically called "environments" in code) and generate convergence plots.

**Setup:** Install dependencies with `pip install gymnasium numpy matplotlib pytest`. The code files are:

- `hw1.py` — fill in every function marked `TODO`. Do not change function signatures.
- `utils.py` — shared utilities (transition model helpers, plotting). **Do not modify.**

**Submission:** Submit your completed `hw1.py`, the generated plots in `plots/`, and your written answers below. Make sure to include the generated plots relevant to each question in your pdf solution as well as the zip folder containing the code.

## 3.1 Value Iteration on FrozenLake-v1 (25 points)

You will implement three variants of value iteration on the `FrozenLake-v1` environment (4×4 grid, stochastic transitions with `is_slippery=True`, $\gamma = 0.99$). All variants should converge to the same optimal value function $V^*$.

**(coding, 5 pts)** Implement `value_iteration`: standard synchronous VI. Stop when $\max_s |V_{k+1}(s) - V_k(s)| < \theta$.

**(coding, 5 pts)** Implement `gauss_seidel_vi`: in-place (Gauss-Seidel) VI. Same as standard VI except that $V$ is updated in-place, so when computing $Q(s, a)$ the latest values are used for states $s' < s$.

**(coding, 5 pts)** Implement `prioritized_sweeping_vi`: instead of sweeping states in order, sweep in order of highest Bellman error.

**(coding, 3 pts)** Implement `extract_policy`: given a value function $V$, return the greedy policy $\pi$

**(written)** Run `python hw1.py` to generate convergence plots [NOTE: you may get errors until you finish the second half of the implementations, so feel free to comment out code while you're debugging this part. Don't forget to uncomment if you do this though]. Compare the convergence rates of all three VI variants. Which converges fastest in terms of equivalent sweeps? Put the relevant value iteration convergence plots into your solution here.

Solution

Your solution here...

## 3.2 TD Learning on CliffWalking-v0 (25 points)

You will implement three TD learning algorithms on the `CliffWalking-v0` environment (4×12 grid, deterministic transitions, $\gamma = 0.99$, $\epsilon = 0.1$). You will see instructions in the code along with function definitions.

**(coding, 3 pts)** Implement `epsilon_greedy`: with probability $\epsilon$ return a uniformly random action, otherwise return $\arg\max_a Q(s, a)$.

**(coding, 6 pts)** Implement `sarsa`.

**(coding, 6 pts)** Implement `expected_sarsa`.

**(coding, 6 pts)** Implement `q_learning`.

**(written, 4 pts)** Now, run `python hw1.py` to generate the policy arrow plots and reward curves. You're running the code on the CliffWalking example we covered in class.

You should see several plots in the `plots/` directory, including the policy arrow plots and reward curves. Include those plots in your solution along with a writeup of your answers to the following questions:

- Compare the policies learned by Expected SARSA and Q-Learning. Which takes the "safe" path and which takes the "optimal" path? Relate this to the on-policy vs. off-policy distinction.
- Notice that the plotted reward curves (`td_convergence_cliffwalking_v0.png`) for Q-learning include a "behavioral" and a "target" curve. What is the difference between the two? Why is the target curve stable and better than all other curves (what is not happening there that is happening in others)?
- One of the generated plots is a "sweep" over learning rates. That is, we try a bunch of different learning rates that we pass to the algorithms. Using the learning-rate sweep plot (`alpha_sweep_cliffwalking.png`), compare how sensitive each algorithm is to the choice of $\alpha$. You should see one algorithm fall apart at large learning rates? Which one is it and why?

Solution

INCLUDE ALL THE PLOTS GENERATED FOR THIS QUESTION IN YOUR SOLUTION HERE IN ADDITION TO A WRITTEN RESPONSE TO THE QUESTIONS ABOVE.