Assignment 4: Reinforcement Learning
Human and Machine Learning
Chiba Institute of Technology, School of Design & Science
Prof. Joseph Austerweil
Due Fri Jul 10, 2026 at 8:00pm

In this problem set you will implement the reinforcement-learning algorithm *Q-learning*, and then use it to see one of the central lessons of RL: an agent optimizes the *reward you write*, which is not always the *goal you meant*. We work in the "GardenPath" domain (Ho et al., 2015). The goal of the domain is to teach an agent to reach the top-right square (the "goal") by moving along the left-most and top-most squares (the "path") while avoiding the bottom-right 2×2 block (the "garden"). Figure 1 shows the domain. A person sees the picture, but the learning algorithm only knows that there is a 3×3 world it can move around in, receiving feedback as it goes.

Figure 1: The GardenPath domain. Agents learn to reach the goal by moving along the path and avoiding the garden.

The goal of reinforcement learning is to learn a *policy* $\pi : S \to A$ telling the agent which action $a \in A$ to take in each state $s \in S$ so as to maximize its reward. Q-learning does this by learning a matrix Q that stores the expected future reward of taking an action in a state. It is guaranteed to find the optimal policy for a Markov decision process. The algorithm is:

1. Initialize a $|S| \times |A|$ matrix Q to some value (usually zero).
2. For each training episode, start at the initial state and set $t = 1$.
   - 2a. Select an action with an ϵ-greedy rule: with probability $1 - \epsilon$ pick a best action in the current state according to Q (break ties at random), and with probability ϵ pick a random valid action.
   - 2b. Take that action, move to a new state s_{t+1}, and receive feedback f_{t+1}.
   - 2c. With learning rate α and discount factor γ, update Q:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \Big(\underbrace{f_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)}_{\text{prediction error}} \Big) \qquad (1)$$

Intuitively, the update moves $Q(s_t, a_t)$ toward the reward just received plus the best reward expected from there on.

Two ways to give feedback. The point of the assignment is to compare two feedback schemes on this domain (Ho et al., 2015).

- **Reward-maximizing (RM)** — *outcome-based* feedback: moving into the garden is punished (-10), reaching the goal pays $+20$, and ordinary moves along the path are neutral (0). This is the reward you would write if you cared only about the outcome.
- **Action-feedback (AF)** — the way *people* tend to teach: forward progress along the path is praised ($+10$), and — being encouraging — *backtracking* along the path earns faint praise ($+4$) rather than punishment. Both directions are positive.

You can also experiment with both schemes, step by step, in the interactive Q-learning widget from textbook Ch 22 (Q-Learning) (it uses this exact GardenPath world).

Stencils. Three starter stencils are provided in this directory; pick one and submit your completed version. Each ships with a shared helper, `rl_gridworld.py` (the world, the reward schemes, the figure renderer, and an in-notebook interactive explorer); keep it next to your notebook.

- `rl_genjax.ipynb` — GenJAX (canonical, runs in Google Colab). The environment is written as a `@gen` generative model and Q-learning learns by *sampling* it.
- `rl_python.ipynb` — numpy + matplotlib (Python without GenJAX).
- `rl_nosoln.Rmd` — base R + ggplot2 (self-contained; static figures).

A Matlab stencil is available on request — email Prof. Austerweil.

Prep reading. Textbook Tutorial 3 — Ch 21 (Markov Decision Processes) and Ch 22 (Q-Learning) — which walk through everything this assignment asks you to implement. Both chapters embed the interactive Q-learning widget (open it fullscreen to play with α, γ, ϵ and the feedback schemes). This assignment uses the same GardenPath world and the same figures, so the chapters are the fastest way in.

1. Implement Q-learning

In your stencil, fill in the one-line *prediction error* from Equation (1) (the term under the brace; Ch 22 (Q-Learning) steps through it one stage at a time if you want to see each piece). All the setup — the world, ϵ-greedy selection, the training loop, and the figures — is provided.

Then train under **reward-maximizing (RM)** feedback and visualize the learned policy with the provided renderer (it draws, for each cell, the four action-values as green/red wedges, the greedy policy as arrows, and the route the greedy policy takes from the start). *Deliverable:* include the RM figure and a sentence — does the learned policy reach the goal, and does it make sense?

2. Diagnose a reward-hacking failure

Now train under **action-feedback (AF)** feedback and visualize the learned policy. *Deliverable:* the AF figure, the **net reward per lap** of the loop (the renderer / `greedy_rollout` reports it), and a few sentences:

(a) Does the learned greedy policy reach the goal? If not, what does it do?

(b) Why does a reward-maximizing agent prefer to farm that positive cycle forever rather than collect the one-time +20 at the goal?

(c) What does this show about the relationship between the reward you write and the goal you intend? (This phenomenon is called *reward hacking* or *specification gaming*.)

Ch 22 (Q-Learning)'s "Reward Shaping and Positive Cycles" section discusses this failure if you want to compare your findings.

3. Design and verify a fix

AF tried to give helpful per-step feedback but accidentally created a reward cycle. The principled way to add dense feedback is *potential-based shaping* (Ng et al., 1999): choose a **potential** $\Phi(s)$ (a rough "how good is this state") and add a potential *difference* to the true (RM) reward,

$$r'(s,a) \;=\; r_{\mathrm{RM}}(s,a) \;+\; \gamma\,\Phi(s') - \Phi(s). \tag{2}$$

Because it is a *difference* of potentials, a step toward the goal earns a bonus and a step away pays an equal penalty — so there is no farmable cycle.

(a) *Design.* Choose your own potential $\Phi(s)$ (a good default is the negative distance to the goal, so states nearer home have higher potential — but try your own and justify it). Build the shaped reward (2), train, and visualize. *Deliverable:* the figure — does it now reach the goal?

(b) *Verify it is safe (invariance).* Shaping should fix learning *without* changing which policy is optimal. Using the provided value-iteration helper, compute the exact optimal policy for RM and for shaped-RM and confirm they are identical.

(c) *Explain.* In a few sentences, explain why $\gamma\Phi(s') - \Phi(s)$ cannot change the optimal policy (hint: sum it along a trajectory — what telescopes?), and contrast this with AF: both add dense feedback, but only one changed the optimum. Why?

Bonus (+5)

Q-learning is *model-free* — it never sees the transition rules, it learns only from samples. If we *know* the model (Ch 21 (Markov Decision Processes)), we can compute the optimal policy exactly with *value iteration*. Using the provided helper, confirm your model-free learner found the same route to the goal that the model-based planner computes exactly.

(In the GenJAX stencil, the value-iteration step reads the same `@gen` transition model you sampled to learn.)

Submission. Submit by DM or email to the instructor *one* of the following:

- your completed notebook (or knitted `.Rmd`) — it must run end-to-end and contain your figures, inline text answers, and descriptions; *or*
- a single PDF report containing your code, figures, text answers, and descriptions.

References

Ho, M. H., Littman, M. L., Cushman, F., and Austerweil, J. L. (2015). Teaching with rewards and punishments: Reinforcement or communication? In Noelle, D. C., Dale, R., Warlaumont, A. S., Yoshimi, J., Matlock, T., Jennings, C. D., and Maglio, P. P., editors, *Proceedings of the 37th Annual Meeting of the Cognitive Science Society*, pages 920–925, Austin, TX. Cognitive Science Society.

Ng, A. Y., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*, pages 278–287.