CS639: Introduction to Reinforcement Learning

Tengyang Xie

Fall 2026

Logistics

- Lectures: 11am-12:15pm, Tue & Thu, MH B2590
- Instructor: Tengyang Xie (tx@cs.wisc.edu)
- TAs: Sarina Xi (sarina.xi@cs.wisc.edu) and Yurun Yuan (yurun_yuan@cs.wisc.edu)

Tengyang Xie

Sarina Xi

Yurun Yuan

Logistics

- Lectures: 11am-12:15pm, Tue & Thu, MH B2590
- Instructor: Tengyang Xie (tx@cs.wisc.edu)
- TAs: Sarina Xi (sarina.xi@cs.wisc.edu) and Yurun Yuan (yurun_yuan@cs.wisc.edu)
- Office hours
  - Instructor (Tengyang Xie): Wed 1-2pm, MH 5530
  - Sarina Xi: Mon 9-11am, MH 5683
  - Yurun Yuan: Fri 9-11am, MH 5683

Logistics

- Course Homepage: https://mdp.sh/teaching/2026fall_cs639
  - All information about this course
- Canvas: https://canvas.wisc.edu/courses/529609
  - Announcements (email as well) + non-published info / materials (e.g., HW solutions)
- Gradescope: https://www.gradescope.com/courses/1382223
  - Homeworks
- Piazza: https://piazza.com/class/mtjb0gfydoz2sn
  - Q&A **Please use Piazza first for questions**
- Anonymous Feedback: link will be posted on Canvas and Piazza

Prerequisites (1/4): Probability & Statistics

The most important prerequisite: almost everything in RL is an expectation, or a sample estimate of one.

Can you answer these?

- What is a random variable?
- How do you compute its expectation and variance?
- What is conditional probability?
- How do you compute $\mathbb{E}[f(X)]$ when X takes finitely many values?
- How do you sample from a discrete distribution?
- Why does the average of many samples converge to the expected value?

Prerequisites (2/4): Linear Algebra, Calculus & Algorithms

Linear algebra

- How do you multiply a matrix by a vector?
- How do you solve $AX = b$, and when is the solution unique?

Calculus

- What is a gradient? Why does moving against it decrease a function?
- What is the chain rule?

Algorithms

- What is recursion? What problem does dynamic programming solve?
- What problem does Dijkstra's (or BFS) algorithm solve?

Prerequisites (3/4): Programming (Python & NumPy)

Python

- Functions, loops, and classes
- Reading data from a file; plotting curves with matplotlib
- How do you set a random seed, and why does it matter?

NumPy

- How would you compute an elementwise product instead of a matrix product?
- How do you take the max / argmax along an axis of an array?
- How would you replace a for-loop with a vectorized operation?

Prerequisites (4/4): What about Deep Learning?

Assignments in the second half of the course use PyTorch — at a basic level only.

We assume an introductory ML course (regression, loss functions, gradient descent). Later in the course you should also be able to

- Explain what a neural network is and what "training" means
- Say what a loss function is and what SGD / Adam do
- In PyTorch: define a small MLP with nn.Module, compute a loss, call backward(), and take an optimizer step

Some warm-up material of PyTorch will be released later.

Coursework & Grading

No project and no final exam. Three kinds of graded work:

- Homework: 40% (four programming-focused homeworks, 10% each; HW0 is required but not graded)
- Six in-class quizzes: 30% (6% each, best 5 of 6 count)
- One in-class midterm, currently scheduled for the last week of October: 30%

Weights change only in your favor; the midterm will not exceed 30%.

Homework is programming-focused: NumPy first, then a provided PyTorch harness. Google Colab is recommended but a laptop CPU is enough.

Students who want to waive the prerequisites and enroll in the class: Please send me (tx@cs.wisc.edu) your answers to HW0 as well as your transcript **by next class (9/8)**.

Quizzes and Midterm

Quizzes (30%)

- About 15 minutes at the start of class, closed book: a hand trace, a concept question, a short derivation or computation
- Six quizzes, announced one or two lectures ahead. Best 5 of 6 count; no make-ups
- Covers the lectures since the previous quiz; together the six replace a final exam

Midterm (30%)

- Currently scheduled for the last week of October; in class, 75 minutes.
- More details before the midterm

Homework

HW0 – out today, due Fri 09/11

- Checks your comfort with the prerequisites. Everyone must submit it; it does not count toward the grade
- If parts of it feel unfamiliar, reach out early

Four programming-focused homeworks

- About one every three weeks; dates on Canvas
- Submitted via Gradescope
- They reinforce the lectures and prepare you for the quizzes and the midterm

Late Policy

Slip days

- HW1–HW4 share 4 slip days; at most 2 on any one assignment
- Whole days only: any part of a 24-hour period uses a full slip day (1 hour late = 1 slip day; 25 hours late = 2)
- Applied automatically from the submission timestamp. No request, no reason needed

Hard close

- Every assignment closes 48 hours after its deadline; solutions come out then
- Late time not covered by slip days costs 10% of the maximum per started 24-hour period, counted the same way. After the hard close the score is 0

Exceptions

- Slip days do not apply to HW0, to quizzes (best 5 of 6 instead), or to the midterm
- Documented emergencies, accommodations, religious observance: talk to me; handled outside the slip-day budget

Collaboration and AI Tools

Collaboration on homework

- Discuss ideas and debugging approaches with at most two named classmates. Write every proof, every line of code, and every plot yourself
- Every submission lists collaborators and sources

Generative AI: traffic-light label on each assignment

- **Green:** free use (self-study, extra practice)
- **Yellow:** limited, disclosed use; the default for programming parts. Never the core update rule, proof, or analysis you are asked to produce
- **Red:** no AI. Quizzes, the midterm, and problems marked as individual derivation checks
- Every submission includes an Assistance Statement. You must be able to explain everything you submit

Full text in the syllabus; the university academic-integrity process applies

Materials

Companion text: Sutton & Barto, Reinforcement Learning: An Introduction (2nd ed., free online)

Optional, deeper: Agarwal, Brantley, Jiang, Kakade & Sun, RL: Theory and Algorithms; Szepesvári, Algorithms for RL

Notes and the schedule are on the course homepage

Questions: Piazza first (private posts are fine); email only for personal matters

Programming: Python 3, NumPy, matplotlib; PyTorch later in the course. HW0 walks you through the setup

What is this course about?

- A senior / master's-level introduction to RL: making good decisions in sequence, when each decision changes what happens next
- **Focus:** the framework (MDPs) and the core algorithms, from value iteration to RLHF: how each is designed, why it works, when it fails
- **Math:** algorithms are derived on the board; a few central theorems are proved in full, other results are stated without proof and checked on a small example
- **Programming:** in homework you implement the algorithms, run them on small problems, and watch them work, or fail
- **By the end** you should be able to read an RL paper, implement its algorithm on a small problem, and explain why it works or fails
- **Take it if** expectations, matrices, and gradients do not scare you. It is not a deep-RL engineering course, and not a theory course for research

Course roadmap (planned)

Part	Topics
1 Planning under tabular MDPs	MDPs, value functions, Bellman equations, value iteration, policy iteration
2 Learning in MDPs	Monte Carlo, TD, SARSA, Q-learning, importance sampling, linear function approximation
Midterm	In class, last week of October; details before the midterm
3 Function approximation and policy optimization	DQN, policy gradient, actor-critic, PPO
4 Data regimes and outcome-based feedback	Offline RL, exploration, imitation, RLHF / DPO / RLVR, model-based RL and search

No class on Tue 12/08 (NeurIPS). No final exam. Six quizzes, each announced one or two lectures in advance.

Where RL shows up

Games: TD-Gammon (1992), Atari from pixels (2015), AlphaGo and AlphaZero (2016–17)

Robotics and control: locomotion, manipulation, datacenter cooling, plasma control

Recommendation, advertising, inventory and pricing decisions

Science and engineering: AlphaTensor (faster matrix multiplication), chip floorplanning

Language models: learning from human preferences (RLHF), reasoning trained with verifiable rewards

What do these have in common? A sequence of decisions, judged by a score that arrives late.

Which one is it?

Classify each as supervised learning, contextual bandit, planning, or RL. Name one assumption that would change your answer

- **Supervised learning:** you are shown the right answer for every example
- **Contextual bandit:** you see only the payoff of the action you chose
- **Planning:** the rules are known; compute the best policy
- **RL:** rules unknown, you see only your action's payoff, and your action changes what comes next

- (a) Predict whether a customer will cancel next month
- (b) Choose which notification to show; observe only whether it was clicked
- (c) Control a thermostat when the building's heat dynamics are known
- (d) Order inventory each morning; today's order and demand determine tomorrow's stock
- (e) Choose moves in chess to maximize the chance of eventually winning
- (f) Fine-tune a chatbot from thumbs-up / thumbs-down feedback

Before you leave, and next time

Next time, Tue 09/08: Markov decision processes

Reading: Sutton & Barto, Chapter 1 (1.7 optional), Sections 2.1, 2.9, and 3.1–3.3

HW0 is out today, due Fri 09/11. Set up your environment this week

Quizzes are announced one or two lectures in advance