Reinforcement Learning

Guide to AI and Data Courses · Part II: Branches of AI

Agent and environment | Markov decision process | Value functions | Exploration vs. exploitation | Deep RL | Neighboring courses

Sercan KÜLCÜ · Giresun University · Computer Engineering

© Sercan Külcü. All rights reserved. · Interactive HTML version of these slides:
https://sercankulcu.github.io/files/ai_courses/slides/Chapter_07_Reinforcement_Learning.html

What Question Does It Ask?

> **How can good decisions be learned by trial and error from a reward signal alone?**

Reinforcement learning studies how an agent, interacting with its environment and guided only by a **reward signal**, learns by trial and error the behavior that maximizes total reward in the long run. The right action is never given in advance, and the effects of actions often arrive with a delay.

At a glance

APPROACH	INPUT	OUTPUT
trial and error with rewards	environment: state, reward	policy: state → action

SUCCESS MEASURE	MATH LOAD	DATA NEEDS
total expected reward	high: probability, dynamic programming	generated through interaction

A Short History

1957
Bellman's Dynamic Programming introduces the Bellman equation and the idea of a value function.

1988
Sutton publishes temporal-difference (TD) learning: correcting a prediction with the next prediction.

1989
Watkins proposes Q-learning in his PhD thesis.

1992
Tesauro's TD-Gammon nears expert level at backgammon by playing against itself.

2015
DeepMind's deep Q-network (DQN) learns Atari games from raw pixels (Nature).

2016
AlphaGo beats Lee Sedol at Go: tree search, deep networks and reinforcement learning together.

THE COURSE

Core Concepts

Agent and environment
The agent acts; the environment returns a new state and a reward.

Markov decision process
States, actions, transition probabilities and rewards.

Return
The sum of discounted future rewards.

Policy
A rule that says which action to choose in each state.

Value function
The expected return from a state or an action.

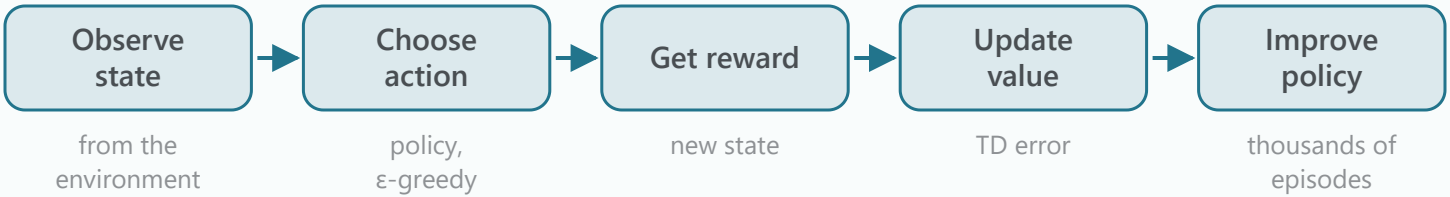
Exploration vs. exploitation
Balancing trying new actions against using what is known.

Temporal difference
TD, Q-learning, SARSA: model-free, step-by-step learning.

Policy gradient
Improving the policy directly with the gradient of the return.

THE COURSE

A Typical Workflow



Example: A warehouse robot observes its position on a grid, usually picks the best direction it knows and sometimes a random one, and gets +10 for reaching the shelf and −1 for hitting a wall. Q-learning updates action values at every step; after thousands of trials a short, safe path emerges.

THE COURSE

What Does the Course Cover?

1. Multi-armed bandits and exploration
2. Markov decision processes and Bellman equations
3. Dynamic programming: policy and value iteration
4. Monte Carlo methods
5. Temporal-difference learning: TD(0) and SARSA
6. Q-learning and off-policy learning
7. n-step methods and eligibility traces
8. Value estimation with function approximation
9. Deep Q-networks: experience replay, target network
10. Policy gradients and REINFORCE
11. Actor-critic methods and PPO
12. Model-based RL and Monte Carlo tree search

Skills you will gain

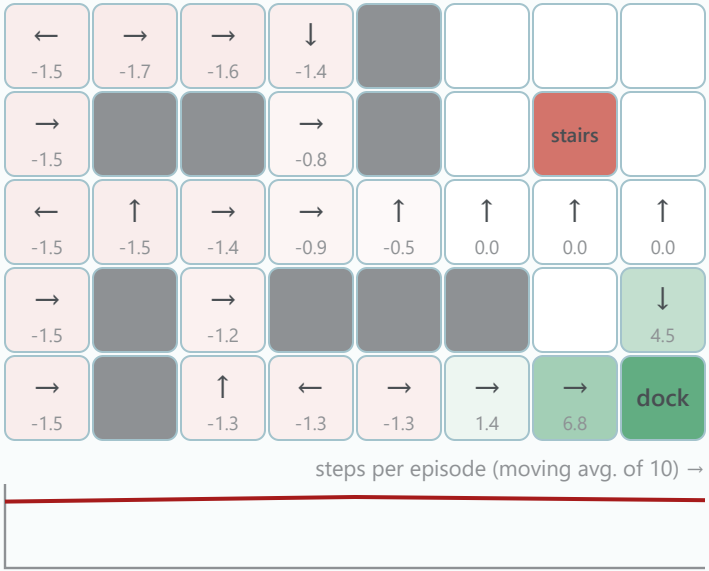
- building environments with Gymnasium
- writing Q-learning from scratch
- training agents with Stable-Baselines3
- designing reward functions
- running and reporting multi-seed experiments

A typical semester; the order and coverage vary between universities.

TRY IT

The Vacuum Finds Its Charger

The robot vacuum has no map; each step is −1, the charging dock +10, falling down the stairs −10. Let it learn by trial over episodes: train it, then send the robot from the top-left corner.



Train 10 episodes | Train 100 episodes

Send the robot | Reset

Exploration rate ε = **0.20**

3 episodes in total. Last 10 episodes: **80.7** steps on average, 0 falls down the stairs.

From the top left, the arrows' path does not reach the dock yet: train more.

Nobody showed the robot the way: after each step Q-learning updates the value of the action in that cell with **reward + best future value**. With no exploration no new paths are found; with too much the robot falls often.

CONTEXT

Prerequisites

Courses worth knowing before this one, and why they are needed.

Course	Why it is needed
Machine Learning	learning from rewards instead of labels
Algorithms	dynamic programming and shortest paths in graphs
Programming	simulation loops and arrays in Python

CONTEXT

Are You Ready for This Course?

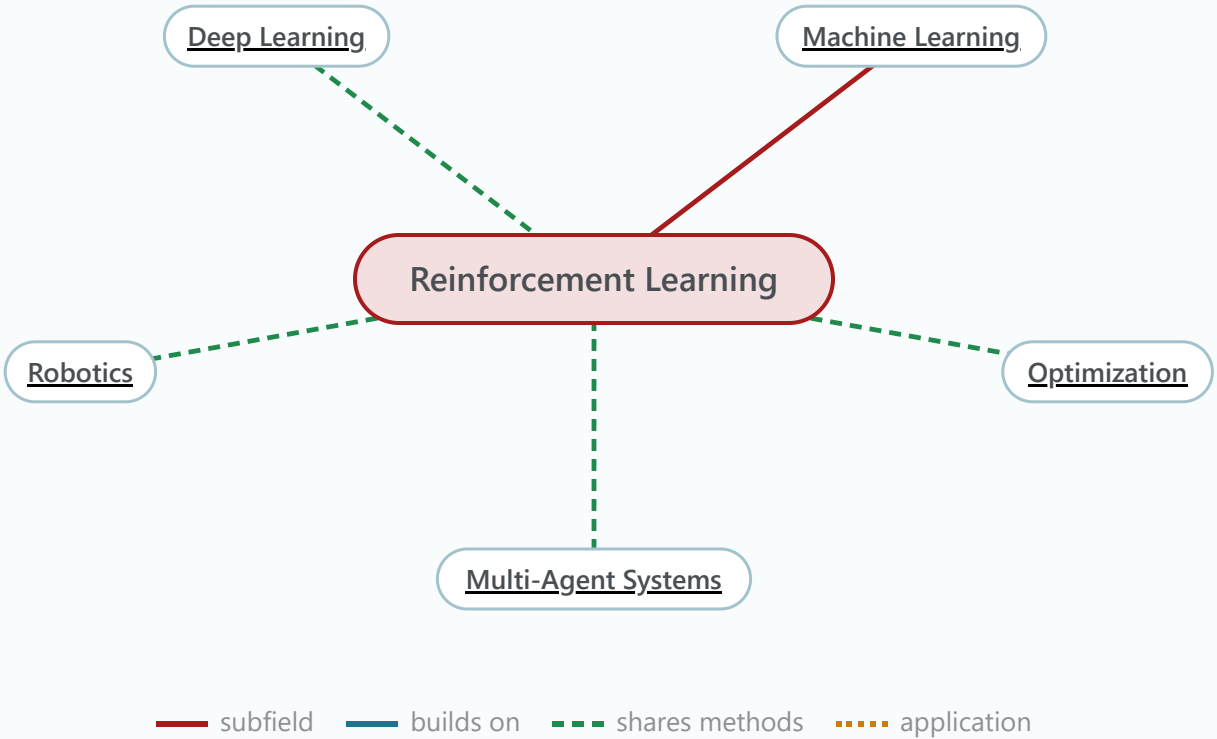
Ask yourself; for every "no", the right-hand column shows where to start.

Ask yourself	If not, start with
Can you compute an expected value and a conditional probability?	Probability and Statistics
Can you solve a shortest-path problem with dynamic programming?	Dynamic programming from an algorithms course
Can you explain a loss function and gradient descent?	Machine Learning
Can you write a simulation loop in Python and plot the result?	An intro programming course, then a short Matplotlib tutorial

If you can say "yes" to most of these, you are ready; one or two gaps can be closed during the semester.

CONTEXT

Neighbouring Courses



Overlapping Courses

Relation	Course	Shared ground
subfield	This course is a subfield of Machine Learning	learning from rewards instead of labels
shares methods	Shares methods with Deep Learning	deep Q-networks, deep policy networks
shares methods	Shares methods with Multi-Agent Systems	multi-agent reinforcement learning, game theory
shares methods	Shares methods with Robotics and Robot Vision	control, learning motor skills
shares methods	Shares methods with Optimization	dynamic programming, policy gradient

Where It Differs (1/2)

Criterion	Reinforcement Learning	Machine Learning	Optimization
Core question	How can good decisions be learned by trial and error from a reward signal alone?	How does a program generalize from examples?	How do we find the values that make an objective function best?
Approach	trial and error with rewards	statistical learning	mathematical model of objective and constraints
Input	environment: state, reward	labeled or unlabeled examples	objective function and constraints
Output	policy: state → action	a model that makes predictions	optimal or near-optimal solution
Success measure	total expected reward	error on unseen data	objective value, convergence rate
Math load	high: probability, dynamic programming	high: linear algebra, probability, optimization	the theory itself
Data needs	generated through interaction	moderate to large	none

- **Machine Learning:** In supervised learning every example comes with the right answer; in RL there is only a delayed reward, and data is collected by the agent's own actions.
- **Optimization:** Optimization solves a problem whose model is known; RL learns by interacting, without knowing the environment's model.

Where It Differs (2/2)

Criterion	Reinforcement Learning	Multi-Agent Systems	Robotics
Core question	How can good decisions be learned by trial and error from a reward signal alone?	How do several autonomous agents decide together or against each other?	How does a machine sense the physical world, know where it is and move?
Approach	trial and error with rewards	interacting autonomous agents	sense-plan-act loop
Input	environment: state, reward	agents, goals, protocols	sensor measurements
Output	policy: state → action	joint behavior, equilibrium	motor commands, path
Success measure	total expected reward	social welfare, stability	task success, safety
Math load	high: probability, dynamic programming	medium: game theory	high: transformations, control
Data needs	generated through interaction	little; simulation	physical experiments and simulation

- **Multi-Agent Systems:** Multi-agent systems study coordination and conflict among agents; this course mostly focuses on a single agent learning.
- **Robotics:** Robotics covers perception, kinematics and control; RL contributes only learning behavior from experience.

PRACTICE

Applications

Games
Go, chess, Atari and strategy games.

Robot control
Learning to walk, grasp and move.

Resource management
Data center cooling, network traffic routing.

Recommender systems
Caring about long-term user satisfaction.

Traffic signals
Reducing waiting times at intersections.

Language model fine-tuning
Reinforcement learning from human feedback (RLHF).

PRACTICE

Risks and Ethics

Reward hacking

OpenAI's boat-racing agent circled endlessly to hit the same scoring targets instead of finishing the race.

Unsafe exploration

Trial and error is costly in the real world: a robot or a treatment policy can cause harm while it explores.

Results that do not reproduce

The same algorithm gives very different results across random seeds; an advantage reported over a few seeds can mislead.

Shaping the user

A recommendation policy that maximizes clicks can push users toward more extreme content and compulsive use.

In depth: Explainable AI and AI Ethics.

PRACTICE

Tools and Datasets

Tools and libraries

Gymnasium | Stable-Baselines3 | PyTorch | RLlib | MuJoCo | PettingZoo

Classic environments

Gridworld | Multi-armed bandit | CartPole | MountainCar | Atari games (ALE) | MuJoCo control tasks

RL gives you an **environment** instead of a ready-made dataset: the agent produces data by its own actions, so simulation is the course's lab.

PRACTICE

Common Misconceptions

Common belief	In fact
RL is just another name for supervised learning.	The right action is never given; the agent learns only from rewards and the experience it collects.
Writing the reward is easy.	An agent exploits a poorly designed reward with unexpected shortcuts (reward hacking).
A robot that learns in simulation works in reality.	The gap between simulation and the real world (sim-to-real) must be handled separately.
RL is only for games.	Control, resource management and training language models from human feedback also use RL.

NEXT STEPS

After This Course

Courses that build on or extend this one

This course is mostly an end point: it builds on the others.

Related careers

Reinforcement learning engineer | Robotics software engineer | Game AI developer | Researcher

After reinforcement learning you can move to multi-agent systems, where many agents learn together, or to robotics, for machines that learn in the physical world. Deep learning helps with deep RL, and optimization makes policy gradient methods easier to understand.

NEXT STEPS

Further Reading

- Sutton and Barto, *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, 2018.
- Szepesvári, *Algorithms for Reinforcement Learning*, Morgan & Claypool, 2010.
- Bertsekas, *Reinforcement Learning and Optimal Control*, Athena Scientific, 2019.
- Mnih et al., "Human-level control through deep reinforcement learning", *Nature* 518, 2015.

The End

Chapter 7: Reinforcement Learning

← Previous chapter · Map of the field · Next chapter →