ARTIFICIAL INTELLIGENCE AND NEURAL NETWORKS

Lecture 6a – Reinforcement Learning for Sequential Decisions

Chizhi Chris ZHANG

zhangchizhi@ciomp.ac.cn
Advanced Computing and Digital Technology Research Center
University of Chinese Academy of Sciences

Spring 2026

Today's Question

What we are trying to answer

How can a machine learn good behavior when nobody hands it the right action for every moment in advance?

Why this feels different

Earlier lectures mostly learned from existing examples. Reinforcement learning learns from interaction, consequence, and adjustment.

What changes from AI5

AI5 searched through possible plans. AI6 adds learning: the agent improves its future choices by experiencing what its current actions lead to.

From AI5 to AI6

Last time

We looked at search problems where the system tries to find a good arrangement or path among many possibilities.

Today

The system is no longer only searching once. It is acting again and again, learning from what those actions cause.

One sentence

AI5 was about choosing well in a search space. AI6 is about learning to choose better over time.

Why Reinforcement Learning Matters

Some decisions do not come with answer sheets

Driving assistance, game playing, tutoring, dialogue, navigation, and adaptive recommendation all involve sequences of actions whose quality becomes clear only through later outcomes.

Why labels are not enough

There may be no neat table saying what the correct move is at every instant. What we often have is feedback after the fact.

That is why RL matters. It is the AI route for learning from consequences rather than from pre-labeled targets.

A Human Story of Feedback

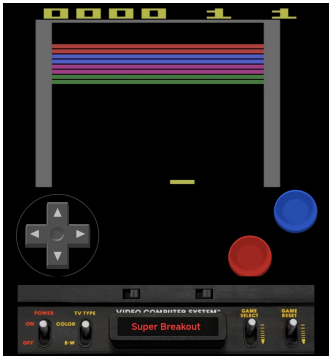
Why this analogy helps

Learning to ride a bicycle or play an instrument rarely works by receiving a perfect label for every tiny movement.

What actually happens

We try something, wobble, adjust, and slowly discover which actions lead to better outcomes.

A Game Story Makes the Idea Visible



Why games are useful in class

One move changes the next screen, the next options, and eventually the score. That makes sequential decision making easy to see.

What the game is really teaching

RL is not about games only. Games are just a clean place to see why a decision now can matter much later.

Why Static Labels Are Not Enough

Three things become harder

- There may be no single obviously correct action right now.
- The reward can arrive long after the important decision.
- The action changes the next state, so the world is reacting to the learner.

Why this is the real shift

Prediction problems ask what the world is like. RL problems ask how our behavior changes what the world becomes next.

Delayed Reward Changes the Whole Story

The difficulty

Some actions feel unrewarding now but open the path to success later. Other actions feel good now but quietly damage the future.

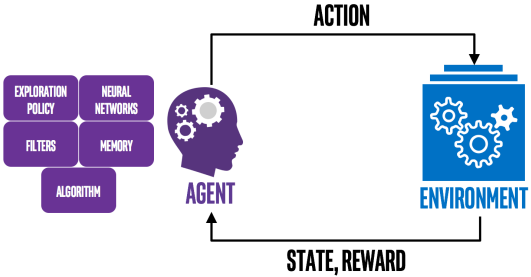
A simple example

In a maze, stepping away from the goal might still be smart if it leads to the only path that reaches the exit later.

The key point

RL must judge actions by their long path, not only by their next tiny payoff.

The Reinforcement Loop



The repeating loop

An agent sees a state, chooses an action, receives reward, and lands in a new state.

Why this picture matters

Almost every RL method is trying to learn how to behave better inside this loop.

The Five Pieces of RL

The vocabulary

- **State**: the current situation
- **Action**: the move the agent makes
- **Reward**: the feedback signal
- **Policy**: the rule for choosing actions
- **Return**: the longer-term total reward

Why this is enough to begin

Once these five ideas are clear, later terms like value function, Q-table, and deep RL stop sounding like unrelated jargon.

Episode, Trajectory, and Goal

Episode

A complete run from some starting point to some ending condition, such as one game, one route, or one full task attempt.

Trajectory

The actual sequence of states, actions, and rewards that happened inside that episode.

These ideas matter because RL is judging a chain of events, not one isolated prediction.

Reward Design Quietly Controls Behavior

Reward is not neutral

The reward tells the agent what is worth repeating. If the reward is narrow, the behavior it learns can become narrow too.

A simple example

If we reward a tutoring system only for keeping students online longer, it may learn a very different behavior from a system rewarded for genuine learning progress.

The warning

An RL agent does not learn what we meant. It learns what the reward actually favors.

How Bad Reward Design Creates Shortcuts

What can go wrong

The agent may find loopholes, over-optimize one easy metric, or discover a behavior that raises reward while missing the intended goal.

Why this is interesting

RL makes the alignment problem very visible. The system is always asking, “what behavior gets rewarded here?”

This is one reason RL is conceptually powerful and practically dangerous at the same time.

Why the Current State Matters

The same action is not always the same decision

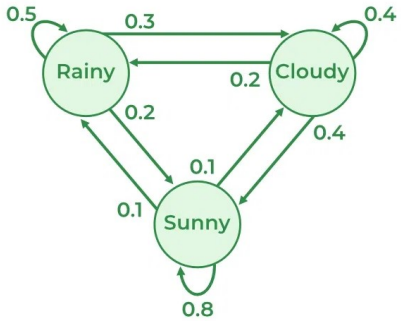
Turning left can be wise in one location and foolish in another because the future possibilities are different.

Why this is deeper than it sounds

RL has to judge actions relative to the situation, not only relative to a general rule like “left is good” or “right is bad.”

That is why state is the first thing the agent has to understand.

A Small Markov Story



Markov intuition

What matters most for the next step is the situation we are in now, because different current states open different future paths.

Why this helps

It gives us a workable way to describe an evolving world without carrying the entire history every time.

Looking Beyond One Step

One useful formula

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots$$

How to read it

The return G_t is the future reward stream, usually with a discount factor γ so rewards farther away count less than immediate ones.

This formula is the mathematical version of one simple idea: a good action is one that leads to a good future, not just a good next second.

Value Means Future Promise

State value

The value of a state means how promising that situation is if we continue from there in a sensible way.

Action value

The value of a state-action pair asks something more specific: if I take this move here, how good is the longer-term future likely to be?

This is why RL often feels like learning a map of future promise.

Why Delayed Credit Is Hard

The real challenge

If success appears only after many steps, we still need some way to assign credit backward to the earlier actions that made it possible.

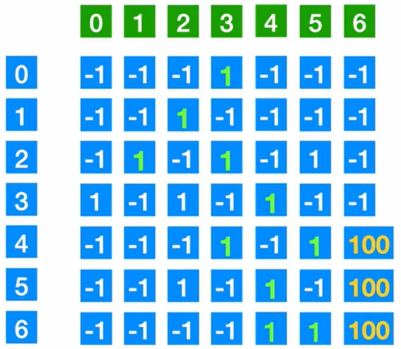
Why this matters

Without some credit-assignment logic, the agent keeps seeing final success or failure without learning which earlier decision truly mattered.

The bridge forward

Value learning is one answer to this problem. It moves useful reward information backward through experience.

Why a Small Reward Table Helps First



	0	1	2	3	4	5	6
0	-1	-1	-1	1	-1	-1	-1
1	-1	-1	1	-1	-1	-1	-1
2	-1	1	-1	1	-1	1	-1
3	1	-1	1	-1	1	-1	-1
4	-1	-1	-1	1	-1	1	100
5	-1	-1	1	-1	1	-1	100
6	-1	-1	-1	-1	1	1	100

Why start with a toy world

In a tiny environment, we can list states and actions clearly and see how value learning works before hiding it inside a large model.

What the table shows

Some actions are useless, some are mildly helpful, and a few are connected to the path that really matters.

What a Q-Table Stores

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0

Q-value idea

For each state-action pair, the table stores an estimate of how promising that move is in the long run.

Why this is useful

The agent does not need the full future written down. It only needs to keep improving these estimates through experience.

The Smallest Useful Update Rule

Key formula

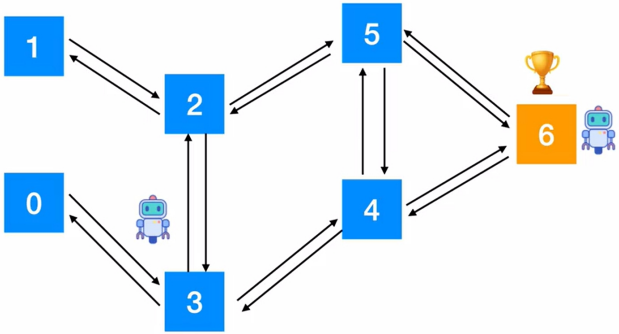
$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big)$$

Read it plainly

Take the old guess, compare it with a better target built from reward plus future value, then move the guess a little toward that target.

The formula matters less than the story: present actions are being judged through future consequences.

A Routing Example



Why routing is a good classroom case

Each move changes location, future options, and eventual path length to the destination.

What the learner discovers

The agent is not just finding the next hop. It is learning which local move tends to point toward a valuable future.

How Values Spread Through Experience

	0	1	2	3	4	5	6
0	-1	-1	-1	0	-1	-1	-1
1	-1	-1	0	-1	-1	-1	-1
2	-1	0	-1	0	-1	0	-1
3	0	-1	0	-1	1	-1	-1
4	-1	-1	-1	0	-1	1	0
5	-1	-1	0	-1	0	-1	100
6	-1	-1	-1	-1	0	0	-1

Earlier stage

	0	1	2	3	4	5	6
0	-1	-1	-1	0	-1	-1	-1
1	-1	-1	0	-1	-1	-1	-1
2	-1	0	-1	0	-1	0	-1
3	0	-1	0	-1	81	-1	-1
4	-1	-1	-1	0	-1	81	100
5	-1	-1	0	-1	0	-1	100
6	-1	-1	-1	-1	0	0	-1

Better paths emerge

	0	1	2	3	4	5	6
0	-1	-1	-1	0	-1	-1	-1
1	-1	-1	0	-1	-1	-1	-1
2	-1	0	-1	0	-1	0	-1
3	0	-1	0	-1	65.8	-1	-1
4	-1	-1	-1	0	-1	81	0
5	-1	-1	0	-1	0	-1	100
6	-1	-1	-1	-1	0	0	-1

More structure appears

	0	1	2	3	4	5	6
0	-1	-1	-1	65.8	-1	-1	-1
1	-1	-1	65.8	-1	-1	-1	-1
2	-1	53.64	-1	65.8	-1	81	-1
3	53.64	-1	65.8	-1	81	-1	-1
4	-1	-1	-1	65.8	-1	81	100
5	-1	-1	65.8	-1	81	-1	100
6	-1	-1	-1	-1	-1	-1	-1

Strong preferences form

How to read the sequence

Value does not stay only at the final rewarding state. As learning repeats, nearby actions that lead toward success also become more valuable. That is how reward information travels backward through time.

Explore or Exploit

Explore

Try actions that may reveal something better than the current habit.

Exploit

Use the action that currently looks most promising according to what the agent has already learned.

The tension

Too much exploration wastes time. Too little exploration can trap the learner in a mediocre routine.

Where You Already Meet RL

Common places

Games, adaptive recommendation, traffic control, tutoring, robotic control, and some dialogue systems all involve repeated action under feedback.

Why this matters

RL is not a niche curiosity. It is the natural language of problems where decisions keep changing the next situation.

Where RL Can Mislead

Three recurring problems

Reward hacking, unsafe exploration, and huge data demand can all make RL look more successful in toy settings than in real ones.

Why students should be skeptical

An impressive reward curve can still hide shallow or unsafe behavior if the task was designed carelessly.

The warning

RL is powerful, but it can be very literal and very expensive.

Why the Real World Is Harder

Toy worlds are forgiving

Mistakes are cheap, feedback is clean, and the agent can try again many times.

Real settings are harsher

Experiments can be expensive, safety matters, human behavior is messy, and rewards may be delayed or ambiguous.

The jump from a classroom grid world to a real social or physical system is much larger than hype often suggests.

Why Tables Stop Scaling

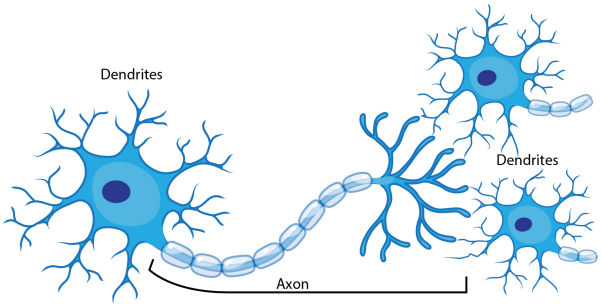
Small worlds fit in a table

If the number of states and actions is tiny, explicit storage is possible and transparent.

Large worlds do not

Images, language, robotics, and real-life interaction generate far too many possible situations to list one by one.

How Neural Networks Meet RL



What changes

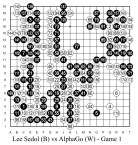
Instead of storing one value for every exact state-action pair, a neural network learns a function that can generalize across similar situations.

Why this matters

This is the bridge from classical tabular RL to deep reinforcement learning.

From Atari to AlphaGo

Atari gave RL many repeated trials



AlphaGo combined search and learning

Why this page matters

The famous RL milestones did not come from one trick alone. They came from combining learning, value estimation, search, large-scale experience, and enough computation to make repeated improvement possible.

Why AI6 Leads to NN6

What AI6 added

We studied actions over time, delayed consequences, and the difficulty of carrying useful information forward through a sequence of decisions.

Why NN6 is next

NN6 stays with the idea of time and sequence, but shifts from choosing actions to processing ordered information such as words, signals, and tokens.

Summary

- Reinforcement learning learns from interaction, where actions change future states and rewards may arrive later.
- The core RL loop is state, action, reward, policy, and long-term return.
- Q-learning works by updating present action values using reward plus an estimate of future value.
- Exploration and exploitation pull in opposite directions and must be balanced.
- RL becomes much harder in the real world because reward design, safety, scale, and delayed feedback are all difficult.

Next Time

Where this story goes next

NN6 stays with time and sequence, but now the problem is not action selection. It is how a neural network remembers and processes information that arrives step by step.

What to watch for

AI6 asked how reward travels through time. NN6 will ask how information travels through time.

Thank You