ECE8803-AI4

AI for Semiconductor Manufacturing

Module / Reinforcement Learning · Lecture 1

Introduction to Reinforcement Learning

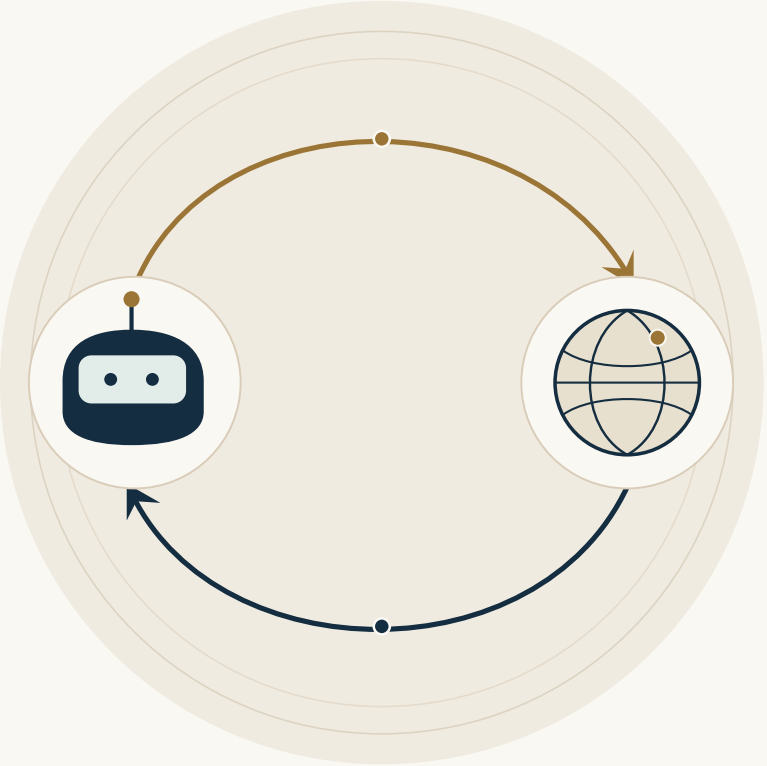
Asif Khan

Associate Professor
School of Electrical and Computer Engineering
School of Materials Science and Engineering
Georgia Institute of Technology

akhan40@gatech.edu

Course website https://aikhan123.github.io/khan-lab-website/ai-semiconductors-course.html

YouTube playlist https://www.youtube.com/watch?v=JOOhPoMm58I&list=PLF-ca21MPQt4

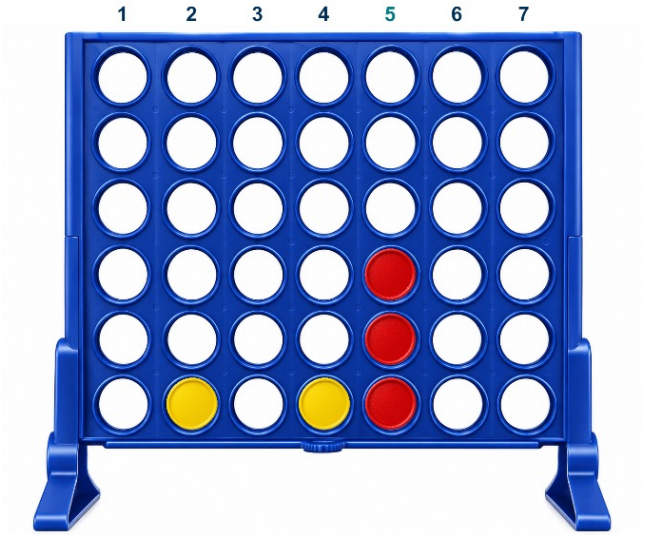


In this lecture

Basics and the formal setup
A worked Connect Four example
TD learning, models and networks
Search, training and policy gradients
AlphaGo, history, and when it works well

What is reinforcement learning?

An agent takes actions and uses rewards, which can arrive several steps later, to learn which earlier choices lead to better outcomes.



What does Yellow learn?

Yellow chooses a column and sees how the game unfolds.

A move can earn 0 immediately and still help Yellow win later.

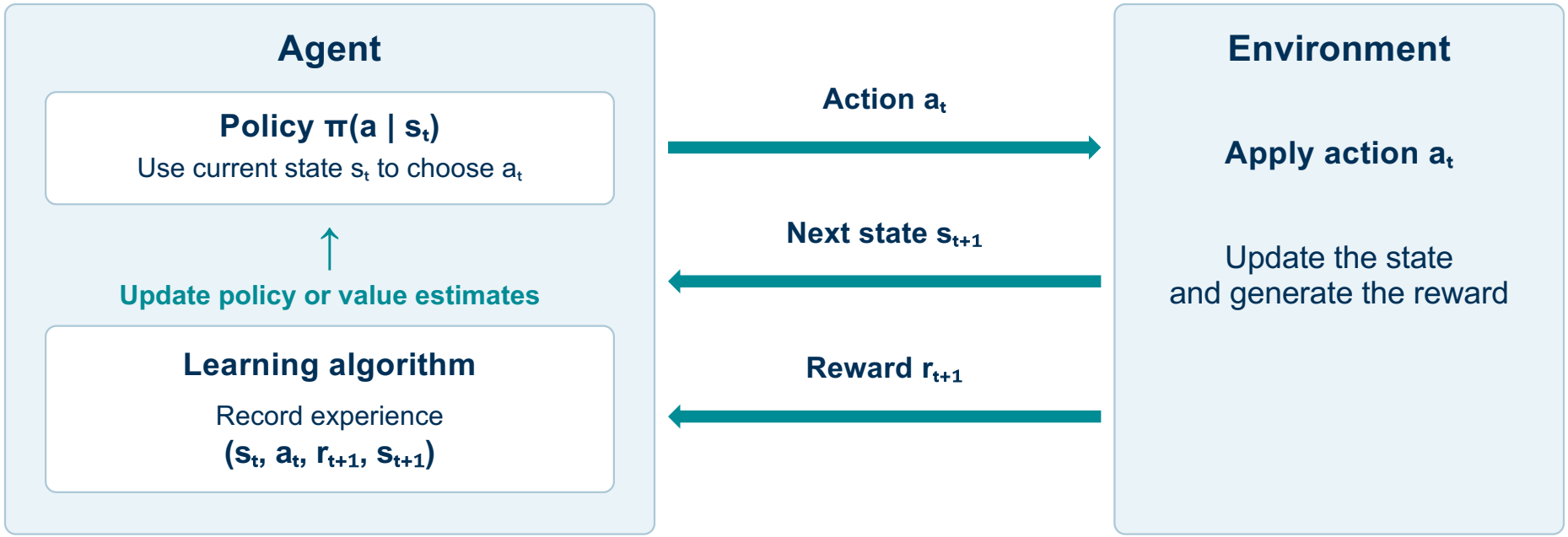
The basics of reinforcement learning

Agent	The learner that makes decisions.
Environment	The system the agent interacts with. It responds to actions and supplies rewards.
State	A description of the current situation. We write the state at step t as s_t.
Action	A choice the agent can make in that situation. We write the action taken at step t as a_t.
Reward	Feedback from the environment that can arrive several steps after a choice. We write the reward received just after action a_t as r_{t+1}.
Return	The sum of future rewards, with discounting if used. We write the return from step t as G_t, starting with reward r_{t+1}.
Episode	One sequence of interaction from a starting state to an ending state.

How an agent decides and learns

Policy	A rule for choosing an action in a state. We write the probability of choosing action a in state s as $\pi(a \mid s)$.
Value and action value	State value estimates the return from a board. Action value specifies the first move. We write the state value as $V(s)$ and the action value as $Q(s, a)$.
Search	The agent simulates possible actions and outcomes before deciding. Search score$(s, a) = Q(s, a) + U(s, a)$, where U is the exploration bonus.
Visit count	The number of times search tried a particular action from a state. We write the number of visits to action a from state s as $N(s, a)$.
Policy training target	The action probabilities we train the policy to predict. $\pi_{search}(a \mid s)$ is the distribution we compute from the search visit counts.
Value training target	The number we train the value prediction to match. In this example, z_t is the final game result from the viewpoint of the player at step t.

The agent–environment loop



A later reward can contribute to the return G_t used to learn about an earlier action.

REINFORCEMENT LEARNING

The goal of reinforcement learning is to learn a policy that maximizes expected return over time

Policy A policy tells the agent how to choose an action in each state.

Expected return Expected return is the total reward over time, averaged over the possible outcomes.

Two things make this hard

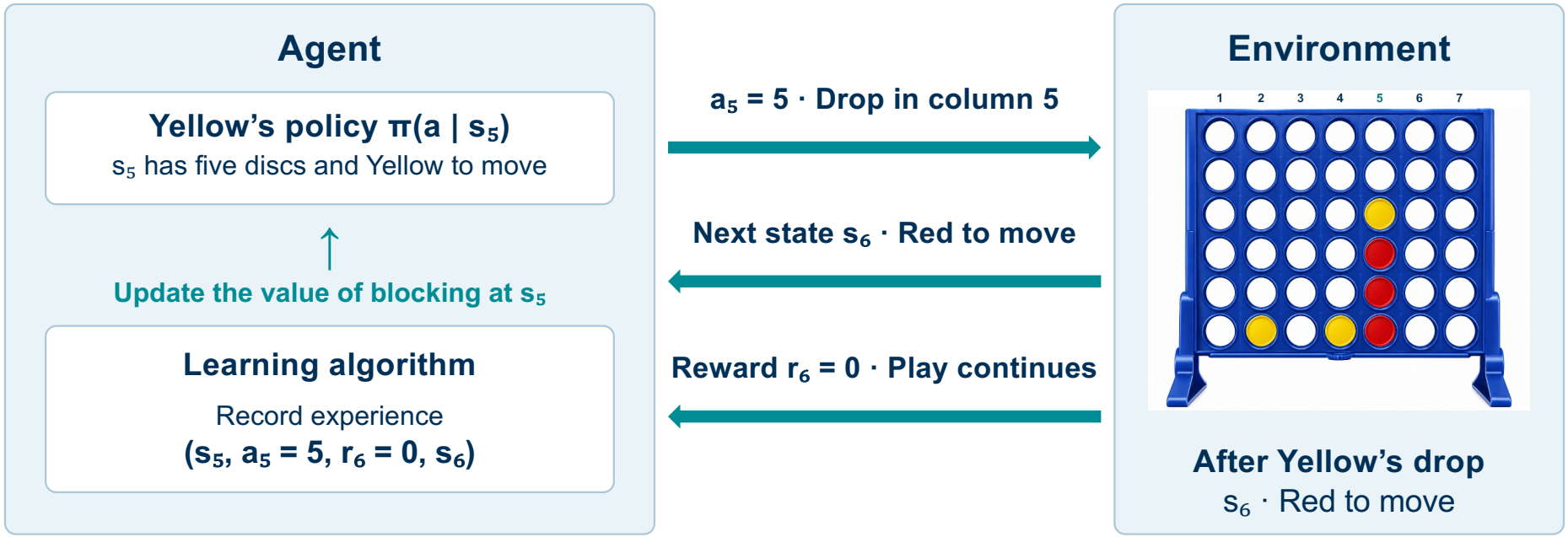
Rewards can arrive later

The agent has to learn which earlier actions helped produce them.

The environment can be hard to predict

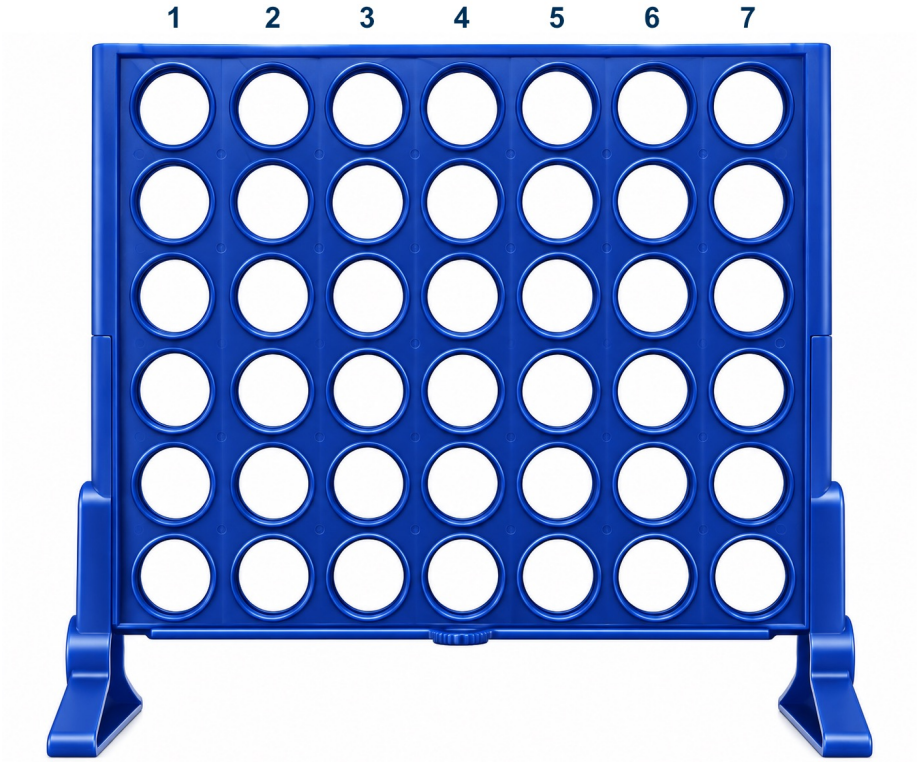
Given a state and an action, the agent may be unsure what state or reward will come next.

The agent–environment loop in Connect Four



Yellow later wins on move 10. Its reward $r_{10} = +1$ contributes to the return G_5 for the block.

Each game starts with an empty board



One disc per turn

Choose a column. The disc falls to the lowest open spot.

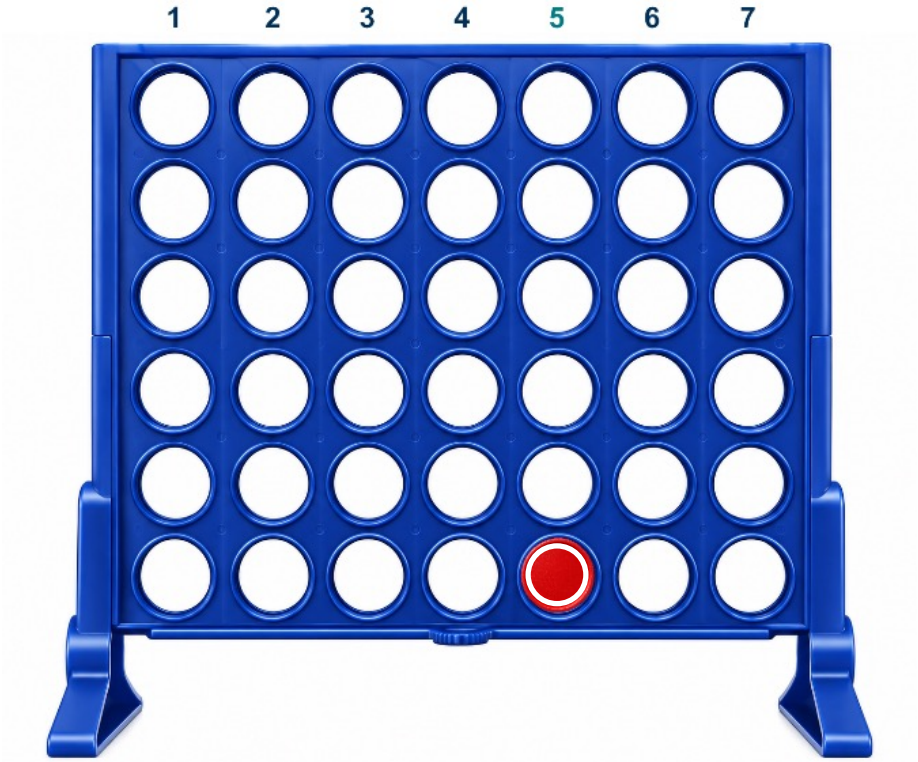
Four in a line wins

The line can be horizontal, vertical, or diagonal.

Red goes first in both games.

CONNECT FOUR / GAME 1 / MOVE 1 OF 7

Red opens the first game

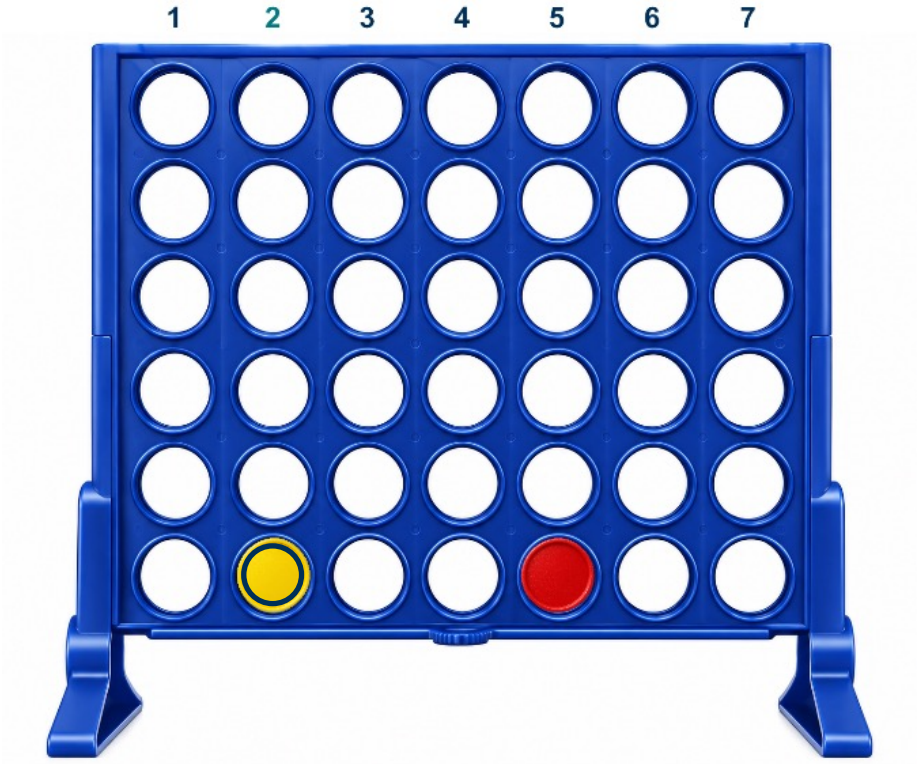


Red plays column 5

The first red disc falls to the bottom of the empty column.

Yellow to move next.

Yellow places its first disc



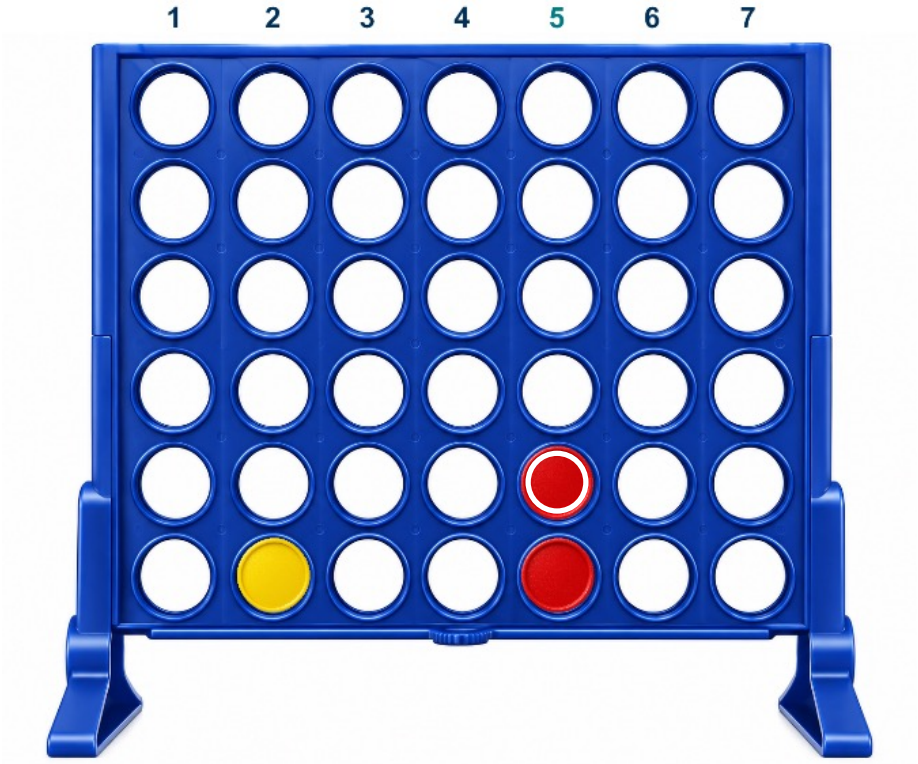
Yellow plays column 2

Yellow opens in column 2. Its disc also falls to the bottom.

Red to move next.

CONNECT FOUR / GAME 1 / MOVE 3 OF 7

Red starts a stack

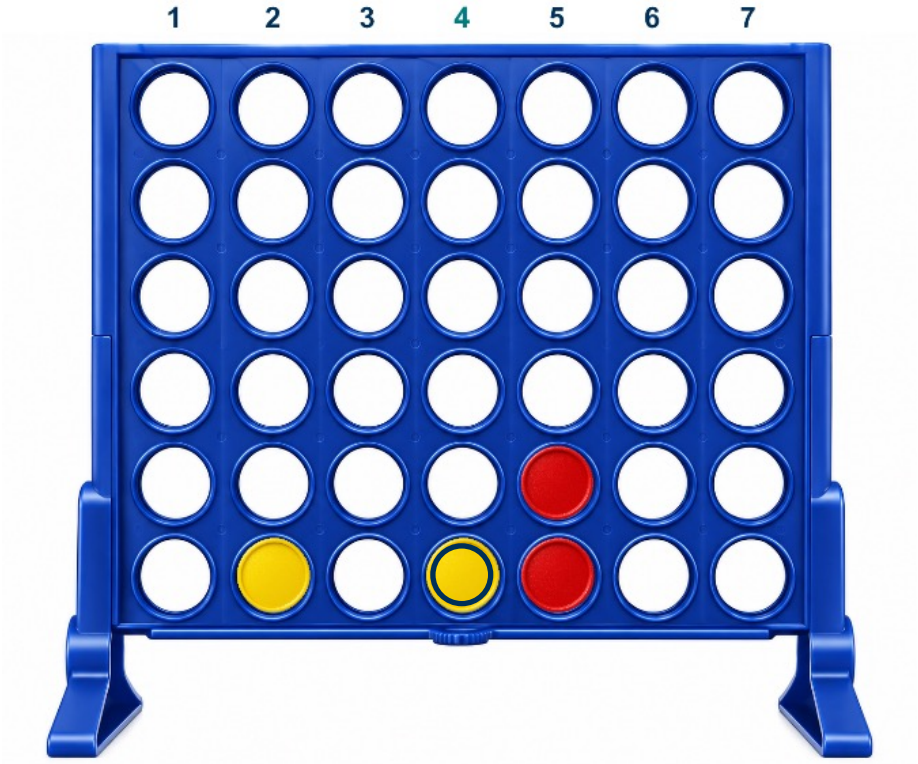


Red plays column 5

Red stacks a second disc directly above the first.

Yellow to move next.

Yellow adds a disc along the bottom

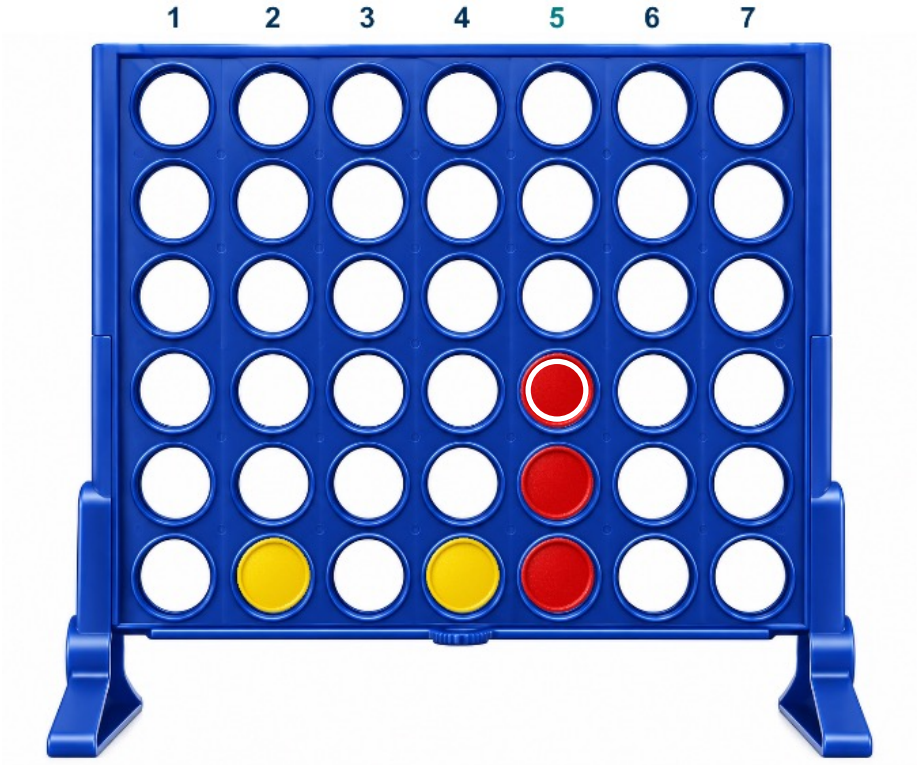


Yellow plays column 4

Yellow places a second disc along the bottom row.

Red to move next.

Red is one disc away from a win

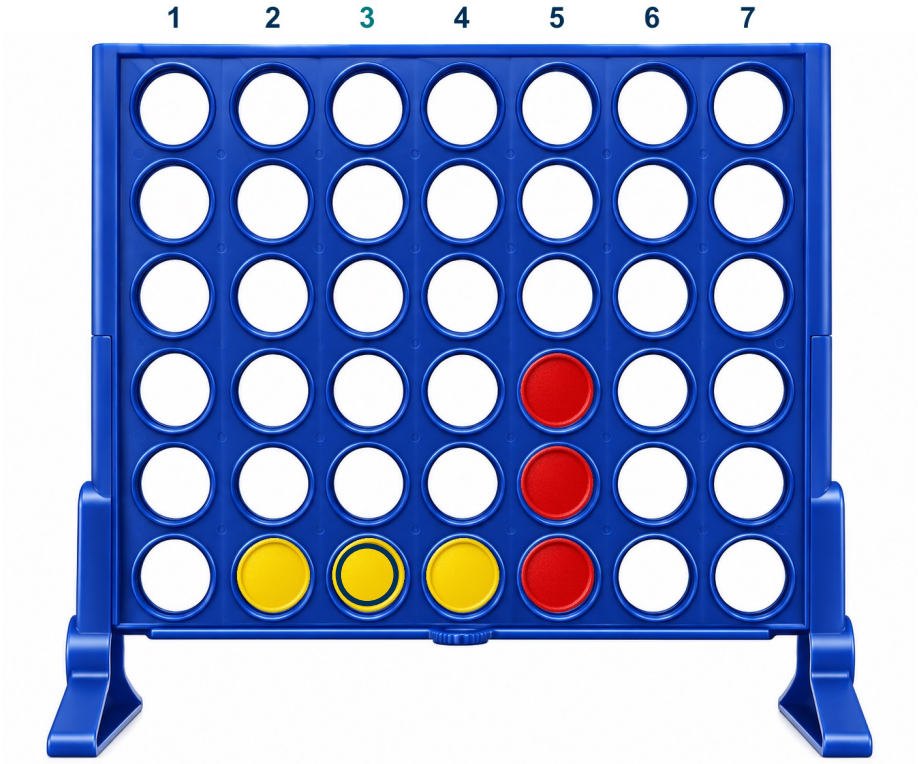


Red plays column 5

Column 5 now holds three red discs.

Which column would you choose as Yellow?

Yellow makes three in a row



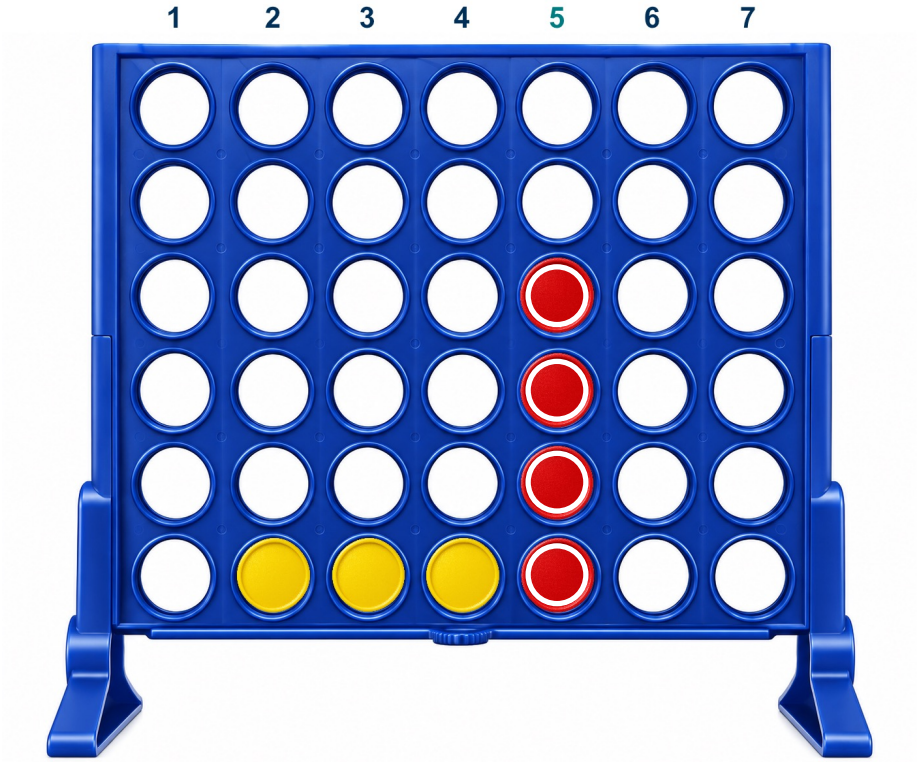
Yellow plays column 3

The new disc fills the gap between Yellow's discs in columns 2 and 4.

Red gets the next move.

CONNECT FOUR / GAME 1 / MOVE 7 OF 7

Red wins with four in column 5

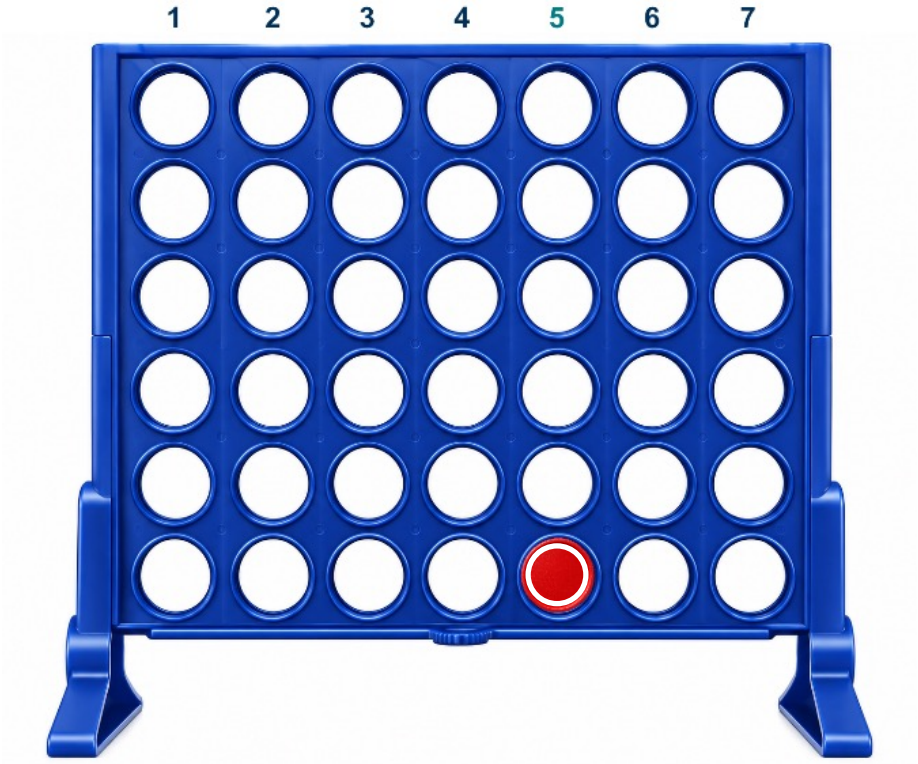


Four in a row

Red drops a fourth disc on its stack and completes a vertical line.

Reward for Yellow: −1

Game 2 starts with the same opening move

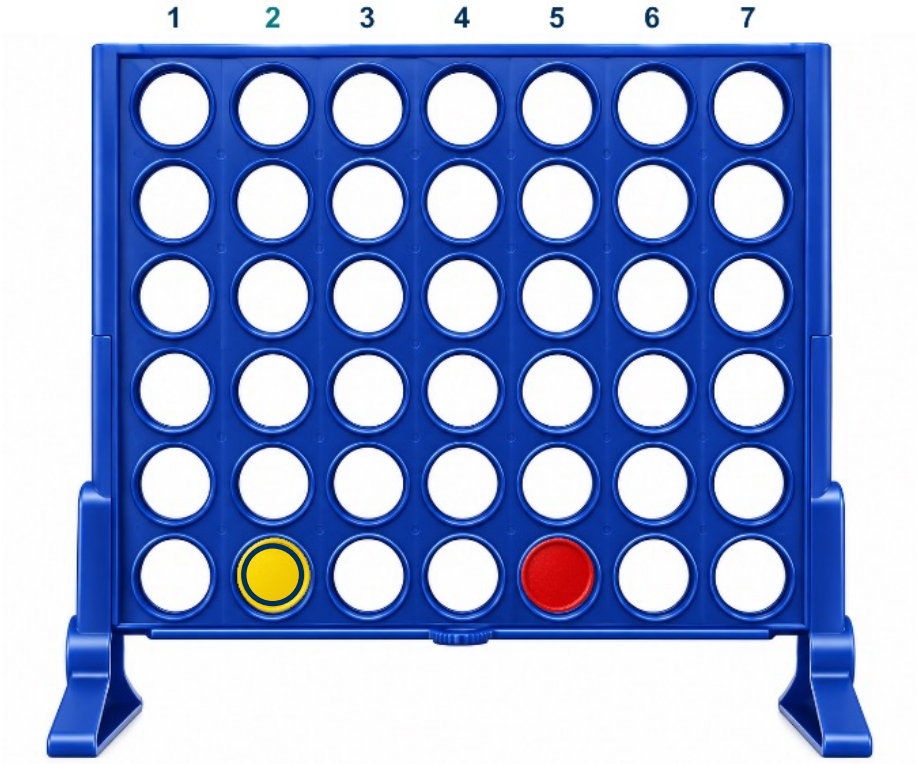


Red plays column 5

The first red disc falls to the bottom of the empty column.

Game 2 restarts from an empty board.

Yellow places its first disc



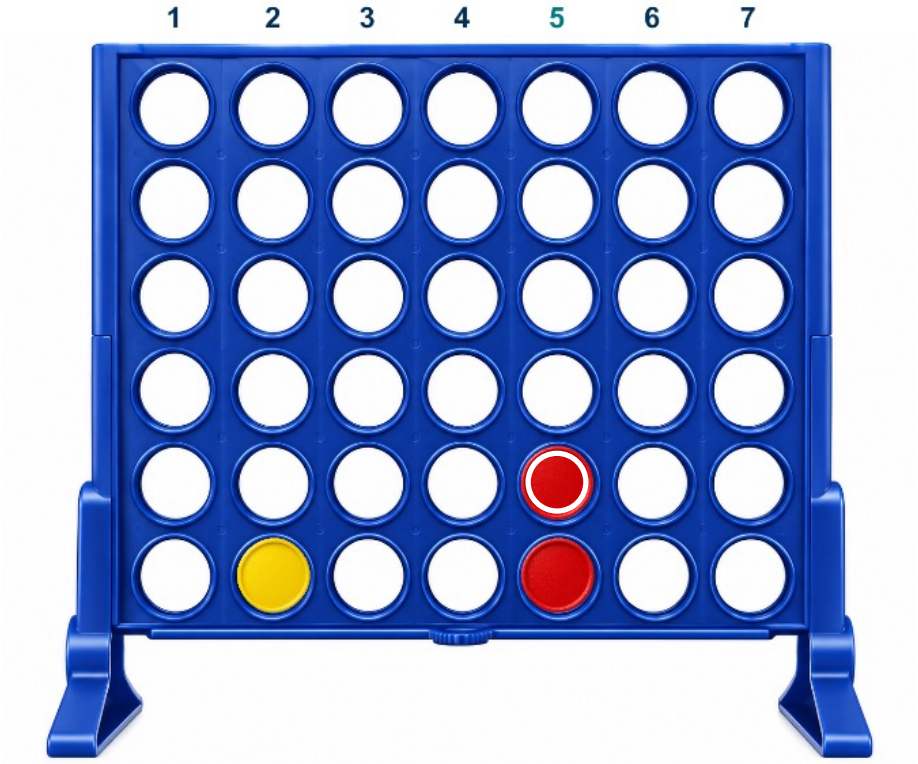
Yellow plays column 2

Yellow opens in column 2. Its disc also falls to the bottom.

Red to move next.

CONNECT FOUR / GAME 2 / MOVE 3 OF 10

Red starts a stack

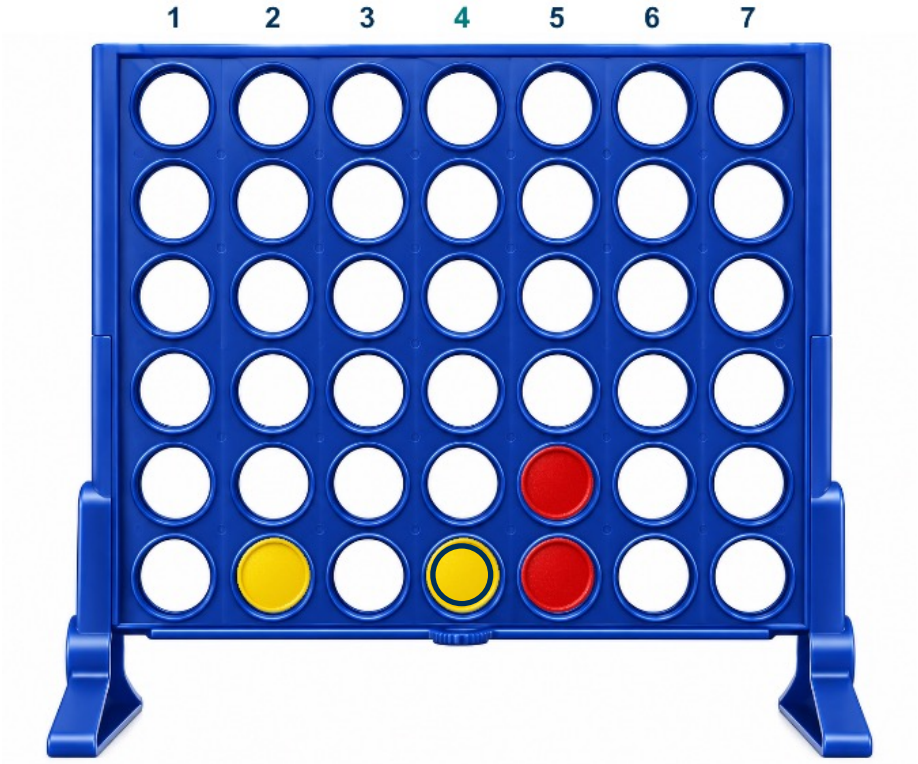


Red plays column 5

Red stacks a second disc directly above the first.

Yellow to move next.

Yellow adds a disc along the bottom



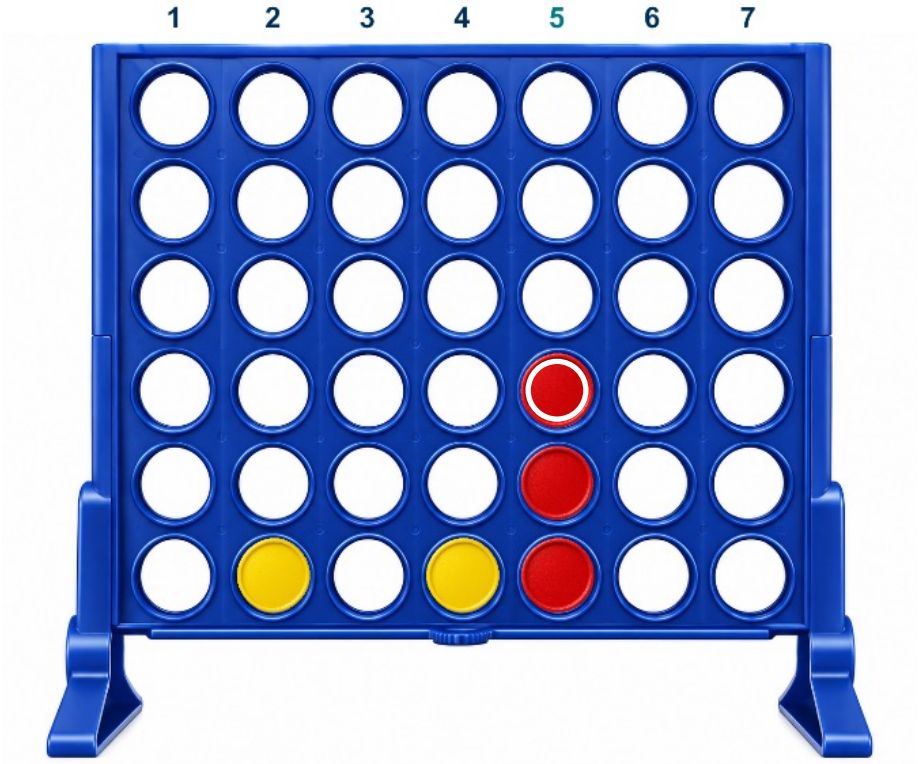
Yellow plays column 4

Yellow places a second disc along the bottom row.

Red to move next.

CONNECT FOUR / GAME 2 / MOVE 5 OF 10

Red is one disc away from a win



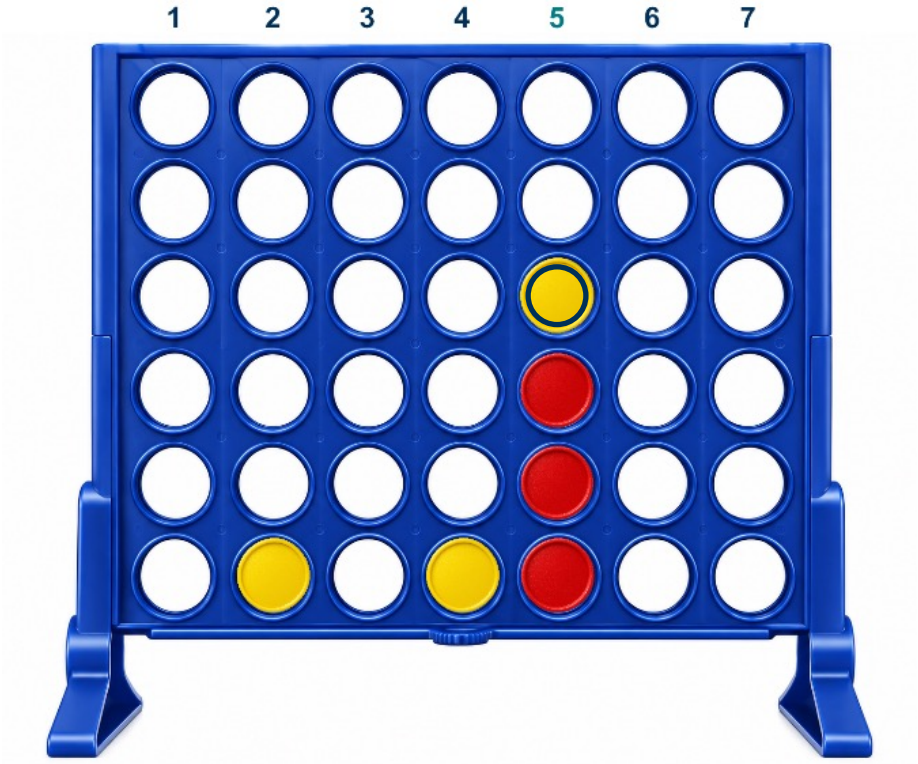
Red plays column 5

Column 5 now holds three red discs.

The first five moves match Game 1.

CONNECT FOUR / GAME 2 / MOVE 6 OF 10

Yellow blocks column 5

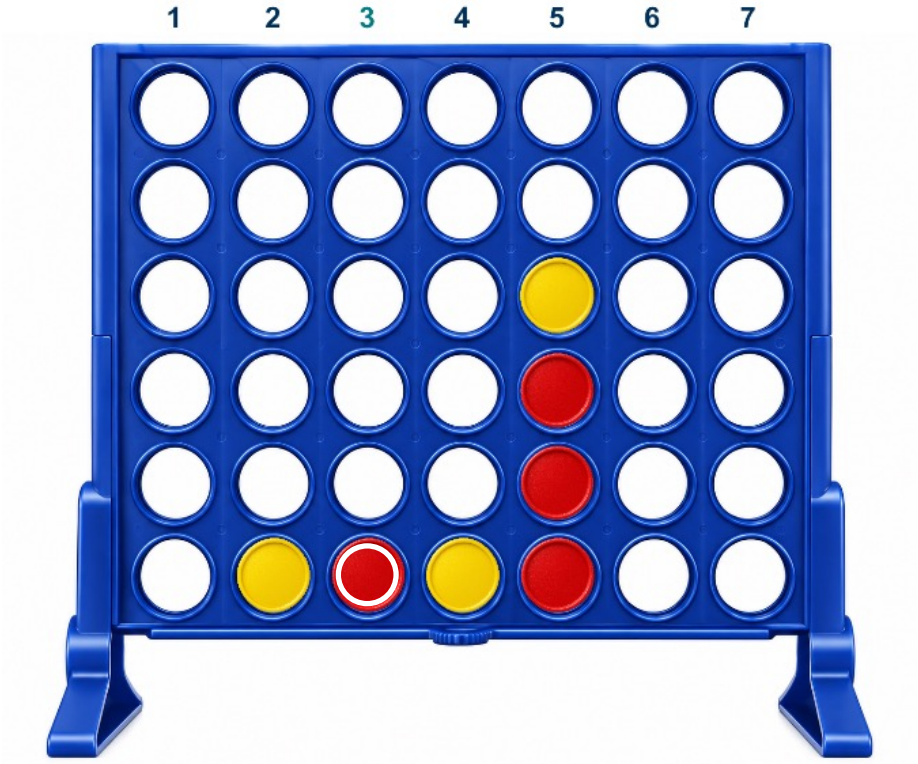


Yellow plays column 5

Yellow takes the open spot above Red's three discs.

Immediate reward for Yellow: 0

Red fills the gap on the bottom row

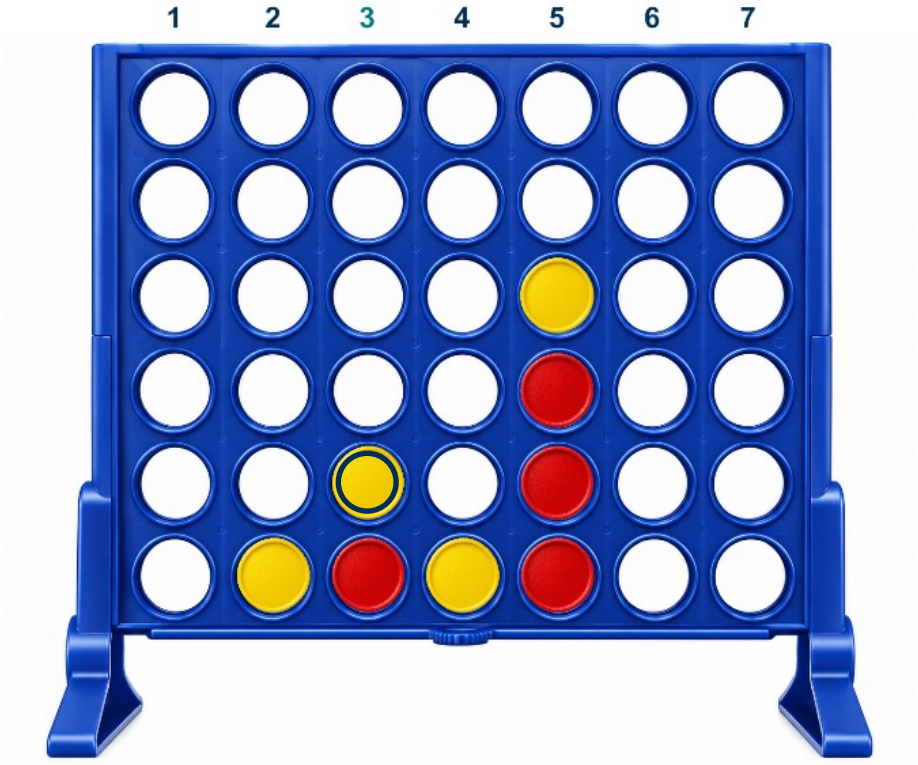


Red plays column 3

Red fills the gap between Yellow's discs along the bottom.

Yellow to move next.

Yellow builds a diagonal

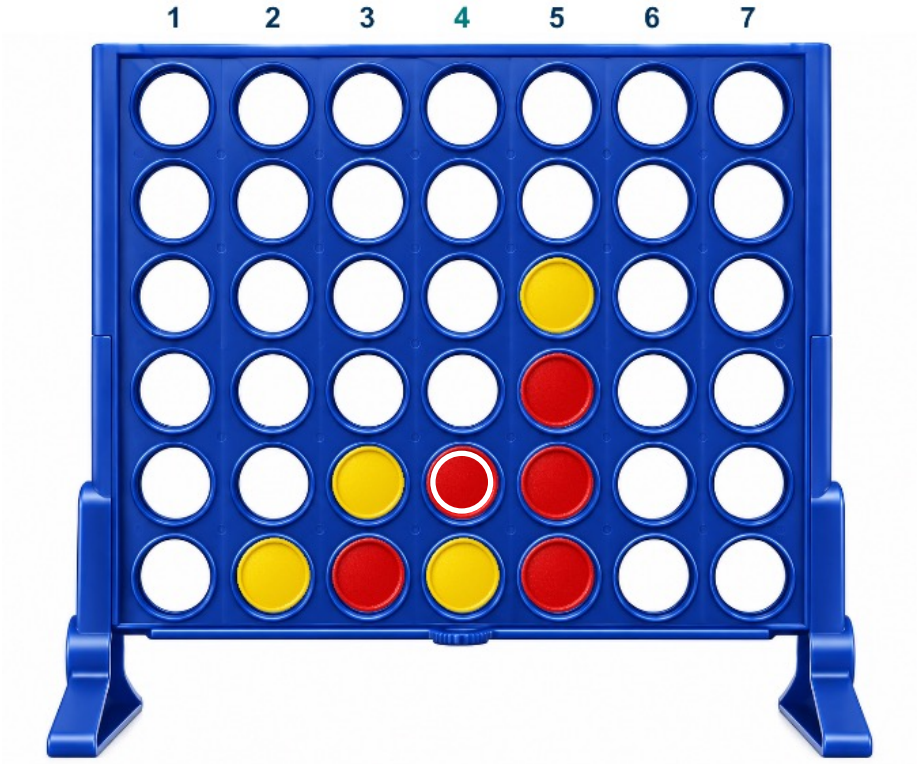


Yellow plays column 3

Yellow now has three on the diagonal from column 2 to column 5.

Red to move next.

Red fills the spot below the gap



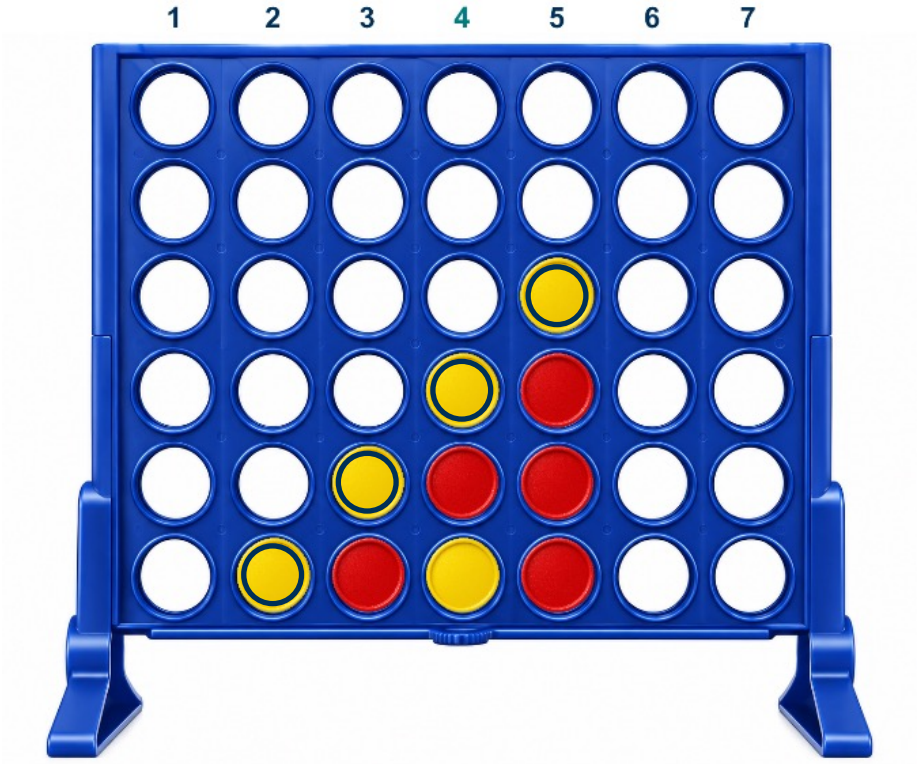
Red plays column 4

The next disc in column 4 can now land on Yellow's diagonal.

Yellow to move next.

CONNECT FOUR / GAME 2 / MOVE 10 OF 10

Yellow wins with four on a diagonal



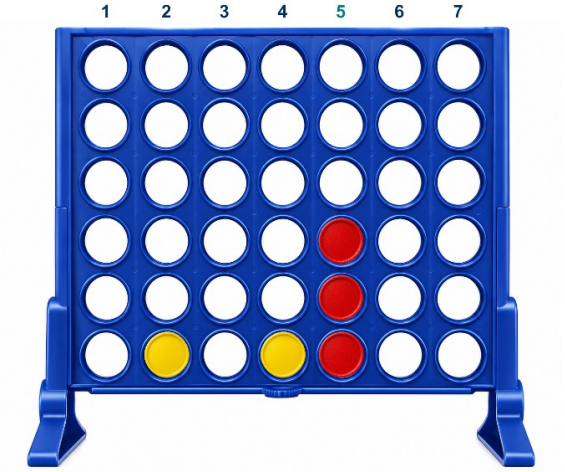
Four in a row

The new disc completes a diagonal from column 2 to column 5.

Reward for Yellow: +1

The basics in Connect Four

Agent	The program that chooses Yellow's moves.
Environment	The game applies gravity, updates the board, and checks for a win or draw.
State	The board layout and whose turn it is. At s_5, Yellow is to move.
Action	Drop a disc into a column with room.
Reward	For Yellow, 0 while play continues, +1 for a win, −1 for a loss, and 0 for a draw.
Return	From s_5 in Game 2, Yellow's future rewards add up to +1.
Episode	One complete game from the empty board to the result.



s_5 · Yellow to move

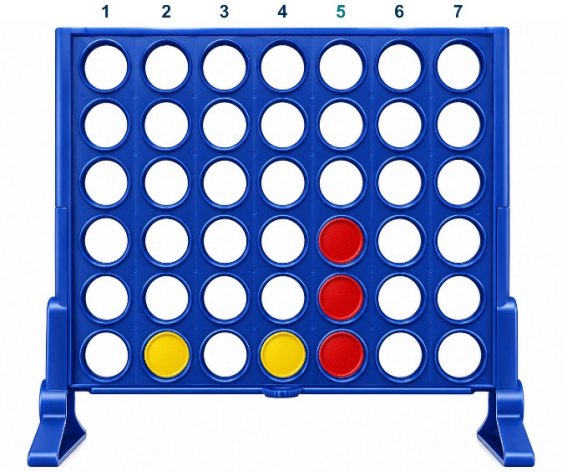
Deciding and learning in Connect Four

Policy $\pi(a \mid s_5)$ is the probability that Yellow chooses column a at s_5.

State value $V(s_5)$ estimates Yellow's return from this board.

Action value $Q(s_5, a)$ estimates Yellow's return if Yellow first drops in column a.

Column 3 leaves Red a winning reply. Column 5 removes it. So $Q(s_5, 3)$ and $Q(s_5, 5)$ differ.

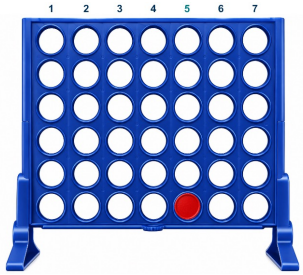


s_5 · Yellow to move

The return when Red wins

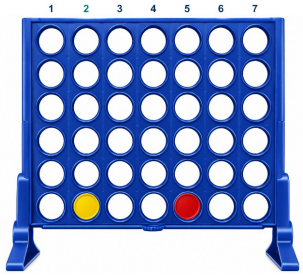
After move t, r_t is the reward just earned and G_t is the reward still to come for Yellow.

1 Red in column 5



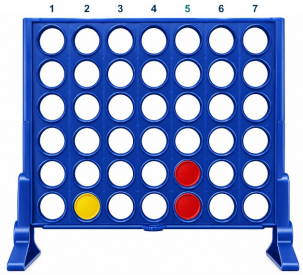
$r_1 = 0$ **$G_1 = -1$**

2 Yellow in column 2



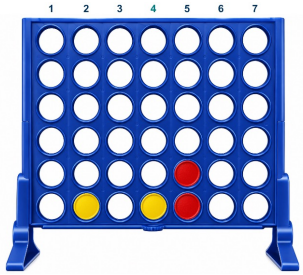
$r_2 = 0$ **$G_2 = -1$**

3 Red in column 5



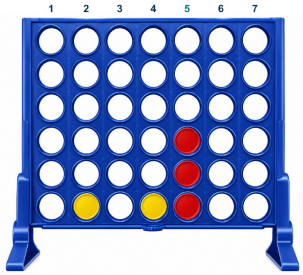
$r_3 = 0$ **$G_3 = -1$**

4 Yellow in column 4



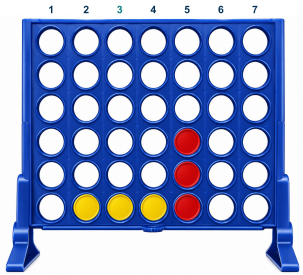
$r_4 = 0$ **$G_4 = -1$**

5 Red in column 5



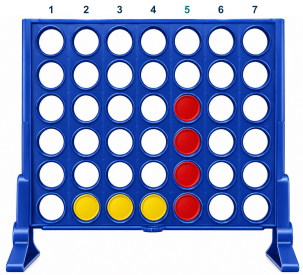
$r_5 = 0$ **$G_5 = -1$**

6 Yellow in column 3



$r_6 = 0$ **$G_6 = -1$**

7 Red wins in 5



$r_7 = -1$ **$G_7 = 0$**

Two moves remain after board 5

$G_5 = 0 + (-1) = -1$

Yellow attacks in column 3. Red's reply earns −1 for Yellow.

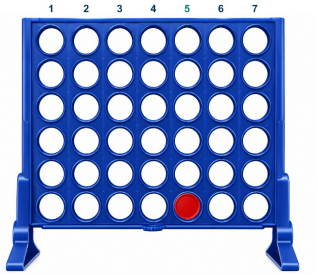
Once the game ends, we can compute G for every earlier board.

Game 1 from Yellow's viewpoint. No discounting ($\gamma = 1$).

The return when Yellow wins

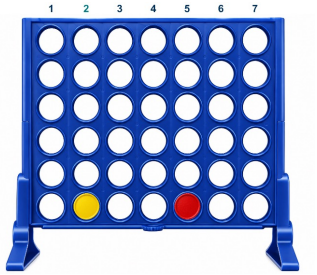
After move t, r_t is the reward just earned and G_t is the reward still to come for Yellow.

1 Red in column 5



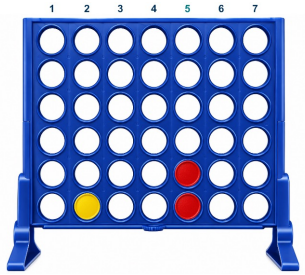
$r_1 = 0$ **$G_1 = +1$**

2 Yellow in column 2



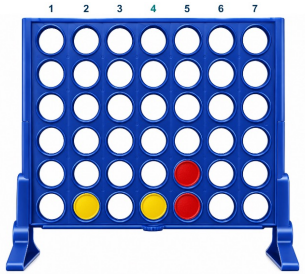
$r_2 = 0$ **$G_2 = +1$**

3 Red in column 5



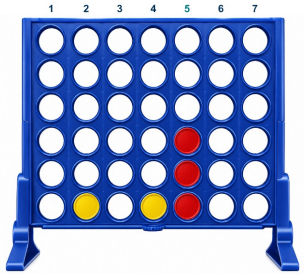
$r_3 = 0$ **$G_3 = +1$**

4 Yellow in column 4



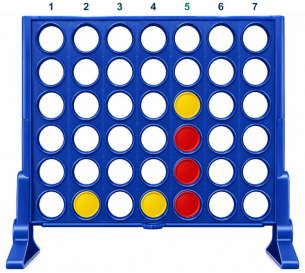
$r_4 = 0$ **$G_4 = +1$**

5 Red in column 5



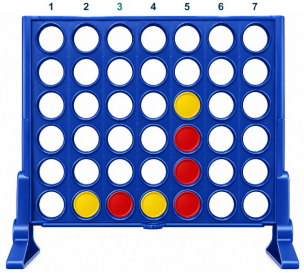
$r_5 = 0$ **$G_5 = +1$**

6 Yellow in column 5



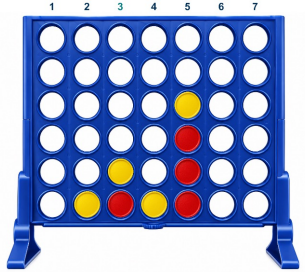
$r_6 = 0$ **$G_6 = +1$**

7 Red in column 3



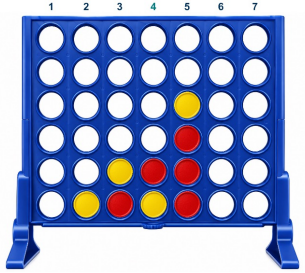
$r_7 = 0$ **$G_7 = +1$**

8 Yellow in column 3



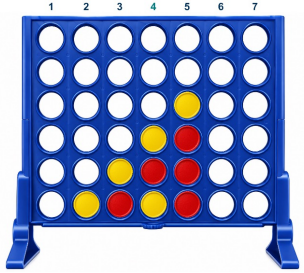
$r_8 = 0$ **$G_8 = +1$**

9 Red in column 4



$r_9 = 0$ **$G_9 = +1$**

10 Yellow wins in 4



$r_{10} = +1$ **$G_{10} = 0$**

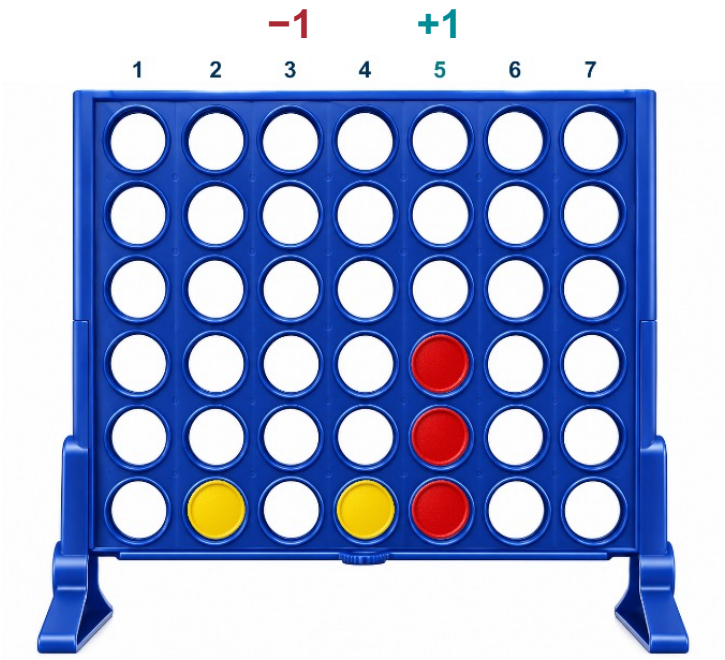
$G_t = r_{t+1} + \gamma\, G_{t+1}$

With $\gamma = 1$, the five moves after board 5 give $G_5 = 0 + 0 + 0 + 0 + 1 = +1$.

Game 2 from Yellow's viewpoint. Return notation follows Sutton & Barto (2018), Ch. 3.

Yellow to move

$$Q(s, a) = \frac{\sum G_t}{N(s, a)}$$



What each quantity means

Q(s, a)	Estimated return for choosing a on board s.
s = s_5	Shared board after move 5, with Yellow to play.
a	Column Yellow chooses for move 6.
t	Number of moves already played. Here t = 5.
G_t	Yellow's return after move t in one completed game.
$\sum G_t$	Sum over games in which Yellow chose a from s.
N(s, a)	Number of games in which Yellow chose a from s.

Average each column's returns from s_5

Game 1
$G_5 = -1$

$$Q(s_5, 3) = \frac{\sum G_5 \text{ where } a = 3}{N(s_5, a = 3)} = (-1) / 1 = -1$$

Game 2
$G_5 = +1$

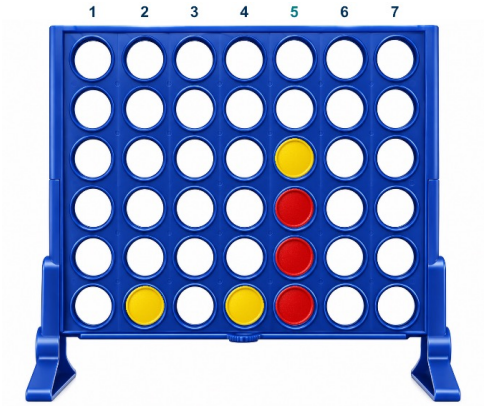
$$Q(s_5, 5) = \frac{\sum G_5 \text{ where } a = 5}{N(s_5, a = 5)} = (+1) / 1 = +1$$

The block earns 0 immediately. The later +1 reward sets its first estimate at +1.

What gets reinforced?

Yellow uses the later win to update its estimate of the earlier block.

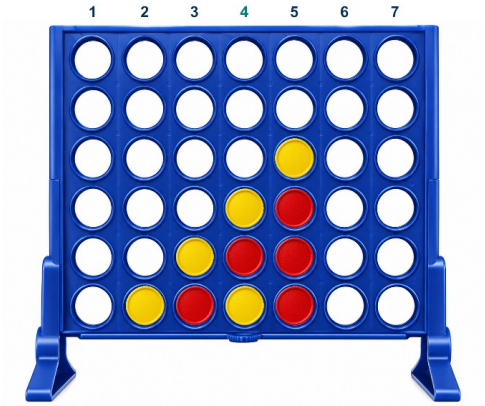
Yellow blocks in column 5



Immediate reward = 0

→

Yellow later wins



Reward at the end = +1

→

Yellow updates its estimate

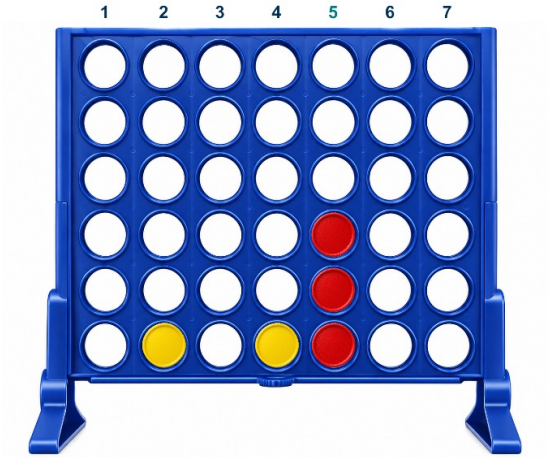
First estimate for blocking from s_5 = +1.

At s_5 again, that estimate gives Yellow a reason to choose the block.

The same board led to two different moves

After move 5

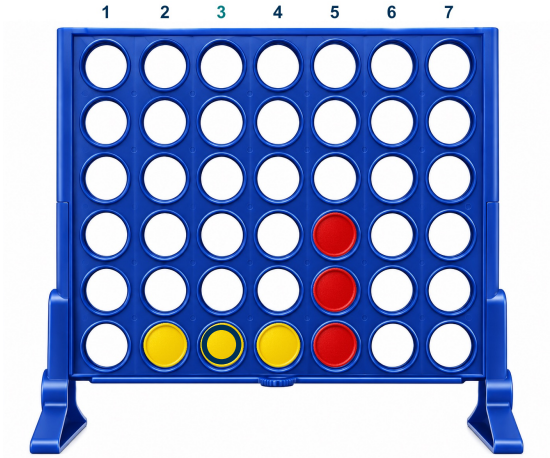
Yellow to move in both games



Red threatens to win in column 5.

Game 1

Yellow drops in column 3



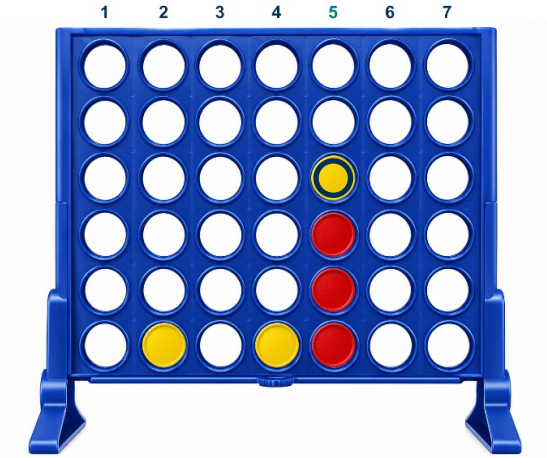
Red wins on move 7

$G_5 = 0 + (-1) = -1$

$Q(s_5, 3) = (-1) / 1 = -1$

Game 2

Yellow drops in column 5



Yellow wins on move 10

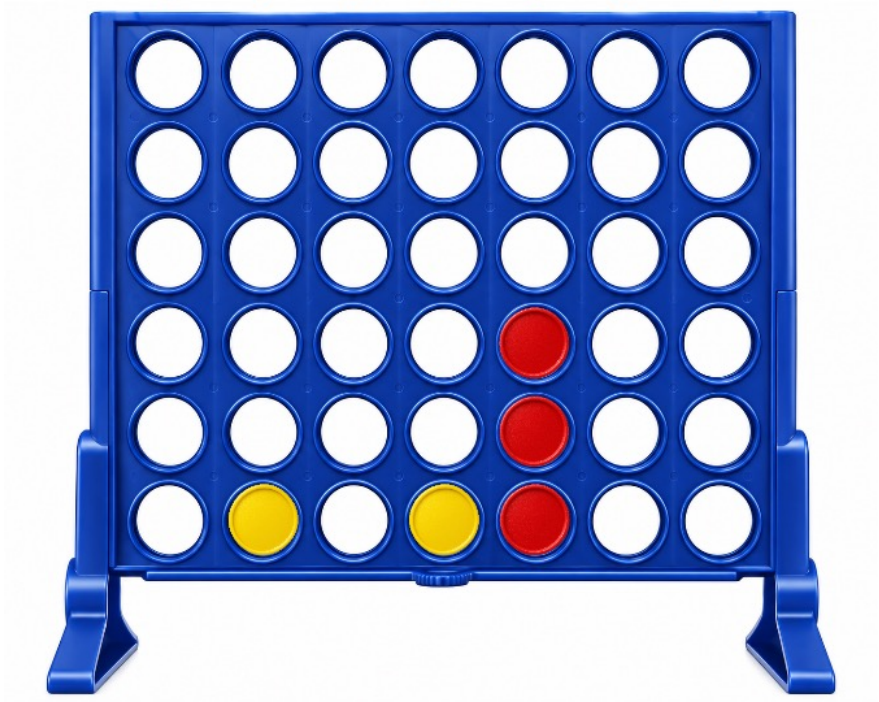
$G_5 = 0 + 0 + 0 + 0 + 1 = +1$

$Q(s_5, 5) = (+1) / 1 = +1$

A policy is the rule Yellow uses to choose a column from the board it sees.

Yellow's estimates across all seven columns

$Q(s_5, a)$	?	?	−1	?	+1	?	?
$N(s_5, a)$	0	0	1	0	1	0	0
	1	2	3	4	5	6	7



After Games 1 and 2

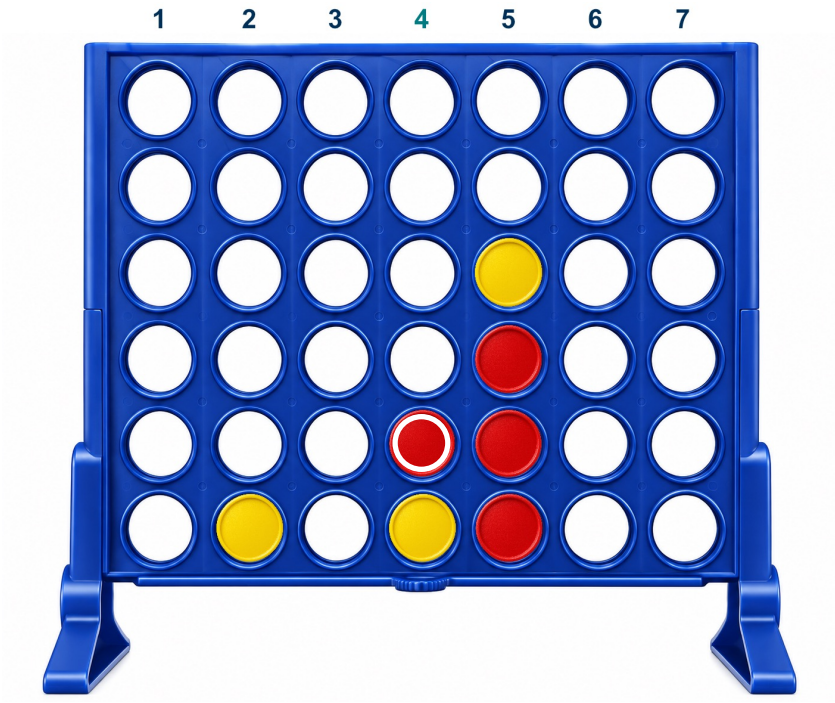
N counts how many games tried each column from this board.

? means the sample count is zero.

Yellow plays column 5 in Game 3

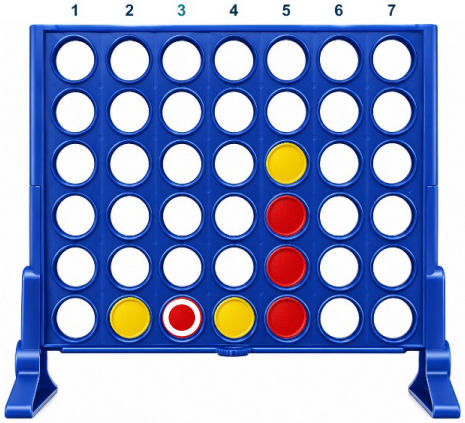
It has the higher estimate of the two moves Yellow has tried.

Red replies in column 4



Game 2 after move 7

Red plays column 3
after the same block.



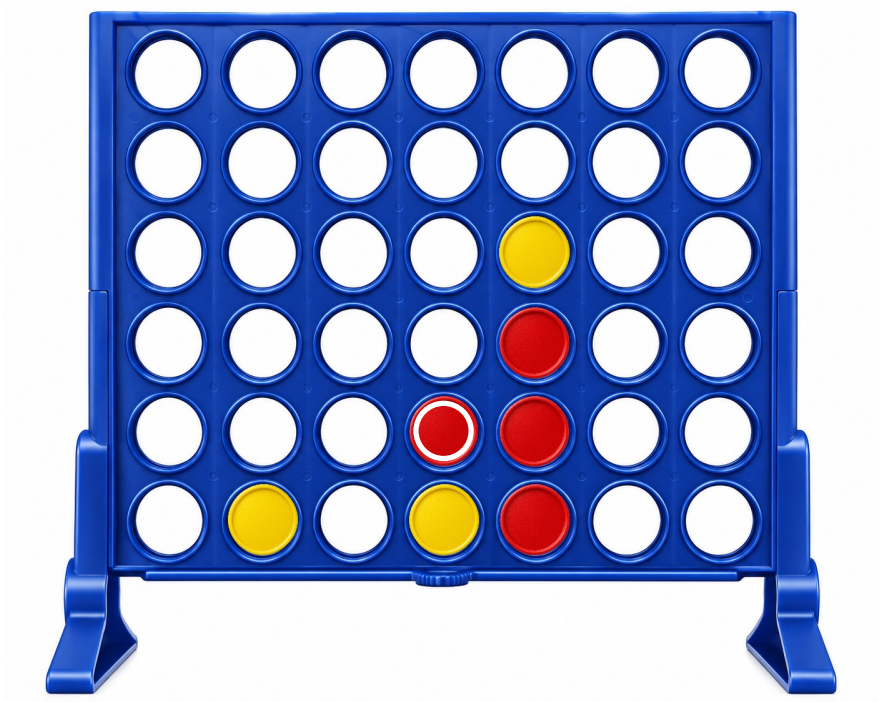
Yellow now faces a new board

The earlier Q values are for the move-5 board.

What should Yellow play on move 8?

All seven columns have zero samples at s_7

$Q(s_7, a)$	?	?	?	?	?	?	?
$N(s_7, a)$	0	0	0	0	0	0	0
	1	2	3	4	5	6	7



After Red's column 4 reply

This is Yellow's first visit to s_7. All seven columns are legal.

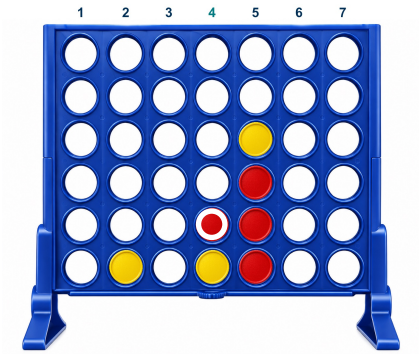
? means the sample count is zero.

The next move will be a trial

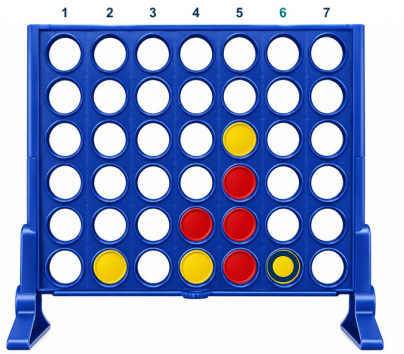
The completed game will give the chosen column its first return from this board.

Yellow explores a new board

After move 7



Yellow plays column 6



$N(s_7, a) = 0$ for every column

First visit to this board: no estimates, so the pick is random, 1 in 7 per column.

Exploration rule: ε-greedy

In every game, with probability ε = 0.2, Yellow picks a legal column at random. Otherwise it picks the highest estimate.

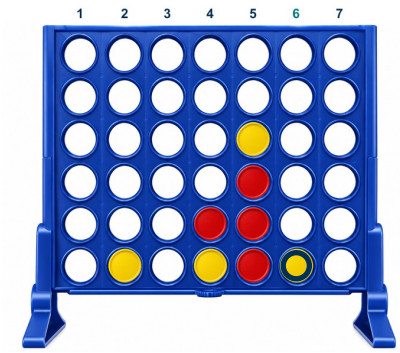
1 2 3 4 5 6 7

The example draw lands on 6

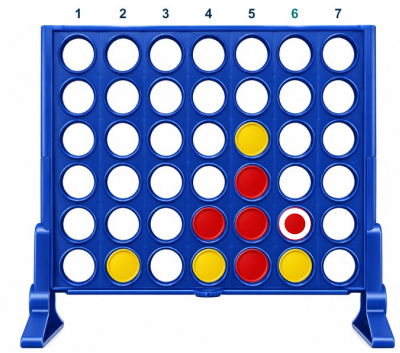
The outcome of Game 3 will give this choice its first return.

Game 3 ends on move 11

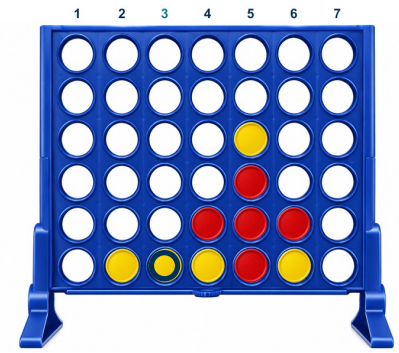
8 Yellow in column 6



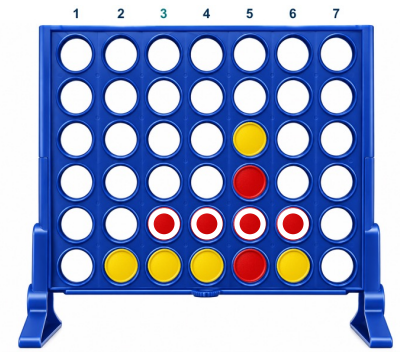
9 Red in column 6



10 Yellow in column 3



11 Red in column 3



Red wins with four across.

After Red wins, Yellow updates two estimates

Move 6 at s_5

Game 2 gives $G_5 = +1$. Game 3 gives $G_5 = -1$.

$Q(s_5, 5) = (+1 + (-1)) / 2 = 0$

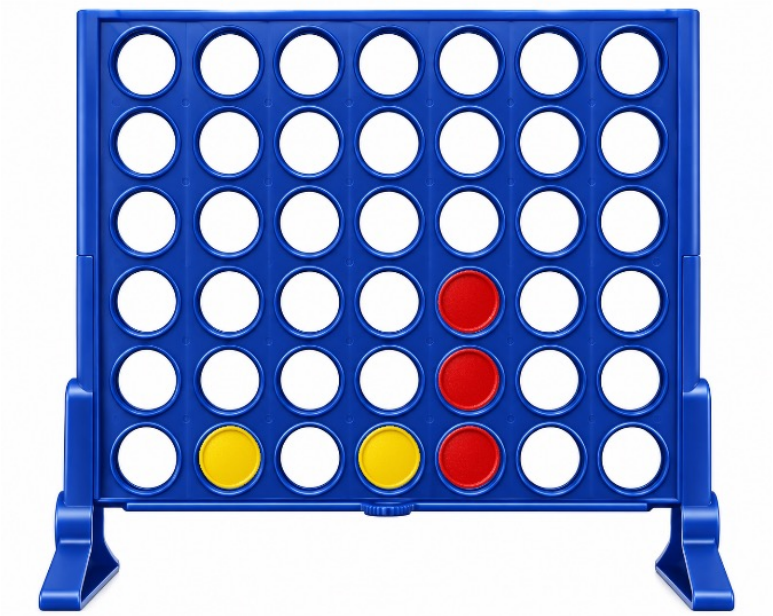
Move 8 at s_7

$G_7 = 0 + 0 + 0 + (-1) = -1$

$Q(s_7, 6) = (-1) / 1 = -1$

Column 5 gets a second sample at s_5

Q after 2 games	?	?	−1	?	**+1**	?	?
N after 2 games	0	0	1	0	**1**	0	0
Q after 3 games	?	?	−1	?	**0**	?	?
N after 3 games	0	0	1	0	**2**	0	0
	1	**2**	**3**	**4**	**5**	**6**	**7**



Column 5 has two samples

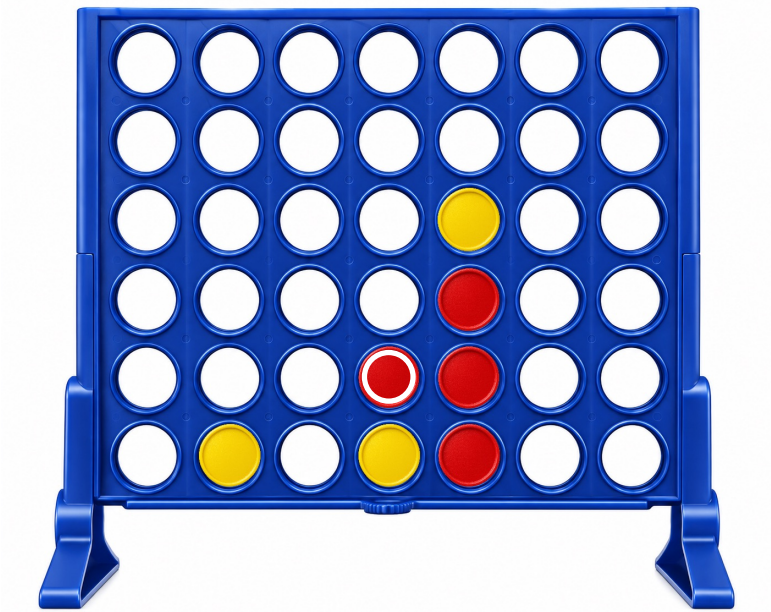
Game 2 gives $G_5 = +1$.
Game 3 gives $G_5 = -1$.

$$Q(s_5, 5) = \frac{+1 + (-1)}{2} = 0$$

0 is an average of two returns.
? marks a column with zero samples.

Column 6 gets its first estimate at s_7

Q after 2 games	?	?	?	?	?	**?**	?
N after 2 games	0	0	0	0	0	**0**	0
Q after 3 games	?	?	?	?	?	**−1**	?
N after 3 games	0	0	0	0	0	**1**	0
	1	**2**	**3**	**4**	**5**	**6**	**7**



Column 6 has one sample

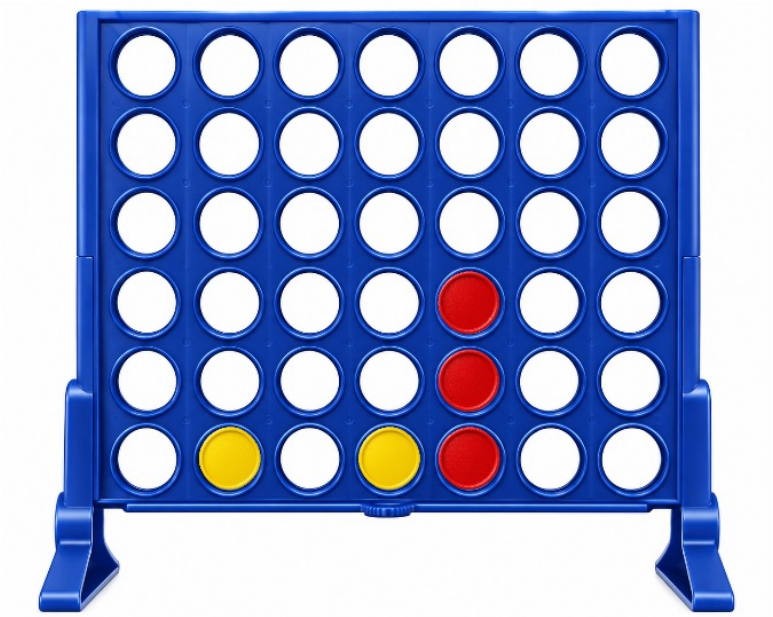
Yellow tried column 6 on move 8.
Red's win gives $G_7 = -1$.

$$Q(s_7, 6) = \frac{-1}{1} = -1$$

The other six columns still have zero samples.

What the estimates could look like after 30 games

	1	2	3	4	5	6	7
Q after 3 games	?	?	−1	?	**0**	?	?
N after 3 games	0	0	1	0	**2**	0	0
Q after 30 games	−1	?	−1	?	**+0.38**	−1	?
N after 30 games	1	0	2	0	**16**	1	0



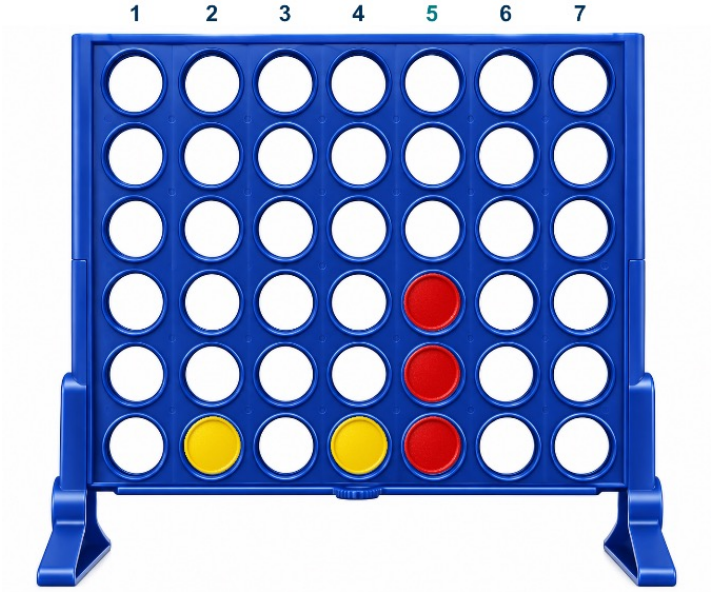
20 of the 30 games reached s_5

s_5 is this disc layout with Yellow to move.
Column 5 gave 11 wins and 5 losses in 16 trials.

$$Q(s_5, 5) = \frac{11(+1) + 5(-1)}{16} \approx +0.38$$

Greedy picks chose 5. Random picks (ε = 0.2) tried 1, 3 and 6.
Columns 2, 4 and 7 have no samples yet.

I am at s_5. Which column should I play?



s_5 · Yellow to move

What I know: the 30-game table on slide 39

Column	1	2	3	4	5	6	7
$Q(s_5, a)$	−1	?	−1	?	+0.38	−1	?

Three rules that answer the question

Greedy	Play column 5, the best estimate, every time.
Random	Play any column, each with a 1 in 7 chance.
ε-greedy, ε = 0.2	Play column 5, but 20% of the time play a random column.

A policy is a rule that answers this question at every board: which move to make.

A policy gives every move a probability: π(a | s)

$\pi(a \mid s) = \Pr(a_t = a \mid s_t = s)$

The probability that the agent plays action a at step t, given that the state at step t is s.
At every state, the probabilities of all the moves add to 1.

$\pi(a \mid s_5)$ for the three rules on slide 40

Column	1	2	3	4	5	6	7
Greedy	0	0	0	0	**1**	0	0
Random	1/7	1/7	1/7	1/7	**1/7**	1/7	1/7
ε-greedy, ε = 0.2	1/35	1/35	1/35	1/35	**29/35**	1/35	1/35

Greedy is deterministic: all of the probability sits on one move.
ε-greedy gives every column ε/7 = 1/35 ≈ 0.03, and column 5 another 1 − ε = 0.8, so 29/35 ≈ 0.83.

A board's value averages its Q's, weighted by the policy: $V(s) = \Sigma_a \pi(a \mid s)\, Q(s, a)$

Greedy at s_5: $V(s_5) = 1 \times Q(s_5, 5) \approx +0.38$, using the 30-game estimate on slide 39.

Sutton & Barto (2018), Sec. 3.5 (policy) and Sec. 5.4 (ε-greedy probabilities).

If I block at s_5, how will the game go on average?

Before: keep a record of every game

$Q = (G_1 + G_2 + \ldots + G_N) / N$

Needs the G of every past game.

Now: update after each game, no record

new Q = old Q + (1/N) × (target − old Q)

target = G of the game just played. Store only Q and N.

Check: slide 39 has Q = 6/16 = 0.375. The next game is a loss.

Record of all 17 games	G's add to 5, so Q = 5/17 ≈ 0.29
Update from Q and N	0.375 + (−1 − 0.375)/17 ≈ 0.29

Why they match: new Q = [(N − 1) × old Q + G] / N = old Q + (1/N) × (G − old Q).

Yellow decides at s_5 with Q. The target arrives only after the game ends, and then Q moves toward it.

Target = the number Q moves toward. Here it is G; the next slide uses a target that arrives sooner.

Incremental average: Sutton & Barto (2018), Sec. 2.4.

A table cannot cover every board

4.5 trillion

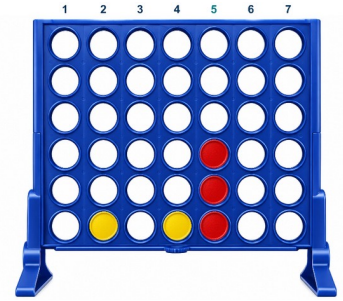
legal Connect Four positions (4,531,985,219,092)

2.1×10^{170}

legal Go positions

630 at most

Yellow-to-move boards visited in 30 games



A table stores one number per board, or per board and move.
It learns nothing about a board it has not visited.

A board that differs from s_5 by one disc far from column 5 starts at zero in the table, even though the same block is right.

We need a function that maps any board to a prediction and shares what it learns across similar boards.

Counts: Tromp, Connect Four positions (OEIS A212693); Tromp and Farnebäck, Combinatorics of Go (2016).

What learning adjusts: the parameters θ

The tables stored Q(s, a), one entry per board and move. A predictor $f_\theta(s)$ computes V(s), Q(s, a) or π(a | s) from the board instead. Learning changes θ so its outputs move toward their targets.

	Lookup table	Linear model	Trees or GP	Neural network
Predicts	**Q(s, a)**	**V(s) (slide 65)**	**V(s) or Q(s, a)**	**V(s) and π(a | s) (next slide); DQN: Q(s, a)**
θ is	one number per board (and move)	one weight per hand-made feature	split points and leaf values, or kernel settings	the weights and biases of every layer
Prediction	read the entry	weighted sum of features	follow the splits, or compare with stored boards	layers of weighted sums and nonlinearities
A new board	gets no information	shares weights through its features	borrows from similar boards	shares learned features
Connect Four	4.5 trillion entries	3 weights: bias, Red threat, Yellow threat	grows with the data	8,648 weights (next slide)
Example	slides 29–39	Samuel's checkers program (1959)	the Gaussian process (GP) in Module 9	DQN, AlphaGo Zero

The network in this lecture could be any of these. A deep neural network (DNN) is the usual choice for large raw inputs such as boards or images.

A network that predicts V(s) and π(a | s)

Input: the board

Three 6 × 7 planes:
Yellow discs, Red discs, player to move.

126 numbers

→

Hidden layer

64 units. Each takes a weighted sum of the 126 inputs, adds a bias, and passes it through a rectifier, max(0, x).

126 × 64 + 64 = 8,128

→

Policy head

7 outputs → softmax → π(a | s), the move probabilities

64 × 7 + 7 = 455

Value head

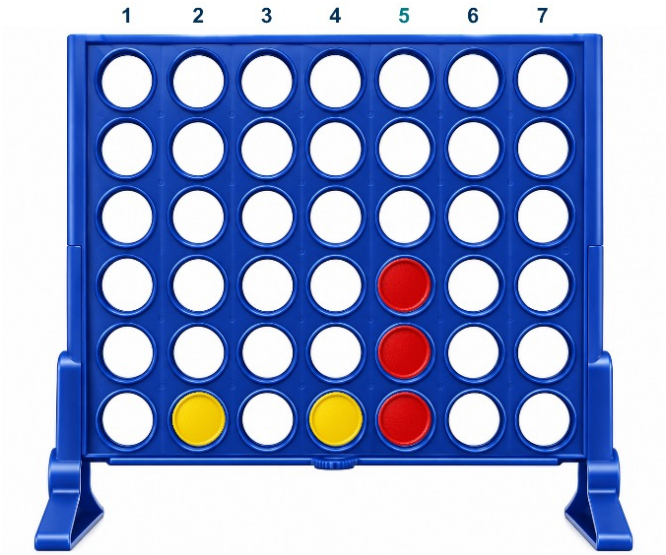
1 output → tanh → V(s) from −1 to +1

64 + 1 = 65

Input: any board s. Output: V(s) and π(a | s). θ = all 8,648 weights and biases; training changes every one of them.

Small illustrative design. A DQN-style network (slide 77) outputs one Q per column instead. AlphaGo Zero used these two heads on a much deeper body.

Why look ahead before moving?



s_5 · Yellow to move

The network is a fast guess

At s_5 it puts 60% on the attack in column 3. But the attack loses: Red replies in column 5 and wins.

Yellow knows the rules, so it can check first

Try a move on a copy of the board, add Red's reply, and see what happens. One imagined line of play is a simulation.

The network still guides the search

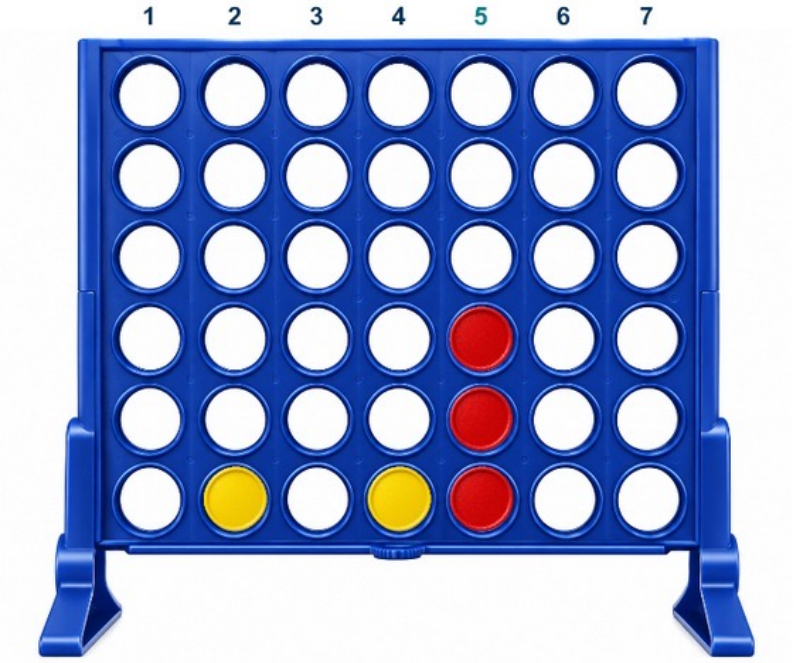
$\pi(a \mid s)$ says which moves to try first. $V(s)$ scores the board where a simulation stops.

Search gives a better move now, and its move probabilities become a better training target (slide 59).

Looking ahead with the rules is model-based planning (slide 79). The search here is Monte Carlo tree search (slide 48).

Can Yellow spot Red’s winning reply?

Return to the board after move 5. Red has three discs stacked in column 5.



Board s_5 · Yellow to move

Attack in column 3

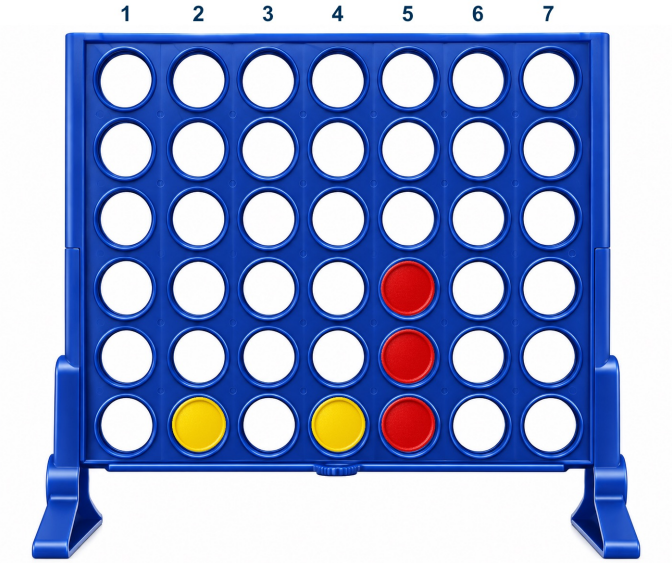
Yellow would make three in a row along the bottom. What could Red play next?

Block in column 5

Yellow would take the open spot directly above Red’s three discs.

Which move should Yellow play?

Why simulation 1 starts in column 3



Two candidate moves

3 builds an attack. 5 blocks Red's threat.

$$P_3 = \pi_{network}(3 \mid s_5) = 60\%$$

$$P_5 = \pi_{network}(5 \mid s_5) = 40\%$$

Search keeps its own N and Q, separate from the game tables. Both start at zero for this move. Set c = 1.

First simulation

Both scores are 0. Column 3 has the higher starting preference, so search tries it first.

P is the move probability predicted by the network before search.

Five-simulation teaching example. Search and training equations adapted from Silver et al. (2017).

How search chooses the next column

$$\text{Search score}(s,a) = \underbrace{Q(s,a)}_{\text{average score so far}} + \underbrace{c\,P(s,a)\frac{\sqrt{N(s)}}{1+N(s,a)}}_{\text{U(s, a) · exploration bonus}}.$$

After each simulation, update Q and the visit counts, then recalculate the search score for every candidate column. Start the next simulation in the column with the highest updated score.

s is the board. a is the column to try.

N(s) counts all simulations from this board.

Q(s,a) starts at 0 for an untried column.

N(s,a) counts simulations that tried this column.

P(s,a) is the network's starting preference.

c sets the strength of the bonus. Here, c = 1.

Count one visit for the column that starts each simulation.

On the actual board, Yellow plays the column that started the most simulations.

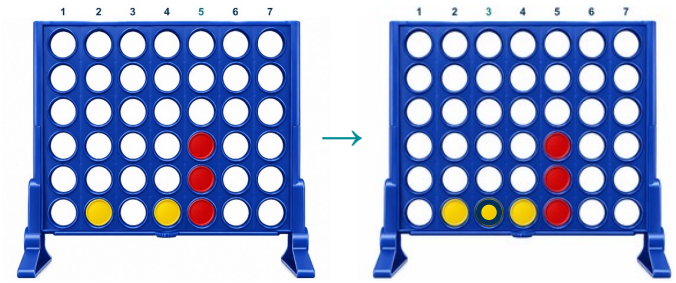
The visit counts and averages change after each simulation. The starting preferences stay fixed.

$$\text{Search score}(s,a) = \underbrace{Q(s,a)}_{\text{average score so far}} + \underbrace{c\,P(s,a)\frac{\sqrt{N(s)}}{1+N(s,a)}}_{\text{exploration bonus}}.$$

Simulation 1

Yellow tries column 3. The simulation stops on the new board with Red to move.

Starting board s_5 → **Yellow plays column 3**



Yellow to move **Red to move**

	Before Simulation 1		After Simulation 1	
Quantity	**Col. 3**	**Col. 5**	**Col. 3**	**Col. 5**
Network preference P(s,a)	0.60	0.40	0.60	0.40
Yellow's first drops $N(s_5,a)$	0	0	1	0
Simulations from s_5 $N(s_5)$	0	0	1	1
Average score Q(s,a)	0	0	+0.25	0
Exploration bonus (c = 1)	0	0	$0.60\sqrt{1/2} = 0.30$	$0.40\sqrt{1/1} = 0.40$
Search score	**0**	**0**	**0.55**	**0.40**

The initial scores tie at 0. The network favors column 3 (0.60 vs. 0.40).

At the new board s′, V(s′) = −0.25 for Red, which is +0.25 for Yellow.

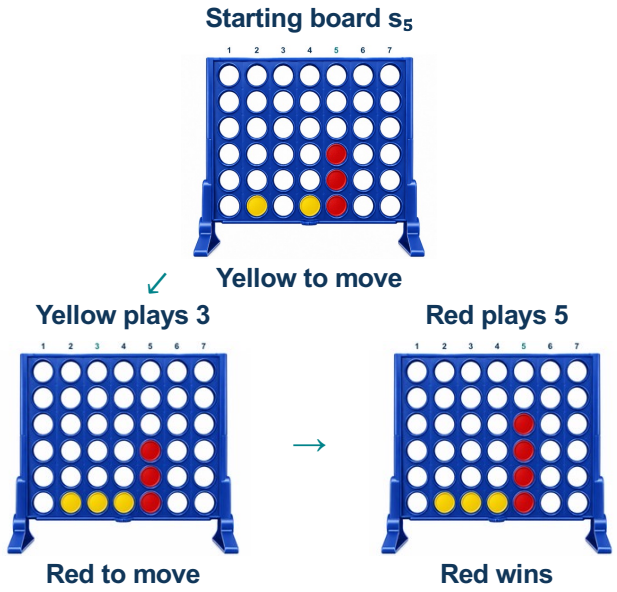
Q(s, 3) = (+0.25) / 1 = +0.25

0.55 is higher than 0.40, so simulation 2 starts in column 3 again.

$$\text{Search score}(s,a) = \underbrace{Q(s,a)}_{\text{average score so far}} + \underbrace{c\,P(s,a)\frac{\sqrt{N(s)}}{1+N(s,a)}}_{\text{exploration bonus}}.$$

Simulation 2

Column 3 starts with the higher search score. Red replies in column 5 and wins.



	Before Simulation 2		After Simulation 2	
Quantity	**Col. 3**	**Col. 5**	**Col. 3**	**Col. 5**
Network preference P(s,a)	0.60	0.40	0.60	0.40
Yellow's first drops $N(s_5,a)$	1	0	2	0
Simulations from s_5 $N(s_5)$	1	1	2	2
Average score Q(s,a)	+0.25	0	−0.375	0
Exploration bonus (c = 1)	0.30	0.40	$0.60\sqrt{2}/3 \approx 0.283$	$0.40\sqrt{2}/1 \approx 0.566$
Search score	**0.55**	**0.40**	**−0.092**	**0.566**

Red's win scores −1 for Yellow.

Q(s, 3) = (+0.25 − 1) / 2 = −0.375

Simulation 3 starts in column 5 because 0.566 is higher than −0.092.

Five-simulation teaching example. Search and training equations adapted from Silver et al. (2017).

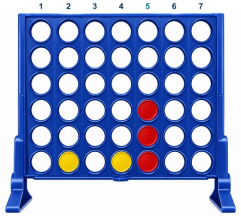
WORKED CONNECT FOUR SEARCH

$$\text{Search score}(s,a) = \underbrace{Q(s,a)}_{\text{average score so far}} + \underbrace{c\,P(s,a)\frac{\sqrt{N(s)}}{1+N(s,a)}}_{\text{exploration bonus}}.$$

Simulation 3

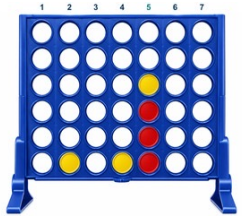
Column 5 has the higher search score, so Yellow starts there.

s_5 · Yellow to move



→

1. Yellow plays 5



−V(s′) = +0.20 for Yellow

	Before Simulation 3		After Simulation 3	
Quantity	**Col. 3**	**Col. 5**	**Col. 3**	**Col. 5**
Network preference P(s,a)	0.60	0.40	0.60	0.40
Yellow's first drops N(s_5,a)	2	0	2	1
Simulations from s_5 N(s_5)	2	2	3	3
Average score Q(s,a)	−0.375	0	−0.375	0.200
Exploration bonus (c = 1)	0.283	0.566	0.60√3/3 ≈ 0.346	0.40√3/2 ≈ 0.346
Search score	**−0.092**	**0.566**	**−0.029**	**0.546**

Q(s, 5) = (+0.20) / 1 = +0.20

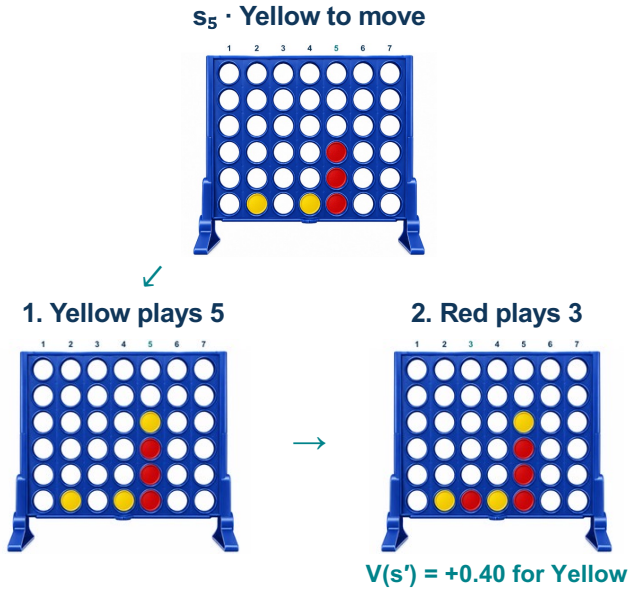
Simulation 4 starts in column 5 because 0.546 is higher than −0.029.

WORKED CONNECT FOUR SEARCH

$$\text{Search score}(s,a) = \underbrace{Q(s,a)}_{\text{average score so far}} + \underbrace{c\,P(s,a)\frac{\sqrt{N(s)}}{1+N(s,a)}}_{\text{exploration bonus}}.$$

Simulation 4

Column 5 has the higher search score, so Yellow starts there.



	Before Simulation 4		After Simulation 4	
Quantity	**Col. 3**	**Col. 5**	**Col. 3**	**Col. 5**
Network preference P(s,a)	0.60	0.40	0.60	0.40
Yellow's first drops $N(s_5,a)$	2	1	2	2
Simulations from s_5 $N(s_5)$	3	3	4	4
Average score Q(s,a)	−0.375	0.200	−0.375	0.300
Exploration bonus (c = 1)	0.346	0.346	0.60√4/3 ≈ 0.400	0.40√4/3 ≈ 0.267
Search score	**−0.029**	**0.546**	**0.025**	**0.567**

Q(s, 5) = (+0.20 + 0.40) / 2 = +0.30

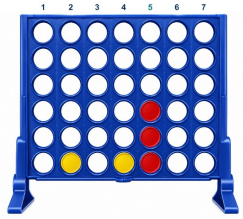
Simulation 5 starts in column 5 because 0.567 is higher than 0.025.

$$\text{Search score}(s,a) = \underbrace{Q(s,a)}_{\text{average score so far}} + \underbrace{c\,P(s,a)\frac{\sqrt{N(s)}}{1+N(s,a)}}_{\text{exploration bonus}}.$$

Simulation 5

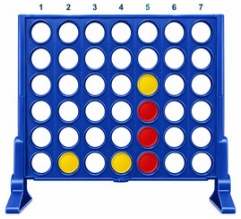
Column 5 has the higher search score, so Yellow starts there.

s_5 · Yellow to move

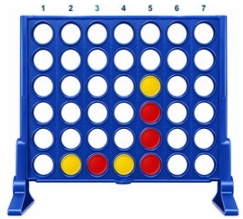


→

1. Yellow plays 5

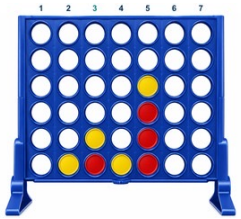


2. Red plays 3



→

3. Yellow plays 3



−V(s′) = +0.50 for Yellow

	Before Simulation 5		After Simulation 5	
Quantity	**Col. 3**	**Col. 5**	**Col. 3**	**Col. 5**
Network preference P(s,a)	0.60	0.40	0.60	0.40
Yellow's first drops N(s_5,a)	2	2	2	3
Simulations from s_5 N(s_5)	4	4	5	5
Average score Q(s,a)	−0.375	0.300	−0.375	0.367
Exploration bonus (c = 1)	0.400	0.267	0.60√5/3 ≈ 0.447	0.40√5/4 ≈ 0.224
Search score	**0.025**	**0.567**	**0.072**	**0.590**

The last scores show the update. The visit counts guide the move.

Q(s, 5) = (+0.20 + 0.40 + 0.50) / 3 ≈ +0.367

The budget is used up. Column 5 has 3 of 5 visits (60%) and column 3 has 2 (40%).

Five simulations favor the block

Sim.	Scores before (3, 5)	Line tried	Value for Yellow	Visits after (3, 5)
1	0.000, 0.000	Yellow 3	+0.25	1, 0
2	0.550, 0.400	Yellow 3, Red 5 wins	−1.00	2, 0
3	−0.092, 0.566	Yellow 5	+0.20	2, 1
4	−0.029, 0.546	Yellow 5, Red 3	+0.40	2, 2
5	0.025, 0.567	Yellow 5, Red 3, Yellow 3	+0.50	2, 3

Q_5 after simulations 3, 4, and 5: 0.20, 0.30, 0.367

$\pi_{search}(3 \mid s_5) = 2 / 5 = 40\%$ $\pi_{search}(5 \mid s_5) = 3 / 5 = 60\%$

The search visits column 5 more often, even though the network initially favored column 3.

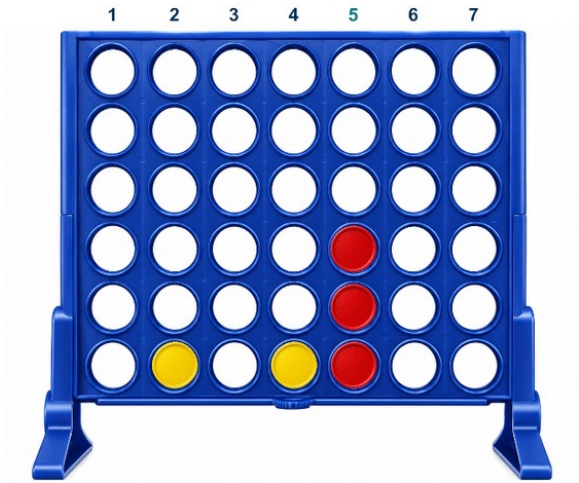
Five-simulation teaching example. Search and training equations adapted from Silver et al. (2017).

Which column does Yellow play?

Search scores guide the simulations. Yellow then plays column 5 on the actual board because it had the most visits during the simulations.

Which board is s_5?

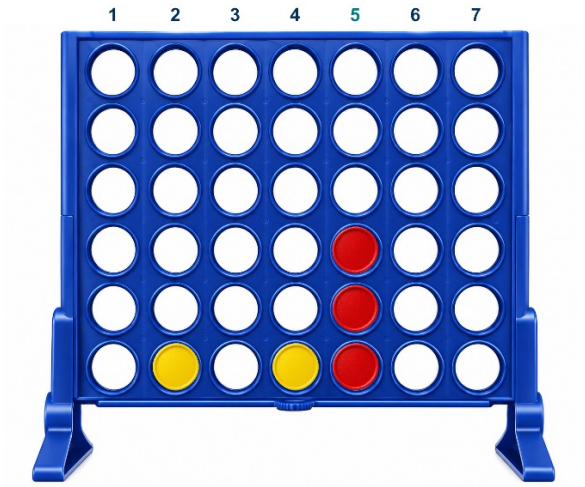
s_5 is the starting board on slides 50–54. Five discs are in place, and Yellow is about to move.



Yellow plays column 5

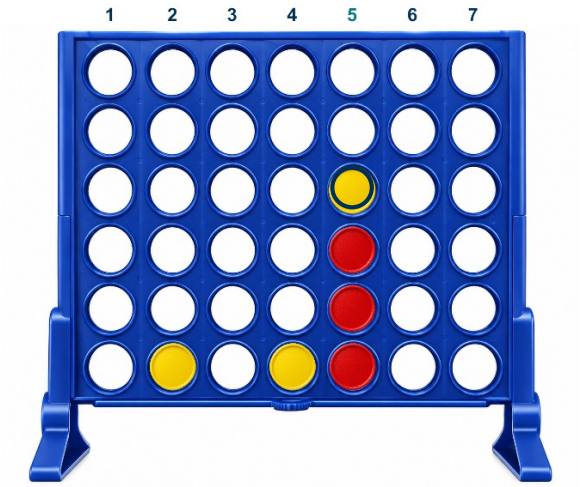
Three of the five simulations began in column 5, so Yellow drops there in the actual game.

Before the drop · Yellow to move



→

After the drop · Red to move

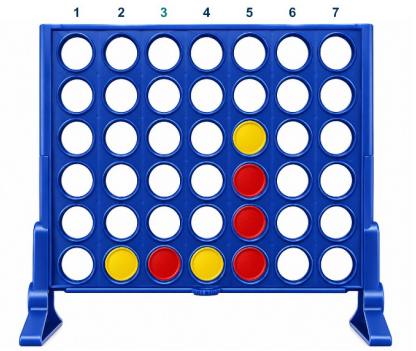


Red starts its own search from the new board.

The players finish the game

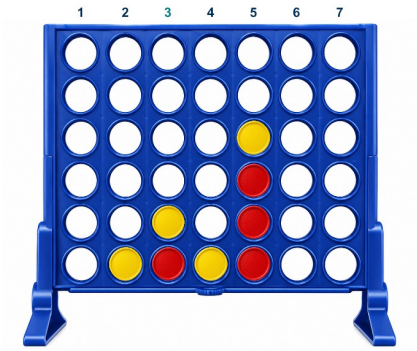
On each turn, the player searches from the current board and then plays one move.

7 · Red plays 3



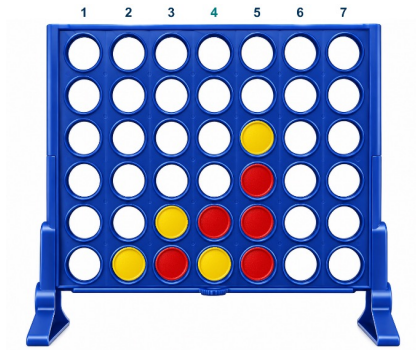
→

8 · Yellow plays 3



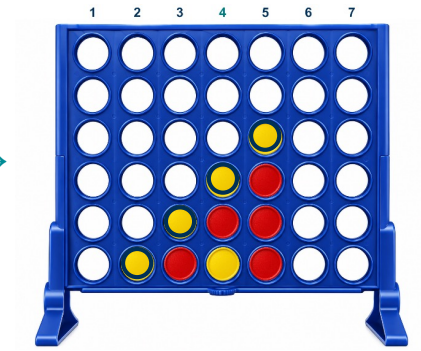
→

9 · Red plays 4



→

10 · Yellow plays 4

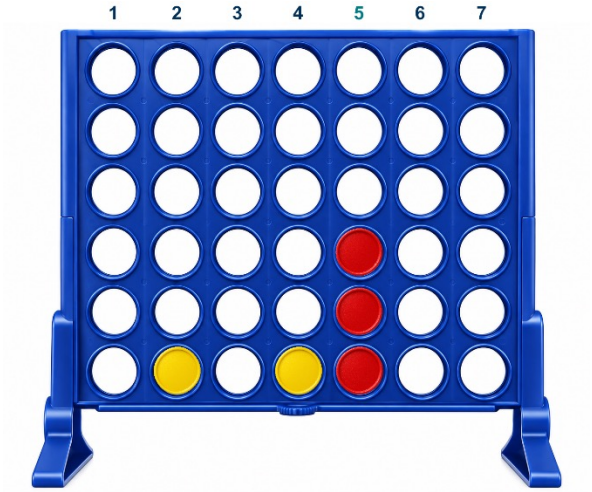


Yellow wins on move 10 with four discs on a diagonal.

What should the network predict at s_5?

Train the network to predict these move probabilities when it sees s_5 again.

s_5 · Yellow to move



$$\pi_{search}(3 \mid s_5) = 2 / 5 = 40\%$$

$$\pi_{search}(5 \mid s_5) = 3 / 5 = 60\%$$

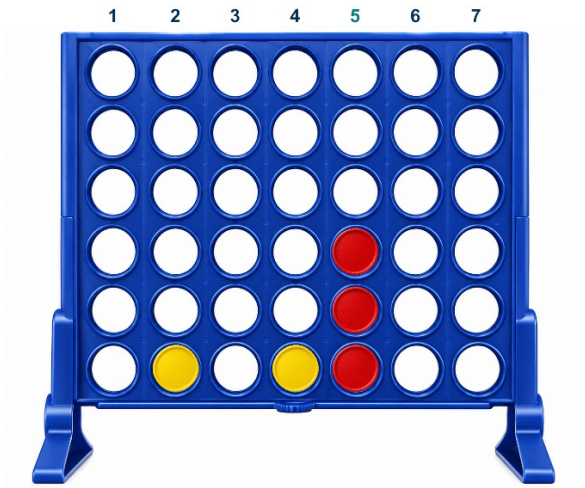
Save the two visit shares as the policy training target.

What is a value target?

The value target is the final result of the actual game, scored from the perspective of the player who was to move at the earlier board.

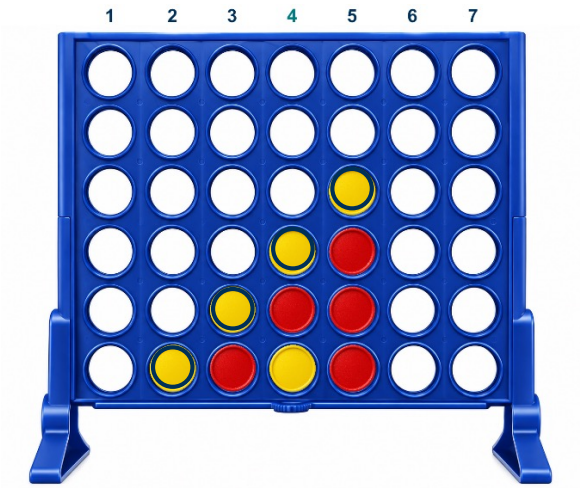
+1 for a win, 0 for a draw, and −1 for a loss.

s_5 · Yellow to move



→

Final board · Yellow wins



Train $V(s_5)$ toward +1 because Yellow went on to win.

From two targets to a weight update

One deep neural network uses board s_5 to predict both policy and value.

Policy training target

Search visits at s_5
2 of 5 visits to column 3 → 40%
3 of 5 visits to column 5 → 60%

Value training target

Completed game
Yellow wins, so training moves $V(s_5)$ toward +1.

How does the network learn from these targets?

Compare each prediction with its target. Use the resulting loss to update the network's weights.

How far are the predictions from the targets?

Board s_5 and Yellow to move → Deep neural network → Policy and value predictions

Policy loss · cross-entropy

Predicted for columns 3 and 5 60%, 40%
Target from search visits 40%, 60%

$$L_p = -\Sigma_a \pi_a \ln p_a$$

$$= -0.40 \ln(0.60) - 0.60 \ln(0.40) \approx 0.7541$$

p is the network policy. π is the search policy.

Value loss · squared error

Predicted value $V(s_5)$ +0.25
Target from Yellow's win +1

$$L_v = (V(s_5) - z)^2$$

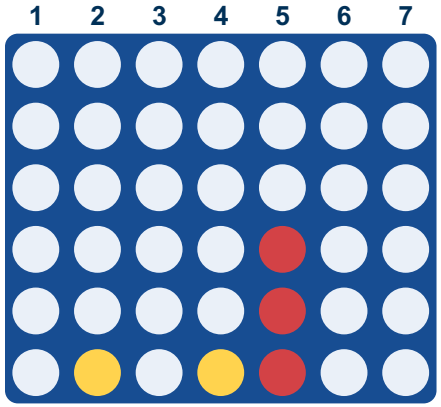
$$= (0.25 - 1)^2 = 0.5625$$

z is the final game result from Yellow's view.

Total data loss for this board $L_p + L_v \approx 1.3166$

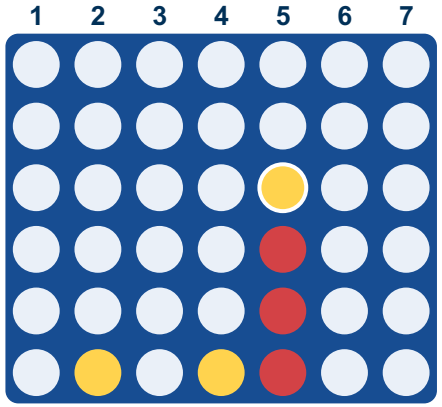
The block earns zero. The win comes later.

Search at board s_5



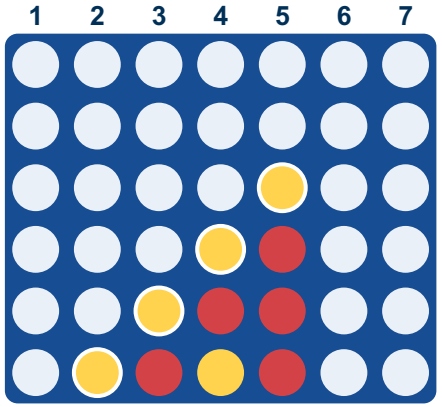
3 of 5 simulations start in column 5.

Yellow blocks in column 5



Immediate reward = 0

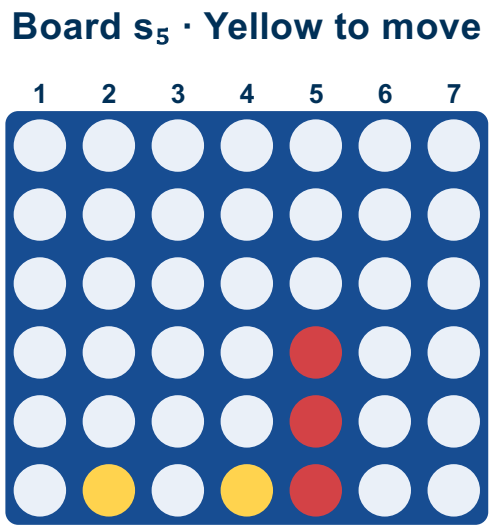
Yellow wins on move 10



Final reward = +1

The later win provides feedback on the earlier block.

One game gives the network two training targets



Policy target from the search

Column 3 starts 2 of 5 simulations → 40%
Column 5 starts 3 of 5 simulations → 60%

Value target from the completed game

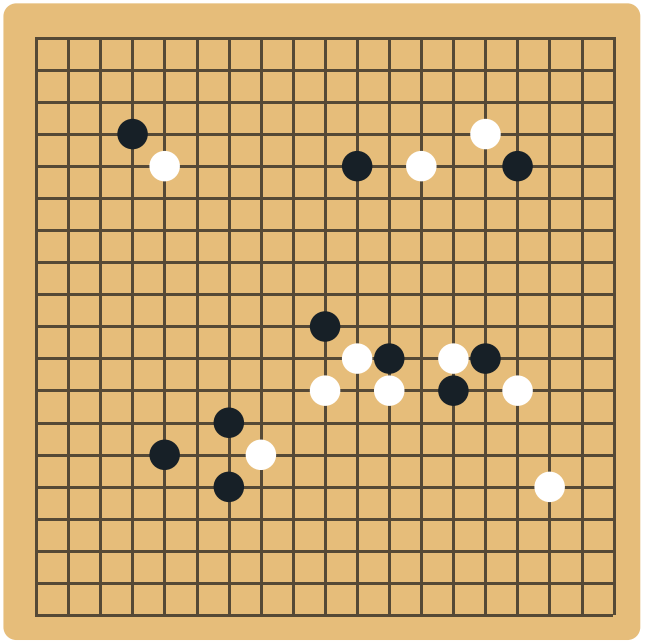
Yellow wins → train $V(s_5)$ toward +1.

Training adjusts the network's predictions so it can guide the next search.

Next

We followed one board from search to a weight update. One more way to train a policy comes first, then AlphaGo Zero at scale.

AlphaGo beats Lee Sedol 4–1



Illustrative Go position

Seoul · March 2016

4–1

Players take turns placing stones on the intersections.

Which next move is worth searching?

Source: Google DeepMind, AlphaGo match recap.

Where did the training data come from?

2016

AlphaGo

Learn to predict expert moves from human games.

Improve through games against itself.

2017

AlphaGo Zero

Start with a randomly initialized network and the rules of Go.

Generate all training games through self-play.

In both systems, search helps decide the next move.

Sources: Silver et al., Nature (2016, 2017).

What happens on one turn?

Board position

Stones on the board and recent position history.

→

One network

Policy p(a | s) gives move preferences. V(s) estimates the outcome.

→

Tree search

Use both predictions to decide which continuations to explore.

→

Next move

Choose a move using π from the search visit counts.

Connect Four used the same search sequence with five simulations.

Adapted from Silver et al., Nature (2017), Fig. 2.

A finished game supplies two training targets

For each board visited in self-play, save two targets.

Policy target π

The search visit distribution tells the network which moves search favored.

Cross-entropy

Value target z

The eventual winner gives the outcome for the player who was to move.

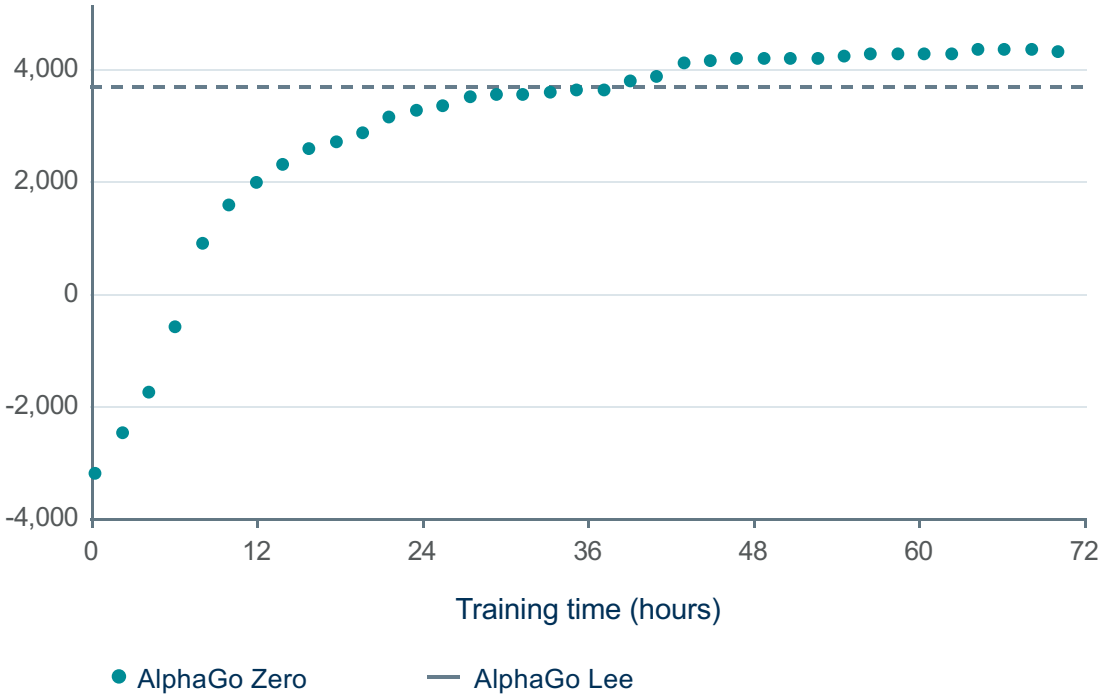
Squared value error

Train the network, then use its updated predictions in the next self-play games.

Adapted from Silver et al., Nature (2017), Fig. 1.

Playing strength rose during self-play

Estimated playing strength (Elo)



After about three days

4.9 million

self-play games in this run

100–0

in a separate match against the version that beat Lee Sedol

Approximate AlphaGo Zero points and AlphaGo Lee reference from Silver et al., Nature (2017), Fig. 3a.

The formal setup: a Markov decision process

Ingredient	Meaning	In Connect Four, from Yellow's side
States S	Every situation the agent can be in.	Every legal board, with the player to move.
Actions A(s)	The choices available in state s.	Columns with room.
Transitions P(s′ | s, a)	Probability that action a in state s leads to the next state, $s' = s_{t+1}$.	Gravity is fixed. Red's reply makes the next board random.
Reward R(s, a, s′)	The reward $r_{t+1} = R(s_t, a_t, s_{t+1})$ for that step.	+1 for a win, −1 for a loss, 0 otherwise.
Discount γ	Weight on later rewards, $0 \le \gamma \le 1$.	γ = 1. Every game ends within 42 moves.
Markov property	**The next state and reward depend only on s_t and a_t. Once the state is known, earlier history adds nothing.**	

Notation adapted from Sutton & Barto (2018), Ch. 3.

The goal: maximize expected return

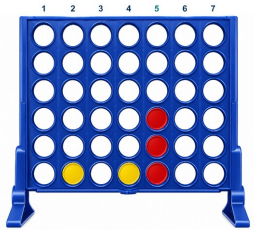
Quantity	Definition	Meaning
Return	$G_t = r_{t+1} + \gamma\, r_{t+2} + \gamma^2\, r_{t+3} + \ldots$	Future rewards, later ones weighted by γ.
Value of a policy	$V^\pi(s) = E_\pi[\, G_t \mid s_t = s \,]$	Expected return from s when following π.
Action value	$Q^\pi(s, a) = E_\pi[\, G_t \mid s_t = s, a_t = a \,]$	The same, with the first action fixed to a.
Goal	Find π^* with $V^{\pi^*}(s) \geq V^\pi(s)$ for every state s and every policy π	An optimal policy does at least as well as any other, in every state.
Acting optimally	$Q^*(s, a) = Q^{\pi^*}(s, a)$ In state s, choose $\text{argmax}_a\, Q^*(s, a)$	Knowing Q* makes acting easy. Many methods estimate Q* from experience.

In Connect Four, the tables on slides 29–39 estimated $Q^\pi(s_5, a)$ for Yellow's policy by averaging returns.

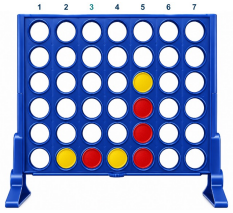
Expected = averaged over the randomness in the policy and the environment. Other slides drop the π when the policy is clear.

A move is worth its reward plus the next board's value

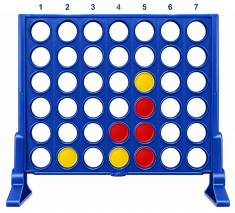
Games 2 and 3 are real games. Both reached s_5 and blocked, then Red replied differently, once each way.



s_5 · Yellow blocks, reward 0



Game 2: Red plays 3
Yellow went on to win, +1



Game 3: Red plays 4
Yellow lost at move 11, −1

A. Wait for the end: score each next board by the game's result

$Q(s_5, 5) = 0 + ½(+1) + ½(−1) = 0$

Each reply came up in 1 of the 2 games, so each gets ½. Same as slide 37.

B. Use the next board's value V instead

Game 3's −1 came later, from Yellow's own move 10 (slide 36).

Suppose V(after Red 3) = +1 and V(after Red 4) = +0.2.

$Q(s_5, 5) = 0 + ½(+1) + ½(+0.2) = +0.6$

Bellman equation

$$Q^{\pi}(s, a) = E_{\pi}[\, r_{t+1} + \gamma\, V^{\pi}(s_{t+1}) \mid s_t = s, a_t = a \,]$$

reward now, plus the value V of the next board, averaged over what can happen

Why V matters: it packs the rest of the game into one number, so a move can be scored one step ahead; the learning rules on slides 45–46, the look-ahead search on slides 55–63 and the critic on slide 78 all build on it.

The V values in B are illustrative. γ = 1. Yellow-to-move boards; Red's reply is part of the transition. Sutton and Barto (2018), Ch. 3.

TD learning updates as soon as the next board appears

The Q averages on slides 29–39 can be kept one sample at a time:

$$\text{new } Q = \text{old } Q + (1/N) \times (\text{target} - \text{old } Q)$$

Monte Carlo target: the final return. **TD target:** reward + γ × V(next board), from the Q table there (0 if no data).

Game 3: Yellow's move	Q before	Monte Carlo: target → after	TD: target → after
 **Block, column 5 at s_5**	+1, N = 1	−1 → 0	**0 + 0 = 0 → +0.5**
 **Column 6 at s_7**	no data	−1 → −1	**0 + 0 = 0 → 0**
 **Column 3 at s_9**	no data	−1 → −1	**−1 → −1**

The Monte Carlo column reproduces slides 37 and 38.

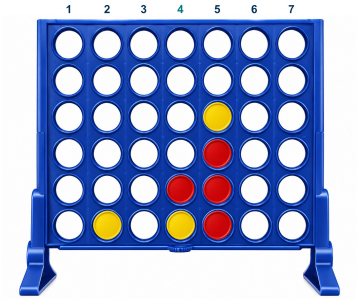
TD puts the loss on column 3 at s_9, the move that opened Red's row. The block keeps a positive Q, and column 6 waits until s_9 has data.

The next board's value $V(s_{t+1})$ lets learning stop one step early; it returns in dynamic programming (slide 50), the network's value head (slide 53), search at unfinished boards (slides 55–63) and the critic (slide 78).

TD = temporal difference: the update is driven by the difference between successive estimates. γ = 1.

Q-learning scores the next board by its best move

Which Q at s_7 should stand in for $V(s_7)$? In Game 3, Yellow's move there was a random pick (slide 35).



Later, after more games (illustrative):
$Q(s_7, 3) = +0.4$, the best
$Q(s_7, 6) = -0.5$
Block: $Q(s_5, 5) = +0.38$ (slide 39)
ε-greedy picks column 6 again.

$V(s_7)$ = Q of the move played next	Q-learning: $V(s_7)$ = the best Q
$0 + Q(s_7, 6) = -0.5$	$0 + \max_{a'} Q(s_7, a') = +0.4$
On-policy: values the moves actually played, exploration included. Pulls the block down.	Off-policy: values the best play while following ε-greedy. Keeps the block's credit.

Bellman optimality equation: the values of the best play

$$Q^*(s, a) = E[\, r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a \,]$$

Q-learning moves its Q estimates toward this equation.

Q-learning learns the values of the best play while the agent explores. The deep Q-network (DQN, 2015) is Q-learning with a neural network in place of the table.

Update: $Q(s, a) \leftarrow Q(s, a) + \alpha [\, r + \gamma \max Q(s', a') - Q(s, a) \,]$. Watkins and Dayan (1992); Mnih et al. (2015).

Monte Carlo or TD?

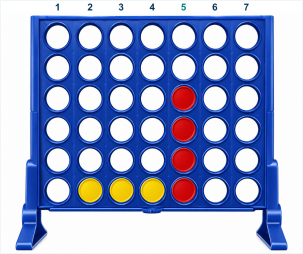
	Monte Carlo (slides 29–39)	TD and Q-learning (slides 45–46)
Target	The final return of the game	Reward + γ × the next board's estimate
Updates	After the game ends	After every move
Game 3's loss	Shared by every Yellow move, the block included	Charged to column 3 at s_9, the losing move
Weak point	Noisy: carries every later random event	Biased while estimates are poor: s_7 started at 0
Needs an end	Yes	No: works on a tool that runs lot after lot
Used in	The Connect Four tables; AlphaGo Zero's value target z	Q-learning, DQN, actor-critic

Monte Carlo targets are unbiased and noisy. TD targets are steadier and start biased.

Model-free and model-based methods

A **model** predicts what the environment does next: P(s′ | s, a) and R(s, a, s′).

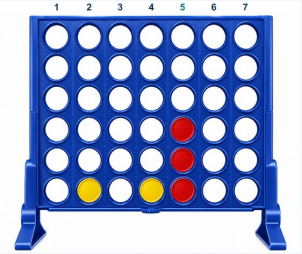
Model-free: learn from real games



Yellow learned that column 3 fails at s_5 by losing Game 1 (slide 14).

The tables, TD and Q-learning learn this way. Each lesson costs a real trial.

Model-based: try it on a copy first



With the rules as a model, Yellow can drop in column 3 on a copy, see Red's winning reply, and skip the loss.

Saves real trials. An error in the model becomes an error in the plan.

In Connect Four the rules are an exact model. The search later in this lecture uses them.

Where the model comes from

Known exactly

Game rules give the board after any drop, and who won.

Search is limited only by compute.

Connect Four here; AlphaGo Zero (2017)

Built from physics

A process simulator or digital twin: plasma chemistry, feature-scale profile, chamber thermal model.

Cheap trials. Wrong wherever the physics or calibration is incomplete.

A calibrated etch simulator

Learned from data

Fit P and R to logged transitions. The ½ for each Red reply on slide 44 is a model learned from two games.

Trustworthy only near its data. Two games is very little.

MuZero (2020) learned its own model for Go, chess, shogi and Atari

A policy tuned on a model inherits the model's errors in the states the policy visits.

Silver et al., Nature (2017); Schrittwieser et al., Nature (2020).

With an exact model, solve the Bellman equation directly

When the model is known and every state can be listed, values can be computed without playing any games.

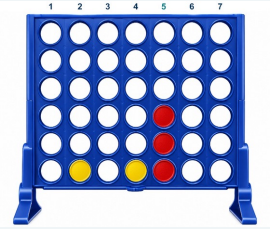
$$V(s) \leftarrow \max_a \Sigma_{s'} P(s' \mid s, a) \, [\, R(s, a, s') + \gamma \, V(s') \,]$$

Value iteration: the Bellman optimality equation, used as an update and applied to every state.

How value iteration works

1. Start every state at 0. A finished game gets its result.
2. Sweep: update every state from the current values of the states it can reach.
3. Repeat until the values stop changing. In a game, values spread back from the finished boards, one move per sweep.

In Connect Four



With the rules as the model, Yellow can check that column 3 lets Red win on the next drop.

Against a Red that takes wins, that branch is worth −1. Game 1 learned the same number by losing.

The catch: every sweep visits every state. The next slide counts the states in Connect Four.

Bellman (1957); Sutton and Barto (2018), Ch. 4. $\gamma = 1$.

What learns, and from what

Method	Stored and adjusted	Moves toward	Slides
Monte Carlo table	one Q per board and move	the average of finished-game returns	29–39
Search	nothing kept between moves; uses the model and the network	produces π_{search}, a training target	46–58
Network training	the weights θ of one two-headed network	π_{search} and the game result z	59–65
TD and Q-learning	Q, in a table or a network	the next reward plus V(next board)	75–78
Dynamic programming	a value for every state, computed from the model	the Bellman equation, swept over all states	81
Policy gradient, actor-critic	policy weights, plus critic weights	the return or the advantage	82–84

Each method in this lecture answers two questions: which numbers are stored, and which target they move toward.

How reinforcement learning reached AlphaGo

1988–92

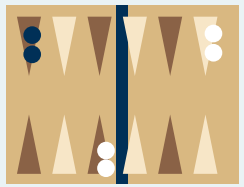


Later feedback

TD and Q-learning

Update earlier predictions using later feedback.

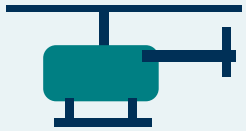
1992



TD-Gammon

A neural network learns backgammon through self-play.

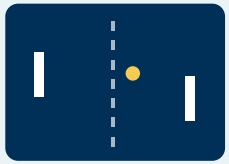
2006



Helicopter control

RL helps a real helicopter perform aerobatic maneuvers.

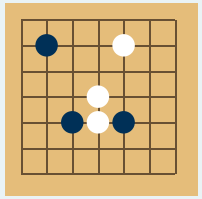
2015



DQN on Atari

Estimate action values from game pixels and scores.

2016–17



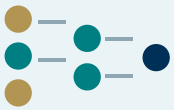
AlphaGo and Zero

Use deep networks, tree search, and self-play in Go.

The same idea runs through all five milestones. Use experience to improve future decisions.

Sources: Sutton and Barto (2018), Tesauro (1992), Abbeel et al. (2006), Mnih et al. (2015), and Silver et al. (2016, 2017).

Why RL took off in the 2010s



Deep networks

Pixels or board positions become predictions about moves and outcomes.



Faster compute

Accelerators made many network updates and search simulations practical.

Play → Score ↻

Repeatable experience

Game rules generate the next state and tell the agent who won.

The Lee Sedol match brought these advances to a worldwide audience.

These advances made it practical to learn from large amounts of experience.

Sources: Mnih et al. (2015), Silver et al. (2017), and Google DeepMind's AlphaGo match recap.

When reinforcement learning works well

The task needs repeated decisions, usable feedback, and a way to gather experience.

	Works well when	Struggles when
Decisions	Earlier actions change the options available later.	A single choice has no later decision to improve.
Feedback	The reward tracks the result we actually care about.	Rewards are rare, noisy, or reward the wrong behavior.
Experience	A simulator or safe trials provide many useful examples.	Trials are costly or risky, or the simulator misses important behavior.

Lecture 2 applies these conditions to plasma etch.

Key ideas from this lecture

The setup	An agent acts in an environment, modeled as a Markov decision process, and seeks the policy with the highest expected return.
Delayed reward	A move's worth shows up in the return. Learning carries a later reward back to the earlier move.
Values	V and Q predict returns. The Bellman equation ties each value to the value one step later.
Learning values	Monte Carlo averages finished returns. TD and Q-learning learn from the next estimate.
Exploration	Untried moves have to be tried. ε-greedy picks at random; the search bonus favors promising, rarely tried moves.
Models	A model lets the agent plan before acting. Its errors become errors in the plan.
Function approximation	Parameters θ replace the table and carry lessons to boards never seen.
Policy methods	Policy gradients and actor-critic adjust the policy from returns or advantages, knobs included.

Lecture 2 applies these ideas to plasma etch.