https://introml.mit.edu/

6.390 Intro to Machine Learning

Lecture 11: Reinforcement Learning

Shen Shen

April 26, 2024

Outline

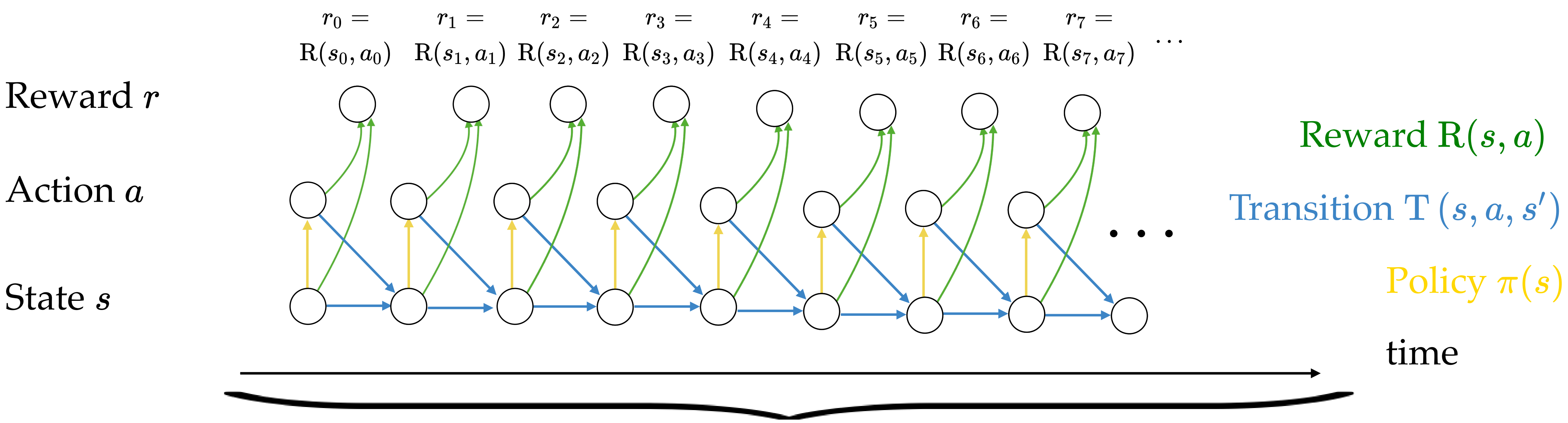
- Recap: Markov Decision Processes
- Reinforcement Learning Setup
    - What's changed from MDP?
- Model-based methods
- Model-free methods
    - (tabular) Q-learning
        - ϵ-greedy action selection
        - exploration vs. exploitation
    - (neural network) Q-learning
- RL setup again
    - What's changed from supervised learning?

MDP Definition and Goal

- $\mathcal{S}$: state space, contains all possible states s.
- $\mathcal{A}$: action space, contains all possible actions a.
- $\mathrm{T}\,(s, a, s')$: the probability of transition from state s to s' when action a is taken.
- $\mathrm{R}(s, a)$: a function that takes in the (state, action) and returns a reward.
- $\gamma \in [0, 1]$: discount factor, a scalar.

- $\pi(s)$: policy, takes in a state and returns an action.

Ultimate goal of an MDP: Find the "best" policy π.

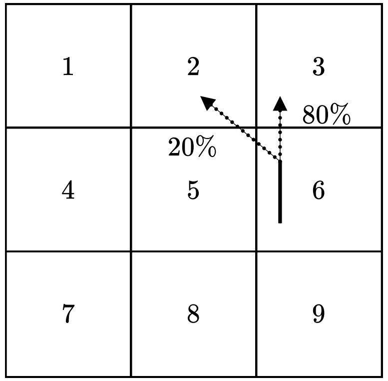


a trajectory (aka an experience or rollout) $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots)$

how "good" is a trajectory?

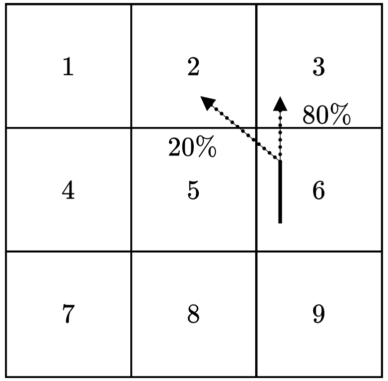
$$\mathrm{R}(s_0, a_0) + \gamma \mathrm{R}(s_1, a_1) + \gamma^2 \mathrm{R}(s_2, a_2) + \gamma^3 \mathrm{R}(s_3, a_3) + \gamma^4 \mathrm{R}(s_4, a_4) + \gamma^5 \mathrm{R}(s_5, a_5) + \gamma^6 \mathrm{R}(s_6, a_6) + \gamma^7 \mathrm{R}(s_7, a_7) \dots$$

Running example: Mario in a grid-world



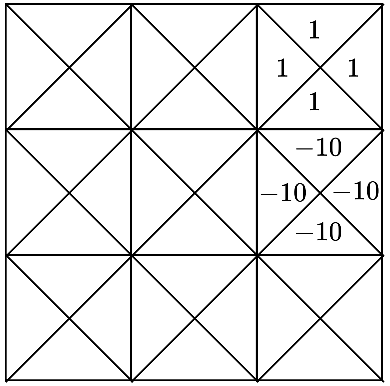
- 9 possible **states**
- 4 possible **actions**: {Up ↑, Down ↓, Left ←, Right →}
- almost all **transitions** are deterministic:
  - Normally, actions take Mario to the “intended” state.
    - E.g., in state (7), action “↑” gets to state (4)
  - If an action would've taken us out of this world, stay put
    - E.g., in state (9), action “→” gets back to state (9)
  - except, in state (6), action “↑” leads to two possibilities:
    - 20% chance ends in (2)
    - 80% chance ends in (3)

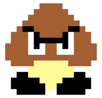
example cont'd



actions: {Up ↑, Down ↓, Left ←, Right →}

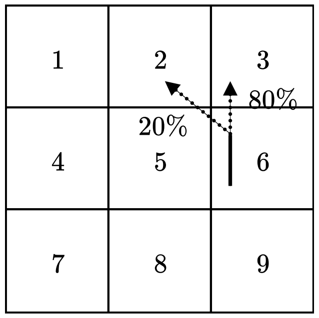
- (state, action) pair can get Mario **rewards**:



- In state (3), any action gets reward +1 
- In state (6), any action gets reward -10 
- Any other (state, action) pairs get reward 0

- goal is to find a gameplay **policy** strategy for Mario, to get maximum expected sum of discounted rewards, with a **discount facotor** $\gamma = 0.9$

Recall:

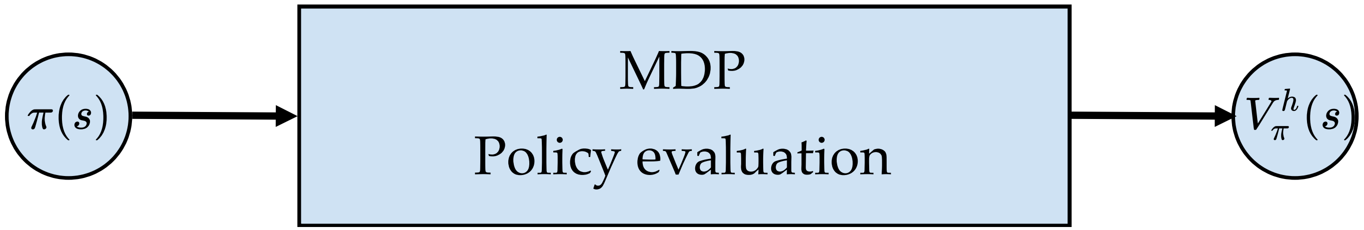


$\pi(s) = \text{“}\uparrow\text{”}, \ \forall s$

$\mathrm{R}(3, \uparrow) = 1$

$\mathrm{R}(6, \uparrow) = -10$

$\gamma = 0.9$



$V_\pi^0(s)$

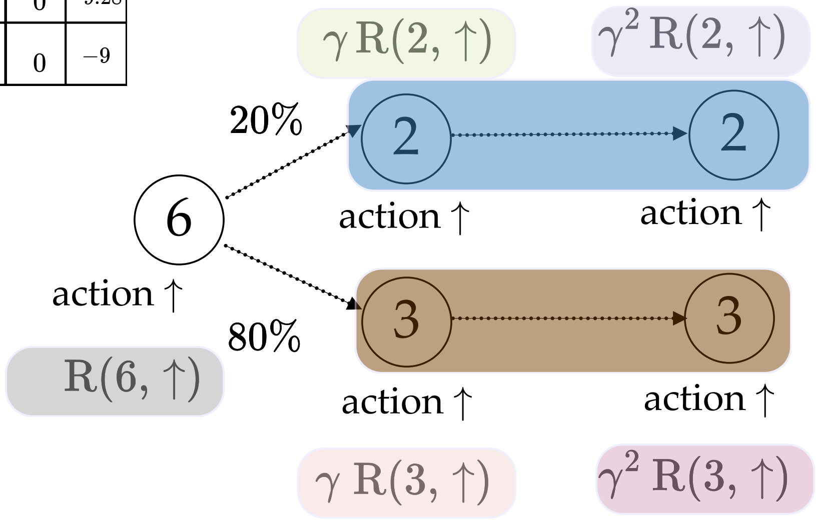
0	0	0
0	0	0
0	0	0

$V_\pi^1(s)$

0	0	1
0	0	−10
0	0	0

$V_\pi^2(s)$

0	0	1.9
0	0	−9.28
0	0	−9



Now, let's think about $V_\pi^3(6)$

$$
\begin{aligned}
V_\pi^3(6) &= \mathrm{R}(6, \uparrow) + 20\% \left[\gamma\,\mathrm{R}(2, \uparrow) + \gamma^2\,\mathrm{R}(2, \uparrow) \right] + 80\% \left[\gamma\,\mathrm{R}(3, \uparrow) + \gamma^2\,\mathrm{R}(3, \uparrow) \right] \\
&= \mathrm{R}(6, \uparrow) + 20\%\,\gamma \left[\mathrm{R}(2, \uparrow) + \gamma\,\mathrm{R}(2, \uparrow) \right] + 80\%\,\gamma \left[\mathrm{R}(3, \uparrow) + \gamma\,\mathrm{R}(3, \uparrow) \right] \\
&= \mathrm{R}(6, \uparrow) + 20\%\,\gamma\, V_\pi^2(2) + 80\%\,\gamma\, V_\pi^2(3)
\end{aligned}
$$

finite-horizon policy evaluation

For a given policy $\pi(s)$, the finite-horizon horizon-h (state) value functions are:

$$V_\pi^h(s) := \mathbb{E}\left[\sum_{t=0}^{h-1} \gamma^t \mathrm{R}\left(s_t, \pi\left(s_t\right)\right) \mid s_0 = s, \pi\right], \forall s$$

Bellman recursion

$$V_\pi^h(s) = \mathrm{R}(s, \pi(s)) + \gamma \sum_{s'} \mathrm{T}\left(s, \pi(s), s'\right) V_\pi^{h-1}\left(s'\right), \forall s$$

infinite-horizon policy evaluation

For any given policy $\pi(s)$, the infinite-horizon (state) value functions are

$$V_\pi(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathrm{R}\left(s_t, \pi\left(s_t\right)\right) \mid s_0 = s, \pi\right], \forall s$$

γ is now necessarily <1 for convergence too in general

Bellman equation

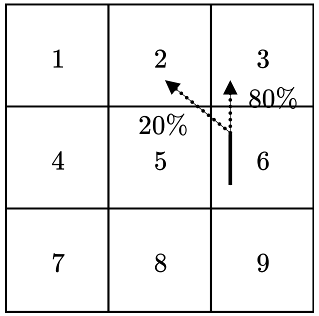
$$V_\pi(s) = \mathrm{R}(s, \pi(s)) + \gamma \sum_{s'} \mathrm{T}\left(s, \pi(s), s'\right) V_\pi\left(s'\right), \forall s$$

- $|\mathcal{S}|$ many linear equations

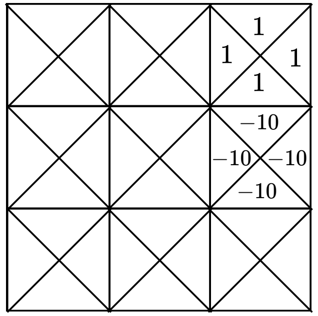
example: recursively finding $Q^h(s,a)$

Recall: $\gamma = 0.9$

States and one special transition:



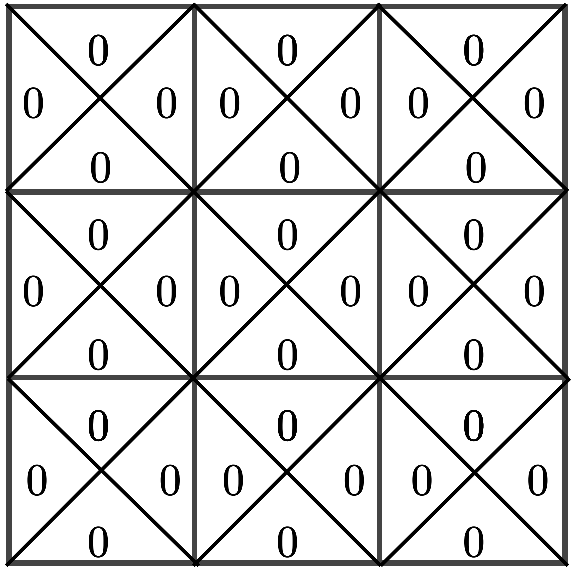
R(s,a)



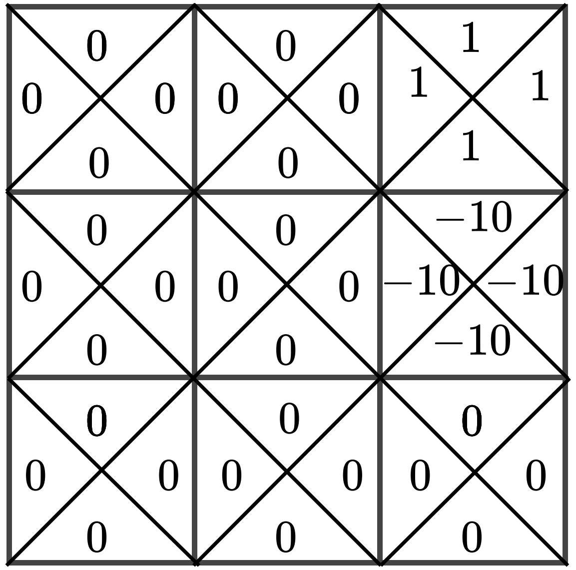
$Q^h(s,a)$ is the expected sum of discounted rewards for

- starting in state s,
- take action a, for one step
- act **optimally** there afterwards for the remaining $(h-1)$ steps

$Q^0(s,a)$



$Q^1(s,a)$

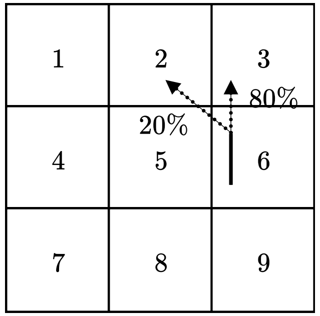


$Q^h(s,a)$ is the expected sum of discounted rewards for

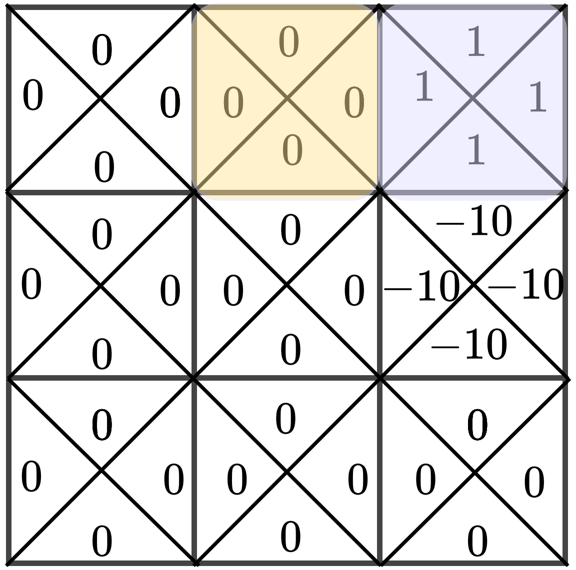
- starting in state s,
- take action a, for one step
- act **optimally** there afterwards for the remaining $(h-1)$ steps

Recall: $\gamma = 0.9$

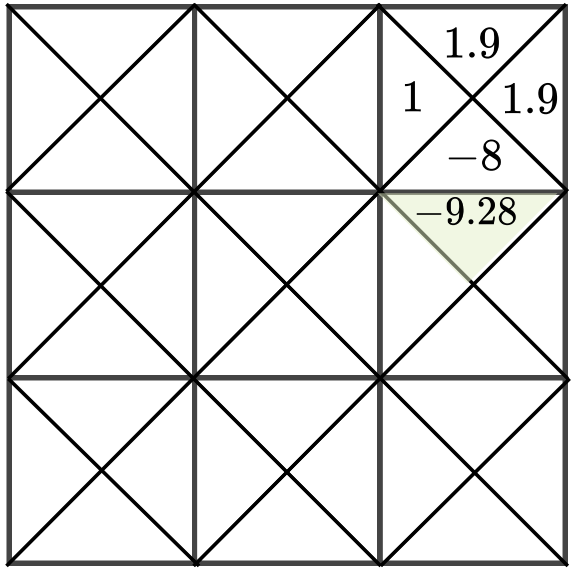
States and one special transition:



$Q^1(s,a) = \mathrm{R}(s,a)$



$Q^2(s,a)$



- receive $\mathrm{R}(6,\uparrow)$
- act **optimally** for one more timestep, at the next state s'
  - 20% chance, $s'=2$, act optimally, receive $\max_{a'} Q^1(2,a')$
  - 80% chance, $s'=3$, act optimally, receive $\max_{a'} Q^1(3,a')$

Let's consider $Q^2(6,\uparrow)$ $= \mathrm{R}(6,\uparrow) + \gamma[.2 \max_{a'} Q^1(2,a') + .8 \max_{a'} Q^1(3,a')]$

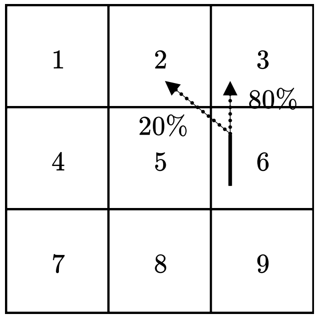
$$= -10 + .9[.2 * 0 + .8 * 1] = -9.28$$

$Q^h(s, a)$ is the expected sum of discounted rewards for

- starting in state s,
- take action a, for one step
- act **optimally** there afterwards for the remaining $(h-1)$ steps

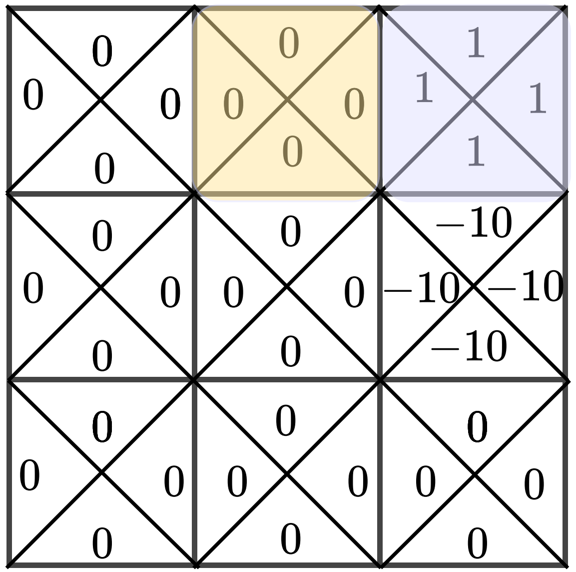
Recall: $\gamma = 0.9$

States and one special transition:

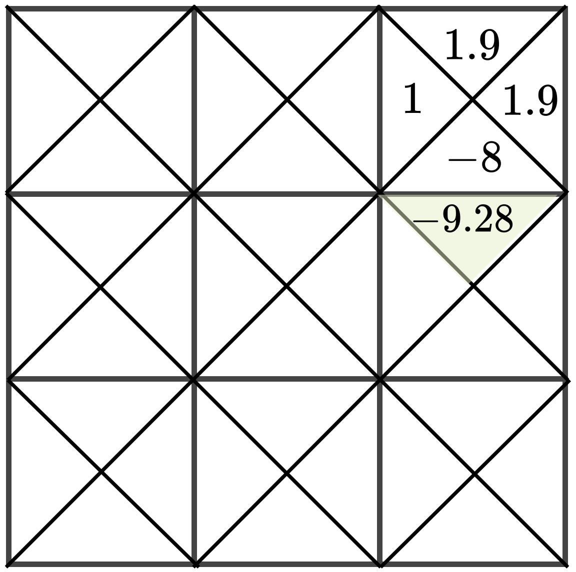


$Q^1(s, a)$

$= \mathrm{R}(s, a)$



$Q^2(s, a)$



$$Q^2(6, \uparrow) = \mathrm{R}(6, \uparrow) + \gamma[.2 \max_{a'} Q^1(2, a') + .8 \max_{a'} Q^1(3, a')]$$

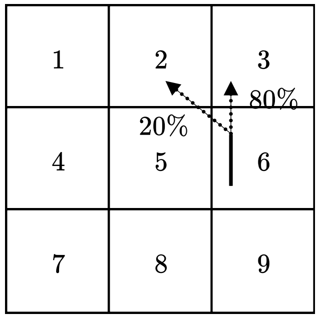
in general $$Q^h(s, a) = \mathrm{R}(s, a) + \gamma \sum_{s'} \mathrm{T}(s, a, s') \max_{a'} Q^{h-1}(s', a'), \forall s, a$$

$Q^h(s,a)$ is the expected sum of discounted rewards for

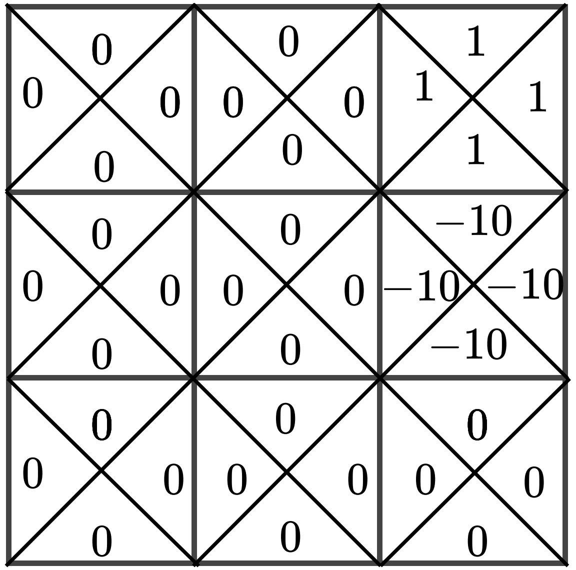
- starting in state s,
- take action a, for one step
- act **optimally** there afterwards for the remaining $(h-1)$ steps

Recall: $\gamma = 0.9$

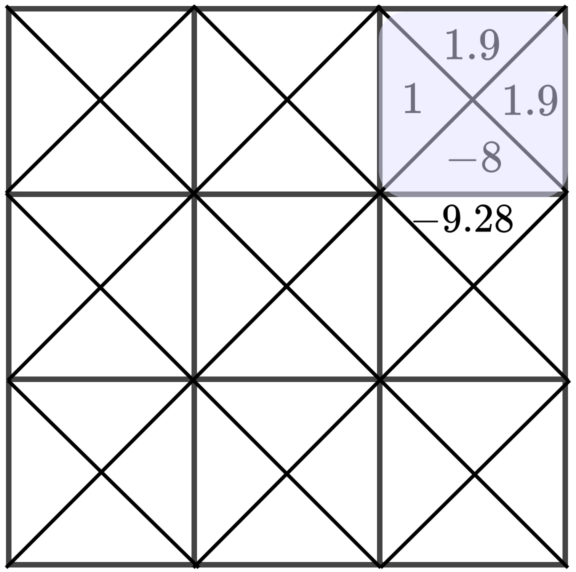
States and one special transition:



$Q^1(s,a)$



$Q^2(s,a)$



what's the optimal action in state 3, with horizon 2, given by $\pi_2^*(3) = ?$

either up or right

in general $\pi_h^*(s) = \arg\max_a Q^h(s,a), \forall s, h$

Given the finite horizon recursion

$$Q^h(s,a) = \mathrm{R}(s,a) + \gamma \sum_{s'} \mathrm{T}(s,a,s') \max_{a'} Q^{h-1}(s',a')$$

We should easily be convinced of the infinite horizon equation

$$Q(s,a) = \boxed{\mathrm{R}(s,a) + \gamma \sum_{s'} \mathrm{T}(s,a,s') \max_{a'} Q(s',a')}$$

Infinite-horizon Value Iteration

for $s \in \mathcal{S}, a \in \mathcal{A}:$

    $\mathrm{Q}_{\mathrm{old}}(\mathrm{s},\mathrm{a}) = 0$

while True:

    for $s \in \mathcal{S}, a \in \mathcal{A}:$

        $\mathrm{Q}_{\mathrm{new}}(s,a) \leftarrow \boxed{\mathrm{R}(s,a) + \gamma \sum_{s'} \mathrm{T}(s,a,s') \max_{a'} \mathrm{Q}_{\mathrm{old}}(s',a')}$

    if $\max_{s,a} |Q_{\mathrm{old}}(s,a) - Q_{\mathrm{new}}(s,a)| < \epsilon:$

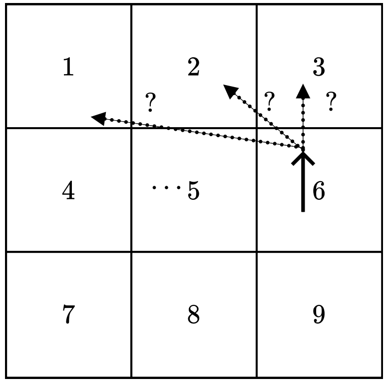
        return $\mathrm{Q}_{\mathrm{new}}$

    $\mathrm{Q}_{\mathrm{old}} \leftarrow \mathrm{Q}_{\mathrm{new}}$

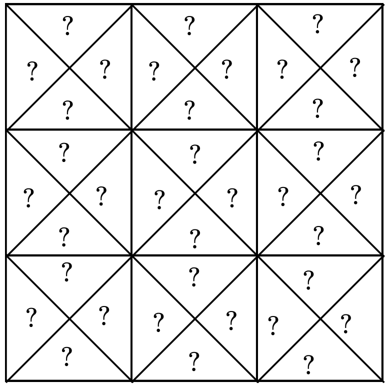
Outline

- Recap: Markov Decision Processes
- Reinforcement Learning Setup
  - What's changed from MDP?
- Model-based methods
- Model-free methods
  - (tabular) Q-learning
    - ϵ-greedy action selection
    - exploration vs. exploitation
  - (neural network) Q-learning
- RL setup again
  - What's changed from supervised learning?

Running example: Mario in a grid-world (the Reinforcement-Learning Setup)



- 9 possible **states**
- 4 possible **actions**: {Up ↑, Down ↓, Left ←, Right →}
- **all transitions** probabilities are **unknown**.



- (state, action) pair gets Mario **unknown rewards.**
- goal is to find a gameplay **policy** strategy for Mario, to get maximum expected sum of discounted rewards, with a **discount facotor** $\gamma = 0.9$

RL Definition and Goal

- $\mathcal{S}$: state space, contains all possible states s.
- $\mathcal{A}$: action space, contains all possible actions a.
- T (s, a, s') : the probability of transition from state s to s' when action a is taken.
- R(s, a) : a function that takes in the (state, action) and returns a reward.
- $\gamma \in [0, 1]$: discount factor, a scalar.

- $\pi(s)$: policy, takes in a state and returns an action.

Ultimate goal of an RL: Find the "best" policy π.

Outline

- Recap: Markov Decision Processes
- Reinforcement Learning Setup
  - What's changed from MDP?
- Model-based methods
- Model-free methods
  - (tabular) Q-learning
    - ϵ-greedy action selection
    - exploration vs. exploitation
  - (neural network) Q-learning
- RL setup again
  - What's changed from supervised learning?

(MDP)-Model-Based Methods (for solving RL)

Keep playing the game to approximate the unknown rewards and transitions.

Rewards are particularly easy:

e.g. by observing what reward r received from being in state 6 and take $\uparrow$ action, we know $\mathrm{R}(6, \uparrow)$

Transitions are a bit more involved but still simple:

e.g. play the game 1000 times, count the # of times (we started in state 6, take $\uparrow$ action, end in state 2), then, roughly, $\mathrm{T}(6, \uparrow, 2) = (\text{that count}/1000)$

Now, with R, T estimated, we're back in MDP setting.

In Reinforcement Learning:

- *Model* typically means MDP tuple $\langle \mathcal{S}, \mathcal{A}, \mathrm{T}, \mathrm{R}, \gamma \rangle$
- The learning objective is not referred to as *hypothesis* explicitly, we simply just call it the *policy*.

Outline

- Recap: Markov Decision Processes
- Reinforcement Learning Setup
  - What's changed from MDP?
- Model-based methods
- Model-free methods
  - (tabular) Q-learning
    - ϵ-greedy action selection
    - exploration vs. exploitation
  - (neural network) Q-learning
- RL setup again
  - What's changed from supervised learning?

How do we learn a good policy without learning transition or rewards explicitly?

We kinda already know a way: Q functions!

2.4) (Recall from MDP lab)

We switch to an infinite-horizon scenario. For our stochastic machine, here is the infinite-horizon Q function (computed via value iteration) for γ near 1.

```
              wash        paint       eject
dirty    [[ 2.32274541 -0.70048204  0.         ]
clean     [ 2.32274541  5.71581775  0.         ]
painted   [ 2.32274541  6.9        10.         ]
ejected   [ 0.          0.          0.         ]]
```

What is the optimal thing to do with a clean object?

What will you do if it becomes dirty?

Does this optimal policy make intuitive sense?

So once we have "good" Q values, we can find optimal policy easily.

But didn't we calculate this Q-table via value iteration using transition and rewards explicitly?

Indeed, recall that, in MDP:

- Finite horizon recursion

$$Q^h(s,a) = \mathrm{R}(s,a) + \gamma \sum_{s'} \mathrm{T}(s,a,s') \max_{a'} Q^{h-1}(s',a')$$

- Infinite horizon equation

$$Q(s,a) = \boxed{\mathrm{R}(s,a) + \gamma \sum_{s'} \mathrm{T}(s,a,s') \max_{a'} Q(s',a')}$$

- Infinite-horizon Value Iteration

1. **for** $s \in \mathcal{S}, a \in \mathcal{A}$:
2. $\quad \mathrm{Q}_{\text{old}}(\mathrm{s}, \mathrm{a}) = 0$
3. **while** True:
4. $\quad$ **for** $s \in \mathcal{S}, a \in \mathcal{A}$:
5. $\quad\quad \mathrm{Q}_{\text{new}}(s,a) \leftarrow \boxed{\mathrm{R}(s,a) + \gamma \sum_{s'} \mathrm{T}(s,a,s') \max_{a'} \mathrm{Q}_{\text{old}}(s',a')}$
6. $\quad$ **if** $\max_{s,a} |Q_{\text{old}}(s,a) - Q_{\text{new}}(s,a)| < \epsilon$:
7. $\quad\quad$ **return** Q_{new}
8. $\quad \mathrm{Q}_{\text{old}} \leftarrow \mathrm{Q}_{\text{new}}$

- value iteration relied on having full access to R and T

$$\mathrm{Q}_{\text{new}}(s,a) \leftarrow \mathrm{R}(s,a) + \gamma \sum_{s'} \mathrm{T}(s,a,s') \max_{a'} \mathrm{Q}_{\text{old}}(s',a')$$

- hmm... perhaps, we could simulate (s,a), observe r and s', and just use

$$r \quad + \gamma \max_{a'} \mathrm{Q}_{\text{old}}(s',a')$$

as the proxy for the r.h.s. assignment?

- BUT, this is basically saying the realized s' is the only possible next state; pretty rough! We'd override any previous "learned" Q values.

e.g. $\mathrm{Q}_{\text{new}}(6,\uparrow) \leftarrow -10 + \gamma \max_{a'} \mathrm{Q}_{\text{old}}(2,a')$ $\qquad$ $\mathrm{Q}_{\text{new}}(6,\uparrow) \leftarrow -10 + \gamma \max_{a'} \mathrm{Q}_{\text{old}}(3,a')$

- **better** way is to smoothly keep track of what's our old belief with new evidence:

$$\mathrm{Q}_{\text{new}}(s,a) \leftarrow \boxed{(1-\alpha)\mathrm{Q}_{\text{old}}(s,a) + \alpha\left(r + \gamma \max_{a'} \mathrm{Q}_{\text{old}}(s',a')\right)}$$

old belief $\qquad$ learning rate $\qquad$ target

VALUE-ITERATION $(\mathcal{S}, \mathcal{A}, \mathrm{T}, \mathrm{R}, \gamma, \epsilon)$

for $s \in \mathcal{S}, a \in \mathcal{A}$:

$\quad \mathrm{Q}_{\mathrm{old}}(\mathrm{s}, \mathrm{a}) = 0$

while True:

$\quad$ **for** $s \in \mathcal{S}, a \in \mathcal{A}$:

$\qquad \boxed{\mathrm{Q}_{\mathrm{new}}(s, a) \leftarrow \mathrm{R}(s, a) + \gamma \sum_{s'} \mathrm{T}(s, a, s') \max_{a'} \mathrm{Q}_{\mathrm{old}}(s', a')}$

$\quad$ **if** $\max_{s,a} |Q_{\mathrm{old}}(s, a) - Q_{\mathrm{new}}(s, a)| < \epsilon$:

$\qquad$ **return** $\mathrm{Q}_{\mathrm{new}}$

$\quad \mathrm{Q}_{\mathrm{old}} \leftarrow \mathrm{Q}_{\mathrm{new}}$

"calculating"

Q-LEARNING $(\mathcal{S}, \mathcal{A}, \gamma, \mathrm{s}_0, \alpha)$

for $s \in \mathcal{S}, a \in \mathcal{A}$:

$\quad \mathrm{Q}_{\mathrm{old}}(\mathrm{s}, \mathrm{a}) = 0$

$s \leftarrow s_0$

while True:

$\quad a \leftarrow \mathrm{select\_action}(s, Q_{\mathrm{old}}(s, a))$

$\quad r, s' = \mathrm{execute}(a)$

$\quad \boxed{\mathrm{Q}_{\mathrm{new}}(s, a) \leftarrow (1 - \alpha)\mathrm{Q}_{\mathrm{old}}(s, a) +}$

$\alpha (r + \gamma \max_{a'} \mathrm{Q}_{\mathrm{old}}(s', a'))$

$\quad s \leftarrow s'$

$\quad$ **if** $\max_{s,a} |Q_{\mathrm{old}}(s, a) - Q_{\mathrm{new}}(s, a)| < \epsilon$:

$\qquad$ **return** $\mathrm{Q}_{\mathrm{new}}$

$\quad \mathrm{Q}_{\mathrm{old}} \leftarrow \mathrm{Q}_{\mathrm{new}}$

"estimating"

Q-LEARNING $(\mathcal{S}, \mathcal{A}, \gamma, \mathrm{s}_0, \alpha)$

1. **for** $s \in \mathcal{S}, a \in \mathcal{A}$:
2. $\quad \mathrm{Q}_{\text{old}}(\mathrm{s}, \mathrm{a}) = 0$
3. $s \leftarrow s_0$
4. **while** True:
5. $\quad a \leftarrow \text{select\_action}\left(s, Q_{\text{old}}(s, a)\right)$
6. $\quad r, s' = \text{execute}(a)$
7. $\quad \mathrm{Q}_{\text{new}}(s, a) \leftarrow (1-\alpha)\mathrm{Q}_{\text{old}}(s, a) + \alpha\left(r + \gamma \max_{a'} \mathrm{Q}_{\text{old}}(s', a')\right)$
8. $\quad s \leftarrow s'$
9. $\quad$ **if** $\max_{s,a} \left|Q_{\text{old}}(s, a) - Q_{\text{new}}(s, a)\right| < \epsilon$:
10. $\quad\quad$ **return** Q_{new}
11. $\quad \mathrm{Q}_{\text{old}} \leftarrow \mathrm{Q}_{\text{new}}$

- Remarkably, still can converge. (So long $\mathcal{S}, \mathcal{A}$ are finite; we visit every state and action infinity-many times; and α decays.

- Line 7 :

$$\mathrm{Q}_{\text{new}}(s, a) \leftarrow (1-\alpha)\mathrm{Q}_{\text{old}}(s, a) + \alpha\left(r + \gamma \max_{a'} \mathrm{Q}_{\text{old}}(s', a')\right)$$

is equivalently:

$$\mathrm{Q}_{\text{new}}(s, a) \leftarrow \mathrm{Q}_{\text{old}}(s, a) + \alpha\left(\left[r + \gamma \max_{a'} \mathrm{Q}_{\text{old}}(s', a')\right] - \mathrm{Q}_{\text{old}}(s, a)\right)$$

$$\text{old belief} + \text{learning rate}\left(\text{target} - \text{old belief}\right)$$

pretty similar to SGD.

- Line 5, a sub-routine.

- If our Q values are estimated quite accurately (nearly converged to the true Q values), then should act greedily
  - $\arg\max_a Q^h(s, a)$, as we did in MDP.

- During learning, especially in early stages, we'd like to explore.

- ϵ-greedy action selection strategy:
  - with probability $1 - \epsilon$, choose $\arg\max_{\mathrm{a}} \mathrm{Q}(s, \mathrm{a})$
  - with probability ϵ, choose an action $a \in \mathcal{A}$ uniformly at random

- Benefit: get to observe more diverse (s,a) consequences.

- *exploration vs. exploitation.*

Outline

- Recap: Markov Decision Processes
- Reinforcement Learning Setup
  - What's changed from MDP?
- Model-based methods
- Model-free methods
  - (tabular) Q-learning
    - ϵ-greedy action selection
    - exploration vs. exploitation
  - (neural network) Q-learning
- RL setup again
  - What's changed from supervised learning?

- Q-learning only is kinda sensible for tabular setting.
- What do we do if $\mathcal{S}$ and/or $\mathcal{A}$ are large (or continuous)?

- Recall from Q-learning algorithm, key line 7 :

$$\mathrm{Q}_{\text{new}}(s, a) \leftarrow (1-\alpha) \mathrm{Q}_{\text{old}}(s, a) + \alpha \left(r + \gamma \max_{a'} \mathrm{Q}_{\text{old}}\left(s', a'\right)\right)$$

is equivalently:

$$\mathrm{Q}_{\text{new}}(s, a) \leftarrow \mathrm{Q}_{\text{old}}(s, a) + \alpha\left(\left[r + \gamma \max_{a'} \mathrm{Q}_{\text{old}}\left(s', a'\right)\right] - \mathrm{Q}_{\text{old}}(s, a)\right)$$

old belief + learning rate (target − old belief)

- Can be interpreted as we're minimizing:

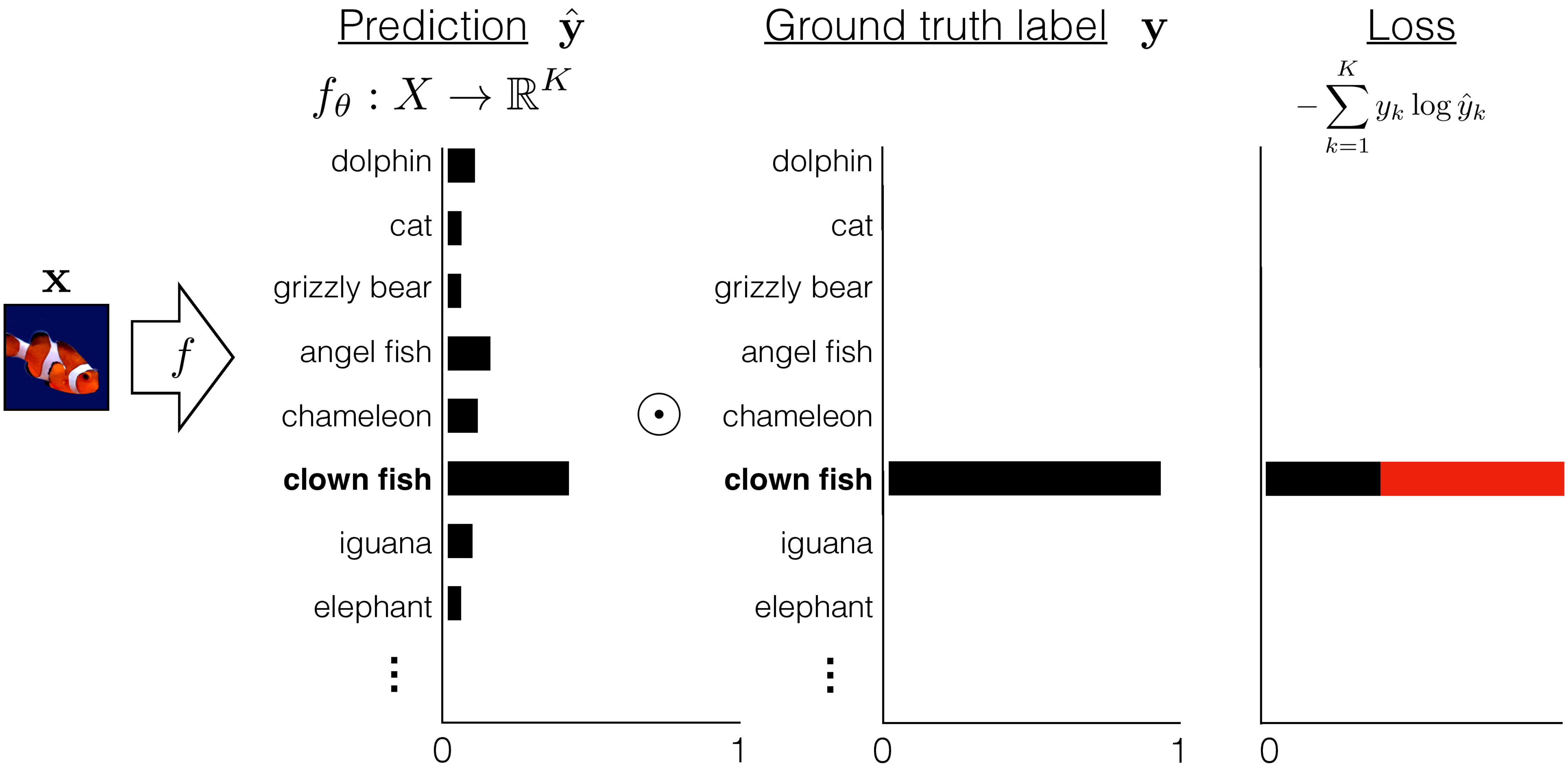
$$\left(Q(s, a) - \left(r + \gamma \max_{a'} Q\left(s', a'\right)\right)\right)^2$$

via gradient method!

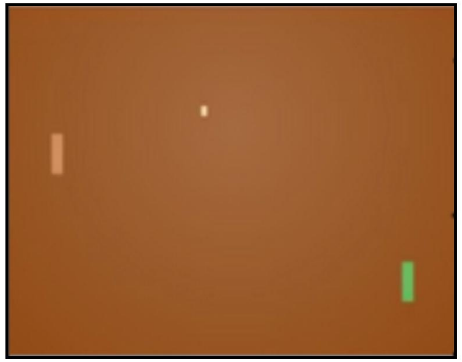
Outline

- Recap: Markov Decision Processes
- Reinforcement Learning Setup
    - What's changed from MDP?
- Model-based methods
- Model-free methods
    - (tabular) Q-learning
        - ϵ-greedy action selection
        - exploration vs. exploitation
    - (neural network) Q-learning
- RL setup again
    - What's changed from supervised learning?

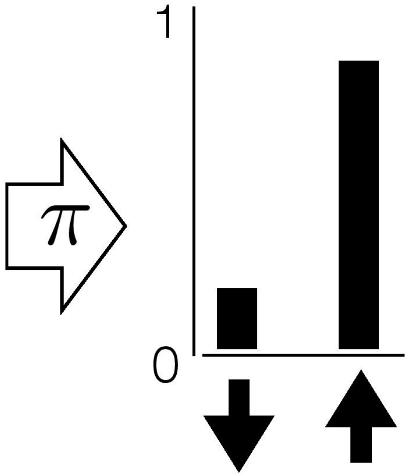
Supervised learning

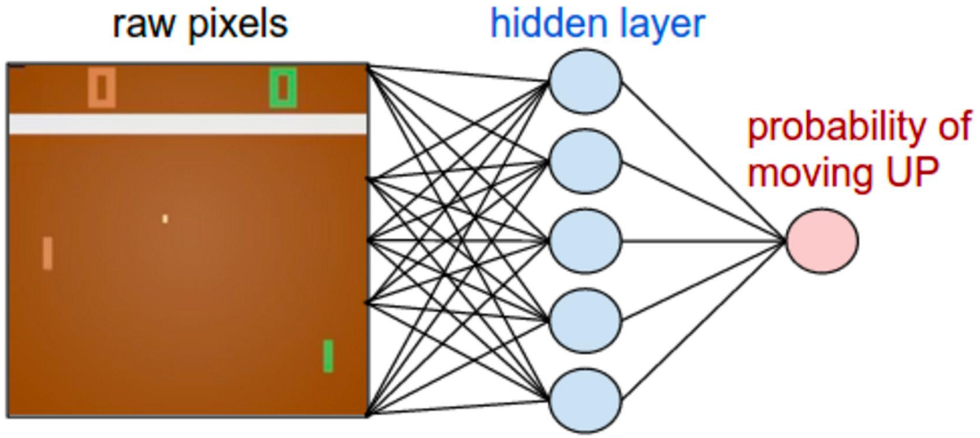


- If explicit “good” state-action pair is given, also supervised learning.
- Behavior cloning or imitation learning.
- But what if no explicit guide?



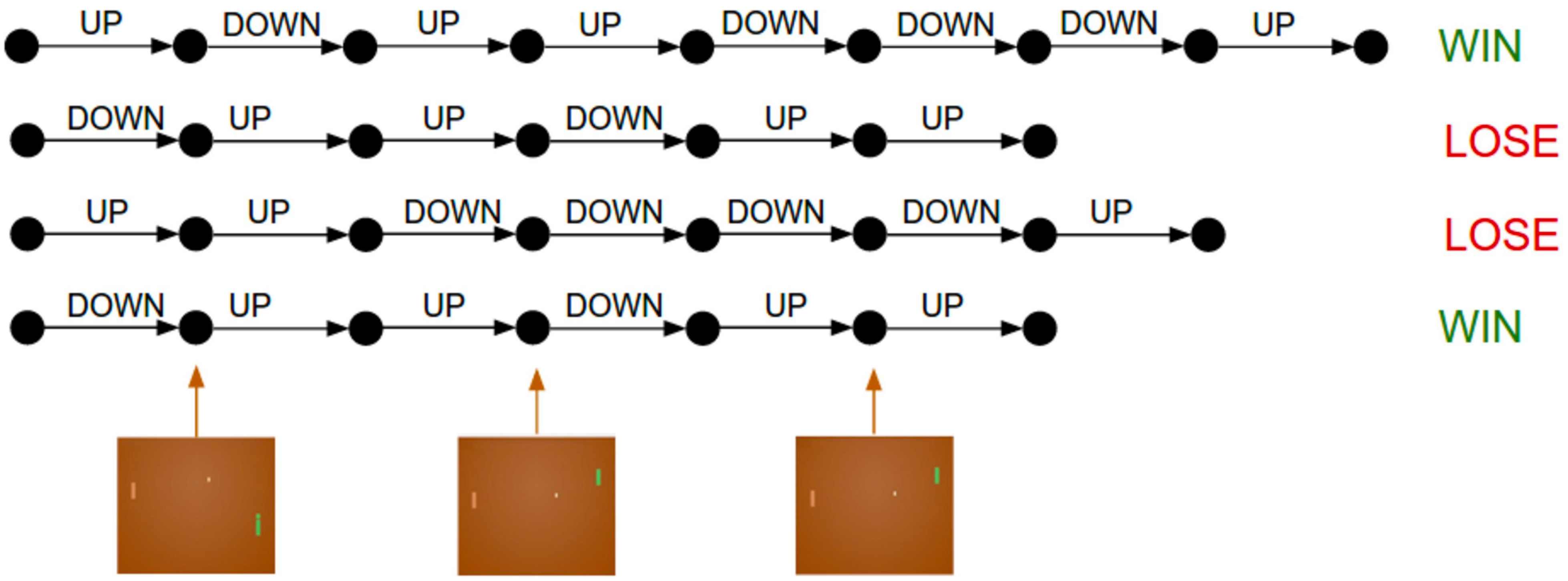
Pong





[Adapted from Andrej Karpathy: http://karpathy.github.io/2016/05/31/rl/]

- If no direct supervision is available?
- Strictly RL setting. Interact, observe and get data. Use rewards/value as "coy" supervision signal.



[Adapted from Andrej Karpathy: http://karpathy.github.io/2016/05/31/rl/]

We'd appreciate your feedback on the lecture.

Thanks!