Model-Free Reinforcement Learning

CS440/ECE448

Mark Hasegawa-Johnson

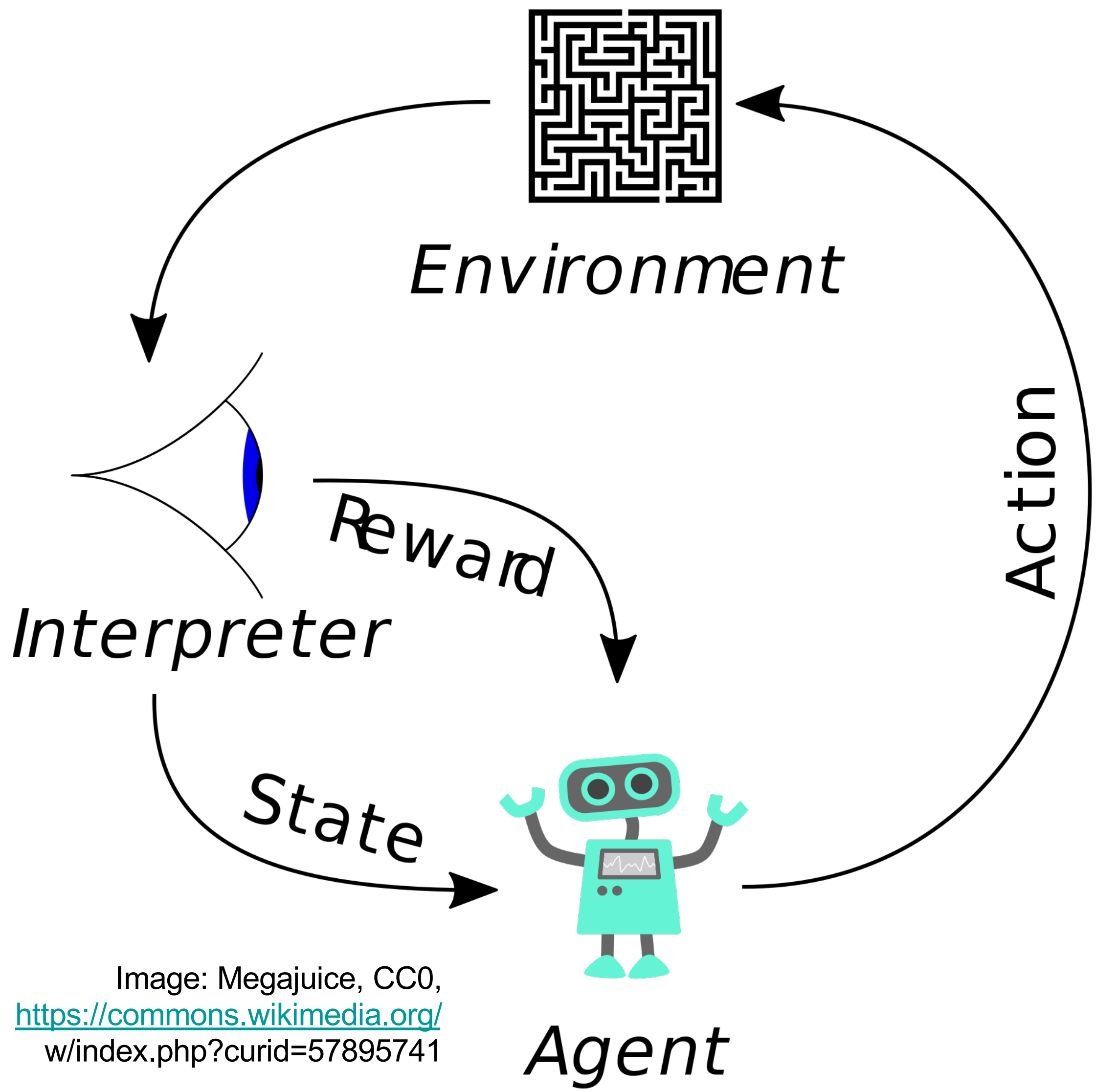


Image: Megajuice, CC0, https://commons.wikimedia.org/w/index.php?curid=57895741

Outline

- Q-learning: What to estimate
- Q-local: The information available to us
- TD-learning: Q-learning with smoothing
- SARSA: on-policy Q-learning

What should reinforcement learning learn?

Last time:

- Model-based learning: $P(s'|s, a)$

Today:

- Q-learning: $q(s, a)$, the quality of action a in state s

Monday:

- Policy gradient: estimate a stochastic policy $P(A_t = a|S_t = s)$

The Quality of an Action

Q-learning splits Bellman's equation into two parts:

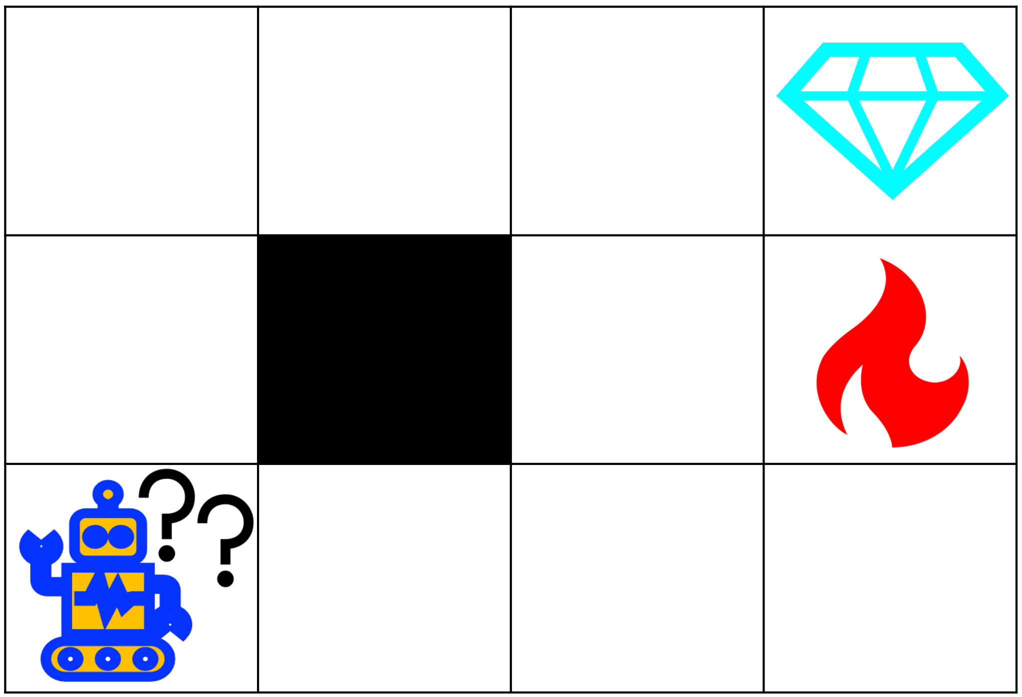
$$u(s) = r(s) + \gamma \max_{a \in \mathcal{A}} \sum_{s'} P(s'|s,a)u(s')$$

…becomes…

$$q(s,a) = r(s) + \gamma \sum_{s'} P(s'|s,a)u(s')$$

$$u(s) = \max_{a \in \mathcal{A}} q(s,a)$$

Example: Gridworld



$$r(s) = \begin{cases} +1 & s = (4,3) \\ -1 & s = (4,2) \\ -0.04 & \text{otherwise} \end{cases}$$

$$P(s'|s,a) = \begin{cases} 0.8 & \text{intended} \\ 0.1 & \text{fall left} \\ 0.1 & \text{fall right} \end{cases}$$

$$\gamma = 1$$

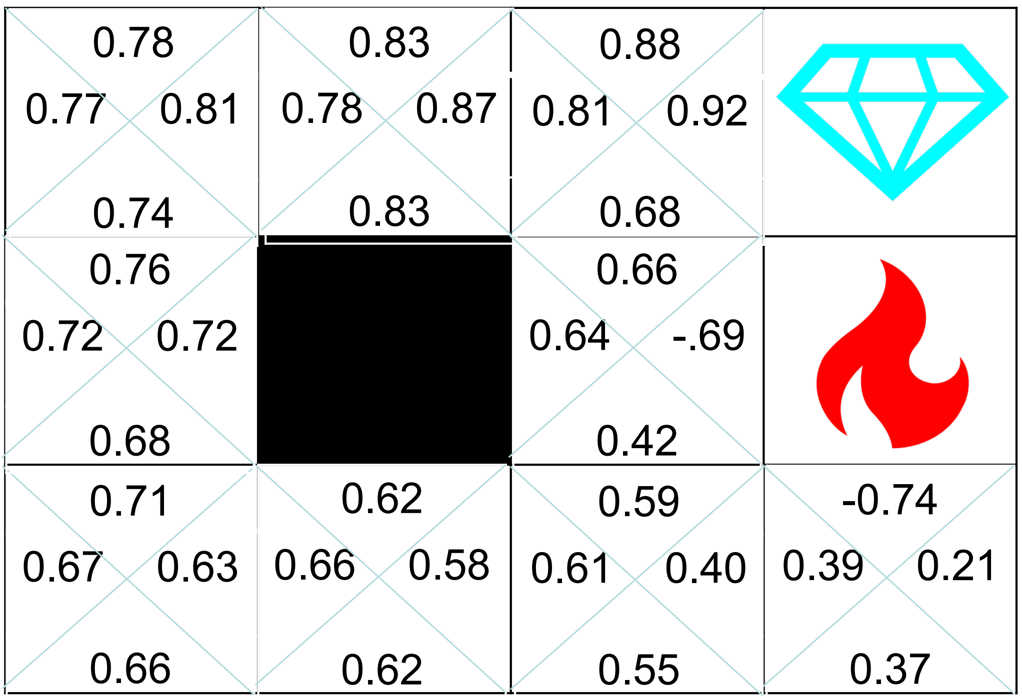
Gridworld: Utility of each state

$$u(s) = r(s) + \gamma \max_{a \in \mathcal{A}} \sum_{s'} P(s'|s,a)u(s')$$

0.81	0.87	0.92	
0.76		0.66	
0.71	0.66	0.61	0.39

(Calculated using value iteration.)

Gridworld: The Q-function



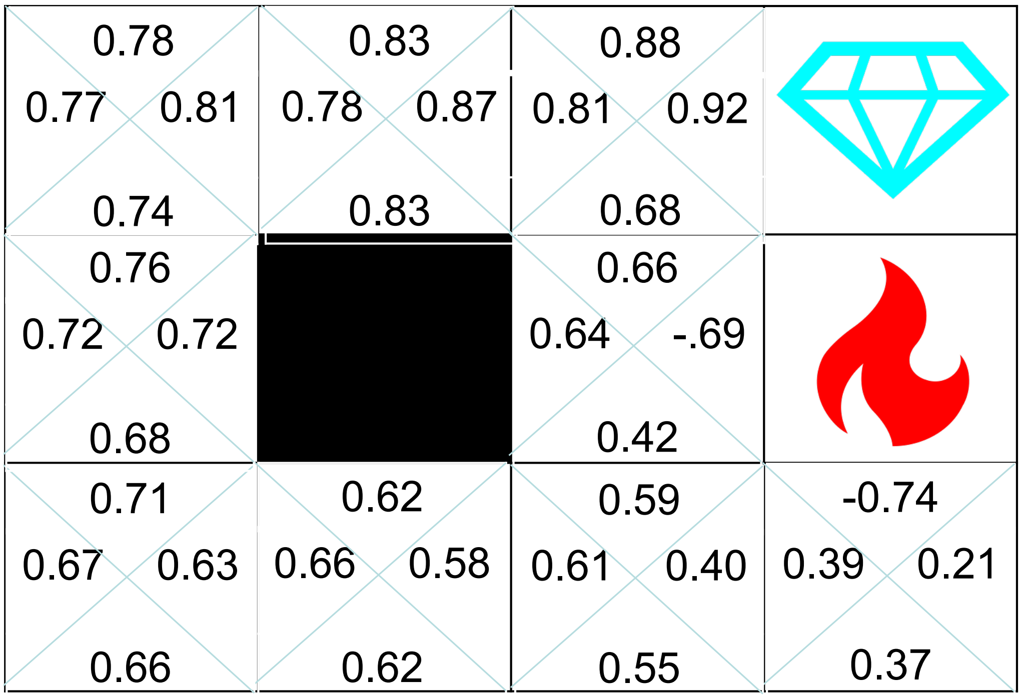
Calculated using a two-step value iteration:

$$q(s,a) = r(s) + \gamma \sum_{s'} P(s'|s,a)u(s')$$

$$u(s) = \max_{a \in \mathcal{A}} q(s,a)$$

Gridworld: Relationship between Q and U

$$u(s) = \max_{a \in \mathcal{A}} q(s, a)$$



0.81	0.87	0.92	
0.76		0.66	
0.71	0.66	0.61	0.39

Q-learning

- In the reinforcement learning scenario, we don't know $P(s'|s,a)$. We just want to play the game, and observe our earned reward, and from it, estimate $q(s,a)$.
- On the t^{th} iteration of q-learning, suppose that we have an estimate $q_t(s,a)$. We can use that as follows:

Try action a_t in state s_t. Measure the reward r_t, and observe the estimated utility of the state we end up in $u_t(s_{t+1})$.

Why?

- Notice that, if there are N states and M actions, $P(s'|s,a)$ is a table that contains MN^2 entries
  - Takes a lot of memory to store it
  - Takes a lot of training data to learn all those parameters
- Q-learning learns $q(s,a)$, which has only MN
  - For example, if N=1000 and M=4, Q-learning requires 1000X less training data

Outline

- Q-learning: What to estimate
- Q-local: The information available to us
- TD-learning: Q-learning with smoothing
- SARSA: on-policy Q-learning

But how can you learn?

Remember in model-based learning we learned:

$$P(s_{t+1}|s_t, a_t) = \frac{N(s_t, a_t, s_{t+1}) + k}{\sum_{s' \in \mathcal{S}} N(s_t, a_t, s') + k|\mathcal{S}|}$$

But how can you learn?

In Q-learning, we can learn:

$$q(s_t, a_t) = r(s_t) + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) u(s_{t+1})$$

$$= r(s_t) + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) \max_{a \in \mathcal{A}} q(s_{t+1}, a)$$

…except that we don't know $P(s_{t+1}|s_t, a_t)$!

But how can you learn?

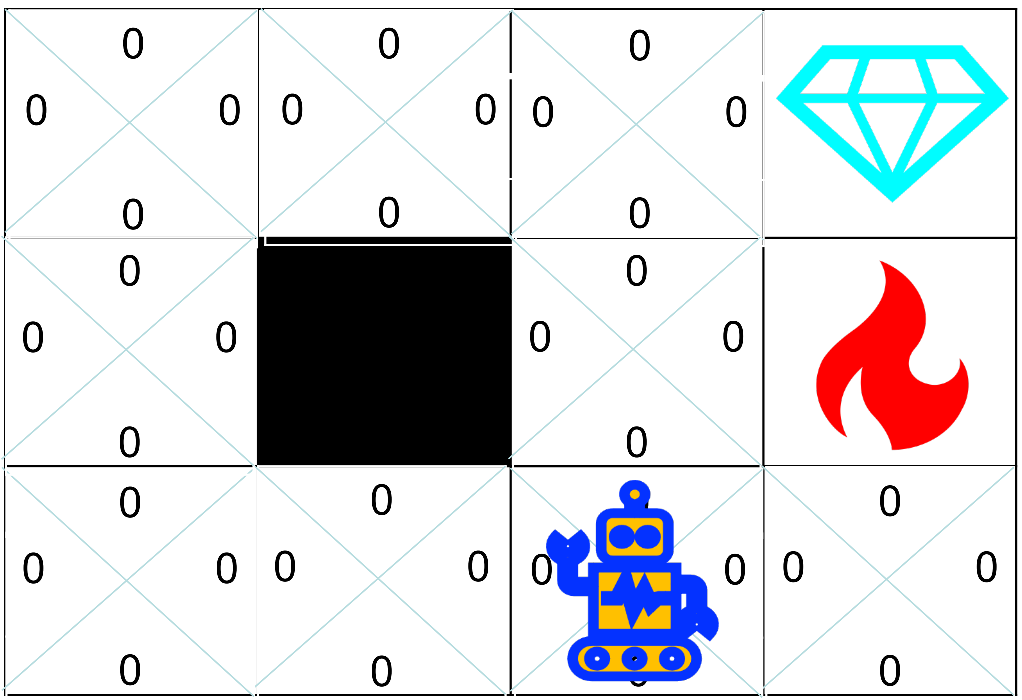
Simplifying assumption: Assume that $P(s_{t+1}|s_t, a_t) \approx 1$ for the s_{t+1} we observe:

$$q(s_t, a_t) = r(s_t) + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) \max_{a \in \mathcal{A}} q(s_{t+1}, a)$$
$$\approx q_{\text{local}}(s_t, a_t, r_t, s_{t+1})$$

…which we define as:

$$q_{\text{local}}(s_t, a_t, r_t, s_{t+1}) = r_t + \gamma \max_{a \in \mathcal{A}} q(s_{t+1}, a)$$

Example: Gridworld



Suppose we start out with no knowledge, so we assume $\mathbf{q(s,a) = 0}$ for all states and actions.

Robot starts out in state s_t=$(2,0)$.

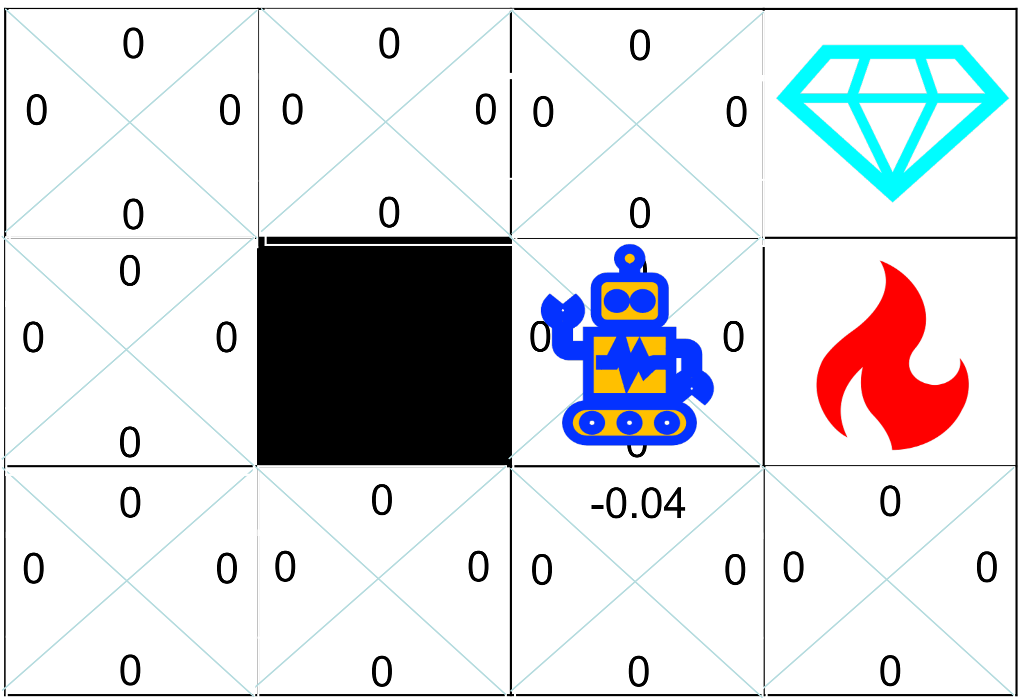
Robot receives a reward of r_t=-0.04.

Example: Gridworld

Robot starts out in state s_t=$(2{,}0)$.

Robot receives a reward of r_t=-0.04.

Robot tries to move UP, ends up in $s_{t+1} = (2{,}1)$.



Now we update $q_{\mathrm{local}}((3{,}1), \mathrm{UP})$:

$$q_{\mathrm{local}}((3{,}1), \mathrm{UP}) = r((3{,}1)) + 0 = -0.04$$

q-local, the short-time estimate

$$q(s_t, a_t) = r(s_t) + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) \max_{a \in \mathcal{A}} q(s_{t+1}, a)$$

$$q_{local}(s_t, a_t, r_t, s_{t+1}) = r_t + \gamma \max_{a \in \mathcal{A}} q(s_{t+1}, a)$$

Q-local approximates the true quality of an action as:

- Instead of summing over $P(s'|s, a)$, just set $s' = s_{t+1}$, i.e., whatever state followed s_t.
- Instead of the true value of $q(s, a)$ on the right-hand-side, use our current estimate

Outline

- Q-learning: What to estimate
- Q-local: The information available to us
- TD-learning: Q-learning with smoothing
- SARSA: on-policy Q-learning

TD learning

$$q_{local}(s_t, a_t, , r_t, s_{t+1}) = r_t + \gamma \max_{a \in \mathcal{A}} q(s_{t+1}, a)$$

Problem: NOISY!

- s_{t+1} is random, and
- $\max_{a \in \mathcal{A}} q(s_{t+1}, a)$ is not the real value of q, only our current estimate, therefore
- $q_{local}(s_t, a_t, , r_t, s_{t+1})$ might be very far away from $q(s, a)$!

TD learning

Solutions:

1. If we're measuring using a table: interpolate, using a small learning rate η that's $0 < \eta < 1$:
   $q(s_t, a_t) \leftarrow q(s_t, a_t) + \eta\big(q_{local}(s_t, a_t, r_t, s_{t+1}) - q(s_t, a_t)\big)$

2. If we're measuring using a neural net, with parameters$\boldsymbol{w}$: use just one gradient update step, so that $\boldsymbol{w}$ becomes the average over many successive gradient steps:
   $$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\partial}{\partial \boldsymbol{w}} \frac{1}{2} \big(q_t(s_t, a_t) - q_{local}(s_t, a_t, , r_t, s_{t+1})\big)^2$$

TD learning

$q_{local}(s_t, a_t, r_t, s_{t+1}) - q_t(s_t, a_t)$ is called the "time difference" or TD.

1. If the TD is positive, it means action a_t was **<u>better</u>** than we expected, so $q_{t+1}(s_t, a_t) = q_t(s_t, a_t) + \eta TD$ is an increase.
2. If the TD is negative, it means action a_t was **<u>worse</u>** than we expected, so $q_{t+1}(s_t, a_t) = q_t(s_t, a_t) + \eta TD$ is a decrease.

TD learning

Putting it all together, here's the whole TD learning algorithm:

1. When you reach state s, use your current exploration versus exploitation policy to choose some action.
2. Observe the state s_{t+1} that you end up in, and the reward you receive, and then calculate q-local:

$$q_{local}(s_t, a_t, r_t, s_{t+1}) = r_t + \gamma \max_{a\prime \in \mathcal{A}} q(s_{t+1}, a')$$

3. Calculate the time difference, and update:

$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \eta\big(q_{local}(s_t, a_t, r_t, s_{t+1}) - q(s_t, a_t)\big)$$

or:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\partial}{\partial \boldsymbol{w}} \frac{1}{2} \big(q_t(s_t, a_t) - q_{local}(s_t, a_t, , r_t, s_{t+1})\big)^2$$

Outline

- Q-learning: What to estimate
- Q-local: The information available to us
- TD-learning: Q-learning with smoothing
- SARSA: on-policy Q-learning

TD learning is an off-policy learning algorithm

- TD learning is called an off-policy learning algorithm because it assumes an action

$$\underset{a' \in \mathcal{A}}{\operatorname{argmax}}\, q(s_{t+1}, a')$$

…which may be different from your real action (e.g., you might explore instead of exploiting).

- Thus, TD-learning might not converge to real q-functions, because it focuses all the learning on the actions you think are best, and doesn't learn very much about actions whose quality you don't yet know.

On-policy learning: SARSA

We can create an “on-policy learning” algorithm by deciding in advance which action (a_{t+1}) we’ll perform in state s_{t+1}, and then using that action in the update equation:

1. Assume that you’re currently in state s_t, and you’ve already chosen action a_t.
2. Observe the state s_{t+1} that you end up in, and then use your current exploration vs. exploitation policy to already choose a_{t+1}!
3. Calculate q-local and the update equation as:

$$q_{\text{local}}(s_t, a_t, r_t, s_{t+1}, a_{t+1}) = r_t + \gamma q(s_{t+1}, a_{t+1})$$
$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \eta\big(q_{local}(s_t, a_t, r_t, s_{t+1}, a_{t+1}) - q(s_t, a_t)\big)$$

Quiz

Try the quiz!

Summary: Q-learning

Q-learning:

$$q(s,a) = r(s) + \gamma \sum_{s'} P(s'|s,a)u(s')$$
$$u(s) = \max_{a\in\mathcal{A}} q(s,a)$$

TD-learning = Q-learning with smoothing

$$q_{\text{local}}(s_t, a_t, r_t, s_{t+1}) = r_t + \gamma \max_{a\in\mathcal{A}} q_t(s_{t+1}, a)$$
$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \eta\big(q_{local}(s_t, a_t, r_t, s_{t+1}) - q(s_t, a_t)\big)$$

SARSA = on-policy Q-learning

$$q_{\text{local}}(s_t, a_t, r_t, s_{t+1}, a_{t+1}) = r_t + \gamma q_t(s_{t+1}, a_{t+1})$$
$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \eta\big(q_{local}(s_t, a_t, r_t, s_{t+1}, a_{t+1}) - q(s_t, a_t)\big)$$