Planning and Reinforcement Learning

Tom Silver

Robot Planning Meets Machine Learning

Princeton University

Fall 2025

Today's Agenda

1. Finish discussion of POMDPs
2. Talk about relationship between planning and RL
3. Discuss details for part 2 of course: papers and projects

Planning and Reinforcement Learning

Tom Silver

Robot Planning Meets Machine Learning

Princeton University

Fall 2025

What’s the Connection to RL?

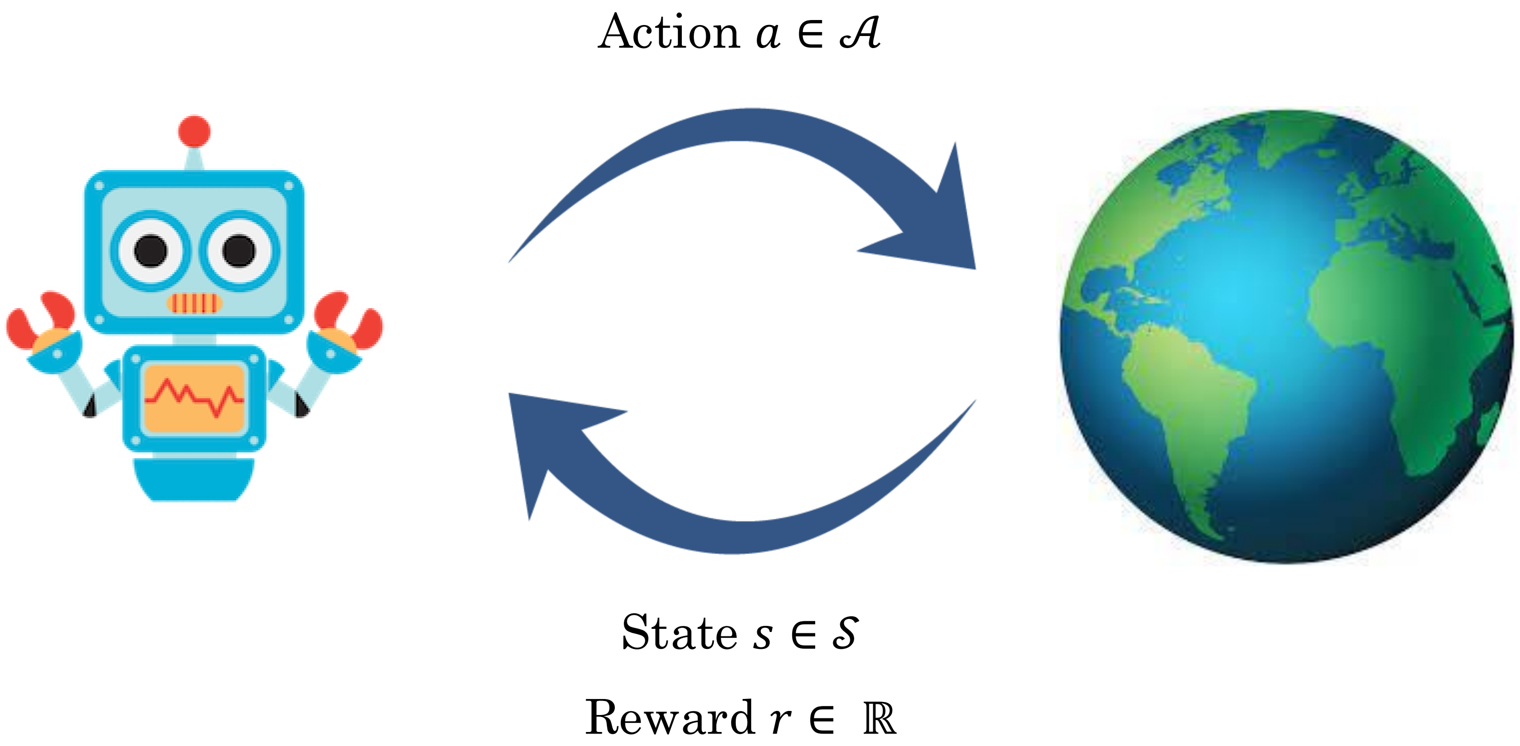
What’s the Connection to RL?

- We’ve been discussing planning in MDPs all this time
- Reinforcement learning (RL) also deals with computing value functions and policies in MDPs
- What’s the difference?

What's the Connection to RL?

- We've been discussing planning in MDPs all this time
- Reinforcement learning (RL) also deals with computing value functions and policies in MDPs
- What's the difference?
- In RL, we *don't know the MDP*. Specifically, $P(s' \mid s, a)$ and R.
- So what do we know?

The Basic RL Model*



*Assuming states are fully observed, which is not always assumed in RL

RL Model vs Simulator Access

- Recall *simulator access* to MDP: we can only sample from transition model, $s' \sim P(\cdot|\, s, a)$.

- RL has this assumption too.

RL Model vs Simulator Access

- Recall *simulator access* to MDP: we can only sample from transition model, $s' \sim P(\cdot|\, s, a)$.

- RL has this assumption too.

- The main additional assumption: we can't "choose our own state" with which to query the transition model.

- We're just at some current state, we take some action, then get one next state sample, and now that's our current state.

RL Model vs Simulator Access

- Recall *simulator access* to MDP: we can only sample from transition model, $s' \sim P(\cdot|\, s, a)$.
- RL has this assumption too.
- The main additional assumption: we can't "choose our own state" with which to query the transition model.
- We're just at some current state, we take some action, then get one next state sample, and now that's our current state.

However! In almost all RL work, it is assumed that you can "start over" (reset). Experience collected in *episodes*.

An API for “RL Access” to MDPs

https://github.com/openai/gym/blob/master/gym/core.py

```python
import gym

env = gym.make("Breakout-v0")

state = env.reset()

for timestep in range(100):
    action = env.action_space.sample()
    state, reward, done, info = env.step(action)
```

Planning by Reinforcement Learning

- There are many RL methods!
- Could we use them to do planning?
- Pretend that we only have RL model, even if we actually have (at least) simulator access to the MDP
- Yes, and people do this.

Planning by Reinforcement Learning

- There are many RL methods!
- Could we use them to do planning?
- Pretend that we only have RL model, even if we actually have (at least) simulator access to the MDP
- Yes, and people do this.

State-setting might be practically difficult.

Arguably, anyone who does RL in a simulator is doing this...

The benefit of state-setting might be small.

Planning by Reinforcement Learning

- There are many RL methods!

Is this synthetic or analytic learning?

- Could we use them to do planning?
- Pretend that we only have RL model, even if we actually have (at least) simulator access to the MDP
- Yes, and people do this.

State-setting might be practically difficult.

Arguably, anyone who does RL in a simulator is doing this...

The benefit of state-setting might be small.

Q-Learning: “Hello World” for RL

Q-LEARNING(`MDP`, α)

// Assume only “RL access” to `MDP`
initialize $\hat{Q}$ arbitrarily
repeat:
    // Start a new episode
    $s =$ `MDP.reset()`
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        $a =$ `SelectAction`$(s, \hat{Q}, \mathcal{A})$
        $s', r, d, i =$ `MDP.step(a)`
        // *Temporal difference* learning
        $\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha(r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a))$
        $s = s'$

Temporal Difference (TD) Learning

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

Temporal Difference (TD) Learning

$$\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha(\underbrace{r + \gamma \max_{a'} \hat{Q}(s',a')}_{\text{Looking ahead 1 step, like in Bellman backups}} - \hat{Q}(s,a))$$

Temporal Difference (TD) Learning

$$\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha(\underbrace{\underbrace{r + \gamma \max_{a'} \hat{Q}(s',a')}_{\text{Looking ahead 1 step, like in Bellman backups}} - \hat{Q}(s,a)}_{\textit{Temporal difference error}})$$

Temporal Difference (TD) Learning

$$\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha(\underbrace{\underbrace{r + \gamma \max_{a'} \hat{Q}(s',a')}_{\text{Looking ahead 1 step, like in Bellman backups}} - \hat{Q}(s,a)}_{\textit{Temporal difference error}})$$

If we did this for not just one sample of s' before updating $\hat{Q}$, but for w samples, then this update is equivalent to a Monte Carlo Bellman Backup with $\alpha = \frac{1}{w}$!

Temporal Difference (TD) Learning

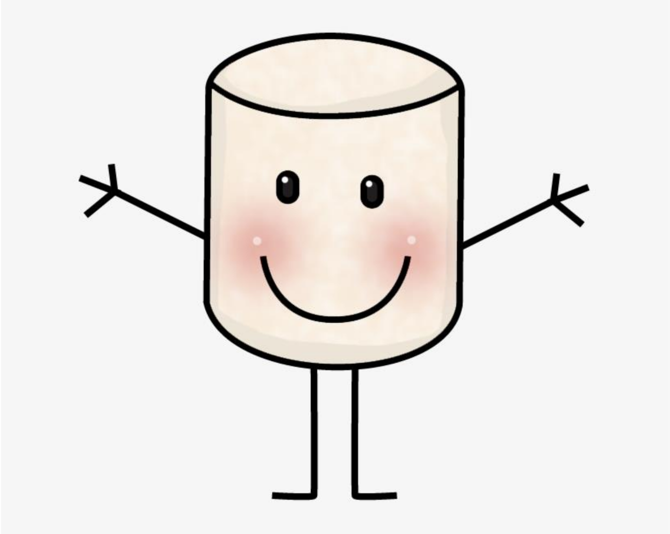
$$\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha\big(\underbrace{\underbrace{r + \gamma \max_{a'} \hat{Q}(s',a')}_{\text{Looking ahead 1 step, like in Bellman backups}} - \hat{Q}(s,a)}_{\textit{Temporal difference error}}\big)$$

If we did this for not just one sample of s' before updating $\hat{Q}$, but for w samples, then this update is equivalent to a Monte Carlo Bellman Backup with $\alpha = \frac{1}{w}$!

And we *are* getting multiple samples of s'. But complication: $\hat{Q}$ is changing in between. Non-stationary target. Nonetheless, Q-learning is guaranteed to converge to optimal.

Example: Marshmallows

- **States**: (hunger level, marshmallow remains)
  - Hunger level: 0, 1, 2 (higher is hungrier)
  - Marshmallow remains: True or False
- **Actions**: *eat* marshmallow, or *wait*
- **Horizon:** finite (horizon $H = 4$)
- **Rewards:** Negative hunger level squared (on next state)
- **Transition distribution**:
  - Marshmallow remains updated in obvious way
  - If *wait:*
    - With probability 0.25, hunger level increases by 1
    - Otherwise, hunger level stays the same
  - If *eat* (and marshmallow remains):
    - With probability 1, hunger level set to 0
  - If *eat* (and marshmallow gone):
    - Same as waiting

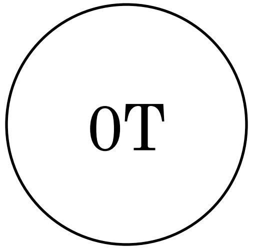


s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	0.0
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

Q-LEARNING(`MDP`, α)

// Assume only "RL access" to `MDP`
initialize $\hat{Q}$ arbitrarily
repeat:
  // Start a new episode
  $s =$ `MDP.reset()`
  repeat until episode done:
    // Use MAB ideas! (E.g., ϵ-greedy)
    $a =$ `SelectAction`$(s, \hat{Q}, \mathcal{A})$
    $s', r, d, i =$ `MDP.step(a)`
    // *Temporal difference* learning
    $\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha(r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a))$
    $s = s'$

s	a	$\widehat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	0.0
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

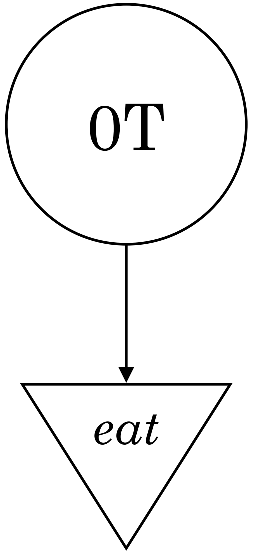


Q-Learning(MDP, α)

// Assume only "RL access" to MDP
initialize $\hat{Q}$ arbitrarily
repeat:
    // Start a new episode
    $\boxed{s = \texttt{MDP.reset()}}$
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        $a = \texttt{SelectAction}(s, \hat{Q}, \mathcal{A})$
        $s', r, d, i = \texttt{MDP.step(a)}$
        // *Temporal difference* learning
        $\hat{Q}(s, a) = \hat{Q}(s, a) + \alpha(r + \gamma \max_{a'} \hat{Q}(s', a') - \hat{Q}(s, a))$
        $s = s'$

s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	0.0
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

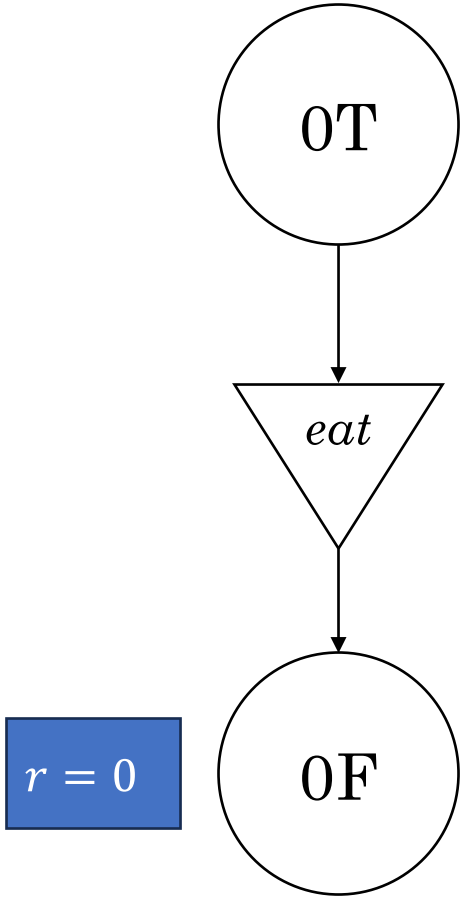
Random tiebreaking



Q-LEARNING(MDP, α)

```
// Assume only "RL access" to MDP
initialize Q̂ arbitrarily
repeat:
    // Start a new episode
    s = MDP.reset()
    repeat until episode done:
        // Use MAB ideas! (E.g., ε-greedy)
        a = SelectAction(s, Q̂, A)
        s', r, d, ı = MDP.step(a)
        // Temporal difference learning
        Q̂(s, a) = Q̂(s, a) + α(r + γ max_a' Q̂(s', a') − Q̂(s, a))
        s = s'
```

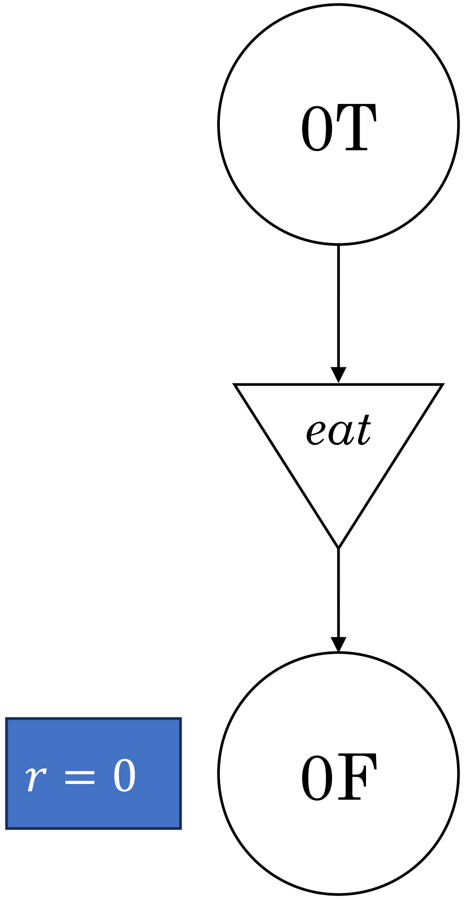
s	a	$\widehat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	0.0
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0



Q-LEARNING(`MDP`, α)

// Assume only "RL access" to `MDP`
initialize $\hat{Q}$ arbitrarily
repeat:
    // Start a new episode
    $s =$ `MDP.reset()`
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        $a =$ `SelectAction`$(s, \hat{Q}, \mathcal{A})$
        $\boxed{s', r, d, i = \texttt{MDP.step(a)}}$
        // *Temporal difference* learning
        $\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha(r + \gamma \max_{a'} \hat{Q}(s', a') - \hat{Q}(s,a))$
        $s = s'$

s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	0.0
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

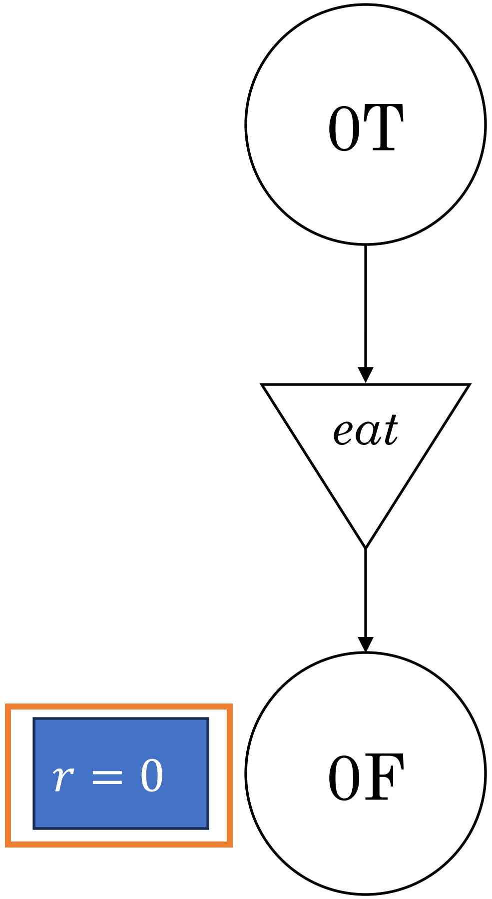


Q-LEARNING(`MDP`, α)

// Assume only "RL access" to `MDP`
initialize $\hat{Q}$ arbitrarily
repeat:
  // Start a new episode
  $s =$ `MDP.reset()`
  repeat until episode done:
    // Use MAB ideas! (E.g., ϵ-greedy)
    $a =$ `SelectAction`$(s, \hat{Q}, \mathcal{A})$
    $s', r, d, i =$ `MDP.step(a)`
    // *Temporal difference* learning
    $\hat{Q}(s,a) = \boxed{\hat{Q}(s,a)} + \alpha(r + \gamma \max_{a'} \hat{Q}(s',a') - \boxed{\hat{Q}(s,a)})$
    $s = s'$

0.0 0.0

s	a	$\widehat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	0.0
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

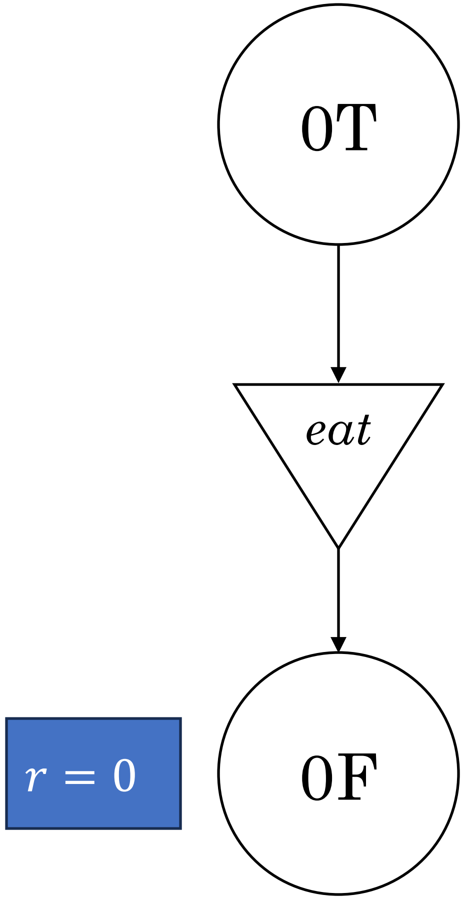


Q-LEARNING(`MDP`, α)

```
// Assume only "RL access" to MDP
initialize Q̂ arbitrarily
repeat:
    // Start a new episode
    s = MDP.reset()
    repeat until episode done:
        // Use MAB ideas! (E.g., ε-greedy)
        a = SelectAction(s, Q̂, A)
        s', r, d, i = MDP.step(a)
        // Temporal difference learning
        Q̂(s, a) = Q̂(s, a) + α(r + γ max_a' Q̂(s', a') − Q̂(s, a))
        s = s'
```

$$\hat{Q}(s,a) = \underbrace{\hat{Q}(s,a)}_{0.0} + \alpha(\boxed{r} + \gamma \max_{a'} \underbrace{\hat{Q}(s',a')}_{0.0} - \underbrace{\hat{Q}(s,a)}_{0.0})$$

s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	0.0
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0



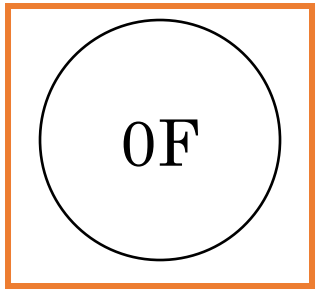
Q-LEARNING(MDP, α)

```
// Assume only "RL access" to MDP
initialize Q̂ arbitrarily
repeat:
    // Start a new episode
    s = MDP.reset()
    repeat until episode done:
        // Use MAB ideas! (E.g., ε-greedy)
        a = SelectAction(s, Q̂, A)
        s', r, d, i = MDP.step(a)
        // Temporal difference learning
        Q̂(s, a) = Q̂(s, a) + α(r + γ max_a' Q̂(s', a') − Q̂(s, a))
        s = s'
```

$$\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha(r + \gamma \boxed{\max_{a'} \hat{Q}(s',a')} - \hat{Q}(s,a))$$

0.0 0.0 0.0 0.0

s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	0.0
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

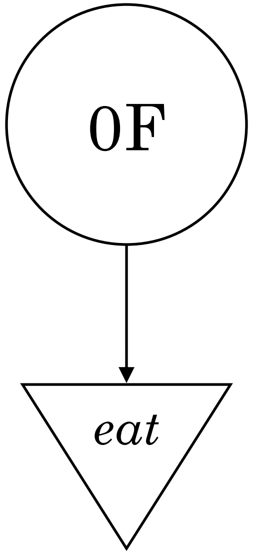


Q-LEARNING(`MDP`, α)

// Assume only "RL access" to `MDP`
initialize $\hat{Q}$ arbitrarily
repeat:
  // Start a new episode
  $s =$ `MDP.reset()`
  repeat until episode done:
    // Use MAB ideas! (E.g., ϵ-greedy)
    $a =$ `SelectAction`$(s, \hat{Q}, \mathcal{A})$
    $s', r, d, i =$ `MDP.step(a)`
    // *Temporal difference* learning
    $\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha(r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a))$
    $\boxed{s = s'}$

s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	0.0
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

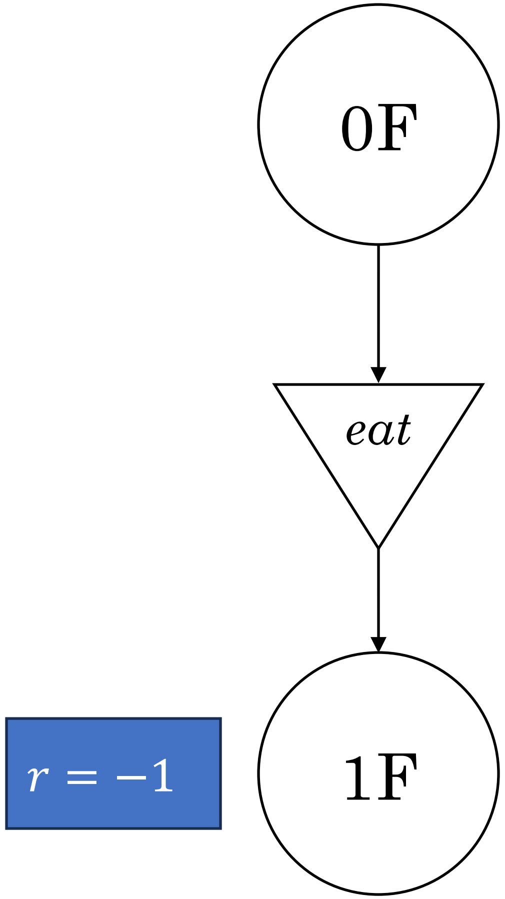
Random tiebreaking



Q-Learning(MDP, α)

// Assume only "RL access" to MDP
initialize $\hat{Q}$ arbitrarily
repeat:
    // Start a new episode
    $s =$ `MDP.reset()`
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        $\boxed{a = \texttt{SelectAction}(s, \hat{Q}, \mathcal{A})}$
        $s', r, d, i =$ `MDP.step(a)`
        // *Temporal difference* learning
        $\hat{Q}(s, a) = \hat{Q}(s, a) + \alpha(r + \gamma \max_{a'} \hat{Q}(s', a') - \hat{Q}(s, a))$
        $s = s'$

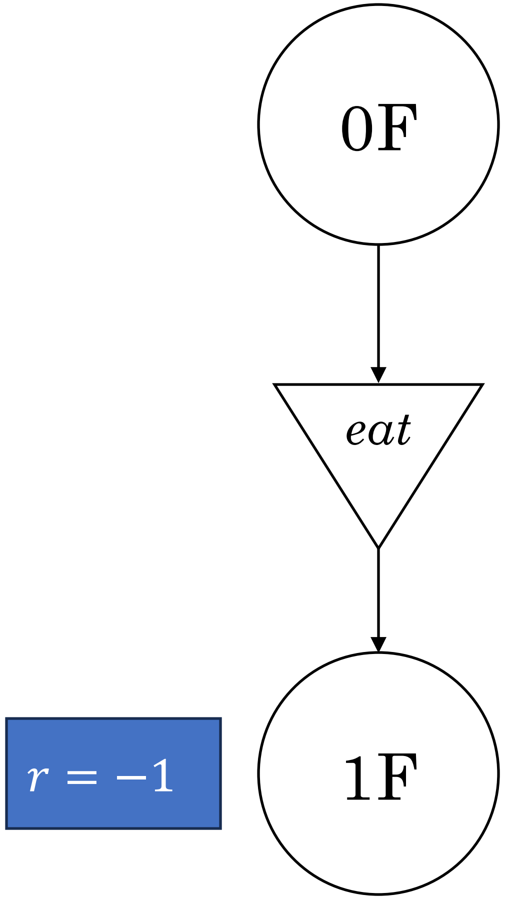
s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	0.0
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0



Q-LEARNING(MDP, α)

// Assume only "RL access" to MDP
initialize $\hat{Q}$ arbitrarily
repeat:
    // Start a new episode
    s = `MDP.reset()`
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        a = `SelectAction`($s, \hat{Q}, \mathcal{A}$)
        $\boxed{s', r, d, i = \texttt{MDP.step(a)}}$
        // *Temporal difference* learning
        $\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha(r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a))$
        $s = s'$

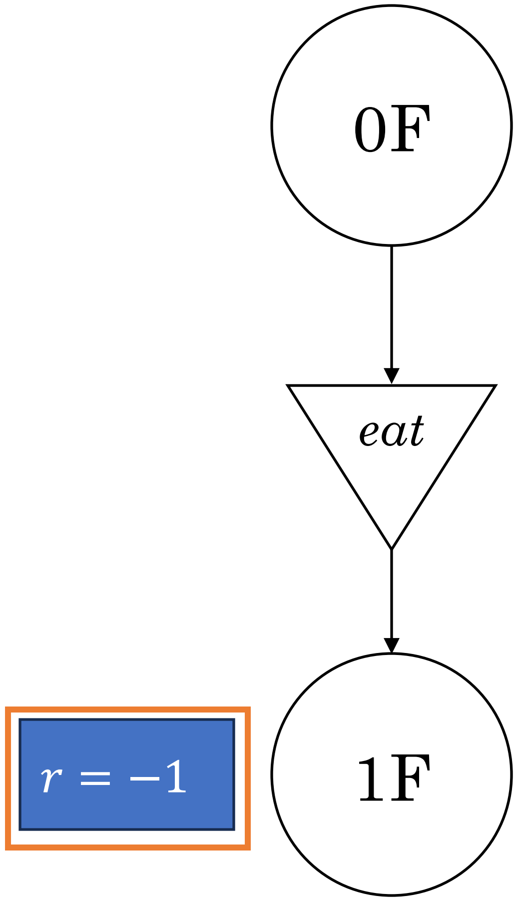
s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	0.0
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0



Q-LEARNING(MDP, α)

// Assume only "RL access" to MDP
initialize $\hat{Q}$ arbitrarily
repeat:
    // Start a new episode
    $s =$ `MDP.reset()`
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        $a =$ `SelectAction`$(s, \hat{Q}, \mathcal{A})$
        $s', r, d, i =$ `MDP.step(a)`
        // *Temporal difference* learning
        $\hat{Q}(s,a) = \underbrace{\boxed{\hat{Q}(s,a)}}_{0.0} + \alpha(r + \gamma \max_{a'} \hat{Q}(s',a') - \underbrace{\boxed{\hat{Q}(s,a)}}_{0.0})$
        $s = s'$

s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	0.0
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0



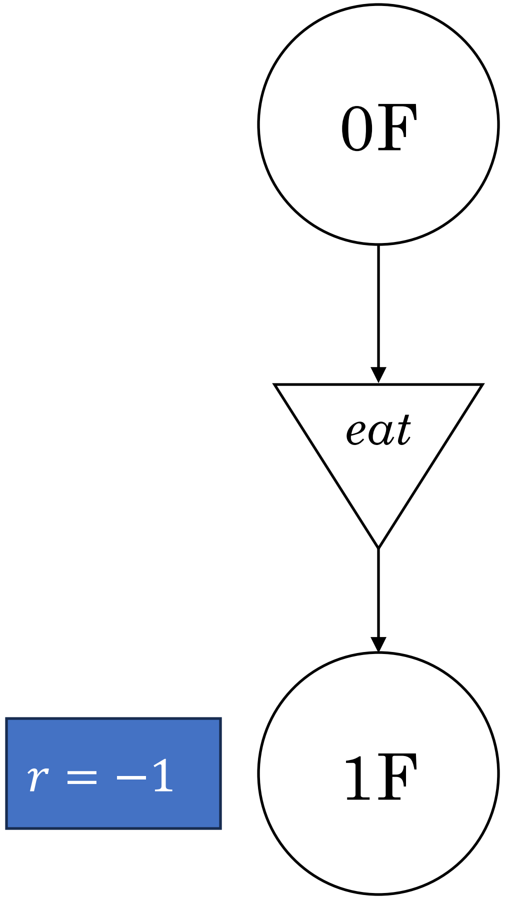
Q-LEARNING(MDP, α)

```
// Assume only "RL access" to MDP
initialize Q̂ arbitrarily
repeat:
    // Start a new episode
    s = MDP.reset()
    repeat until episode done:
        // Use MAB ideas! (E.g., ε-greedy)
        a = SelectAction(s, Q̂, 𝒜)
        s′, r, d, i = MDP.step(a)
        // Temporal difference learning
        Q̂(s, a) = Q̂(s, a) + α(r + γ max_a′ Q̂(s′, a′) − Q̂(s, a))
        s = s′
```

$$\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha(\boxed{r} + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a))$$

0.0 -1.0 0.0

s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	0.0
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0



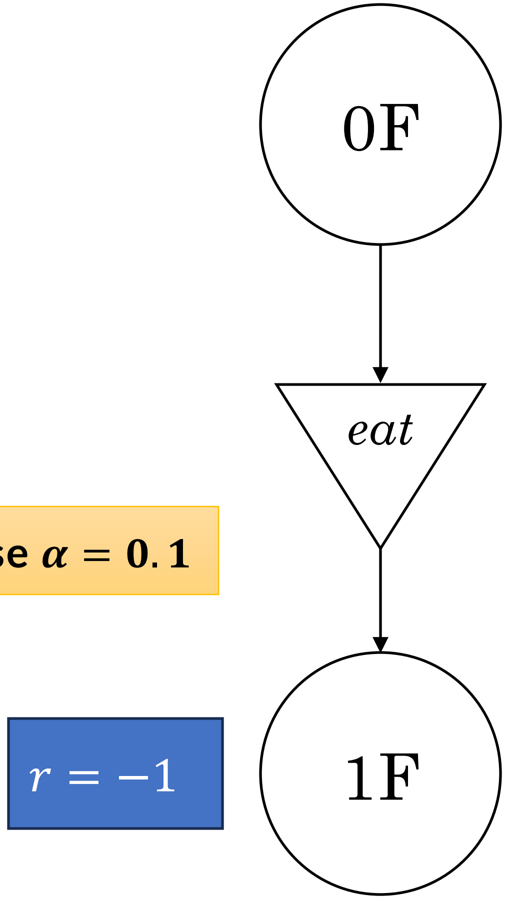
Q-LEARNING(`MDP`, α)

// Assume only "RL access" to `MDP`
initialize $\hat{Q}$ arbitrarily
repeat:
    // Start a new episode
    $s =$ `MDP.reset()`
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        $a =$ `SelectAction`$(s, \hat{Q}, \mathcal{A})$
        $s', r, d, i =$ `MDP.step(a)`
        // *Temporal difference* learning
        $\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha(r + \gamma \boxed{\max_{a'} \hat{Q}(s',a')} - \hat{Q}(s,a))$
        $s = s'$

0.0 −1.0 0.0 0.0

s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	**-0.1**
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

Suppose $\alpha = 0.1$



Q-LEARNING(MDP, α)

```
// Assume only "RL access" to MDP
initialize Q̂ arbitrarily
repeat:
    // Start a new episode
    s = MDP.reset()
    repeat until episode done:
        // Use MAB ideas! (E.g., ε-greedy)
        a = SelectAction(s, Q̂, A)
        s', r, d, i = MDP.step(a)
        // Temporal difference learning
        Q̂(s, a) = Q̂(s, a) + α(r + γ max_a' Q̂(s', a') − Q̂(s, a))
        s = s'
```

0.0 -1.0 0.0 0.0

s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	**-0.1**
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

0T

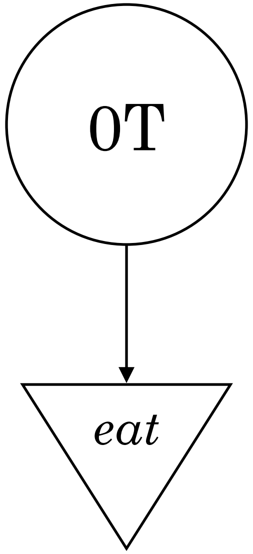
Episode reset

Q-Learning(MDP, α)

```
// Assume only "RL access" to MDP
initialize Q̂ arbitrarily
repeat:
    // Start a new episode
    s = MDP.reset()
    repeat until episode done:
        // Use MAB ideas! (E.g., ε-greedy)
        a = SelectAction(s, Q̂, 𝒜)
        s', r, d, i = MDP.step(a)
        // Temporal difference learning
        Q̂(s, a) = Q̂(s, a) + α(r + γ max_a' Q̂(s', a') − Q̂(s, a))
        s = s'
```

s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	**-0.1**
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

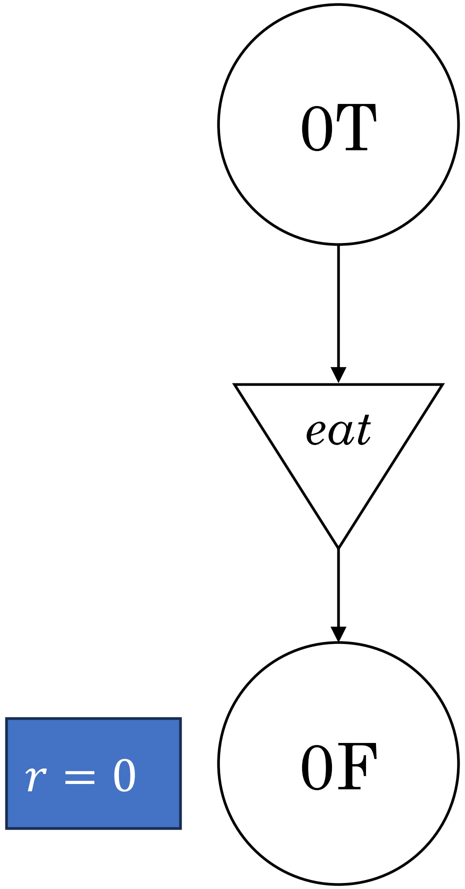
Random tiebreaking



Q-LEARNING(MDP, α)

```
// Assume only "RL access" to MDP
initialize Q̂ arbitrarily
repeat:
    // Start a new episode
    s = MDP.reset()
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        a = SelectAction(s, Q̂, 𝒜)
        s', r, d, ı = MDP.step(a)
        // Temporal difference learning
        Q̂(s, a) = Q̂(s, a) + α(r + γ max_a' Q̂(s', a') − Q̂(s, a))
        s = s'
```

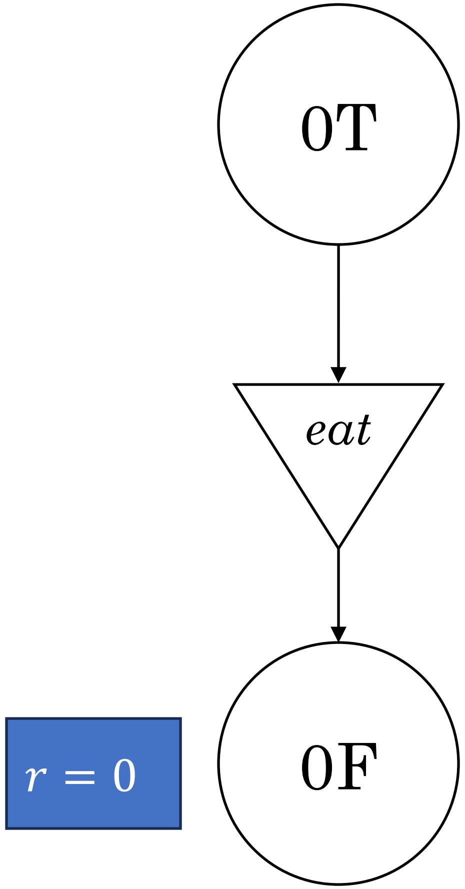
s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	**-0.1**
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0



Q-LEARNING(MDP, α)

```
// Assume only "RL access" to MDP
initialize Q̂ arbitrarily
repeat:
    // Start a new episode
    s = MDP.reset()
    repeat until episode done:
        // Use MAB ideas! (E.g., ε-greedy)
        a = SelectAction(s, Q̂, 𝒜)
        s′, r, d, i = MDP.step(a)
        // Temporal difference learning
        Q̂(s, a) = Q̂(s, a) + α(r + γ max_a′ Q̂(s′, a′) − Q̂(s, a))
        s = s′
```

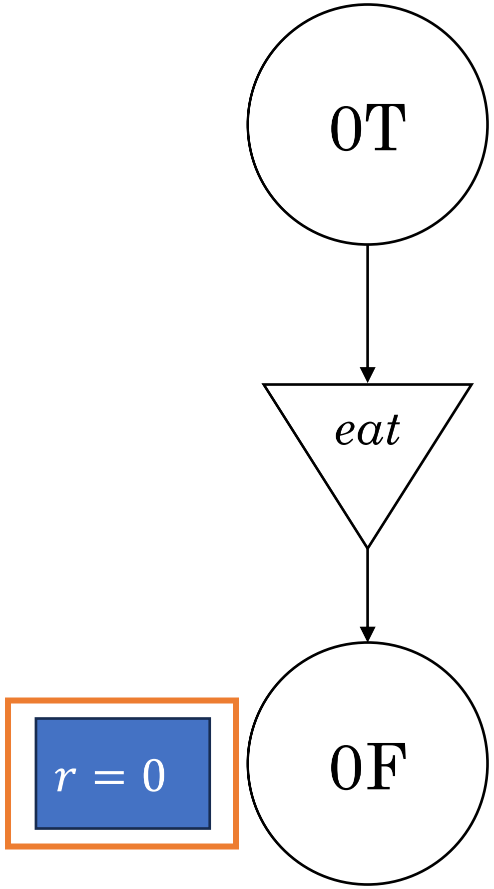
s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	**-0.1**
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0



Q-Learning(MDP, α)

// Assume only "RL access" to `MDP`
initialize $\hat{Q}$ arbitrarily
repeat:
    // Start a new episode
    $s =$ `MDP.reset()`
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        $a =$ `SelectAction`$(s, \hat{Q}, \mathcal{A})$
        $s', r, d, i =$ `MDP.step(a)`
        // *Temporal difference* learning
        $\hat{Q}(s,a) = \underbrace{\boxed{\hat{Q}(s,a)}}_{0.0} + \alpha(r + \gamma \max_{a'} \hat{Q}(s',a') - \underbrace{\boxed{\hat{Q}(s,a)}}_{0.0})$
        $s = s'$

s	a	$\widehat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	**-0.1**
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

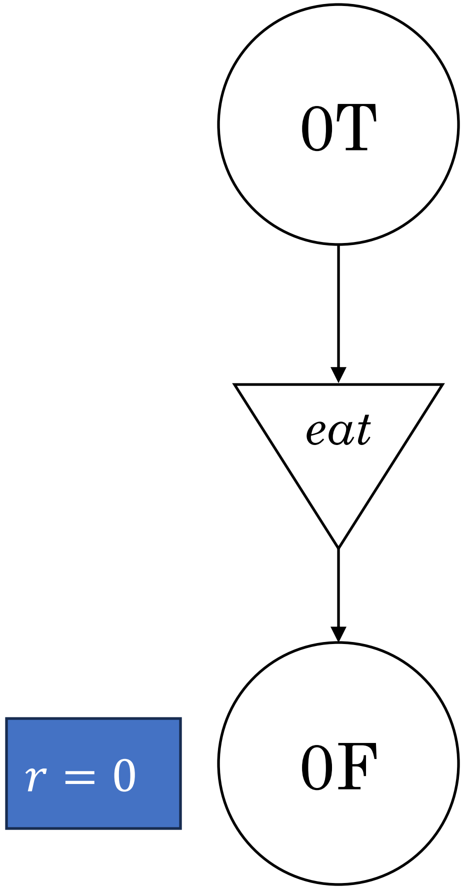


Q-LEARNING(`MDP`, α)

// Assume only "RL access" to `MDP`
initialize $\hat{Q}$ arbitrarily
repeat:
  // Start a new episode
  $s =$ `MDP.reset()`
  repeat until episode done:
    // Use MAB ideas! (E.g., ϵ-greedy)
    $a =$ `SelectAction`$(s, \hat{Q}, \mathcal{A})$
    $s', r, d, i =$ `MDP.step(a)`
    // *Temporal difference* learning
    $\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha(\boxed{r} + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a))$
    $s = s'$

0.0 0.0 0.0

s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	**-0.1**
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0



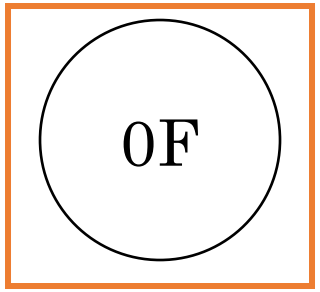
Q-LEARNING(MDP, α)

```
// Assume only "RL access" to MDP
initialize Q̂ arbitrarily
repeat:
    // Start a new episode
    s = MDP.reset()
    repeat until episode done:
        // Use MAB ideas! (E.g., ε-greedy)
        a = SelectAction(s, Q̂, A)
        s', r, d, i = MDP.step(a)
        // Temporal difference learning
        Q̂(s, a) = Q̂(s, a) + α(r + γ max_a' Q̂(s', a') − Q̂(s, a))
        s = s'
```

$$\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha(r + \gamma \boxed{\max_{a'} \hat{Q}(s',a')} - \hat{Q}(s,a))$$

0.0 0.0 0.0 0.0

s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	**-0.1**
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

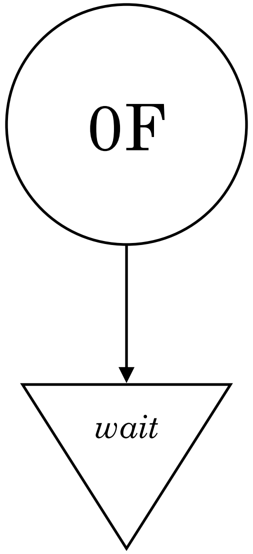


Q-Learning(MDP, α)

```
// Assume only "RL access" to MDP
initialize Q̂ arbitrarily
repeat:
    // Start a new episode
    s = MDP.reset()
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        a = SelectAction(s, Q̂, 𝒜)
        s′, r, d, i = MDP.step(a)
        // Temporal difference learning
        Q̂(s, a) = Q̂(s, a) + α(r + γ max_a′ Q̂(s′, a′) − Q̂(s, a))
        s = s′
```

s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	**-0.1**
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

Suppose we exploit

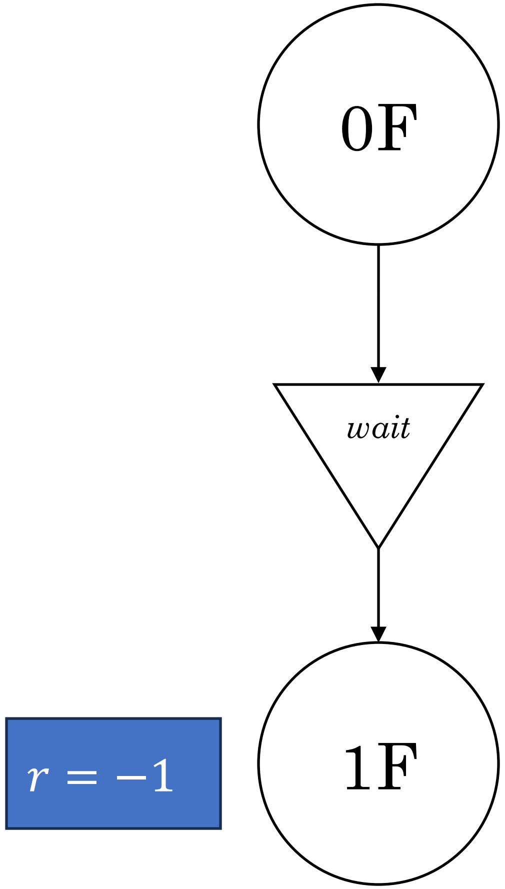


Q-LEARNING(MDP, α)

```
// Assume only "RL access" to MDP
initialize Q̂ arbitrarily
repeat:
    // Start a new episode
    s = MDP.reset()
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        a = SelectAction(s, Q̂, 𝒜)
        s′, r, d, i = MDP.step(a)
        // Temporal difference learning
        Q̂(s, a) = Q̂(s, a) + α(r + γ max_a′ Q̂(s′, a′) − Q̂(s, a))
        s = s′
```

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

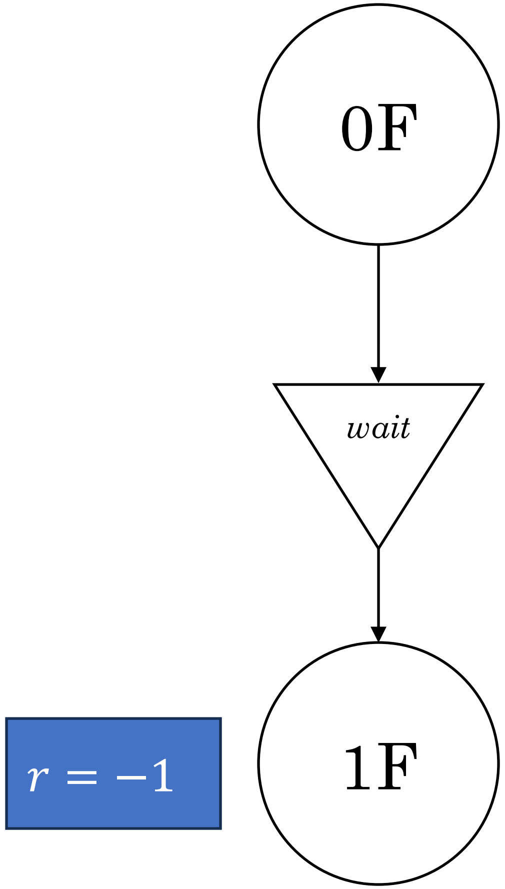
s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	**-0.1**
	wait	0.0
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0



Q-LEARNING(MDP, α)

```
// Assume only "RL access" to MDP
initialize Q̂ arbitrarily
repeat:
    // Start a new episode
    s = MDP.reset()
    repeat until episode done:
        // Use MAB ideas! (E.g., ε-greedy)
        a = SelectAction(s, Q̂, A)
        s', r, d, i = MDP.step(a)
        // Temporal difference learning
        Q̂(s, a) = Q̂(s, a) + α(r + γ max_a' Q̂(s', a') − Q̂(s, a))
        s = s'
```

s	a	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	**-0.1**
	wait	**-0.1**
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0



Q-Learning(MDP, α)

```
// Assume only "RL access" to MDP
initialize Q̂ arbitrarily
repeat:
    // Start a new episode
    s = MDP.reset()
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        a = SelectAction(s, Q̂, 𝒜)
        s', r, d, i = MDP.step(a)
        // Temporal difference learning
        Q̂(s, a) = Q̂(s, a) + α(r + γ max_a' Q̂(s', a') − Q̂(s, a))
        s = s'
```

s	a	$\widehat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	**-0.1**
	wait	**-0.1**
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

0T

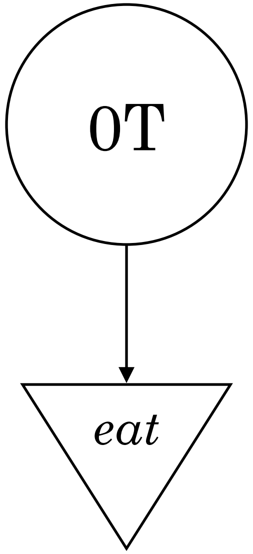
Episode reset

Q-LEARNING(`MDP`, α)

```
// Assume only "RL access" to MDP
initialize Q̂ arbitrarily
repeat:
    // Start a new episode
    s = MDP.reset()
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        a = SelectAction(s, Q̂, 𝒜)
        s', r, d, i = MDP.step(a)
        // Temporal difference learning
        Q̂(s, a) = Q̂(s, a) + α(r + γ max_a' Q̂(s', a') − Q̂(s, a))
        s = s'
```

s	*a*	$\hat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	**-0.1**
	wait	**-0.1**
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

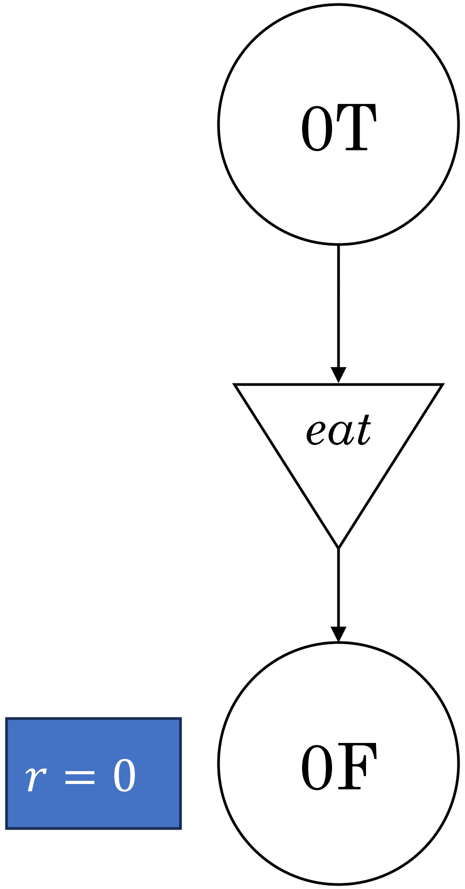
Random tiebreaking



Q-LEARNING(MDP, α)

// Assume only "RL access" to `MDP`
initialize $\hat{Q}$ arbitrarily
repeat:
    // Start a new episode
    $s =$ `MDP.reset()`
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        $\boxed{a = \texttt{SelectAction}(s, \hat{Q}, \mathcal{A})}$
        $s', r, d, i =$ `MDP.step(a)`
        // *Temporal difference* learning
        $\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha(r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a))$
        $s = s'$

s	a	$\widehat{Q}$
0T	*eat*	0.0
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	**-0.1**
	wait	**-0.1**
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

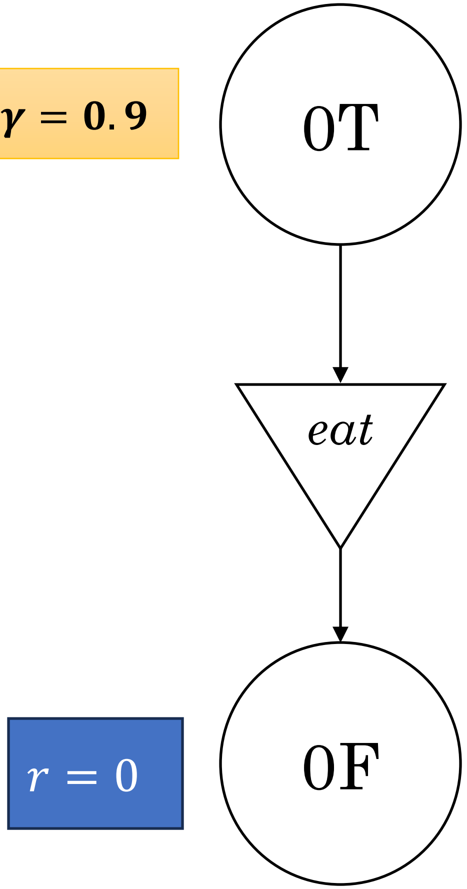


Q-LEARNING(MDP, α)

```
// Assume only "RL access" to MDP
initialize Q̂ arbitrarily
repeat:
    // Start a new episode
    s = MDP.reset()
    repeat until episode done:
        // Use MAB ideas! (E.g., ε-greedy)
        a = SelectAction(s, Q̂, 𝒜)
        s', r, d, i = MDP.step(a)
        // Temporal difference learning
        Q̂(s, a) = Q̂(s, a) + α(r + γ max_a' Q̂(s', a') − Q̂(s, a))
        s = s'
```

s	a	$\hat{Q}$
0T	*eat*	**-0.09**
	wait	0.0
1T	*eat*	0.0
	wait	0.0
2T	*eat*	0.0
	wait	0.0
0F	*eat*	**-0.1**
	wait	**-0.1**
1F	*eat*	0.0
	wait	0.0
2F	*eat*	0.0
	wait	0.0

Suppose $\gamma = 0.9$



Q-LEARNING(MDP, α)

```
// Assume only "RL access" to MDP
initialize Q̂ arbitrarily
repeat:
    // Start a new episode
    s = MDP.reset()
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        a = SelectAction(s, Q̂, 𝒜)
        s', r, d, i = MDP.step(a)
        // Temporal difference learning
        Q̂(s, a) = Q̂(s, a) + α(r + γ max_a' Q̂(s', a') − Q̂(s, a))
        s = s'
```

$$\boxed{\hat{Q}(s,a) = \hat{Q}(s,a) + \alpha(r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a))}$$

Bellman Backups: The ❤️ of MDP Planning and RL

Many MDP planning and RL methods are some variation of:

Repeat:

1. Choose some state s and action a

2. Bellman backup to improve $Q(s, a)$

Bellman Backups: The ❤️ of MDP Planning and RL

Value Iteration

Repeat:

1. Choose some state s and action a

   Loop over *all*

2. Bellman backup to improve $Q(s, a)$

   Compute *exactly*

Bellman Backups: The ❤️ of MDP Planning and RL

Expectimax Search

Repeat:

1. Choose some state s and action a

   Consider *reachable states backwards* in time

2. Bellman backup to improve $Q(s, a)$

   Compute *exactly*

Bellman Backups: The ❤️ of MDP Planning and RL

Real-Time Dynamic Programming

Repeat:

1. Choose some state s and action a

   Sample *trajectories* using current greedy policy

2. Bellman backup to improve $Q(s, a)$

   Compute *exactly*

Bellman Backups: The ❤️ of MDP Planning and RL

Sparse Sampling

Repeat:

1. Choose some state s and action a

   Consider *reachable* states *backwards* in time

2. Bellman backup to improve $Q(s, a)$

   Compute *approximately* by sampling

Bellman Backups: The ❤️ of MDP Planning and RL

Monte-Carlo Tree Search

Repeat:

1. Choose some state s and action a

Sample *trajectories* using explore-exploit policy

2. Bellman backup to improve $Q(s, a)$

Compute *approximately* by sampling

Bellman Backups: The ❤️ of MDP Planning and RL

Q-Learning

Repeat:

1. Choose some state s and action a

   The single transition just witnessed

2. Bellman backup to improve $Q(s, a)$

   Compute *approximately* using single transition

Bellman Backups: The ❤️ of MDP Planning and RL

Many MDP planning and RL methods are some variation of:

Repeat:

1. Choose some state s and action a →
2. Bellman backup to improve $Q(s, a)$

Central question: how to *schedule* state-action updates?

Bellman Backups: The ❤️ of MDP Planning and RL

Many MDP planning and RL methods are some variation of:

Repeat:

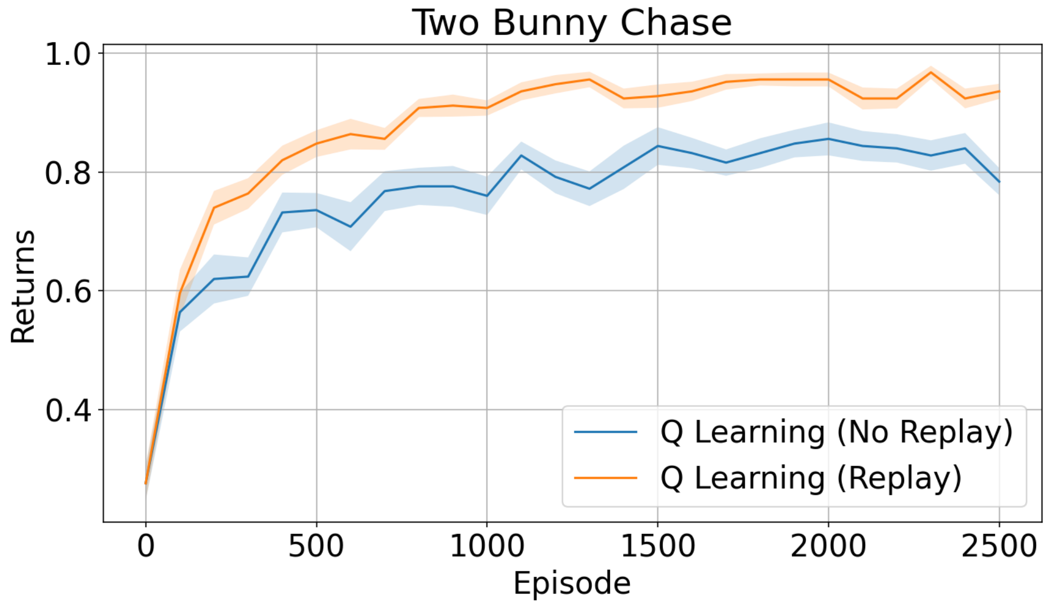
1. Choose some state s and action a →
2. Bellman backup to improve $Q(s, a)$

Central question: how to *schedule* state-action updates?

In RL, would it ever make sense to choose a never-before-seen state / action?

Experience Replay

- Updating based only on most recent state and action is limiting
- Instead, maintain history of transitions ("buffer")
- Randomly sample from the buffer and update Q accordingly



Prioritized Experience Replay

- Experience replay is better, but sampling *uniformly* from the history is still limiting
- Many transitions sampled will result in no change
- Instead, sample based on *expected learning progress*
- Approximate learning progress with TD error

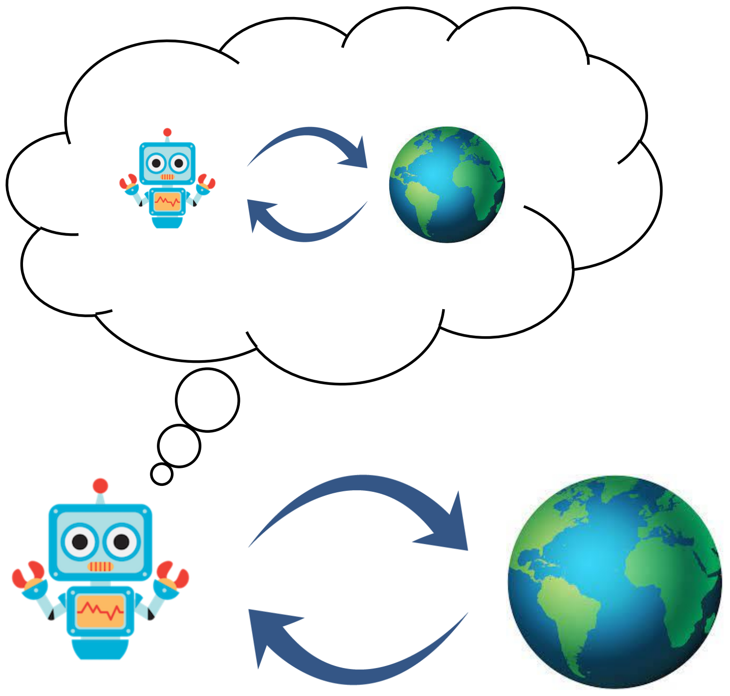
Model-Free vs. Model-Based RL

Q-learning is *model-free*: it requires no explicit transition model

Other RL methods are *model-based*:

1. Use online experience to learn transition model

2. Plan in the learned transition model (offline or online)



Model-based RL

MBRL Example: Dyna-Q

DYNA-Q(`MDP`, α)

// Assume only "RL access" to `MDP`
initialize $\hat{Q}$ arbitrarily
initialize $\hat{P}$ and $\hat{R}$ arbitrarily
repeat:
    // Start a new episode
    $s =$ `MDP.reset()`
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        $a = \texttt{SelectAction}(s, \hat{Q}, \mathcal{A})$
        $s', r, d, i =$ `MDP.step(a)`
        // *Temporal difference* learning
        $\hat{Q}(s, a) = \hat{Q}(s, a) + \alpha(r + \gamma \max_{a'} \hat{Q}(s', a') - \hat{Q}(s, a))$
        // *Model learning*
        $\hat{R}(s, a, s') = r$
        $\hat{P}(x \mid s, a) =$ *[Many choices...]*
        // *Update $\hat{Q}$ by planning [Many choices...]*
        $s = s'$

MBRL Example: Dyna-Q

DYNA-Q(MDP, α)

// Assume only "RL access" to MDP
initialize $\hat{Q}$ arbitrarily
initialize $\hat{P}$ and $\hat{R}$ arbitrarily
repeat:
    // Start a new episode
    $s =$ `MDP.reset()`
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        $a =$ `SelectAction`$(s, \hat{Q}, \mathcal{A})$
        $s', r, d, i =$ `MDP.step(a)`
        // *Temporal difference* learning
        $\hat{Q}(s, a) = \hat{Q}(s, a) + \alpha(r + \gamma \max_{a'} \hat{Q}(s', a') - \hat{Q}(s, a))$
        // *Model learning*
        $\hat{R}(s, a, s') = r$
        $\hat{P}(x \mid s, a) =$ *[Many choices...]*
        // *Update $\hat{Q}$ by planning [Many choices...]*
        $s = s'$

We are doing *both* model-free RL (Q learning) *and* model-based RL (planning)

MBRL Example: Dyna-Q

DYNA-Q(`MDP`, α)

// Assume only "RL access" to `MDP`
initialize $\hat{Q}$ arbitrarily
initialize $\hat{P}$ and $\hat{R}$ arbitrarily
repeat:
    // Start a new episode
    s = `MDP.reset()`
    repeat until episode done:
        // Use MAB ideas! (E.g., ϵ-greedy)
        a = `SelectAction`$(s, \hat{Q}, \mathcal{A})$
        s', r, d, i = `MDP.step(a)`
        // *Temporal difference* learning
        $\hat{Q}(s, a) = \hat{Q}(s, a) + \alpha(r + \gamma \max_{a'} \hat{Q}(s', a') - \hat{Q}(s, a))$
        // *Model learning*
        $\hat{R}(s, a, s') = r$
        $\hat{P}(x \mid s, a) =$ *[Many choices...]*
        // *Update $\hat{Q}$ by planning [Many choices...]*
        $s = s'$

Brainstorm and discuss:
1. What are some reasonable choices for model learning and planning?
2. What are possible issues with those choices?

Model-Free vs. Model-Based RL

> “Our view is that the contrast between the alternatives in all these debates has been exaggerated, that more insight can be gained by recognizing the similarities between these two sides than by opposing them.” Sutton & Barto (2018)

(The same can be said for planning vs. RL in general)

Today's Agenda

1. Review what we've covered so far
2. Talk about relationship between planning and RL
3. **Discuss details for part 2 of course: papers and projects**

Paper Presentations

- We will have 3 presenters and 3 papers per class
- The presenters should present as a team! Here's an ideal agenda:

Presenter 1	Background (spanning all 3 papers, connecting to what we've seen previously in class)
	Overview of a shorter/simpler paper
Presenter 2	Deep dive into "main" paper
	Optionally: some code, demo, interactive element, or other things above-and-beyond
Presenter 3	Overview of the remaining paper
	Reflecting on all the papers and lessons learned

Paper Presentations

- The prepared presentation should last about **50 minutes** in total
- With remaining **25-30 minutes**, we will have brainstorming breakouts, where the goal is to generate research project ideas (for some future, hypothetical time)
- First, we will brainstorm individually for **5 minutes**
- Then, we will share (**5 minutes**) and try to cluster into **3-6 themes**
- Then, we will break off into theme groups and continue brainstorming for **10 minutes**
- With remaining time, we will report back to the class

Paper Presentations: Expectations and Grading

Grading will be based on:

- Clarity of presentation
- Overall effort
- Connections between papers and topics
- Audience engagement
- Feedback from your fellow presenters (collected privately)

Total grade = 50% shared grade for group + 50% individual grade

Paper Matching

Complete the form sent through Ed

Deadline to complete: Friday, September 19

Pre-Class Paper Reviews

- Everyone must read all 3 papers before each class
- Choose 1 of the 3 papers to write a review
- For reviews: imagine you are actually reviewing this paper for possible acceptance to a conference or journal!
- Let's look at the review outline…

Final Projects

- **Teams** of 1-4 people (graded proportionately)
- **Scope**: ambitious! Aim for “something that we would be proud to submit to a workshop, or even a conference / journal”
  - Failure is okay! Final projects are great for taking big research risks
- **Requirements**:
  - Must involve planning and learning
  - Must involve a significant programming effort
  - Must be fun and cool

Final Projects: Possible Starting Points

- Reimplement a planning + learning method from a paper and apply it to a new domain
- Reimplement a planning + learning method and then try to "beat" it on a benchmark domain
- Start with a really interesting domain and try a variety of planning + learning methods to see what works best
- Something completely different… maybe something whacky…

Final Project: Timeline & Deliverables

- **October 6**: Proposal Due (details to come)
  - Not too early to start thinking about teams and general topics!
- **October 31**: Project Update 1 Due
- **November 24:** Project Update 2 Due
- **December 15:** Final Project Due