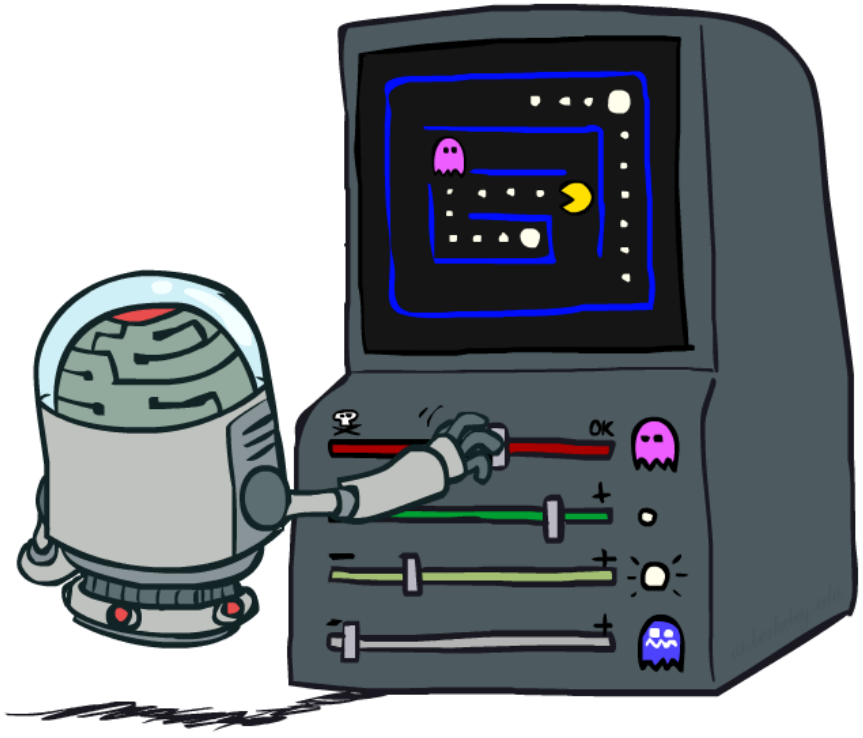


Bonus Lecture: Introduction to Reinforcement Learning

Garima Lalwani, Karan Ganju and Unnat Jain

Credits: These slides and images are borrowed from slides by David Silver and Peter Abbeel

Outline

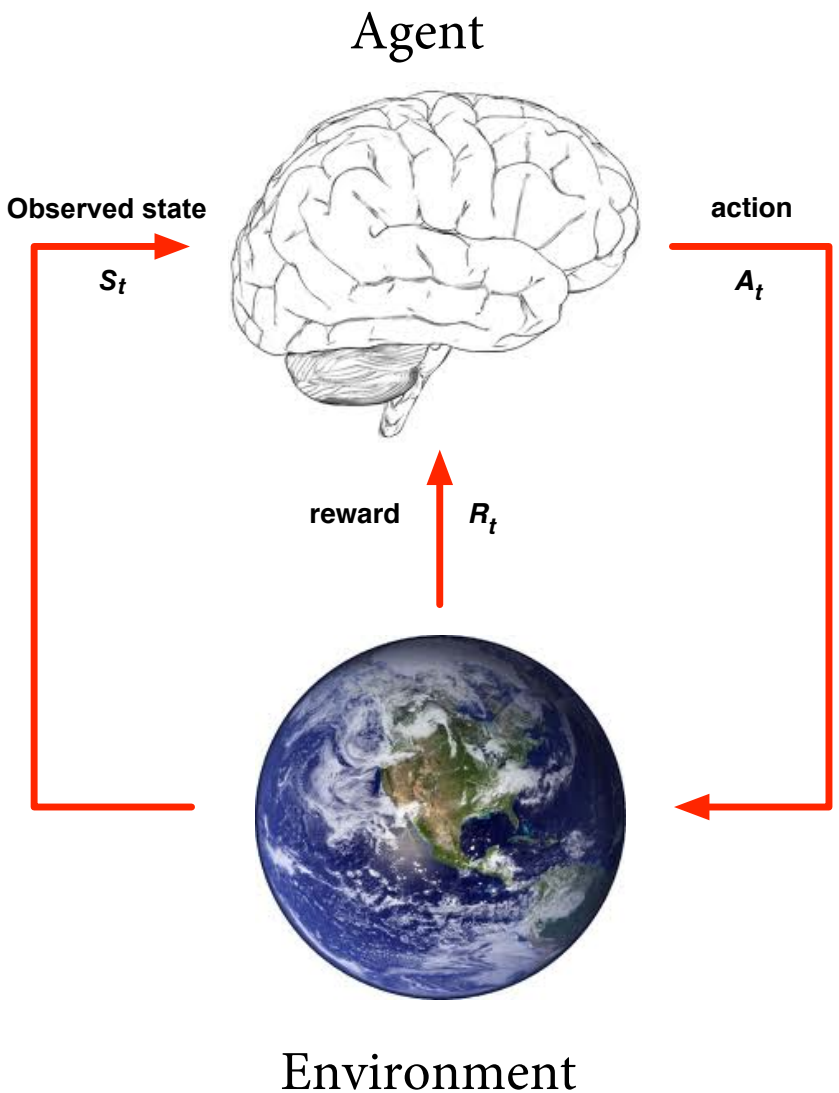
Part 1: RL Problem Formulation

Characteristics of Reinforcement Learning

What makes reinforcement learning different from other machine learning paradigms?

- There is no supervisor, only a *reward* signal
- Feedback is delayed, not instantaneous
- Time really matters (correlated, non i.i.d data)
- Agent's actions affect the subsequent data it receives

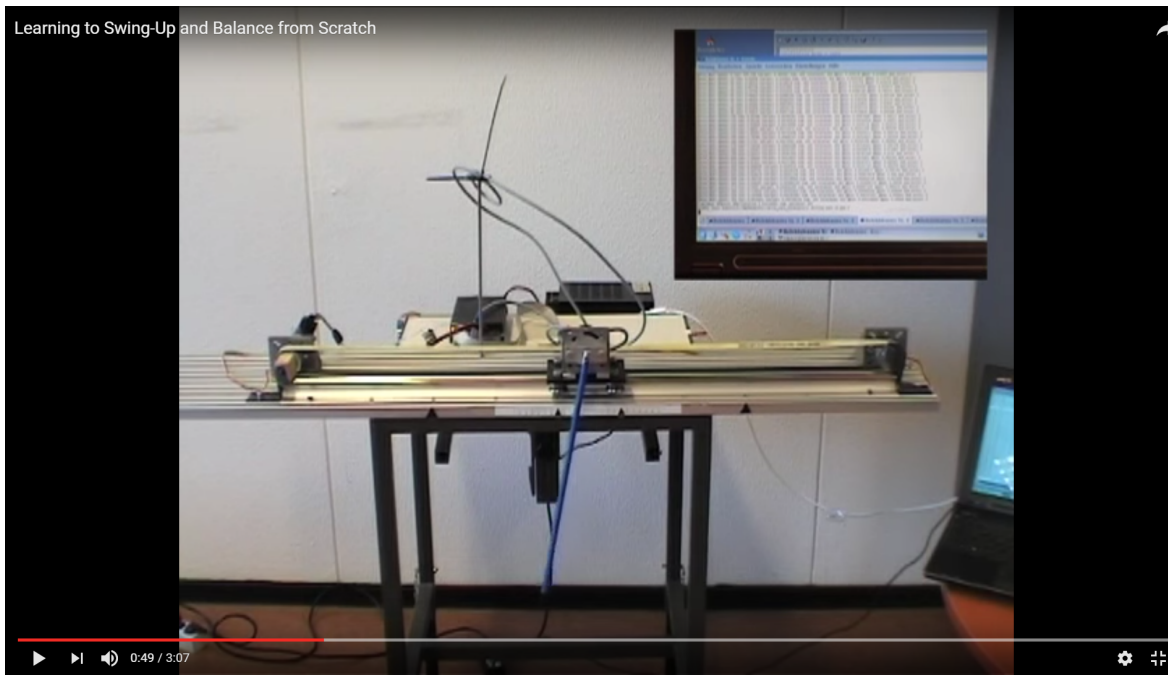
Agent and Environment



Rewards

- A reward R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximise cumulative reward

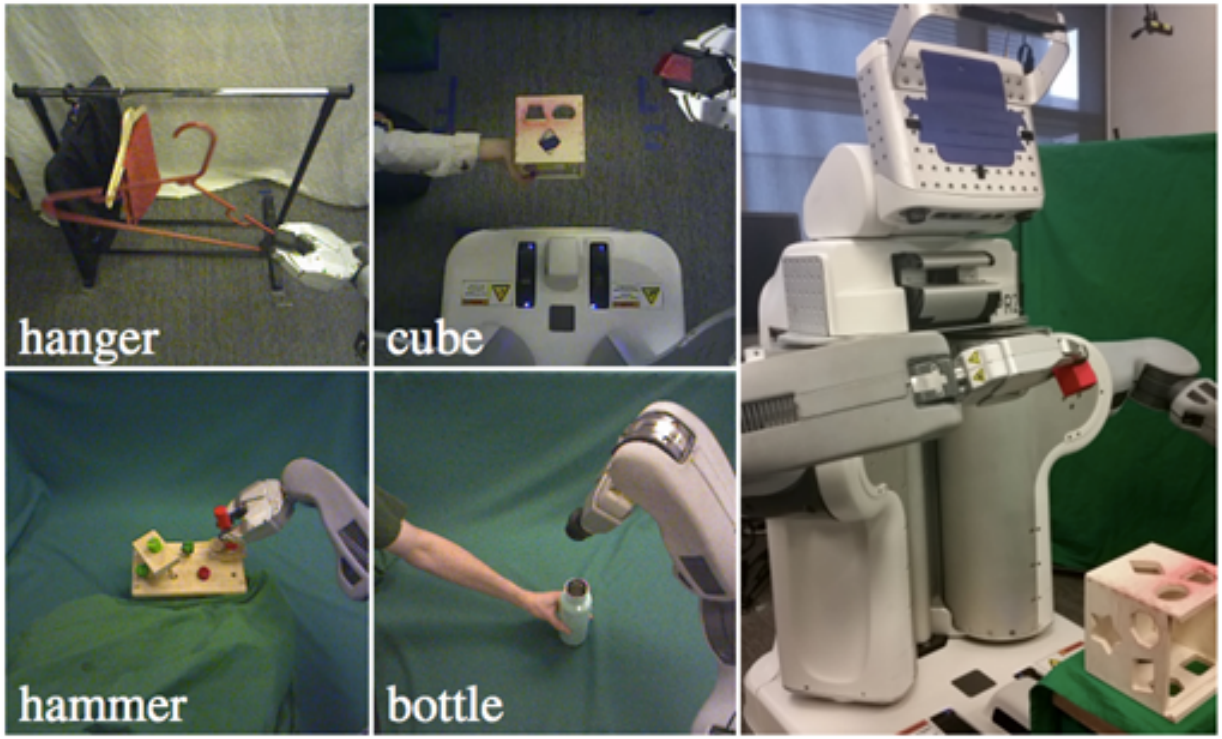
Rod Balancing Demo



Learn to swing up and balance a real pole based on raw visual input data, ICNIP 2012

https://www.youtube.com/watch?v=Lt-KLtkDlh8

RL based visual control



End-to-end training of deep visuomotor policies, JMLR 2016

https://www.youtube.com/watch?v=CE6fBDHPbP8

RL based visual control

Link: https://goo.gl/kY4RmS

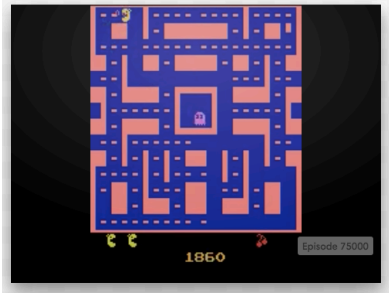
Source: https://68.media.tumblr.com/

Examples of Rewards

- Fly stunt manoeuvres in a helicopter
  - +ve reward for following desired trajectory
  - −ve reward for crashing

Stanford autonomous helicopter
Abbeel et. Al.

- Play many Atari games better than humans
  - +/−ve reward for increasing/decreasing score

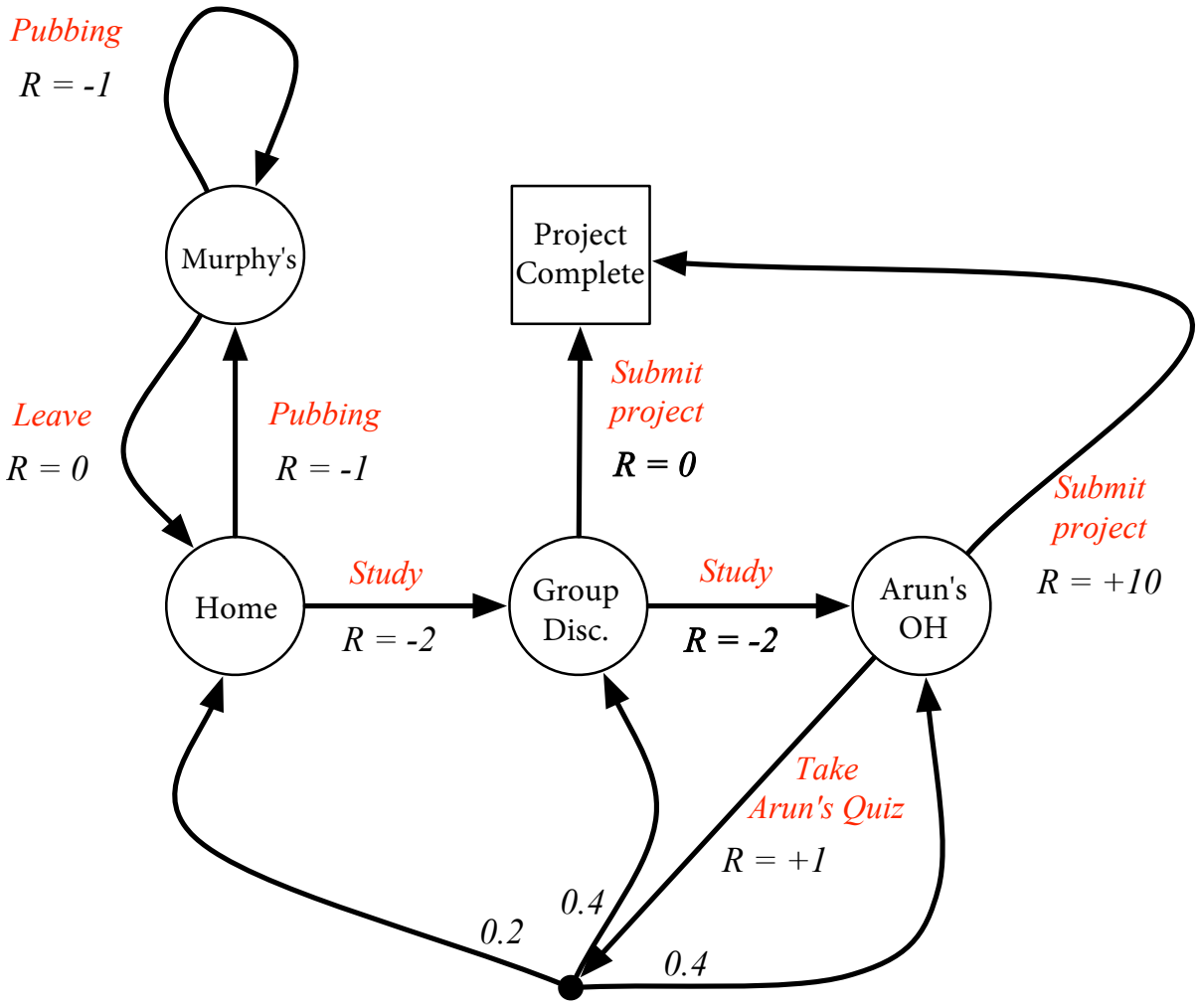


https://gym.openai.com/

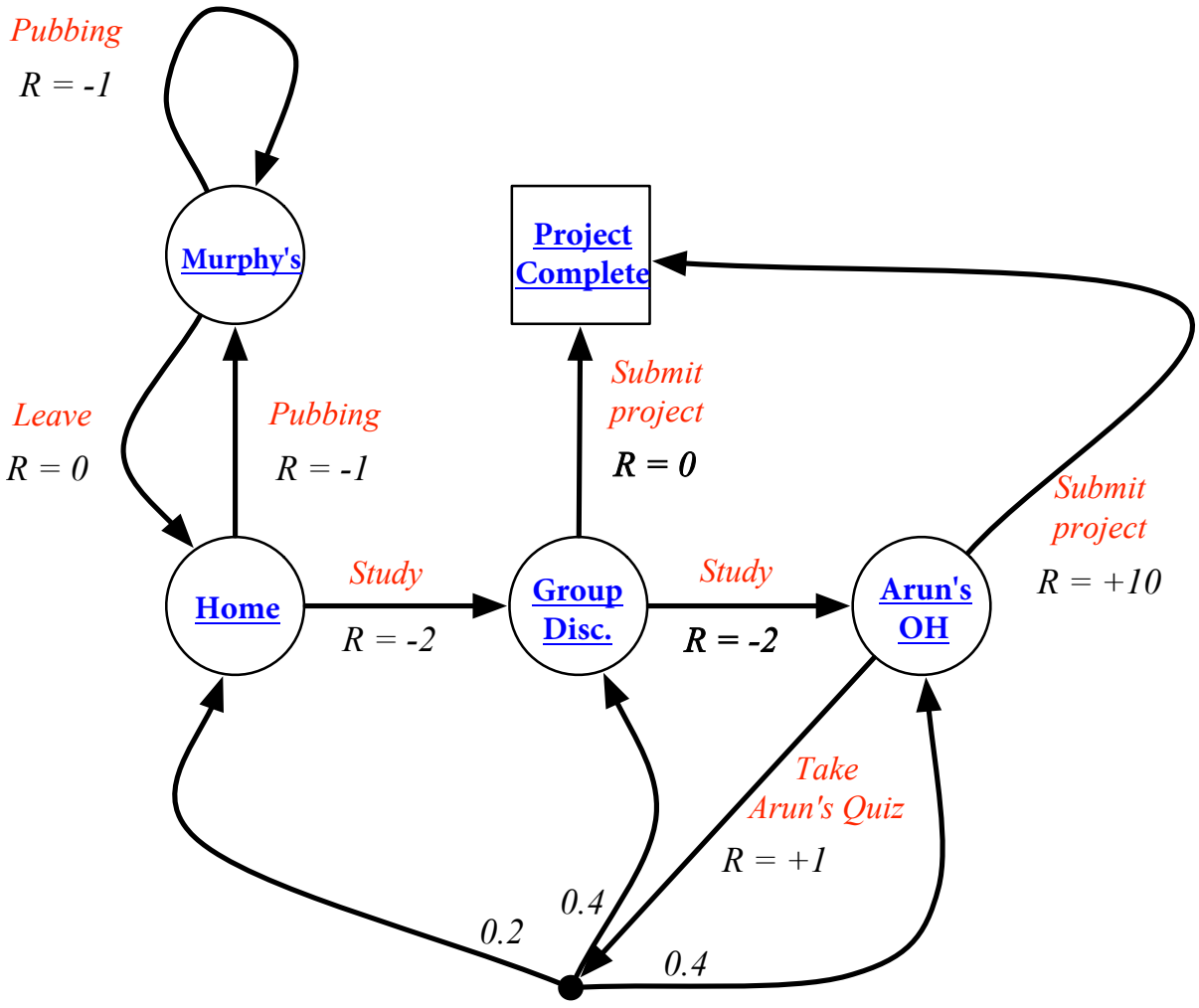
- Defeat the world champion at Go
  - +/−ve reward for winning/losing a game

https://deepmind.com/research/alphago/

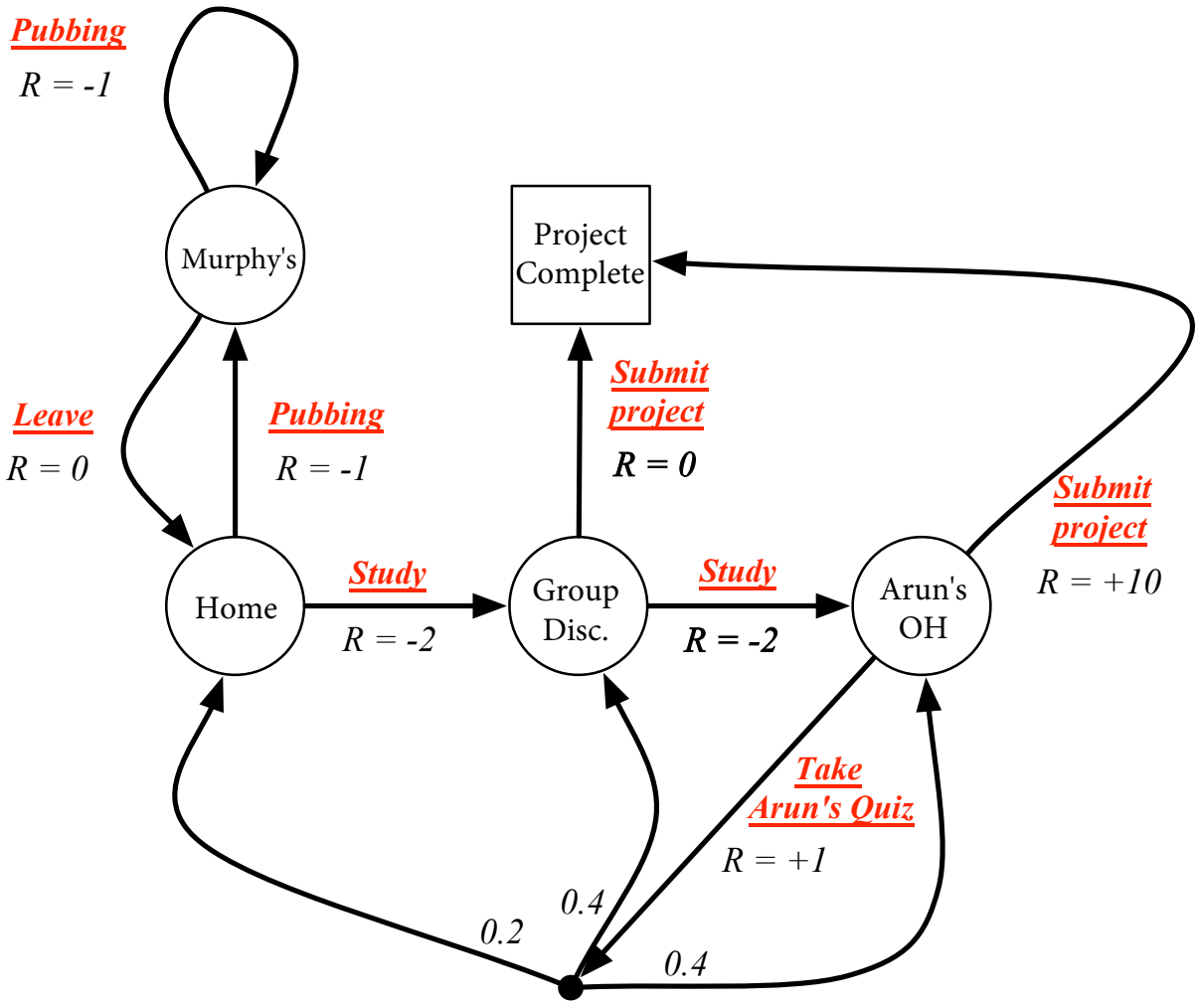
Sample model of RL problem



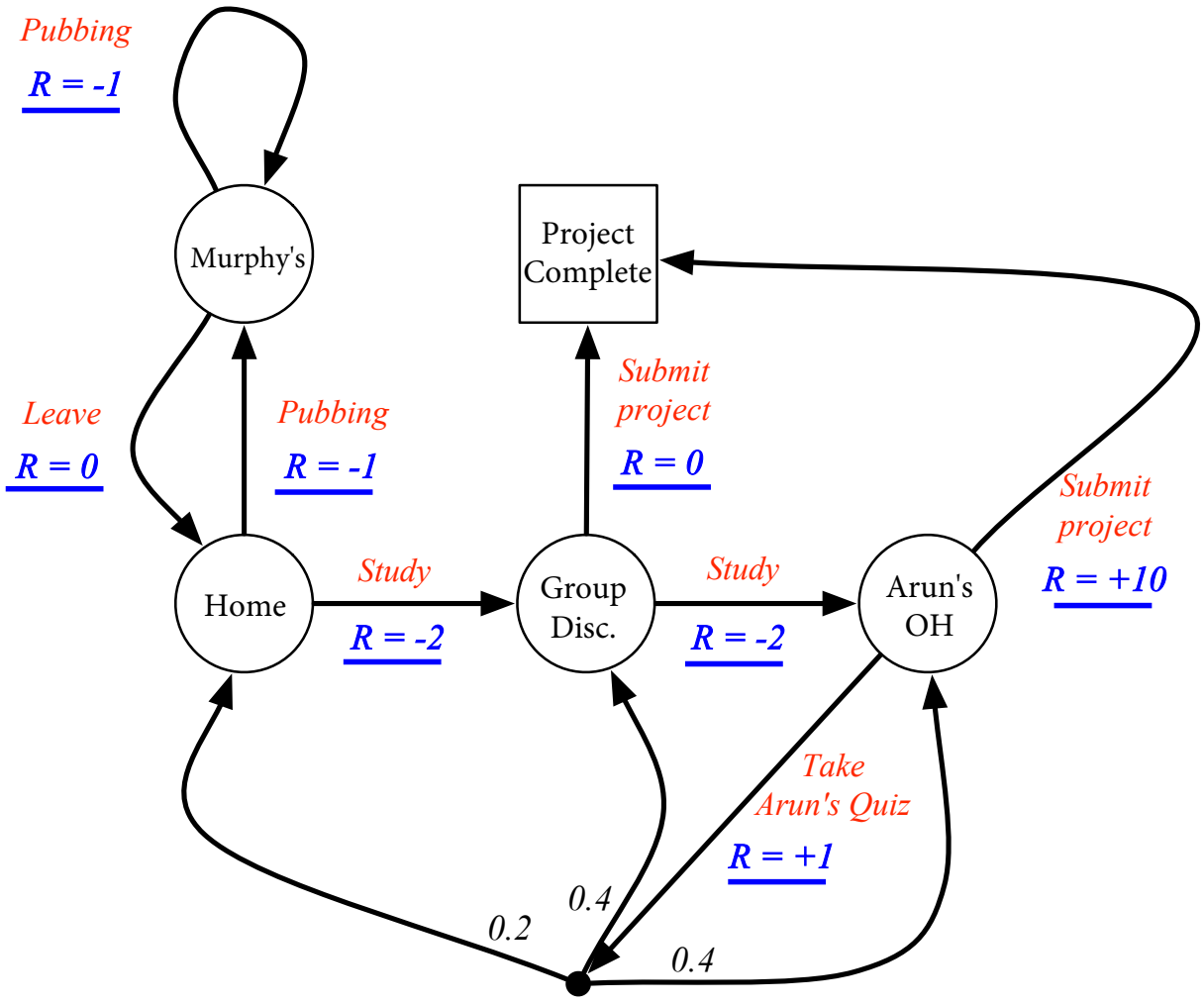
States


Pubbing
R = -1
Murphy's
Project Complete
Leave
R = 0
Pubbing
R = -1
Submit project
R = 0
Submit project
R = +10
Home
Study
R = -2
Group Disc.
Study
R = -2
Arun's OH
Take Arun's Quiz
R = +1
0.2
0.4
0.4

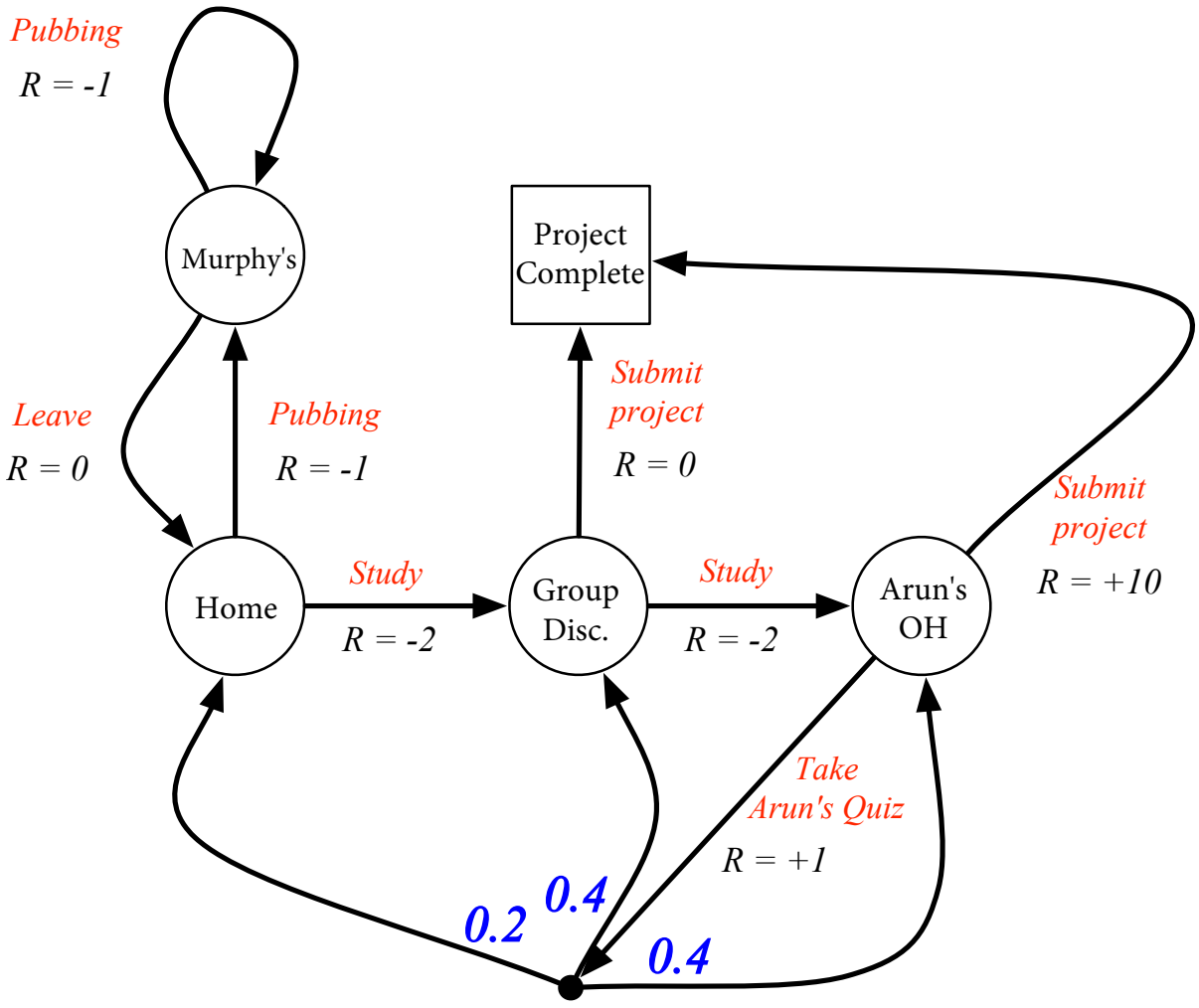
Actions


Pubbing
R = -1
Murphy's
Project Complete
Leave
R = 0
Pubbing
R = -1
Submit project
R = 0
Submit project
R = +10
Home
Study
R = -2
Group Disc.
Study
R = -2
Arun's OH
Take Arun's Quiz
R = +1
0.2
0.4
0.4

Rewards


Pubbing
R = -1
Murphy's
Project Complete
Leave
R = 0
Pubbing
R = -1
Submit project
R = 0
Submit project
R = +10
Home
Study
R = -2
Group Disc.
Study
R = -2
Arun's OH
Take Arun's Quiz
R = +1
0.2
0.4
0.4

Transition probabilities



Markov Decision Process

A Markov decision process (MDP) is an *environment* in which all states are **Markov**.

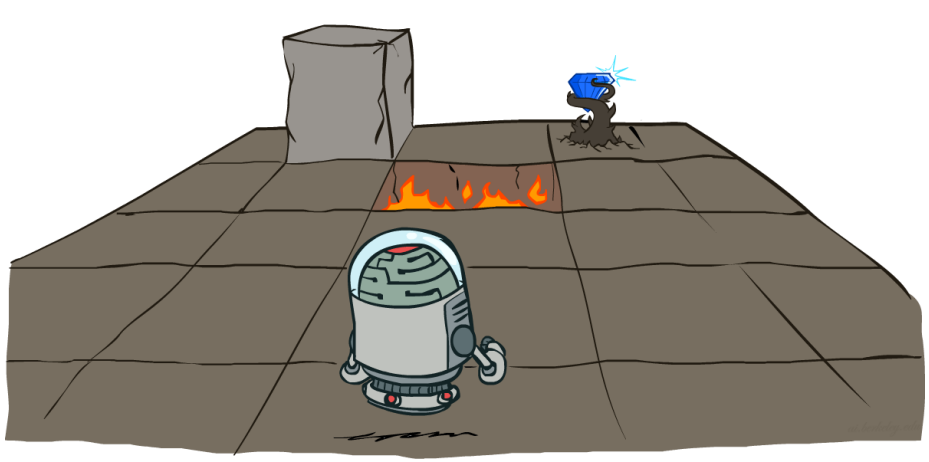
$$\mathbb{P}\left[S_{t+1} \mid S_t, \mathrm{A_t = a}\right] = \mathbb{P}\left[S_{t+1} \mid S_1, ..., S_t, \mathrm{A}_t = \mathrm{a}\right]$$

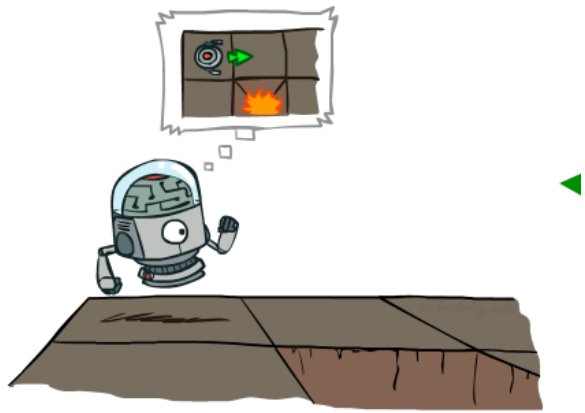
MDP

A *Markov Decision Process* has the following $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix,
- $\mathcal{P}^{a}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$

Major Components of an RL Agent

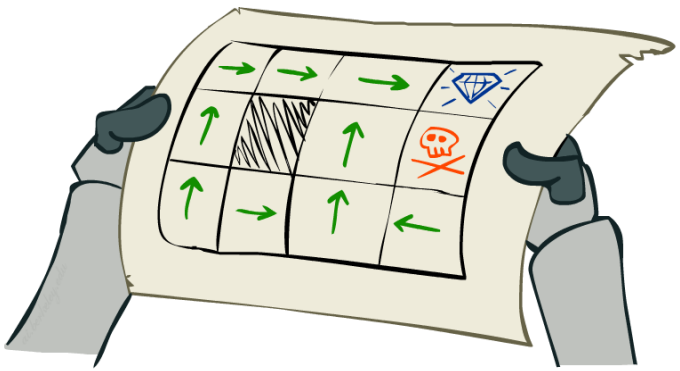




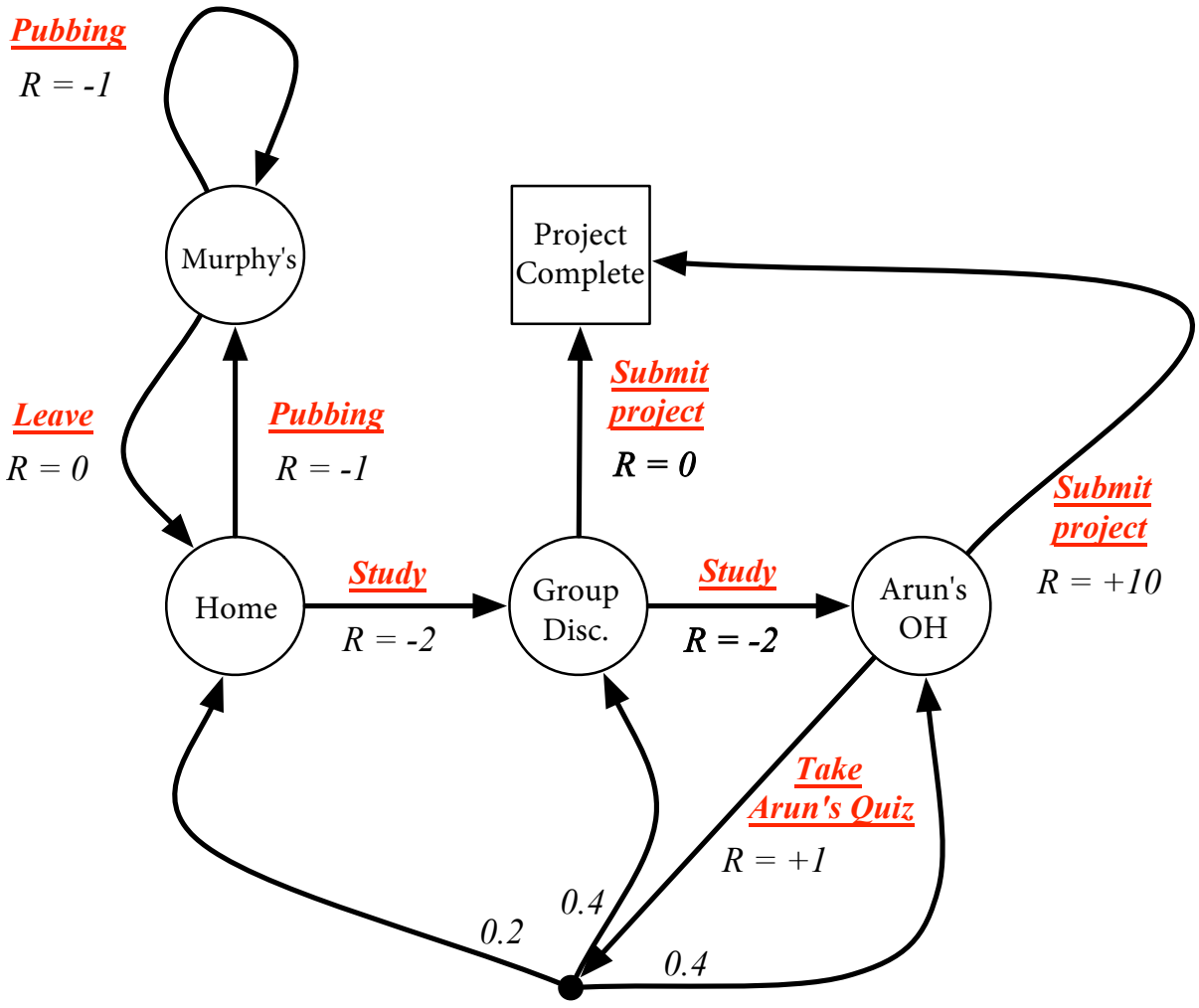
- An RL agent may include one or more of these components:
  - Policy: agent's behaviour function
  - Model: agent's representation of the environment
  - Value function: how good is each state and/or action

Policy

- A policy is the agent's behaviour
- It is a map from state to action, e.g.
- Deterministic policy: $\pi(s) = 1$ for $A_t = \mathrm{a}$
- Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$



Actions

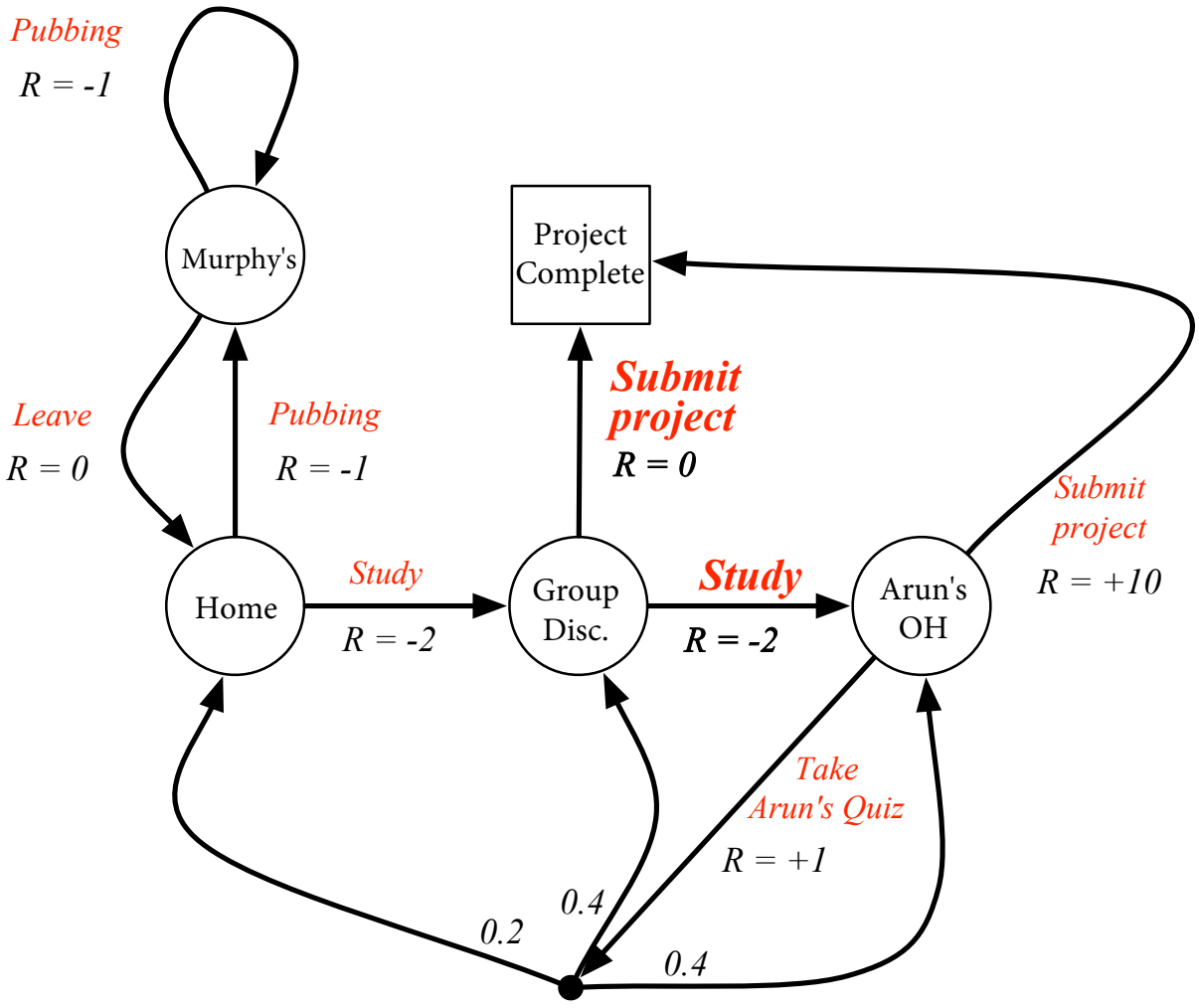

Pubbing
R = -1
Murphy's
Project Complete
Leave
R = 0
Pubbing
R = -1
Submit project
R = 0
Submit project
R = +10
Home
Study
R = -2
Group Disc.
Study
R = -2
Arun's OH
Take Arun's Quiz
R = +1
0.2
0.4
0.4

Model

- A model predicts what the environment will do next
- $\mathcal{P}$: Transition probabilities
- $\mathcal{R}$: Expected rewards

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
$$\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

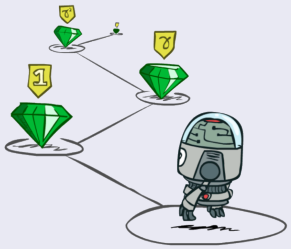
Beyond Rewards



Value function - Concept of Return

Return G_t

The *return* G_t is cumulative **discounted** reward from time-step t.



$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The *discount* $\gamma \in [0, 1]$ is the present value of future rewards
- This values immediate reward above delayed reward.
- Avoids infinite returns in cyclic Markov processes

1

Worth Now

γ

Worth Next Step

γ^2

Worth In Two Steps

Value Function

State Value Function $v_\pi(s)$

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right]$$

$v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π

Action Value Function $q_\pi(s,a)$

$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right]$$

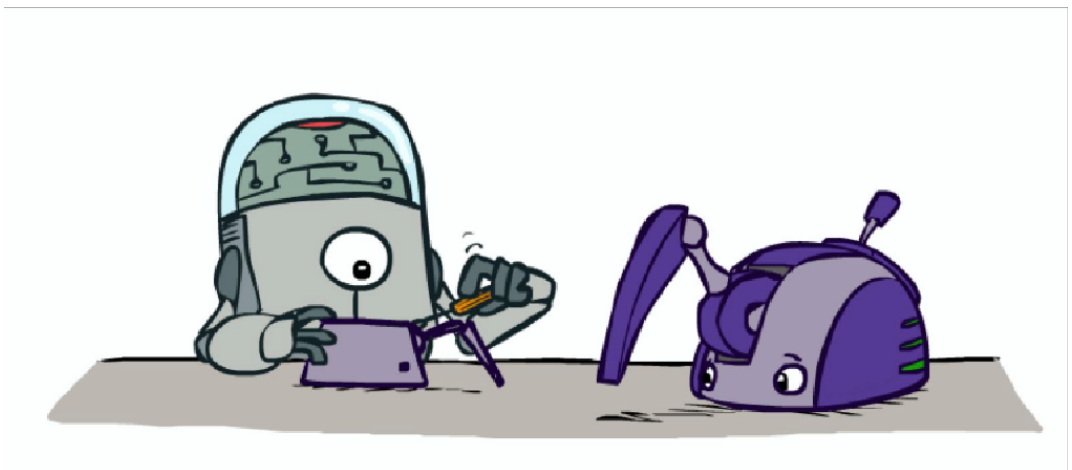
$q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π

Subproblems in RL

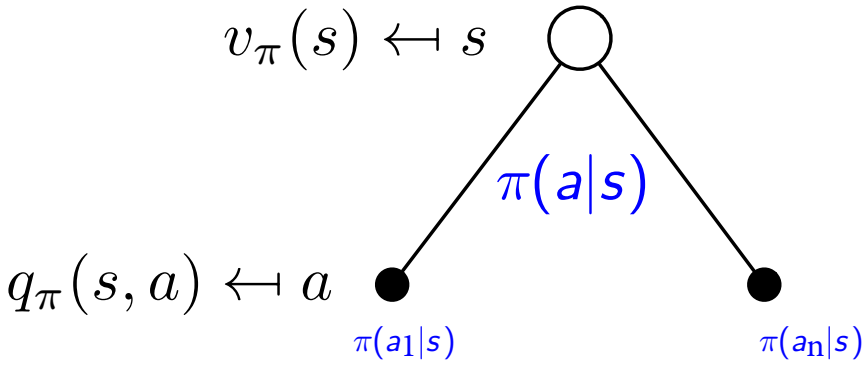
- Model based
- Model free

- Prediction: evaluate the future
  - Given a policy
- Control: optimise the future
  - Find the best policy

Part 2: Model-based Prediction and Control

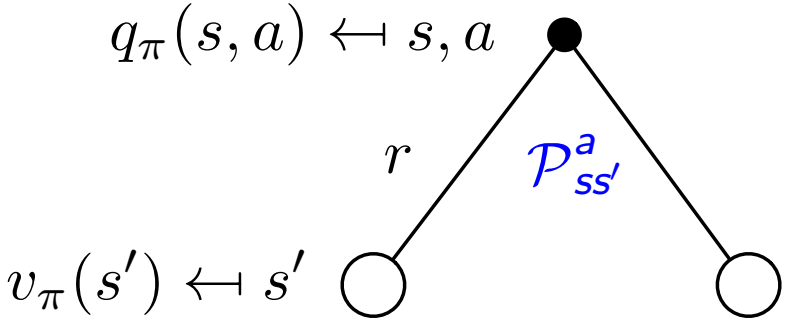


Connecting v(s) and q(s,a): Bellman equations



v in terms of q :

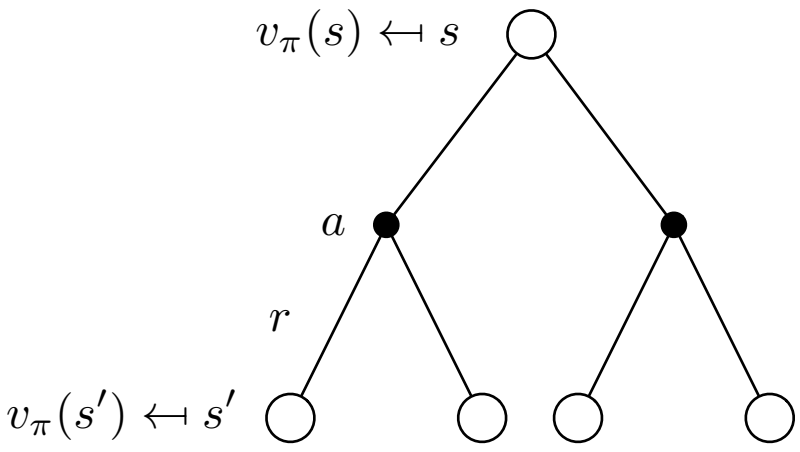
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a)$$



q in terms of v :

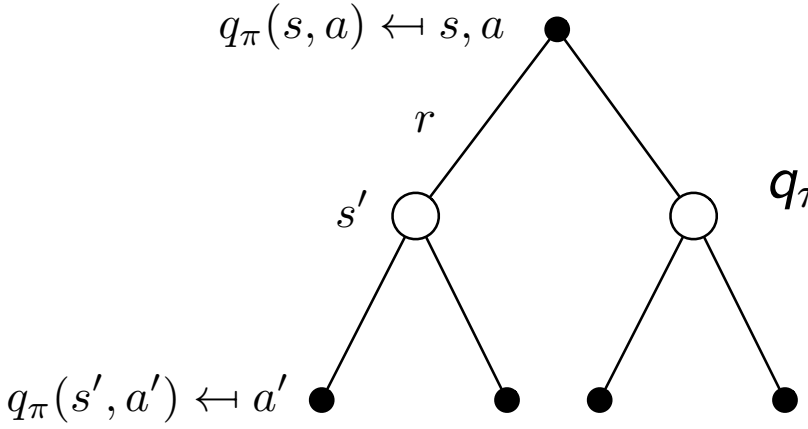
$$q_\pi(s, a) = \mathcal{R}^a_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} v_\pi(s')$$

Connecting v(s) and q(s,a): Bellman equations (2)



v in terms of other v :

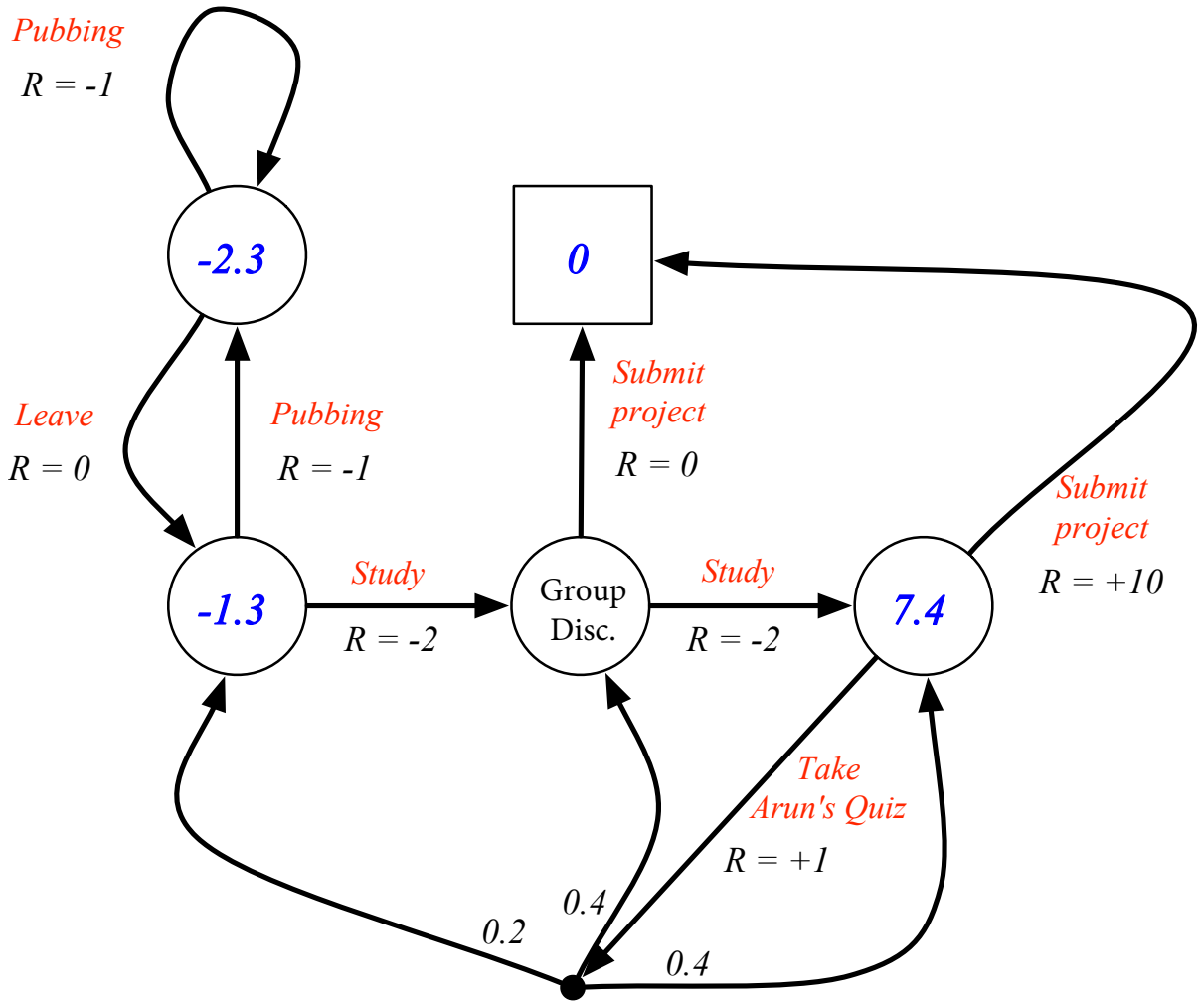
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$



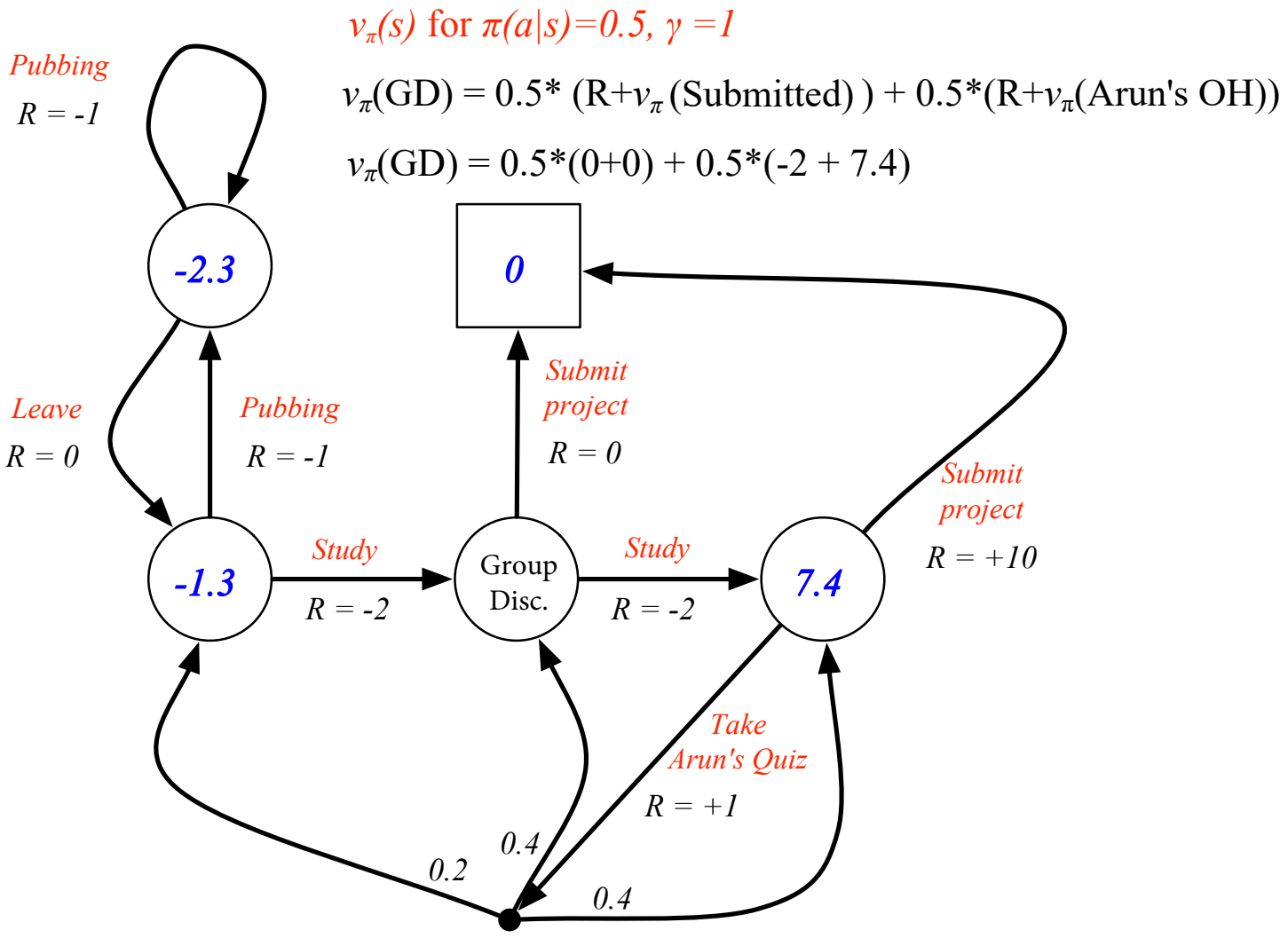
q in terms of other q :

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s', a')$$

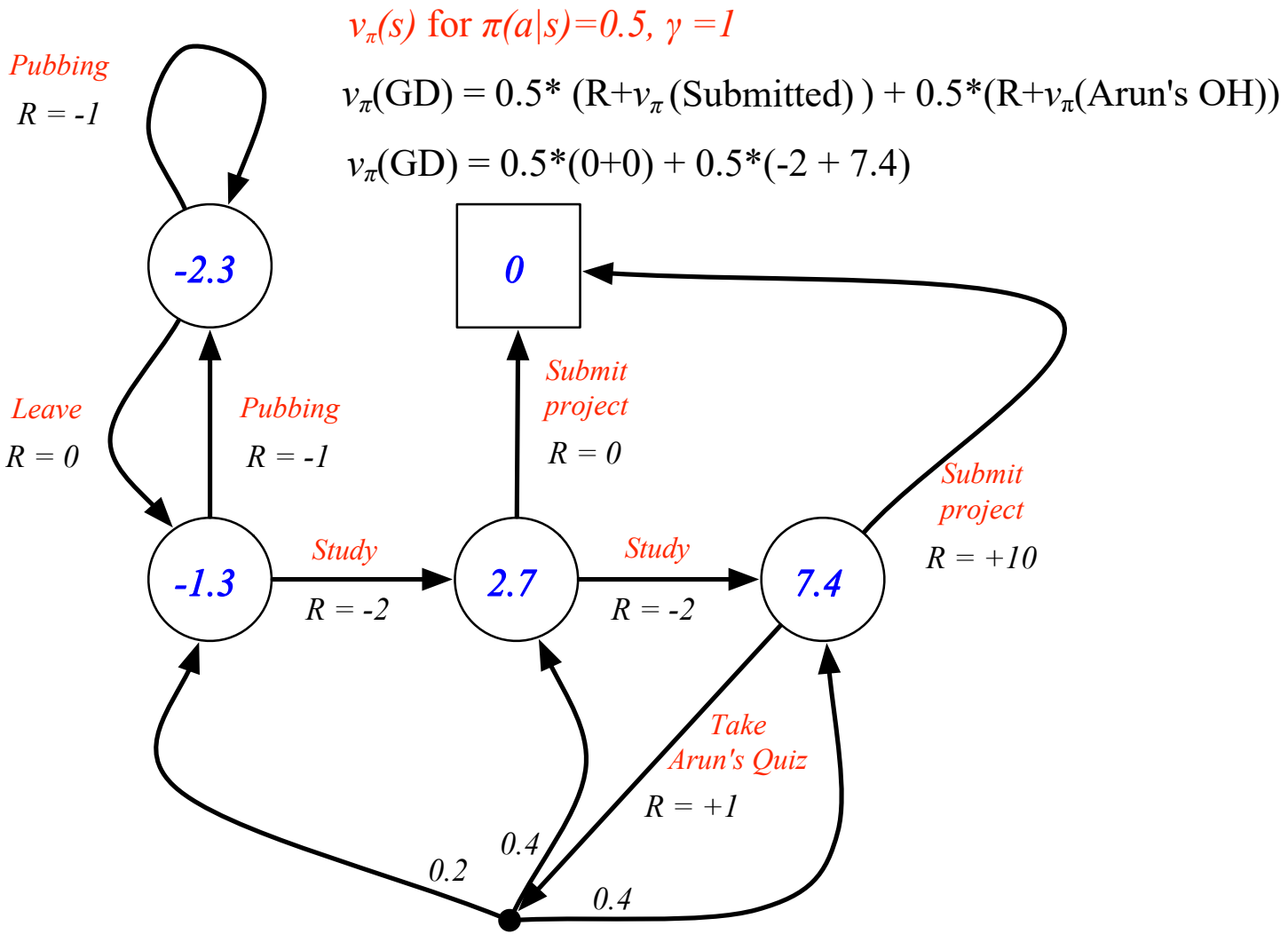
Example: $v_\pi(s)$



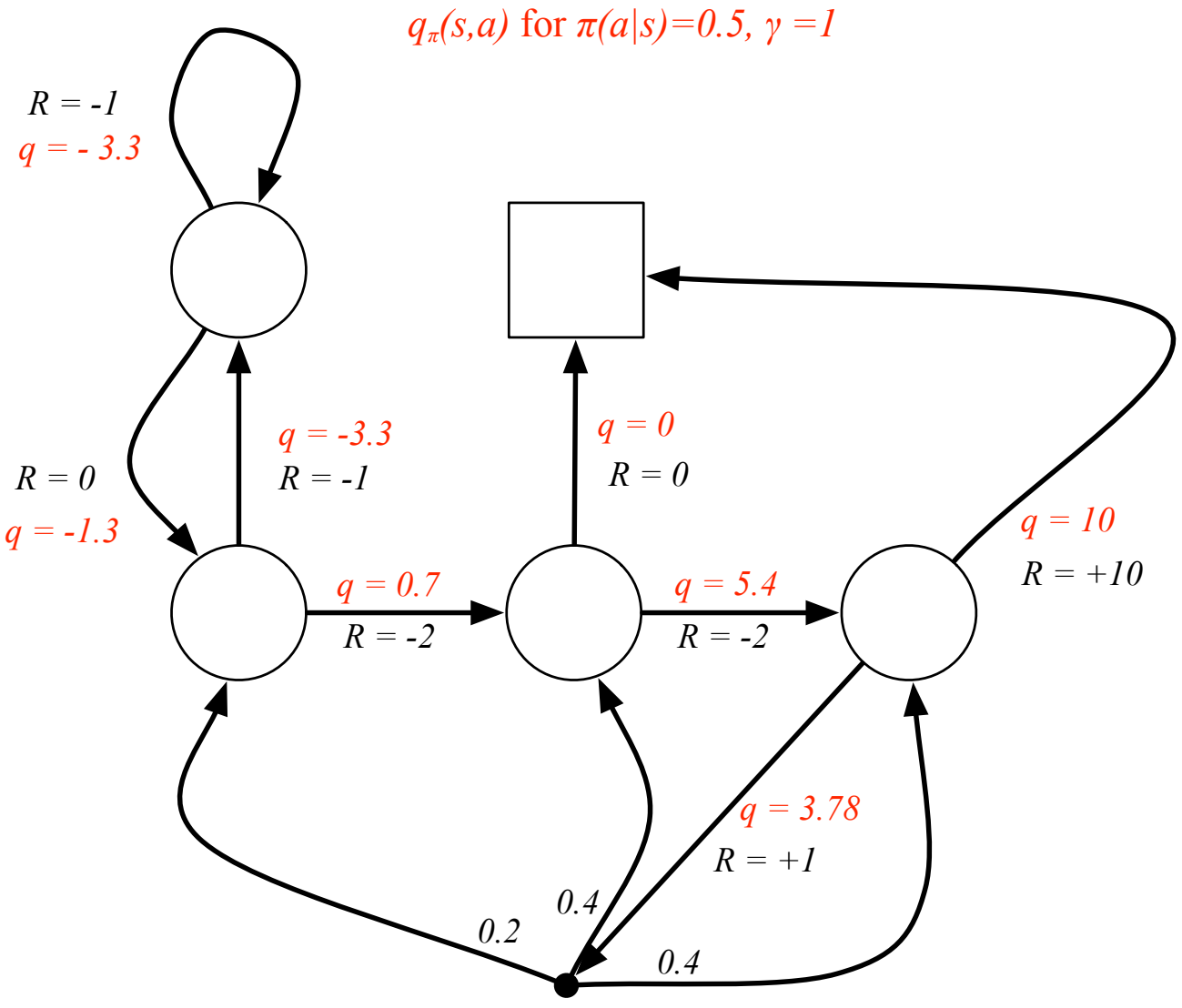
Example: $v_\pi(s)$



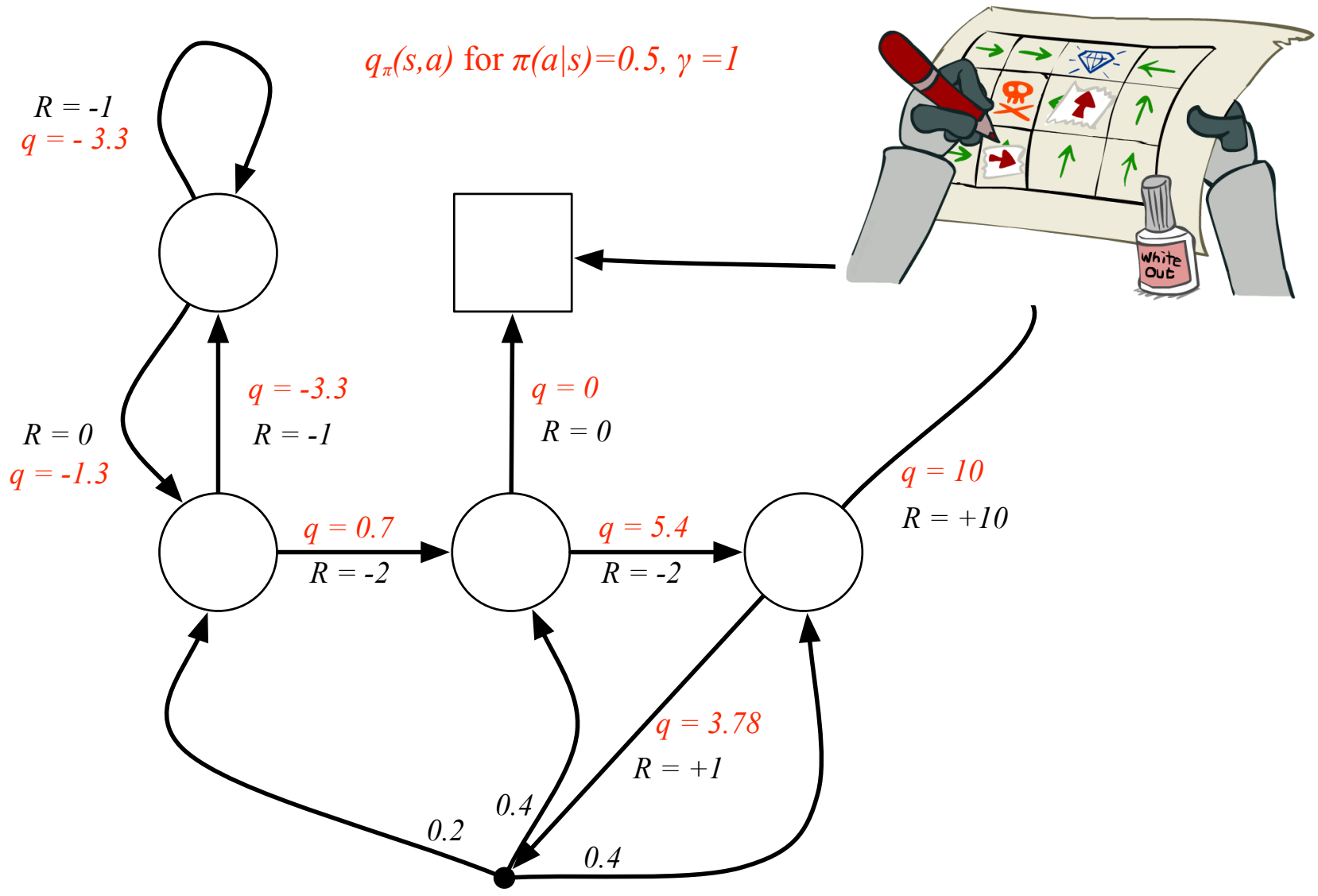
Example: $v_\pi(s)$



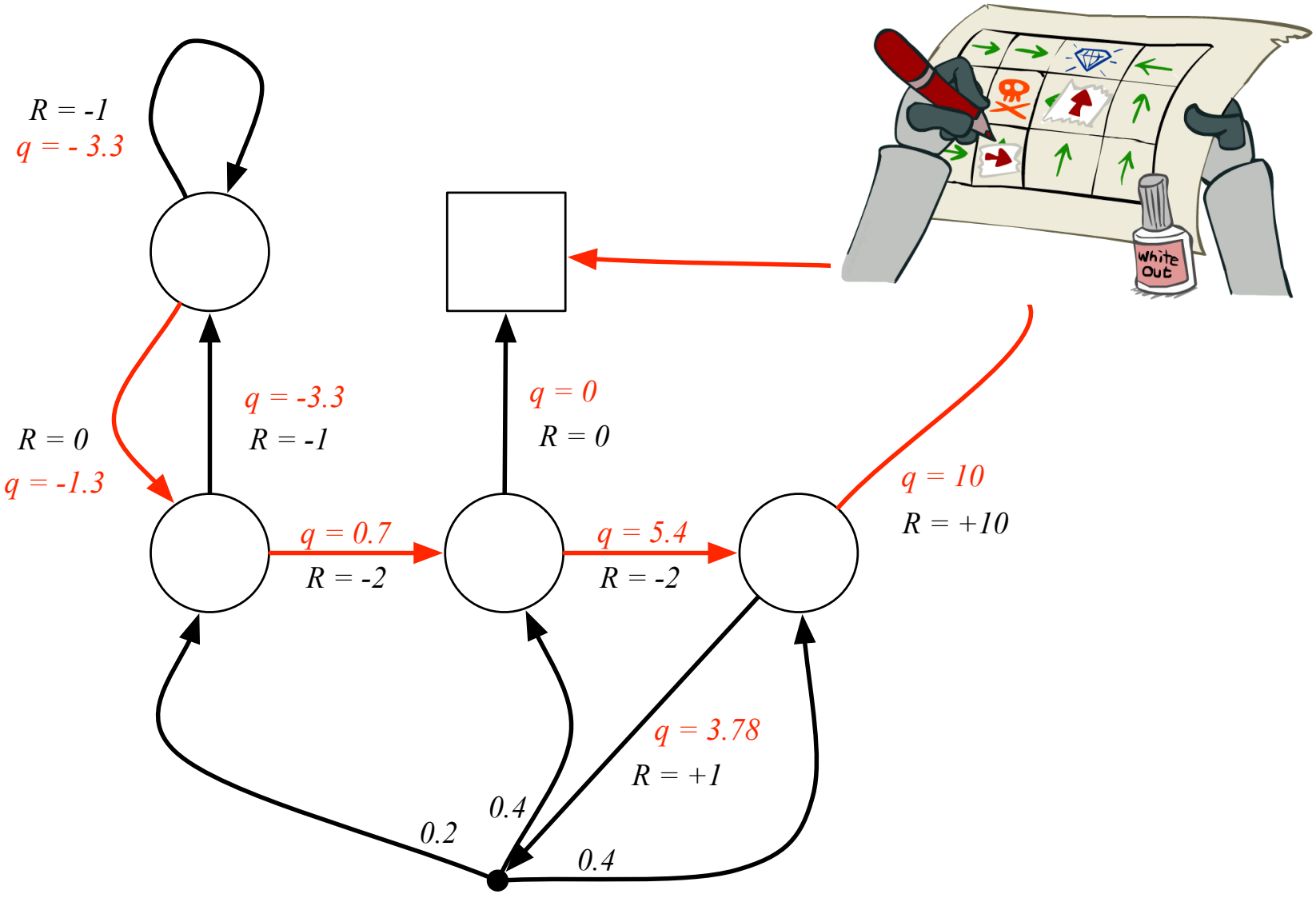
Example: $q_\pi(s,a)$



Example: $q_\pi(s,a)$

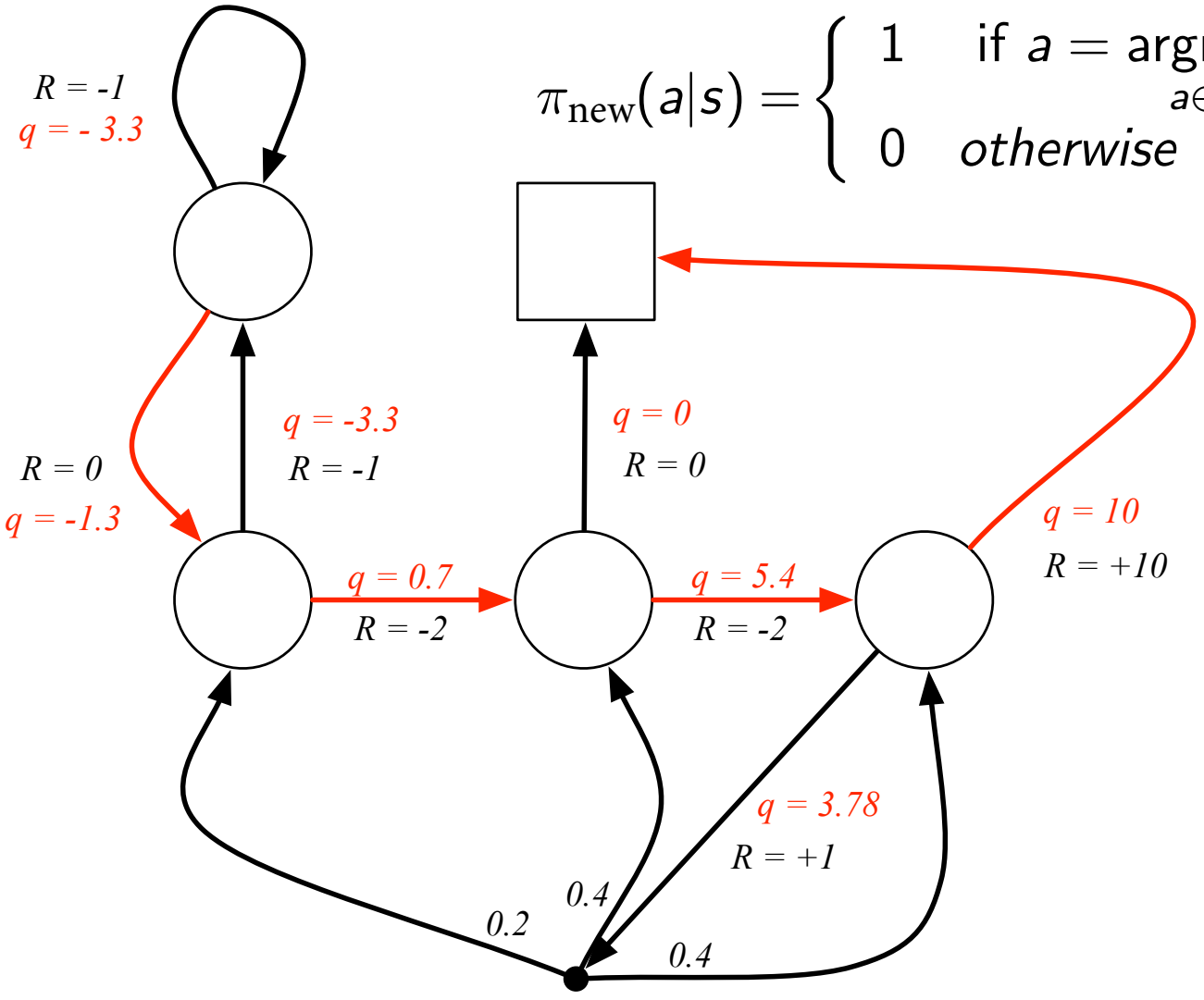


Example: Policy improvement

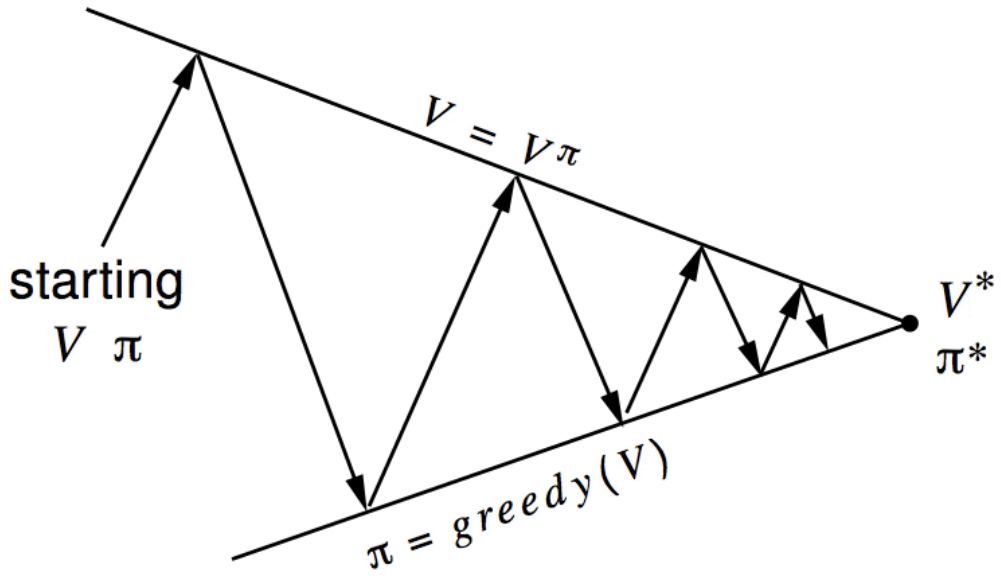


Example: Policy improvement - Greedy

$$\pi_{\text{new}}(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\text{argmax}}\, q_{\text{old}}(s,a) \\ 0 & \text{otherwise} \end{cases}$$



Policy Iteration



Policy evaluation Estimate v_π

Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$

Greedy policy improvement

Iterative Policy Evaluation in Small Gridworld

v_k for the Random Policy

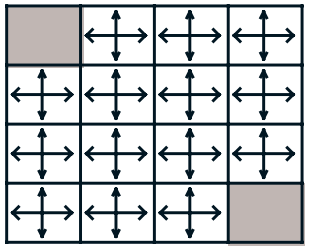
Greedy Policy update w.r.t. v_k

States: 14 cells + 2 terminal cells
Actions: 4 directions
Rewards: -1 for time step

random policy

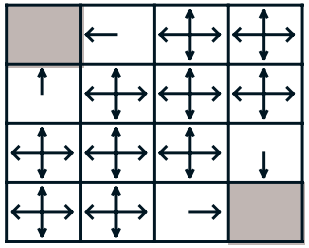
$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



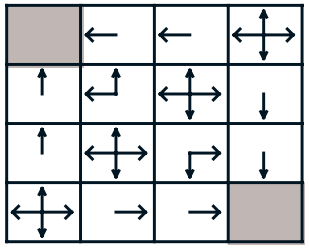
$k = 1$

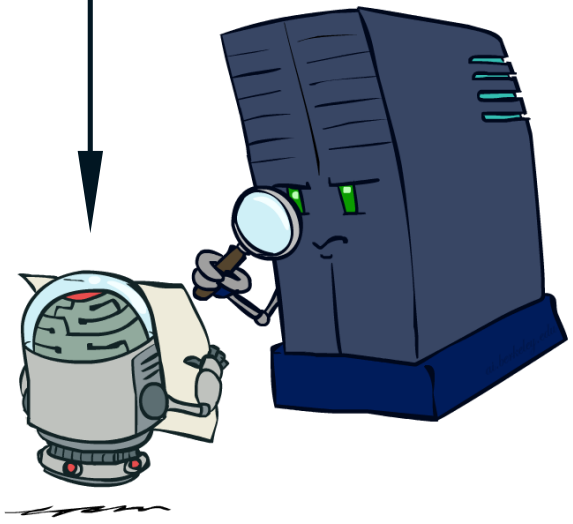
0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0



$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

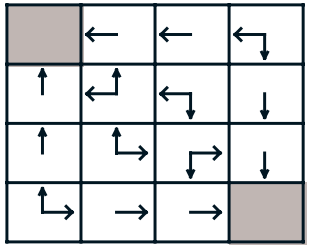




Iterative Policy Evaluation in Small Gridworld (2)

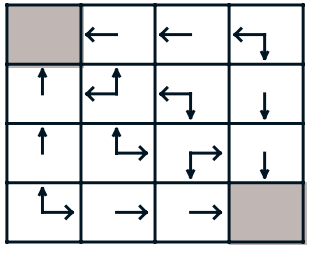
$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0



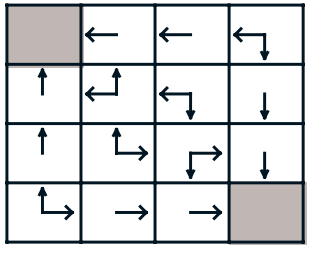
$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0



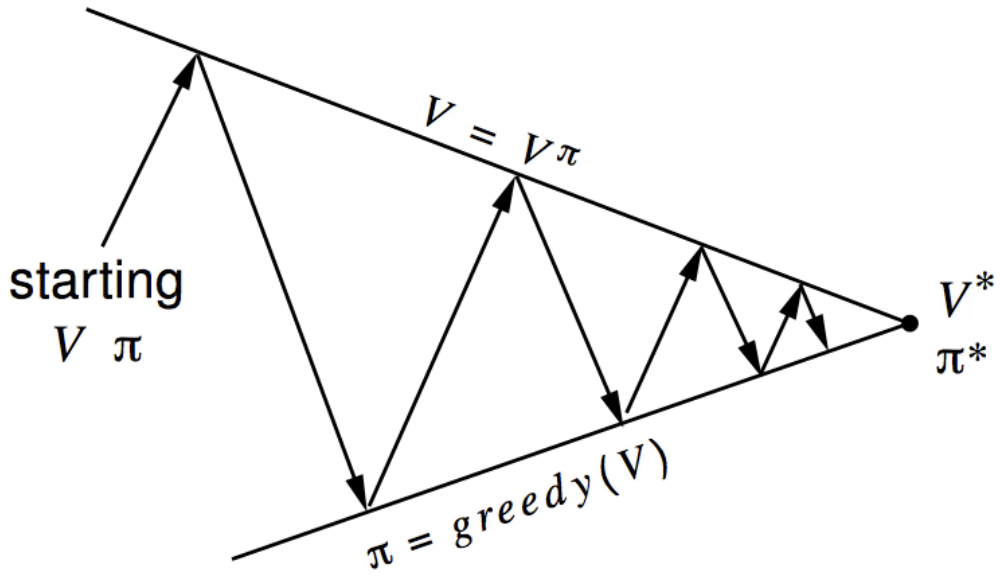
$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0



Saturated policy

Policy Iteration



Policy evaluation Estimate v_π

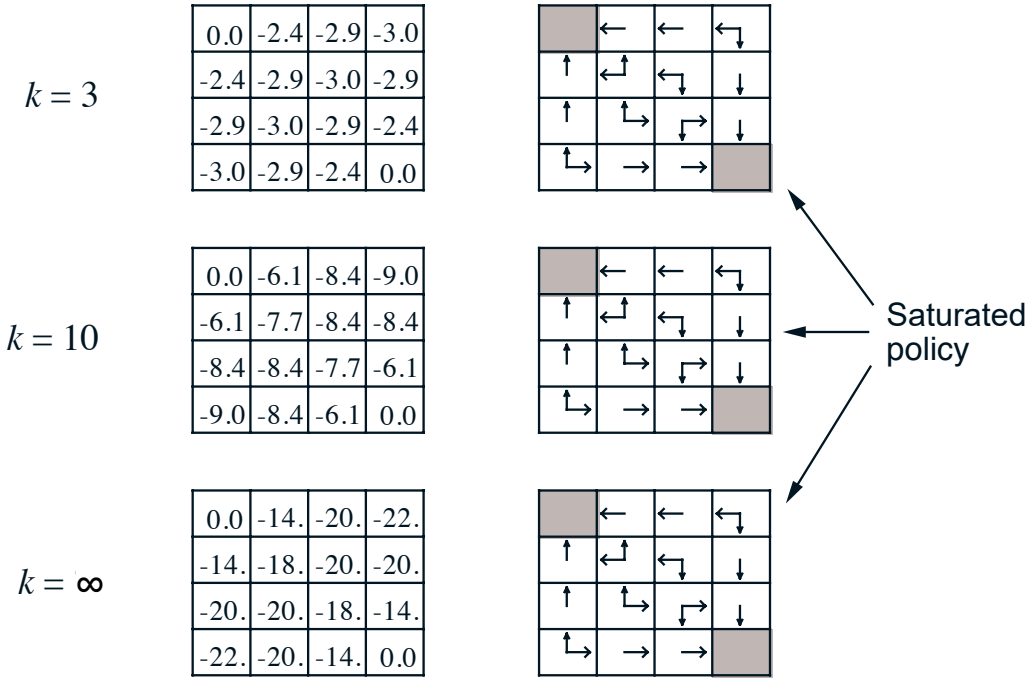
Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$

Greedy policy improvement

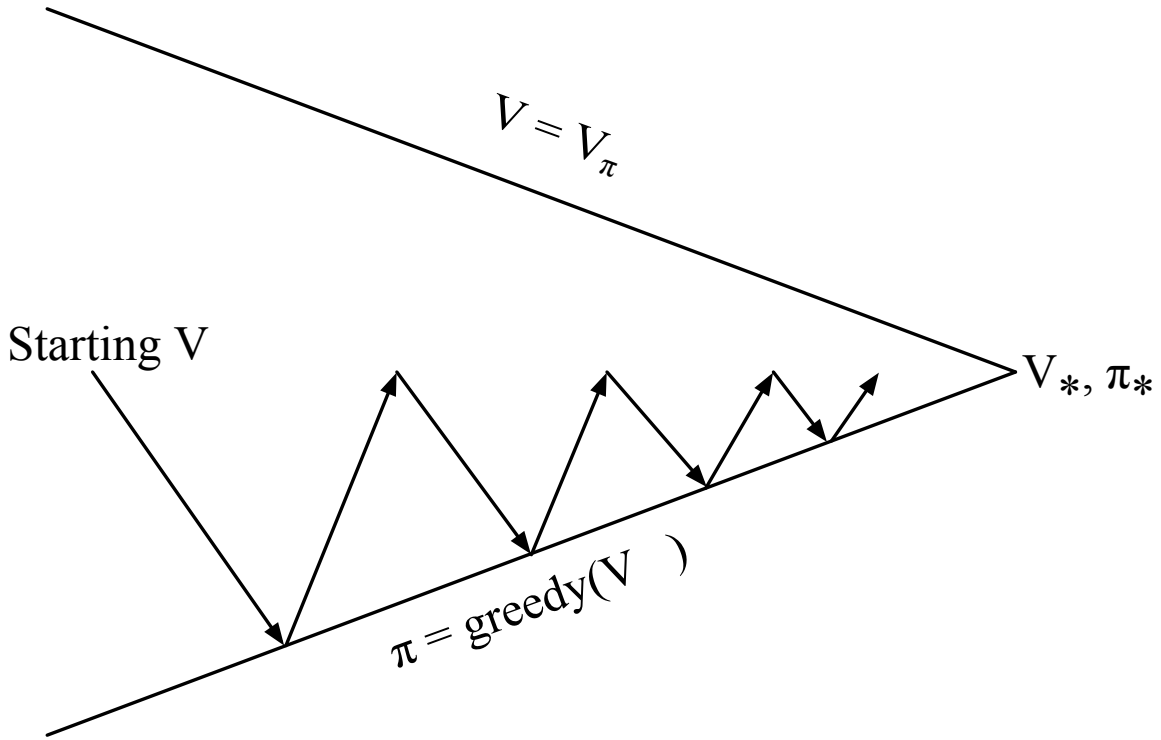
Modified Policy Iteration - Value Iteration

- Policy converges faster than value function
- In the small gridworld $k = 3$ was sufficient to achieve optimal policy
- Why not update policy every iteration? i.e. stop after $k = 1$
  - This is *value iteration*

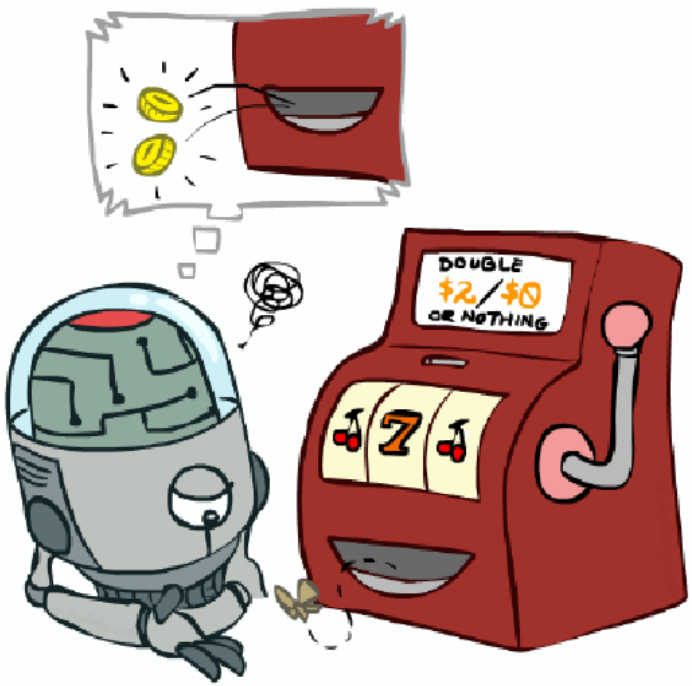


Modified Policy Iteration - Value Iteration

- Policy converges faster than value function
- In the small gridworld $k = 3$ was sufficient t o achieve optimal policy
- Why not update policy every iteration? i.e. stop after $k = 1$
  - This is *value iteration*



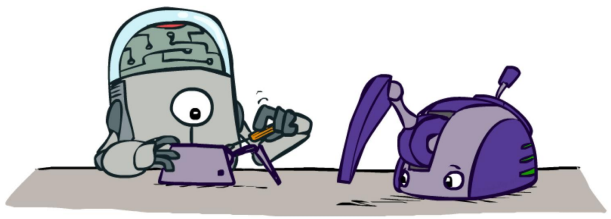
Part 3: Model-Free Prediction

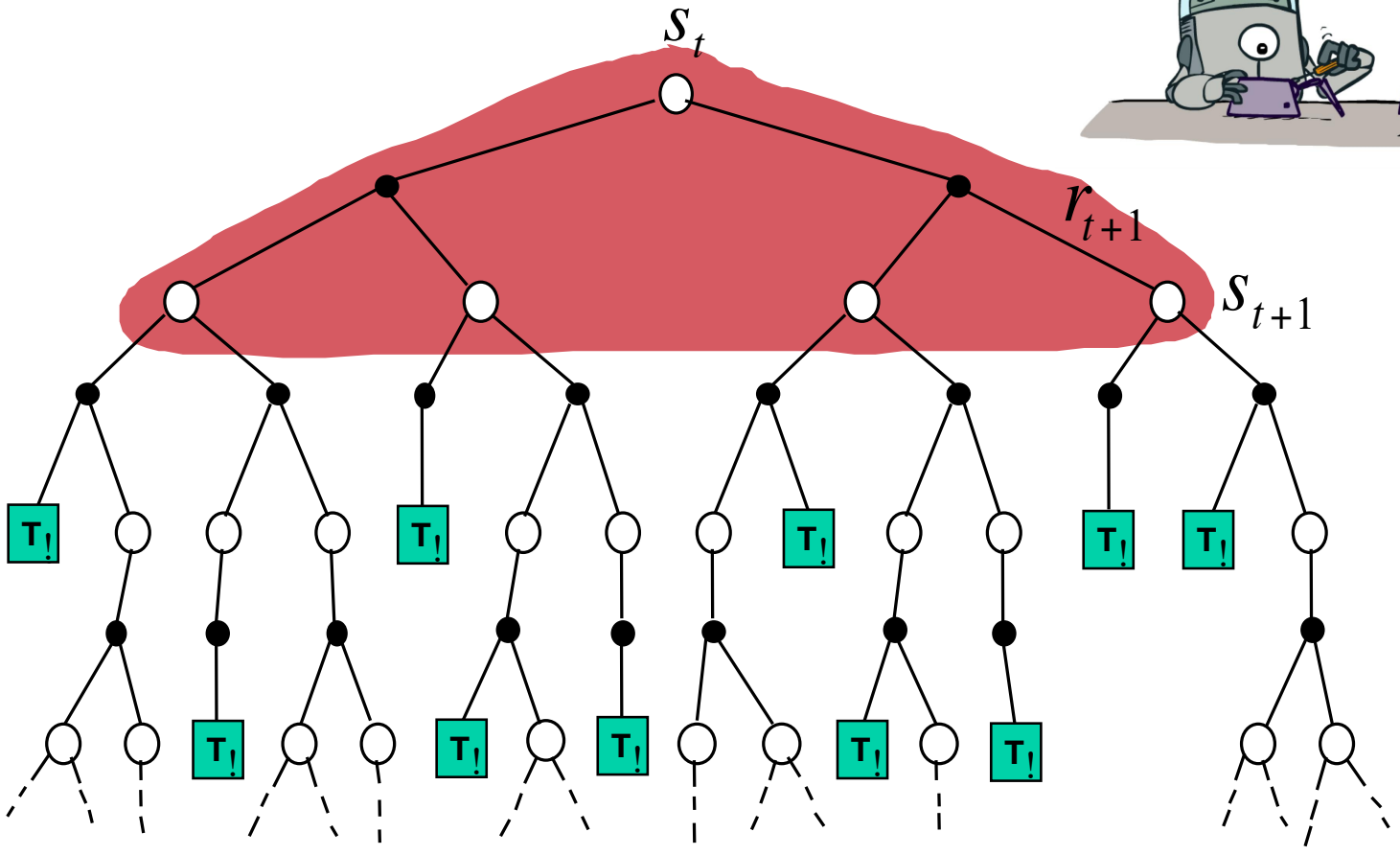


Bellman Equation Estimate

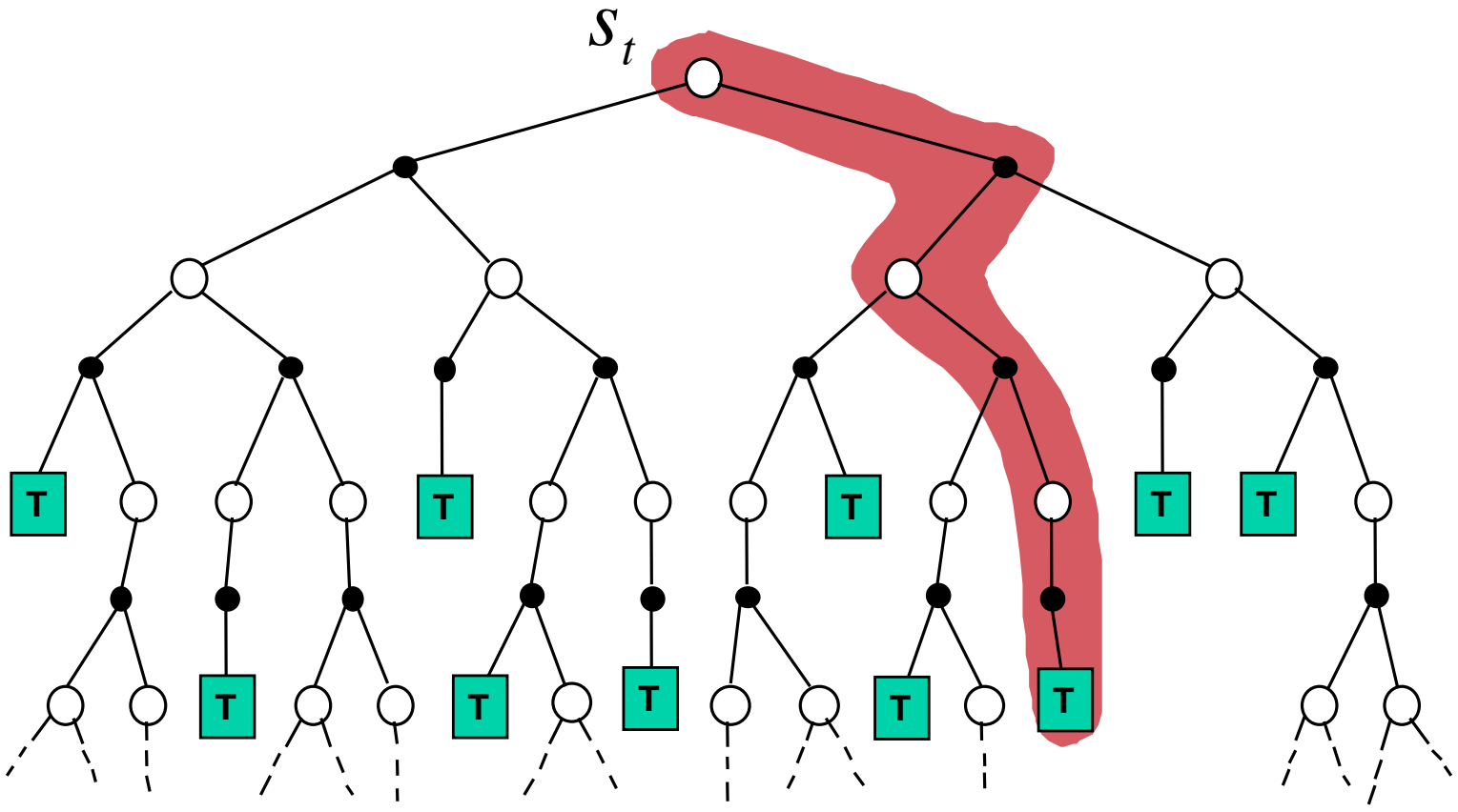
***v* in terms of other *v* :**

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$





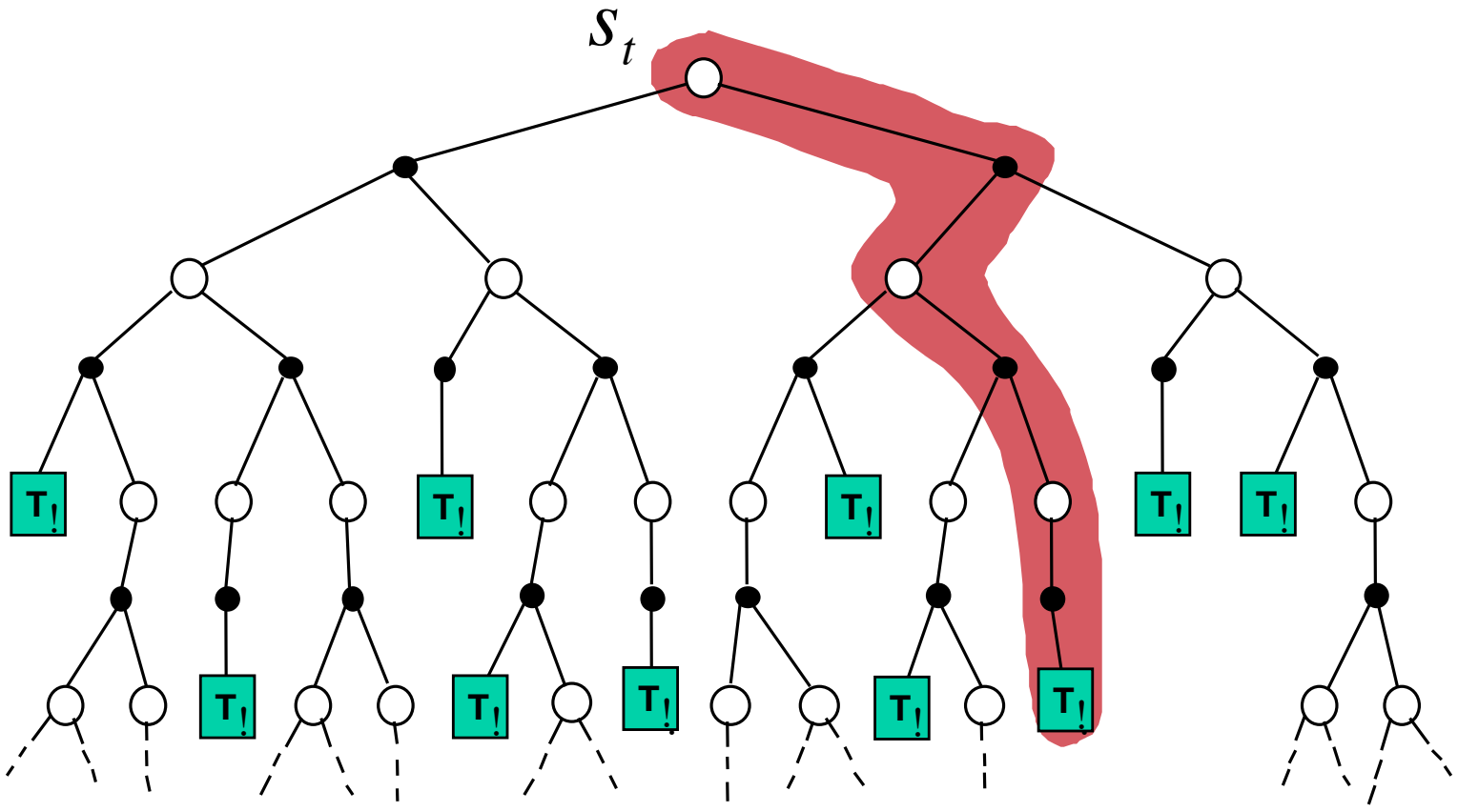
Monte-Carlo Sampling



Monte-Carlo Estimate

$$v_\pi(s) = \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \ldots | S_t = s\right] \quad \text{[actual]}$$

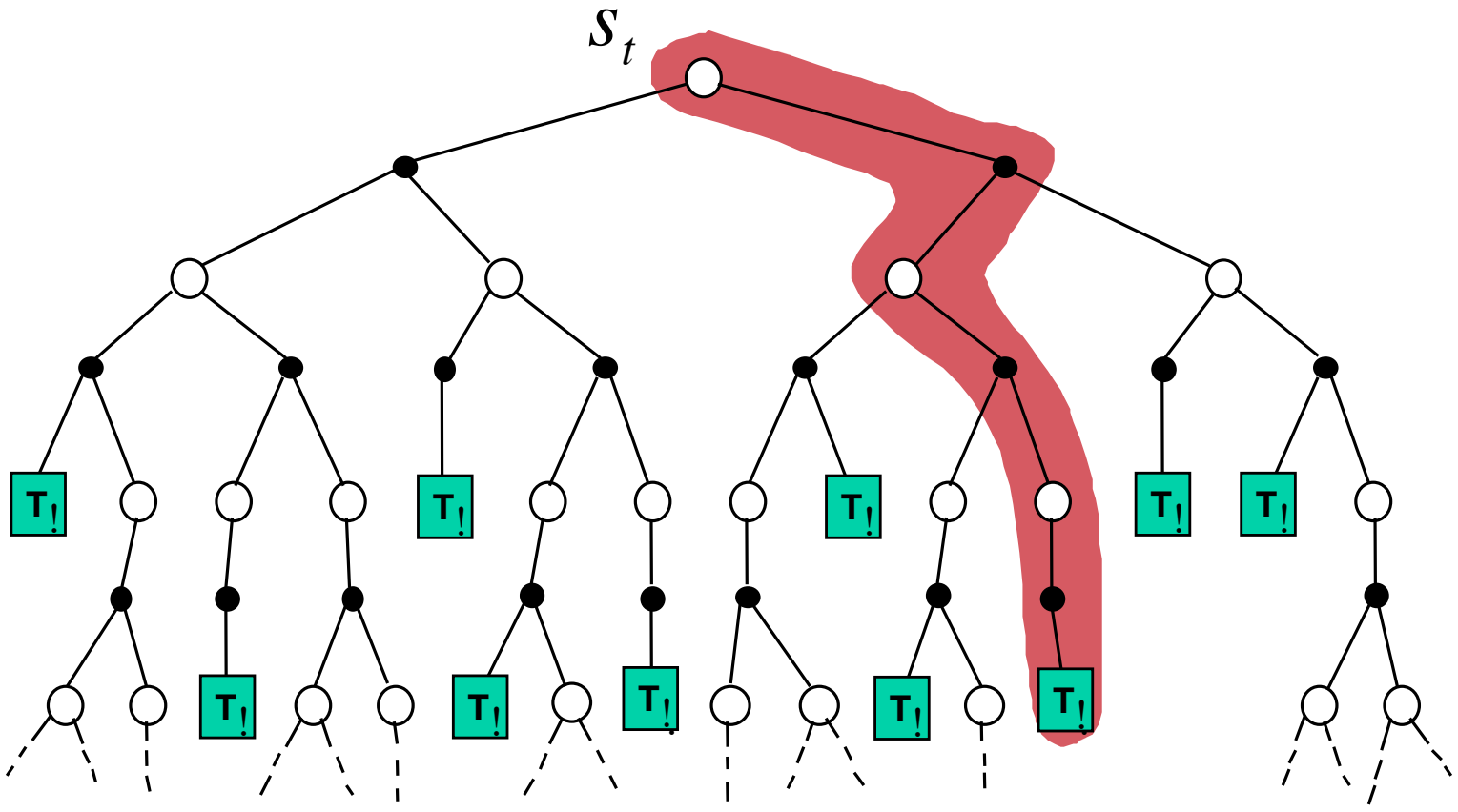
$$V(S_t) \quad \text{[estimate]}$$



Monte-Carlo Estimate

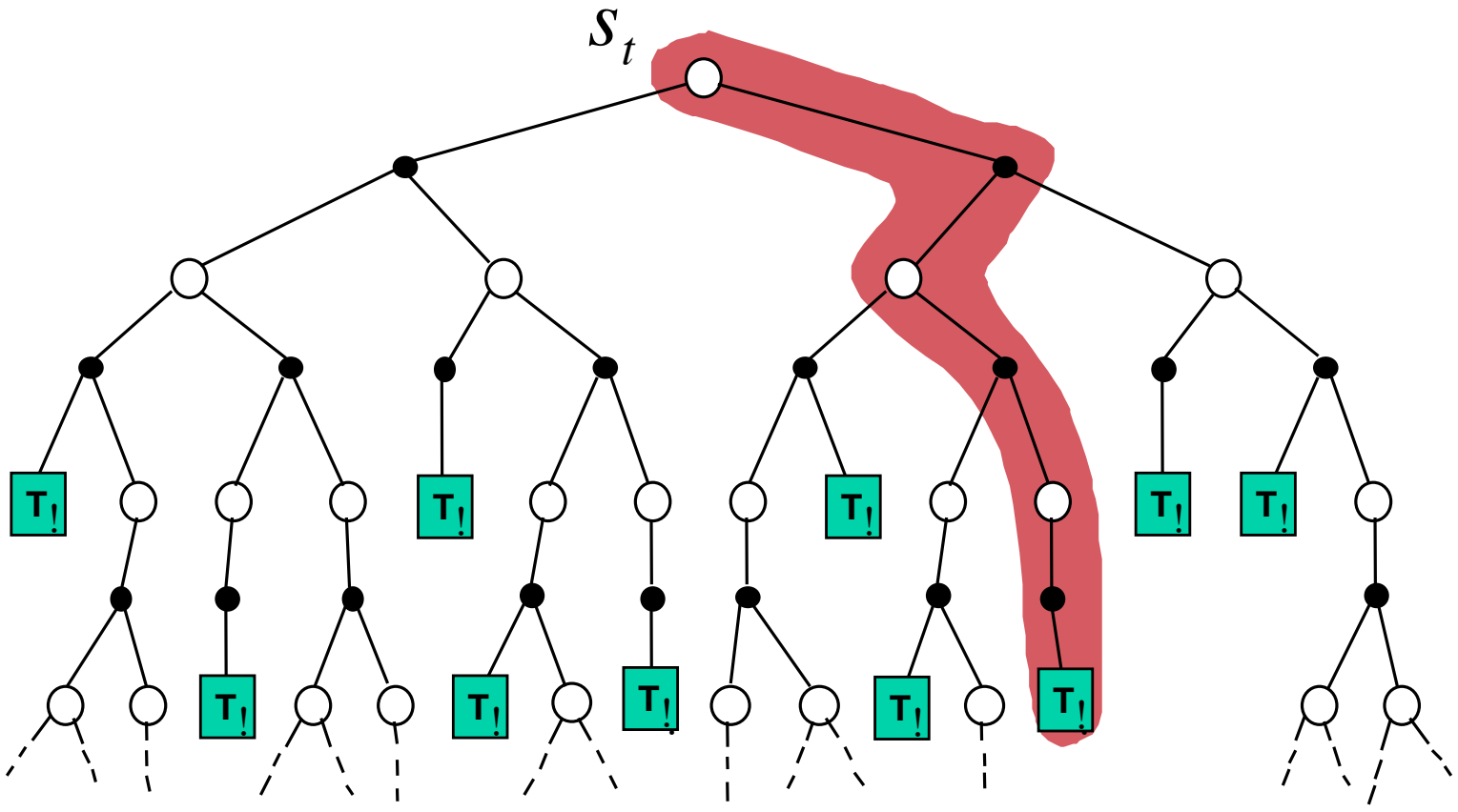
$$v_\pi(s) = \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \ldots | S_t = s\right]$$

$$V(S_t) := V(S_t) + \alpha\left(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots - V(S_t)\right)$$



Monte-Carlo Estimate

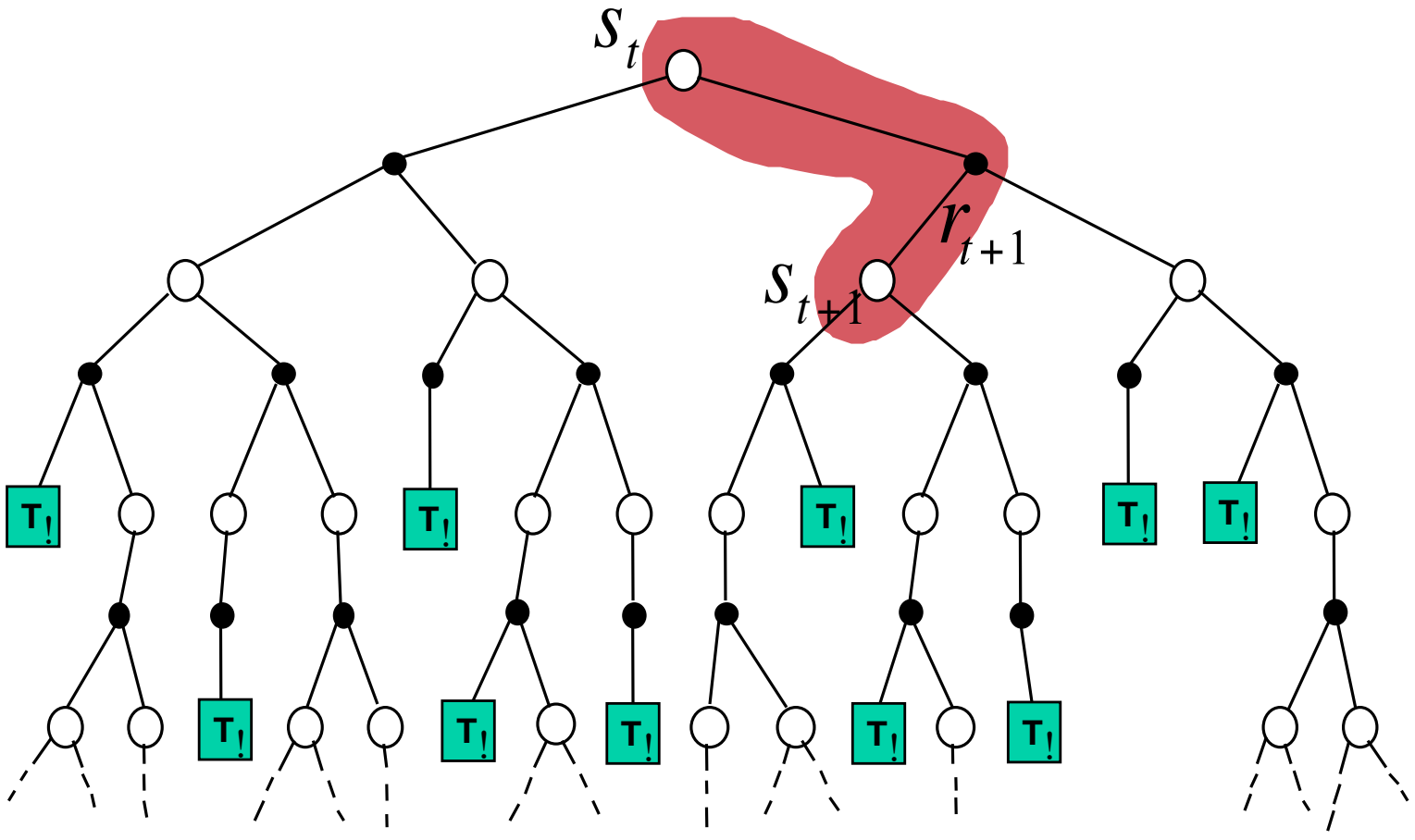
$$V(S_t) := V(S_t) + \alpha\,(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots - V(S_t))$$



Temporal-Difference Estimate

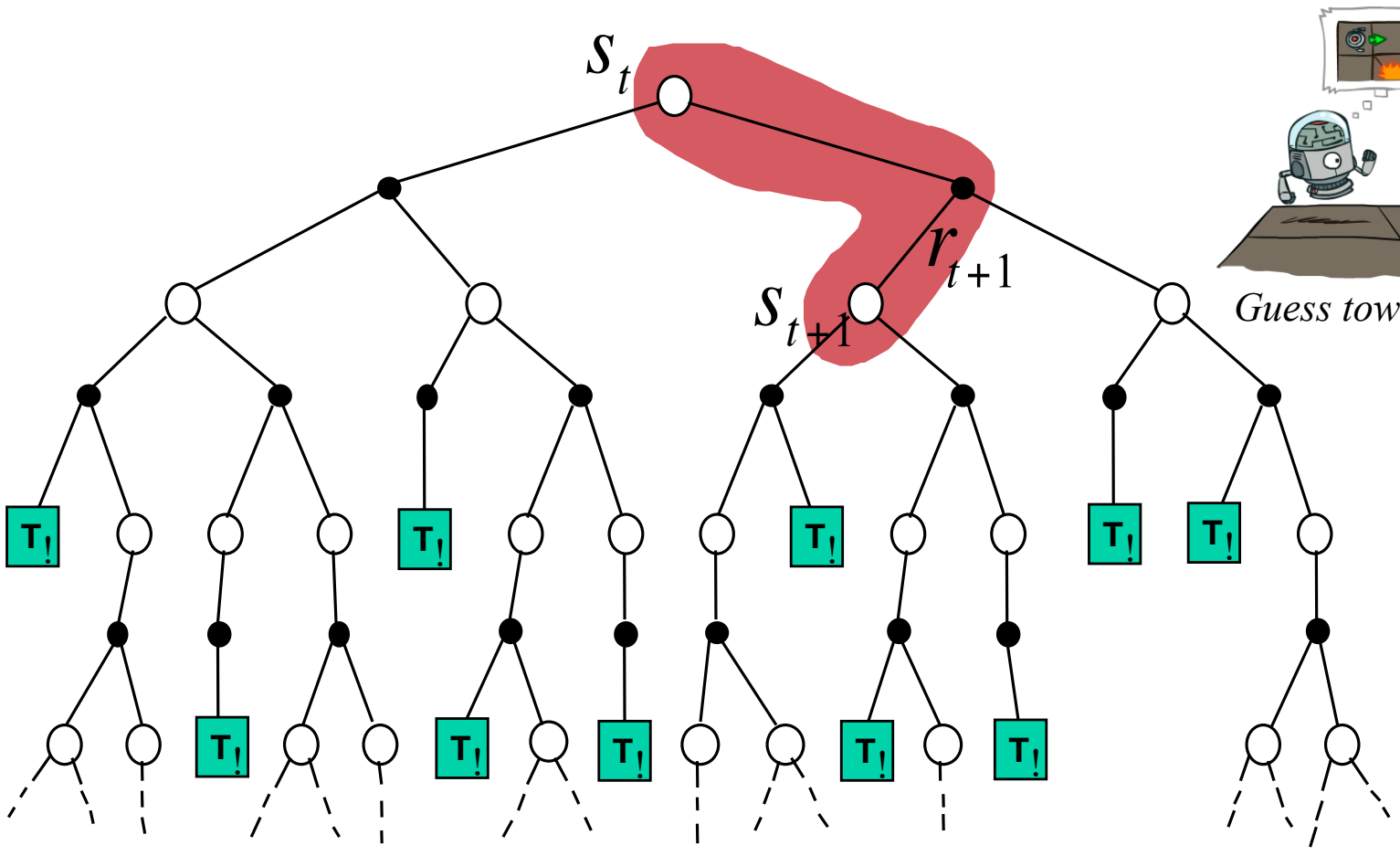
$$V(S_t) := V(S_t) + \alpha\,(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots - V(S_t))$$

$$V(S_t) := V(S_t) + \alpha\,(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$



Temporal-Difference Estimate

$$V(S_t) := V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$



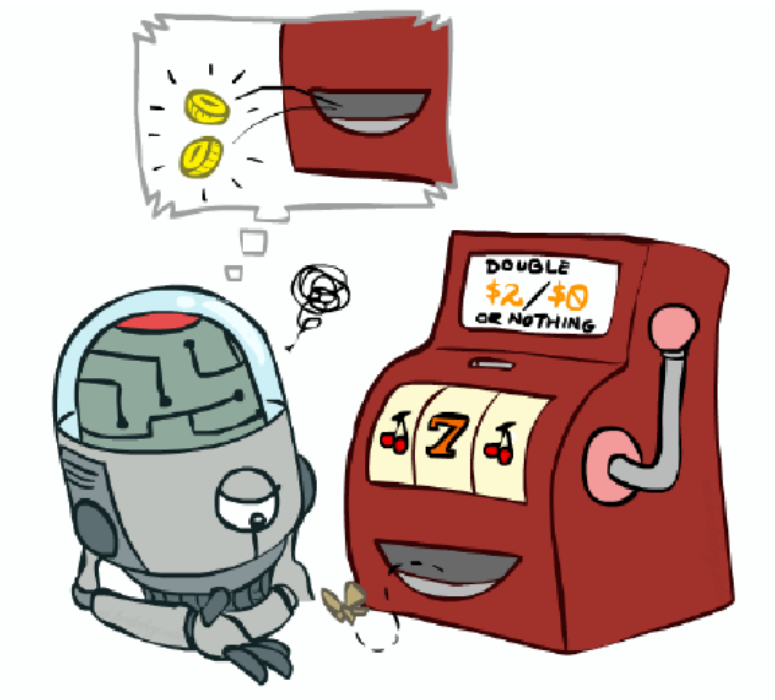
MC vs. TD

$$\text{MC: } V(S_t) := V(S_t) + \alpha\left(\boldsymbol{R_{t+1}} + \gamma \boldsymbol{R_{t+2}} + \boldsymbol{\gamma^2 R_{t+3}} \ldots - V(S_t)\right)$$

$$\text{TD: } V(S_t) := V(S_t) + \alpha\left(\boldsymbol{R_{t+1}} + \gamma \boldsymbol{V(S_{t+1})} - V(S_t)\right)$$

- TD can learn *before* knowing the final outcome
- TD target $R_{t+1} + \gamma V(S_{t+1})$ is *biased* estimate of $R_{t+1} + \gamma v_\pi(S_{t+1})$
- TD target is much lower variance than MC target

Part 4: Model-Free Control

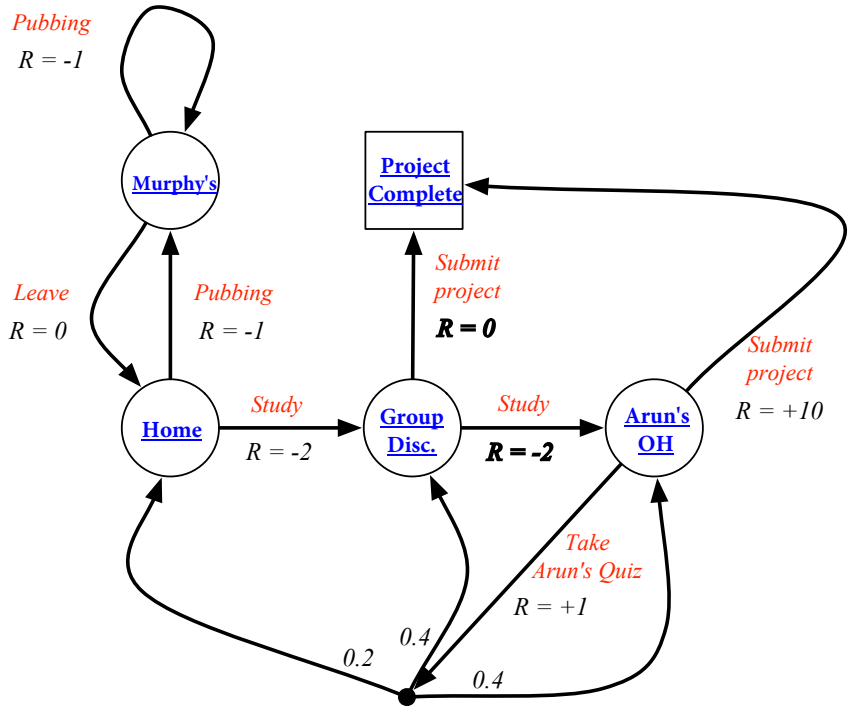


Today's takeaways

- MDP: States, actions
- Environment: Transitions and rewards
- Agent: Policy over actions
- Policy iteration
    - Policy evaluation
    - Policy improvement
- Value Iteration
- Model free policy evaluation
- Model free policy control

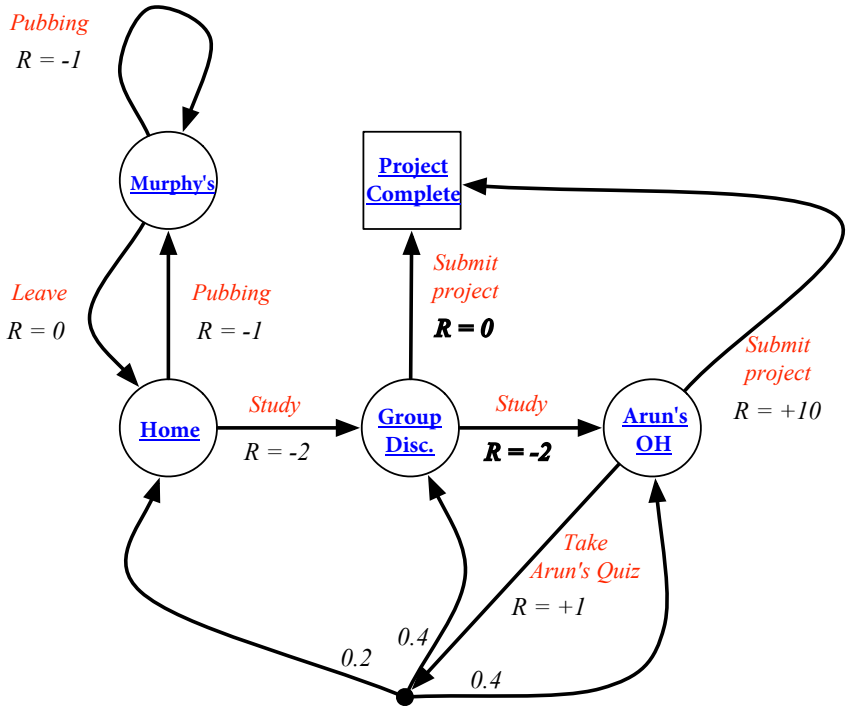
Today's takeaways

- **MDP: States, actions**
- Environment: Transitions and rewards
- Agent: Policy over actions
- Policy iteration
  - Policy evaluation
  - Policy improvement
- Value Iteration
- Model free policy evaluation
- Model free policy control



Today's takeaways

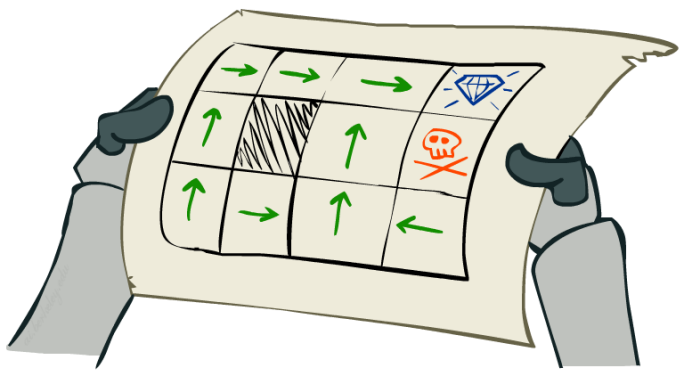
- MDP: States, actions
- **Environment: Transitions and rewards**
- Agent: Policy over actions
- Policy iteration
  - Policy evaluation
  - Policy improvement
- Value Iteration
- Model free policy evaluation
- Model free policy control



Today's takeaways

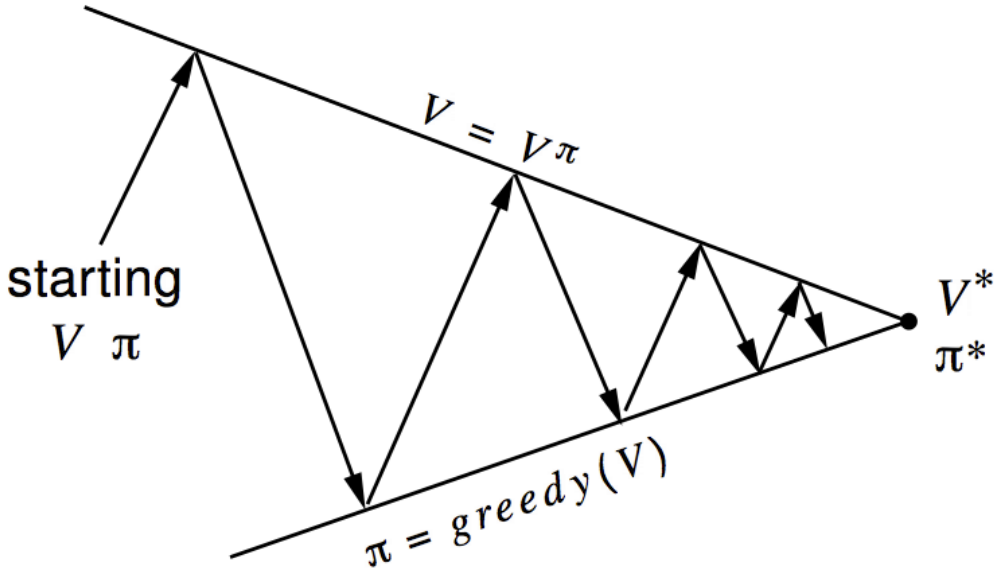
- MDP: States, actions
- Environment: Transitions and rewards
- **Agent: Policy over actions**
- Policy iteration
    - Policy evaluation
    - Policy improvement
- Value Iteration
- Model free policy evaluation
- Model free policy control

$$\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$$



Today's takeaways

- MDP: States, actions
- Environment: Transitions and rewards
- Agent: Policy over actions
- **Policy iteration**
  - **Policy evaluation**
  - Policy improvement
- Value Iteration
- Model free policy evaluation
- Model free policy control

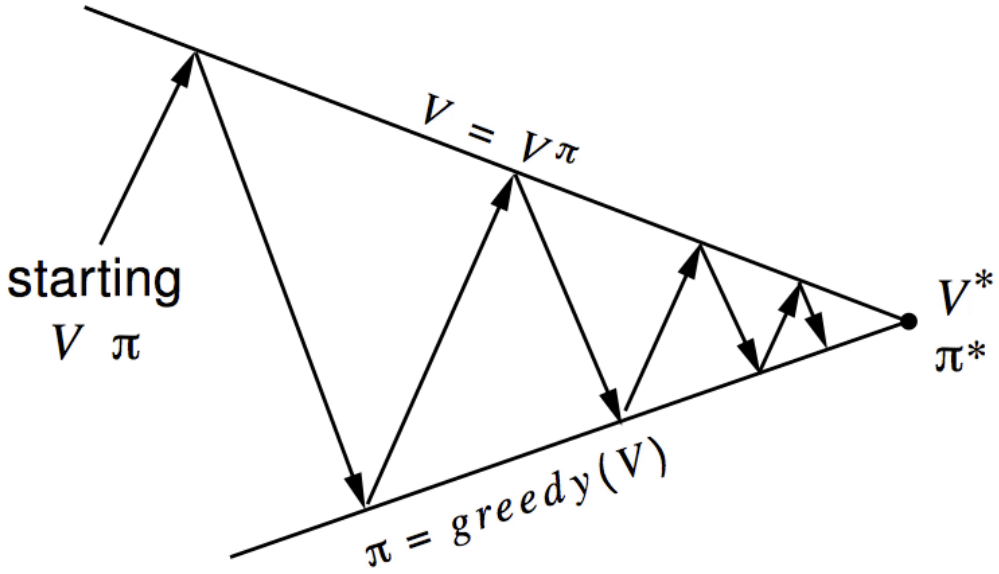


v* in terms of other *v

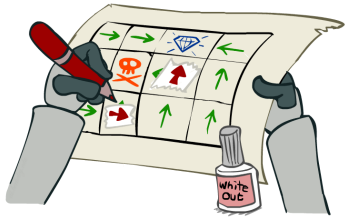
q* in terms of other *q

Today's takeaways

- MDP: States, actions
- Environment: Transitions and rewards
- Agent: Policy over actions
- **Policy iteration**
  - Policy evaluation
  - **Policy improvement**
- Value Iteration
- Model free policy evaluation
- Model free policy control

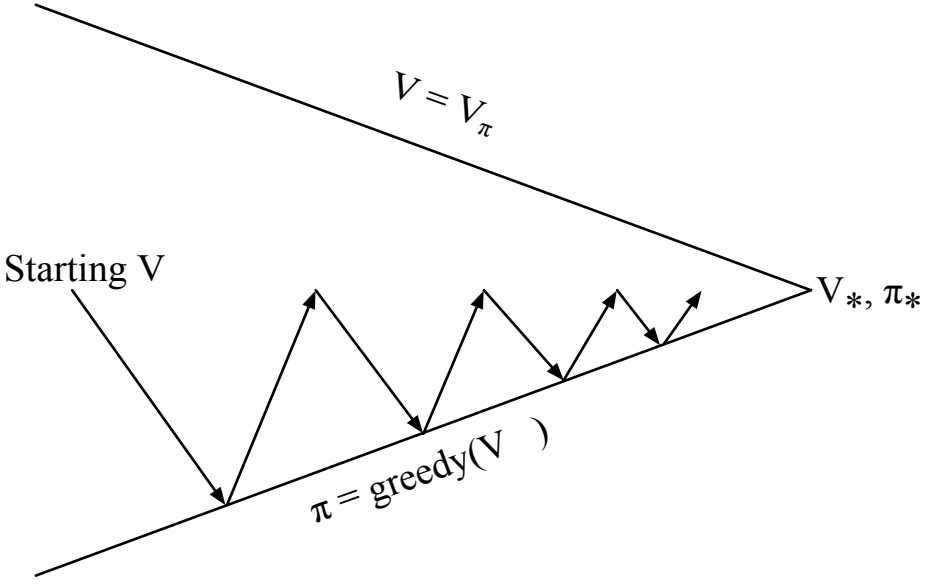


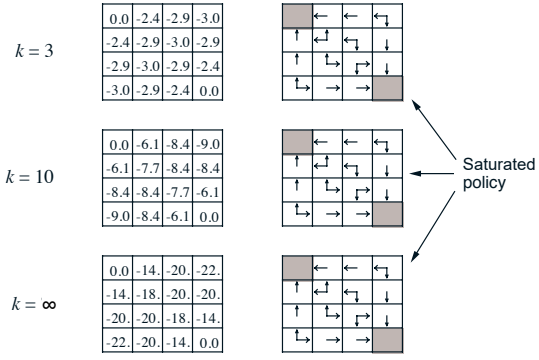
$$\pi_{\text{new}}(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\text{argmax}}\, q_{\text{old}}(s,a) \\ 0 & \textit{otherwise} \end{cases}$$



Today's takeaways

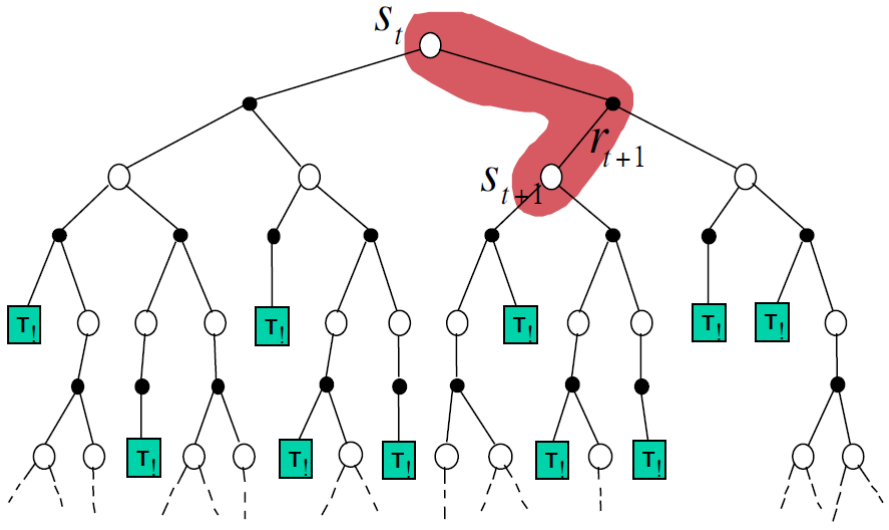
- MDP: States, actions
- Environment: Transitions and rewards
- Agent: Policy over actions
- Policy iteration
  - Policy evaluation
  - Policy improvement
- **Value Iteration**
- Model free policy evaluation
- Model free policy control





Today's takeaways

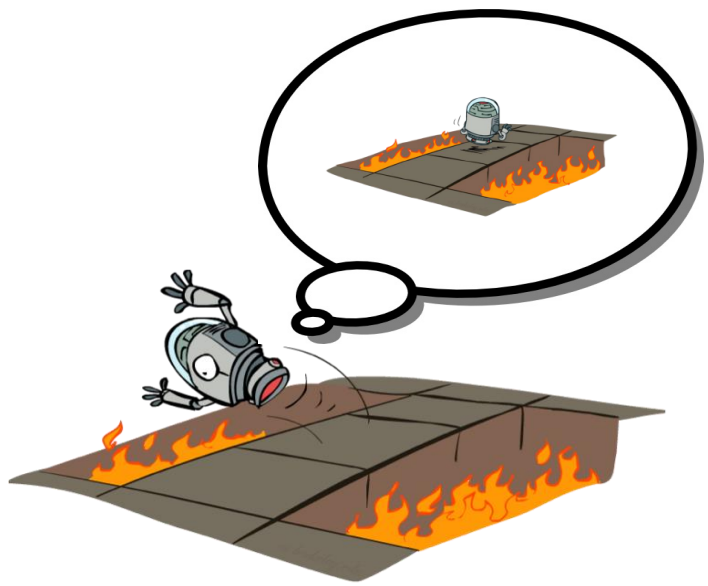
- MDP: States, actions
- Environment: Transitions and rewards
- Agent: Policy over actions
- Policy iteration
    - Policy evaluation
    - Policy improvement
- Value Iteration
- **Model free policy evaluation**
- Model free policy control



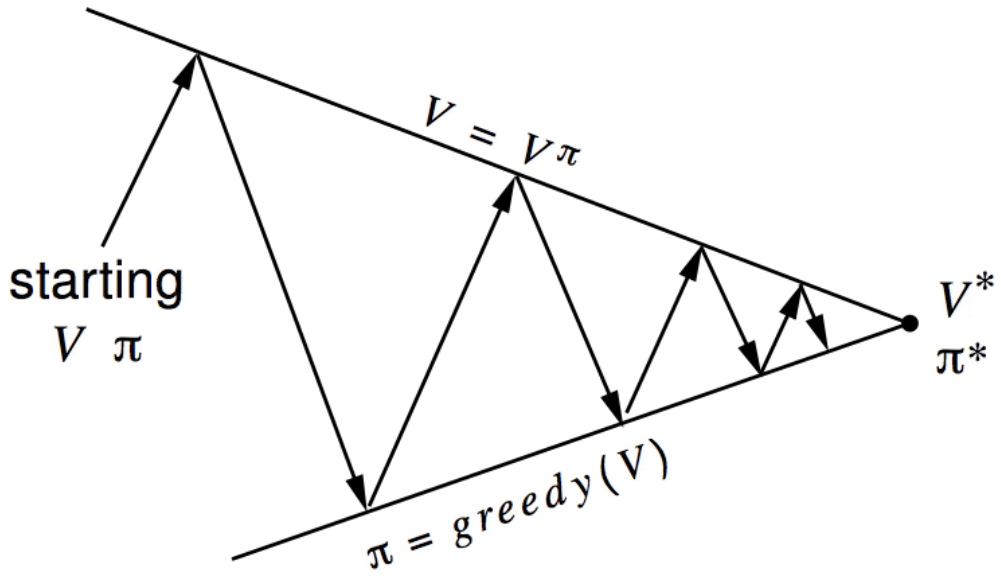
$$V(S_t) := V(S_t) + \alpha\,(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

Today's takeaways

- MDP: States, actions
- Environment: Transitions and rewards
- Agent: Policy over actions
- Policy iteration
    - Policy evaluation
    - Policy improvement
- Value Iteration
- Model free policy evaluation
- **Model free policy control**



Generalised Policy Iteration (Refresher)



Policy evaluation Estimate v_π

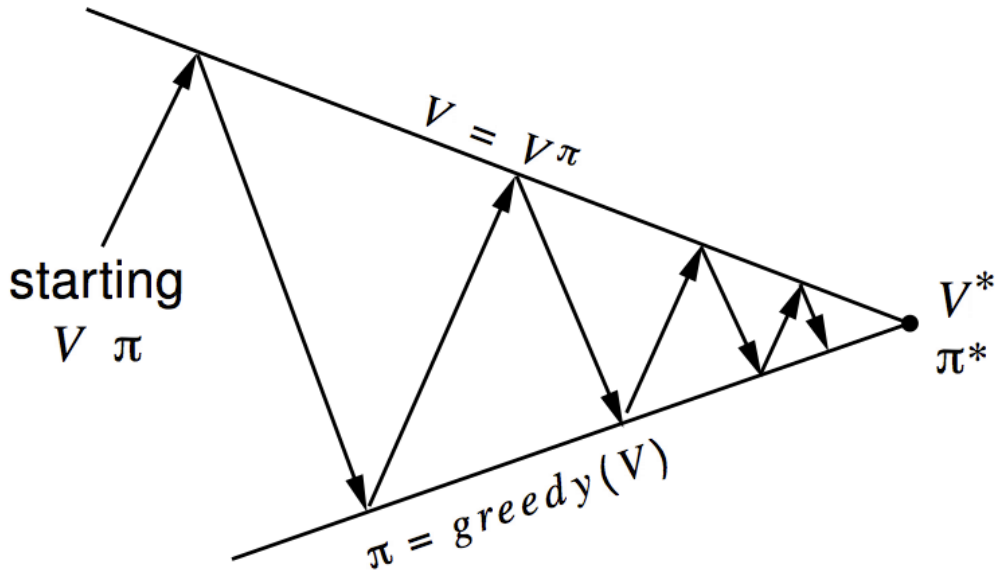
Model-based: Iterative policy evaluation

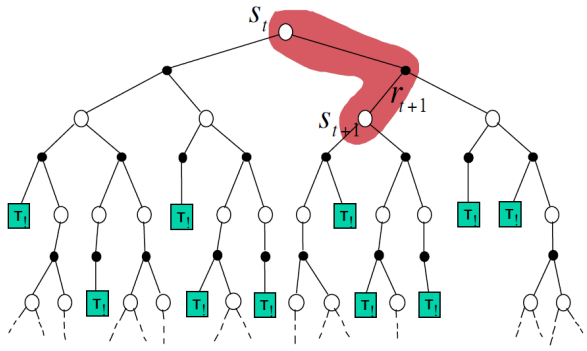
Policy improvement Generate $\pi' \geq \pi$

Model-based Greedy policy improvement

$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s') \right)$$

Generalised Policy Iteration





Policy evaluation Estimate v_π
Model-free: TD Policy evaluation

Policy improvement Generate $\pi' \geq \pi$
Model-free: Greedy policy improvement

Model-Free Policy Improvement

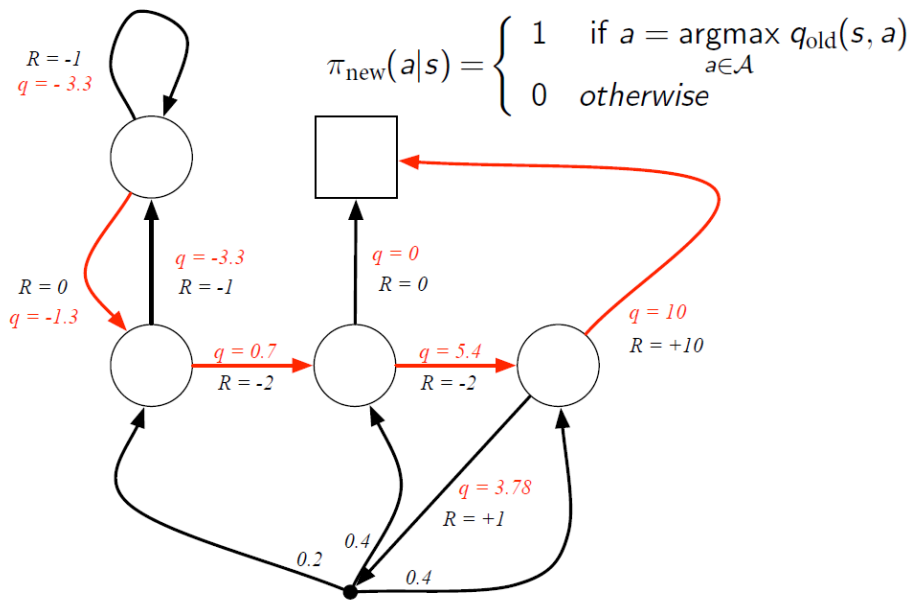
- Greedy policy improvement from V and Q values

$$\pi'(s) = \underset{a \in \mathcal{A}}{\text{argmax}}\ Q(s,a) \qquad \pi'(s) = \underset{a \in \mathcal{A}}{\text{argmax}}\ \mathcal{R}_s^a + \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V(s')$$

Model-Free Policy Improvement

- Greedy policy improvement from V and Q values

$$\pi'(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q(s,a) \qquad \pi'(s) = \operatorname*{argmax}_{a \in \mathcal{A}} \mathcal{R}_s^a + \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V(s')$$



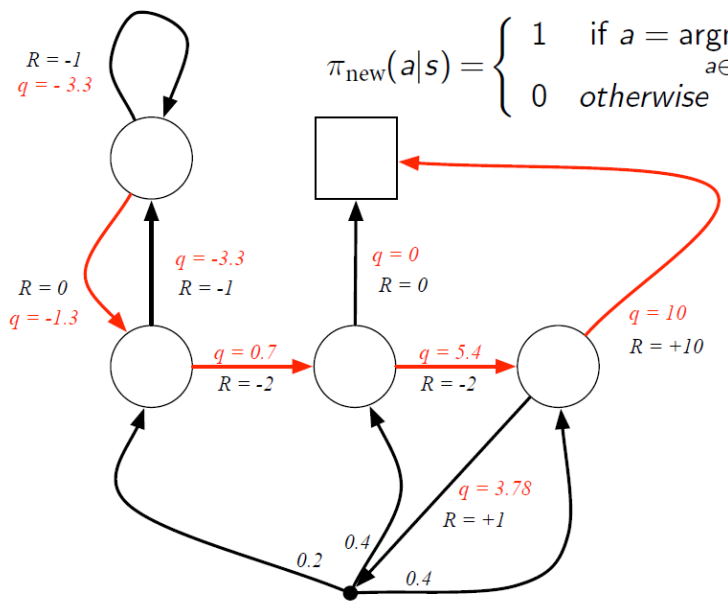
Model-Free Policy Improvement

- Greedy policy improvement over $V(s)$ requires model of MDP

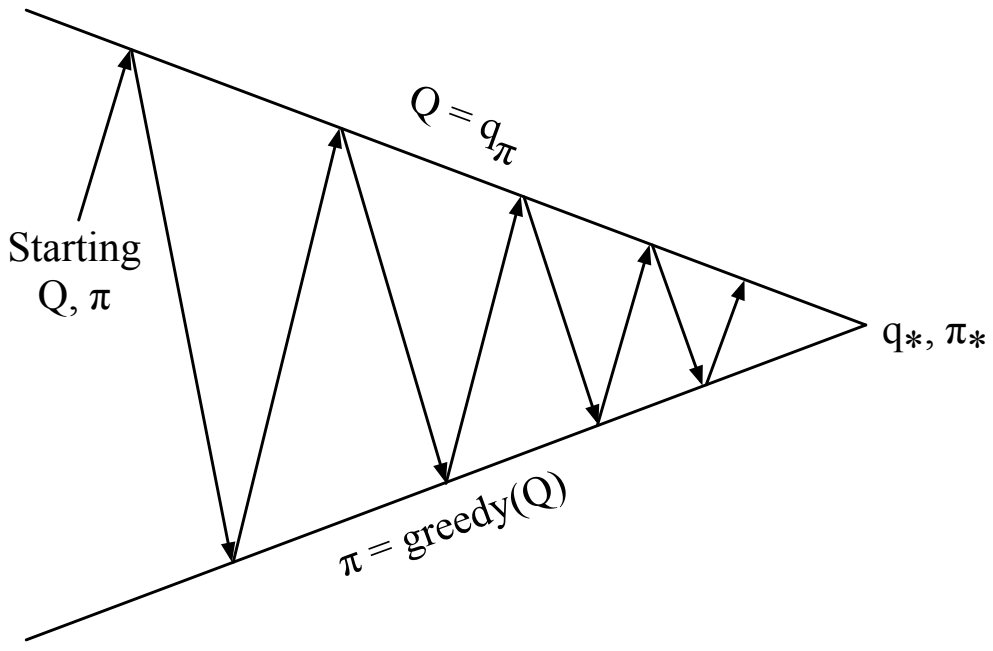
$$\pi'(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q(s,a)$$

$$\pi'(s) = \operatorname*{argmax}_{a \in \mathcal{A}} \mathcal{R}_s^a + \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V(s')$$

$$\pi_{\text{new}}(a|s) = \begin{cases} 1 & \text{if } a = \operatorname*{argmax}_{a \in \mathcal{A}} q_{\text{old}}(s,a) \\ 0 & \text{otherwise} \end{cases}$$



Generalised Policy Iteration with Q values

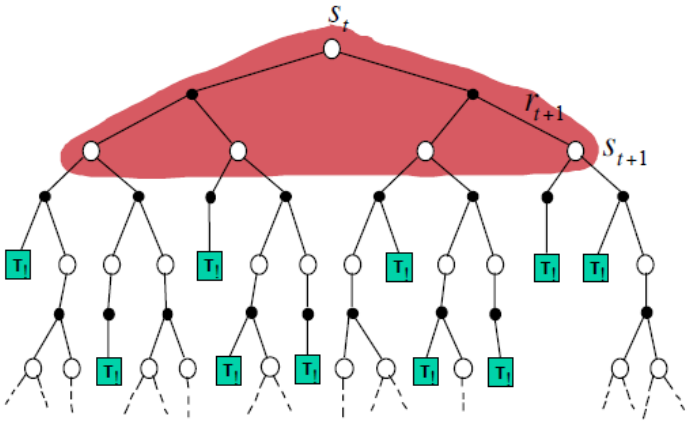


Policy evaluation TD policy evaluation, $Q = q_\pi$

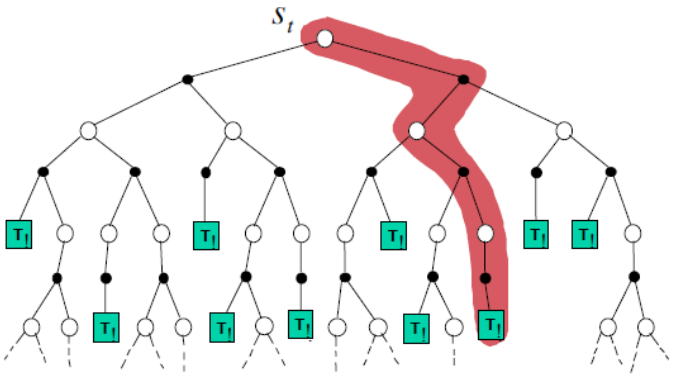
Policy improvement Greedy policy improvement?

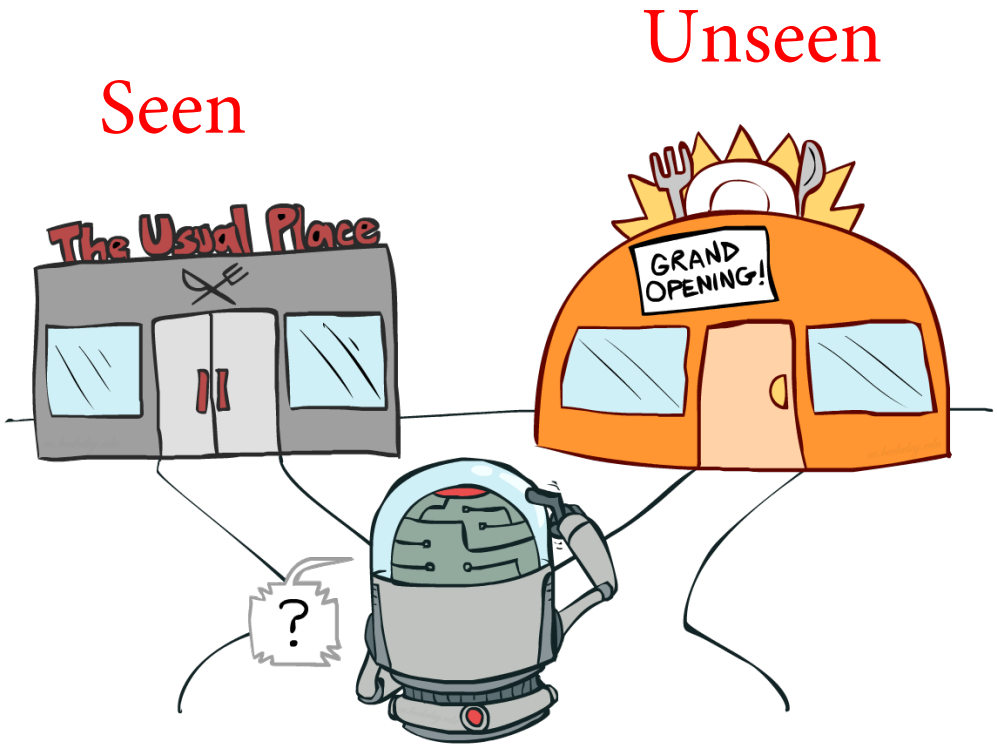
Thinking beyond Greedy - Exploration-Exploitation

What we hoped we had:



What we have:

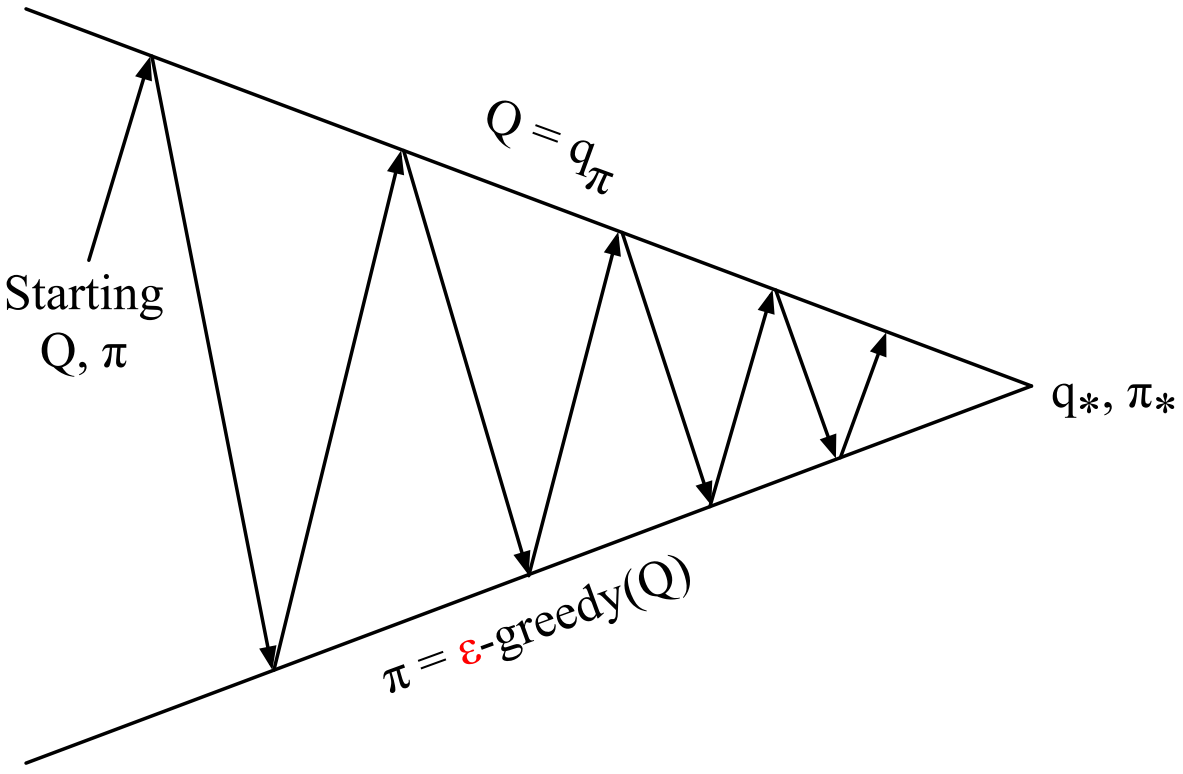




ϵ-Greedy Exploration

- Simplest idea for ensuring continual exploration
- With probability $1 - \epsilon$ choose the greedy action
- With probability ϵ choose an action at random

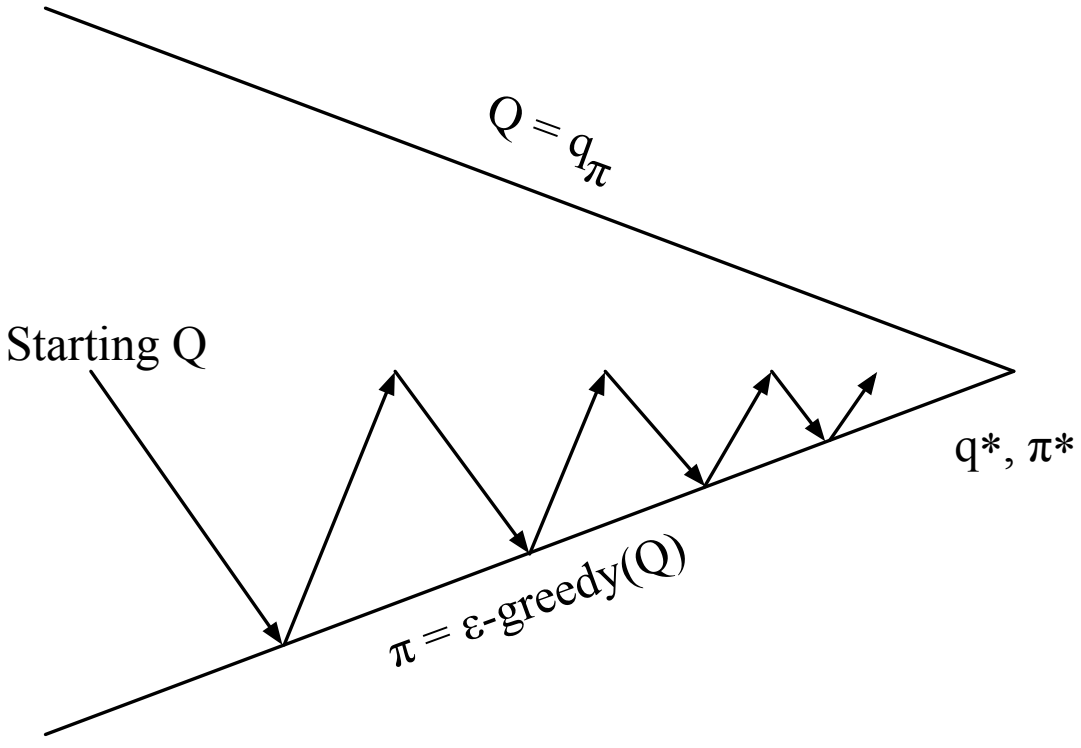
TD Policy Iteration

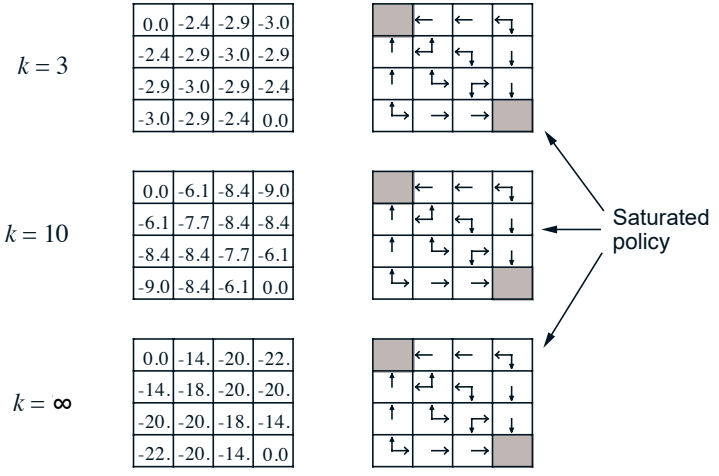


Policy evaluation TD policy evaluation, $Q = q_\pi$

Policy improvement ϵ-greedy policy improvement

SARSA: TD Value Iteration



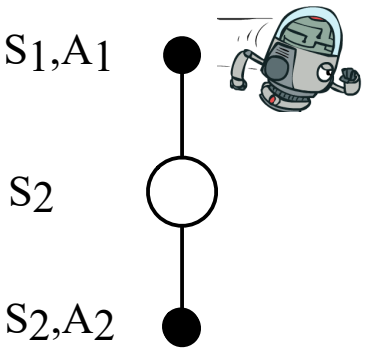


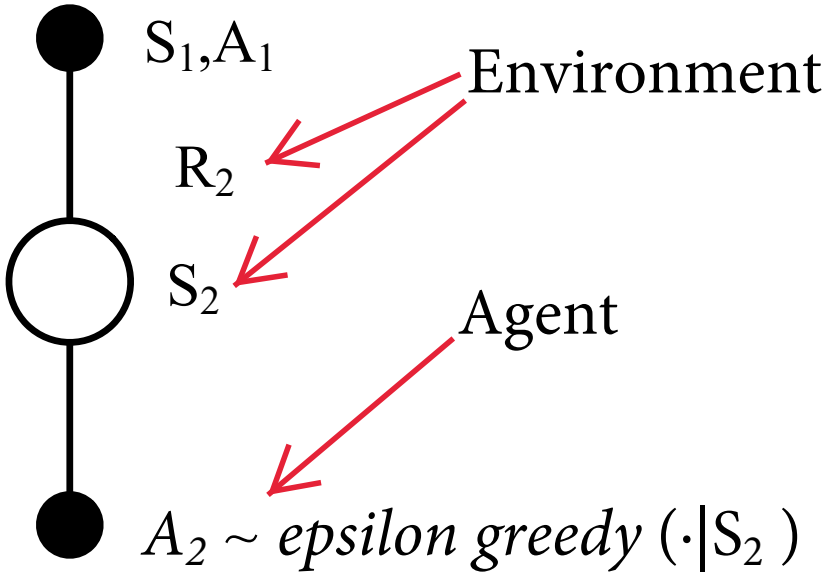
One step of evaluation:

Policy evaluation TD policy evaluation, $Q \approx q_\pi$

Policy improvement ϵ-greedy policy improvement

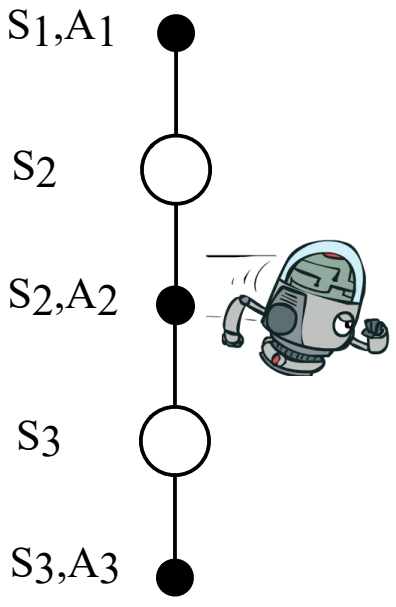
SARSA: Step by Step

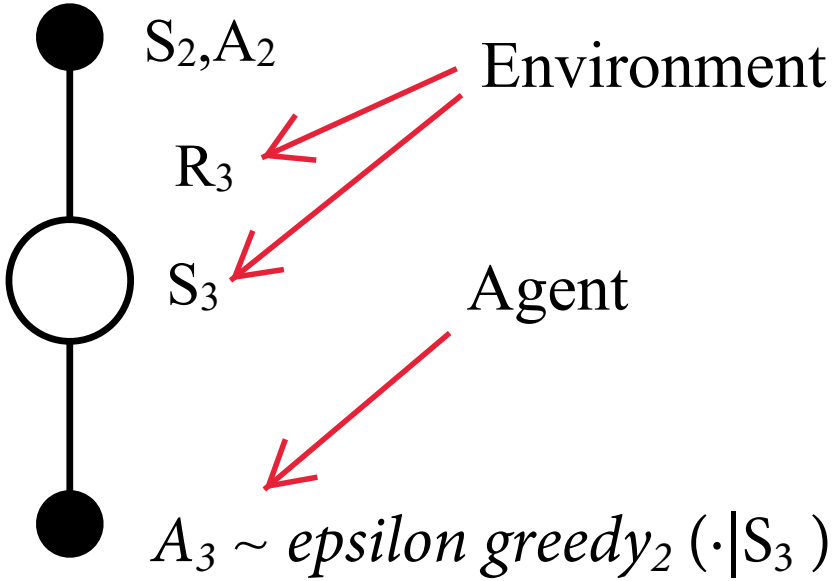




$$Q(S_1, A_1) := Q(S_1, A_1) + \alpha \left(R_2 + \gamma Q(S_2, A_2) - Q(S_1, A_1)\right)$$

SARSA: Step by Step





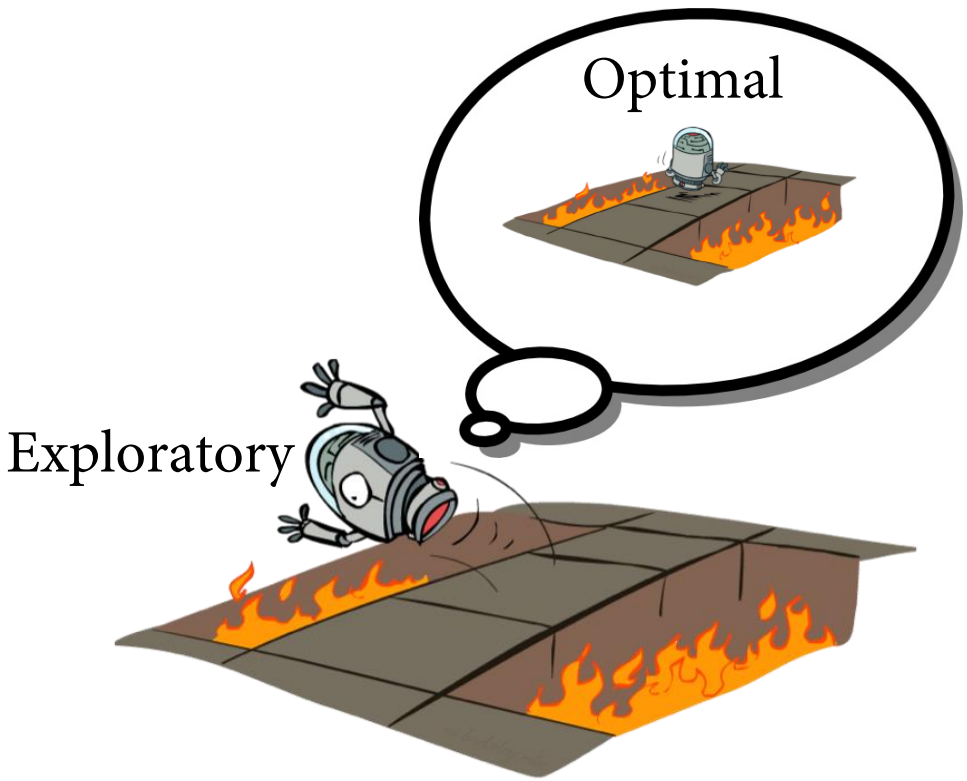
$$Q(S_2, A_2) := Q(S_2, A_2) + \alpha\big(R_3 + \gamma Q(S_3, A_3) - Q(S_2, A_2)\big)$$

Q Learning

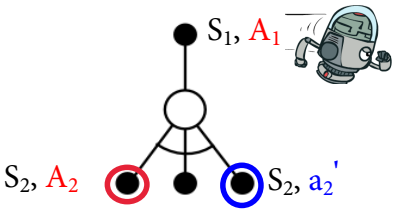
- Learn about optimal policy while following exploratory policy
- Target policy: Greedy [Optimal]
- Behaviour policy: Epsilon-greedy [Exploratory]

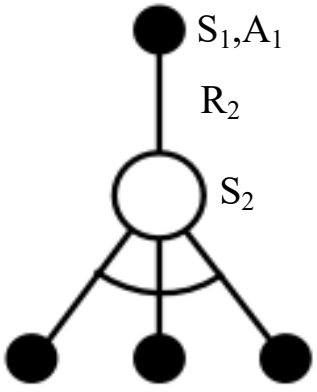
Q Learning

- Learn about optimal policy while following exploratory policy
- Target policy: Greedy [Optimal]
- Behaviour policy: Epsilon-greedy [Exploratory]



Q-Learning Control Algorithm



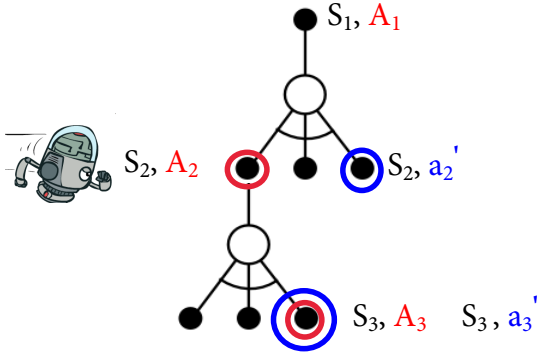


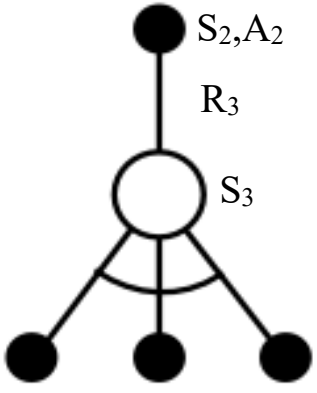
$A_2 \sim$ *epsilon greedy* $(\cdot|S_2)$

$$Q(S_1, A_1) := Q(S_1, A_1) + \alpha\left(R_2 + \gamma \max_{a_2'} Q(S_2, a_2') - Q(S_1, A_1)\right)$$

Sarsa : $Q(S_1, A_1) := Q(S_1, A_1) + \alpha\left(R_2 + \gamma Q(S_2, A_2) - Q(S_1, A_1)\right)$

Q-Learning Control Algorithm

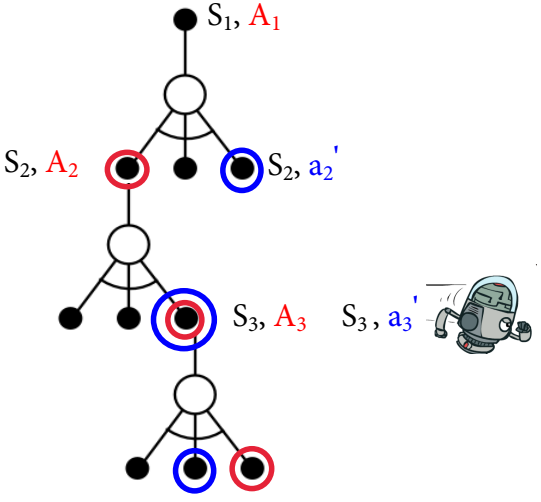


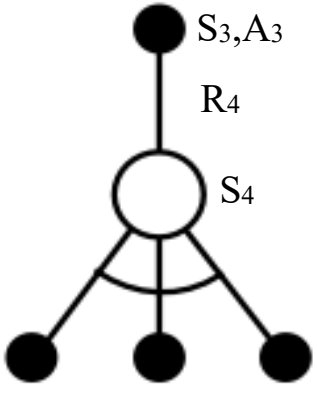


$A_3 \sim$ *epsilon greedy* $(\cdot|S_3)$

$$Q(S_2, A_2) := Q(S_2, A_2) + \alpha\left(R_3 + \gamma \max_{a_3'} Q(S_3, a_3') - Q(S_2, A_2)\right)$$

Q-Learning Control Algorithm

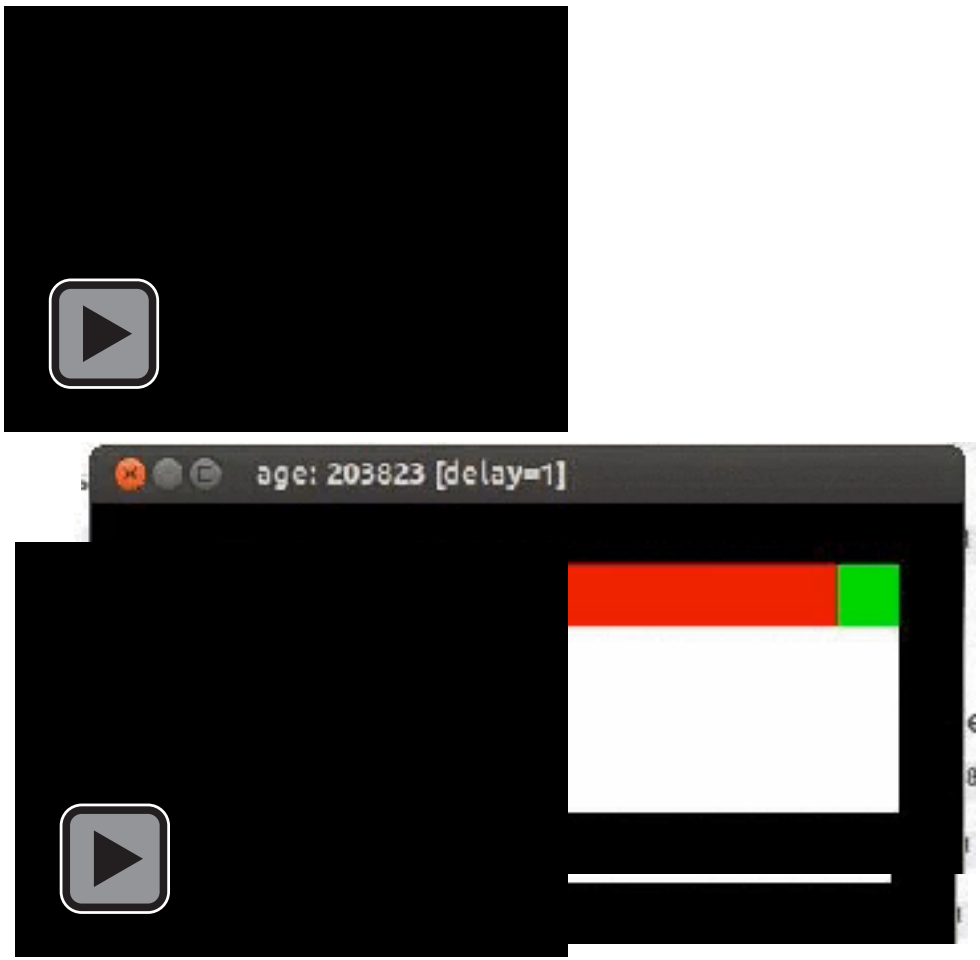




$A_4 \sim$ *epsilon greedy* $(\cdot|S_3)$

$$Q(S_3, A_3) := Q(S_3, A_3) + \alpha\left(R_4 + \gamma \max_{a_4'} Q(S_4, a_4') - Q(S_3, A_3)\right)$$

SARSA and Q-Learning example

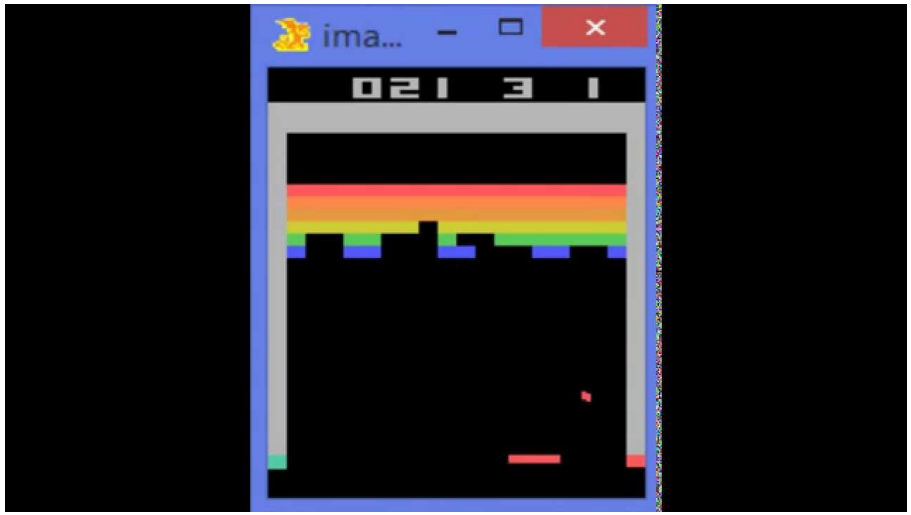


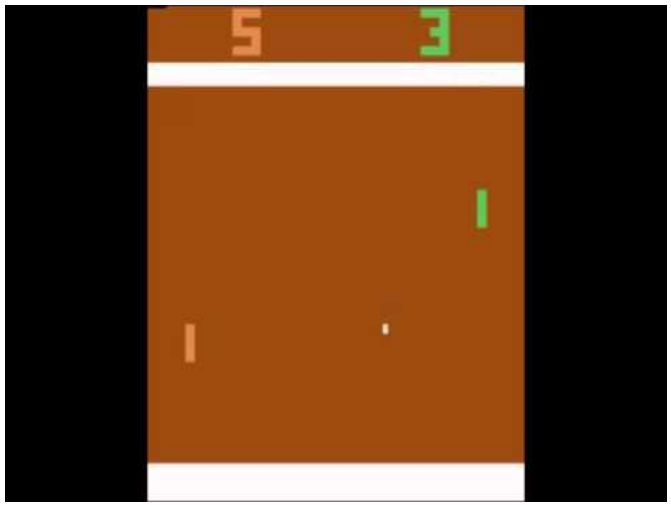
https://studywolf.wordpress.com/2013/07/01/reinforcement-learning-sarsa-vs-q-learning/

What's in store for Lec 13?

https://memegenerator.net/instance/73854727

What's in store for Lec 13?





https://www.youtube.com/watch?v=V1eYniJ0Rnk

https://www.youtube.com/watch?v=60pwnLB0DqY

Questions?

The only stupid question is the one you were afraid to ask but never did.
-Rich Sutton

References

- Introduction to RL by David Silver (UCL & DeepMind)

  www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html [Lec 1-5]

  https://youtu.be/2pWv7GOvuf0

- Artificial Intelligence by Peter Abbeel (UCB)

  https://edge.edx.org/courses/BerkeleyX/CS188x-SP15/SP15/20021a0a32d14a31b087db8d4bb582fd/

- Artificial Intelligence by Svetlana Lazebnik (UIUC)

  http://slazebni.cs.illinois.edu/fall16/

Appendix

Incremental Monte-Carlo Updates

- Update $V(s)$ incrementally after episode $S_1, A_1, R_2, ..., S_T$
- For each state S_t with return G_t

$$N(S_t) := N(S_t) + 1$$

$$V(S_t) := V(S_t) + \frac{1}{N(S_t)}(G_t - V(S_t))$$

Idea:

$$\begin{aligned}\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\ &= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\ &= \frac{1}{k}(x_k + (k-1)\mu_{k-1}) \\ &= \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})\end{aligned}$$

- In non-stationary problems, it can be useful to track a running mean, i.e. forget old episodes.

$$\begin{aligned}V(S_t) &:= (1\text{-}\alpha)\, V(S_t) + \alpha\, G_t \\ &:= V(S_t) + \boldsymbol{\alpha}\,(G_t - V(S_t))\end{aligned}$$

GLIE

Definition

Greedy in the Limit with Infinite Exploration (GLIE)

- All state-action pairs are explored infinitely many times,

$$\lim_{k \to \infty} N_k(s, a) = \infty$$

- The policy converges on a greedy policy,

$$\lim_{k \to \infty} \pi_k(a|s) = \mathbf{1}(a = \underset{a' \in \mathcal{A}}{\text{argmax}}\ Q_k(s, a'))$$

- For example, ϵ-greedy is GLIE if ϵ reduces to zero at $\epsilon_k = \frac{1}{k}$

Convergence of Sarsa

Theorem

Sarsa converges to the optimal action-value function, $Q(s,a) \rightarrow q_*(s,a)$*, under the following conditions:*

- *GLIE sequence of policies* $\pi_t(a|s)$
- *Robbins-Monro sequence of step-sizes* α_t

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$

$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

Monte-Carlo Control

- Sample kth episode using π: $\{S_1, A_1, R_2, ..., S_T\} \sim \pi$
- For each state S_t and action A_t in the episode,

$$N(S_t, A_t) := N(S_t, A_t) + 1$$

$$Q(S_t, A_t) := Q(S_t, A_t) + \frac{1}{N(S_t, A_t)} (G_t - Q(S_t, A_t))$$

- Improve policy based on new action-value function

$$\epsilon = 1/k$$

$$\pi = \epsilon\text{-greedy}(Q)$$

Theorem

Decaying epsilon Monte-Carlo control converges to the optimal action-value function, $Q(s, a) \to q_*(s, a)$

Sarsa Algorithm for On-Policy Control

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Q-Learning Algorithm for Off-Policy Control

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
        $S \leftarrow S'$;
    until S is terminal