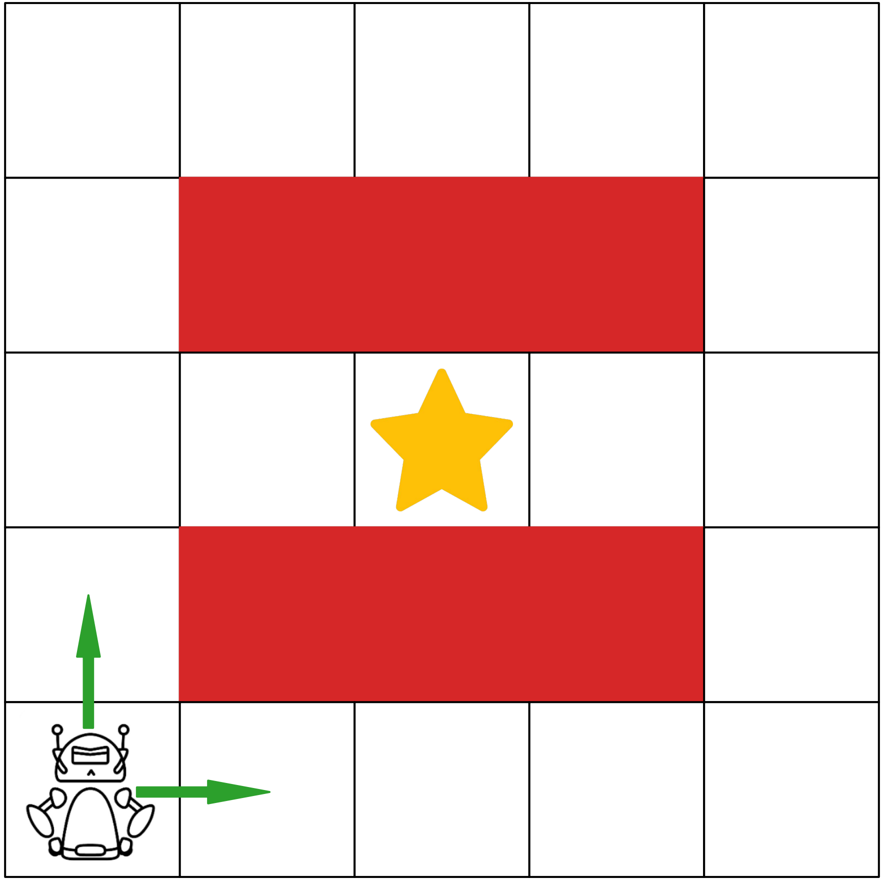
MDPs

Markov Decision Processes (MDPs)

- Another type of search problem! Components:
  - State space
  - Start state
  - Goal state
  - ~~Successor function~~ Transition function: $T(s, a, s')$
  - Reward function: $R(s, a, s')$
  - Discount factor: γ
  - Solution: find optimal policy



Value Iteration

An algorithm to iteratively (after each time step) determine the **optimal value** at each state.
Will always converge if $\gamma < 1$.

$$\mathbf{V} = \begin{pmatrix} v(s_0) \\ v(s_1) \\ \vdots \\ v(s_n) \end{pmatrix}$$

Value Iteration

An algorithm to iteratively (after each time step) determine the **optimal value** at each state.
Will always converge if $\gamma < 1$.

1. At time $t = 0$, initialize $V_0(s) = 0$ for all states
2. For every time $t > 0$, update using

$$V_{t+1}(s) = \max_{a} \sum_{s'} \underbrace{T(s, a, s')\,[R(s, a, s') + \gamma V_t(s')]}$$

... then take the maximum you can get, over all possible actions

For all possible successor states...

... get reward and value you can obtain from that state, times the probability you'll reach that state...

$$\mathbf{V} = \begin{bmatrix} v(s_0) \\ v(s_1) \\ \vdots \\ v(s_n) \end{bmatrix}$$

Q-value Iteration

Determine the optimal value for a **state-action** pair, so we can form a policy by picking the best action for each state.

$$\mathbf{Q} = \begin{pmatrix} Q(s_0, a_0) & Q(s_0, a_1) & \cdots & Q(s_0, a_{|A|}) \\ Q(s_1, a_0) & Q(s_1, a_1) & \cdots & Q(s_1, a_{|A|}) \\ \vdots & \vdots & \ddots & \vdots \\ Q(s_{|S|}, a_0) & Q(s_{|S|}, a_1) & \cdots & Q(s_{|S|}, a_{|A|}) \end{pmatrix}$$

Q-value Iteration

Determine the optimal value for a **state-action** pair, so we can form a policy by picking the best action for each state.

1. At time $t = 0$, initialize $Q_0(s, a) = 0$ for all state-action pairs
2. For every time $t > 0$, update using

$$Q_{t+1}(s, a) = \sum_{s'} \underbrace{T(s, a, s') \, [R(s, a, s') + \gamma \max_{a} Q_t(s', a')]}_{\text{... get the best Q-value you can obtain from that state, assuming you continue to act optimally from that state}}$$

For all possible successor states...

Policy Iteration

An algorithm to iteratively (after each time step) determine the **optimal policy** at each state.

1. At time $t = 0$, initialize $V_0^{\pi}(s) = 0$ for all states
2. For every time $t > 0$,

   a) Policy evaluation: calculate utility of current policy for each state

$$V^{\pi(t)}(s) = \sum_{s'} \underbrace{T(s, \pi_t(s), s')\ [R(s, \pi_t(s), s') + \gamma V^{\pi(t)}(s')]}_{\text{...get the utility of that state}}$$

For all possible successor states...

   b) Policy improvement: determine best action at each state

$$\pi_{t+1}(s) = \operatorname*{argmax}_{a} \sum_{s'} \underbrace{T(s, a, s')\ [R(s, a, s') + \gamma V^{\pi(t)}(s')]}_{\text{...get the utility of that state (for each action)...}}$$

... then take the action that gives you the highest utility

For all possible successor states...

Important Equations!!

- Optimal expected value of taking action *a* from state *s*

$$Q^*(s, a) = \sum_{s'} T(s, a, s') \, [R(s, a, s') + \gamma V^*(s')]$$

all states s' reachable from s

probability of reaching s' from s via action a

reward of reaching s from s' via action s

optimal value from reachable state s', discounted

- Optimal expected value at each state *s*

$$V^*(s) = \max_{a} \sum_{s'} T(s, a, s') \, [R(s, a, s') + \gamma V^*(s')]$$

- Optimal policy from state s

$$\pi^*(s) = \operatorname{argmax}_{a} \sum_{s'} T(s, a, s') \, [R(s, a, s') + \gamma V^*(s')]$$