REINFORCEMENT LEARNING

FUNDAMENTALS + APPLICATIONS

WEEK 3

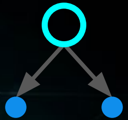
OPTIMALITY + DYNAMIC PROGRAMMING

BELLMAN EXPECTATION EQUATIONS

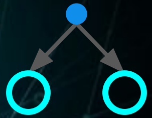
$$v(s) = \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$$

$$q(s,a) = \mathbb{E}[R_{t+1} + \gamma q(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$$

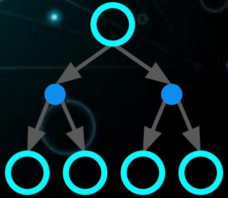
BELLMAN EXPECTATION EQUATIONS



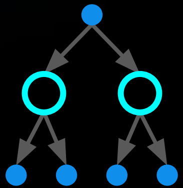
$$v_\pi(s) = \sum \pi(a|s) \cdot q_\pi(s,a)$$



$$q_\pi(s,a) = \sum p(s',r|s,a) \cdot [r + \gamma v_\pi(s')]$$



$$v_\pi(s) = \sum \pi(a|s) \cdot \sum p(s',r|s,a) \cdot [r + \gamma v_\pi(s')]$$



$$q_\pi(s,a) = \sum p(s'|s,a) \cdot [r_{s',a} + \gamma \sum \pi(a'|s') \cdot q_\pi(s',a')]$$

OPTIMAL VALUE FUNCTIONS

$$v_*(s) \equiv \max_{\pi} v_{\pi}(s)$$

$$q_*(s,a) \equiv \max_{\pi} q_{\pi}(s,a)$$

π_*

OPTIMAL POLICIES

$\pi' \geq \pi$ iff $v_{\pi'}(s) \geq v_\pi(s)\ \forall\ s$

For any Markov Decision Process,

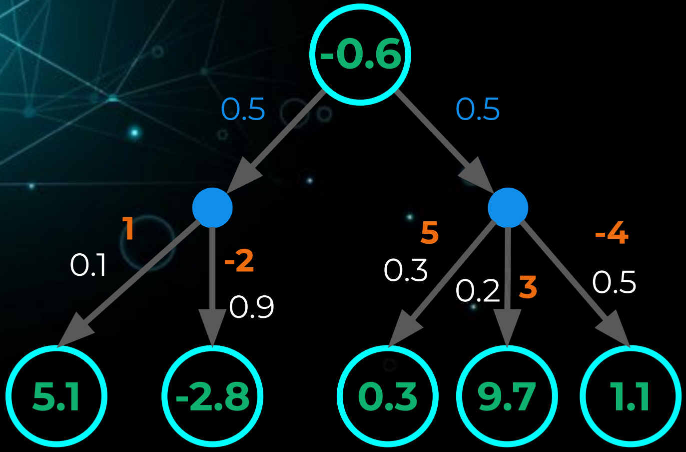
- There exists an optimal policy π_* that is better than or equal to all other policies, $\pi_* \geq \pi$.
- There is always a deterministic optimal policy $\pi_*(a|s) = \text{argmax}\, q_*(s,a)$
- There can be more than one optimal policy, but all optimal policies share the same optimal value functions

$$v_{\pi^*}(s) = v_*(s)$$
$$q_{\pi^*}(s,a) = q_*(s,a)$$

STATE-VALUE FUNCTION

Bellman Expectation Equation (Uniform Random Policy)

$$v_\pi(s) = \sum_a \pi(a|s) \cdot \sum p(s',r|s,a) \cdot [r + \gamma v_\pi(s')]$$



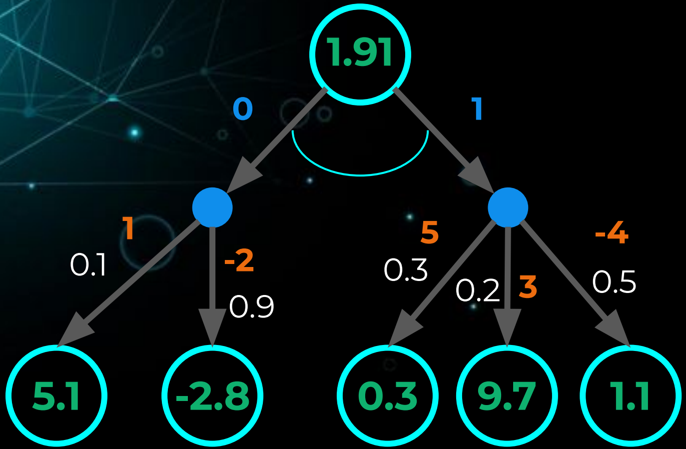
$$(0.5 \cdot 0.1 \cdot [1 + 0.7 \cdot 5.1])$$
$$+$$
$$(0.5 \cdot 0.9 \cdot [-2 + 0.7 \cdot -2.8])$$
$$+$$
$$(0.5 \cdot 0.3 \cdot [5 + 0.7 \cdot 0.3])$$
$$+$$
$$(0.5 \cdot 0.2 \cdot [3 + 0.7 \cdot 9.7])$$
$$+$$
$$(0.5 \cdot 0.5 \cdot [-4 + 0.7 \cdot 1.1])$$
$$=$$
$$\mathbf{-0.6}$$

STATE-VALUE FUNCTION

Bellman Optimality Equation

$$q_\pi(s,a) = \sum p(s',r|s,a) \cdot [r + \gamma v_\pi(s')]$$

$$v_*(s) = \max_a \sum p(s',r|s,a) \cdot [r + \gamma v_*(s')]$$



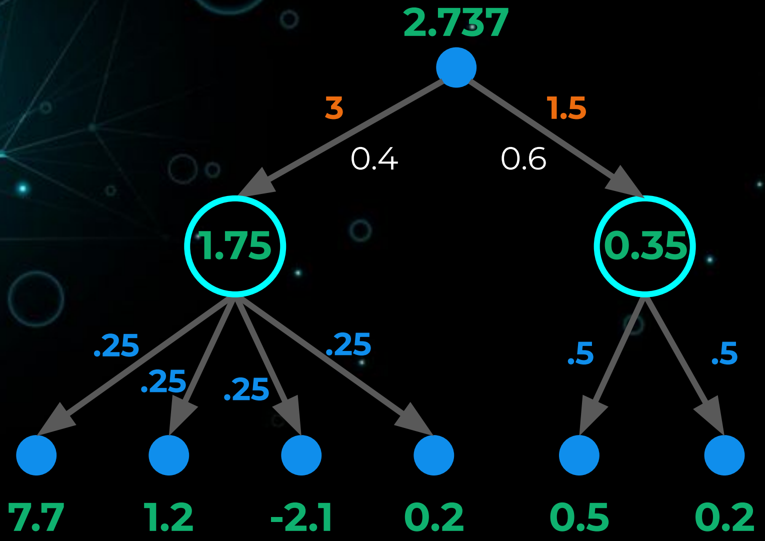
$$(1 \cdot 0.3 \cdot [5 + 0.7 \cdot 0.3])$$
$$+$$
$$(1 \cdot 0.2 \cdot [3 + 0.7 \cdot 9.7])$$
$$+$$
$$(1 \cdot 0.5 \cdot [-4 + 0.7 \cdot 1.1])$$
$$=$$

1.906

ACTION-VALUE FUNCTION

Bellman Expectation Equation (Uniform Random Policy)

$$q_{\pi}(s,a) = \sum p(s'|s,a) \cdot [r_{a,s'} + \gamma \sum \pi(a'|s') \cdot q_{\pi}(s',a')]$$

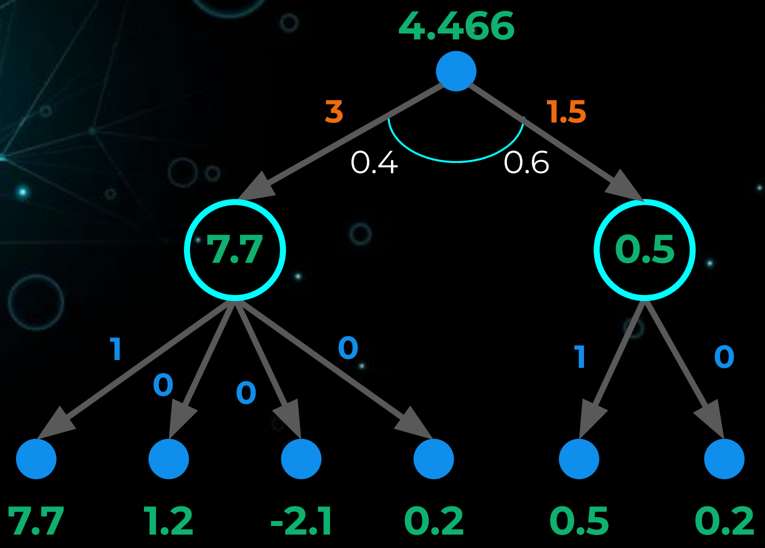


$$(0.4 \cdot [3 + 0.7 \cdot 1.75])$$
$$+$$
$$(0.6 \cdot [1.5 + 0.7 \cdot 0.35])$$
$$=$$
2.737

ACTION-VALUE FUNCTION

Bellman Optimality Equation

$$q_*(s,a) = \sum p(s'|s,a) \cdot [r_{a,s'} + \gamma \max_{a'} q_*(s',a')]$$

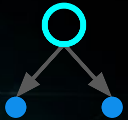


$$(0.4 \cdot [3 + 0.7 \cdot 7.7])$$
$$+$$
$$(0.6 \cdot [1.5 + 0.7 \cdot 0.5])$$
$$=$$
4.466

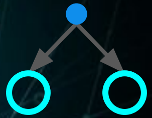
OPTIMALITY

Solving the Bellman Optimality Equation

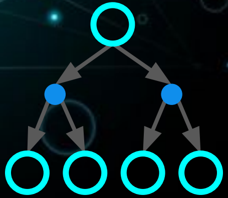
BELLMAN EXPECTATION EQUATIONS



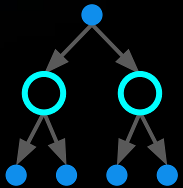
$$v_\pi(s) = \sum \pi(a|s) \cdot q_\pi(s,a)$$



$$q_\pi(s,a) = \sum p(s',r|s,a) \cdot [r + \gamma v_\pi(s')]$$

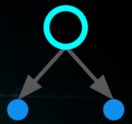


$$v_\pi(s) = \sum \pi(a|s) \cdot \sum p(s',r|s,a) \cdot [r + \gamma v_\pi(s')]$$

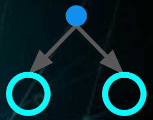


$$q_\pi(s,a) = \sum p(s'|s,a) \cdot [r_{s',a} + \gamma \sum \pi(a'|s') \cdot q_\pi(s',a')]$$

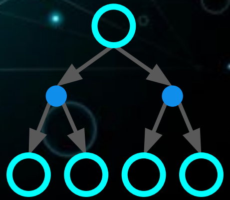
DYNAMIC PROGRAMMING



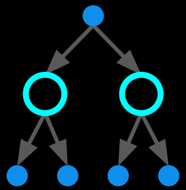
$$v_\pi(s) = \mathbf{max}\, q_\pi(s,a)$$



$$q_\pi(s,a) = \sum p(s',r|s,a) \cdot [r + \gamma v_\pi(s')]$$



$$v_\pi(s) = \mathbf{max} \sum p(s',r|s,a) \cdot [r + \gamma v_\pi(s')]$$



$$q_\pi(s,a) = \sum p(s'|s,a) \cdot [r_{s',a} + \gamma\, \mathbf{max}\, q_\pi(s',a')]$$

BELLMAN OPTIMALITY EQUATIONS

$$v_*(s) = \max \sum p(s',r|s,a) \cdot [r + \gamma v_*(s')]$$

$$q_*(s,a) = \sum p(s'|s,a) \cdot [r_{a,s'} + \gamma \max q_*(s',a')]$$

POLICY EVALUATION

(PREDICTION)

MDP + π *go in,*

v_π *comes out*

POLICY EVALUATION

Prediction

If I follow this (FIXED) policy in this environment, how much reward can I expect starting in each state?

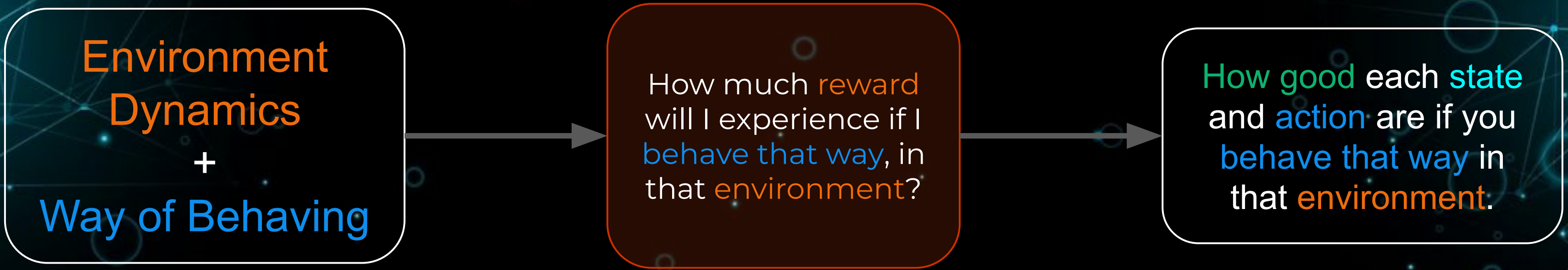
POLICY EVALUATION

Prediction

$$V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow \dots \rightarrow V_\pi$$

POLICY EVALUATION

Prediction



POLICY EVALUATION

Prediction

MDP

$\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{A} \rangle$

Policy π

$\rightarrow$ Policy Evaluation Algorithm $\rightarrow$

v_π

q_π

POLICY EVALUATION

The Algorithm : State-Value Function

Input: π, the MDP ($\mathscr{S}$, $\mathscr{P}$, $\mathcal{R}$, γ, $\mathscr{A}$), small positive number θ
Initialize: An array $\mathbf{V}_a(s) = 0 \ \forall \ s$, $\Delta = \theta+1$

---

while $\Delta > \theta$:
    $\Delta = 0$
    For s in $\mathscr{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum \pi(a|s) \cdot \sum p(s',r|s,a) \cdot [r + \gamma v_\pi(s')]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
return V

$$\boxed{\mathbf{v}_{k+1} = \mathcal{R} + \gamma \mathscr{P}_\pi \mathbf{v}_k}$$!!!

POLICY EVALUATION

The Algorithm : Action-Value Function

Input: π, the MDP $(\mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{A})$, small positive number θ
Initialize: An array $Q(s,a) = 0 \ \forall\ s, a$, $\Delta = \theta+1$

---

while $\Delta > \theta$:
    $\Delta = 0$
    For s in $\mathcal{S}$ and a in $\mathcal{A}$:
        $q \leftarrow Q(s,a)$
        $Q(s,a) \leftarrow \sum p(s',r|s,a) \cdot [r + \gamma \sum \pi(a'|s') \cdot Q(s',a')]$
        $\Delta \leftarrow \max(\Delta, |q - Q(s,a)|)$
return Q

POLICY IMPROVEMENT

(CONTROL)

MDP + π + V *go in,*

$\pi' \geq \pi$ *comes out*

POLICY IMPROVEMENT

Intuition

- We want our agent to learn and improve
- We need a better policy
- We need a way to turn our policy into a better policy

POLICY IMPROVEMENT

Control

Environment Dynamics
+
Value Function

→

WHAT HAPPENS IF WE'RE AS GREEDY AS POSSIBLE?!?!?!

→

How good each state and action *can possibly be* given the environment and its value function.

POLICY IMPROVEMENT

Control

MDP

$\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{A} \rangle$

Policy Improvement Algorithm

v_*

q_*

POLICY IMPROVEMENT

The Algorithm : State-Value Function

POLICY_STABLE = TRUE

---

for s in $\mathscr{S}$:
    old_action = $\pi(s)$
    $\pi(s) \leftarrow \text{argmax}_a \sum p(s',r|s,a) \cdot [r + \gamma V(s')]$
    if old_action != $\pi(s)$:
        POLICY_STABLE = FALSE
if POLICY_STABLE:
    return V, π
else:
    policy_evaluation(π)

POLICY IMPROVEMENT

The Algorithm : Action-Value Function

POLICY_STABLE = TRUE

---

for s in $\mathcal{S}$:
    old_action = $\pi(s)$
    $\pi(s) \leftarrow \operatorname{argmax}_a Q(s,a)$
    if old_action != $\pi(s)$:
        POLICY_STABLE = FALSE
if POLICY_STABLE:
    Return Q, π
else:
    policy_evaluation(π)

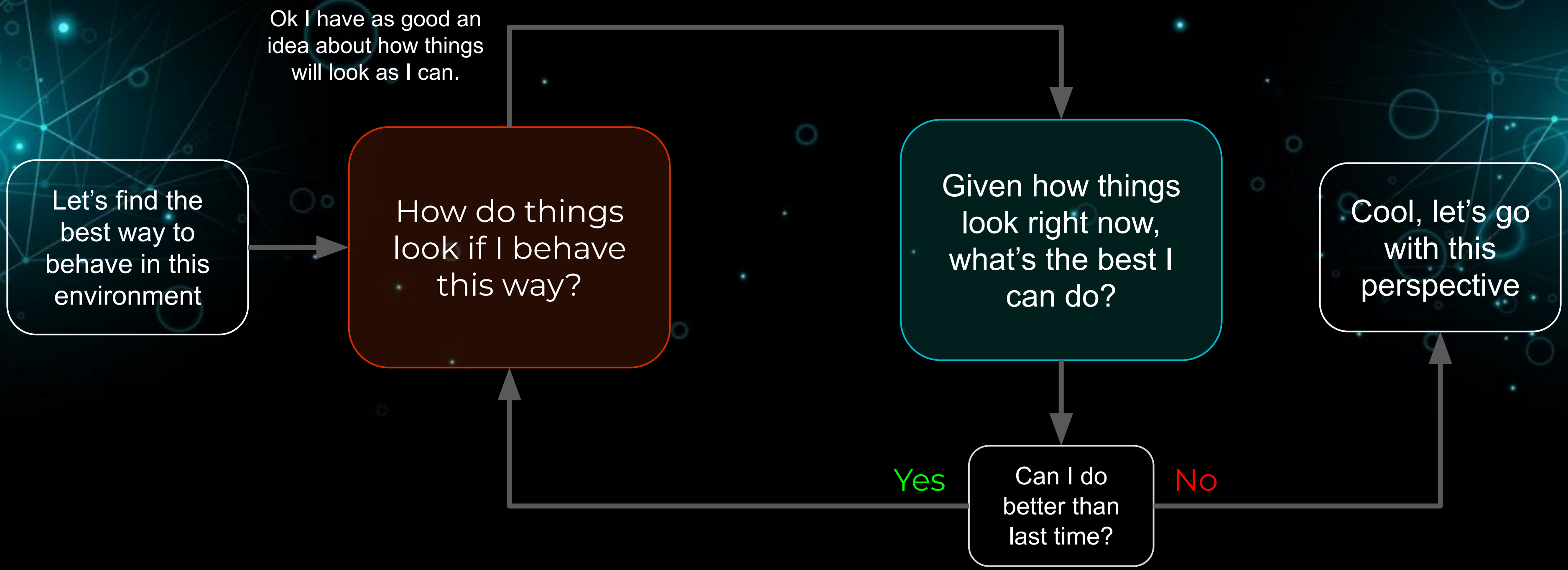
POLICY ITERATION

(PREDICTION + CONTROL)

MDP + π *go in,*
π_* *and* v_* *come out*

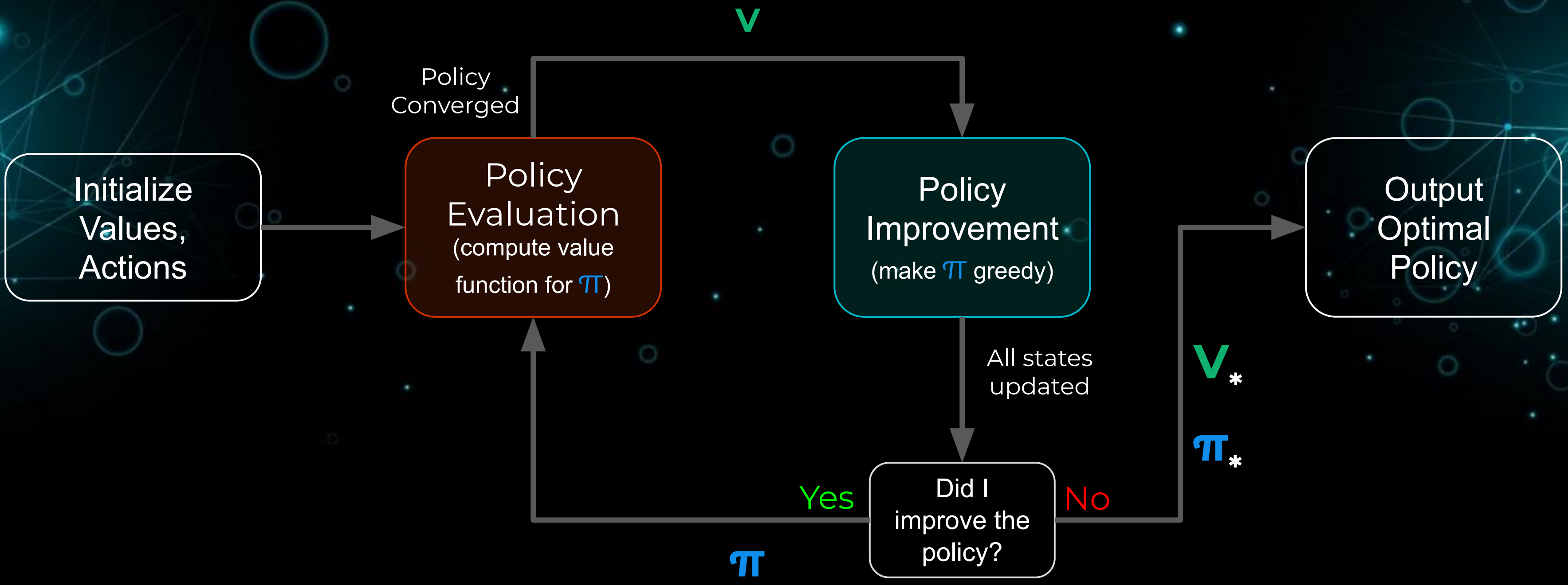
POLICY ITERATION

Generalized Policy Iteration



POLICY ITERATION

The Algorithm



VALUE ITERATION

Policy Iteration with 1 (one) Evaluation Sweep

MDP *goes in,*

π_* *comes out*

MAGIC!

VALUE ITERATION

The Algorithm

Input: The MDP ($\mathscr{S}$, $\mathscr{P}$, $\mathcal{R}$, γ, $\mathscr{A}$), small positive number θ
Initialize: An array $\mathbf{V}_a = 0\ \forall\ s$, $\Delta = \theta + 1$

---

while $\Delta > \theta$:

$\Delta = 0$

For s in $\mathscr{S}$:

$\mathbf{v} \leftarrow \mathbf{V}(s)$

$\mathbf{V}(s) \leftarrow \max \sum p(s', r|s, a) \cdot [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |\mathbf{v} - \mathbf{V}(s)|)$

set $\pi(s) \leftarrow \text{argmax} \sum p(s', r|s, a) \cdot [r + \gamma V(s')]$

Return $\pi \approx \pi_*$