# A Brief Introduction to Reinforcement Learning

**Jingwei Zhang**
**zhang@informatik.uni-freiburg.de**

# Outline

- Characteristics of Reinforcement Learning (RL)
- Components of RL (MDP, value, policy, Bellman)
- Planning (policy iteration, value iteration)
- Model-free Prediction (MC, TD)
- Model-free Control (Q-Learning)
- Deep Reinforcement Learning (DQN)

# Outline

- Characteristics of Reinforcement Learning (RL)
- The RL Problem (MDP, value, policy, Bellman)
- Planning (policy iteration, value iteration)
- Model-free Prediction (MC, TD)
- Model-free Control (Q-Learning)
- Deep Reinforcement Learning (DQN)

# Characteristics of RL
## *SL VS RL*

- Supervised Learning
  - i.i.d data
  - direct and strong supervision (label: what is the right thing to do)
  - instantaneous feedback
- Reinforcement Learning
  - sequential data, non-i.i.d
  - no supervisor, only a reward signal (rule: what you did is good or bad)
  - delayed feedback

# Characteristics of RL

# *SL VS RL*

- Supervised Learning
  - i.i.d data
  - direct and strong supervision (label: what is the right thing to do)
  - instantaneous feedback
- Reinforcement Learning
  - sequential data, non-i.i.d
  - no supervisor, only a reward signal (rule: what you did is good or bad)
  - delayed feedback

# Characteristics of RL

# *SL VS RL*

- Supervised Learning
  - i.i.d data
  - direct and strong supervision (label: what is the right thing to do)
  - instantaneous feedback
- Reinforcement Learning
  - sequential data, non-i.i.d
  - no supervisor, only a reward signal (rule: what you did is good or bad)
  - delayed feedback

# Characteristics of RL

# *SL VS RL*

- Supervised Learning
  - i.i.d data
  - direct and strong supervision (label: what is the right thing to do)
  - instantaneous feedback
- Reinforcement Learning
  - sequential data, non-i.i.d
  - no supervisor, only a reward signal (rule: what you did is good or bad)
  - delayed feedback

# Outline

- Characteristics of Reinforcement Learning (RL)
- Components of RL (MDP, value, policy, Bellman)
- Planning (policy iteration, value iteration)
- Model-free Prediction (MC, TD)
- Model-free Control (Q-Learning)
- Deep Reinforcement Learning (DQN)

# Components of RL

## *MDP*

- A general framework for sequential decision making
- A MDP is a tuple: $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

$\mathcal{S}$ :states

$\mathcal{A}$ :actions

$\mathcal{P}$ :transition probability, $\mathcal{P}^{\mathbf{a}}_{\mathbf{ss'}} = \mathbb{P}\left[\mathbf{S}_{t+1} = \mathbf{s}' | \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}\right]$

$\mathcal{R}$ :reward function, $\mathcal{R}^{\mathbf{a}}_{\mathbf{s}} = \mathbb{E}\left[\mathbf{R}_{t+1} | \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}\right]$

$\gamma$ :discount factor, $\gamma \in [0, 1]$

- Markov property:

"The future is independent of the past given the present"

# Components of RL

## *MDP*

- A general framework for sequential decision making
- A MDP is a tuple: $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

$$\mathcal{S} \text{ :states}$$

$$\mathcal{A} \text{ :actions}$$

$$\mathcal{P} \text{ :transition probability}, \mathcal{P}^{\mathbf{a}}_{\mathbf{ss}'} = \mathbb{P}\left[\mathbf{S}_{t+1} = \mathbf{s}' | \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}\right]$$

$$\mathcal{R} \text{ :reward function}, \mathcal{R}^{\mathbf{a}}_{\mathbf{s}} = \mathbb{E}\left[\mathbf{R}_{t+1} | \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}\right]$$

$$\gamma \text{ :discount factor}, \gamma \in [0, 1]$$

- Markov property:

"The future is independent of the past given the present"

# Components of RL

## *MDP*

- A general framework for sequential decision making
- A MDP is a tuple: $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

$$\mathcal{S} \text{ :states}$$

$$\mathcal{A} \text{ :actions}$$

$$\mathcal{P} \text{ :transition probability}, \mathcal{P}^{\mathbf{a}}_{\mathbf{ss}'} = \mathbb{P}\left[\mathbf{S}_{t+1} = \mathbf{s}' | \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}\right]$$

$$\mathcal{R} \text{ :reward function}, \mathcal{R}^{\mathbf{a}}_{\mathbf{s}} = \mathbb{E}\left[\mathbf{R}_{t+1} | \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}\right]$$

$$\gamma \text{ :discount factor}, \gamma \in [0, 1]$$

- Markov property:

"The future is independent of the past given the present"

# Components of RL
# *Policy & Return & Value*

- Policy:

$$\pi(\mathbf{a}|\mathbf{s}) = \mathbb{P}\left[\mathbf{A}_t = \mathbf{a}|\mathbf{S}_t = \mathbf{s}\right]$$

- Return:

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

- State-value function:

$$v_\pi(\mathbf{s}) = \mathbb{E}_\pi\left[G_t|\mathbf{S}_t = \mathbf{s}\right]$$

- Action-value function:

$$q_\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi\left[G_t|\mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}\right]$$

# Components of RL
## *Policy & Return & Value*

- Policy:

$$\pi(\mathbf{a}|\mathbf{s}) = \mathbb{P}\left[\mathbf{A}_t = \mathbf{a}|\mathbf{S}_t = \mathbf{s}\right]$$

- Return:

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

- State-value function:

$$v_\pi(\mathbf{s}) = \mathbb{E}_\pi\left[G_t|\mathbf{S}_t = \mathbf{s}\right]$$

- Action-value function:

$$q_\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi\left[G_t|\mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}\right]$$

# Components of RL
## *Policy & Return & Value*

- Policy:

$$\pi(\mathbf{a}|\mathbf{s}) = \mathbb{P}\left[\mathbf{A}_t = \mathbf{a}|\mathbf{S}_t = \mathbf{s}\right]$$

- Return:

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

- State-value function:

$$v_\pi(\mathbf{s}) = \mathbb{E}_\pi\left[G_t|\mathbf{S}_t = \mathbf{s}\right]$$

- Action-value function:

$$q_\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi\left[G_t|\mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}\right]$$

# Components of RL
# *Policy & Return & Value*

- Policy:

$$\pi(\mathbf{a}|\mathbf{s}) = \mathbb{P}\left[\mathbf{A}_t = \mathbf{a}|\mathbf{S}_t = \mathbf{s}\right]$$

- Return:

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

- State-value function:

$$v_\pi(\mathbf{s}) = \mathbb{E}_\pi\left[G_t|\mathbf{S}_t = \mathbf{s}\right]$$

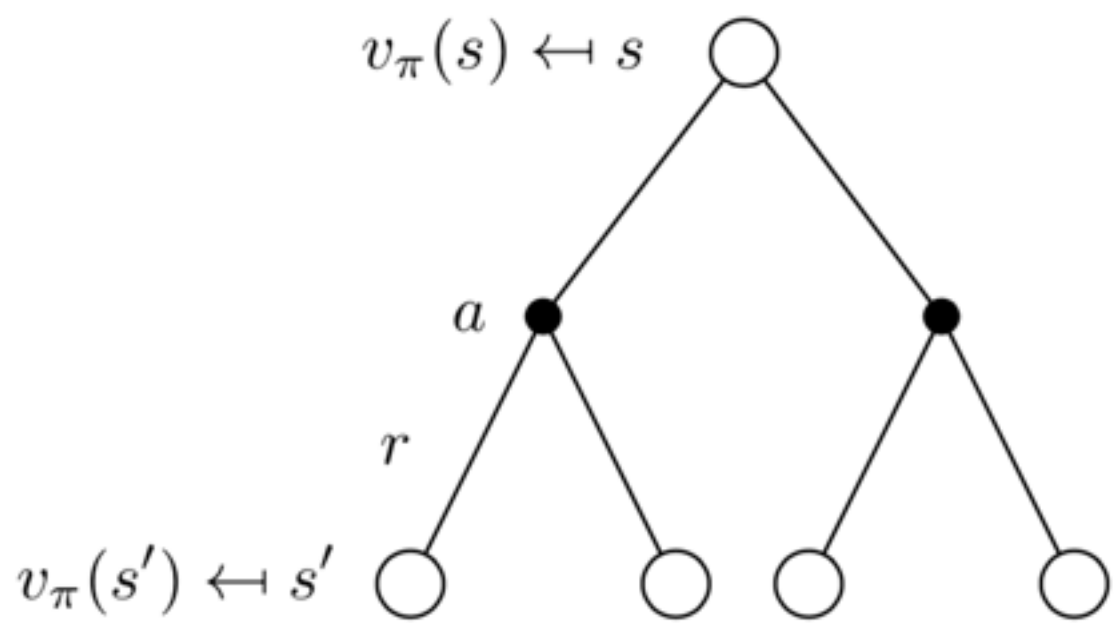
- Action-value function:

$$q_\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi\left[G_t|\mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}\right]$$

# Components of RL
# *Bellman Equations*

- Bellman Expectation Equation

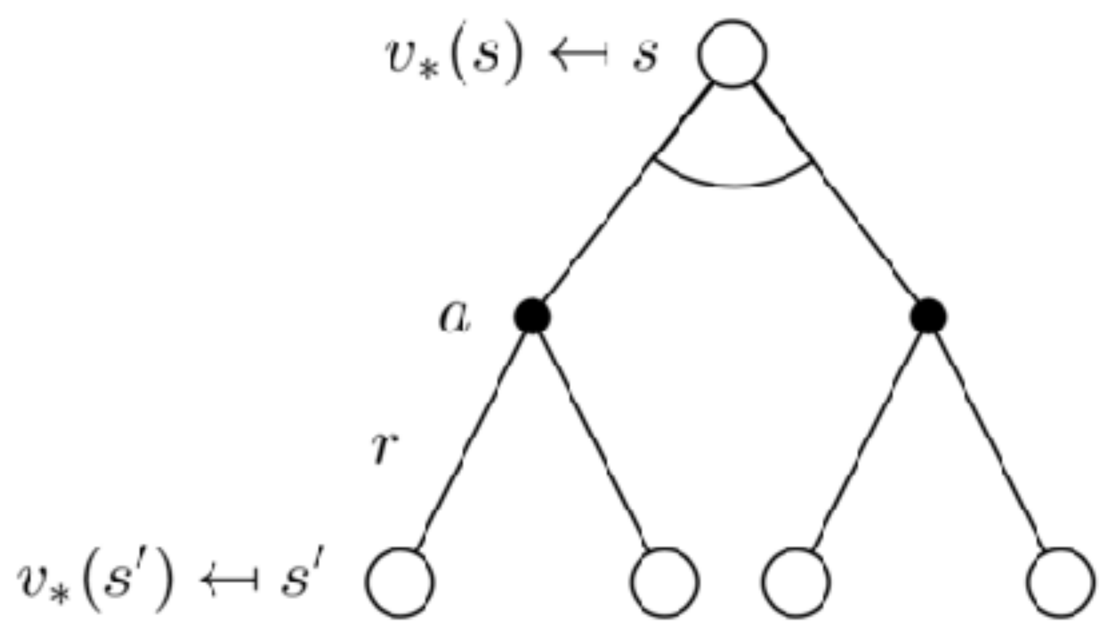
$$v_\pi(\mathbf{s}) = \mathbb{E}_\pi\left[\mathbf{R}_{t+1} + \gamma v_\pi(\mathbf{S}_{t+1}) | \mathbf{S}_t = \mathbf{s}\right]$$




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

# Components of RL
# *Bellman Equations*

- Bellman Optimality Equation




$$v_*(\mathbf{s}) = \max_{\mathbf{a}} \left( \mathcal{R}_{\mathbf{s}}^{\mathbf{a}} + \gamma \sum_{\mathbf{s}' \in \mathcal{S}} \mathcal{P}_{\mathbf{ss}'}^{\mathbf{a}} v_*(\mathbf{s}') \right)$$

# Components of RL
# *Prediction VS Control*

- Prediction

  given a policy, evaluate how much reward you can get by following that policy

- Control

  find an optimal policy that maximizes the cumulative future reward

# Components of RL
# *Prediction VS Control*

- Prediction

  given a policy, evaluate how much reward you can get by following that policy

- Control

  find an optimal policy that maximizes the cumulative future reward

# Components of RL
# *Planning VS Learning*

- Planning
  - the underlying MDP is known
  - agent only needs to perform computations on the given model
  - dynamic programming (policy iteration, value iteration)
- Learning
  - the underlying MDP is initially unknown
  - agent needs to interact with the environment
  - model-free (learn value / policy) / model-based (learn model, plan on it)

# Components of RL
## *Planning VS Learning*

- Planning
  - the underlying MDP is known
  - agent only needs to perform computations on the given model
  - dynamic programming (policy iteration, value iteration)
- Learning
  - the underlying MDP is initially unknown
  - agent needs to interact with the environment
  - model-free (learn value / policy) / model-based (learn model, plan on it)

# Components of RL
## *Planning VS Learning*

- Planning
  - the underlying MDP is known
  - agent only needs to perform computations on the given model
  - dynamic programming (policy iteration, value iteration)
- Learning
  - the underlying MDP is initially unknown
  - agent needs to interact with the environment
  - model-free (learn value / policy) / model-based (learn model, plan on it)

# Components of RL
# *Planning VS Learning*

- Planning
  - the underlying MDP is known
  - agent only needs to perform computations on the given model
  - dynamic programming (policy iteration, value iteration)
- Learning
  - the underlying MDP is initially unknown
  - agent needs to interact with the environment
  - model-free (learn value / policy) / model-based (learn model, plan on it)

# Outline

- Characteristics of Reinforcement Learning (RL)
- The RL Problem (MDP, value, policy, Bellman)
- Planning (policy iteration, value iteration)
- Model-free Prediction (MC, TD)
- Model-free Control (Q-Learning)
- Deep Reinforcement Learning (DQN)

# Planning
# *Dynamic Programming*

- Applied when optimal solutions can be decomposed into subproblems

- For prediction:
  - Input: $< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{S}, \gamma >, \pi$
  - Output: $v_\pi$
- For control:
  - Input: $< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{S}, \gamma >$
  - Output: $v_*, \pi_*$

# Planning
# *Dynamic Programming*

- Applied when optimal solutions can be decomposed into subproblems
- For prediction: (iterative policy evaluation)
  - Input: $< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{S}, \gamma >, \pi$
  - Output: $v_\pi$
- For control: (policy iteration, value iteration)
  - Input: $< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{S}, \gamma >$
  - Output: $v_*, \pi_*$

# Planning
# *Dynamic Programming*

- Applied when optimal solutions can be decomposed into subproblems

- For prediction: (iterative policy evaluation)

  - Input: $< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{S}, \gamma >, \pi$

  - Output: $v_\pi$

- For control: (policy iteration, value iteration)

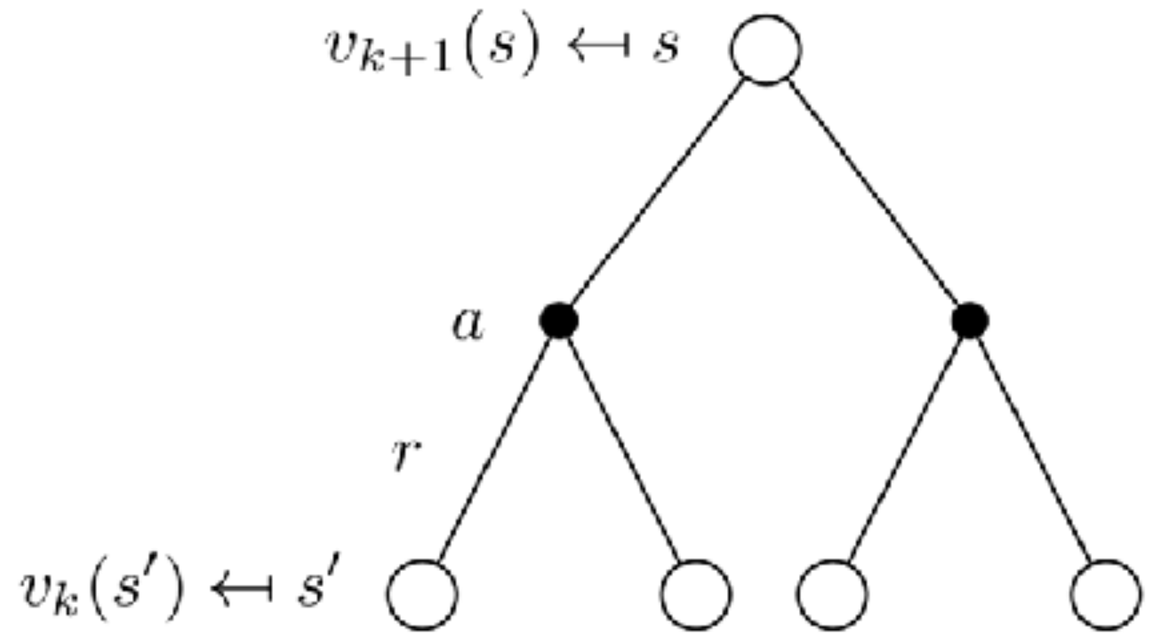
  - Input: $< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{S}, \gamma >$

  - Output: $v_*, \pi_*$

# Planning
## *Iterative Policy Evaluation*

- Iterative application of Bellman Expectation backup

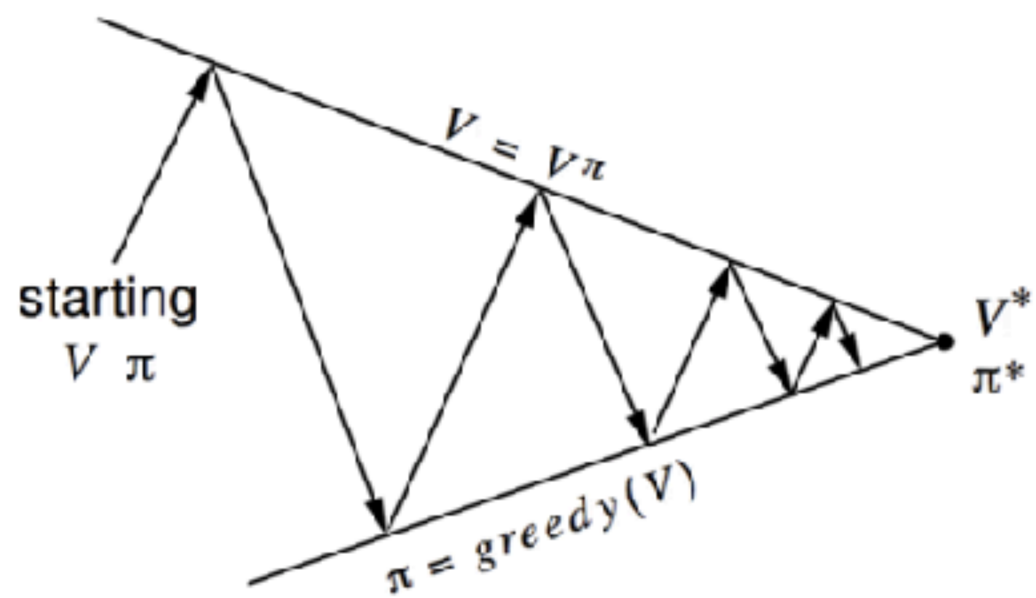
$$v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_\pi$$




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

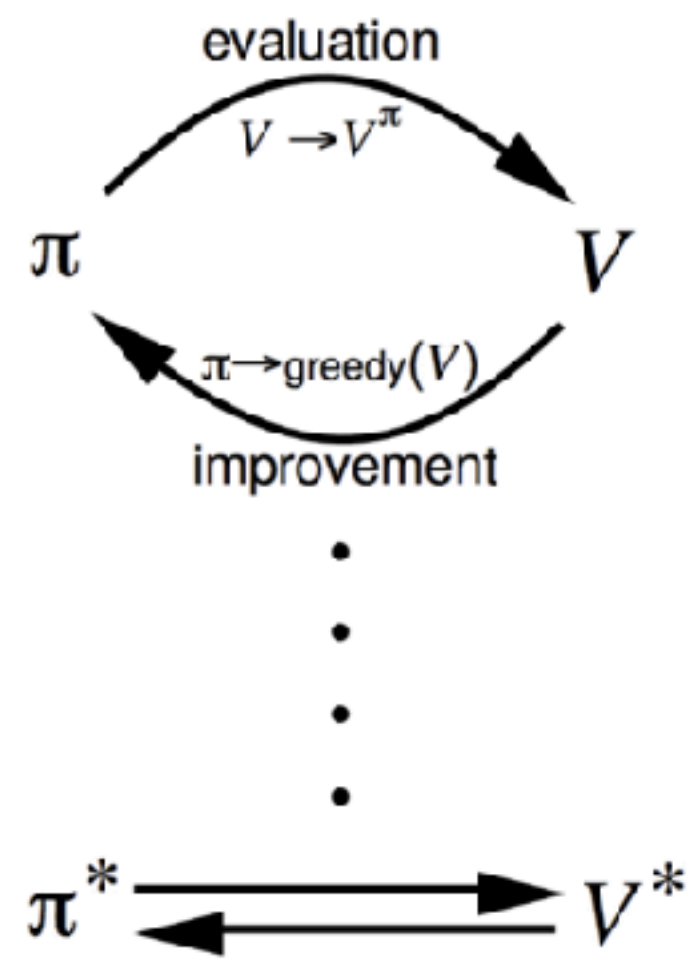
# Planning
## *Policy Iteration*

- Evaluate the given policy and get: $v_\pi$
- Get an improved policy by acting greedily: $\pi' = \mathrm{greedy}(v_\pi)$




Policy evaluation Estimate $v_\pi$
Iterative policy evaluation

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

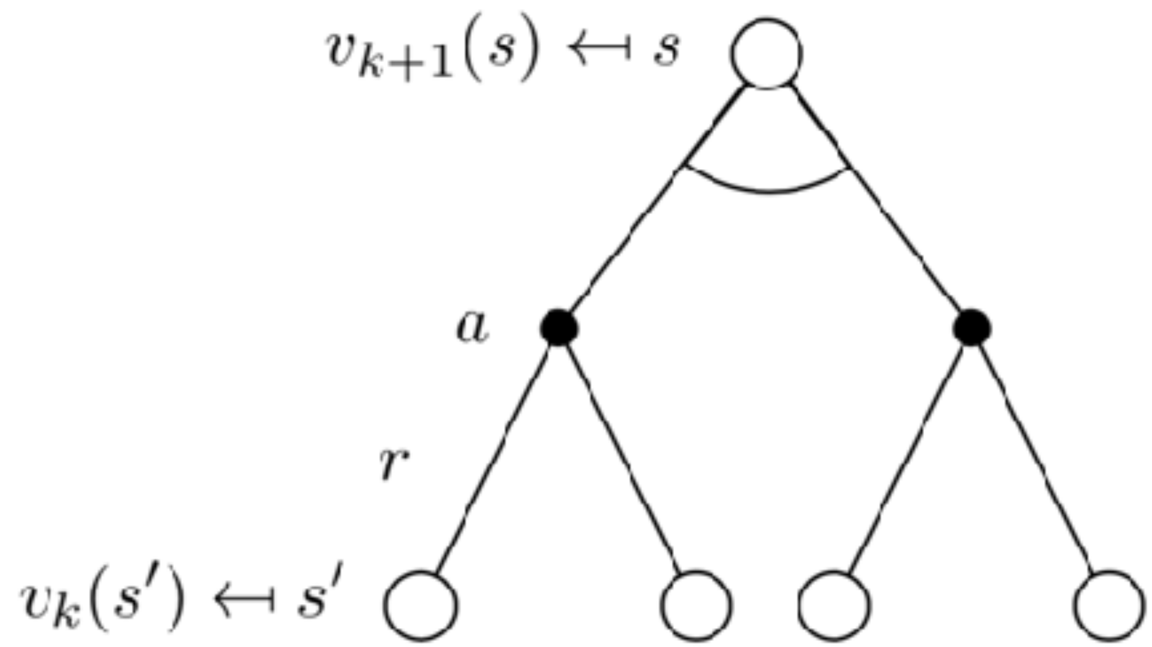


# Planning

# *Value Iteration*

- Iterative application of Bellman Optimality backup

$v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_*$




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

# Planning
## *Synchronous DP Algorithms*

| Problem | Bellman Equation | Algorithm |
|---|---|---|
| Prediction | Bellman Expectation Equation | Iterative Policy Evaluation |
| Control | Bellman Expectation Equation + Greedy Policy Improvement | Policy Iteration |
| Control | Bellman Optimality Equation | Value Iteration |

# Outline

- Characteristics of Reinforcement Learning (RL)
- The RL Problem (MDP, value, policy, Bellman)
- Planning (policy iteration, value iteration)
- Model-free Prediction (MC, TD)
- Model-free Control (Q-Learning)
- Deep Reinforcement Learning (DQN)

# Recap: Components of RL

## *Planning VS Learning*

- Planning
  - the underlying MDP is known
  - agent only needs to perform computations on the given model
  - dynamic programming (policy iteration, value iteration)
- Learning
  - the underlying MDP is initially unknown
  - agent needs to interact with the environment
  - model-free (learn value / policy) / model-based (learn model, plan on it)

# Model-free Prediction

## *MC VS TD*

- Monte Carlo Learning
    - learns from complete trajectories, no bootstrapping
    - estimates values by looking at sample returns, empirical mean return
- Temporal Difference Learning
    - learns from incomplete episodes, by bootstrapping, substituting the remainder of the trajectory with our estimate
    - updates a guess towards a guess

# Model-free Prediction
## *MC VS TD*

- Monte Carlo Learning
  - learns from complete trajectories, no bootstrapping
  - estimates values by looking at sample returns, empirical mean return
- Temporal Difference Learning
  - learns from incomplete episodes, by bootstrapping, substituting the remainder of the trajectory with our estimate
  - updates a guess towards a guess

# Model-free Prediction
# *MC VS TD*

- Monte Carlo Learning
  - learns from complete trajectories, no bootstrapping
  - estimates values by looking at sample returns, empirical mean return
- Temporal Difference Learning
  - learns from incomplete episodes, by bootstrapping, substituting the remainder of the trajectory with our estimate
  - updates a guess towards a guess

# Model-free Prediction
# *MC*

- Goal:

$$\text{learn } v_\pi \text{ from episodes of experience under policy } \pi$$

- Recall: Return is the total discounted reward:

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

- Recall: Value function is the expected return:

$$v_\pi(\mathbf{s}) = \mathbb{E}_\pi \left[ G_t | \mathbf{S}_t = \mathbf{s} \right]$$

- MC policy evaluation (every visit MC):

uses ***empirical mean return*** instead of ***expected return***

# Model-free Prediction
# *MC*

- Goal:

$$\text{learn } v_\pi \text{ from episodes of experience under policy } \pi$$

- Recall: Return is the total discounted reward:

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

- Recall: Value function is the expected return:

$$v_\pi(\mathbf{s}) = \mathbb{E}_\pi \left[ G_t | \mathbf{S}_t = \mathbf{s} \right]$$

- MC policy evaluation (every visit MC):

uses ***empirical mean return*** instead of ***expected return***

# Model-free Prediction
## *MC*

- Goal:

  learn $v_\pi$ from episodes of experience under policy $\pi$

- Recall: Return is the total discounted reward:

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

- Recall: Value function is the expected return:

$$v_\pi(\mathbf{s}) = \mathbb{E}_\pi \left[ G_t | \mathbf{S}_t = \mathbf{s} \right]$$

- MC policy evaluation (every visit MC):

  uses ***empirical mean return*** instead of ***expected return***

# Model-free Prediction
# *MC*

- Goal:

$$\text{learn } v_\pi \text{ from episodes of experience under policy } \pi$$

- Recall: Return is the total discounted reward:

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

- Recall: Value function is the expected return:

$$v_\pi(\mathbf{s}) = \mathbb{E}_\pi \left[ G_t | \mathbf{S}_t = \mathbf{s} \right]$$

- MC policy evaluation (every visit MC):

uses ***empirical mean return*** instead of ***expected return***

# Model-free Prediction
## *MC -> TD*

- Goal:

learn $v_\pi$ from episodes of experience under policy $\pi$

- MC:

updates $V(\mathbf{S}_t)$ towards actual return: $G_t$

$$V(\mathbf{S}_t) \leftarrow V(\mathbf{S}_t) + \alpha(G_t - V(\mathbf{S}_t))$$

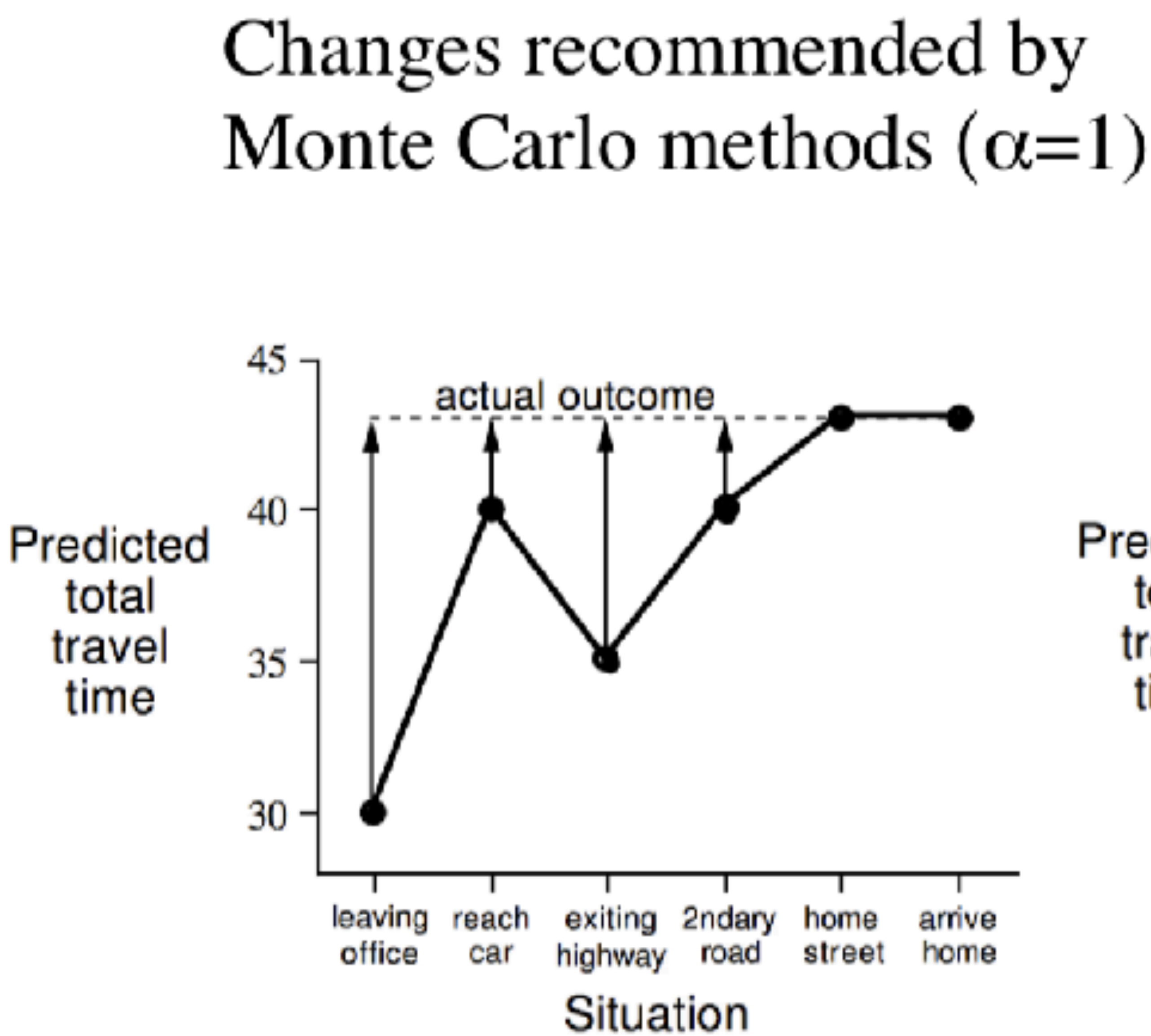
- TD:

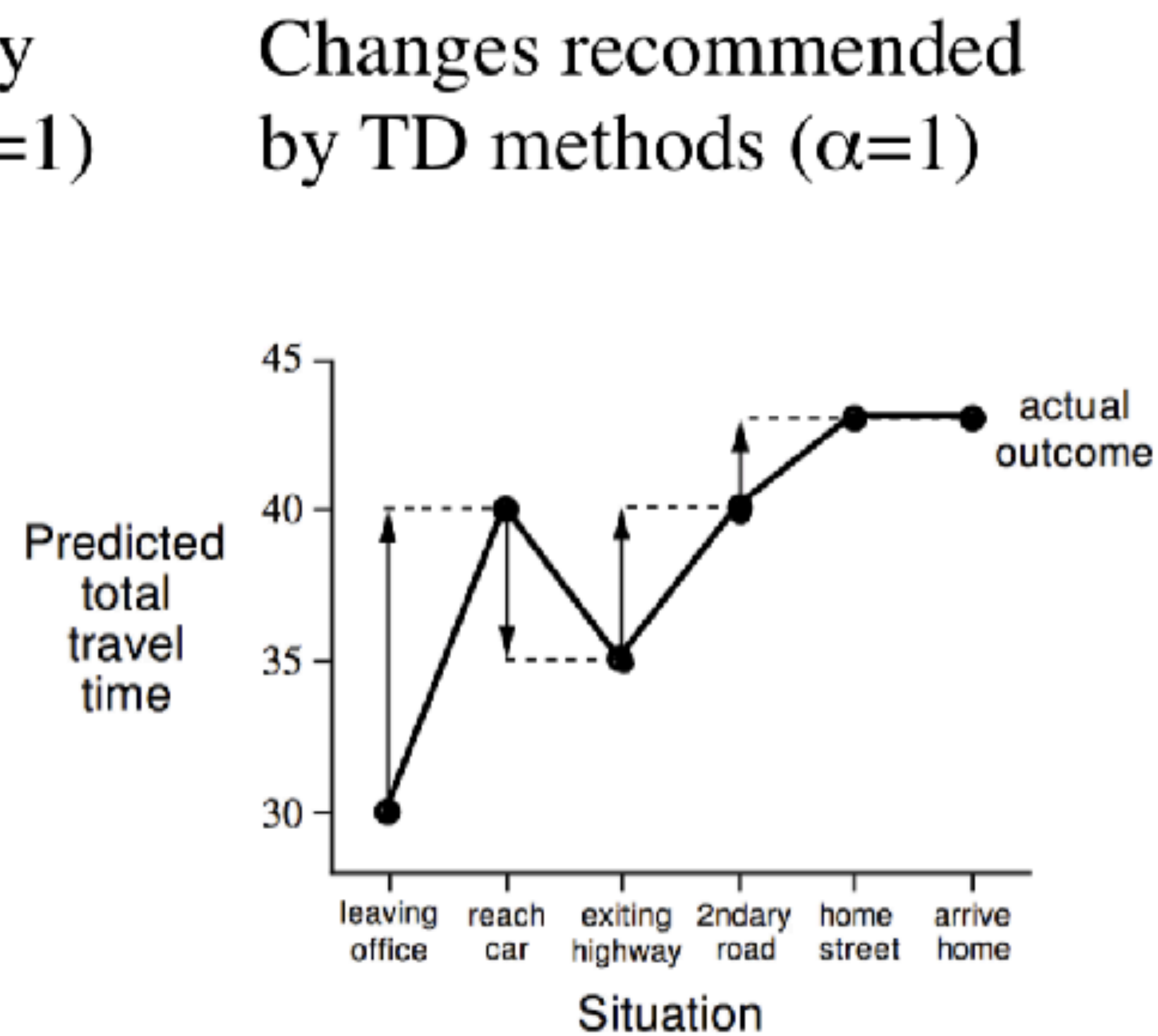
updates $V(\mathbf{S}_t)$ towards estimated return: $\mathbf{R}_{t+1} + \gamma V(\mathbf{S}_{t+1})$

$$V(\mathbf{S}_t) \leftarrow V(\mathbf{S}_t) + \alpha(\mathbf{R}_{t+1} + \gamma V(\mathbf{S}_{t+1}) - V(\mathbf{S}_t))$$

# Model-free Prediction

## *MC VS TD: Driving Home*

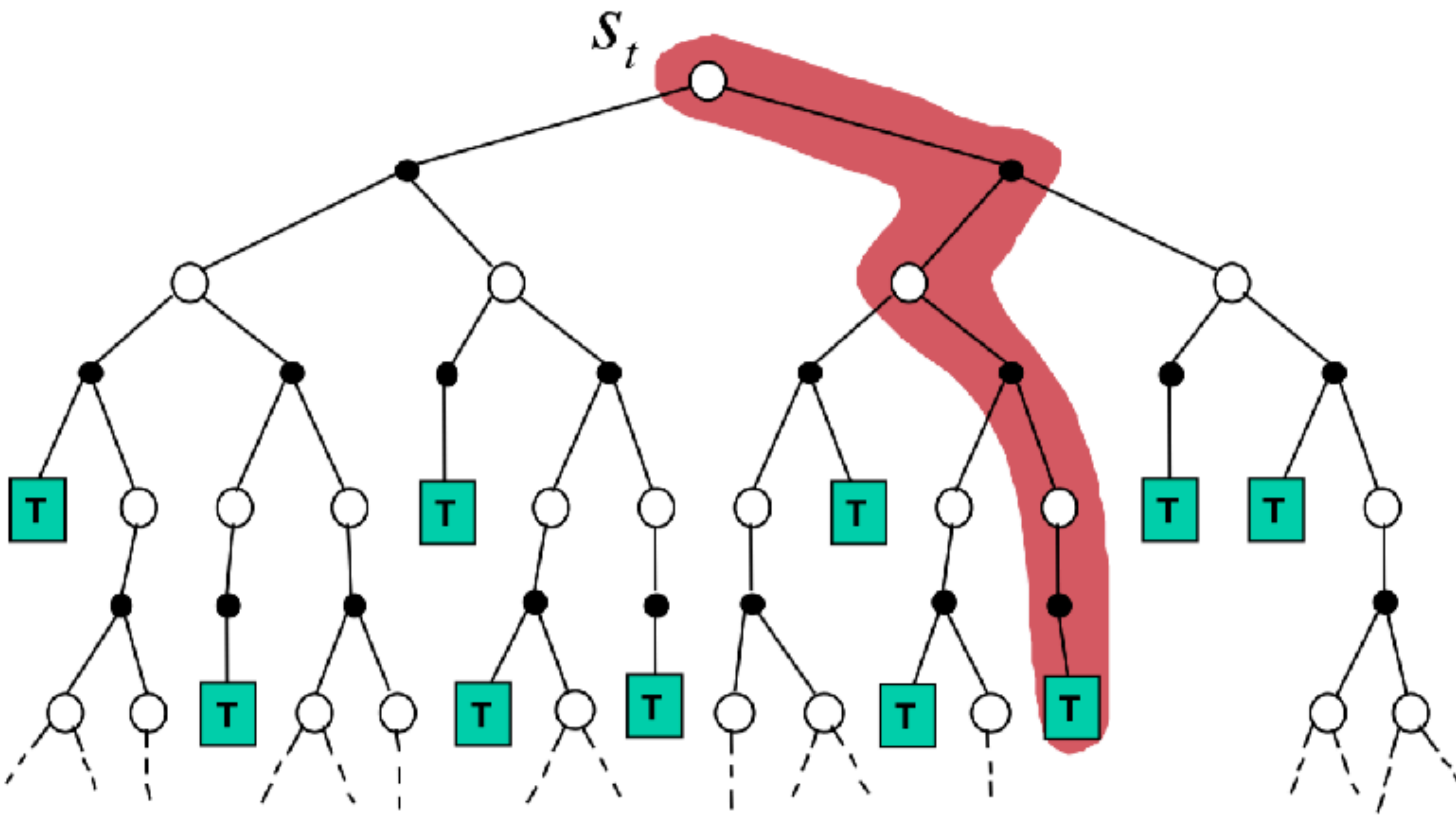


# Model-free Prediction

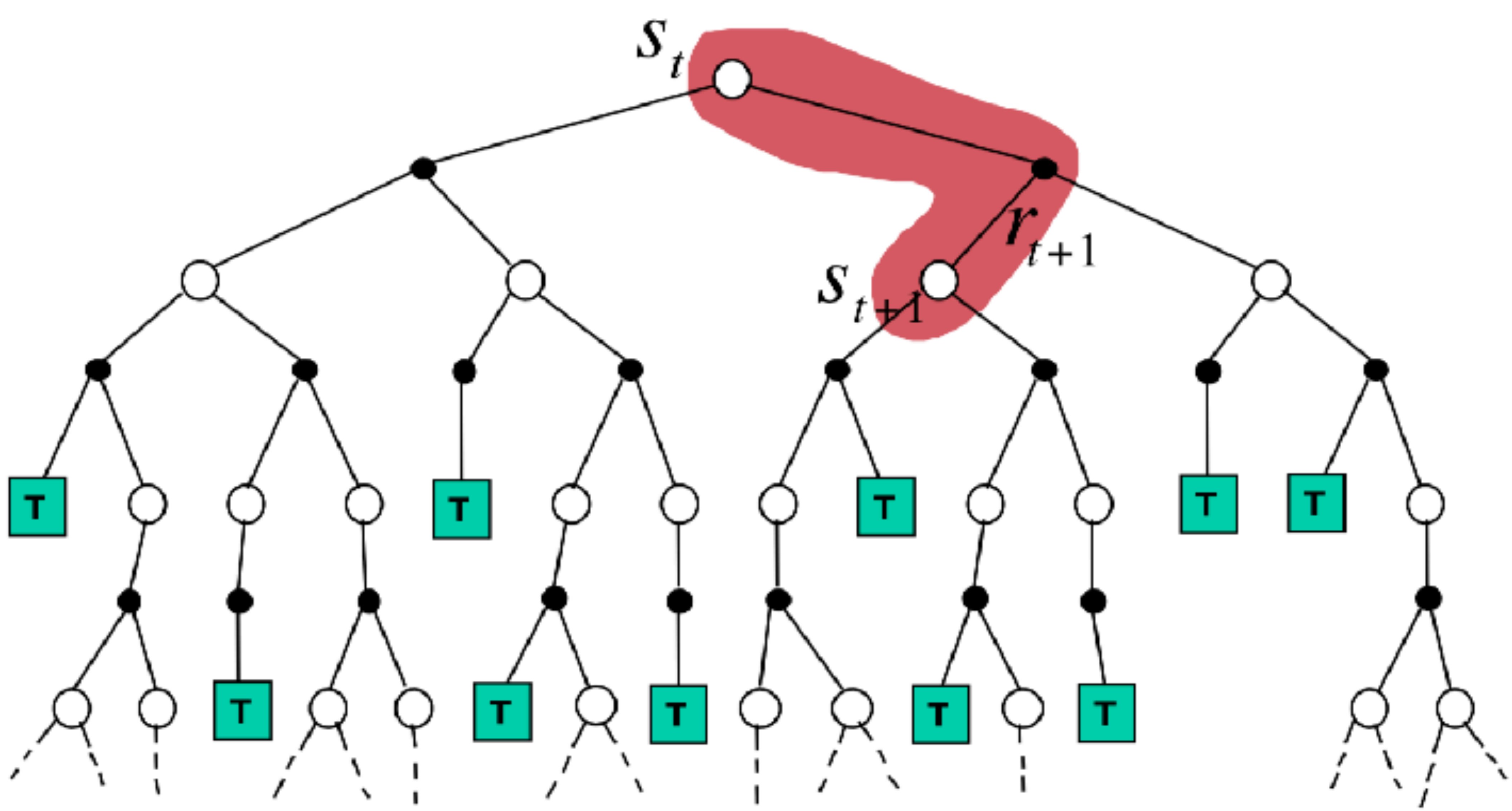
## *MC Backup*

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$



# Model-free Prediction
## *TD Backup*

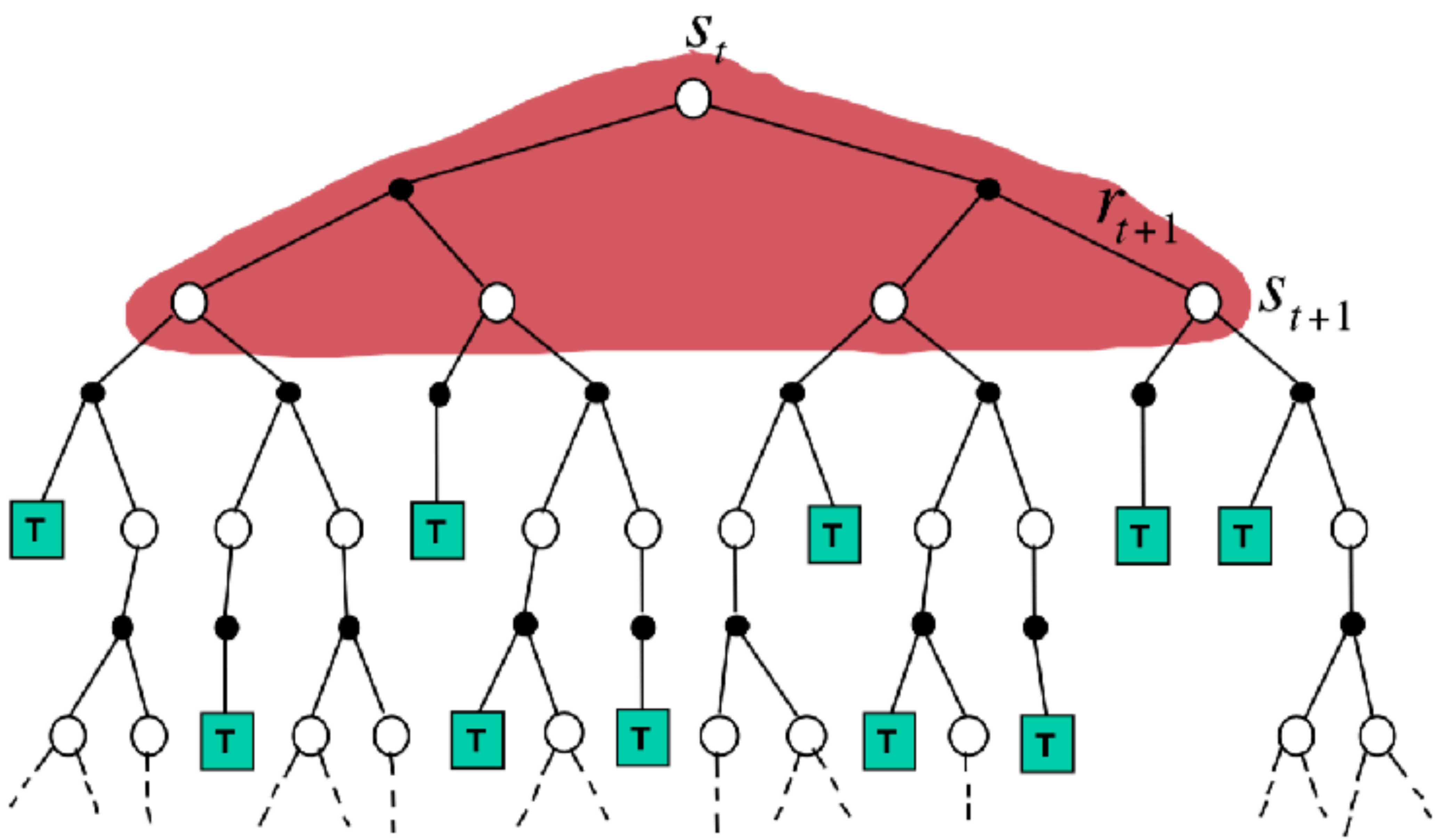
$$V(S_t) \leftarrow V(S_t) + \alpha \left( R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right)$$



# Model-free Prediction

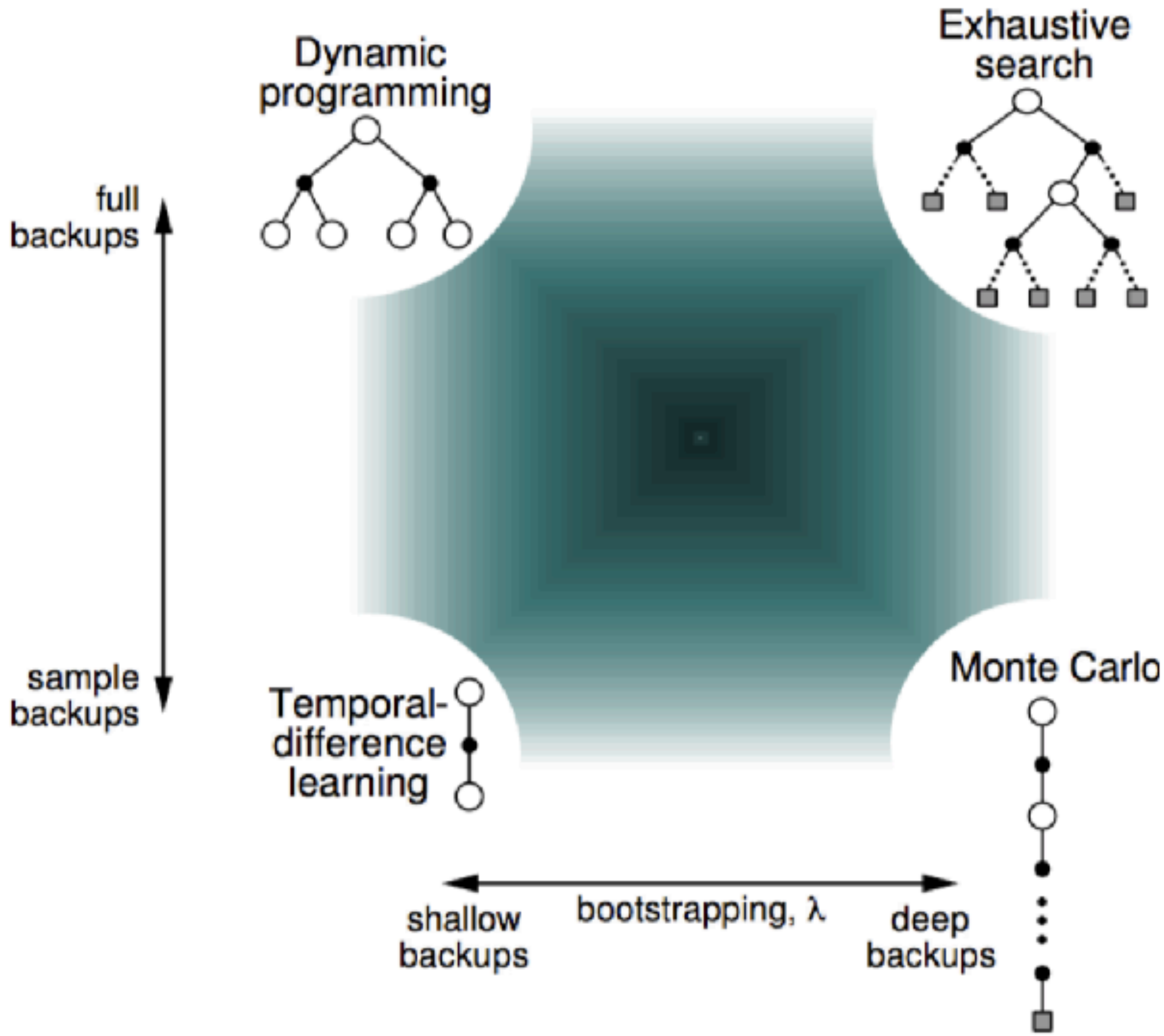
## *DP Backup*

$$V(S_t) \leftarrow \mathbb{E}_\pi \left[ R_{t+1} + \gamma V(S_{t+1}) \right]$$



# Model-free Prediction

## *Unified View*



# Outline

- Characteristics of Reinforcement Learning (RL)
- The RL Problem (MDP, value, policy, Bellman)
- Planning (policy iteration, value iteration)
- Model-free Prediction (MC, TD)
- Model-free Control (Q-Learning)
- Deep Reinforcement Learning (DQN)

# Model-free Control

## *Why model-free?*

- MDP is unknown:

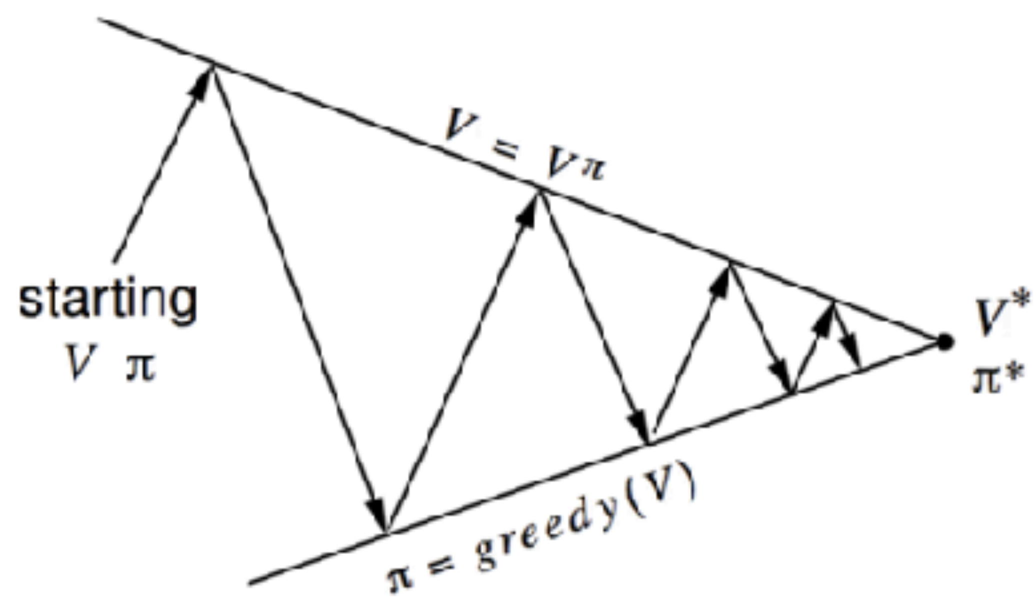
  but experience can be sampled

- MDP is known:

  but too big to use except to sample from it

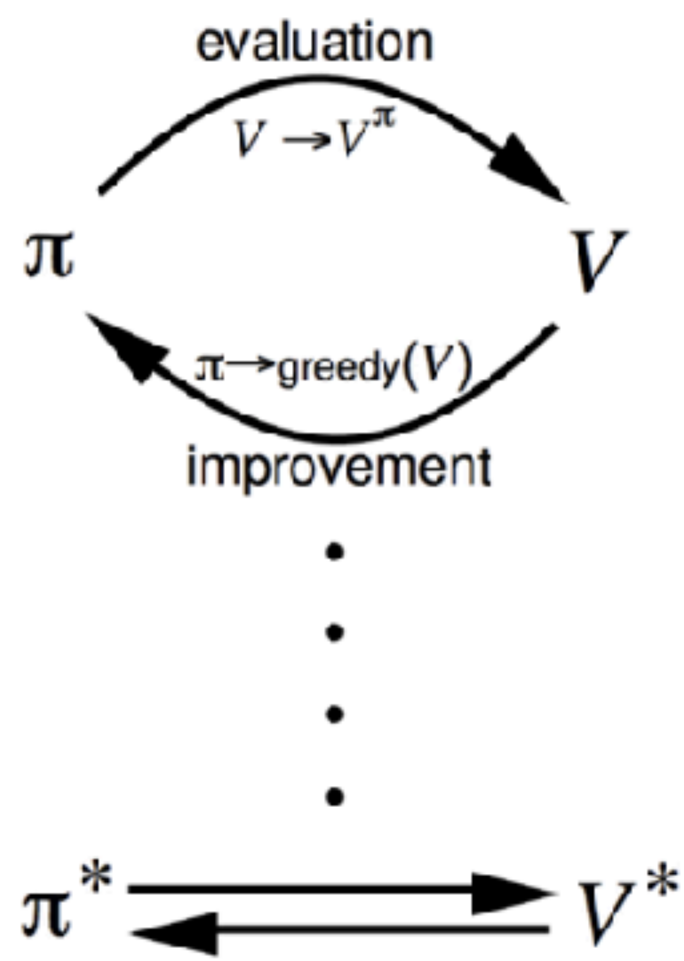
# Recap: Planning
## *Policy Iteration*

- Evaluate the given policy and get: $v_\pi$
- Get an improved policy by acting greedily: $\pi' = \text{greedy}(v_\pi)$




Policy evaluation Estimate $v_\pi$
Iterative policy evaluation

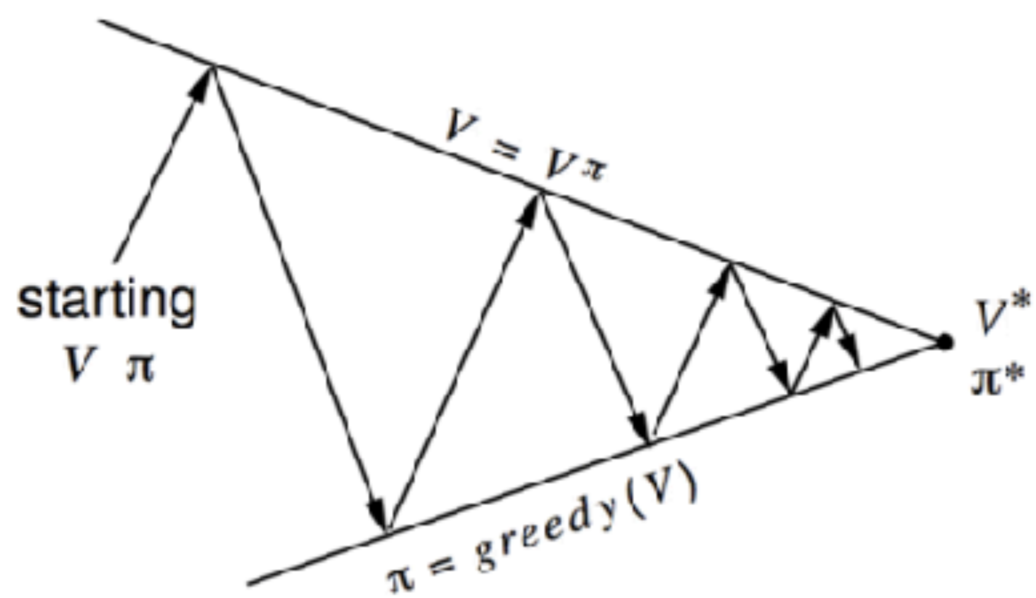
Policy improvement Generate $\pi' \geq \pi$
Greedy policy improvement



# Model-free Control

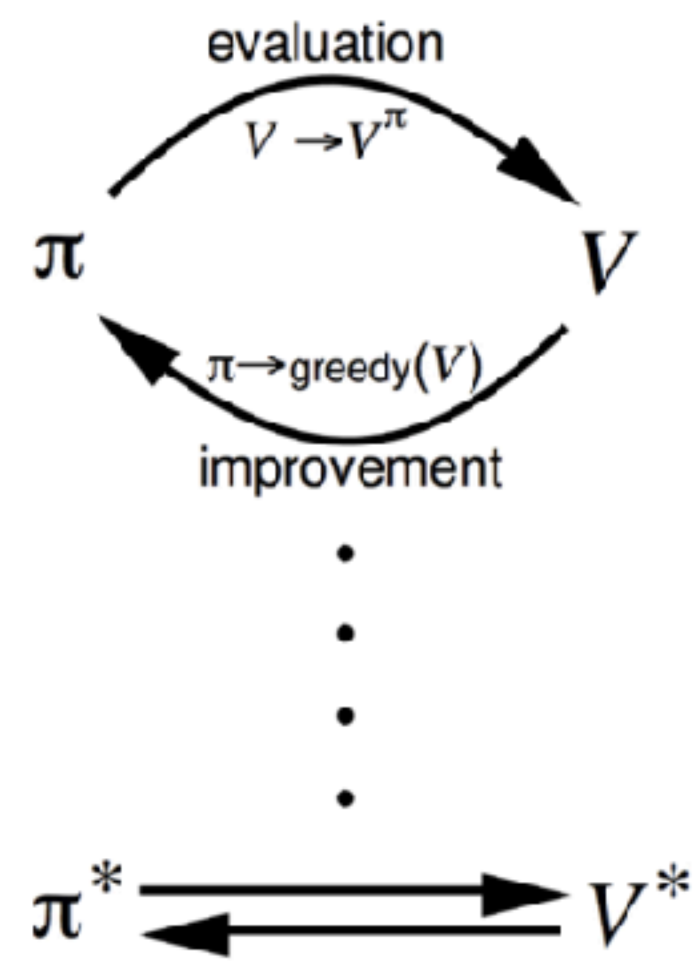
## *Generalized Policy Iteration*

- Evaluate the given policy and get: $v_\pi$
- Get an improved policy by acting greedily: $\pi' = \text{greedy}(v_\pi)$




Policy evaluation Estimate $v_\pi$
Any policy evaluation algorithm

Policy improvement Generate $\pi' \geq \pi$
Any policy improvement algorithm



# Model-free Control
# *V -> Q*

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

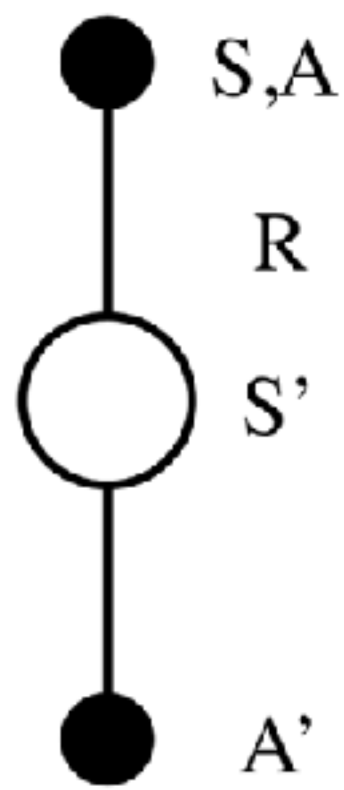
$$\pi'(\mathbf{s}) = \arg\max_{\mathbf{a} \in \mathcal{A}} \left(\mathcal{R}_{\mathbf{s}}^{\mathbf{a}} + \mathcal{P}_{\mathbf{ss}'}^{\mathbf{a}} V(\mathbf{s}')\right)$$

- Greedy policy improvement over Q(s,a) is model-free

$$\pi'(\mathbf{s}) = \arg\max_{\mathbf{a} \in \mathcal{A}} Q(\mathbf{s}, \mathbf{a})$$

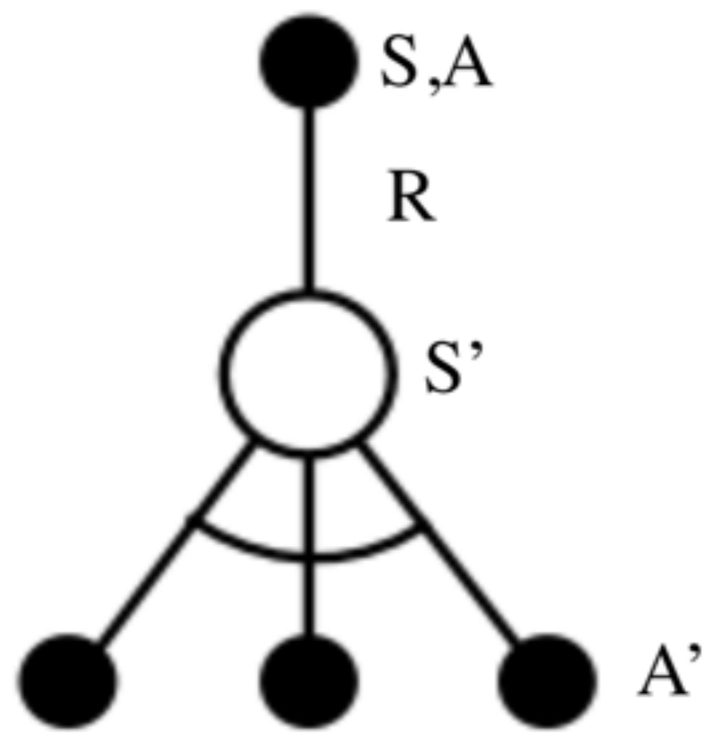
# Model-free Control

## *SARSA*




$$Q(\mathbf{S},\mathbf{A}) \leftarrow Q(\mathbf{S},\mathbf{A}) + \alpha\left(\mathbf{R} + \gamma Q(\mathbf{S}',\mathbf{A}') - Q(\mathbf{S},\mathbf{A})\right)$$

# Model-free Control
## *Q-Learning*




$$Q(\mathbf{S},\mathbf{A}) \leftarrow Q(\mathbf{S},\mathbf{A}) + \alpha\left(\mathbf{R} + \gamma \max_{\mathbf{a}'} Q(\mathbf{S}',\mathbf{a}') - Q(\mathbf{S},\mathbf{A})\right)$$

# Model-free Control

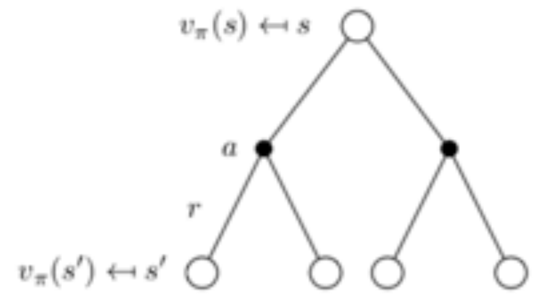
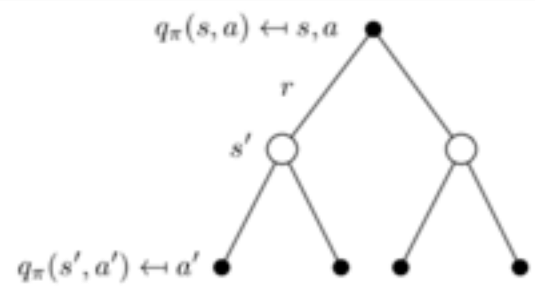
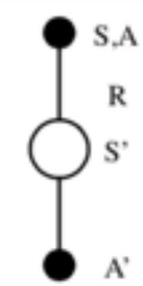
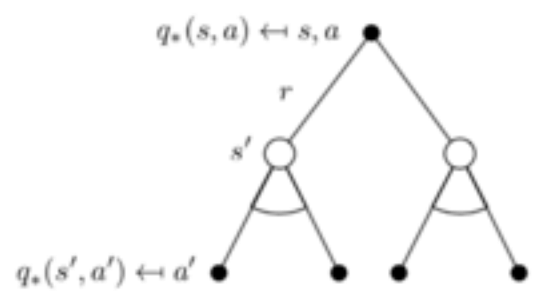
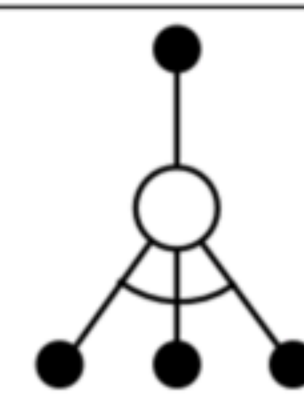
## *SARSA VS Q-Learning*

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., $\varepsilon$-greedy)
    Repeat (for each step of episode):
        Take action $A$, observe $R$, $S'$
        Choose $A'$ from $S'$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        $Q(S,A) \leftarrow Q(S,A) + \alpha\left[\boxed{R + \gamma Q(S',A')} - Q(S,A)\right]$
        $S \leftarrow S'$; $\boxed{A \leftarrow A'}$;
    until $S$ is terminal

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., $\varepsilon$-greedy)
        Take action $A$, observe $R$, $S'$
        $Q(S,A) \leftarrow Q(S,A) + \alpha\left[\boxed{R + \gamma \max_a Q(S',a)} - Q(S,A)\right]$
        $S \leftarrow S'$;
    until $S$ is terminal

# Model-free Control

## *DP VS TD*

| | *Full Backup (DP)* | *Sample Backup (TD)* |
|---|---|---|
| Bellman Expectation Equation for $v_\pi(s)$ |   Iterative Policy Evaluation |  TD Learning |
| Bellman Expectation Equation for $q_\pi(s, a)$ |   Q-Policy Iteration |   Sarsa |
| Bellman Optimality Equation for $q_*(s, a)$ |   Q-Value Iteration |  Q-Learning |

# Model-free Control

## *DP VS TD*

| *Full Backup (DP)* | *Sample Backup (TD)* |
| --- | --- |
| Iterative Policy Evaluation | TD Learning |
| $V(s) \leftarrow \mathbb{E}\left[R + \gamma V(S') \mid s\right]$ | $V(S) \xleftarrow{\alpha} R + \gamma V(S')$ |
| Q-Policy Iteration | Sarsa |
| $Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma Q(S',A') \mid s,a\right]$ | $Q(S,A) \xleftarrow{\alpha} R + \gamma Q(S',A')$ |
| Q-Value Iteration | Q-Learning |
| $Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q(S',a') \mid s,a\right]$ | $Q(S,A) \xleftarrow{\alpha} R + \gamma \max_{a' \in \mathcal{A}} Q(S',a')$ |

# Outline

- Characteristics of Reinforcement Learning (RL)
- The RL Problem (MDP, value, policy, Bellman)
- Planning (policy iteration, value iteration)
- Model-free Prediction (MC, TD)
- Model-free Control (Q-Learning)
- Deep Reinforcement Learning (DQN)

# Deep Reinforcement Learning
## *Why?*

- So far we represented value function by a lookup table
    - every state s has an entry V(s)
    - every state-action pair (s, a) has an entry Q(s, a)
- Problem w/ large MDPs
    - too many states and/or actions to store in memory
    - too slow to learn the value of each state individually

# Deep Reinforcement Learning

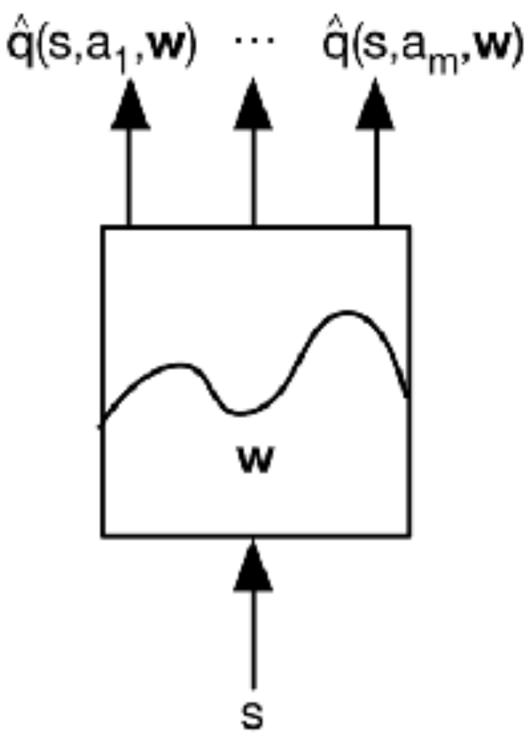
## *Why?*

- So far we represented value function by a lookup table
  - every state s has an entry V(s)
  - every state-action pair (s, a) has an entry Q(s, a)
- Problem w/ large MDPs
  - too many states and/or actions to store in memory
  - too slow to learn the value of each state individually

# Deep Reinforcement Learning
## *How to?*

- Use deep networks to represent:
  - value function (value-based methods)

$$\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$$
$$\text{or } \hat{q}(s, a, \mathbf{w}) \approx q_\pi(s, a)$$




  - policy (policy-based methods)
  - model (model-based methods)
- Optimize value function / policy / model end-to-end

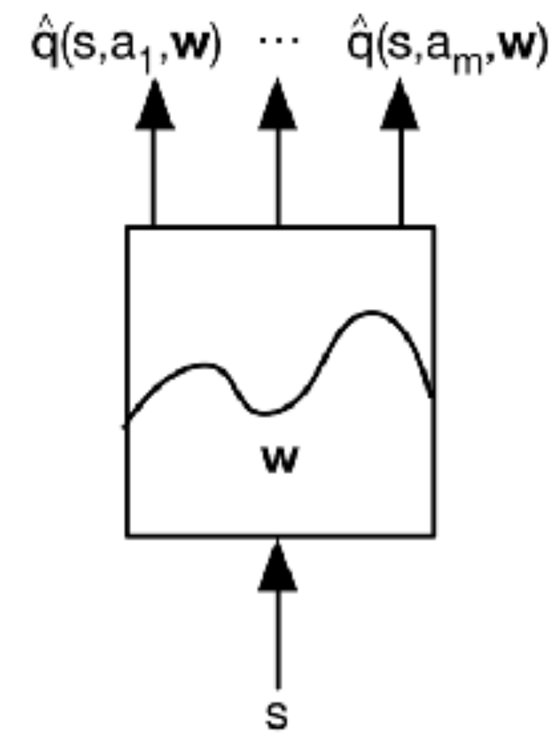
# Deep Reinforcement Learning

## *How to?*

- Use deep networks to represent:
    - value function (value-based methods)

$$\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$$

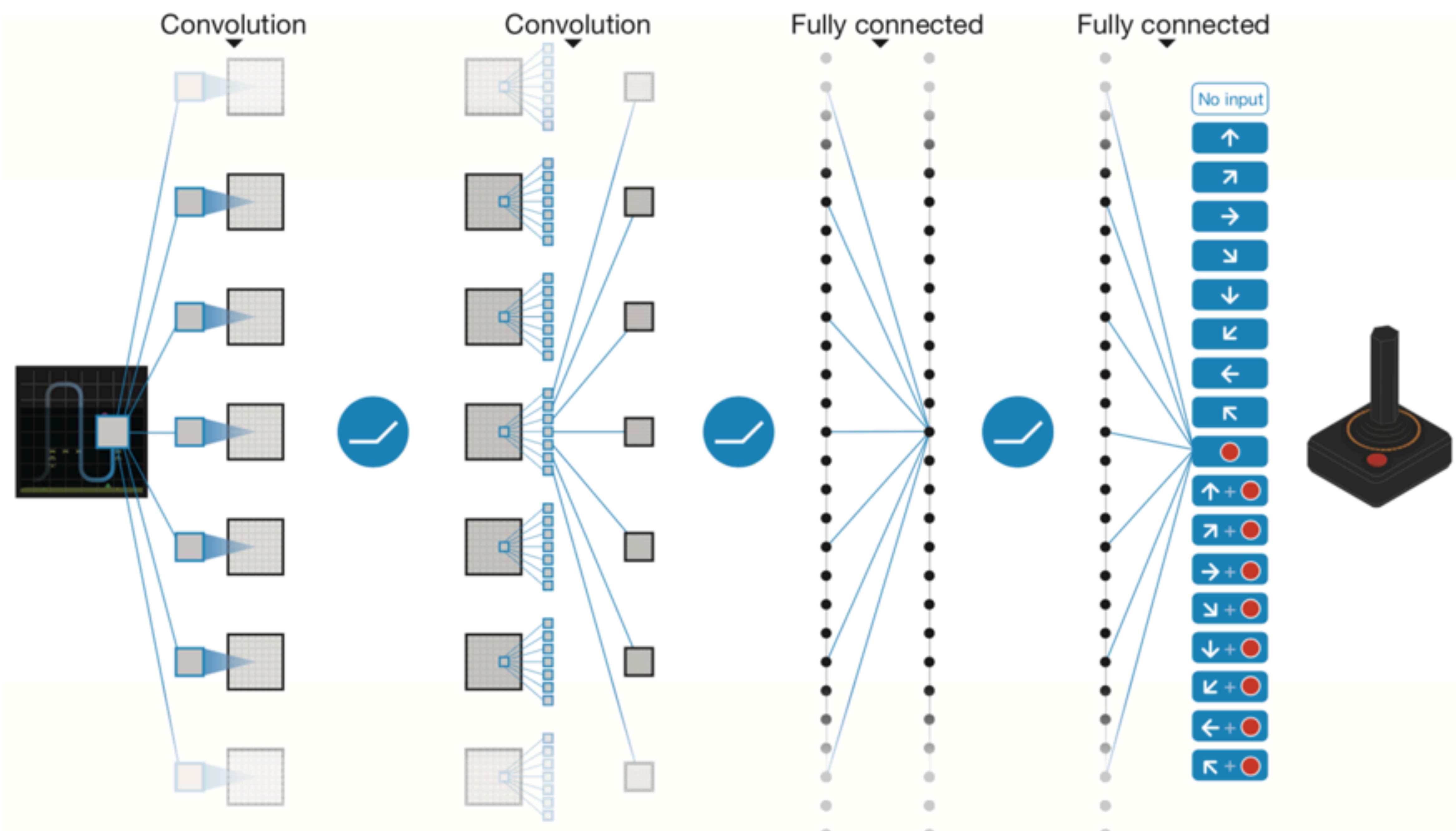
$$\text{or } \hat{q}(s, a, \mathbf{w}) \approx q_\pi(s, a)$$




    - policy (policy-based methods)
    - model (model-based methods)
- Optimize value function / policy / model end-to-end

# Deep Reinforcement Learning

## *Q-learning -> DQN*



# Deep Reinforcement Learning

## *Q-learning -> DQN*

DQN uses experience replay and fixed Q-targets

- Take action $a_t$ according to $\epsilon$-greedy policy
- Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in replay memory $\mathcal{D}$
- Sample random mini-batch of transitions $(s, a, r, s')$ from $\mathcal{D}$
- Compute Q-learning targets w.r.t. old, fixed parameters $w^-$
- Optimise MSE between Q-network and Q-learning targets

$$\mathcal{L}_i(w_i) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}_i}\left[\left(r + \gamma \max_{a'} Q(s', a'; w_i^-) - Q(s, a; w_i)\right)^2\right]$$

- Using variant of stochastic gradient descent

# Deep Reinforcement Learning

## *AI = RL + DL*

- Reinforcement Learning (RL)
  - a general purpose framework for **decision making**
  - learn policies to maximize future reward
- Deep Learning (DL)
  - a general purpose framework for **representation learning**
  - given an objective, learn representation that is required to achieve objective
- DRL: a single agent which can solve any human-level task
  - RL defines the objective
  - DL gives the mechanism
  - RL + DL = general intelligence

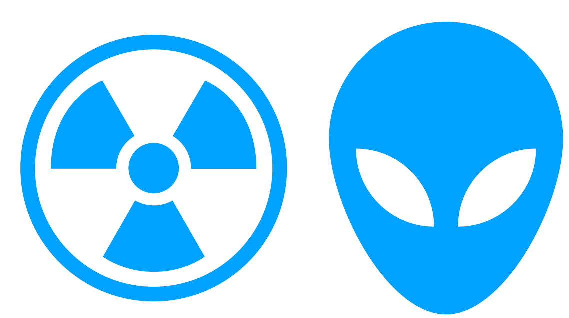
# Deep Reinforcement Learning
## *AI = RL + DL*

- Reinforcement Learning (RL)
  - a general purpose framework for **decision making**
  - learn policies to maximize future reward
- Deep Learning (DL)
  - a general purpose framework for **representation learning**
  - given an objective, learn representation that is required to achieve objective
- DRL: a single agent which can solve any human-level task
  - RL defines the objective
  - DL gives the mechanism
  - RL + DL = general intelligence

# Deep Reinforcement Learning

## *AI = RL + DL*

- Reinforcement Learning (RL)
  - a general purpose framework for **decision making**
  - learn policies to maximize future reward
- Deep Learning (DL)
  - a general purpose framework for **representation learning**
  - given an objective, learn representation that is required to achieve objective
- DRL: a single agent which can solve any human-level task
  - RL defines the objective
  - DL gives the mechanism
  - RL + DL = general intelligence



# Some Recommendations

- Reinforcement Learning from David Silver on YouTube
- Reinforcement Learning, An Introduction, Richard Sutton, 2nd Edition
- DQN Nature Paper: Human-level Control Through Deep Reinforcement Learning
- Flappy Bird:
  - Tabular RL: https://github.com/SarvagyaVaish/FlappyBirdRL
  - Deep RL: https://github.com/songrotek/DRL-FlappyBird
- Many many 3rd party implementations, just search for “deep reinforcement learning”, “dqn”, “a3c” on github
- My implementations in pytorch: https://github.com/jingweiz/pytorch-rl