Introduction to Reinforcement Learning and its Applications to Finance

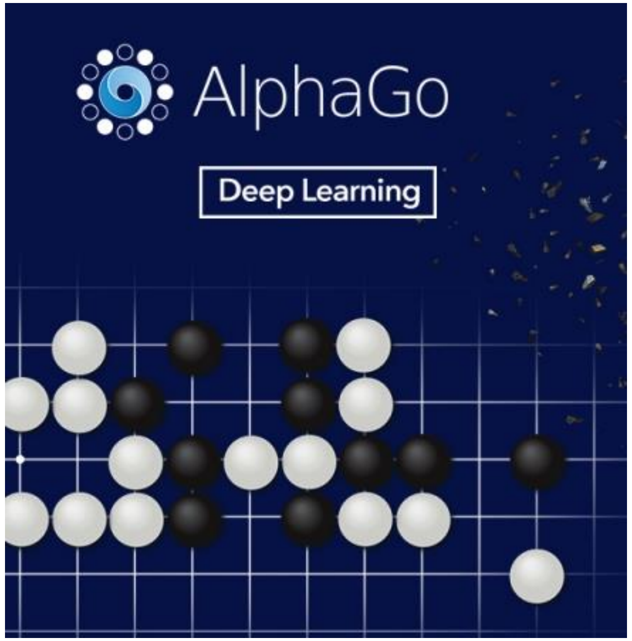
Financial Engineering, Ajou University
Chanho Min

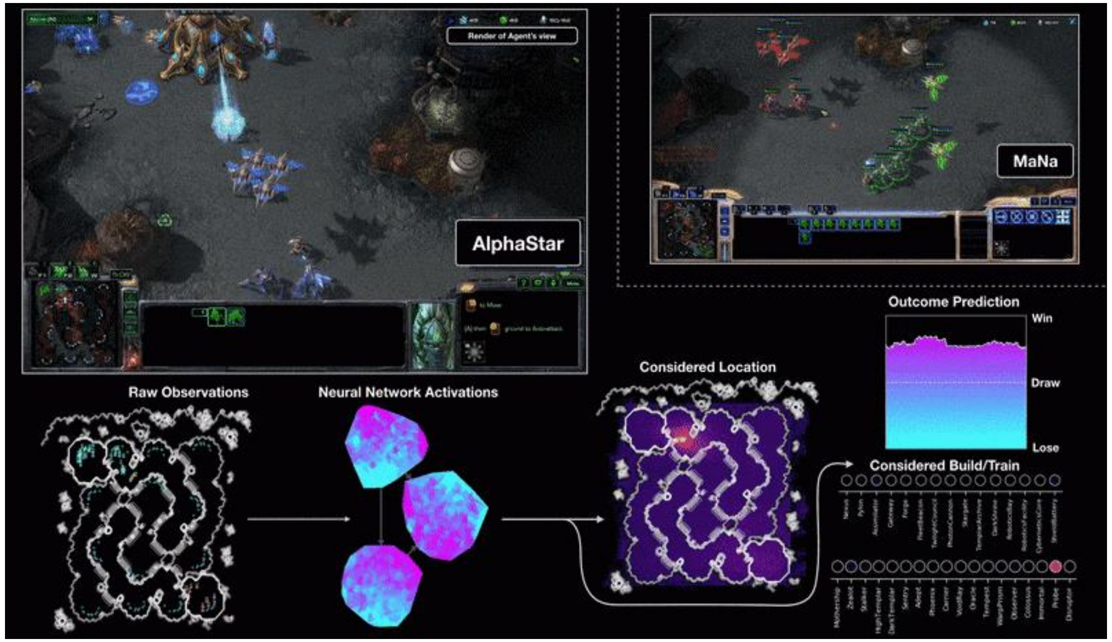
AJOU SCHOOL OF BUSINESS

Contents

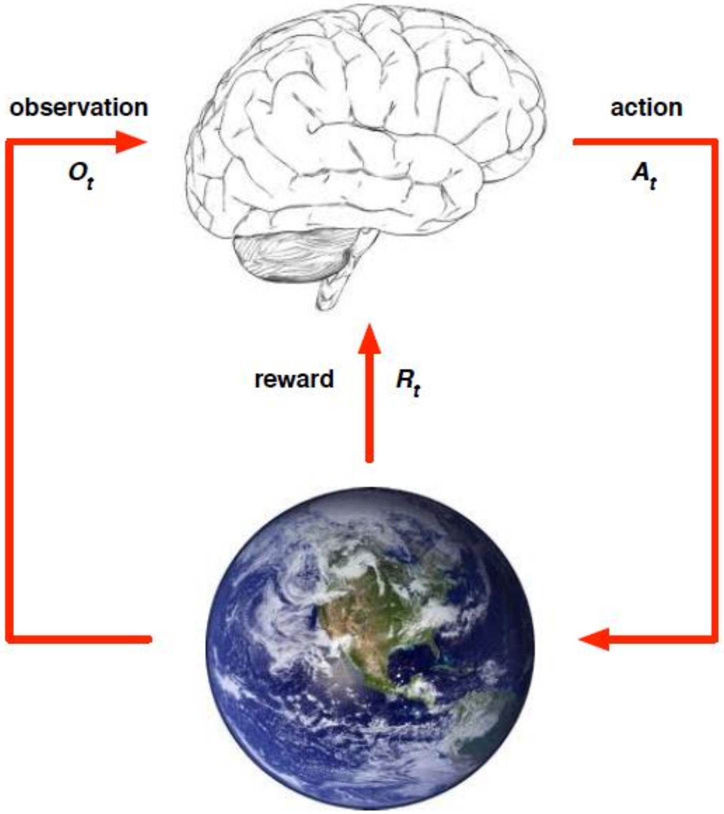
- Reinforcement Learning Introduction
- Mathematical Formulation of RL
  - Value Function and Policy
  - Bellman equation
- Deep Q Networks
  - Monte-Carlo and Temporal Difference
- Categorization of RL methods
- Finance Example

Reinforcement Learning Introduction



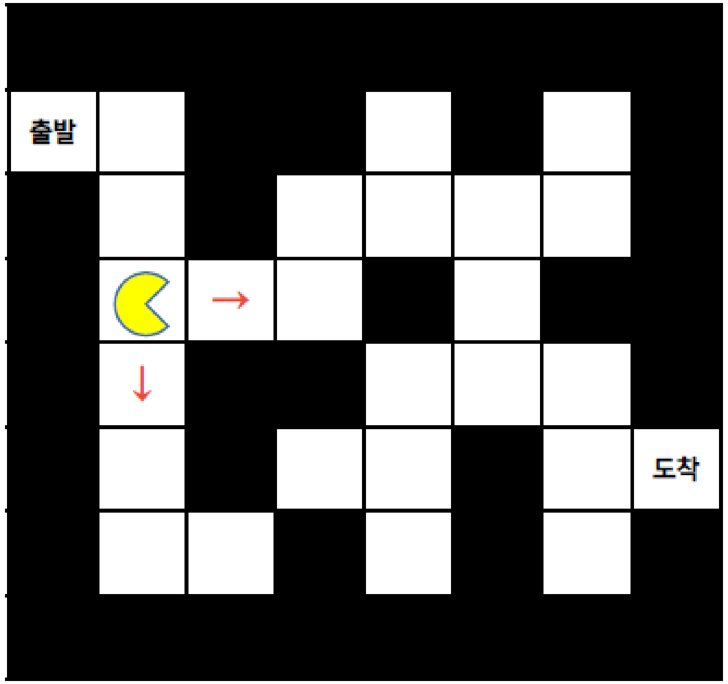


Reinforcement Learning Introduction: Components

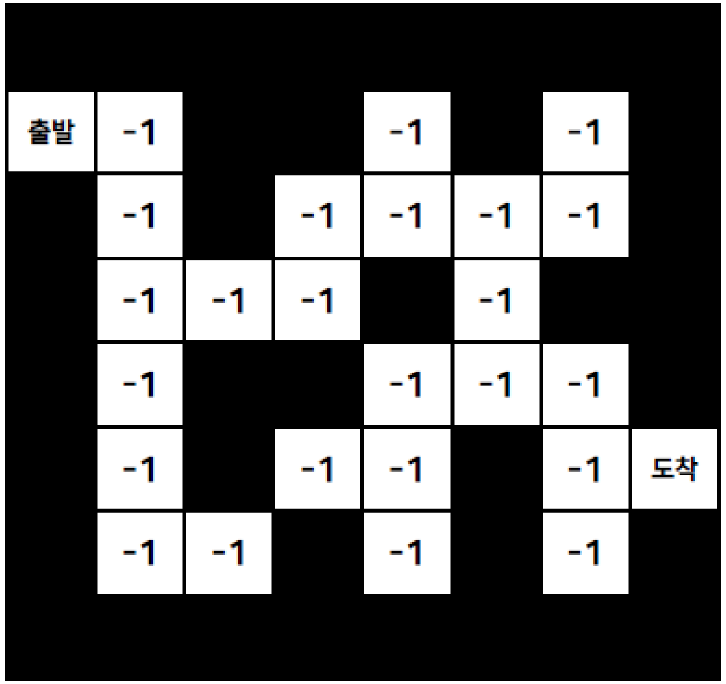


- At each step t the agent:
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at env. step

Reinforcement Learning Introduction: Toy example



Environment



Reward

Mathematical Formulation of Reinforcement Learning: Markov Process

Definition

A state S_t is *Markov* if and only if

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, ..., S_t]$$

Definition

A *Markov Process* (or *Markov Chain*) is a tuple $\langle \mathcal{S}, \mathcal{P} \rangle$

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{P}$ is a state transition probability matrix,
  $\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$

Mathematical Formulation of Reinforcement Learning: MRP

A Markov reward process is a Markov chain with values.

Definition

A *Markov Reward Process* is a tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{P}$ is a state transition probability matrix,
  $\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}_s = \mathbb{E}[R_{t+1} \mid S_t = s]$
- γ is a discount factor, $\gamma \in [0, 1]$

Mathematical Formulation of Reinforcement Learning

Definition

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

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

- The *discount* $\gamma \in [0, 1]$ is the present value of future rewards
- The value of receiving reward R after $k + 1$ time-steps is $\gamma^k R$.
- This values immediate reward above delayed reward.
  - γ close to 0 leads to "myopic" evaluation
  - γ close to 1 leads to "far-sighted" evaluation

Mathematical Formulation of Reinforcement Learning: Value Function

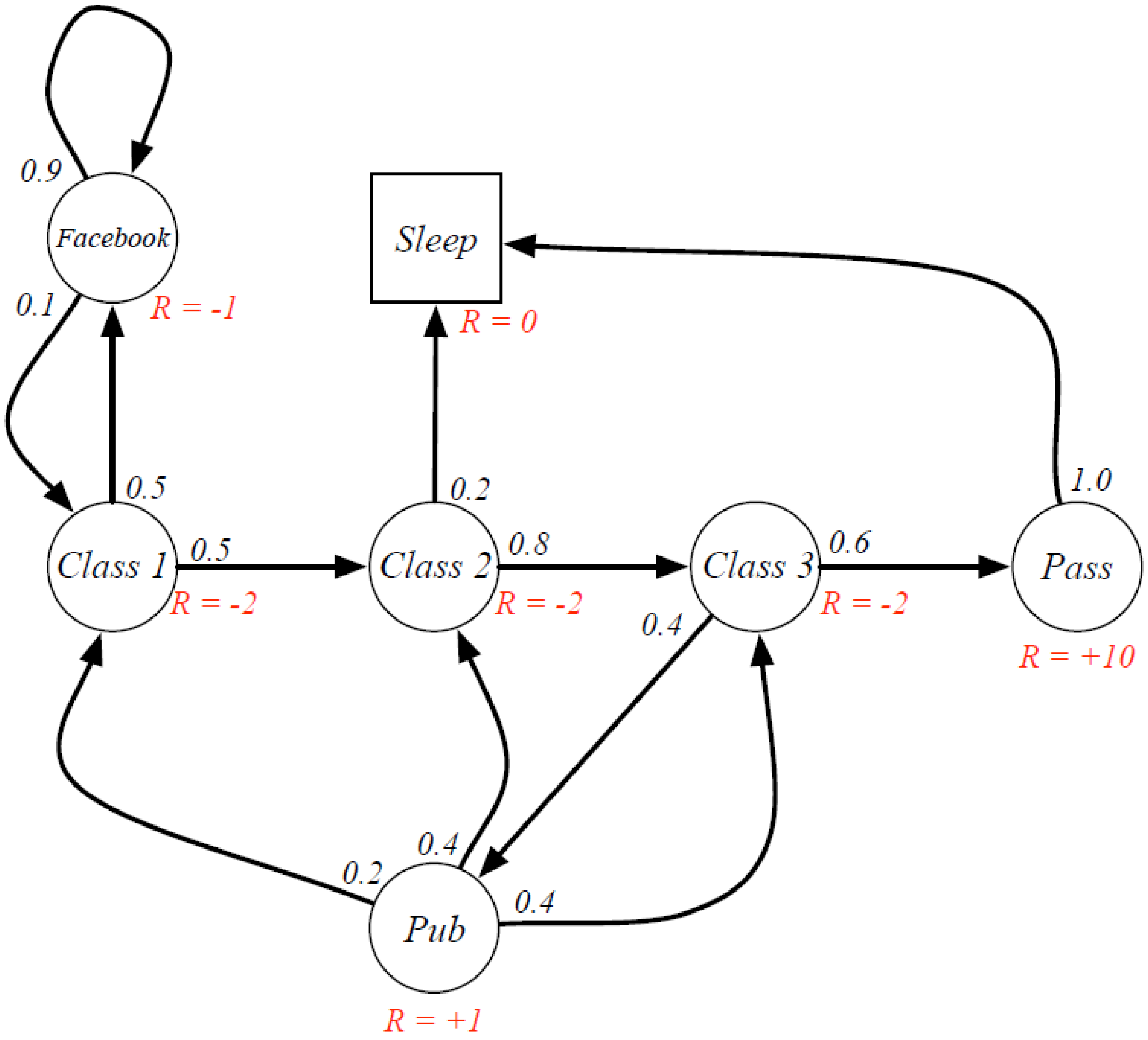
The value function $v(s)$ gives the long-term value of state s

Definition

The *state value function* $v(s)$ of an MRP is the expected return starting from state s

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

Mathematical Formulation of Reinforcement Learning



Mathematical Formulation of Reinforcement Learning

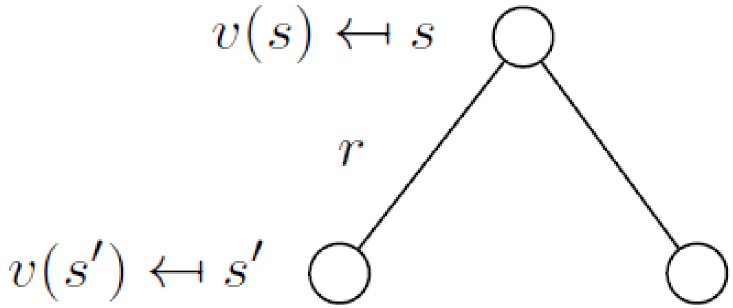
The value function can be decomposed into two parts:

- immediate reward R_{t+1}
- discounted value of successor state $\gamma v(S_{t+1})$

$$
\begin{aligned}
v(s) &= \mathbb{E}\left[G_t \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \ldots\right) \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]
\end{aligned}
$$

Mathematical Formulation of Reinforcement Learning

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



$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')$$

Mathematical Formulation of Reinforcement Learning

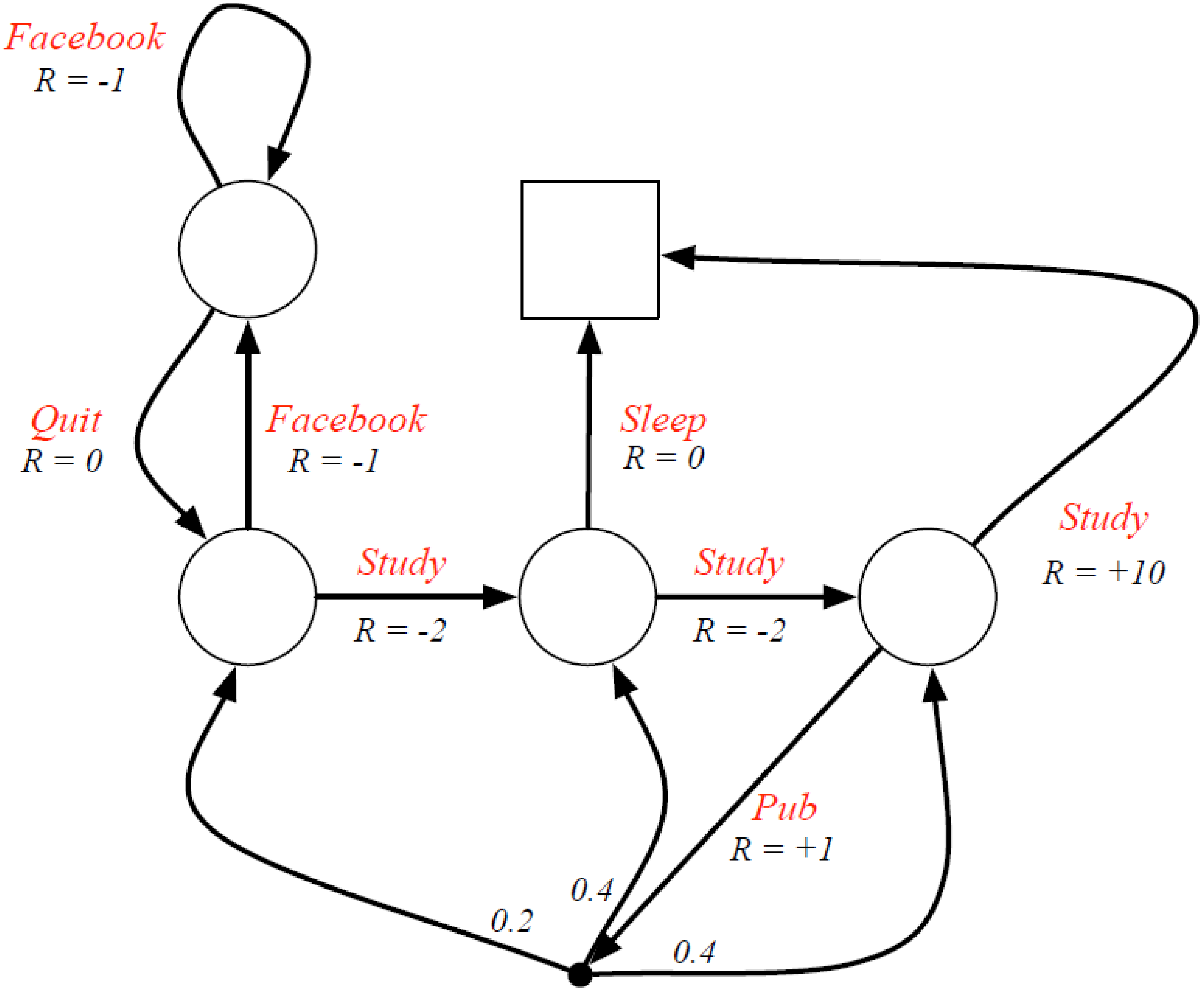
A Markov decision process (MDP) is a Markov reward process with decisions. It is an *environment* in which all states are Markov.

Definition

A *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix,
  $\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
- γ is a discount factor $\gamma \in [0, 1]$.

Mathematical Formulation of Reinforcement Learning



Mathematical Formulation of Reinforcement Learning

Definition

A *policy* π is a distribution over actions given states,

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

- A policy fully defines the behaviour of an agent
- MDP policies depend on the current state (not the history)
- i.e. Policies are *stationary* (time-independent), $A_t \sim \pi(\cdot|S_t), \forall t > 0$

Mathematical Formulation of Reinforcement Learning

Definition

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

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

Definition

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

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

Mathematical Formulation of Reinforcement Learning: Optimal value

Definition

The *optimal state-value function* $v_*(s)$ is the maximum value function over all policies

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

The *optimal action-value function* $q_*(s, a)$ is the maximum action-value function over all policies

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

- The optimal value function specifies the best possible performance in the MDP.
- An MDP is "solved" when we know the optimal value fn.

Solving Reinforcement Learning: Bellman eq.

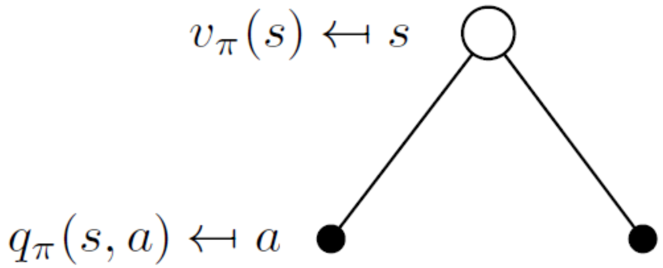
The state-value function can again be decomposed into immediate reward plus discounted value of successor state,

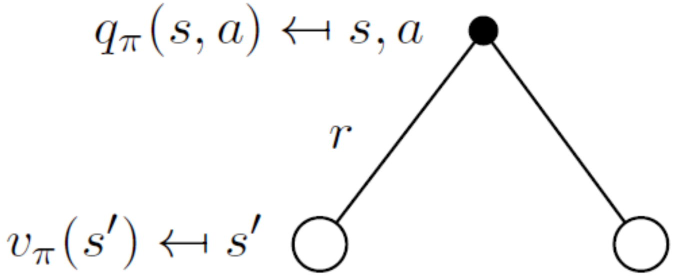
$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s \right]$$

The action-value function can similarly be decomposed,

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

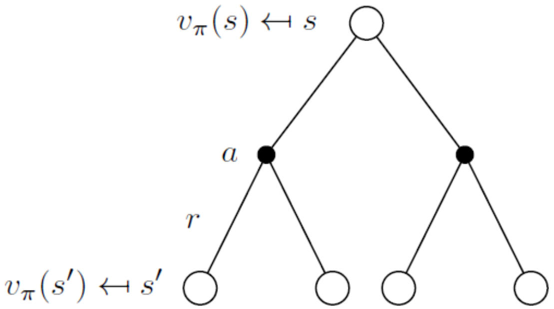
Solving Reinforcement Learning: Bellman eq.

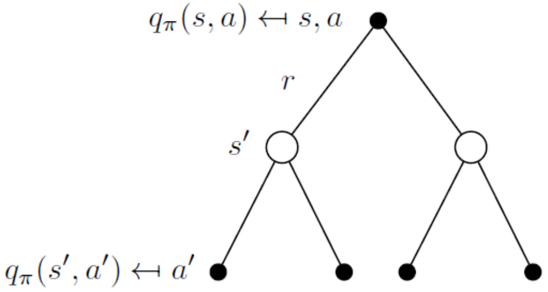




$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

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

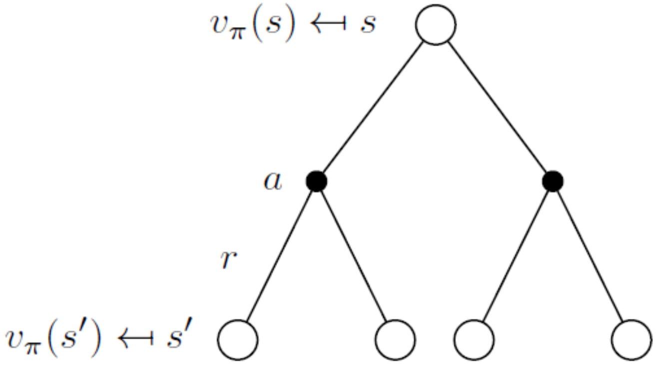




$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

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

Solving Reinforcement Learning: Bellman eq.



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

$$v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi$$

$$v_\pi = (I - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi$$

Solving Reinforcement Learning: Greedy Policy

An optimal policy can be found by maximising over $q_*(s, a)$,

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

- There is always a deterministic optimal policy for any MDP
- If we know $q_*(s, a)$, we immediately have the optimal policy

Solving Reinforcement Learning: Optimal Policy

Define a partial ordering over policies

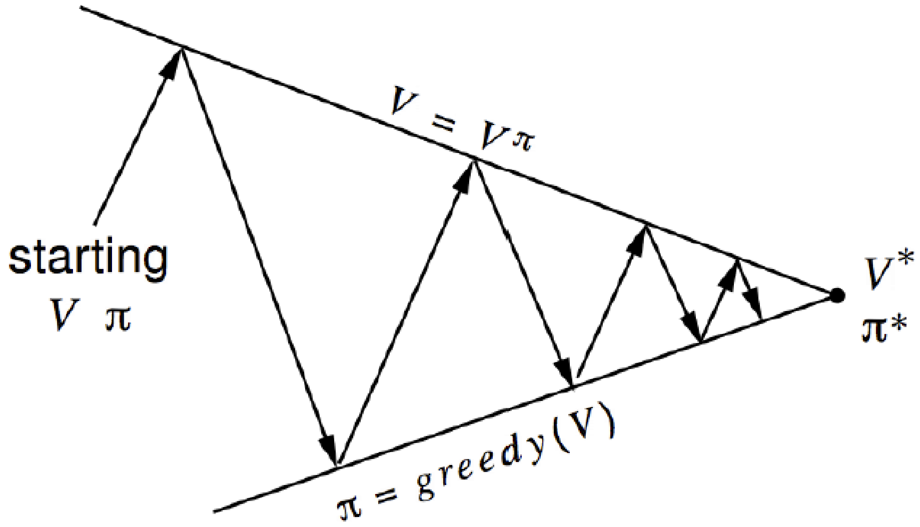
$$\pi \geq \pi' \text{ if } v_\pi(s) \geq v_{\pi'}(s), \forall s$$

Theorem

For any Markov Decision Process

- *There exists an optimal policy* π_* *that is better than or equal to all other policies,* $\pi_* \geq \pi, \forall \pi$
- *All optimal policies achieve the optimal value function,* $v_{\pi_*}(s) = v_*(s)$
- *All optimal policies achieve the optimal action-value function,* $q_{\pi_*}(s, a) = q_*(s, a)$

Solving Reinforcement Learning: Iterative evaluation

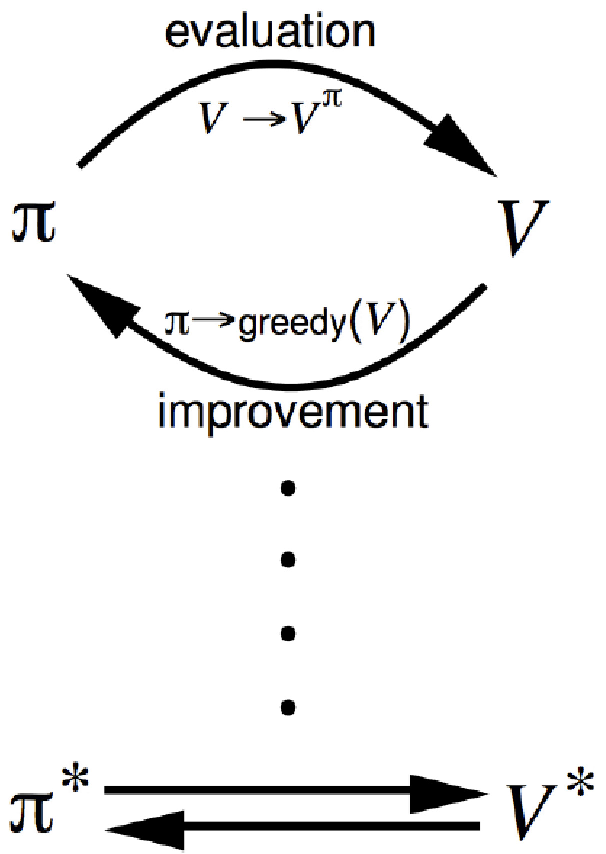


Policy evaluation Estimate v_π

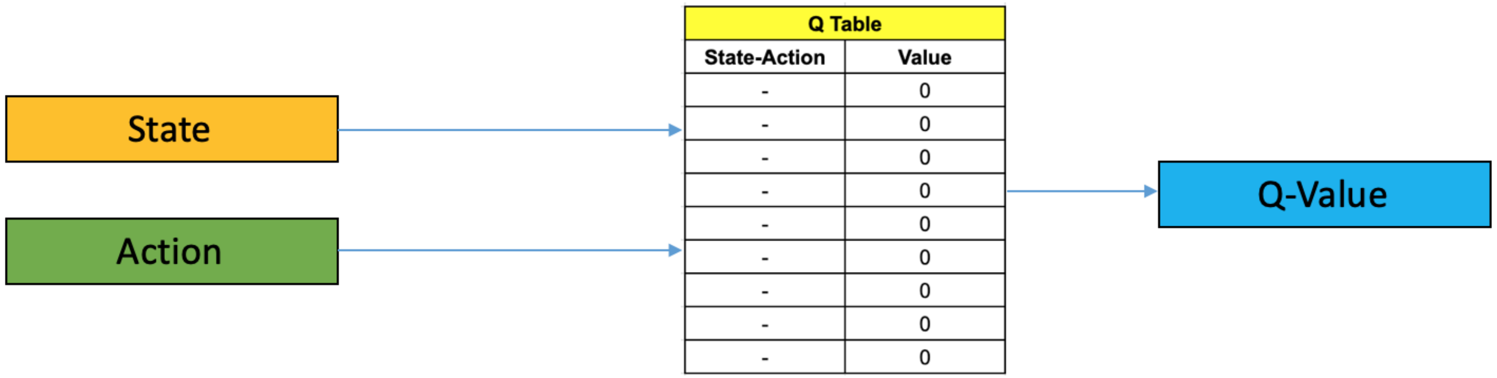
Iterative policy evaluation

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

Greedy policy improvement

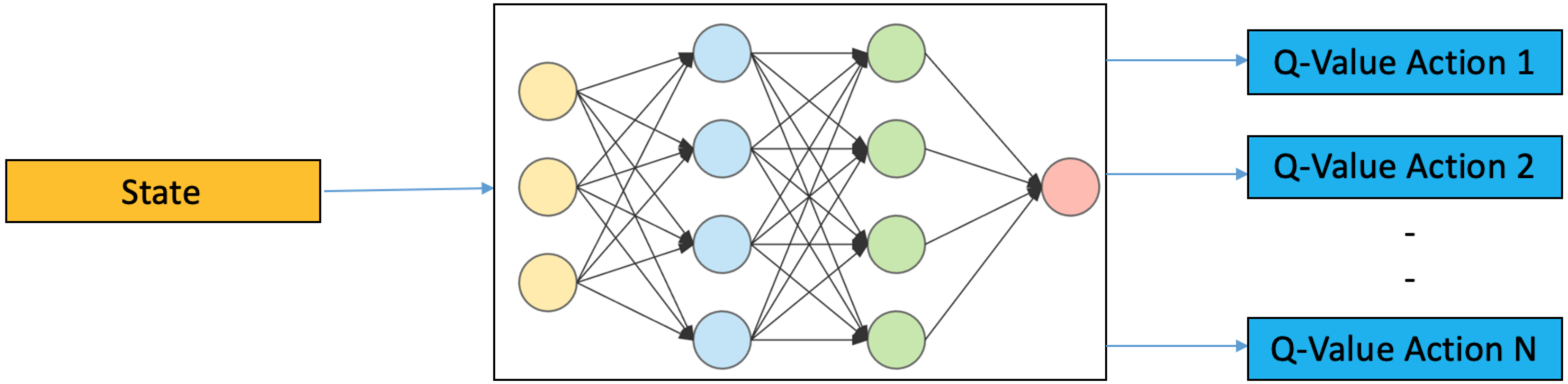


Reinforcement Learning and Deep learning



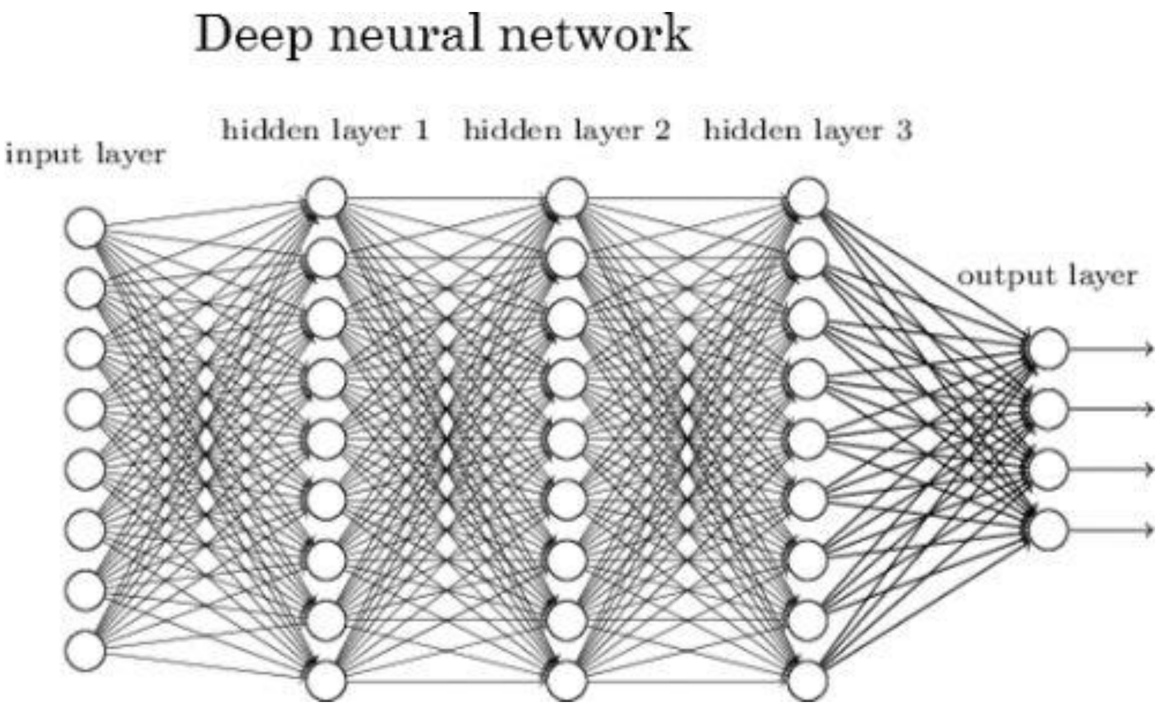
Q Table	
State-Action	Value
-	0
-	0
-	0
-	0
-	0
-	0
-	0
-	0
-	0

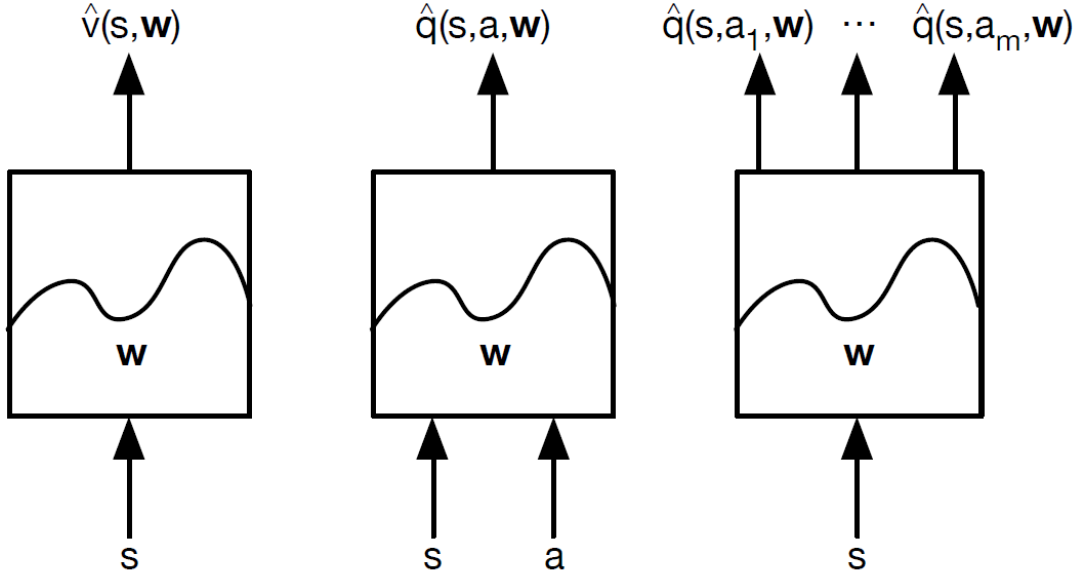
Q Learning



Deep Q Learning

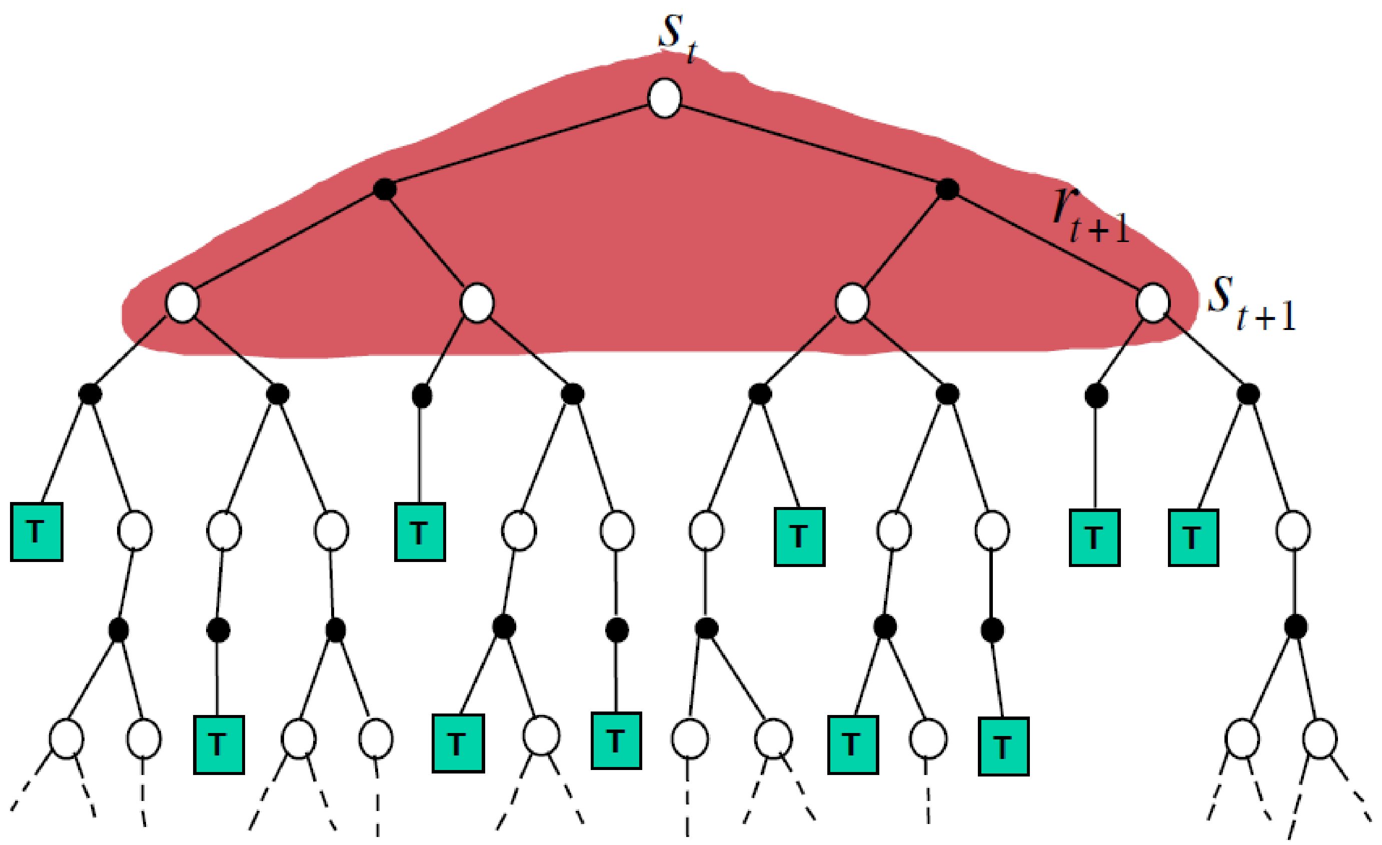
Deep Learning, Reinforcement Learning: Function Approximation





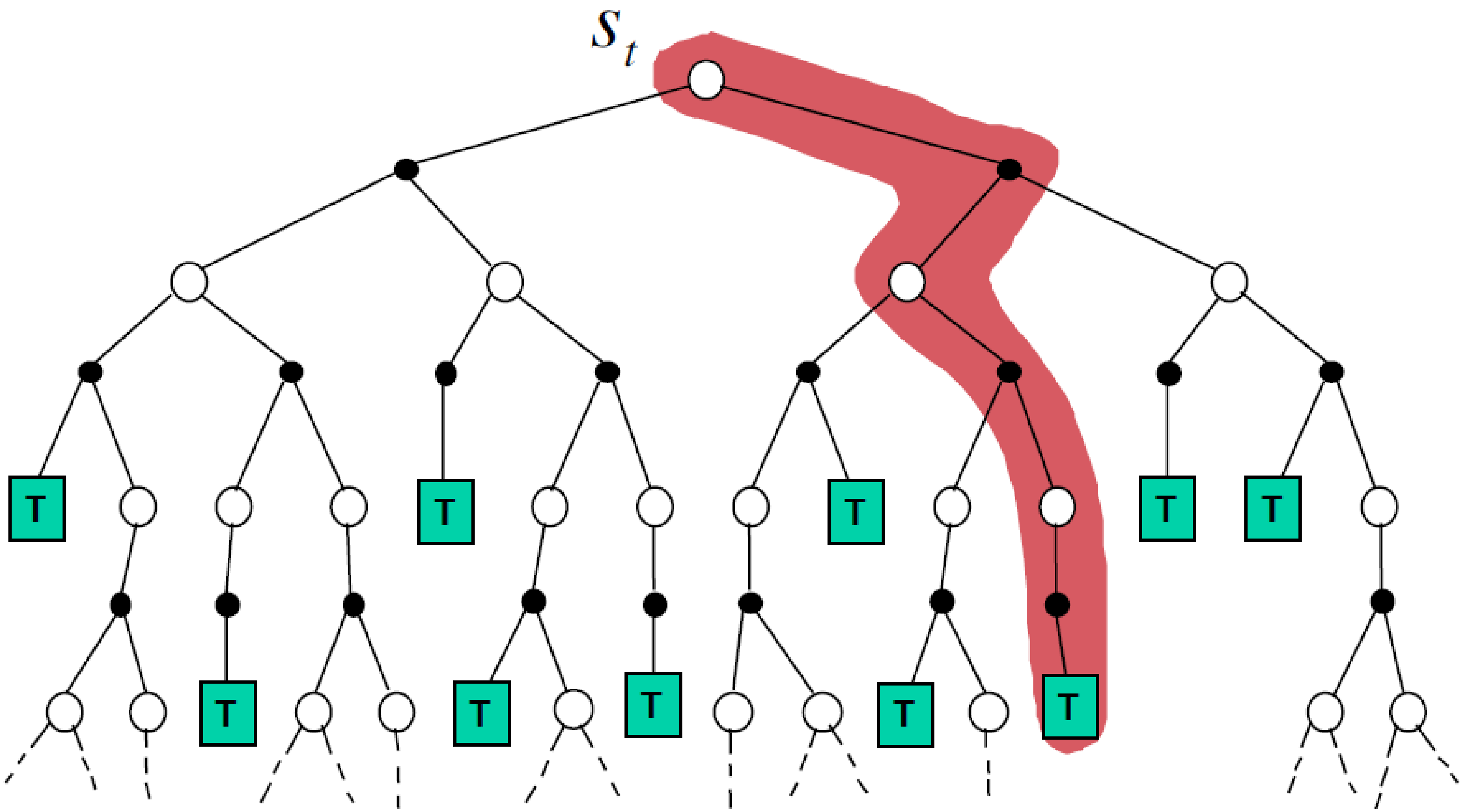
Deep Q Networks: Dynamic programming

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



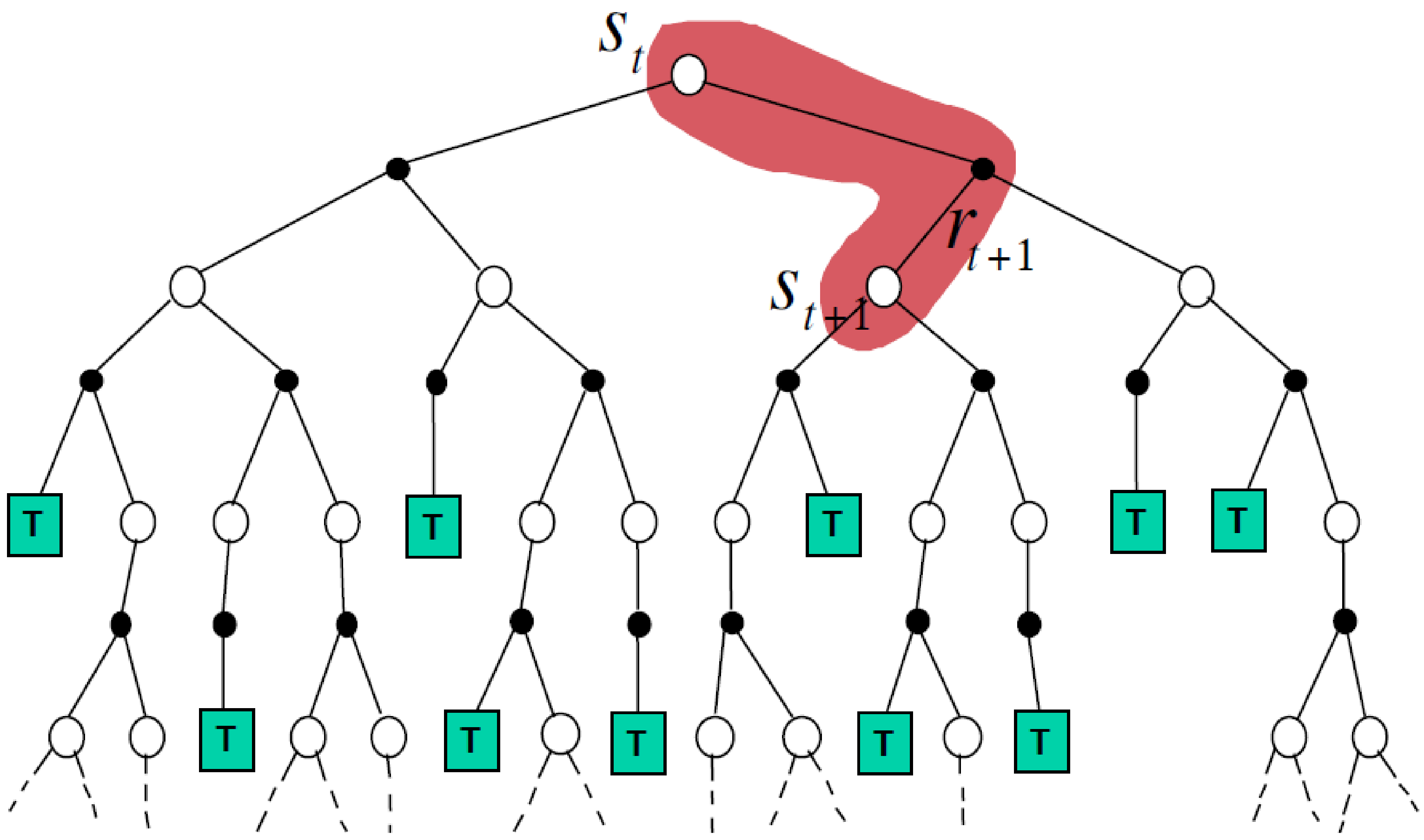
Deep Q Networks: Monte-Carlo

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



Deep Q Networks: Temporal Difference

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

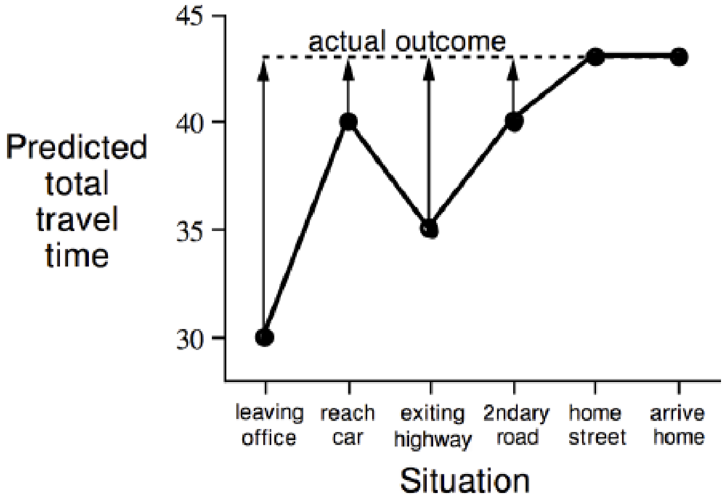


DQN: MC-TD example

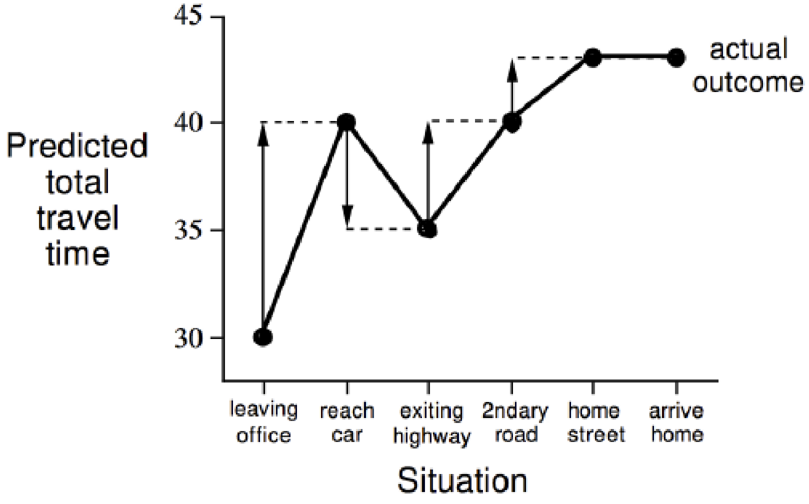
State	Elapsed Time (minutes)	Predicted Time to Go	Predicted Total Time
leaving office	0	30	30
reach car, raining	5	35	40
exit highway	20	15	35
behind truck	30	10	40
home street	40	3	43
arrive home	43	0	43

DQN: MC-TD example

Changes recommended by Monte Carlo methods (α=1)



Changes recommended by TD methods (α=1)



Purpose of Reinforcement Learning

Define a partial ordering over policies

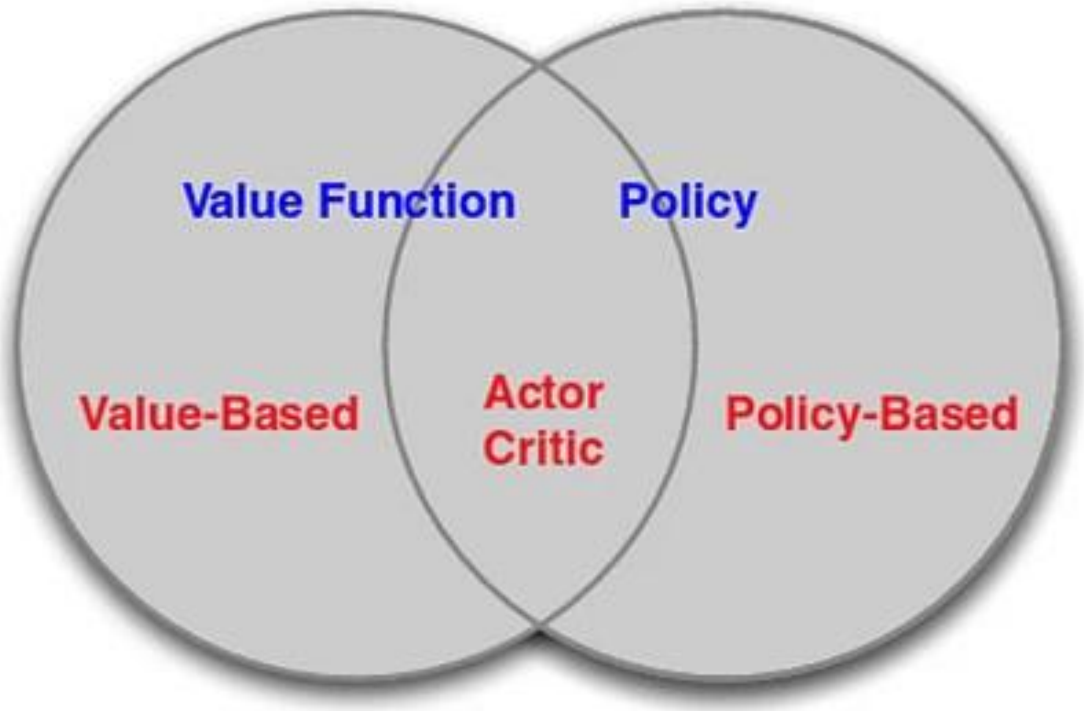
$$\pi \geq \pi' \text{ if } v_\pi(s) \geq v_{\pi'}(s), \forall s$$

Theorem

For any Markov Decision Process

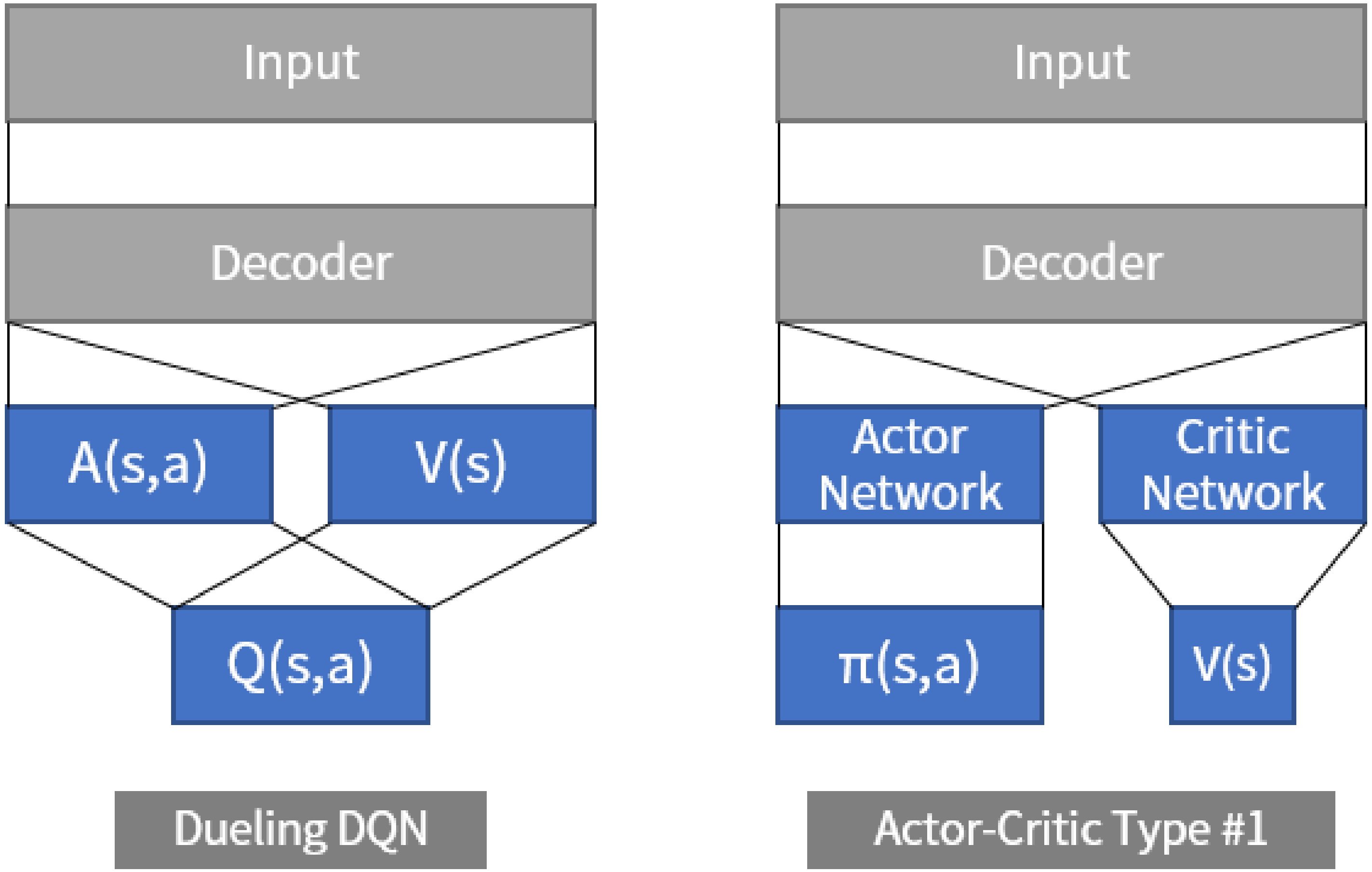
- *There exists an optimal policy* π_* *that is better than or equal to all other policies,* $\pi_* \geq \pi, \forall \pi$
- *All optimal policies achieve the optimal value function,* $v_{\pi_*}(s) = v_*(s)$
- *All optimal policies achieve the optimal action-value function,* $q_{\pi_*}(s, a) = q_*(s, a)$

Reinforcement Learning classification



- Value Function method approximates the $q_\pi(s, a)$ function using deep neural networks.
- Policy gradient method directly calculate the policy π using deep neural networks.

Reinforcement Learning classification: Actor-critic



Dynamic Replication and Hedging: A Reinforcement Learning Approach

Kolm and Ritter (2019)

Reinforcement Learning application: Problem Formulation

- Define automatic hedging to be the practice of using trained RL agents to handle hedging
- With frictions and where only discrete trading is possible the goal becomes to minimize variance and cost
- We will use this to define the reward and the state of the reinforcement learning.

Reinforcement Learning application: RL Settings

- We can seek the agent's optimal portfolio as the solution to a mean-variance optimization problem with risk-aversion κ

$$\max\left(\mathbb{E}[w_T] - \frac{\kappa}{2}\mathbb{V}[w_T]\right)$$

- where the final wealth w_T is the sum of individual wealth

Reinforcement Learning application: RL Settings

- We choose the reward in each period to

$$R_t := \delta w_t - \frac{\kappa}{2}(\delta w_t)^2$$

- Thus, training reinforcement learners with this kind of reward function amounts to training automatic hedgers who tradeoff costs versus hedging variance

Deep Learning and Reinforcement Learning

- For European options, the state must minimally contain
  - (1) the current price of the underlying, S_t
  - (2) the time remaining to expiry, $\tau := T - t > 0$
  - (3) our number of shares n.

- The state is thus naturally an element of

$$\mathcal{S} := \mathbb{R}_+^2 \times \mathbb{Z} = \{(S, \tau, n) \mid S > 0, \tau > 0, n \in \mathbb{Z}\}.$$

Deep Learning and Reinforcement Learning

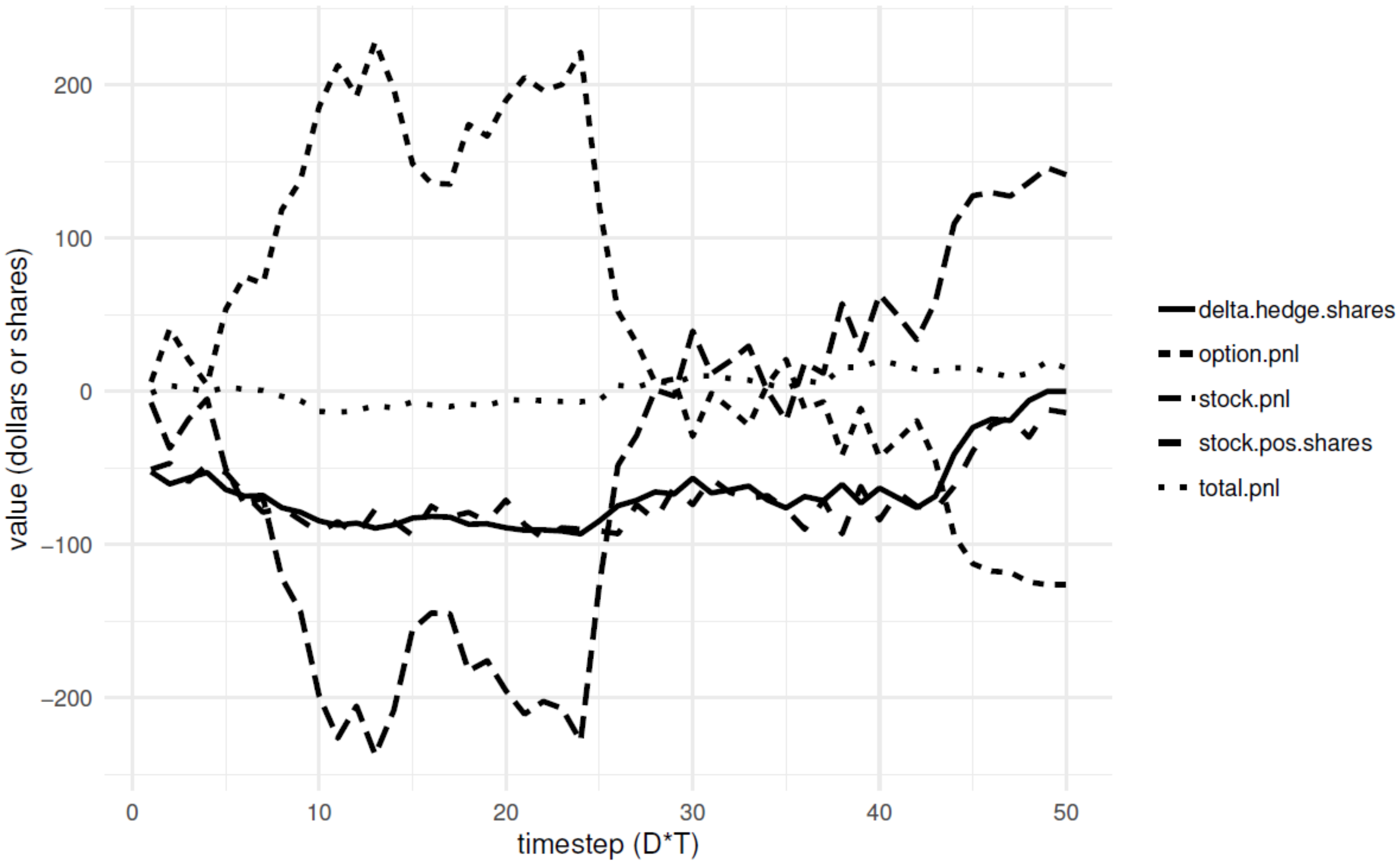


Figure 1: Stock & options P&L roughly cancel to give the (relatively low variance) total P&L. The agent's position tracks the delta

Deep Learning and Reinforcement Learning: Disadvantage

- The RL agent is at a disadvantage: It does not know any of the following information:
  - the strike price K
  - that the stock price process is a geometric Brownian motion
  - the volatility of the price process
  - the BSM formula
  - the payoff function at maturity
  - any of the Greeks
- Thus, it must infer the relevant information, insofar as it affects the value function, by interacting with a simulated environment

Concluding Remarks

- Reinforcement Learning is much more difficult than conventional supervised learning
- Careful settings for reward and environment is crucial for convergence of RL
- Can be applied to problems that needs to make a sequence of decisions
- Can observe feedback to state or choice of actions and this information can be partial and noisy