

From Code to Chip

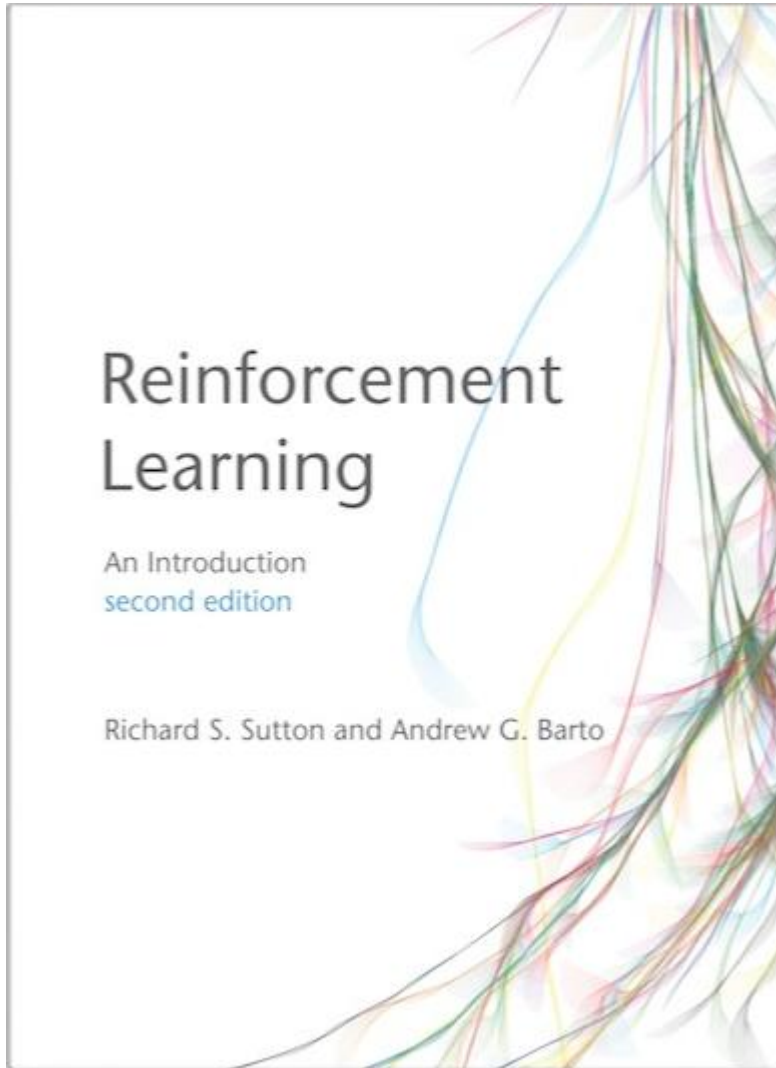
Open-Source Automated Analog Layout Design

Chapter 2: Theoretical Basics

目次

1. VLSIにおける配置配線手法についての紹介
2. 強化学習の基礎
3. 強化学習における最適化方法

強化学習のおすすめ参考書



Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. Vol. 1. No. 1. Cambridge: MIT, Second Edition, 2018

VLSIにおける配置配線手法

- 本書に述べる配置配線の最適化方法

- セルの配線

- A* algorithm

- セルの配置

- 強化学習
 - Sequence-Pair表現方式^[1]によるSimulated Annealingアルゴリズム

[1] Murata, Hiroshi, et al. "VLSI module placement based on rectangle-packing by the sequence-pair." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 15.12 (1996): 1518-1524.

Simulated Annealing, placement

Algorithm 2.2: Simulated annealing algorithm

```
1  $S \leftarrow S_0; T \leftarrow T_0; L \leftarrow L_0$ 
2 while stop criterion not fulfilled do
3   for  $L$  iterations do
4     Generate new state  $S'$  from  $S$ 
5     if  $E(S') \leq E(S)$  then  $S \leftarrow S'$ 
6     else if  $\exp\left(-\frac{E(S') - E(S)}{T}\right) > \text{random}[0, 1)$  then  $S \leftarrow S'$ 
7   end
8   Calculate next  $T$  and  $L$ 
9 end
```

セル配置のアプローチ - sequence pair

- 配置 (placement) とは: セル/ブロックの形状と座標を決定すること

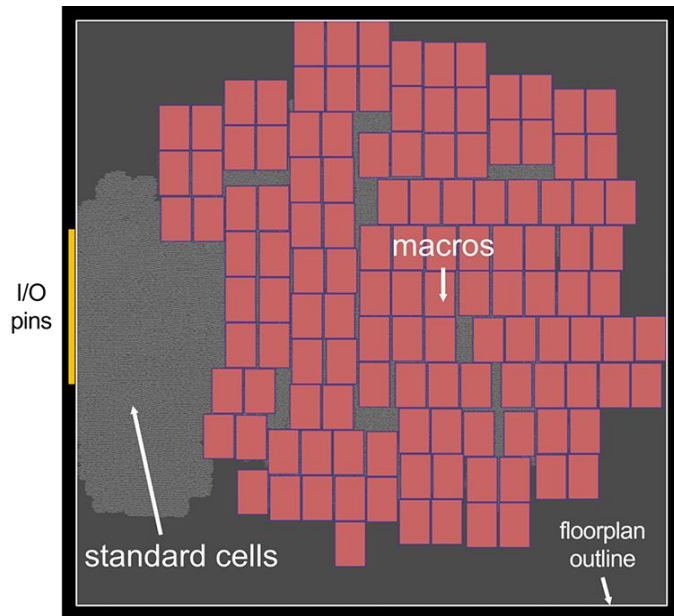
- フロアプラン (floorplan) とは:

- 大まかな配置を決定
- 部分回路の領域を見積もり、チップ内に割り当てること

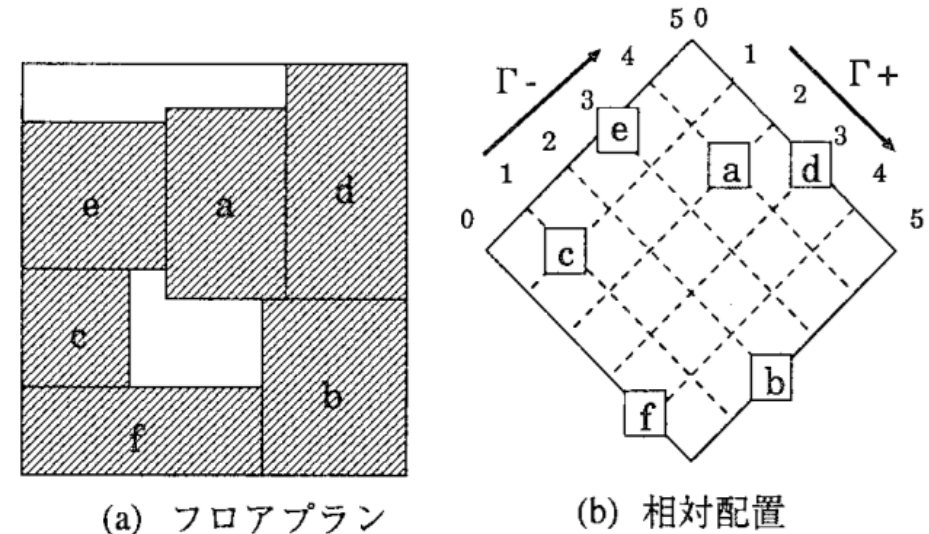
- 配置の表現方法:

- 絶対座標による方法
- 相対位置関係による方法: sequence-pair [2]
- チップ全体の分割方法

フロアプランの例 [1]:



六つのセル配置はsequence-pairによる表現の例 [3]:



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
f	c	b	e	a	d	e	c	a	d	f	b	a	b	c	d	e	f
Γ^-						Γ^+						向き					

(c) 個体表現

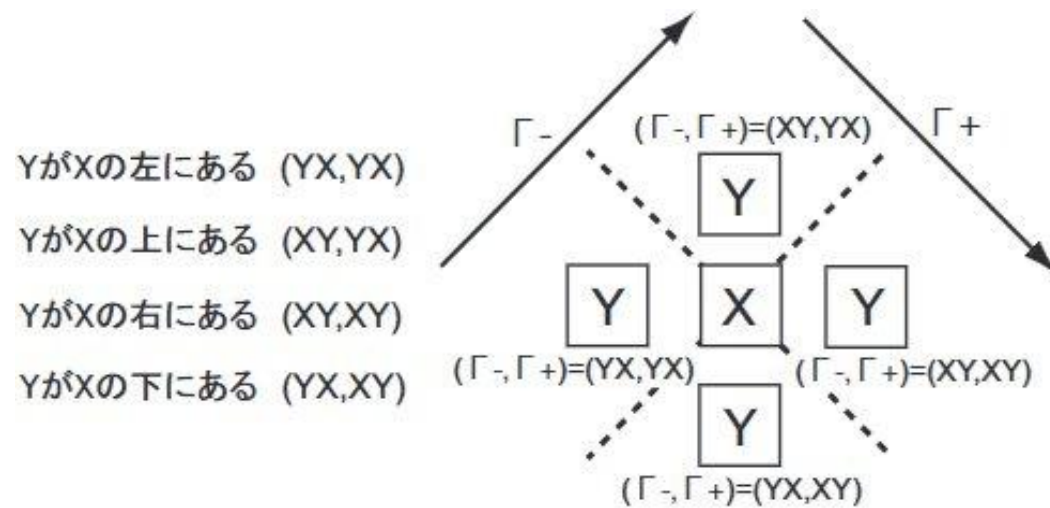
[1] Agnesina, Anthony, et al. "AutoDMP: Automated DREAMPlace-based Macro Placement." *Proceedings of the 2023 International Symposium on Physical Design*. 2023.

[2] Murata, Hiroshi, et al. "VLSI module placement based on rectangle-packing by the sequence-pair." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 15.12 (1996).

[3] Nakaya, Shingo, et al. "An adaptive genetic algorithm for VLSI floorplanning based on sequence-pair." *2000 IEEE International Symposium on Circuits and Systems (ISCAS)*. Vol. 3. IEEE, 2000.

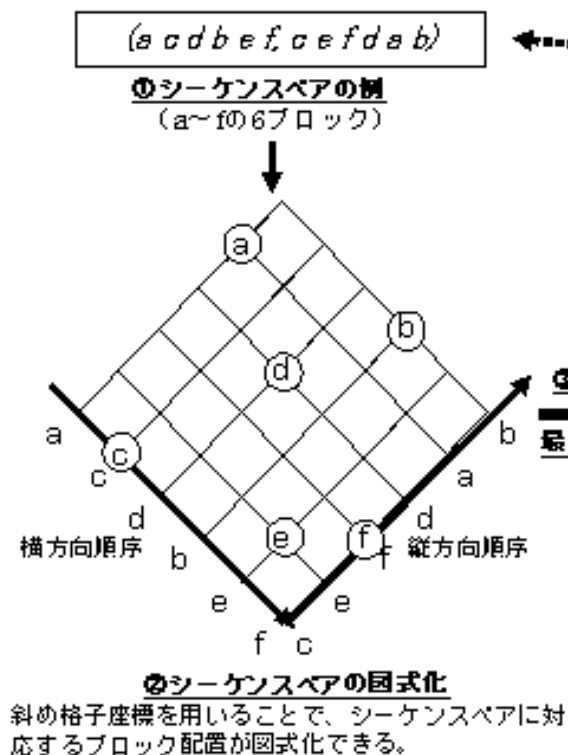
セル配置のアプローチ - Sequence-Pair

任意の二つのセルXとYが配置後の上下左右の相対位置を **sequence-pair**によって表現 (credit to: [Shingo Nakaya, et al.](#)):



セル配置のアプローチ - Sequence-Pair

Sequence-pairを用いるブロック配置方法:



⑤配置結果が棄ければ、別のシーケンスペアで再度検討

④配列最適化

YがXの左にある (YX,YX)
YがXの上にある (XY,YX)
YがXの右にある (XY,XY)
YがXの下にある (YX,XY)

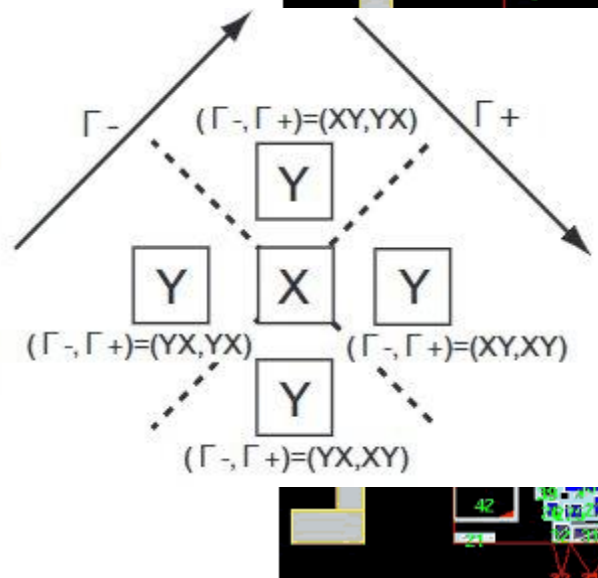
Sequence-pairを用いるブロック最適化の例:

①最適化初期



ブロック配置)

(シーケンスペアによる概略配置)



Credit to: [Kajitani Yoji](#)

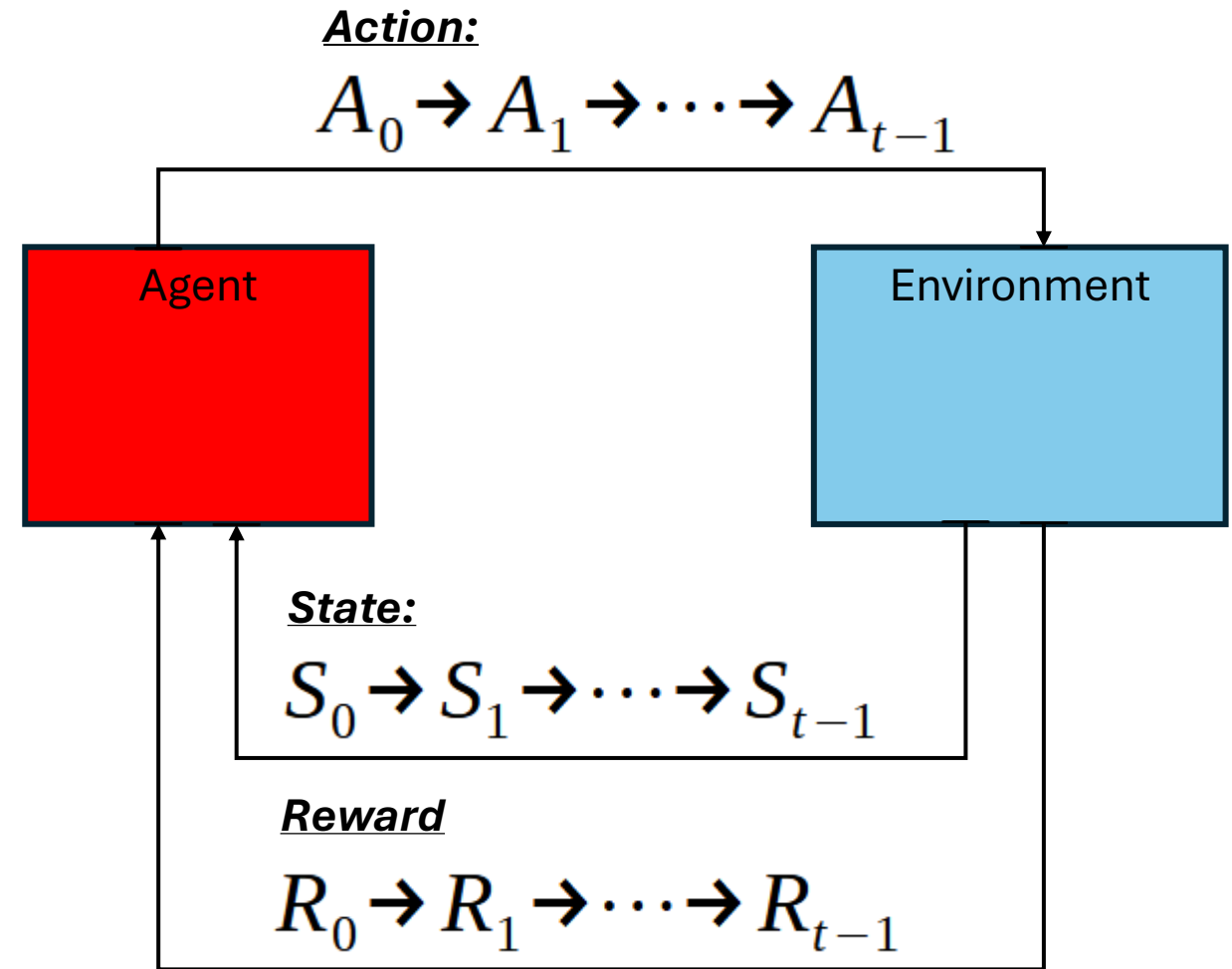
Reinforcement Learning – Outline

- 強化学習の構成要素

- 環境 (environment)
- エージェント (agent)

- 主なパラメーターの調節を通じて学習

- 行動 (action)
- 状態 (state)
- 報酬 (reward)
- 方策 (policy)
- 累積報酬 (cumulative reward)



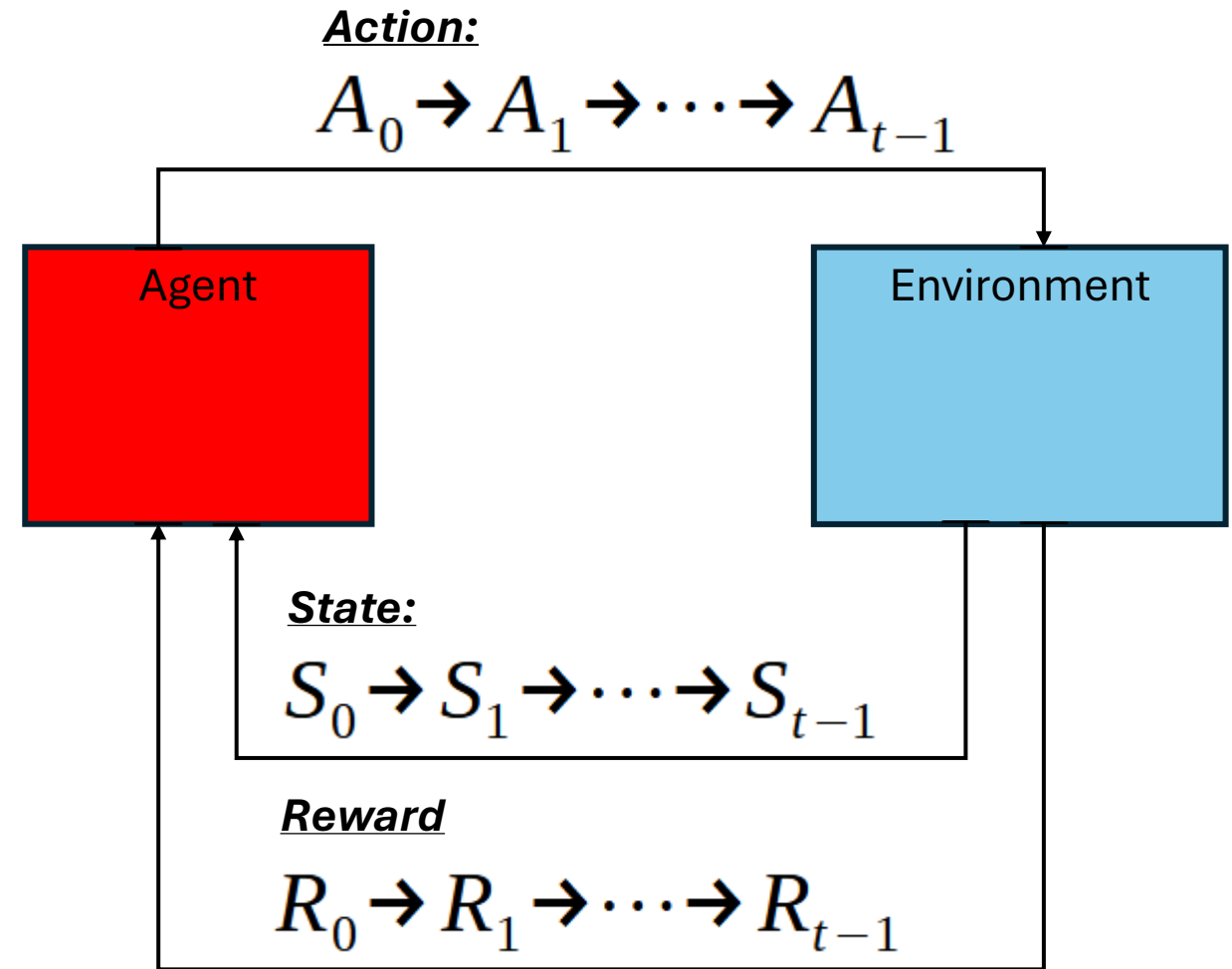
Reinforcement Learning – Outline

- 強化学習の構成要素

- 環境 (environment)
- エージェント (agent)

- 主なパラメーターの調節を通じて学習

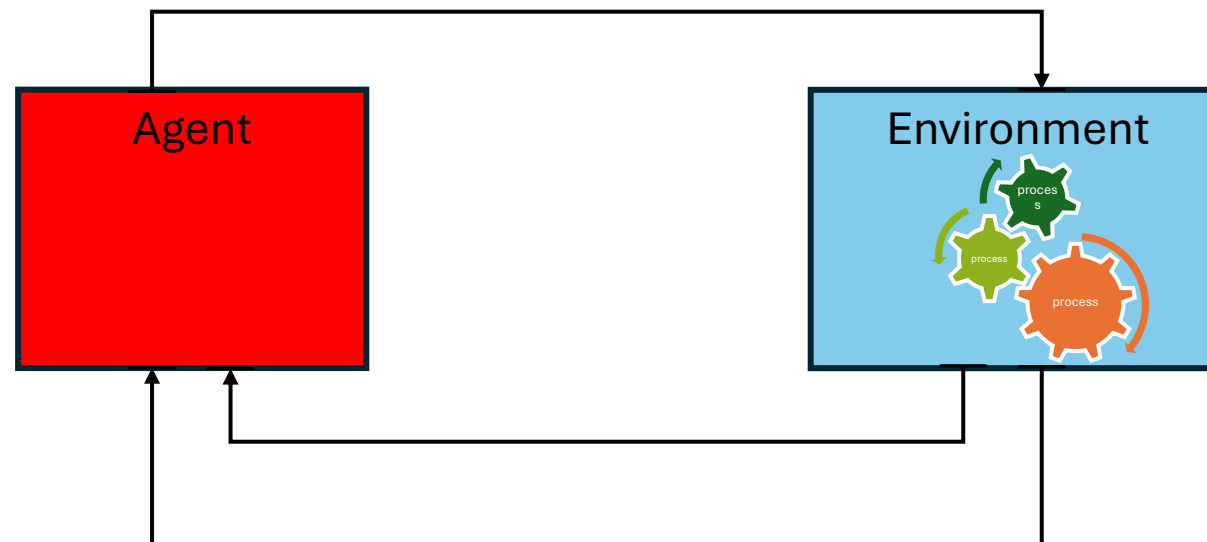
- 行動 (action)
- 状態 (state)
- 報酬 (reward)
- 方策 (policy)
- 累積報酬 (cumulative reward)



Reinforcement Learning – Environment (CONT'D)

環境の定義::

An external system/model/simulation the agent interacts with.

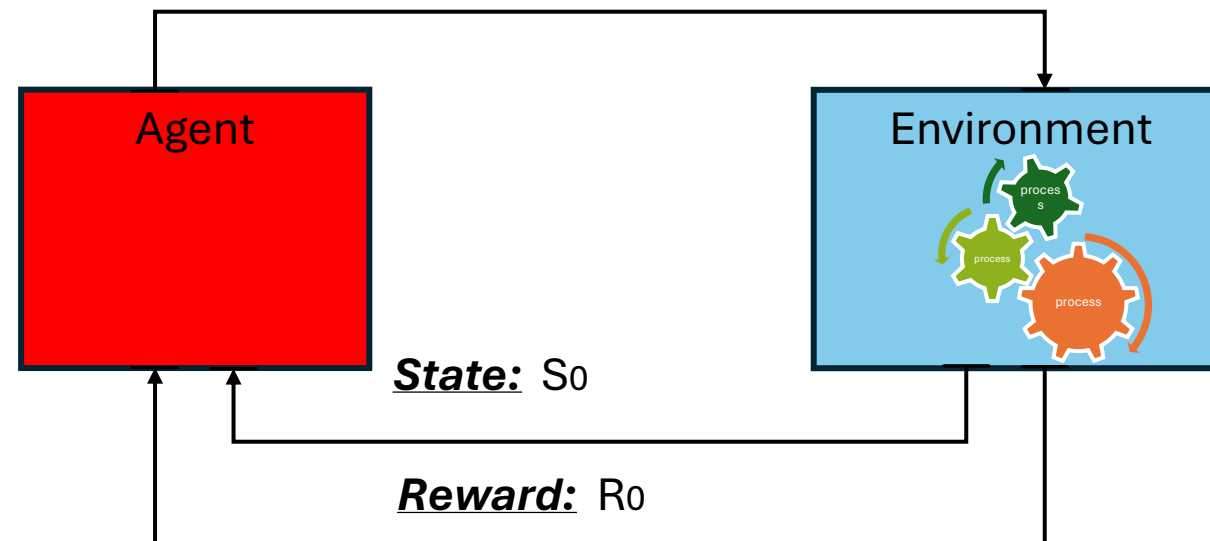


Reinforcement Learning – Environment (CONT'D)

Definition of environment::

An external system/model/simulation the agent interacts with. Their interaction is demonstrated as follows:

- ① **Environment** provides an initial state to the **agent**

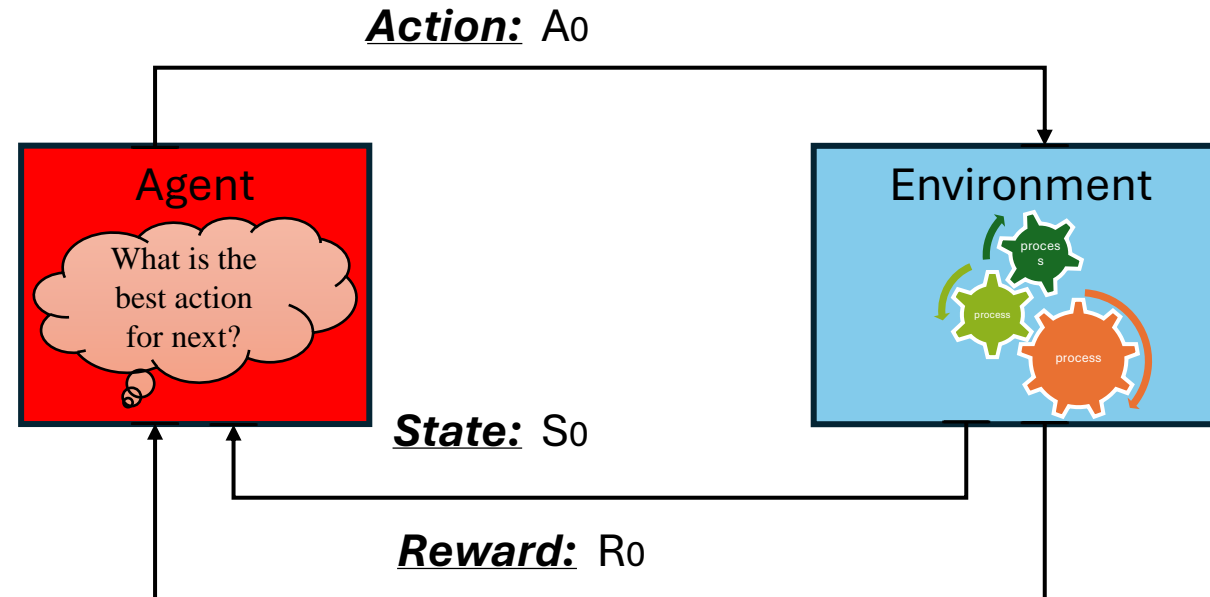


Reinforcement Learning – Environment (CONT'D)

Definition of environment::

An external system/model/simulation the agent interacts with. Their interaction is demonstrated as follows:

- ① **Environment** provides an initial state to the **agent**
- ② **Agent** determines an action and feed it to the **environment**

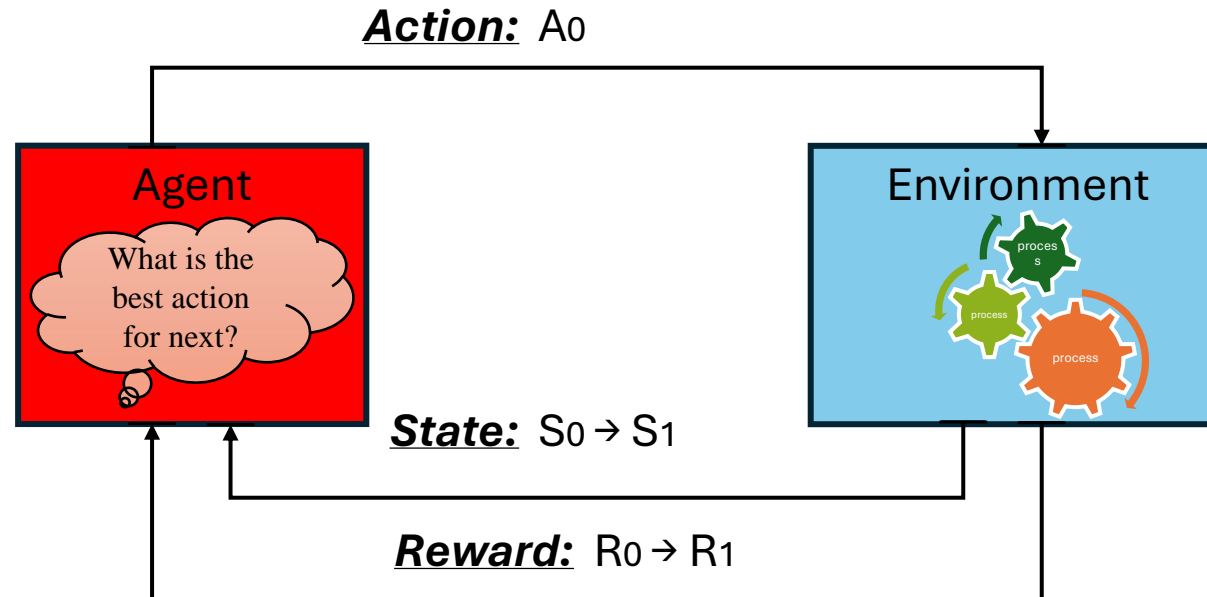


Reinforcement Learning – Environment

Definition of environment::

An external system/model/simulation the agent interacts with. Their interaction is demonstrated as follows:

- ① **Environment** provides an initial state to the **agent**
- ② **Agent** determines an action and feed it to the **environment**
- ③ **Environment** responds to the **agent** by giving a state and reward



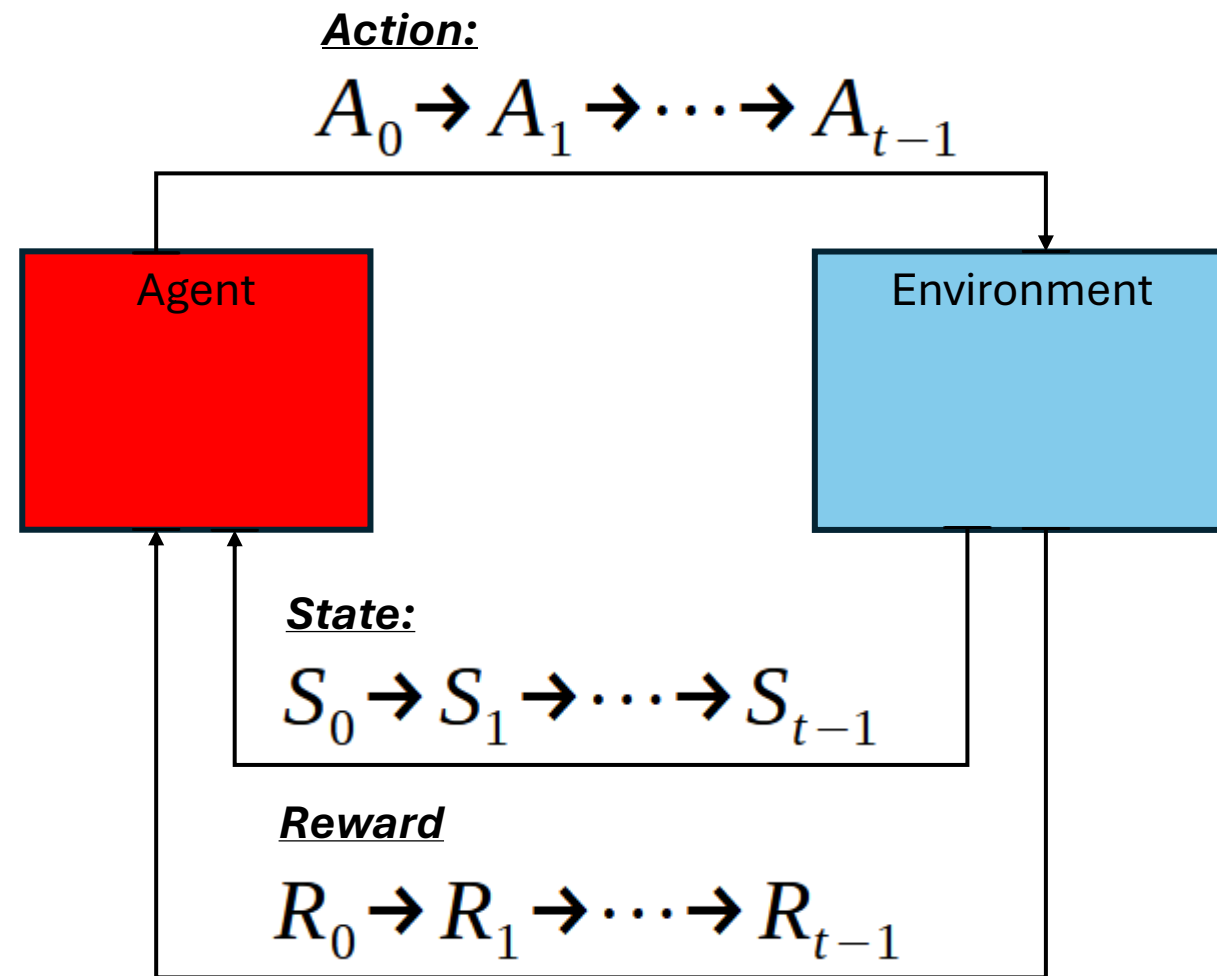
Reinforcement Learning – Agent (CONT'D)

- 強化学習の構成要素

- 環境 (environment)
- エージェント (agent)

- 主なパラメーターの調節を通じて学習

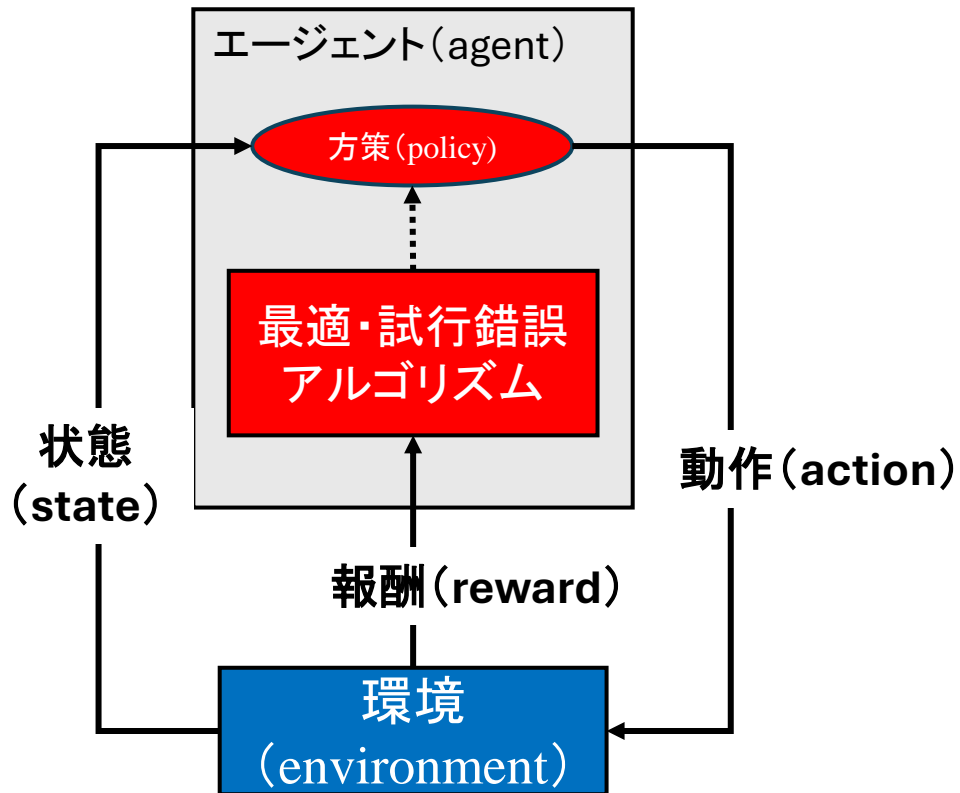
- 行動 (action)
- 状態 (state)
- 報酬 (reward)
- 方策 (policy)
- 累積報酬 (cumulative reward)



Reinforcement Learning – Agent (CONT'D)

Definition of agent:

A decision maker or learner determining the best action for the environment to behave. Mathematically, its objective is to maximise cumulative rewards by learning from a sequence of state-action pairs.



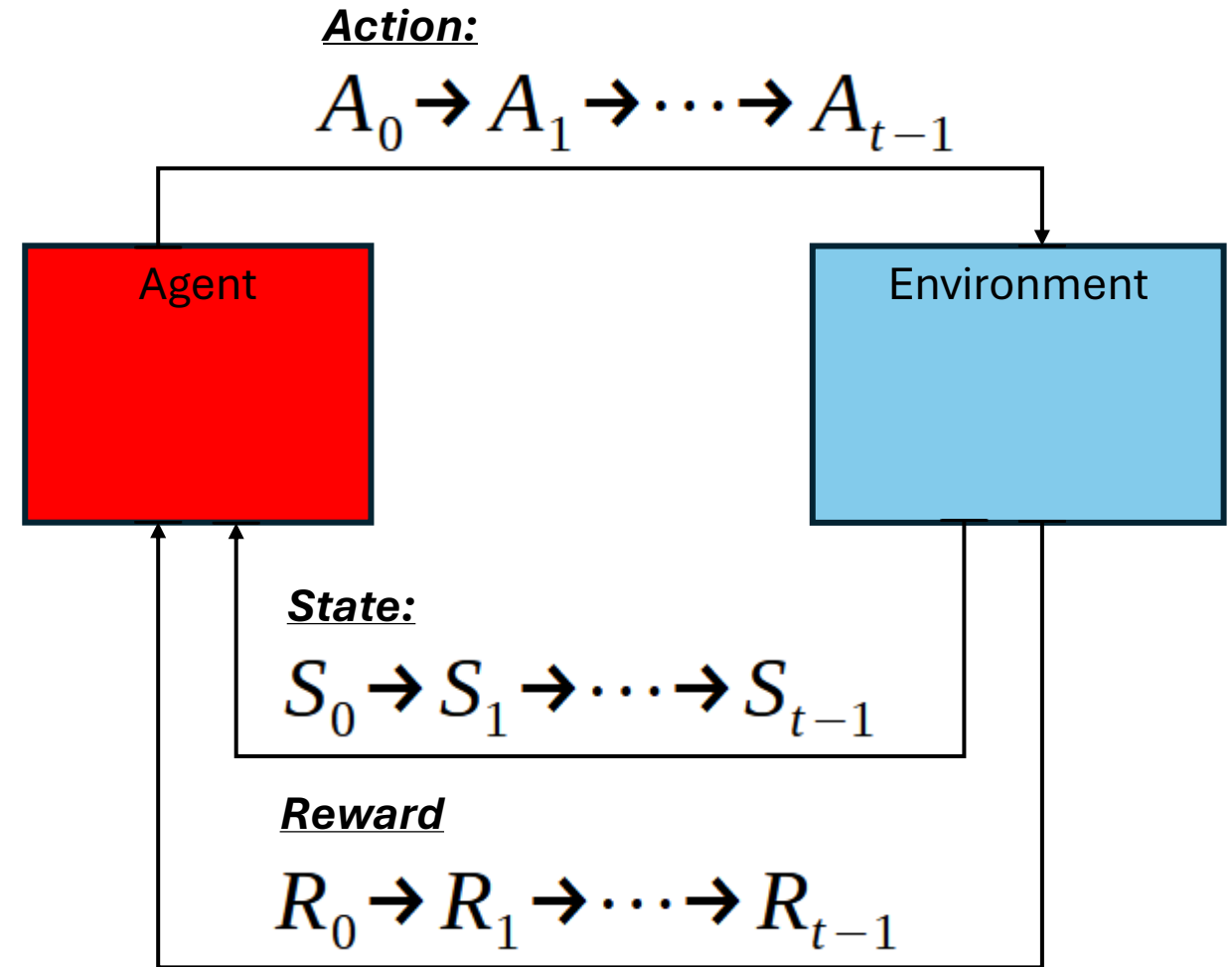
Reinforcement Learning – Agent (CONT'D)

- 強化学習の構成要素

- 環境 (environment)
- エージェント (agent)

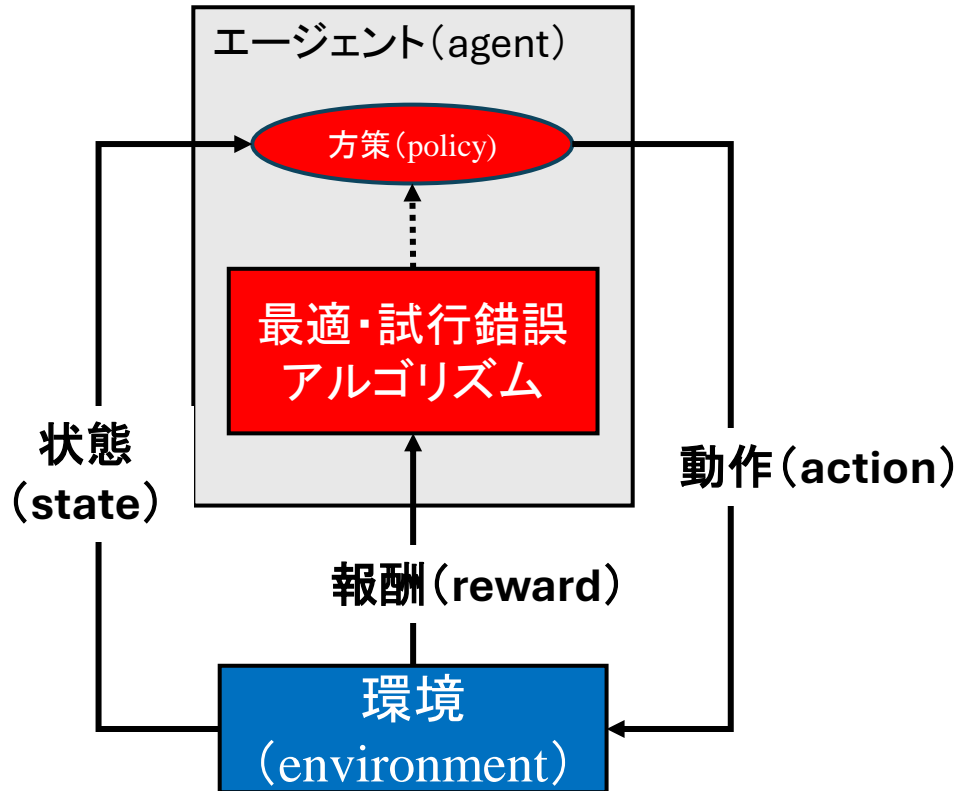
- 主なパラメーターの調節を通じて学習

- 行動 (action)
- 状態 (state)
- 報酬 (reward)
- 方策 (policy)
- 累積報酬 (cumulative reward)



強化学習の基礎 - パラメータ・用語のまとめ

- **行動** (*action*), A_t - Decision made by the agent that affects the state transition
- **状態** (*state*), S_t - Current situation of the agent in the environment
- **方策** (*policy*), π - 与えられた状態のもとで行動を決定するルール
- **Reward**, R_t - Feedback given by the environment based on the agent's action
- **Return** (or called *cumulative reward*), G_t - Rewards accumulate over a sequence of states



強化学習の基礎 – マルコフ決定過程

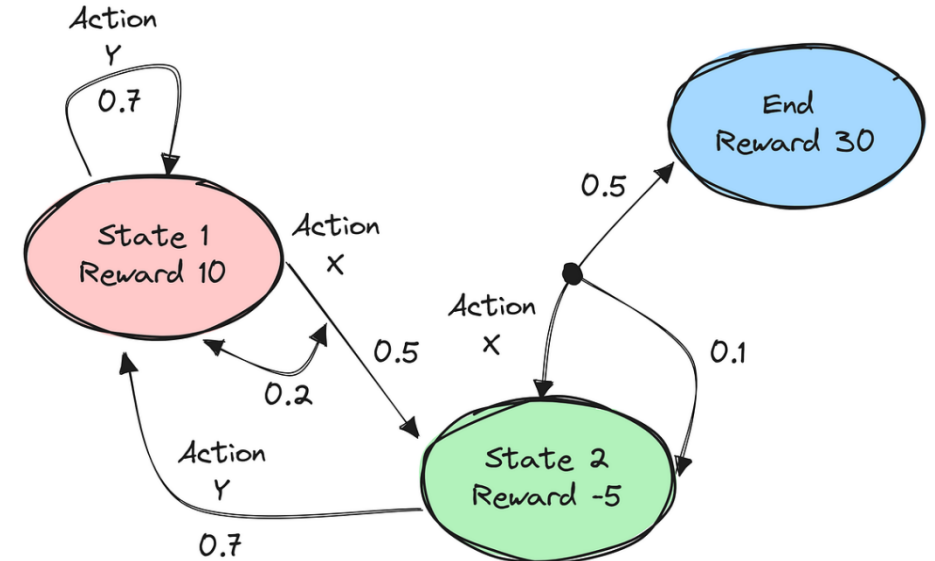
マルコフ決定過程 (Markov Decision Process, MDP) とは

強化学習の行動 (***action***) に対する状態 (***state***)、報酬 (***reward***)、状態遷移の確率などの要素を表す数学的なモデルのこと。

- *Action, At*
- *State, Reward, Rt*
- *Return, Gt*
- *Prob., $p(S_{t+1}, R_{t+1} / S_t, A_t)$*

モデル化

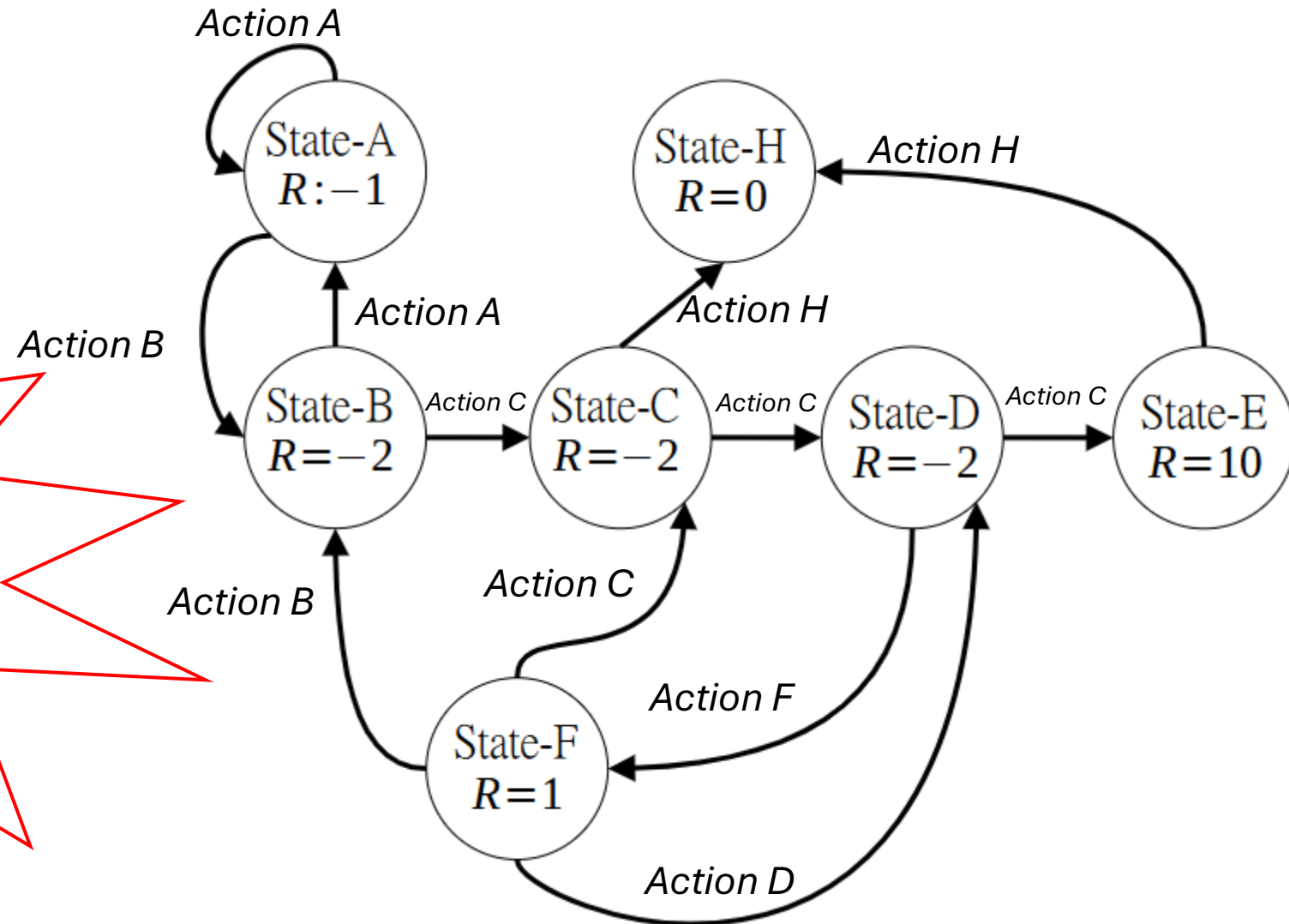
MDP example:



Credit to: [Rafał Buczyński](#)

強化学習の基礎 – マルコフ決定過程

- 例: 七つの状態 (state)が存在する環境 (environment)を表す状態遷移図

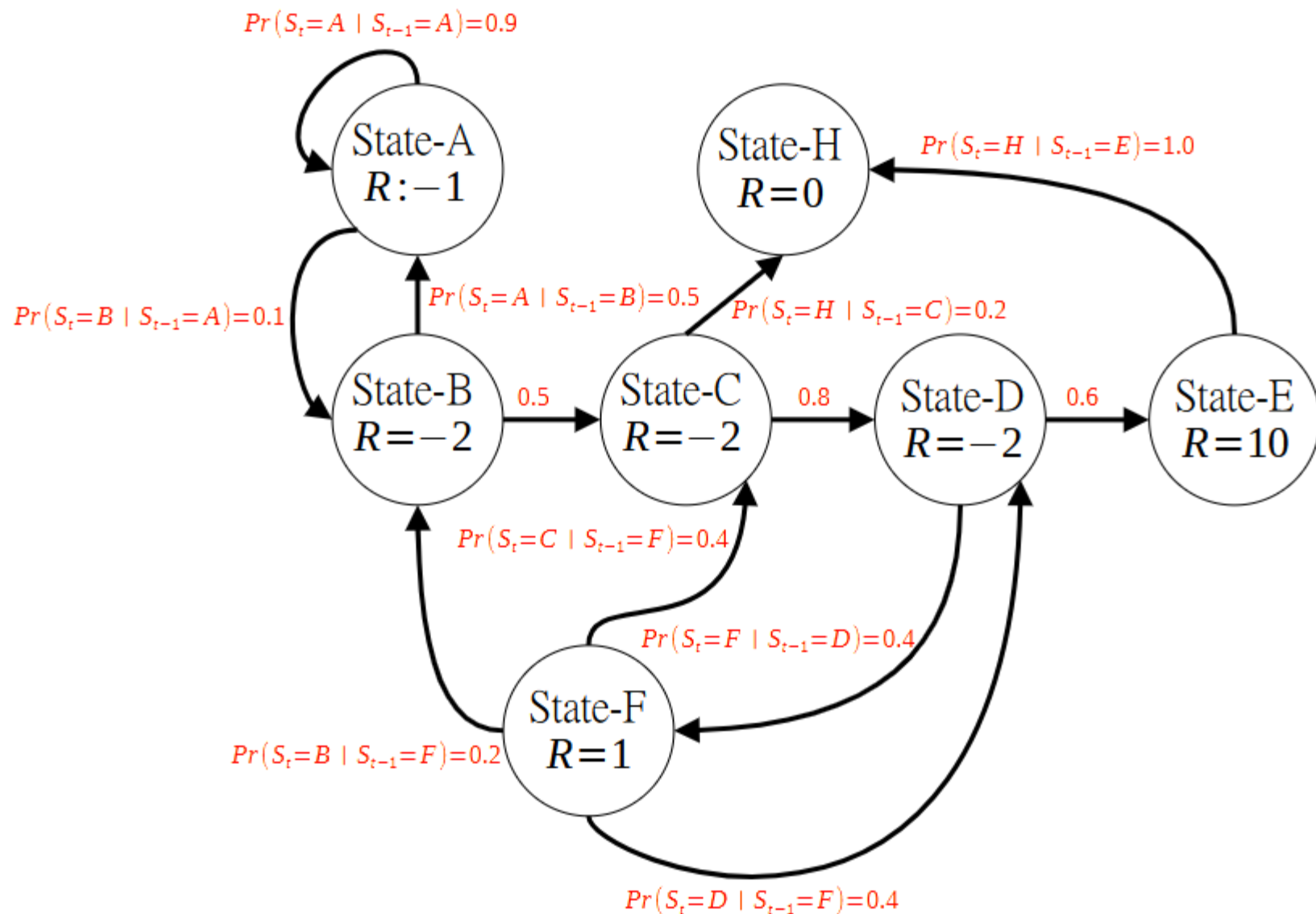


次の状態に推移する確率が過去の状態に依存せず、現在の状態のみから予測する場合、マルコフ性 (Markov property) を持つ

強化学習の基礎 – マルコフ決定過程

- 各状態から次の状態へ遷移する確率

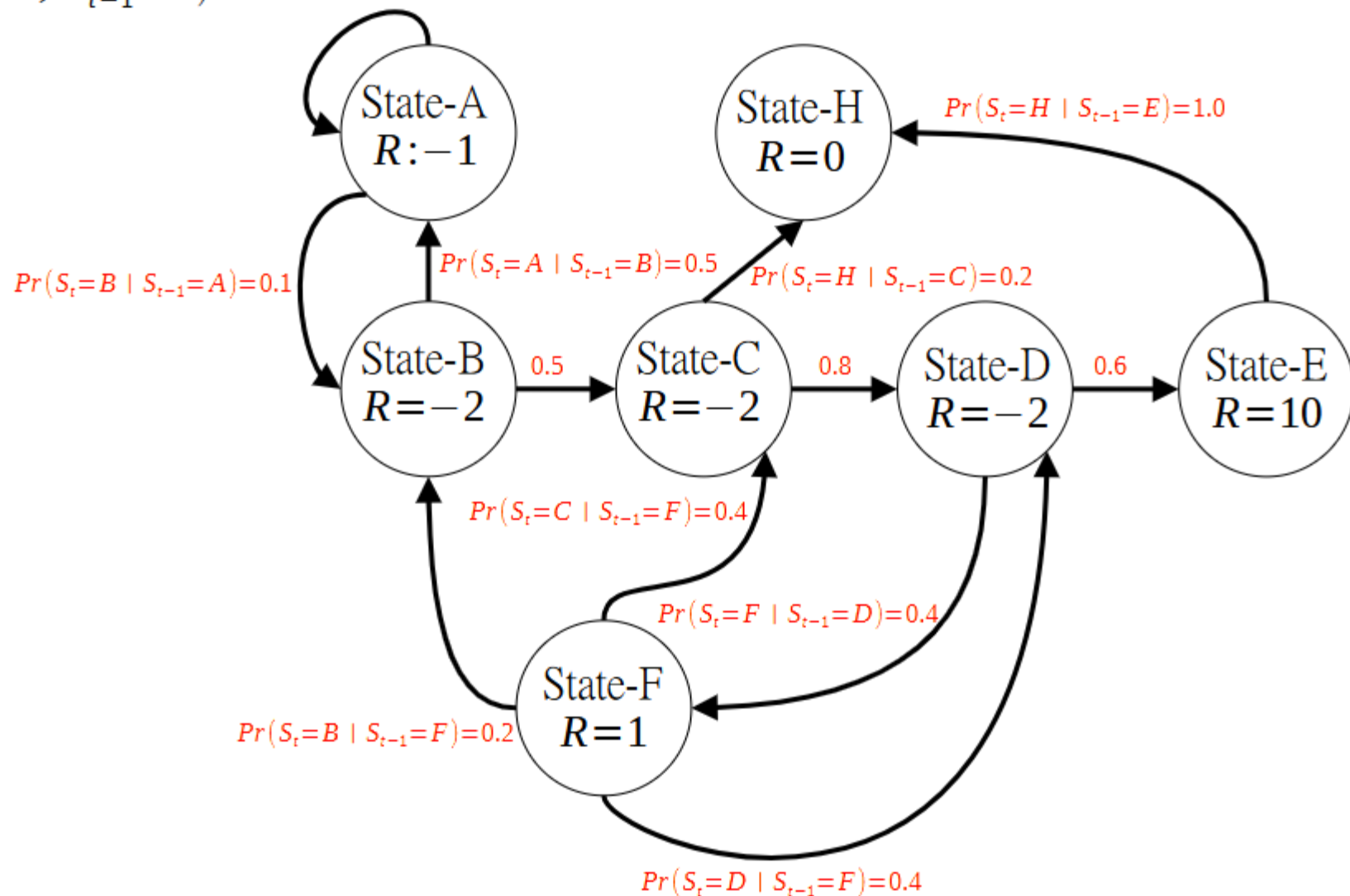
$$p(s'|s) = \Pr(S_t = s' | S_{t-1} = s)$$



強化学習の基礎 – マルコフ決定過程

- 各状態から次の状態へ遷移する確率

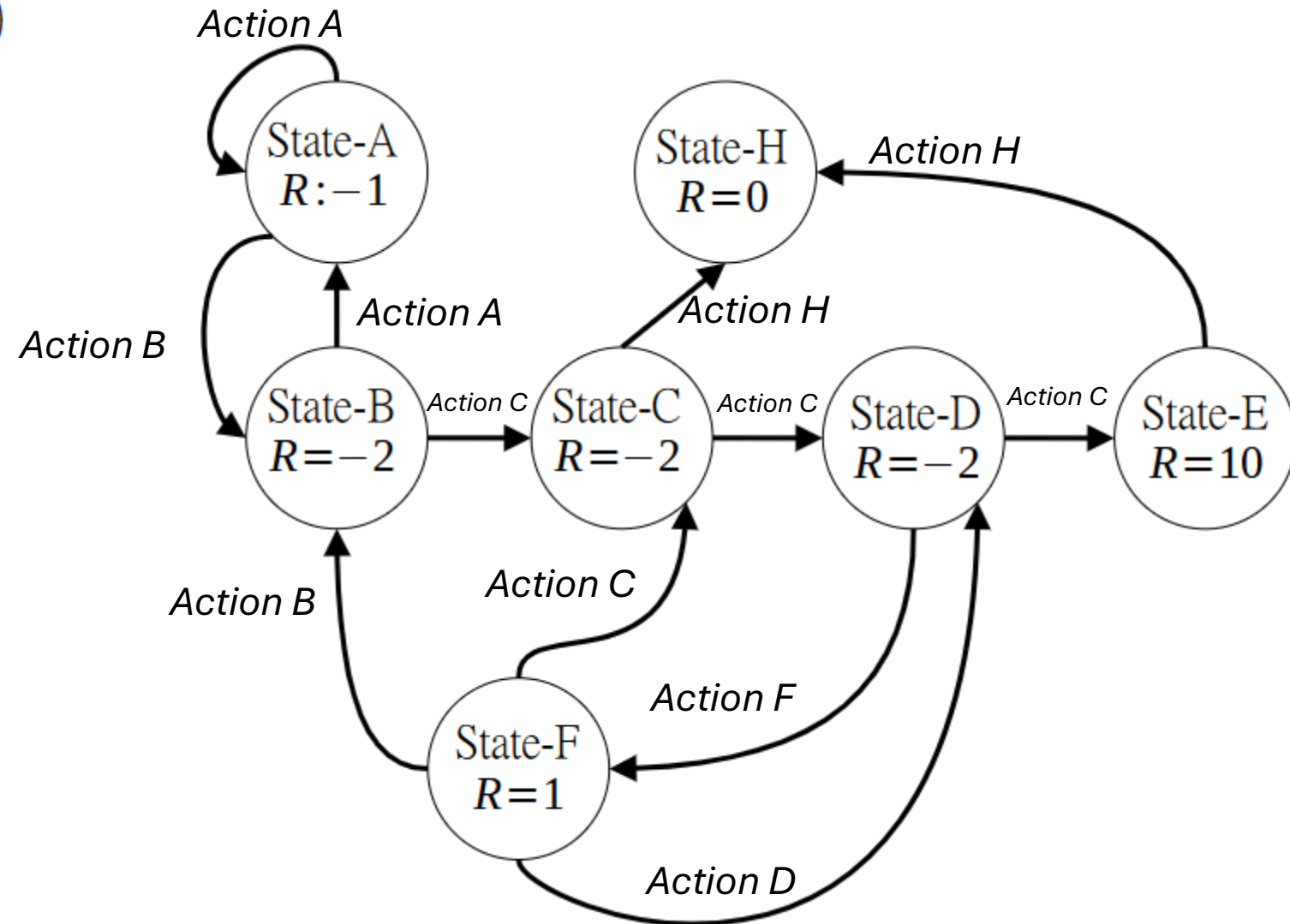
$$p(s', r | s, a) = \Pr(S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a) \quad \Pr(S_t = A | S_{t-1} = A) = 0.9$$



強化学習の基礎 – マルコフ決定過程

- Trajectory (called episodes or rollouts in other references), e.g.

$$\tau = (R_0, S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \dots)$$



強化学習の基礎 – マルコフ決定過程

- Discounted Return – a sum of received rewards over a given trajectory

$$\gamma \in [0, 1]$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

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

$$= R_{t+1} + \gamma G_{t+1}$$

割引率が高いときは遠い状態 (state) の報酬まで考える方策を、小さいときには直近の状態までの報酬しか考えない方策を学習する

Definition of γ

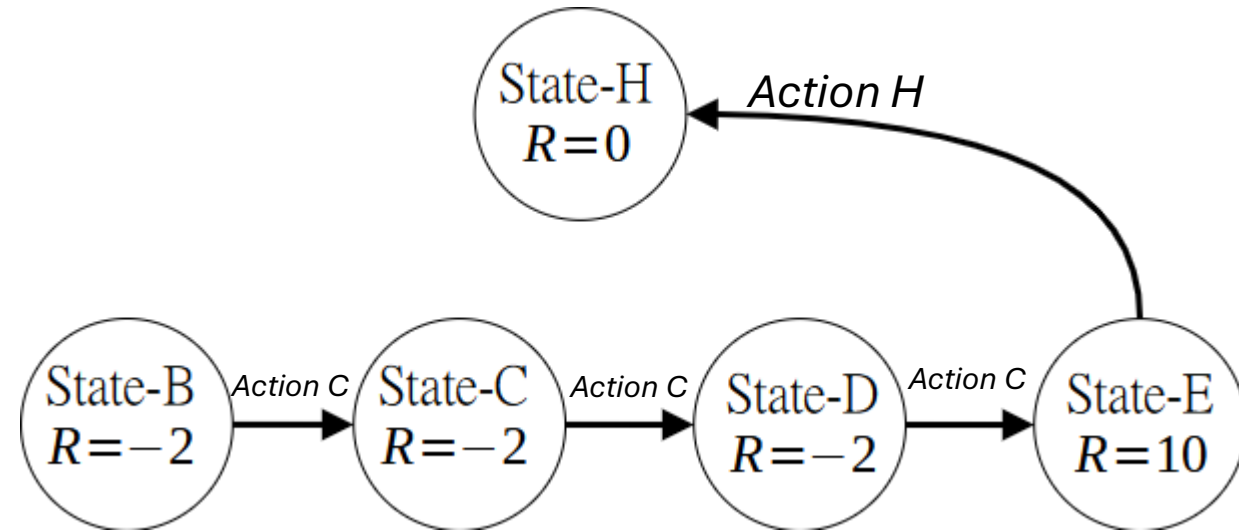
A discount factor determining how much future rewards are valued compared to immediate rewards

The goal is to maximise the G_t

強化学習の基礎 – マルコフ決定過程

- A sample trajectory

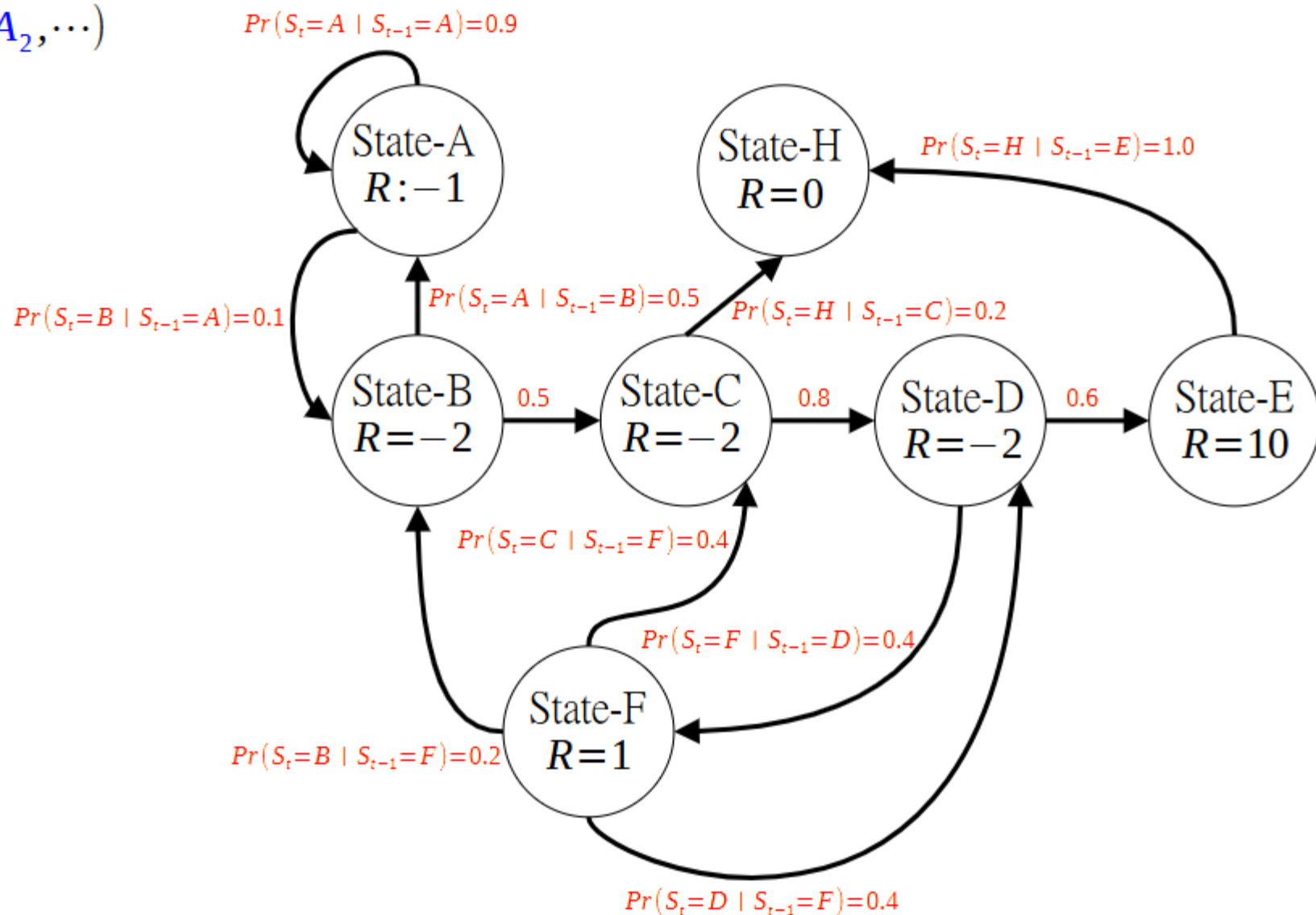
$$\tau = (S_0=B, A_0=C, R_1=-2, S_1=C, A_1=C, R_2=-2, S_2=D, A_2=C, R_3=10, S_3=E, A_3=H)$$



強化学習の基礎 – マルコフ決定過程

- Trajectory (called episodes or rollouts in literatures), e.g.

$$\tau = (R_0, S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \dots)$$



強化学習 – 価値関数と方策

価値関数 (value function) とは

ある 状態 (state) までにわたって獲得できる累積報酬 (cumulative rewards) の期待値。つまり、現在の状態と行動がどのくらいよいのかを計る関数。

• ある 状態 s で 行動 a をとることであり、 $\pi(s, a)$ や $\pi(a|s)$ と表す方策 π のもとで価値関数を以下のように定式化できる

• **状態価値関数 (state-value function for policy π)** $v_\pi(s)$: 方策 π のもとで、取った 行動 (action) と関係なく初期状態を s とした場合の累積報酬の期待値

$$v_\pi(s) = E_\pi[G_t \mid S_t = s] = E_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

• **行動価値関数 (action-value function for policy π)** $q_\pi(s, a)$: 方策 π のもとで、初期 状態 s において 行動 (action) a を取った場合の累積報酬の期待値

$$q_\pi(s, a) = E_\pi[G_t \mid S_t = s, A_t = a] = E_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a]$$

強化学習 – 価値関数と方策

強化学習の目的:

最適な累積報酬を獲得するために、価値関数の結果を最大にするような方策(policy)を求めること。

- 最適（最大化）な状態価値関数を得られる方策 π'

$$\pi'(s) = \underset{\forall \pi}{\operatorname{argmax}} v_{\pi}(s)$$

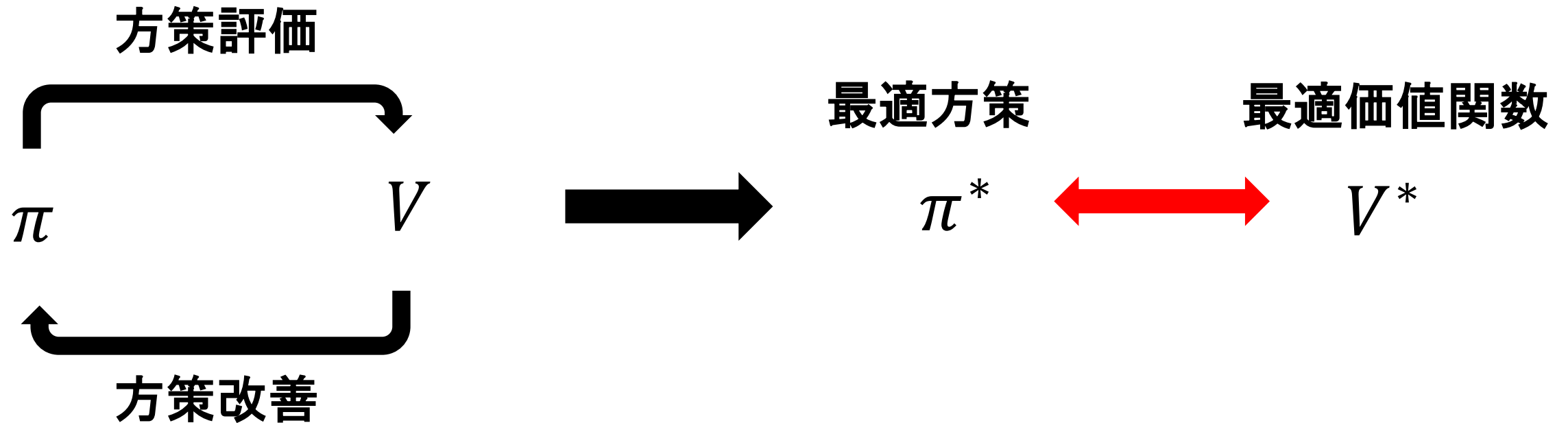
- 最適（最大化）な行動価値関数を得られる方策 π'

$$\pi'(s, a) = \underset{\forall \pi}{\operatorname{argmax}} q_{\pi}(s, a)$$

価値ベースの手法

一般化方策反復 (Generalized Policy Iteration; GPI)

方策評価と方策改善を繰り返すことで最適価値関数および最適方策を探索



価値ベースの手法

ランダムな方策 π_0

				+1
	壁			
				-1

方策評価
(価値関数推定)



方策改善
 $\pi_0 \rightarrow \pi_1$

方策 π_0 における価値関数

0.08	0.09	0.38	
0.00	壁	0.00	0.00
-0.07	-0.18	-0.38	

π_0 から改善した方策 π_1

	壁			

方策評価
(価値関数推定)



方策 π_1 における価値関数

0.81	0.90	1.00	
0.73	壁	0.90	1.00
0.66	0.59	0.81	

モンテカルロ法

最も基本的な未知の環境におけるアルゴリズム(モデルフリー)

$$v_{\pi}(s) = \mathbb{E}_{\pi}[G|s]$$

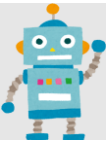
↓ n 回の試行におけるそれぞれの収益 $(G^{(1)}, G^{(2)}, \dots, G^{(n)})$
を平均することで近似

$$V_{\pi}(s) = \frac{G^{(1)} + G^{(2)} + \dots + G^{(n)}}{n}$$

モンテカルロ法


1回目の試行
(割引率: $\gamma = 1$)

S 

	$R_0 = 1$	$R_1 = 1$	$R_2 = 1$
---	-----------	-----------	-----------

$$G^{(1)} = R_0 + \gamma R_1 + \gamma^2 R_2 = 1 + 1 + 1 = 3$$

2回目の試行

	$R_0 = 2$	$R_1 = 2$	$R_2 = 2$
---	-----------	-----------	-----------

$$G^{(2)} = R_0 + \gamma R_1 + \gamma^2 R_2 = 2 + 2 + 2 = 6$$

平均すると

$$V_{\pi}(s) = \frac{G^{(1)} + G^{(2)}}{2} = \frac{3 + 6}{2} = 4.5$$

モンテカルロ法

$$V_n(s) = \frac{G^{(1)} + G^{(2)} + \dots + G^{(n)}}{n}, \quad V_{n-1}(s) = \frac{G^{(1)} + G^{(2)} + \dots + G^{(n-1)}}{n-1}$$

↓ インクリメンタルな方式

$$\begin{aligned} V_n(s) &= \frac{1}{n} \left(G^{(1)} + G^{(2)} + \dots + G^{(n-1)} + G^{(n)} \right) \\ &= \frac{1}{n} \left((n-1)V_{n-1}(s) + G^{(n)} \right) \end{aligned}$$

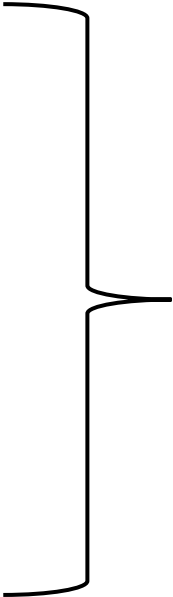
$$V_n(s) = V_{n-1}(s) + \frac{1}{n} \{ G^{(n)} - V_{n-1}(s) \} \quad \frac{1}{n} \rightarrow \alpha: \text{指数移動平均}$$

モンテカルロ法

1回目の試行 $\rightarrow G^{(1)}$

2回目の試行 $\rightarrow G^{(2)}$

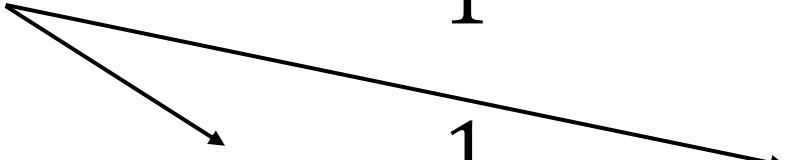
3回目の試行 $\rightarrow G^{(3)}$


$$V_3(s) = \frac{G^{(1)} + G^{(2)} + G^{(3)}}{3}$$

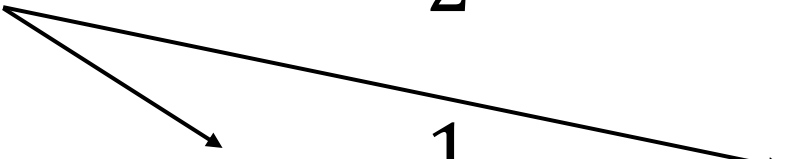
$$V_n(s) = \frac{G^{(1)} + G^{(2)} + \dots + G^{(n)}}{n},$$

モンテカルロ法

1回目の試行 → $G^{(1)}$ \longrightarrow $V_1(s) = V_0(s) + \frac{1}{1} \{G^{(1)} - V_0(s)\}$



2回目の試行 → $G^{(2)}$ \longrightarrow $V_2(s) = V_1(s) + \frac{1}{2} \{G^{(2)} - V_1(s)\}$



3回目の試行 → $G^{(3)}$ \longrightarrow $V_3(s) = V_2(s) + \frac{1}{3} \{G^{(3)} - V_2(s)\}$

$$V_n(s) = V_{n-1}(s) + \frac{1}{n} \{G^{(n)} - V_{n-1}(s)\}$$

モンテカルロ法 $V_0(s) = 0$

1回目の試行 $(R_0, R_1, R_2) = (1, 1, 1)$ $G^{(1)} = 1 + 1 + 1 = 3$

$$V_1(s) = V_0(s) + \frac{1}{1} \{3 - V_0(s)\} = 3$$

2回目の試行 $(R_0, R_1, R_2) = (2, 2, 2)$ $G^{(1)} = 2 + 2 + 2 = 6$

$$V_2(s) = V_1(s) + \frac{1}{2} \{6 - V_1(s)\} = 3 + \frac{1}{2} \{6 - 3\} = 4.5$$

TD法

モンテカルロ法

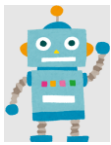
$$V_{\pi}'(S_t) = V_{\pi}(S_t) + \alpha\{\textcolor{red}{G}_t - V_{\pi}(S_t)\}$$

TD法

$$V_{\pi}'(S_t) = V_{\pi}(S_t) + \alpha\{\textcolor{red}{R}_{t+1} + \gamma V_{\pi}(\textcolor{red}{S}_{t+1}) - V_{\pi}(S_t)\}$$

$$\begin{aligned} V_{\pi}(S_t) &= \mathbb{E}_{\pi}[G_t | S_t] \\ &= \mathbb{E}_{\pi}[R_{t+1} + \gamma \textcolor{red}{G}_{t+1} | S_t] \\ &= \mathbb{E}_{\pi}[R_{t+1} + \gamma \textcolor{red}{V}_{\pi}(\textcolor{red}{S}_{t+1}) | S_t] \end{aligned}$$

$$\begin{aligned} \mathbb{E}_{\pi}[\textcolor{red}{G}_{t+1} | S_t] &= (s \text{ から } S_{t+1} \text{ への遷移確率}) \cdot \mathbb{E}_{\pi}[G_{t+1} | S_{t+1}] \\ &= (s \text{ から } S_{t+1} \text{ の遷移確率}) \cdot \textcolor{red}{V}_{\pi}(\textcolor{red}{S}_{t+1}) \\ &= \mathbb{E}_{\pi}[\textcolor{red}{V}_{\pi}(\textcolor{red}{S}_{t+1}) | S_t = s] \end{aligned}$$

S_t	S_{t+1}	S_{t+2}	S_{t+3}
	$R_{t+1} = 1$	$R_{t+2} = 2$	$R_{t+3} = 3$

$$V_{\pi}(S_{t+1}) = R_{t+2} + \gamma R_{t+3} = 5$$

モンテカルロ法

$$V_{\pi}(S_t) = \frac{G^{(1)}}{1} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} = 6$$

TD法

$$V_{\pi}(S_t) = R_{t+1} + \gamma V_{\pi}(S_{t+1}) = 1 + 5 = 6$$

モンテカルロ法 $V_{\pi}(S_t) = \frac{G^{(1)}}{1} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} = 6$

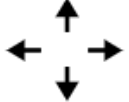
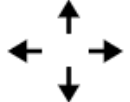
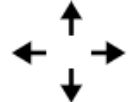

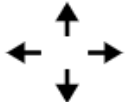
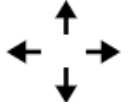
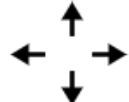
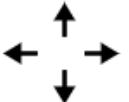
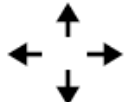


エピソード終了までのすべての報酬を考える
→エピソードが終わらないと推定できない

TD法 $V_{\pi}(S_t) = R_{t+1} + \gamma V_{\pi}(S_{t+1}) = 1 + 5 = 6$

今の報酬と次の状態における価値関数のみ考える
→ステップ毎に逐次推定(更新)できる(ブートストラップ)

価値ベースの手法

ランダムな方策 π_0



				+1
	壁			
				-1

方策評価
(価値関数推定)














方策改善
 $\pi_0 \rightarrow \pi_1$

方策 π_0 における価値関数

0.08	0.09	0.38	
0.00	壁	0.00	0.00
-0.07	-0.18	-0.38	

π_0 から改善した方策 π_1

				
	壁			
				

方策評価
(価値関数推定)



方策 π_1 における価値関数

0.81	0.90	1.00	
0.73	壁	0.90	1.00
0.66	0.59	0.81	

方策ベースの手法

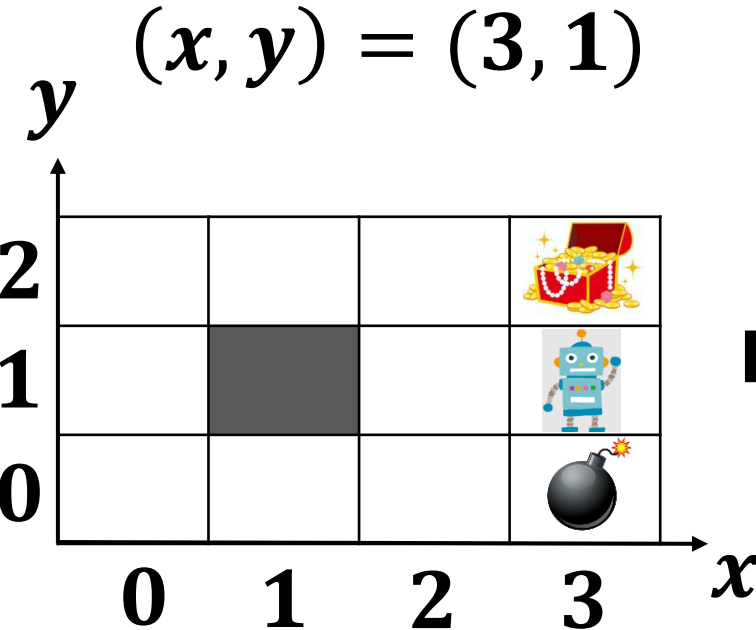
価値関数を経由せず,直接方策を表す

方策勾配法

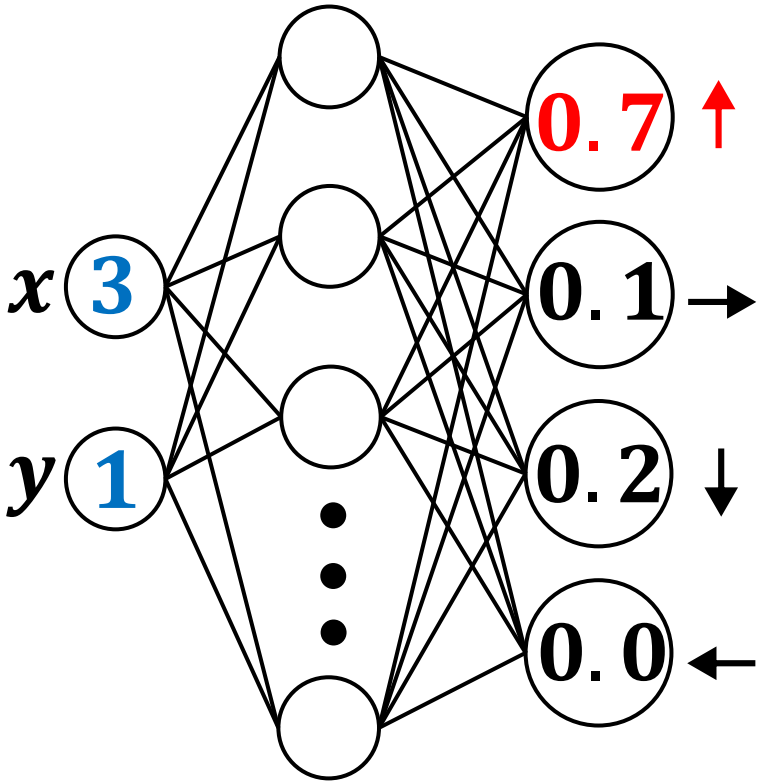
方策をニューラルネットでモデル化し,勾配を用いて最適化

方策ベースの手法

入力

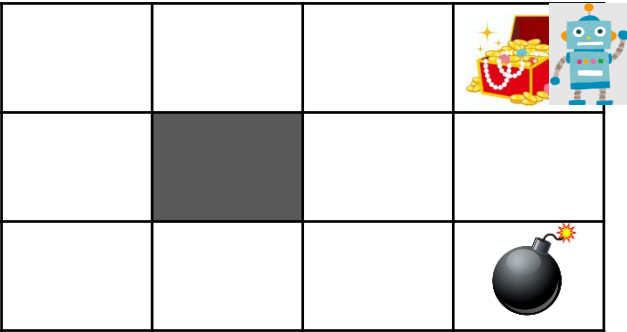


ニューラルネットワーク



出力

行動: \uparrow



REINFORCE アルゴリズム(モンテカルロ方策勾配)

目的関数

$$J(\theta) = v_{\pi_{\theta}}(S_0)$$

更新式(勾配上昇法)

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$$

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} \left[\sum_{t=0}^T G_t \frac{\nabla_{\theta} \pi_{\theta}(A_t | S_t)}{\pi_{\theta}(A_t | S_t)} \right]$$

REINFORCE アルゴリズム(モンテカルロ方策勾配)

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T G(t) \frac{\nabla_{\theta} \pi_{\theta}(A_t | S_t)}{\pi_{\theta}(A_t | S_t)} \right]$$

$\nabla_{\theta} \pi_{\theta}(A_t | S_t)$: 状態 S_t で行動 A_t を取る確率が最も増える方向

$\frac{1}{\pi_{\theta}(A_t | S_t)}$: 状態 S_t で行動 A_t を取る確率の逆数
→ 確率が低い(高い)行動ほど更新量が多く(少なく)なる

$G(t)$ → 収益が大きい(小さい)ほど, 更新量が増える(減る)

REINFORCE アルゴリズム(モンテカルロ方策勾配)

(1) π_θ に従ってエピソードを生成 $\rightarrow S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

(2)時刻 t における収益 $G(t)$ を算出

(3)算出した収益などを用いて勾配上昇法でパラメータ更新

(4)すべての時刻において(2),(3)を繰り返す

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T G(t) \frac{\nabla_\theta \pi_\theta(A_t | S_t)}{\pi_\theta(A_t | S_t)} \right]$$

モンテカルロ法と同じくエピソードが終わらないと更新できない

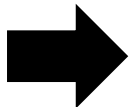
ベースライン付きREINFORCEアルゴリズム

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} \left[\sum_{t=0}^T (G_t - b(S_t)) \frac{\nabla_{\theta} \pi_{\theta}(A_t | S_t)}{\pi_{\theta}(A_t | S_t)} \right]$$

重み(G_t)の分散を小さくしたい

→ G_t となるべく近い値(通常 $V_{\pi_{\theta}}(S_t)$)をベースラインにする

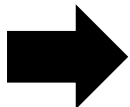
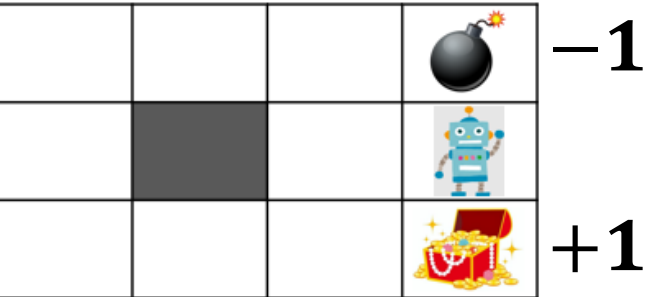
ベースライン付きREINFORCEアルゴリズム



報酬 $R = +1 \rightarrow +1$ 分の重み更新



報酬の位置を入れ替え






$R = -1 \rightarrow -1$ 分の重み更新

学習が進むと

$R = +1$ のとき \rightarrow 重みを更新する必要性 **低**

$R = -1$ のとき \rightarrow 重みを更新する必要性 **高**




価値関数によるネットワークの学習

				+1
				
				-1

$V_{(x,y)} : (x,y)$ における価値関数

➡ $R = +1, V_{(3,2)} = 0.9 \rightarrow +0.1$ 分の重み更新

↓ 報酬の位置を入れ替え

				-1
				
				+1

➡ $R = -1, V_{(3,2)} = 0.9 \rightarrow -1.9$ 分の重み更新

RとVが近い値→重み更新少

RとVが離れた値→重み更新多

➡ 効率的な学習

Actor-Critic法

REINFORCE アルゴリズム(モンテカルロ方策勾配)の問題点

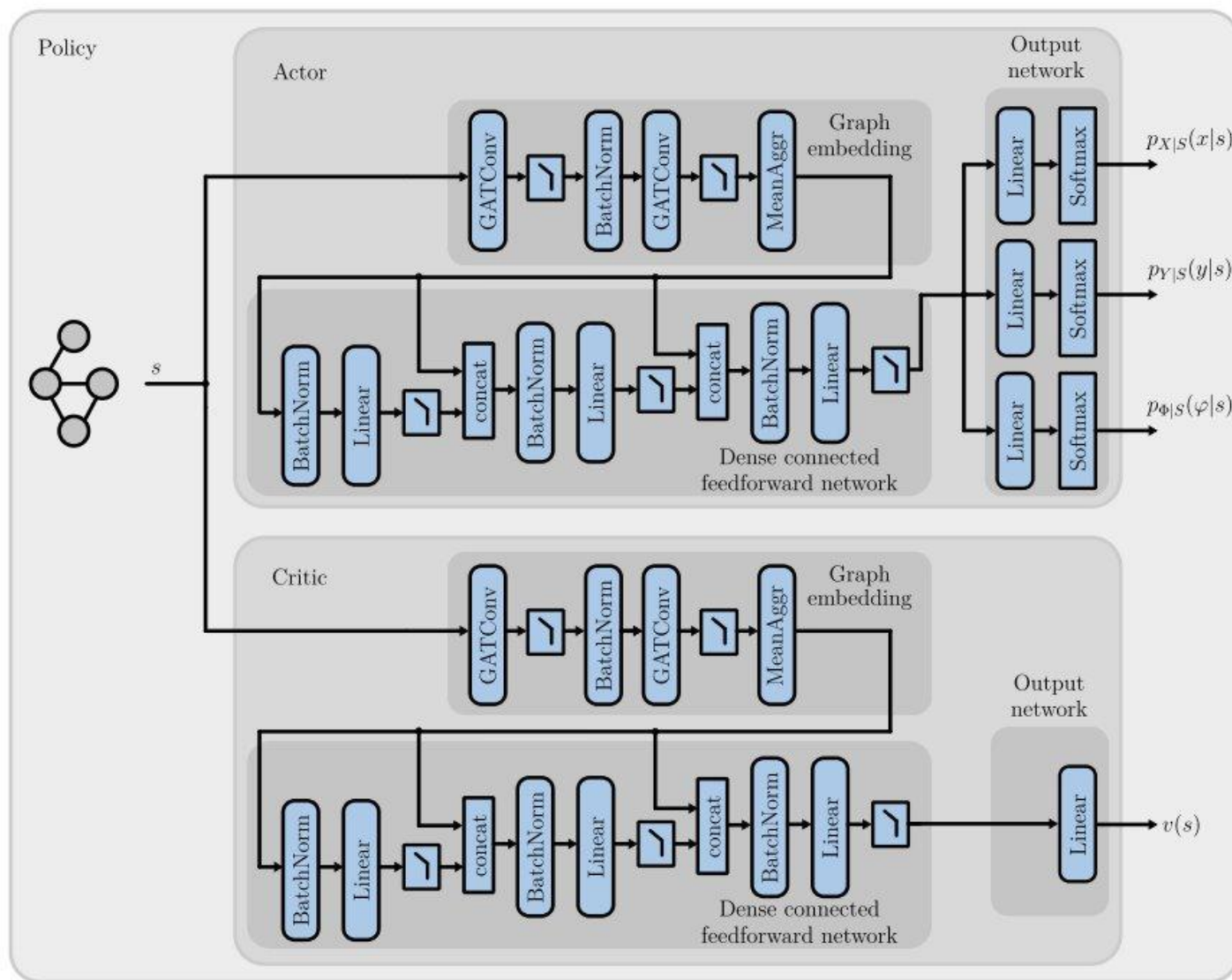
エピソードが終わらないと更新できない →TD法を用いて逐次更新

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} \left[\sum_{t=0}^T (G_t - b(S_t)) \frac{\nabla_{\theta} \pi_{\theta}(A_t | S_t)}{\pi_{\theta}(A_t | S_t)} \right]$$

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T (R_{t+1} + \gamma V_w(S_{t+1}) - V_w(S_t)) \frac{\nabla_{\theta} \pi_{\theta}(A_t | S_t)}{\pi_{\theta}(A_t | S_t)} \right]$$

方策だけでなく, 価値関数 $V_w(S_t)$ もニューラルネットワークで近似

Actor-Critic法



方策を表すNN→Actor

価値関数を表すNN→Critic

Fig. 5.3 Actor-critic styled policy network architecture

Actor-Critic法

- (1) π_θ に従って行動 A_t を取り, R_{t+1}, S_{t+1} を得る
- (2) 時刻 t における収益 $G(t)$ を算出
- (3) 算出した収益などを用いて勾配上昇法でパラメータ更新
- (4) すべての時刻において(2),(3)を繰り返す

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T (R_{t+1} + \gamma V_w(S_{t+1}) - V_w(S_t)) \frac{\nabla_\theta \pi_\theta(A_t | S_t)}{\pi_\theta(A_t | S_t)} \right]$$

モンテカルロ法と同じくエピソードが終わらないと更新できない

アクター・クリティック法

- ・ニューラルネットワークの学習に価値関数を用いる
 - 方策ベースかつ価値ベースの手法
- ・価値関数と実際に得られた報酬の**差分**に応じて、重みを更新
 - 脳の活動と似ている(ドーパミンによる**報酬予測誤差仮説**)

初当たりのとき

価値関数 小
↑
↓
報酬 大



確変中に当たったとき

価値関数 大
↑
↓
報酬 大



[Schultz. W, "Predictive reward signal of dopamine neurons", journal of Neurophysiology., 1998]

方策勾配法の学習改善: TRPO

方策勾配法の学習 $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$

Actor-Critic

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T (R_{t+1} + \gamma V_w(S_{t+1}) - V_w(S_t)) \frac{\nabla_{\theta} \pi_{\theta}(A_t | S_t)}{\pi_{\theta}(A_t | S_t)} \right]$$

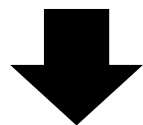
更新すべきベクトル $\nabla_{\theta} J(\theta)$ によって適切な更新幅 α は変わる
がどう設定して良いかわからない

方策勾配法の学習改善: TRPO

更新幅に制約をつけ, 学習を安定化させる

→ TRPO (Trust Region Policy Optimization)

更新前と更新後の行動の確率分布の差 (KL 距離) が適度になるような更新幅を設定する



毎回の更新毎に KL 距離が一定値以下という制約のもとで目的関数を最大化するパラメータを求める (制約付き最大化問題)

方策勾配法の学習改善:PPO

TRPOの問題点

目的関数を最大化するパラメータを毎回求める計算量がとても多く,実装も複雑になりすぎる

***From Code to ChipでのMDPの対応

- Problem=packing
- State=セルのパラメータのベクトルとセル間の接続情報(グラフ)
- Reward= $-(\text{HPWL} + \gamma \sqrt{\text{congestion}})$
- Action=グリッド上の座標(x,y)とセルの配置の向き(90度、180度, e.t.c)