

CS 知的システム演習

# 強化学習で 迷路を解くアルゴリズム

---

松吉 俊

# 強化学習を利用する手順

---

- (1) 状態を定義する
- (2) 報酬関数を定義する
- (3)  $Q$  学習のアルゴリズムを適用する
- (4) 十分な学習を行ったのちに最適政策が得られる(はずな)ので、それに従って行動する

# 強化学習を利用する手順

---

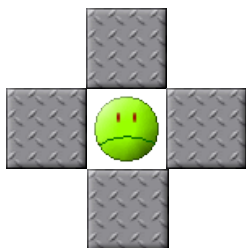
- (1) 状態を定義する
- (2) 報酬関数を定義する
- (3)  $Q$  学習のアルゴリズムを適用する
- (4) 十分な学習を行ったのちに最適政策が得られる(はずな)ので、それに従って行動する

# 状態

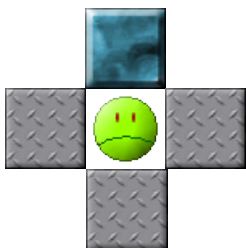
状態とは、エージェントが利用可能な情報のこと

## <例その1>

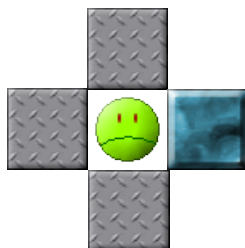
もしエージェントが上下左右のマスを見ることができれば...



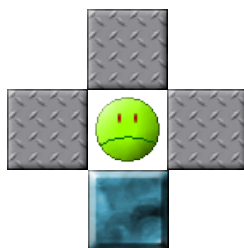
状態1



状態2

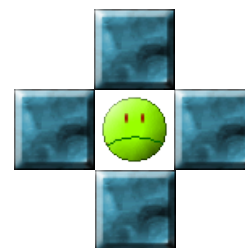


状態3



状態4

...



状態16

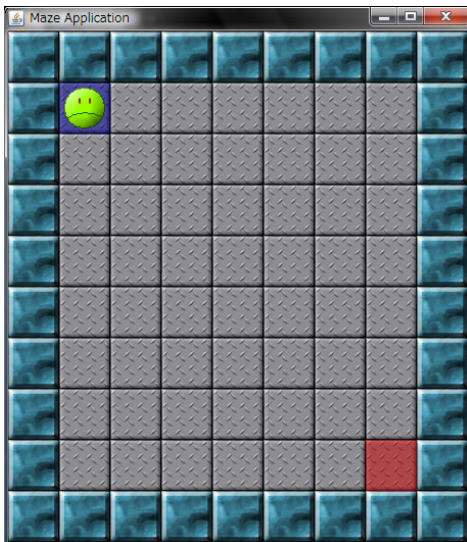
全部で16状態ある

# 状態

状態とは、エージェントが利用可能な情報のこと

## <例その2>

もしエージェントが現在座標を利用できるならば...



状態1 = (0,0)

状態2 = (0,1)

状態3 = (0,2)

⋮

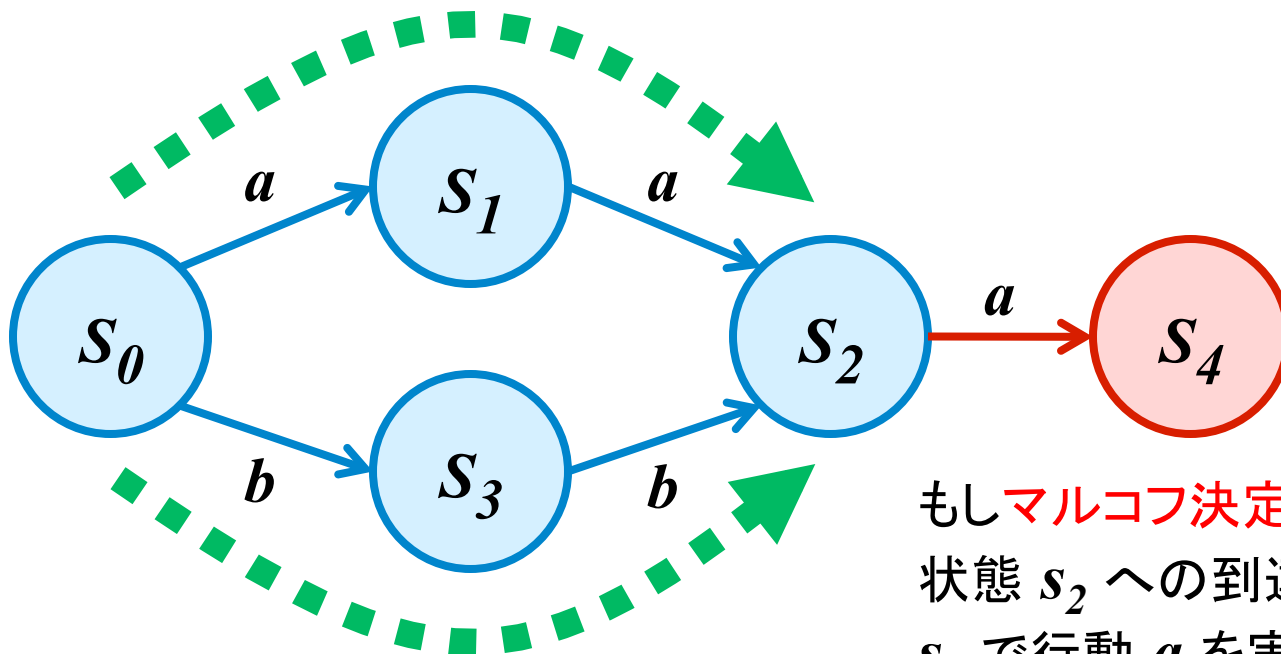
状態89 = (9,9)

状態90 = (9,10)

迷路のサイズが  $9 \times 10$  のとき  
90個の状態がある

# マルコフ決定過程(Markov decision process)

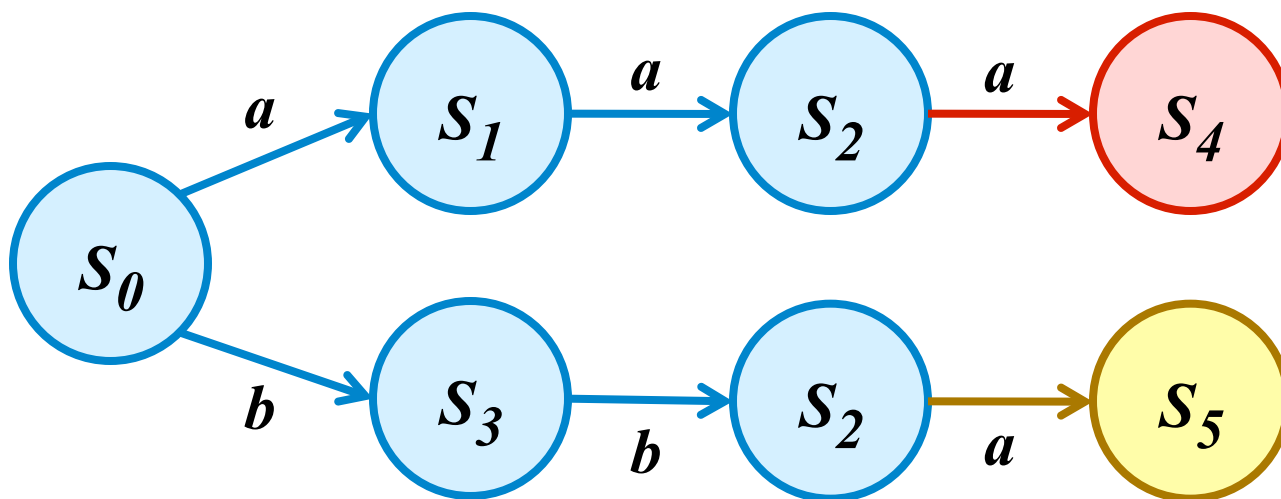
現在の状態からどの状態に遷移するかは、行動と  
現在の状態のみに依存し、過去の状態に依存しない



もしマルコフ決定過程ならば、  
状態  $s_2$  への到達経路にかかわらず、  
 $s_2$  で行動  $a$  を実行すると  $s_4$  に遷移する

# 非マルコフ決定過程

現在の状態からどの状態に遷移するかが、  
過去の状態に依存する

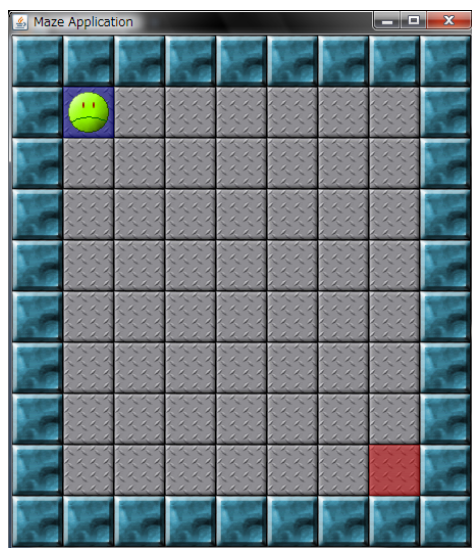


非マルコフ決定過程では、  
 $s_2$  で行動  $a$  を実行したときの遷移先が  
状態  $s_2$  への到達過程に依存する

# マルコフ決定過程の例

## <例その2>

もしエージェントが**現在座標**を利用できるならば...



状態1 = (0,0)

状態2 = (0,1)

状態3 = (0,2)

⋮

状態89 = (9,9)

状態90 = (9,10)

迷路のサイズが  $9 \times 10$  のとき  
90個の状態がある

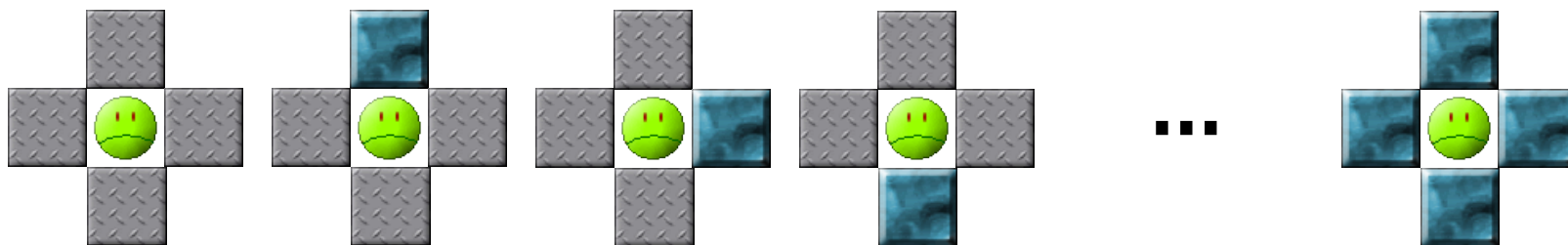
状態  $(x, y)$  において、右に移動すると状態  $(x+1, y)$  に遷移する  
(状態  $(x, y)$  にどのように到達したかは関係ない)



# 非マルコフ決定過程の例 (1)

## <例その1>

もしエージェントが**上下左右のマス**を見ることができるならば...



状態1

状態2

状態3

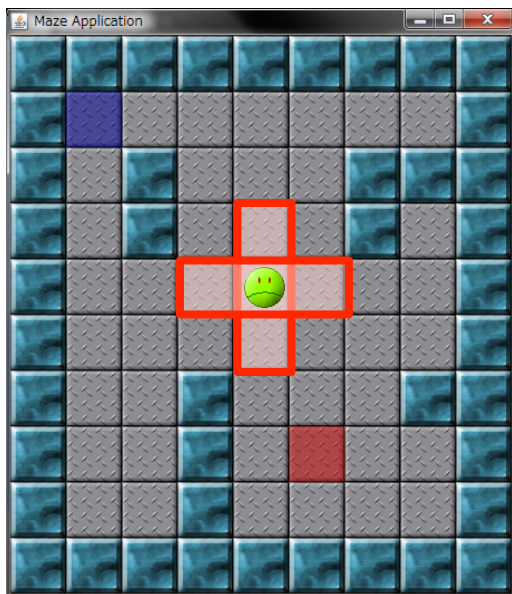
状態4

状態16

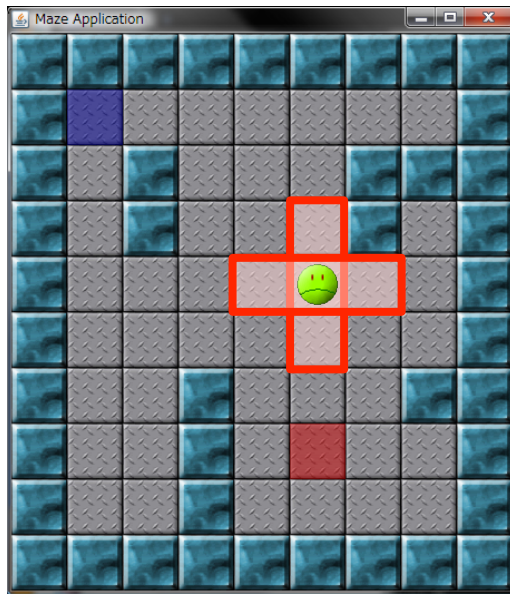
全部で16状態ある

状態1で右に移動したときの遷移先は、  
状態1への到達過程に依存する

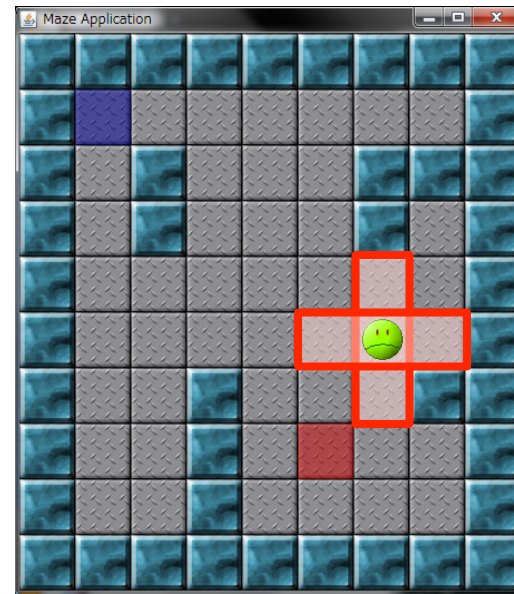
# 非マルコフ決定過程の例 (2)



右に移動⇒状態1



右に移動⇒状態2



右に移動⇒状態10

すべて状態1だが、右に移動したときの  
遷移先はすべて異なる

# 強化学習の対象とするモデル

---

## マルコフ決定過程

(本によっては英語の略称の「MDP」で呼ばれる)

- 学習対象がマルコフ決定過程であるならば、  
十分な試行の後に最適政策が得られる
- 非マルコフ決定過程の場合は、遷移確率を  
正しく推定できない

# 非マルコフ問題

---

一般に、エージェントが環境のすべてを観測することは困難であり、部分的にしか観測できない



マルコフ決定過程であるとは限らない

タスクが非マルコフ決定過程だからといって  
強化学習が適用できないわけではない

# 非マルコフ問題の解法

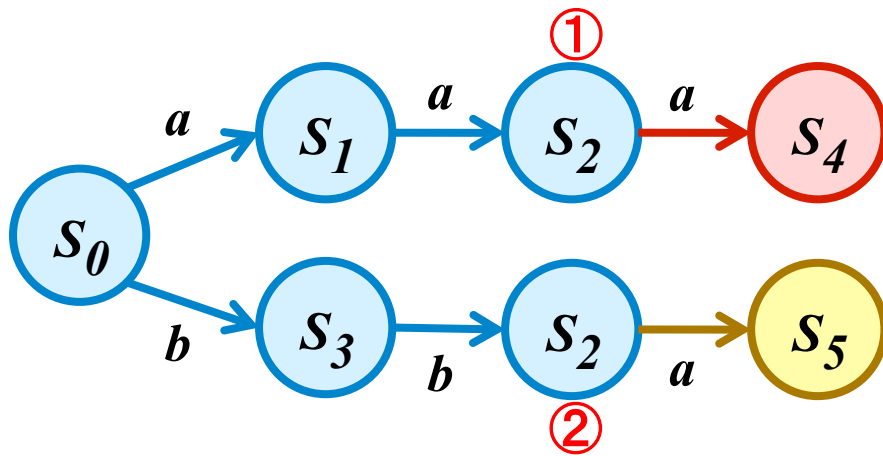
## ● 解法1：気にしない

観測の不完全性が小さければ、強化学習をそのまま適用しても、うまくいく場合がある

## ● 解法2：有限長の過去の履歴を状態に追加する

過去の状態と行動の履歴を現在の状態の一部として考える

例：2ステップ前までの履歴を考慮する場合



①の状態  $s_2$  には、 $\langle (s_0, a), (s_1, a) \rangle$  という履歴を状態の一部として追加

②の状態  $s_2$  には、 $\langle (s_0, b), (s_3, b) \rangle$  という履歴を状態の一部として追加

2つの状態  $s_2$  が区別可能になる

# 強化学習を利用する手順

---

- (1) 状態を定義する
- (2) 報酬関数を定義する
- (3)  $Q$  学習のアルゴリズムを適用する
- (4) 十分な学習を行ったのちに最適政策が得られる(はずな)ので、それに従って行動する

# 報酬関数を定義する

---

## ● 「状態」に対する報酬

例：定義した「状態」によるが、  
座標を「状態」にとったとすると・・・

1. ゴール座標：プラスの値
2. 壁座標：マイナスの値
3. 通路座標：適当な値

# 強化学習を利用する手順

---

- (1) 状態を定義する
- (2) 報酬関数を定義する
- (3)  $Q$  学習のアルゴリズムを適用する
- (4) 十分な学習を行ったのちに最適政策が得られる(はずな)ので、それに従って行動する



# Q 学習のアルゴリズム

(0) Qテーブル (状態数×行動数の2次元表) を作る

(1) 全ての状態  $s$  と行動  $a$  に対して、 $Q(s,a)$  の値を 0 で初期化

(2) 現在の状態が  $s$  であるとき、 $\epsilon$ -greedy 戦略で行動  $a$  を選択し、それを実行する

(3) 遷移先の状態  $s'$  を観測し、 $Q(s,a)$  の値を次式で更新する

$$Q(s,a) \leftarrow Q(s,a) + \alpha[(r(s') + \gamma \max_{a'} Q(s',a')) - Q(s,a)]$$

(4)  $s'$  を  $s$  としてステップ (2) へ戻る

(2)-(4)を  
サイクルさせる

## (2)-(4)のサイクル

---

```
int trials = 500;    // 強化学習の試行回数
int steps  = 100;    // 1試行あたりの最大ステップ数

for(int t=1; t <= trials; t++) {    // 試行回数だけ繰り返し

    /* ロボットを初期位置に戻す */

    for (int st=0; st < steps; st++) {    // ステップ数だけ繰り返し

        /*  $\epsilon$ -Greedy 法により行動を選択 */

        /* 選択した行動を実行 (ロボットを移動する) */

        /* 新しい状態を観測 & 報酬を得る */

        /* Q 値を更新 */

        /* もし時間差分誤差が十分小さくなれば終了 */

    }
}
```

## (2)-(4)のサイクル

```
int trials = 500;    // 強化学習の試行回数
int steps  = 100;    // 1試行あたりの最大ステップ数

for(int t=1; t <= trials; t++) {    // 試行回数だけ繰り返し

    /* ロボットを初期位置に戻す */

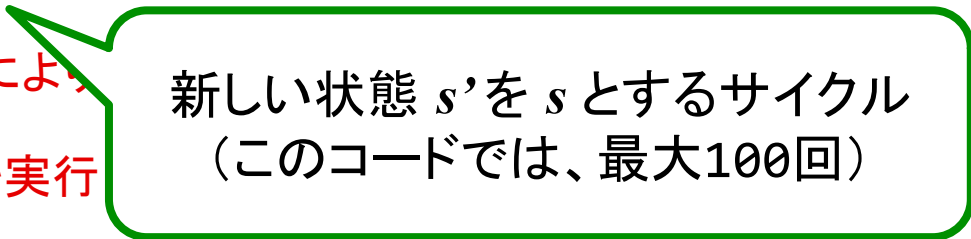
    for (int st=0; st < steps; st++) {    // ステップ数だけ繰り返し

        /*  $\epsilon$ -Greedy 法による選択 */
        /* 選択した行動を実行 */
        /* 新しい状態を観測 & 報酬を得る */

        /* Q 値を更新 */

        /* もし時間差分誤差が十分小さくなれば終了 */

    }
}
```



新しい状態  $s'$  を  $s$  とするサイクル  
(このコードでは、最大100回)

## (2)-(4)のサイクル

```
int trials = 500;    // 強化学習の試行回数
int steps  = 100;    // 1試行あたりの最大ステップ数

for(int t=1; t <= trials; t++) {    // 試行回数だけ繰り返し
```

```
/* ロボットを初期位置にリセット */
```

```
for (int st=0; st < st_max; st++)
```

初期位置から改めて試行錯誤するサイクル  
(このコードでは、500回)

```
/*  $\epsilon$ -Greedy 法による選択 */
```

```
/* 選択した行動を実行 */
```

新しい状態  $s'$  を  $s$  とするサイクル  
(このコードでは、最大100回)

```
/* 新しい状態を観測 & 報酬を得る */
```

```
/* Q 値を更新 */
```

```
/* もし時間差分誤差が十分小さくなれば終了 */
```

```
    }
}
```

# デモ

---

- Step数: 100
  - ロボットは初期位置から100マス移動する
- Trial数: 500
  - 初期位置から、500回、試行錯誤する

上の試行を経て、Qテーブルが更新される

[illegible]

# 強化学習を利用する手順

---

- (1) 状態を定義する
- (2) 報酬関数を定義する
- (3)  $Q$  学習のアルゴリズムを適用する
- (4) 十分な学習を行ったのちに**最適政策が得られる(はずな)ので、それに従って行動する**

# $Q$ 値のテーブル: 最終状態

		行動					
		$a_1$	$a_2$	$a_3$	$a_4$	....	$a_m$
状態	$S_1$	10	3	87	-5		17
	$S_2$	32	5	2	78		0
	$S_3$	67	13	23	9		20
	$S_4$	0	-5	94	43		2
	$\vdots$						
	$S_n$	17	42	8	32		102

greedy 戦略

各状態において  
もっとも大きい  $Q$  値をとる  
ような行動を選択する



---

● Special thanks:

● 山本 泰生先生

● 鍋島 英知先生