



The 24th
LSI 2021
Design Contest in Okinawa

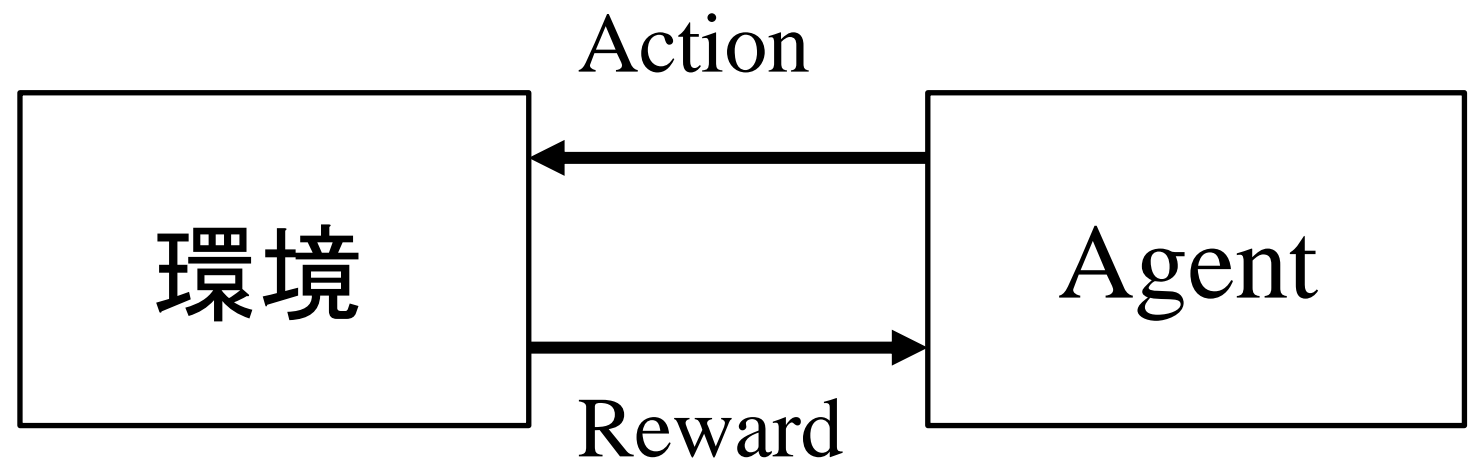
About Q-learning

■ Q-learningとは

- 強化学習の一種で「価値を最大化するような行動」を学習する方法
- 与えられた「環境」における価値を最大化させるように「Agent」を学習させる

■ 要素

- Agent: 行動主体 (プレイヤー)
- State: 状態 (盤面)
- Action: 行動 (コマを置く)
- Reward: 報酬





About Q-learning

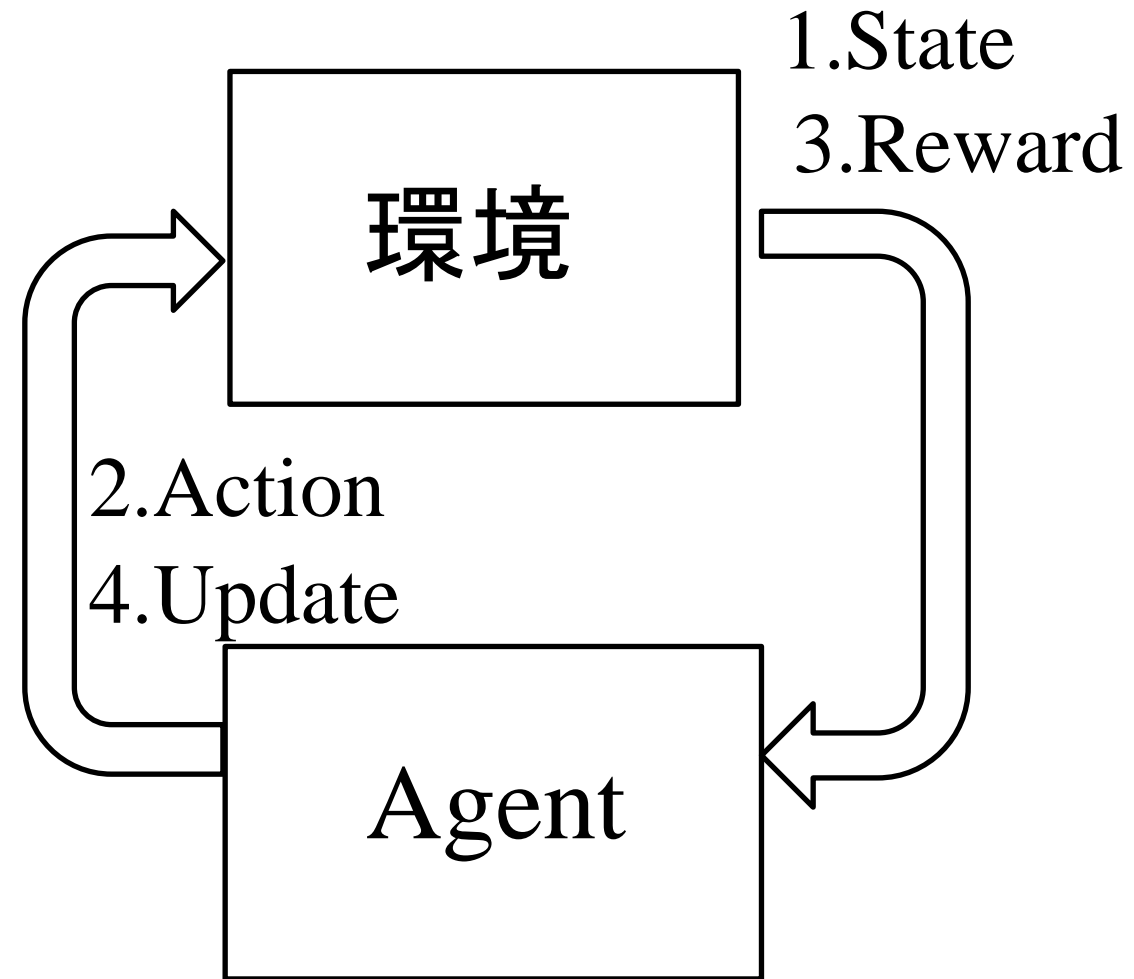
■ 学習の方法

- ある状態 S において行動 A を取った時の価値を最適化していく
- この価値を Q 値または状態行動価値と呼び, $Q(s, a)$ で表される
- $Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha\{r + \gamma \max_{a'} Q(s_{t+1}, a')\}$
- α : 学習率
- r : 報酬
- γ : 割引率

About Q-learning

■ 学習の順序

1. 現在の状態を取得
2. Agentの行動を選択(Q値最大)
3. 報酬を受け取る
4. Q値を更新して1に戻る



Shortest route search

■ 最短経路探索でQ-learningを考える

■ 要素

□ Agent : Person

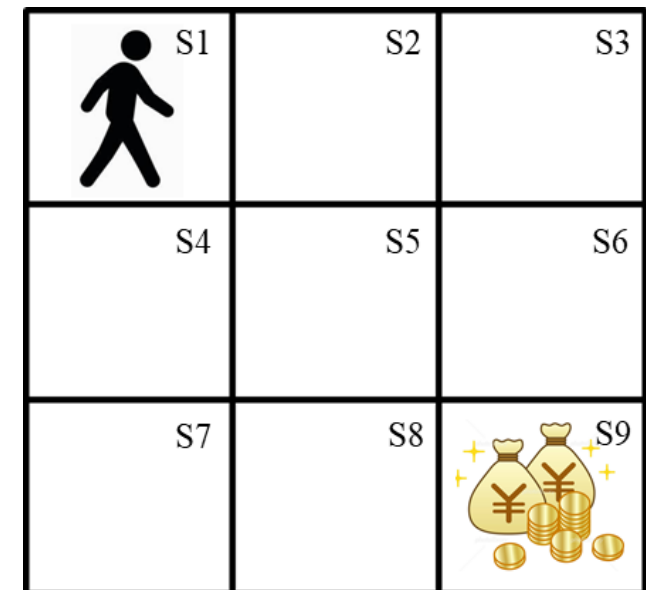
□ Reward : Money

□ Action : \rightarrow or \uparrow or \leftarrow or \downarrow

□ State : S1 ~ S9 (※ Where is the person)



□ スタート : S1, ゴール : S9

環境



Shortest route search

- それぞれのActionとStateにおけるQ table(Q値の表)を作る
- (Q値はランダム値で初期化)

 S1	S2	S3
S4	S5	S6
S7	S8	 S9

		Action			
		→	↑	←	↓
S t a t e	S1	0.1	Out	Out	0.4
	S2	0.2	Out	0.6	0.3
	S3	Out	Out	0.8	0.5
	S4	0.1	0.2	Out	0.9
	S5	0.2	0.4	0.1	0.3
	S6	Out	0.5	0.6	0.7
	S7	0.9	0.4	Out	Out
	S8	0.3	0.4	0.5	Out
	S9	0	0	0	0



Shortest route search

■ 学習の順序

1. Agentの場所を取得
2. Agentの動く方向を選択(Q値最大)
3. 報酬を受け取る
4. Q値を更新して1に戻る

■ 2においてQ値が最大の行動だけを取るとループや局所解に陥ってしまう

→ ϵ - greedy法: ϵ の確率でランダムな行動を取る

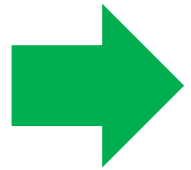
□ $1 - \epsilon$: Q値が最大の行動

□ ϵ : ランダムな行動



ϵ - greedy algorithm

- Agent selects the most valuable action in that state
 - easy to fall into **a local solution** at the learning stage



a certain probability(ϵ), select another action

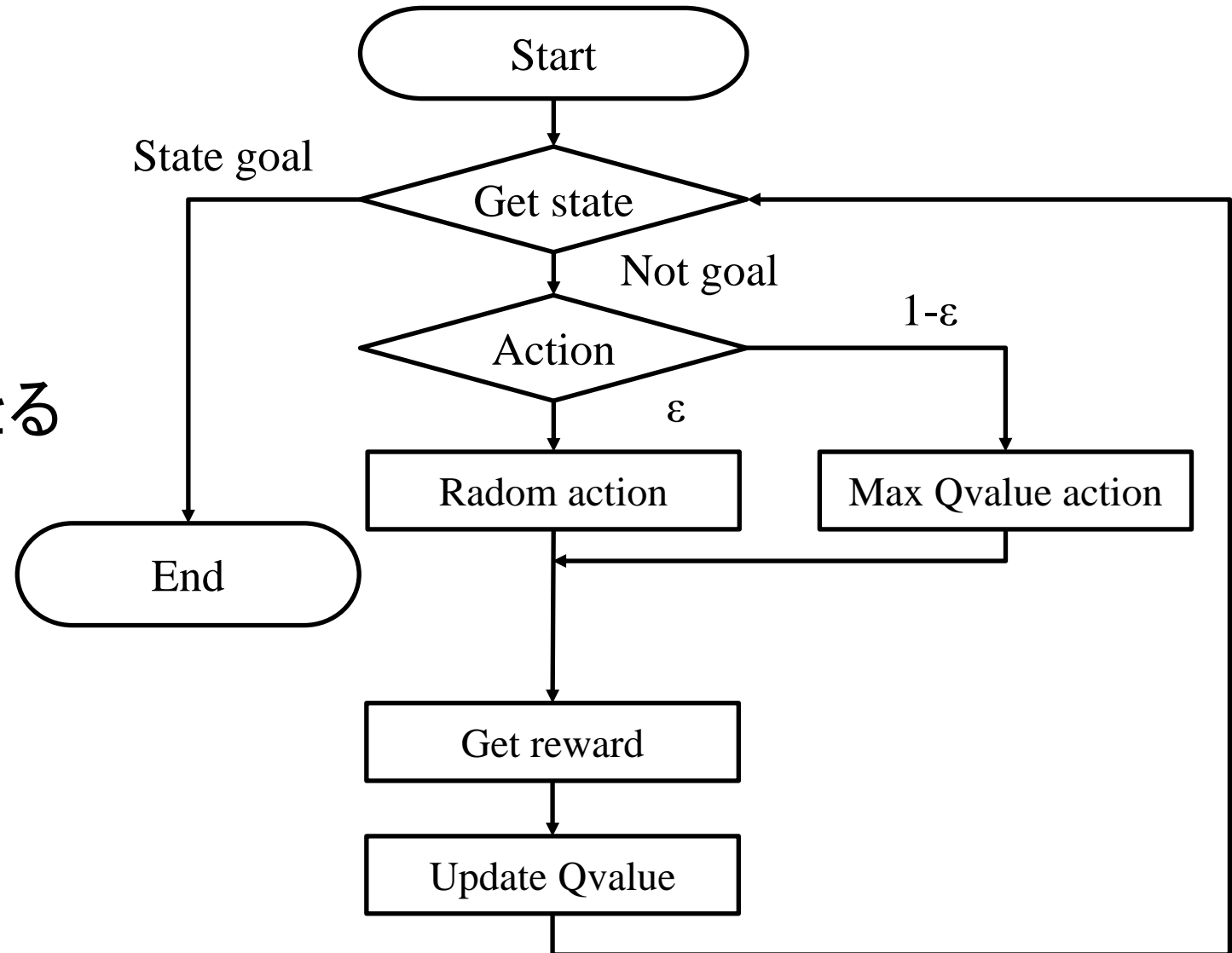
$$\epsilon = 1 - \frac{Episode}{50}$$

⇒ As learning progresses, the percentage of choosing the optimal behavior increases

Program flow

■ プログラムのフローチャート

- Get stateよりゴールに到着した場合は学習を終了する
- Actionは ϵ - greedy法より ϵ の確率でランダムな行動をとる





Explain source code (Matlab)

■ 今回使用しているプログラム

- Search_Location.m

- Action.m



- Q_Learning.m

- Update_Qvalue.m

Explain source code (Search_Location.m)

- 現在Agentがいる場所を探索する関数
 - 引数: 盤面を表す 3×3 配列
 - 返り値: 現在Agentがいる場所のインデックス, 1~9までのState

- 右図のような場合では返り値は (state, 行, 列)=(1, 1, 1)

 S1	S2	S3
S4	S5	S6
S7	S8	 S9

Explain source code (Action.m)

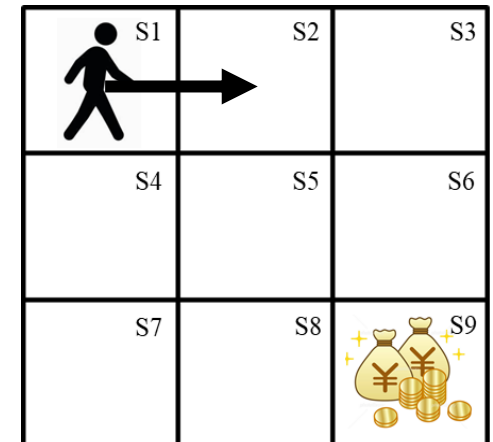
■ Agentが行動を行う関数

□ 引数: 盤面配列, Q_table, epsilon

□ 戻り値: Agentの行動, 新しい盤面配列

■ 現在のStateにおけるQ値が最大の行動を選択する

□ Epsilonの確率によってランダムな行動を選択する



Explain source code (Update_Qvalue.m)

■ Stateを元にQ_tableを更新する関数

□ 引数: State, 更新後のState, Q_table, Agentの行動, 学習率, 割引率

□ 戻り値: Q_table, 更新後のState

□ 更新後のStateを元に報酬の決定, Q_tableの更新を行う

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha\{r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})\}$$

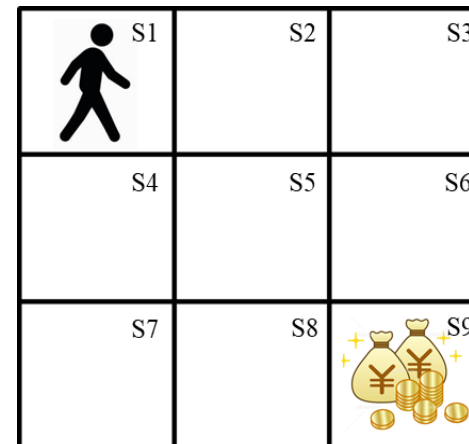
		Action			
		→	↑	←	↓
S t a t e	S1	0.1	Out	Out	0.4
	S2	0.2	Out	0.6	0.3
	S3	Out	Out	0.8	0.5
	S4	0.1	0.2	Out	0.9
	S5	0.2	0.4	0.1	0.3
	S6	Out	0.5	0.6	0.7
	S7	0.9	0.4	Out	Out
	S8	0.3	0.4	0.5	Out
	S9	0	0	0	0

Explain source code (Q_Learning.m)

■ Q-Learningを行うメインプログラム

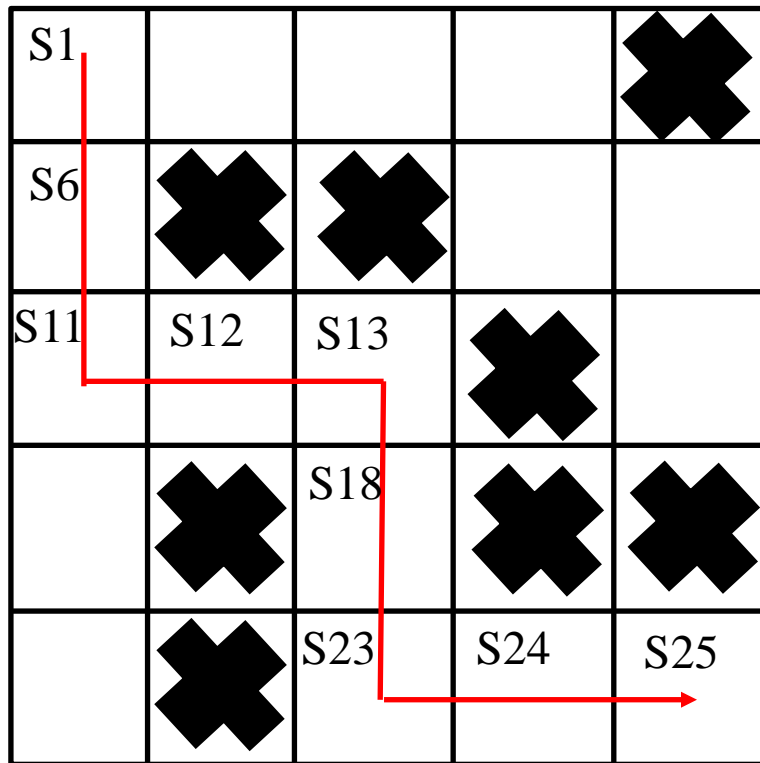
- ハイパーパラメータ, 盤面配列, Q_tableの初期化を行う
- 学習回数: 50世代
- 学習率: 0.5, 割引率: 0.9

	→	↑	←	↓
S1	5.0000	0.0000	0.0000	7.5000
S2	1.0000	0.0000	6.2000	7.0000
S3	0.0000	0.0000	4.0000	3.0000
S4	5.6263	9.0000	0.0000	10.0000
S5	1.0000	7.1500	2.0000	6.0750
S6	0.0000	6.0000	6.0000	2.0000
S7	10.0000	4.5000	0.0000	0.0000
S8	50.0000	4.0000	9.0000	0.0000
S9	0.0000	0.0000	0.0000	0.0000



5 × 5 shortest route search (MATLAB)

■ 盤面5 × 5 (障害物あり) のときの最短経路探索のQLの結果



	→	↑	←	↓
S1	37.5991	0.6787	0.7094	47.8297
S2	0.6688	0.7577	42.3917	-55.1289
S3	0.5989	0.7431	16.0209	-50.0469
S4	-99.3745	0.3922	0.6688	-3.4919
S5	0.6324	0.6555	-0.9608	-24.1453
S6	-47.765	41.8529	0.1626	53.1441
S7	-59.4335	-27.4675	23.2295	58.852
S8	-1.9328	0.6688	-90.8367	60.1969
S9	-39.7581	0.3089	-83.6748	-98.593
S10	0.9649	-99.2546	-4.9928	0.7224
S11	59.049	44.8767	0.5853	28.1177
S12	65.61	-53.1123	52.6627	-35.0871
S13	-44.9239	-52.98	58.2435	72.9
S14	0.719	0.3224	64.3871	-57.6621
S15	0.8003	-39.7564	-60.3461	-61.8221
S16	-50.0201	48.0088	0.6991	0.4777
S17	72.8179	44.3223	-6.4908	-33.1105
S18	-19.2884	61.6634	-38.5205	81
S19	-81.9512	-63.0542	53.3833	89.9971
S20	0.9595	0.741	-74.5067	75.0135
S21	-46.6416	-5.4622	0.1493	0.5308
S22	77.6379	-79.6107	0.476	0.7792
S23	90	67.9878	-31.7123	0.934
S24	100	-19.4903	79.7621	0.1299
S25	0	0	0	0