



HOKKAIDO
UNIVERSITY

講義「人工知能」 第8回 強化学習 2

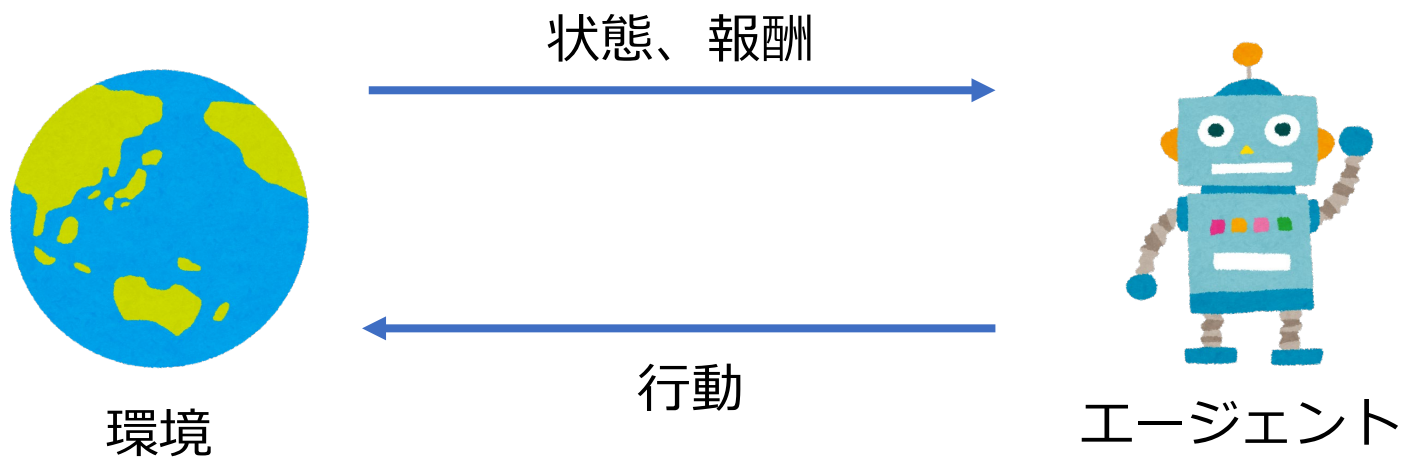
北海道大学大学院情報科学研究院
情報理工学部門 複合情報工学分野
調和系工学研究室 准教授 山下倫央

<http://harmo-lab.jp>
tomohisa@ist.hokudai.ac.jp
2024年5月2日(木)

- 教師あり学習
 - データと正解(ラベル)をセットで与えて学習
 - データを与えたらラベルを出力するようなモデルを構築
 - 例) 分類、回帰
- 教師なし学習
 - データのみを与えて学習(正解なし)
 - データの特徴を抽出できるようなモデルを構築
 - 例) クラスタリング

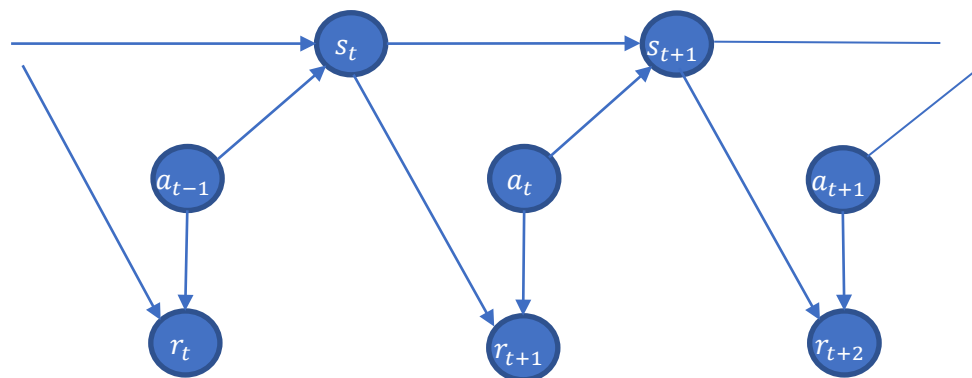
- 強化学習

- 試行錯誤を通じて価値を最大化する行動を学習
- エージェント(行動主体)は環境の状態を観測して行動を決定
- 行動の良さの指標として報酬を与える



- 状態 s_t
 - 時刻 t のエージェントの振る舞いにより変化する環境の要素
- 行動 a_t
 - 時刻 t のエージェントの振る舞い
- (即時)報酬 r_t
 - 時刻 t に得られる行動の即時的な良さ
- ステップ
 - 状態 s_t で行動 a_t を取って、次状態 s_{t+1} に遷移して報酬 r_{t+1} を獲得する過程
- エピソード
 - 以下を繰り返しのうちの1回
 - 終端状態(例: 迷路のゴールにエージェントが到達)に到達
 - 事前に設定した最大ステップ数に到達

- 強化学習
 - 与えられた「環境」が一定のルールに従っていることを想定
 - 環境に対してマルコフ性を仮定
 - マルコフ性を持つ環境をマルコフ決定過程(MDP)と呼ぶ
- マルコフ性
 - 遷移先の状態は直前の状態と行動にのみ依存
- マルコフ決定過程の構成要素
 - 状態集合: $S = \{s_1, s_2, s_3, \dots\}$
 - 行動集合: $A = \{a_1, a_2, a_3, \dots\}$
 - 状態遷移確率: $T(s_t, a_t, s_{t+1}) = P(s_{t+1}|s_t, a_t)$
 - 報酬関数: $R(s_t, a_t, s_{t+1})$



マルコフ決定過程の図

- 方策

- エージェントの行動の選び方、 $\pi(a|s)$

- 収益

- 割引率を考慮した累積報酬和

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k}$$

γ : 割引率($0 \leq \gamma \leq 1$)

- 割引率により将来の報酬を現時点で割り引いて評価

- 強化学習において最大化する目的関数
= 状態価値関数 $V_{\pi}(s)$

- 状態価値関数

- ある状態 s において、方策 $\pi(a|s)$ に従う行動により期待できる収益
- 状態 s の価値

$$V_{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s]$$

- 行動価値関数

- ある状態 s において行動 a を取り、その後方策 $\pi(a|s)$ に従って行動した場合に期待できる収益
- 状態 s において行動 a を取る価値（Q値と呼ぶ）

$$Q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a]$$

強化学習の目的

価値関数 $V_\pi(s)$ を全ての状態 s で最大化する方策を見つける

- 価値関数を再帰的に計算する必要あり

- **ベルマン方程式**：価値関数が満たす再帰的な式
- ベルマン方程式の導出

$$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi[r_{t+1} | S_t = s] + \gamma \mathbb{E}_\pi[G_{t+1} | S_t = s]$$

第1項: $\mathbb{E}_\pi[r_{t+1} | S_t = s] = \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) r(s, a, s')$

第2項: $\gamma \mathbb{E}_\pi[G_{t+1} | S_t = s] = \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) V_\pi(s')$

(価値関数の)ベルマン方程式

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) (r(s, a, s') + \gamma V_\pi(s'))$$

行動価値関数に関しても、以下の式が得られる

行動価値関数のベルマン方程式

$$Q_\pi(s, a) = \sum_{s'} P(s'|s, a) [r(s, a, s') + \gamma \sum_{a'} \pi(a'|s') Q_\pi(s', a')]$$

強化学習の目的

価値関数 $V_\pi(s)$ を全ての状態 s で最大化する方策を見つける

- 価値関数を再帰的に計算する必要あり

- ベルマン方程式：価値関数が満たす再帰的な式
- ベルマン方程式の導出

$$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi[r_{t+1} | S_t = s] + \gamma \mathbb{E}_\pi[G_{t+1} | S_t = s]$$

第1項: $\mathbb{E}_\pi[r_{t+1} | S_t = s] = \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) r(s, a, s')$

第2項: $\gamma \mathbb{E}_\pi[G_{t+1} | S_t = s] = \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) V_\pi(s')$

価値関数のベルマン方程式

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) (r(s, a, s') + \gamma V_\pi(s'))$$

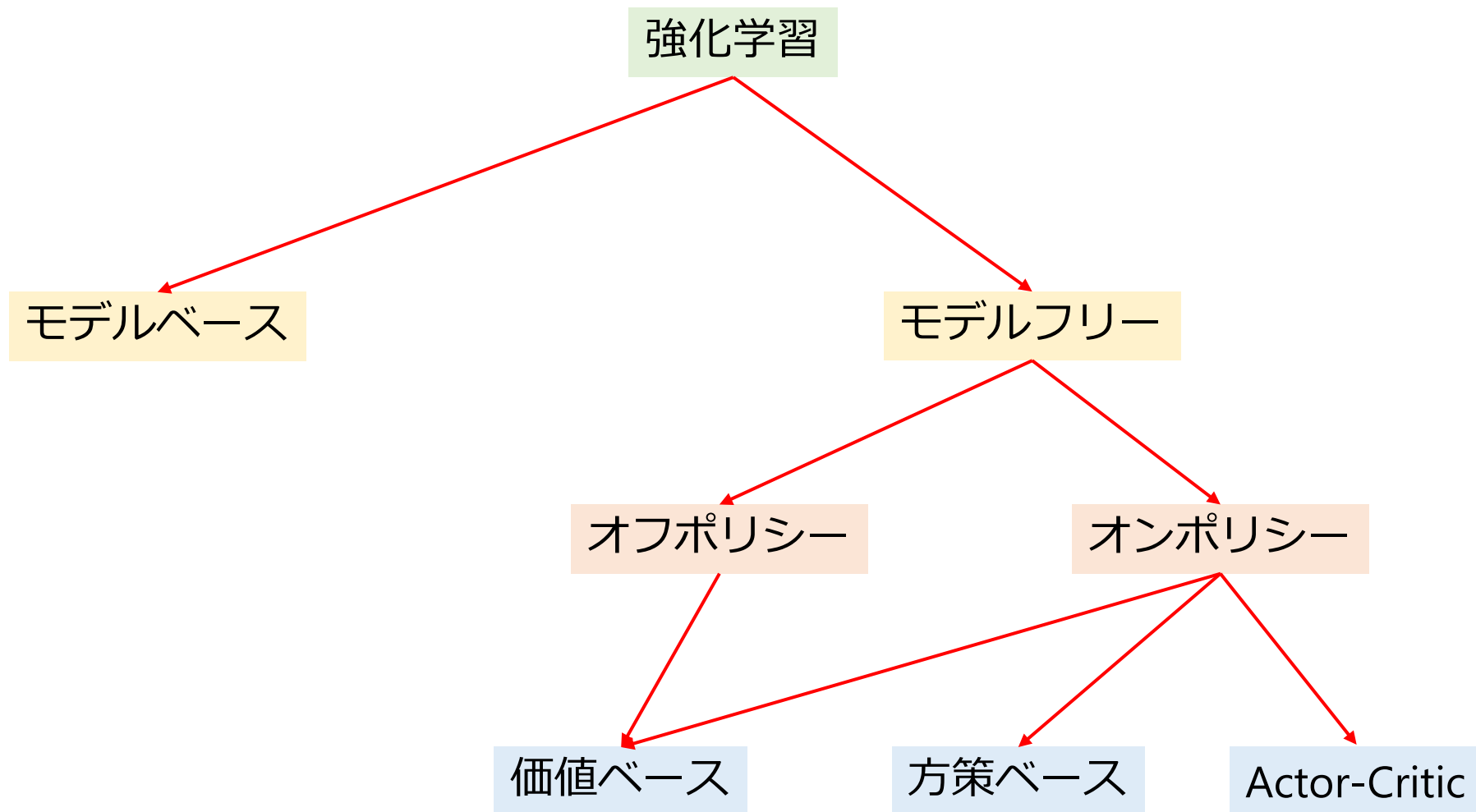
方策 π の元での
状態 s の状態価値

行動 a を取る確率

次の状態 s' になる確率

次の状態 s' での報酬

- 強化学習アルゴリズムは以下のように分類できる



- エージェントが「環境モデル」を使用するかしないかで分類
 - 環境モデルは状態と行動を与えれば報酬と次の状態を返す関数のようなもの
 - 環境モデルは未知であることの方が多い
- モデルベース(Model-base)
 - 環境モデルを使用するアルゴリズム
 - 例) AlphaGo
- モデルフリー(Model-free)
 - 環境モデルを使用しないアルゴリズム
 - 例) Q学習

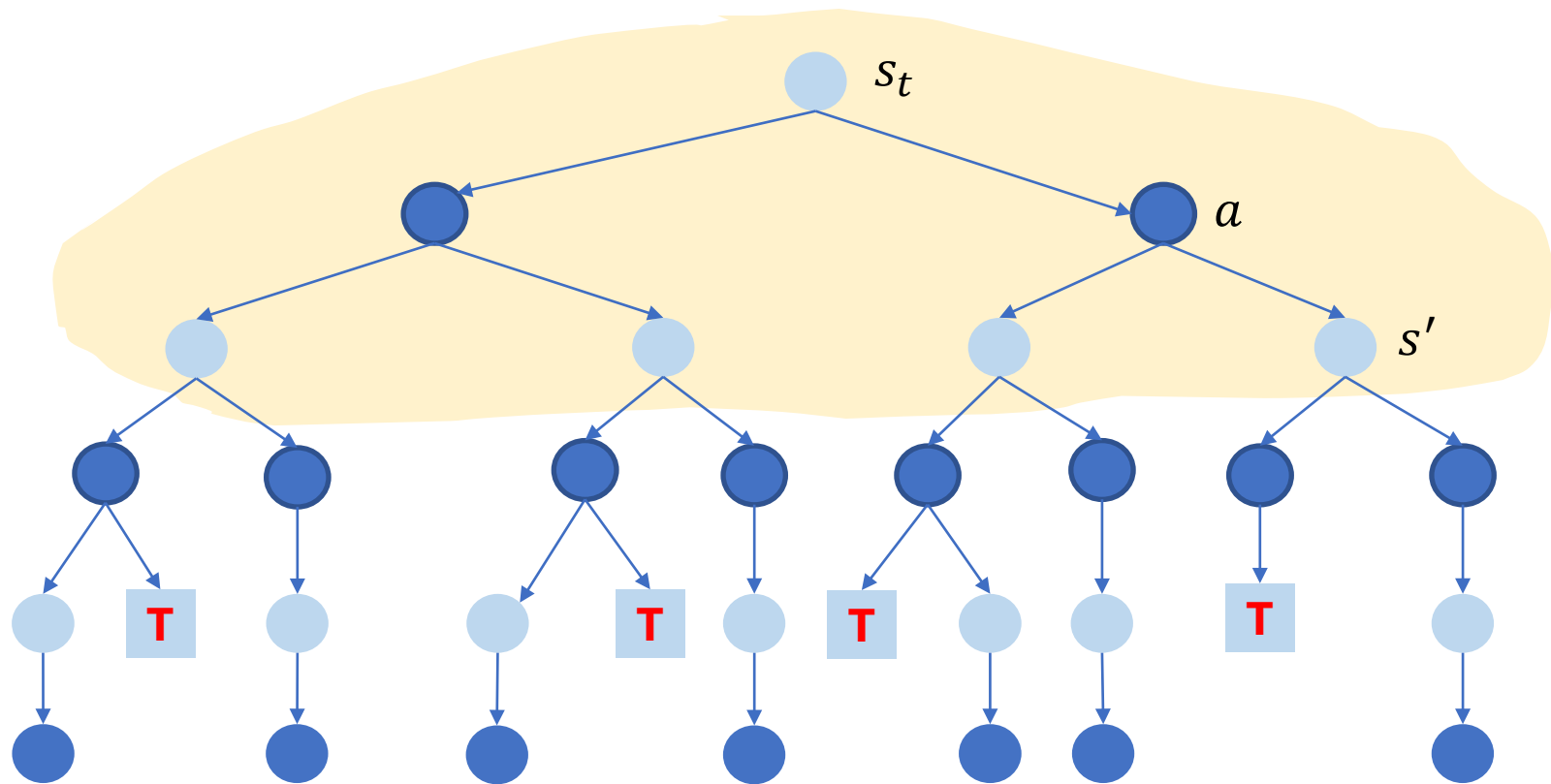
- 環境モデルが既知
 - 状態遷移確率と報酬関数が既知
- 所与のMDPとベルマン方程式をもとに最適方策を学習
 - 最適化問題を分割して再帰的に解いてベルマン方程式の解を求める方法を動的計画法と呼ぶ

方策反復法

1. 価値関数、行動価値関数、方策の初期化
2. (方策評価) 時刻 t で方策 π_t に従う行動価値 Q^{π_t} を計算
3. (方策改善) 行動価値 Q^{π_t} が最大となる行動を選択するように方策を変更(greedy法)
4. 方策の更新がなくなるまで繰り返す

価値反復法

1. 価値関数、行動価値関数、方策の初期化
2. 時刻 t での価値関数 V_t から行動価値 Q_t を計算
3. Q_t の最大値を次時点 $t + 1$ での価値関数 V_{t+1} とする
4. Q_t を最大とする行動 a を次時 $t + 1$ 点での方策 π_{t+1} とする
5. 価値関数が収束するまで繰り返す



- 動的計画法の欠点
 - 状態遷移確率がわかっている必要がある

- 環境モデルが未知
 - 所与の環境の下で行動選択と報酬の獲得を繰り返して状態・行動系列をサンプリング
 - 探索的に最適方策を学習
- 検討ポイント
 1. 経験の蓄積と活用のバランス
 2. 学習を実績と予測のどちらに従って行うか
 3. 経験を価値関数の更新に使うか方策の更新に使うか

- 環境モデルが未知
 - 自らの行動により状態の遷移と得られる報酬のデータを収集
=探索
 - 過去の経験を活用して最も良いと考えられる行動を取る
=活用
- 探索と活用のトレードオフ
 - 探索と活用のバランスをうまく調整する必要がある
 - 探索ばかりだと過去の経験を活かさない
 - 活用ばかりだともっと良い行動を見つけられない

- 探索と活用のトレードオフのバランスを取る手法
 - ϵ – greedy法
 - softmax行動選択
- ϵ – greedy法
 - ϵ (0~1の間の乱数)の確率でランダムな行動を行い(=探索)、 $(1 - \epsilon)$ の確率でQが最大となる行動を取る(=活用)
 - 探索の際に全ての行動を等しく選択するという欠点あり

ϵ – greedy 法の手順

- ある適当な定数 ϵ を用意
- 行動選択の際、0~1の間の乱数 x を生成
 - if $x \leq \epsilon$;
 ランダムな行動;
 - else;
 Q値が最大となる行動;

- Softmax 行動選択

- ϵ – greedy法の欠点を解決する手法
- Softmaxは合計が1になるように正規化する関数

$$\pi(a) = \frac{e^{\frac{Q(a)}{T}}}{\sum_{b \in A} e^{\frac{Q(b)}{T}}}$$

$\pi(a)$: 行動 a を選択する確率

$Q(a)$: 行動 a の行動価値

A : 行動集合

T : 温度

温度が高いほど全ての行動を等しく選択

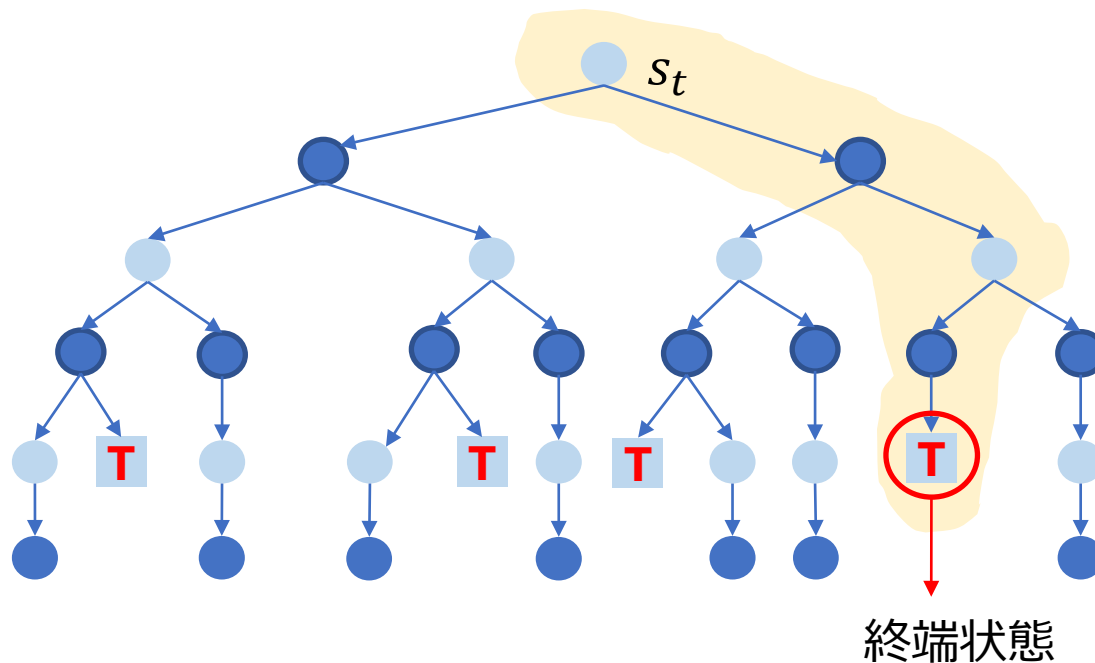
- 学習方法として以下の2つに分類
 1. 1エピソード動かしてから修正する(実績)
 - エピソード終了後に学習を行うため、報酬が確定して学習が正確に行える
 2. エピソードの終了を待たずに途中の状態を使って修正(予測)
 - エピソードの終了を待たなくてよいため学習が速く進む
- 実績の手法として、モンテカルロ法
- 予測の手法として、TD学習
- この分類方法はエピソード終了が定まっている場合に限る(エピソード的タスク)
↔連続タスク

- 状態報酬系列 $\{s_t, r_t | t = 0, 1, \dots, T\}$ を終端状態に到達するまでサンプリング

更新式

$$V(s_t) \leftarrow V(s_t) + \alpha \{G_t - V(s_t)\}$$

α : 学習率



- モンテカルロ法の欠点
 - エピソードが終了するまで学習できない

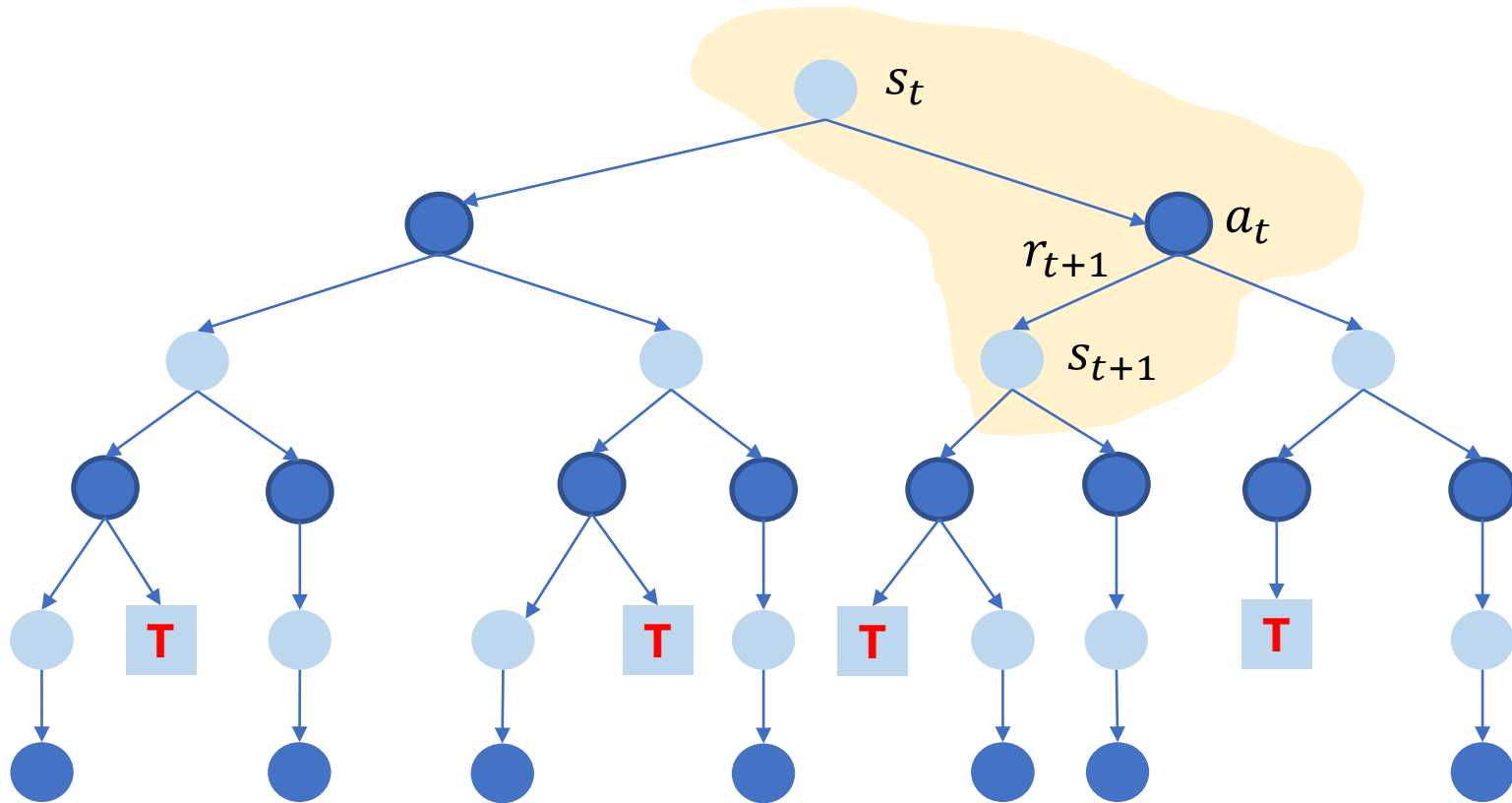
- 動的計画法とモンテカルロ法の中間的な手法
 - 状態遷移確率が未知でも価値関数の推定が可能
 - エピソードの終了を待たなくてよい
- TD誤差を小さくするように価値関数を推定する学習法
- ある時間ステップ t で得た経験 $\{s_t, a_t, r_{t+1}, s_{t+1}\}$ を用いて価値関数を更新

更新式

$$V(s_t) \leftarrow V(s_t) + \alpha \underbrace{(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))}_{\text{TD誤差}}$$

α : 学習率
 γ : 割引率

TD誤差



- 状態遷移のモデルがわからなくてもよい
- 次の状態を用いて価値関数の推定(=**ブートストラップ**)が可能

- オンポリシーとオフポリシーで分類
- 挙動方策
 - 学習時の探索に使用する方策
- 推定方策
 - 価値関数の更新に用いる方策
- オンポリシー(On-Policy)
 - 挙動方策=推定方策
- オフポリシー(Off-Policy)
 - 挙動方策 \neq 推定方策

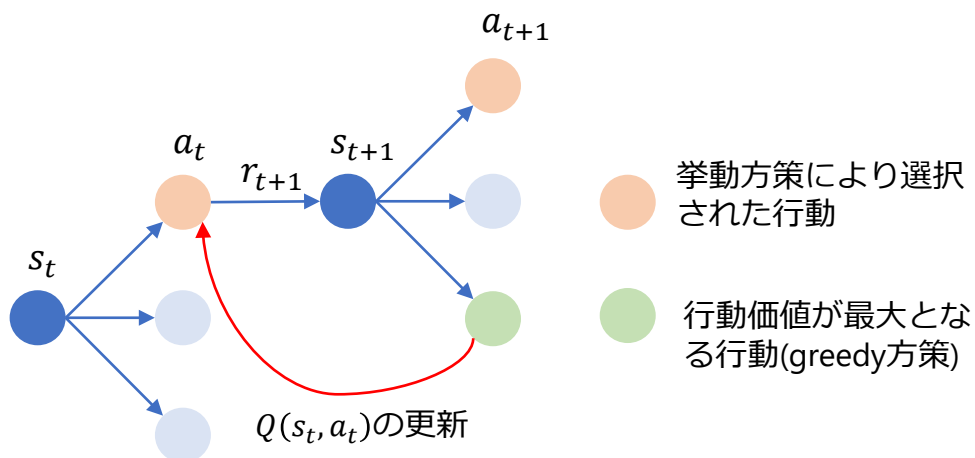
- 経験の使い方で分類
 - 経験を価値関数の更新に使うか、方策の更新に使うか
 - 経験を価値関数の更新に使う：価値ベース
 - 経験を方策の更新に使う：方策ベース
- 価値ベース(value-base)
 - 行動価値関数を直接的に推定
 - それをもとに最適な行動を選択するように方策を改善
- 方策ベース(policy-base)
 - 方策の条件付確率 $\pi(a|s)$ を直接的に推定
 - 方策勾配法により方策を改善
- Actor-Critic
 - 価値ベースと方策ベースを組み合わせた手法

更新式

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

 α : 学習率 γ : 割引率

- 価値ベース手法
- 推定方策としてgreedy方策を使用し、行動価値が最大となる行動を選択したとして $\max_a Q(s_{t+1}, a)$ を使用
- 実際の行動を選択する挙動方策と価値関数の更新に使用する推定方策が異なるオフポリシー型手法



- 学習手順

$Q(s, a)$ を任意に初期化

各エピソードで以下を繰り返し:
 s を初期化

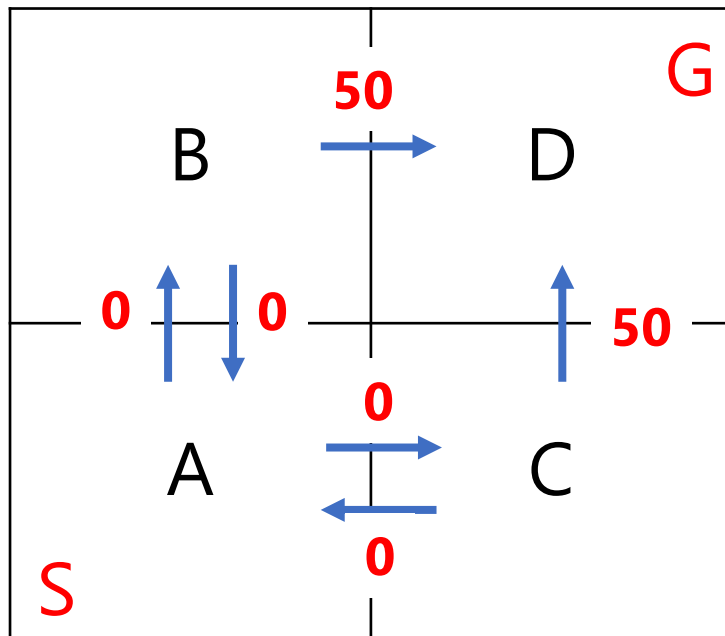
エピソードの各ステップで以下を繰り返し:

- 現在の状態 s_t で、 $Q(s_t, a_t)$ に従ってある行動 a_t を選択し実行
- 環境から遷移先状態 s_{t+1} と報酬 r_{t+1} を受け取る
- 得られた遷移先状態 s_{t+1} と報酬 r_{t+1} を元に以下のように $Q(s_t, a_t)$ を更新

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

s が終端状態なら繰り返しを終了

- 以下のAがスタート、Dがゴールの迷路を考える



状態集合

マスA,B,C,D

行動集合

各マスから各マスへの移動

報酬

ゴールDに到達→50

それ以外→0

エピソード

ゴールDに到達した時点で終了

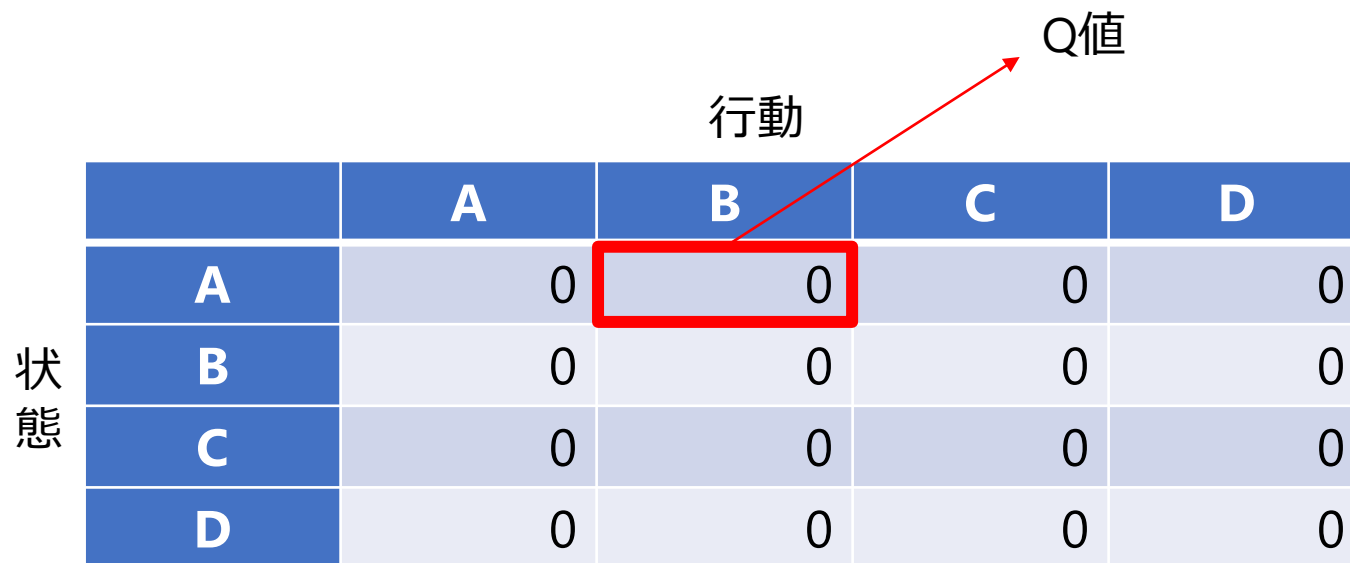
学習率

$\alpha=0.1$

割引率

$\gamma=0.95$

- $Q(s, a)$ を任意に初期化
 - Qテーブルを使用
 - Qテーブルの各値を0に初期化

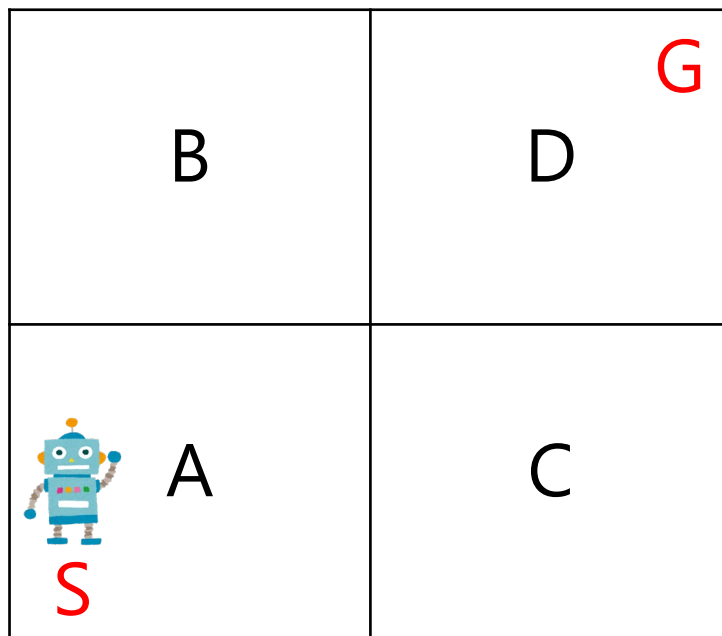


Q値

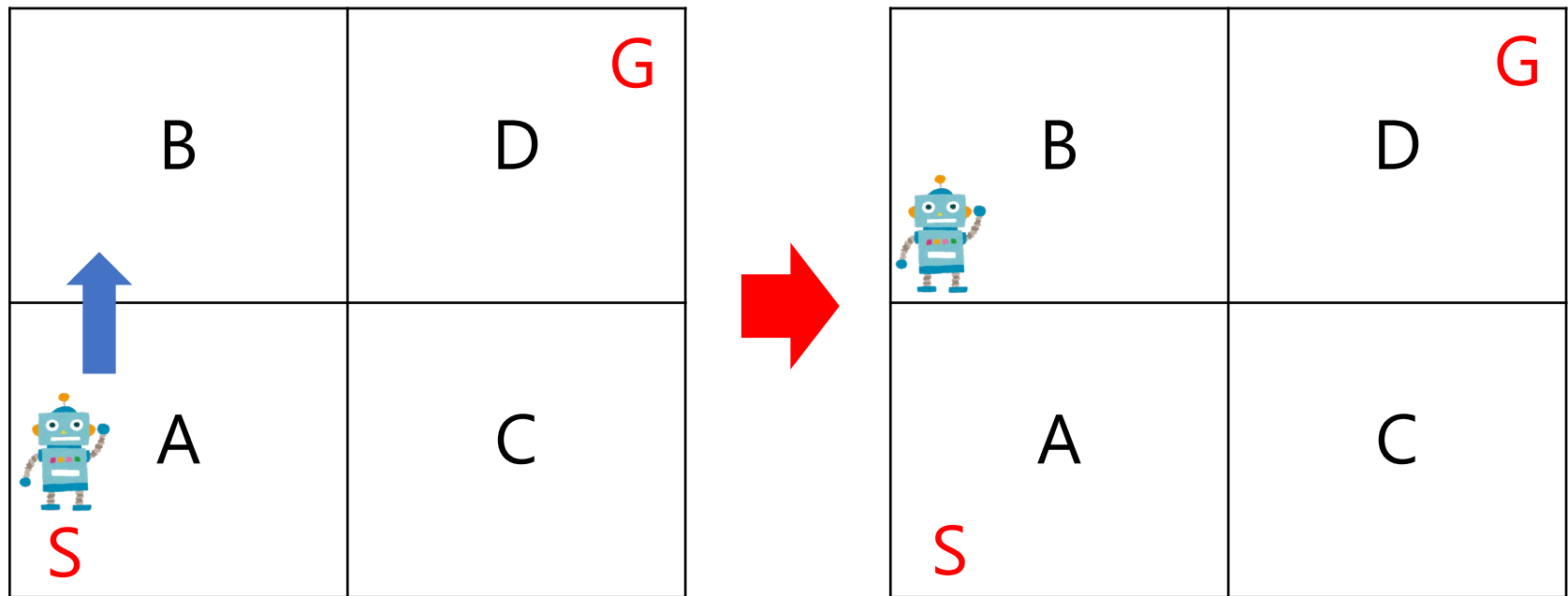
行動

	A	B	C	D
状態 A	0	0	0	0
状態 B	0	0	0	0
状態 C	0	0	0	0
状態 D	0	0	0	0

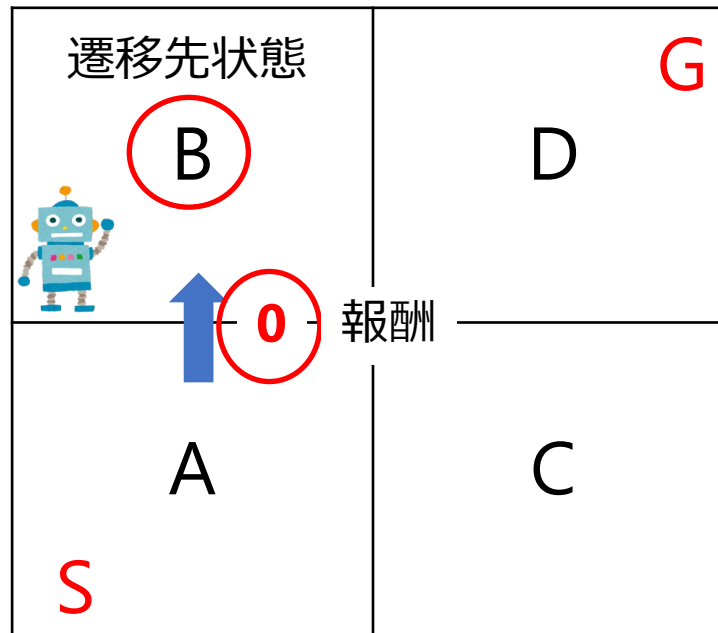
- s を初期化
 - エージェントをスタートAに配置



- 現在の状態 s_t で、 $Q(s_t, a_t)$ に従ってある行動 a_t を選択し実行
 - 移動可能なマスはBとC
 - ϵ - greedy法などにより行動を決める
 - Bに移動するとする



- 環境から遷移先状態 s_{t+1} と報酬 r_{t+1} を受け取る
 - 遷移先状態はマスB
 - 報酬は 0



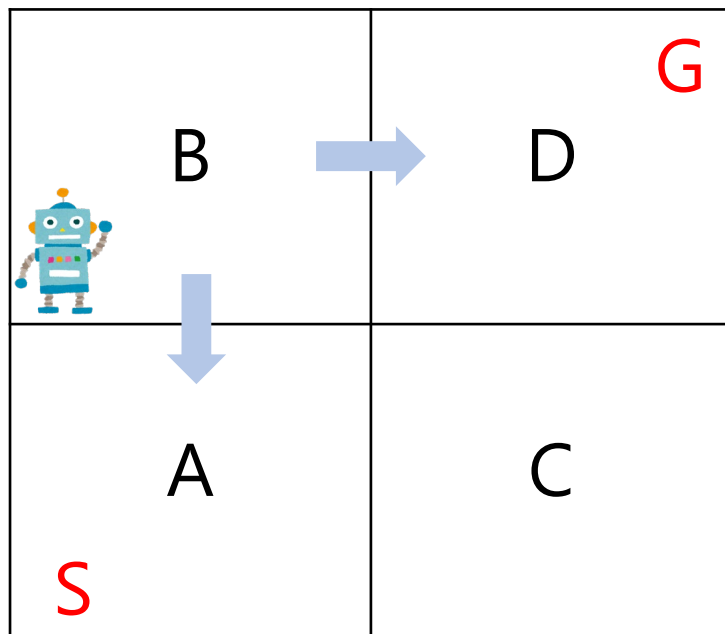
- 得られた遷移先状態 s_{t+1} と報酬 r_{t+1} を元に以下のように $Q(s_t, a_t)$ を更新

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

今回の場合で書くと

$Q(A, B)$

$$\leftarrow Q(A, B) + 0.1 \left[0 + 0.95 \max_a (Q(B, A), Q(B, D)) - Q(A, B) \right]$$



Bから移動可能な方向はAとD

- Qテーブルを試してみる
 - $Q(A,B)$ 、 $Q(B,A)$ 、 $Q(B,D) \rightarrow$ すべて0

状態		A	B	C	D
	A	0	0	0	0
	B	0	0	0	0
	C	0	0	0	0
	D	0	0	0	0

$Q(A, B)$

$$\leftarrow Q(A, B) + 0.1 \left[0 + 0.95 \max_a (Q(B, A), Q(B, D)) - Q(A, B) \right]$$

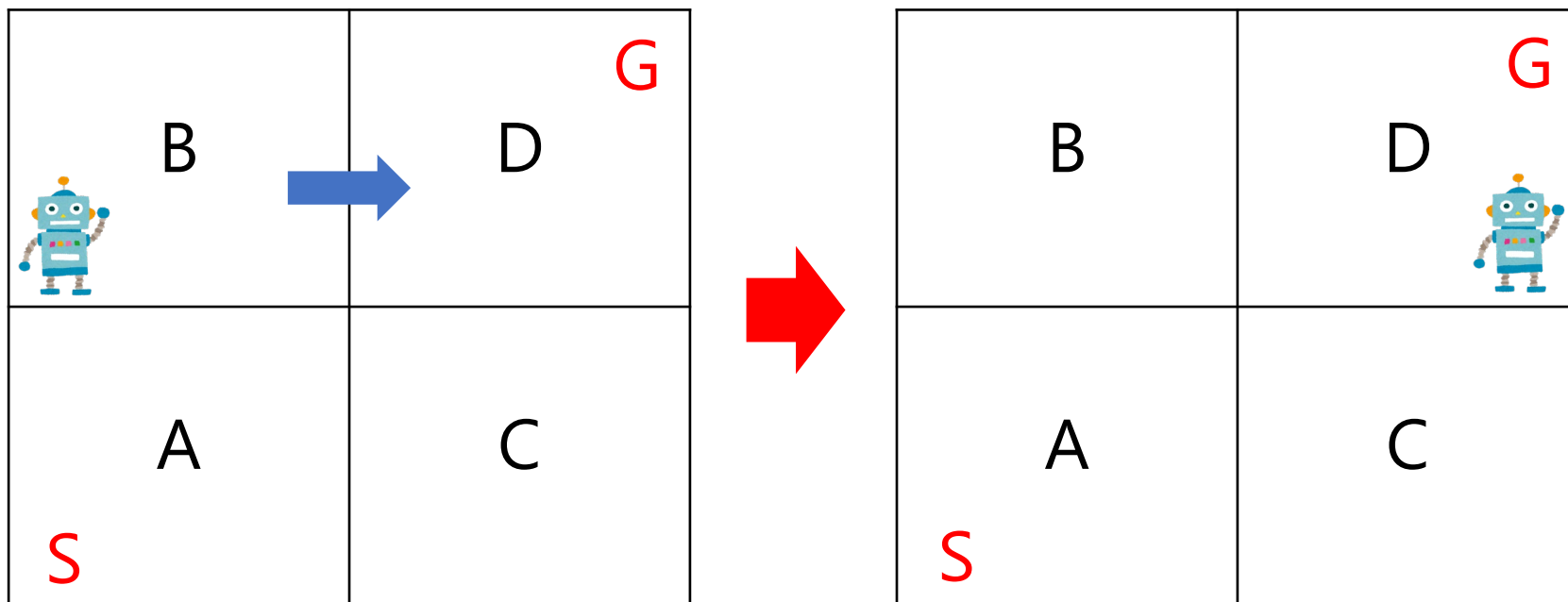
$$Q(A, B) \leftarrow 0 + 0.1 [0 + 0.95 * \text{Max}(0, 0) - 0] = 0$$

- Qテーブルの更新

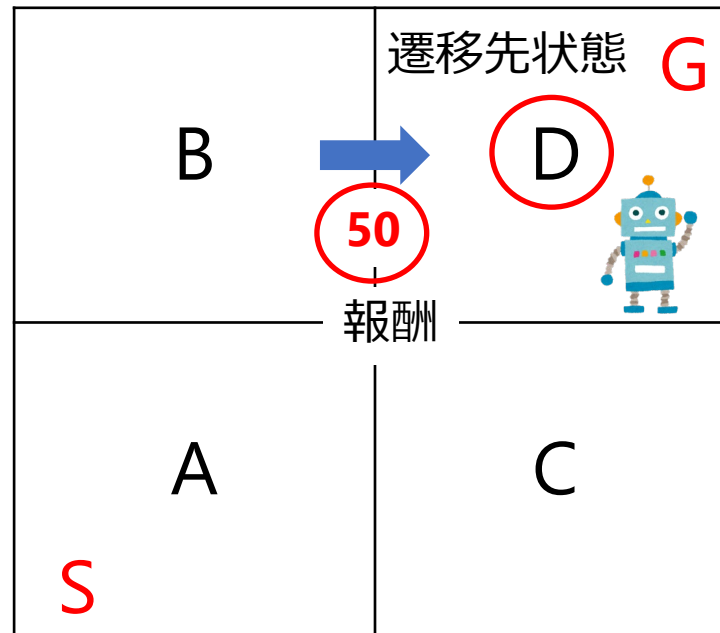
		行動			
状態	A	B	C	D	
	A	0	0	0	0
	B	0	0	0	0
	C	0	0	0	0
	D	0	0	0	0

- エージェントはゴールDに到達していないため続行

- 次のステップでBからDに移動する場合



- 環境から遷移先状態 s_{t+1} と報酬 r_{t+1} を受け取る
 - 遷移先状態はマスD(ゴール)
 - 報酬は50

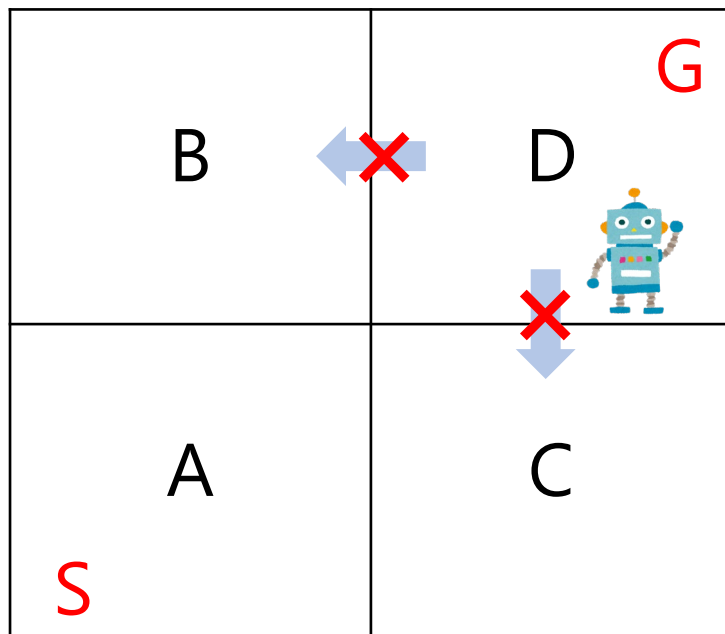


- 得られた遷移先状態 s_{t+1} と報酬 r_{t+1} を元に以下のように $Q(s_t, a_t)$ を更新

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

今回の場合で書くと

$$Q(B, D) \leftarrow Q(B, D) + 0.1[50 + 0.95 * 0 - Q(B, D)]$$



Dから移動可能な方向はない

- Qテーブルを試みる
 - $Q(B,D) \rightarrow 0$

		行動			
		A	B	C	D
状態	A	0	0	0	0
	B	0	0	0	0
	C	0	0	0	0
	D	0	0	0	0

$$Q(B, D) \leftarrow Q(B, D) + 0.1[50 + 0.95 * 0 - Q(B, D)]$$

$$Q(B, D) \leftarrow 0 + 0.1[50 + 0.95 * 0 - 0] = 5$$

- Qテーブルの更新

		行動			
		A	B	C	D
状態	A	0	0	0	0
	B	0	0	0	5
	C	0	0	0	0
	D	0	0	0	0

- ゴールへ到達する行動の価値が高くなった

- エージェントはゴールDに到達したためエピソードを終了

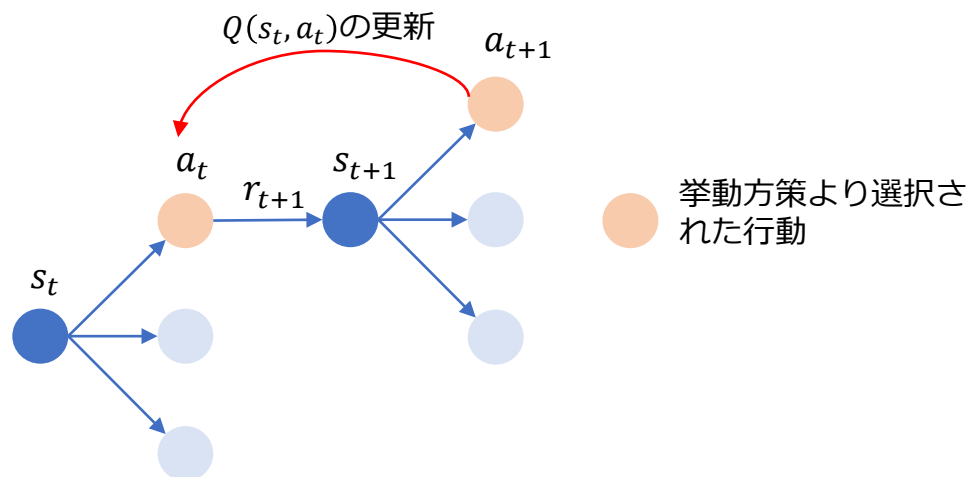
更新式

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

α : 学習率

γ : 割引率

- 価値ベース手法
- 状態行動対 s_t, a_t と遷移先の状態遷移対 s_{t+1}, a_{t+1} とその時に得られる報酬 r_{t+1} の5つの要素 $\{s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}\}$ を使用
 - 頭文字からSARSA
- $Q(s_t, a_t)$ の更新に方策により選択された実際の行動 a_{t+1} を使用
 - 行動選択に用いる方策と価値関数の更新に用いる方策が同じオンポリシー型手法



- 価値ベースと方策ベースを組み合わせた手法
- 行動を選択する行動器(Actor)と行動器により選択された行動を評価する評価器(Critic)

更新式

評価器(Critic)

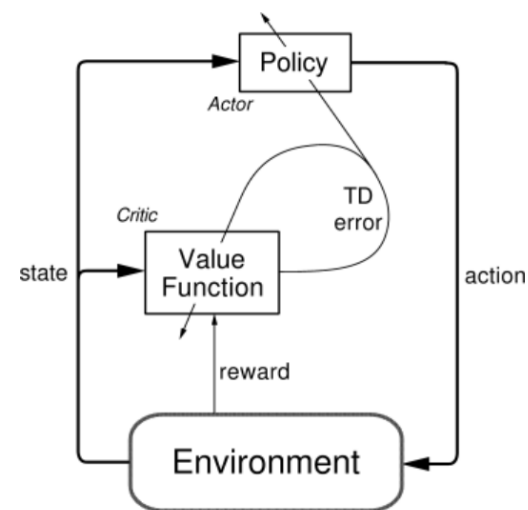
$$V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$$

行動器(Actor)

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$$

- 学習手順

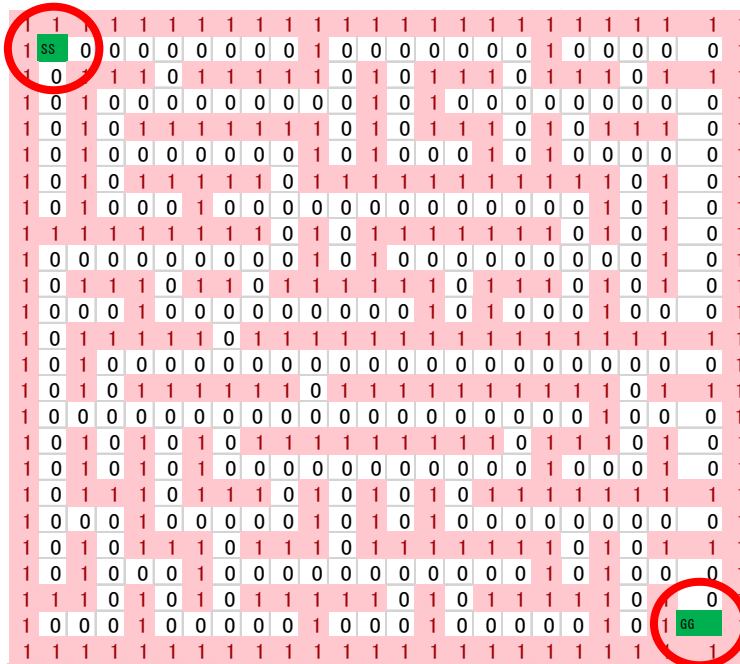
1. $Q(s, a)$ を初期化
2. 行動器は(Q値によって決定される)方策 π に従って行動 a_t を選択し実行
3. 評価器は環境から次状態 s_{t+1} と報酬 r_{t+1} を獲得
4. 評価器は次状態 s_{t+1} と報酬 r_{t+1} からTD誤差 $r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ を求めて価値関数 $V(s_t)$ を更新
5. 評価器で算出されたTD誤差を用いて行動器の方策 π を更新



- q_learn_onefile.c でQ学習の学習手順を追う
問題設定

- 以下のような迷路(SSがスタート、GGがゴール)
- エージェントは毎ステップ壁のない上下左右を移動
- ゴールに到達すると報酬(デフォルトは100)を獲得

スタート



迷路のデータ

ゴール

状態空間

```
26 // 座標
27 typedef struct position
28 {
29     int row;
30     int col;
31 } position;
```

行動空間

```
49 const position AROUND[DIRS] =
50 {
51     {-1, 0}, // 北
52     {0, +1}, // 東
53     {+1, 0}, // 南
54     {0, -1} // 西
55 };
```