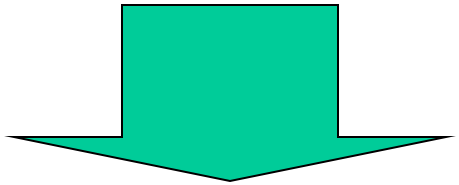
強化学習とは何か？
何に応用できるか？
どのようなしくみで動作するか？

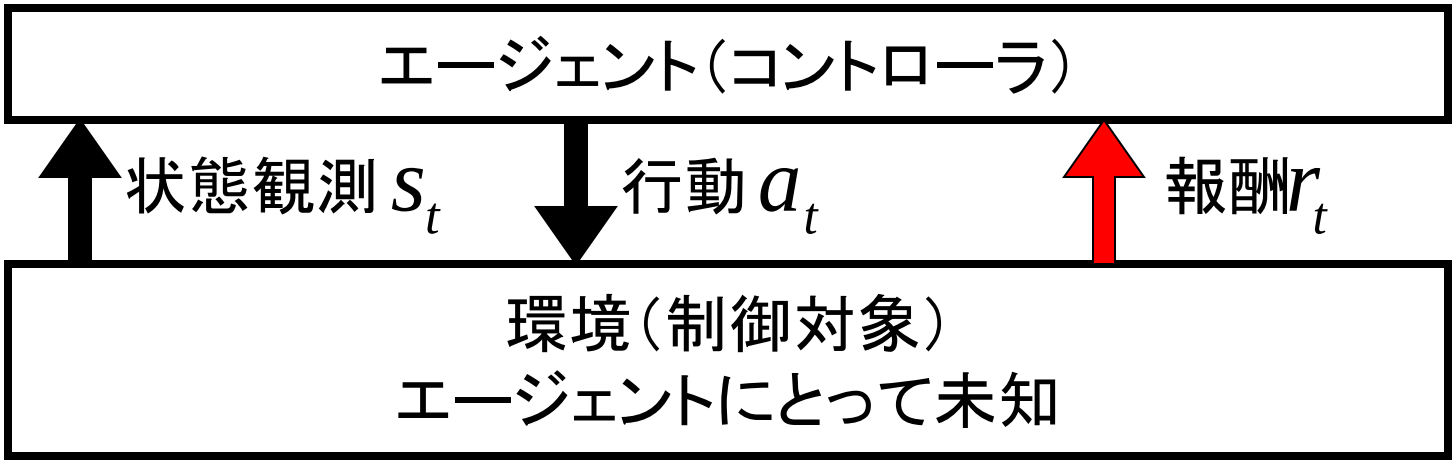
強化学習とは

- 試行錯誤を通じて環境に適応する学習制御の枠組み
- 学習目標:「報酬」の獲得が最大になるような「状態入力→行動出力」を獲得
- タスクを達成したら「報酬」を与えるように設定するだけで、複雑な制御規則を自動的に獲得



制御プログラミングの自動化

強化学習問題の定式化



●状態観測 → 行動選択 → （状態遷移）→ 報酬　　繰返し

●何回か状態遷移した後，やっと報酬を得る
　→ 多段決定過程 （報酬に遅れ）
　　目標状態に達したら大きな報酬
　　タスクを達成したら大きな報酬
　→ 正しい行動 ＝ 獲得報酬合計が最大の行動

●試行錯誤を繰り返し，より多くの報酬を得る行動を学習
　→ 報酬獲得最適化問題
　　「試行錯誤による適応」＝最適化のための探索過程
　　強化学習アルゴリズム ＝報酬獲得最適化手法

強化学習の理論的特徴

- 状態遷移に不確実性を伴う制御問題を理論的に扱う
- 離散的な状態遷移も含んだ段取り的な制御も理論的に扱う

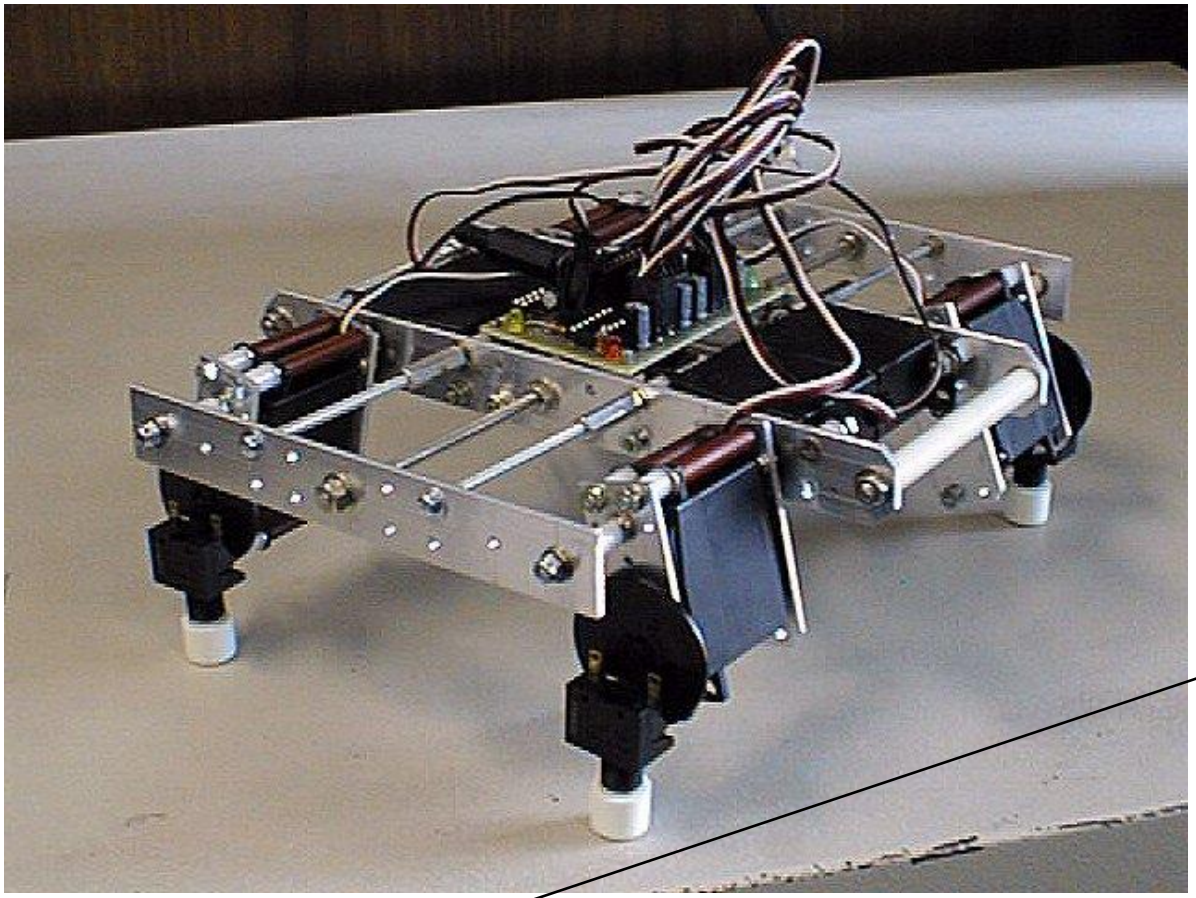
→　環境を確率過程（マルコフ決定過程）でモデル化

応用上の特徴

「何をすべきか」を「報酬」によって簡単に指示するだけで
「どのように実現するか」という制御規則を学習により自動的に獲得

1）制御プログラミングの自動化・省力化
2）ハンドコーディングよりも優れた解：
　特に不確実な要素（摩擦やガタ，振動，誤差など）や計測困難な未知パラメータが多い場合に有利
3）自律性と想定外の環境変化への対応：
　通信が物理的に困難だったり現象のダイナミクスが人間にとって早過ぎる場合や，機械故障など急激な変化やプラント経年変化など予め想定しておくことが困難な環境の変化に対し自動的に追従

強化学習の適用例：ロボットの歩行動作獲得(1)

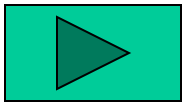


前進するために歩行

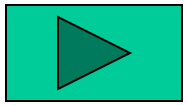
まっすぐ前進したら
最大の報酬を与える
ように設定

最大の報酬を得る
制御則を強化学習で
自動的に獲得

簡単な報酬設定から複雑な制御規則を
自動的に獲得
→　制御プログラミング自動化

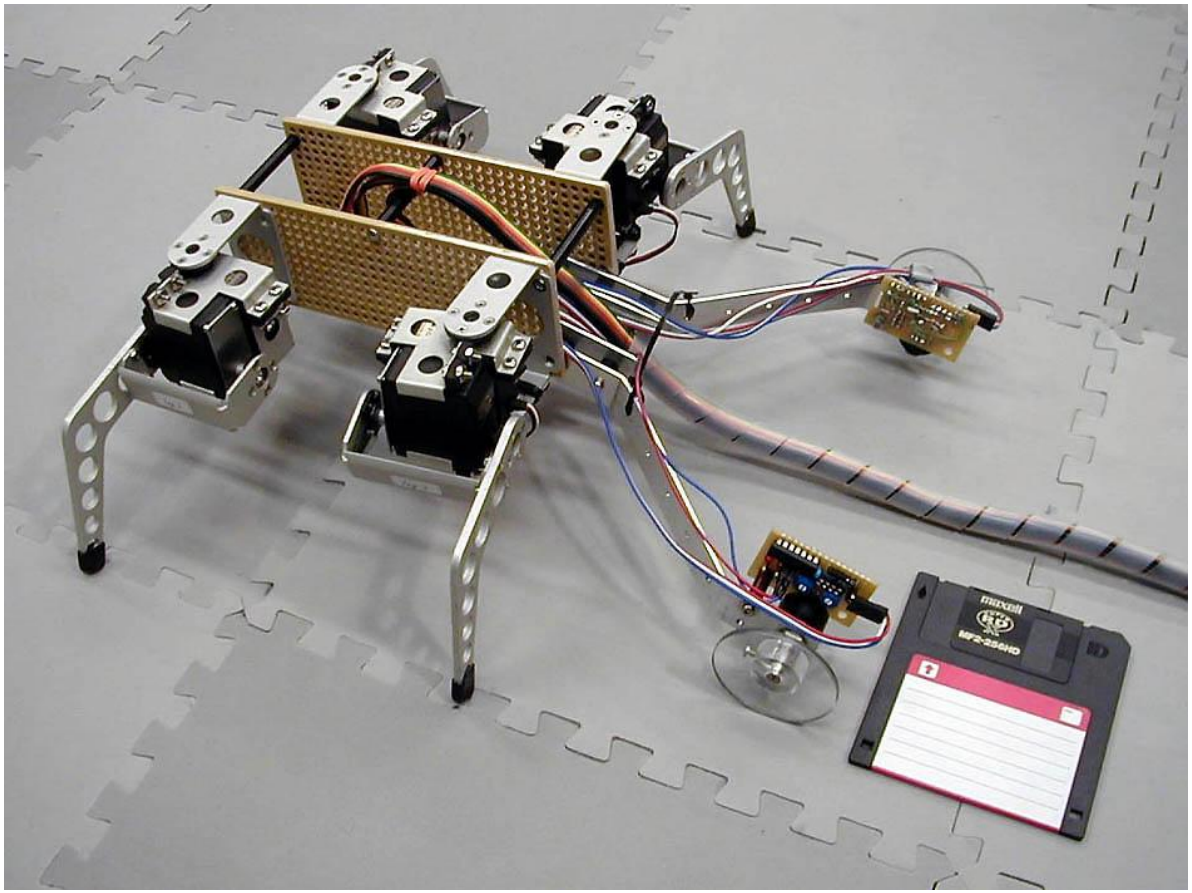


学習初期



学習途中

強化学習の適用例：ロボットの歩行動作獲得(2)
同一の学習器で異なるロボットを学習



感覚入力：関節の角度（8次元連続値）
行動出力：関節モータの角度（8次元連続値）
報酬：　　毎ステップの移動距離　（移動速度）

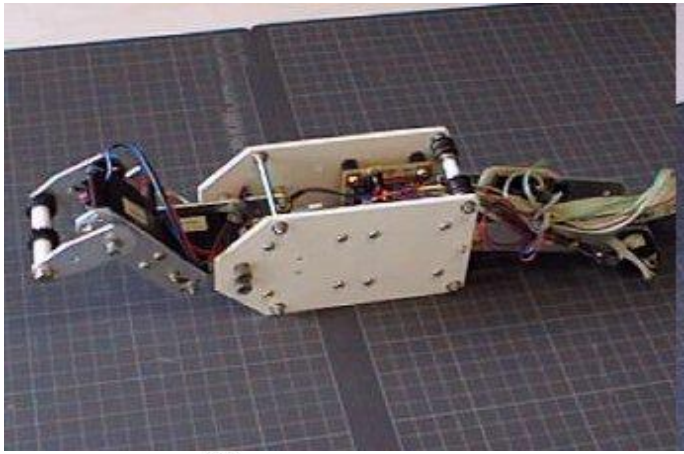
コントローラから見ると
同一のセンサとモータに
見える

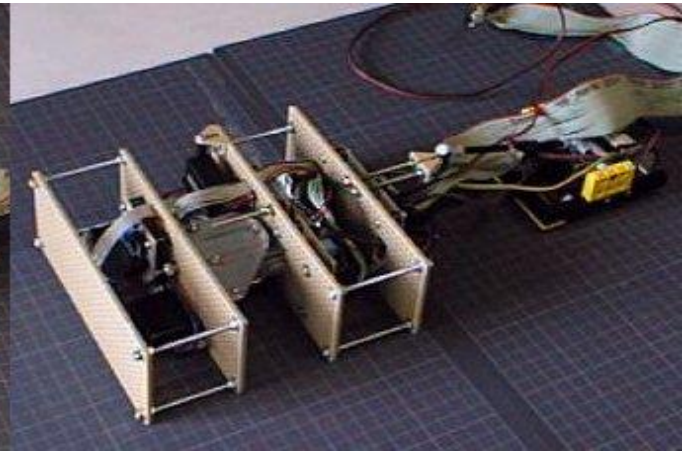
ダイナミクスは異なる

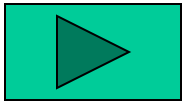
同一の学習器より各ロボットに適した
複数の制御プログラムを自動的に獲得

学習初期　学習後1　学習後2

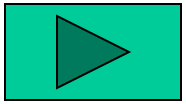
強化学習の適用例2：
新しい形状のロボットで制御規則を獲得



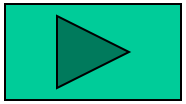




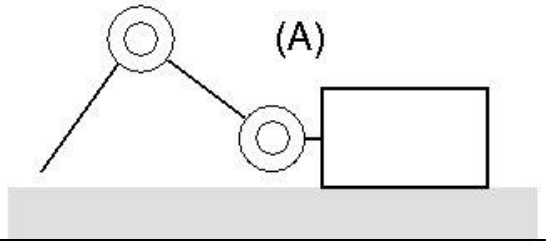
ロボットA

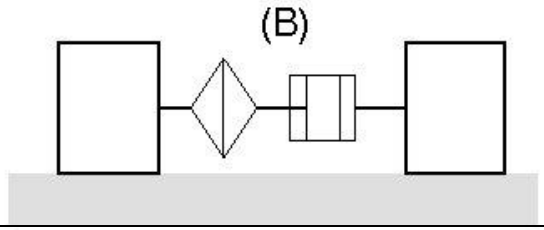


ロボットB

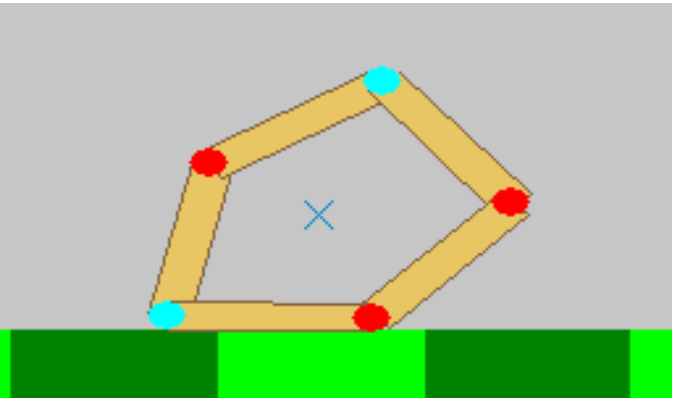


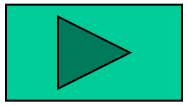
ロボットAバック



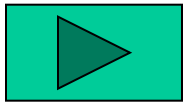


感覚入力：関節の角度（2次元連続値）
行動出力：関節モータの角度（2次元連続値）
報酬：　　毎ステップの移動距離　（移動速度）



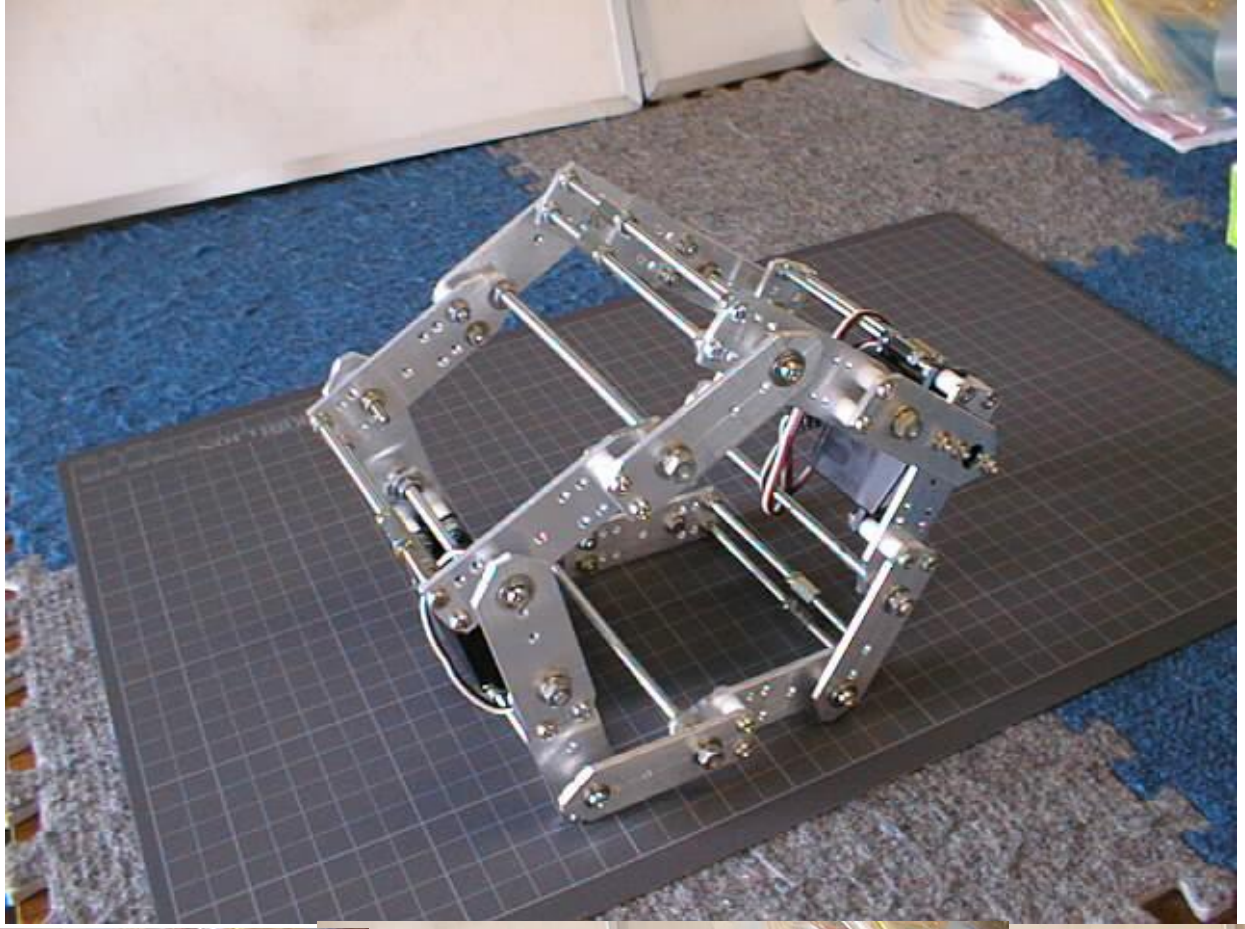


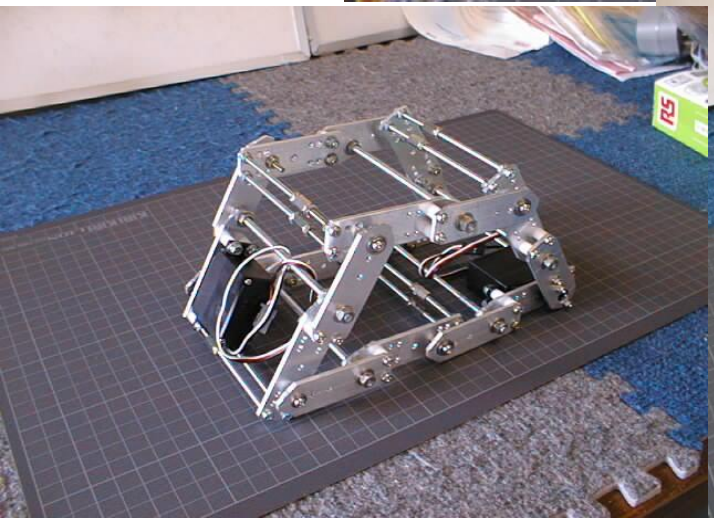
学習初期

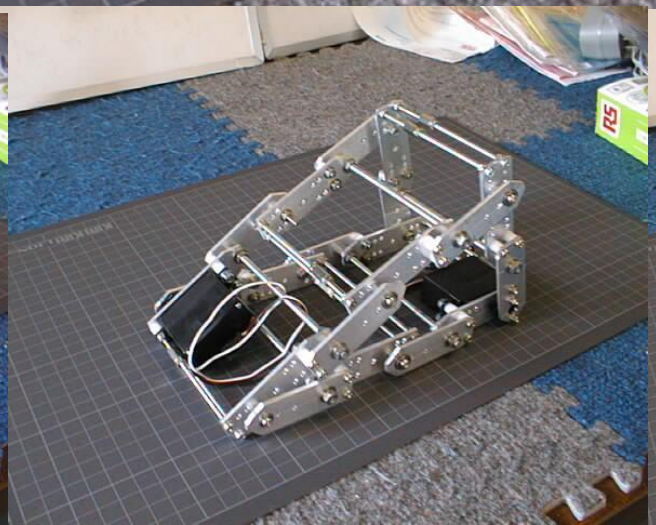


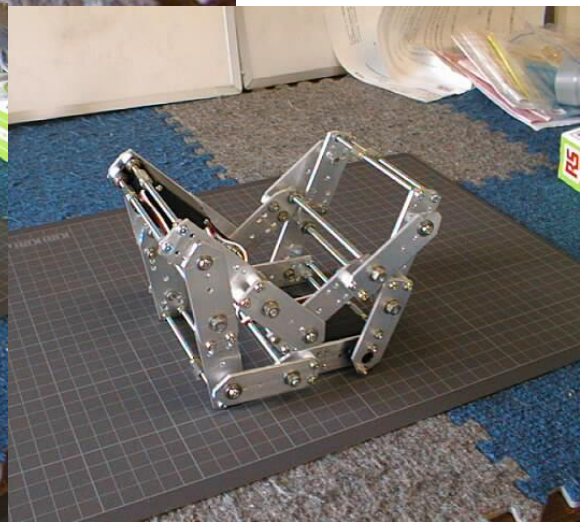
学習後

摩擦やガタを巧みに利用した制御規則を発見
→ハンドコーディングより優れた解









ロボット以外の応用例

●ネットワークルーティング
(Boyan et al. 1993, Subramanian et al. 1997, Kumer et al. 1999)

●セルラー通信システム（PHS）におけるバンド割当て
(Singh et al. 1997)

●生産システム管理（在庫管理） (Wang and Mahadevan 1999)

●エレベータ群制御 (Crites and Barto 1996)

●株式売買エージェント，Finance関連 (Neuneier 1998)

●データベースシステムにおけるタイムアウト時間間隔制御
(後藤，木村，小林 2001)

「環境」のモデル化

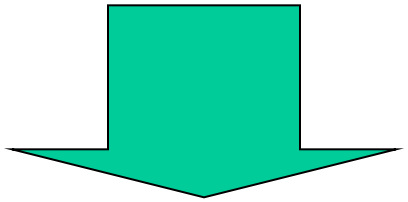
迷路問題の学習で成功した強化学習方法が4足ロボットや
ネットワークルーティングの学習問題で成功するとは限らない

そこで‥

- 迷路問題
- 4足ロボット
- ネットワーク問題
- 在庫管理問題
- Etc..

これらに共通する問題の特徴を
マルコフ決定過程という数理モデル
によって表現（モデル化）

マルコフ決定過程において学習することが示されたアルゴリズム



マルコフ決定過程で表現される全ての問題において学習が保証

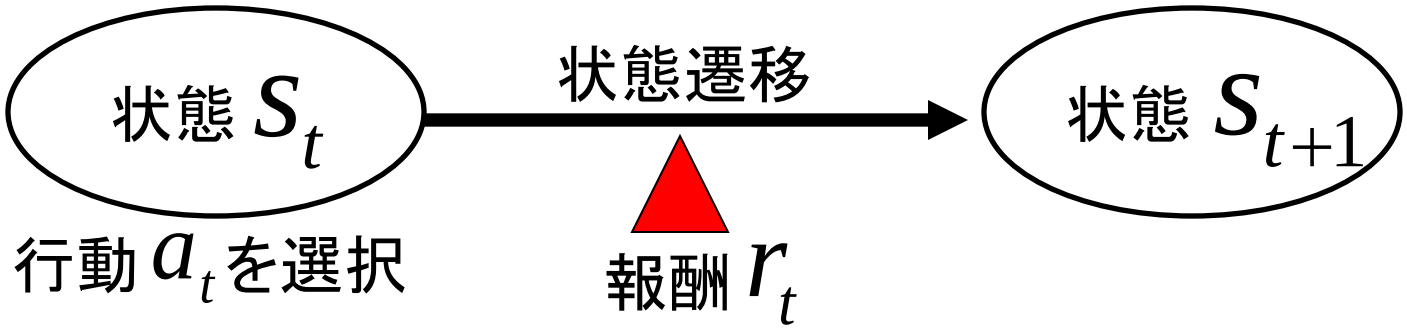
マルコフ決定過程(MDP)とは？

S　　：状態の集合

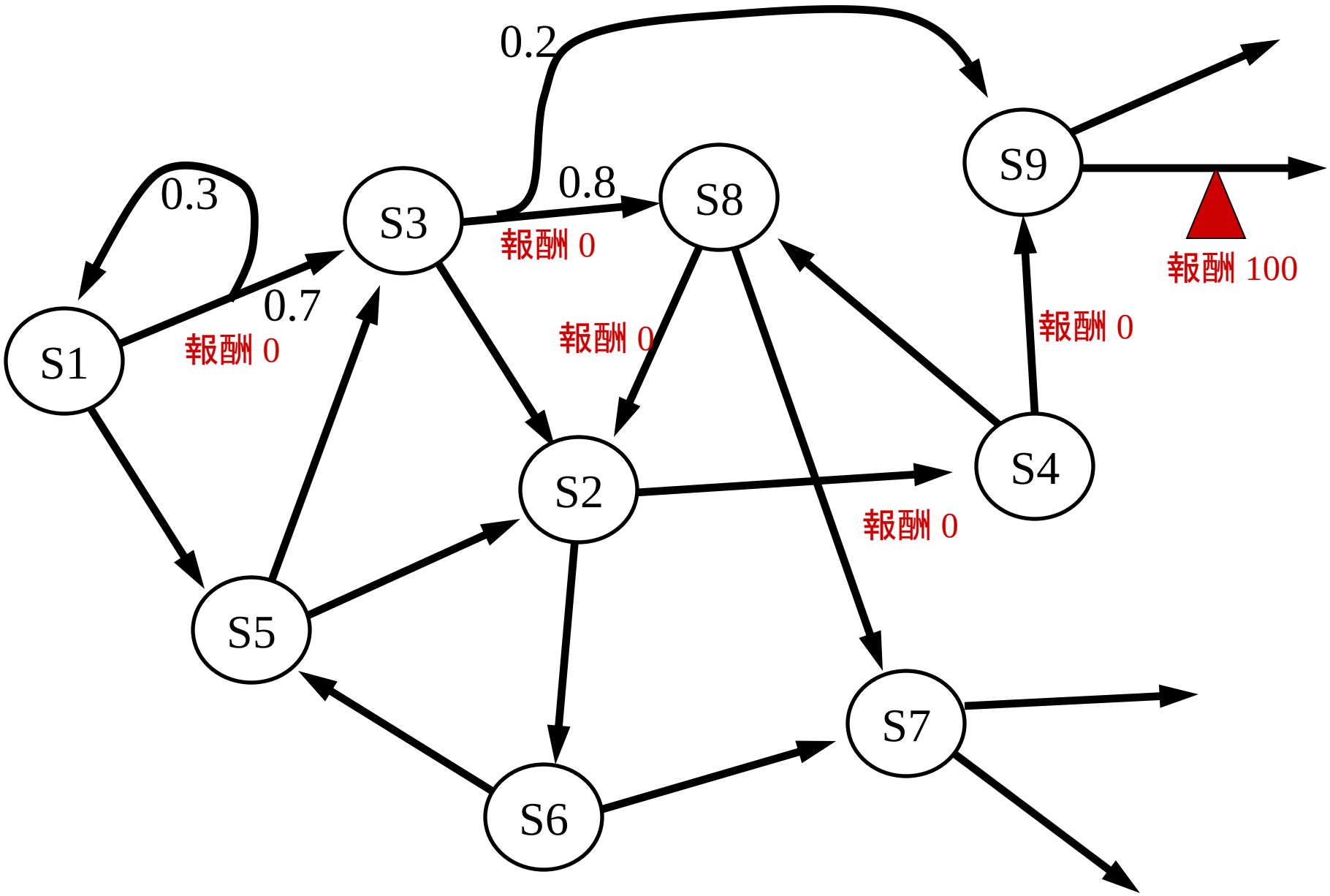
A　　：行動の集合

Pr(s'|s,a)　：状態sで行動aをとったときs'へ遷移する確率

$R^a(s,s')$　：状態sで行動aをとってs'へ遷移したときの報酬の期待値



マルコフ決定過程(MDP)の状態遷移



この他：迷路問題の例 (http://www.fe.dis.titech.ac.jp/~gen/edu/applets/MazeDemo.html)

終端状態のあるタスク(episodic tasks)

スタートからゴールまでを1エピソードとして、
独立なエピソードを何度も繰り返すタスク
例） 迷路問題，トランザクション処理

終端状態のないタスク(continuing tasks)

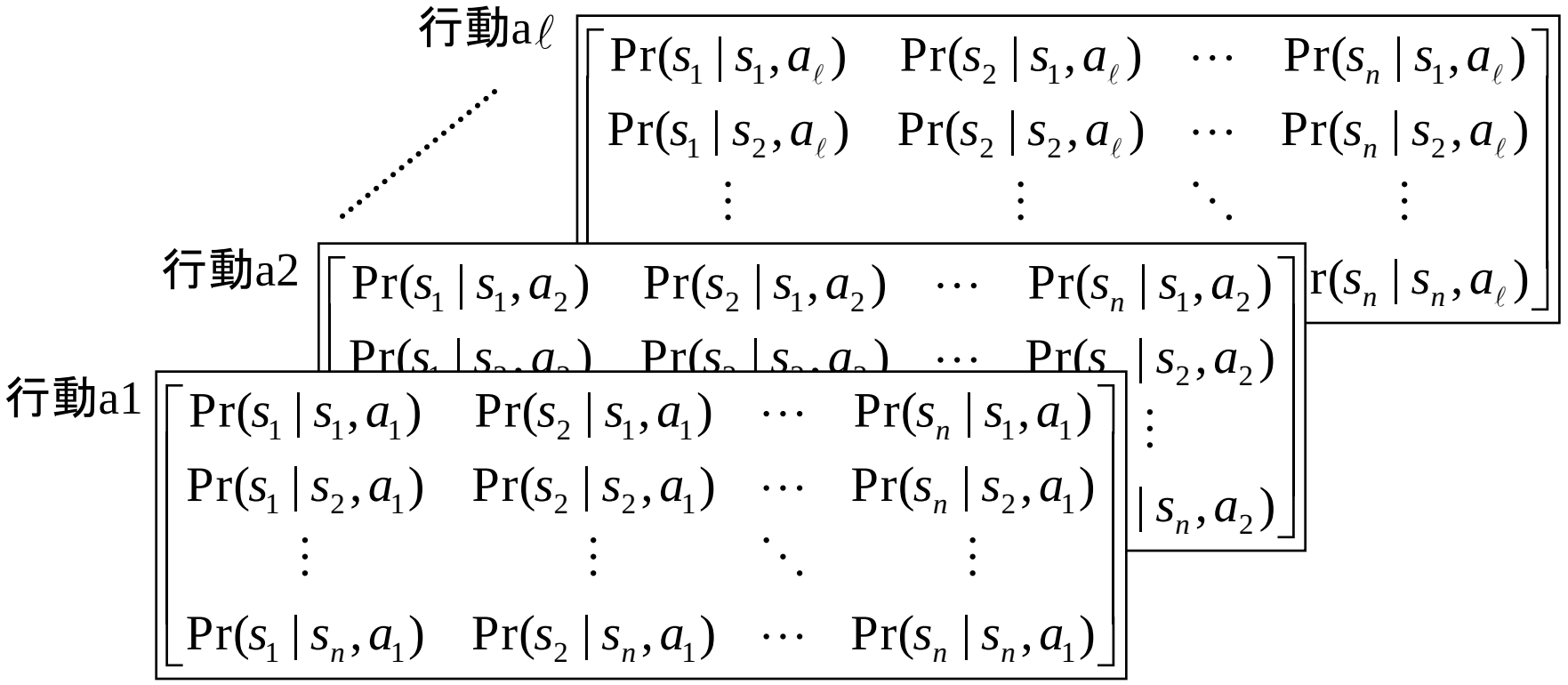
明示的なゴールが存在しないが，動きつづけることが必要なタスク
運転コストを最小化するような場合が多い
例） ロボットの歩行制御，上下水道制御，生産システム管理

MDPの状態遷移マトリクス

行動a1を選択する場合

MDPの状態遷移マトリクス

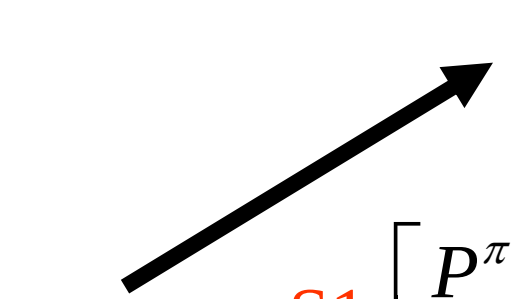
状態数n，行動数ℓのとき，
マトリクスの大きさは $n \times n \times \ell$



報酬関数も同様

エージェントの制御規則：政策

- 各状態Sで選択する行動aを規定
- ある政策πが定義されると，状態遷移確率はn×n正方行列になる



状態S’

状態S

$$\begin{array}{c} \\ S1 \\ S2 \\ \cdot \\ Sn \end{array} \begin{array}{c} \begin{array}{cccc} S1 & S2 & \cdots & Sn \end{array} \\ \begin{bmatrix} P^{\pi}(s_1,s_1) & P^{\pi}(s_1,s_2) & \cdots & P^{\pi}(s_1,s_n) \\ P^{\pi}(s_2,s_1) & P^{\pi}(s_2,s_2) & \cdots & P^{\pi}(s_2,s_n) \\ \vdots & \vdots & \ddots & \vdots \\ P^{\pi}(s_n,s_1) & P^{\pi}(s_n,s_2) & \cdots & P^{\pi}(s_n,s_n) \end{bmatrix} \end{array} = \mathbf{P}^{\pi}$$

各ステップで得る
報酬の期待値は
1×nの行列になる

$$\mathbf{R}^{\pi} = \begin{bmatrix} R^{\pi}(s_1) \\ R^{\pi}(s_2) \\ \vdots \\ R^{\pi}(s_n) \end{bmatrix}$$

←状態S1の報酬の期待値

←状態S2の報酬の期待値

←状態Snの報酬の期待値

最適性の定義

報酬合計の期待値を最大化する政策を見つける

ただし，

１－γの確率で停止する場合の報酬合計：割引報酬

割引率γ

１ステップあたりγの確率で活動を続ける

γ→１　長期的な利益最大化

γ→０　目先の利益最大化

最適性の定義（もう少し形式的に）

政策π：各状態における行動選択確率
以下の**割引報酬の合計**を最大化する政策を見つける

$$\sum_{t=0}^{\infty} \gamma^t r_t$$ ただしγは割引率

報酬を割引く理由：
1）未来の報酬はあてにならない
　→未来に得る報酬を現時点では割引いて評価
γ＝1のとき：　単なる報酬合計
γ＜1のとき：　1－γの確率で停止する場合の報酬合計
2）計算の利便性

MDPの最適性

MDPにおいて定常政策πをとるとき，
割引報酬合計の期待値(value)は状態Sの関数になる：

$$V(s) = E\left\{ \sum_{t=0}^{\infty} \gamma^t r_t \mid s_{t=0} = s \right\}$$

すなわち，**政策の評価値はベクトル**

$$\left(V(s_1), V(s_2), \cdots V(s_n) \right)$$

この全要素を最大化する政策が最適政策 π^*
そのときのvalueが最適value関数 V^*

$1-\gamma$の確率で停止する場合の報酬合計評価はベクトル

$$\mathbf{V}^{\pi} = \begin{bmatrix} V^{\pi}(s_1) \\ V^{\pi}(s_2) \\ \vdots \\ V^{\pi}(s_n) \end{bmatrix}$$

←S1からスタートした場合の報酬合計の期待値

←S2からスタートした場合の報酬合計の期待値

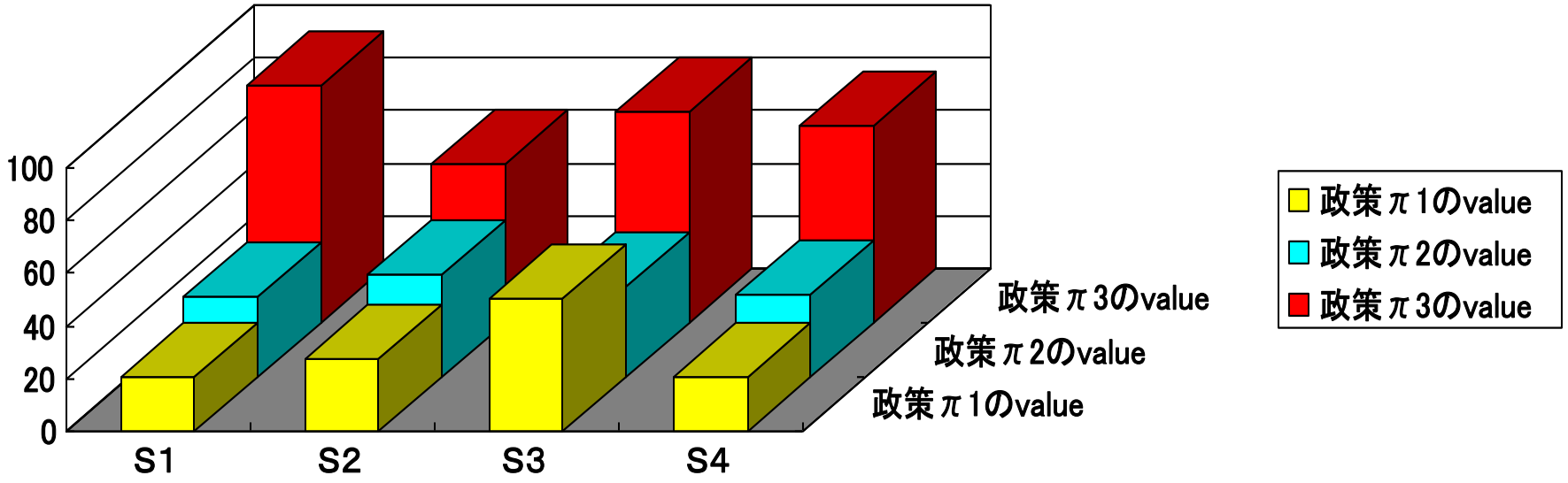
←Snからスタートした場合の報酬合計の期待値

$$= \mathbf{R}^{\pi} + \gamma \mathbf{P}^{\pi}\mathbf{R}^{\pi} + \gamma^2 (\mathbf{P}^{\pi})^2 \mathbf{R}^{\pi} + \cdots$$

$$= \mathbf{R}^{\pi} + \gamma \mathbf{P}^{\pi}\mathbf{V}^{\pi}$$

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

評価値(value)がベクトルの場合の政策の比較方法



- 政策π1とπ2は状態によって評価値の大小関係が入れ替わる→政策の良し悪しははっきり決められない
- 政策π3は全てのvalueの要素が他の政策の値をドミネートしている→π3が最も良い政策
- MDPでは他の政策のvalueをドミネートする最適政策が必ず存在

代表的な強化学習アルゴリズム

１） Temporal Difference (TD)法
状態の評価値 (value)のみを学習するアルゴリズム

２） Actor-Criticアルゴリズム
状態の評価値をTD法で学習しながら政策を改善する

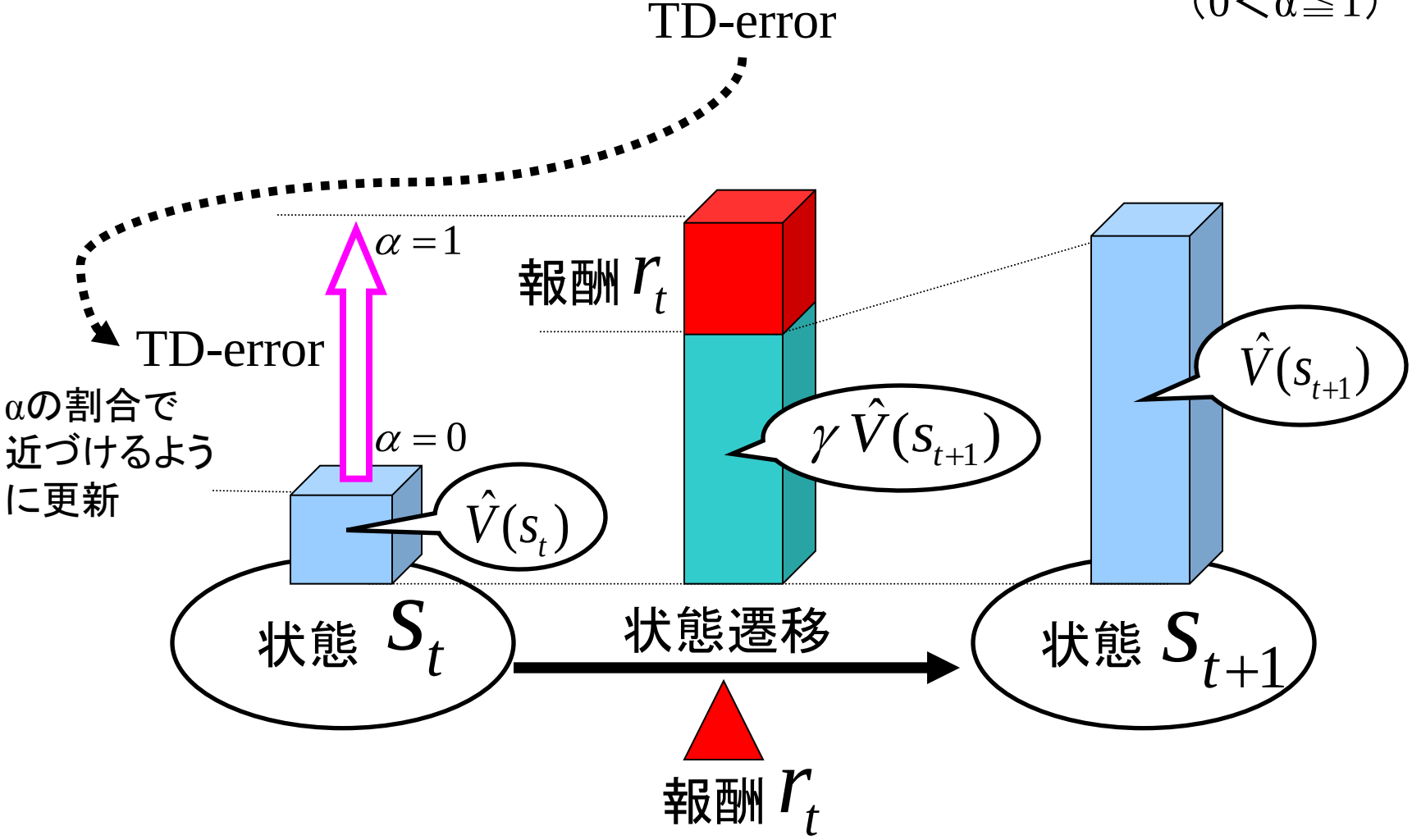
３） Q-learning
行動の評価値を学習
最大の評価値の行動をとれば最適政策になる

1）Value学習アルゴリズム： Temporal Difference (TD)法

ある政策πをとるとき，以下の式で逐次更新：

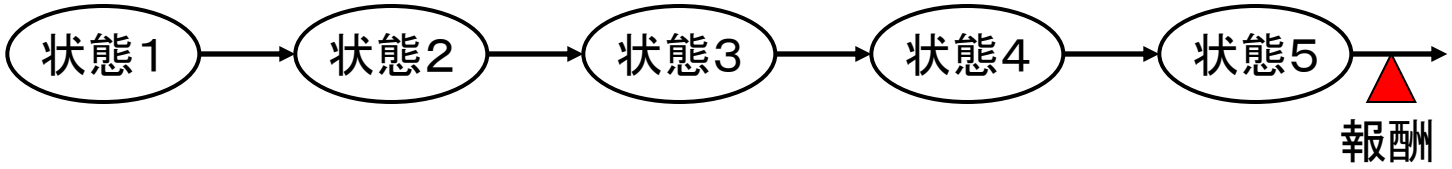
$$\hat{V}(s_t) \leftarrow \hat{V}(s_t) + \alpha\underbrace{\left(r_t + \gamma\hat{V}(s_{t+1}) - \hat{V}(s_t)\right)}_{\text{TD-error}}$$

γは割引率
（$0 \leqq \gamma \leqq 1$）
αは学習率
（$0 < \alpha \leqq 1$）

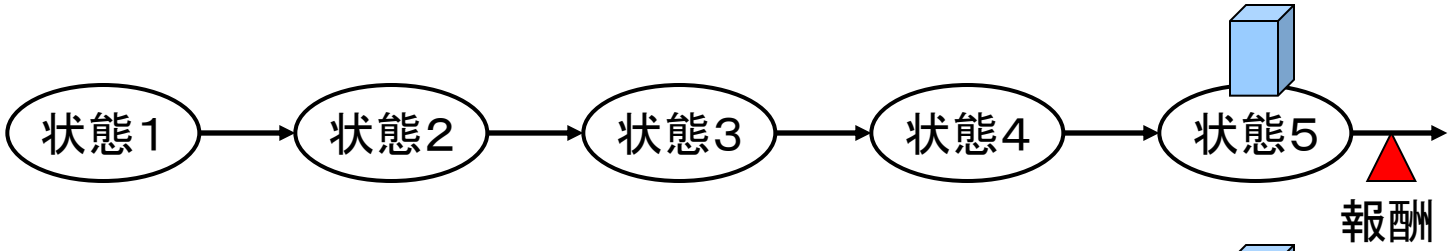


TD法による学習例：直線迷路におけるvalueの伝播

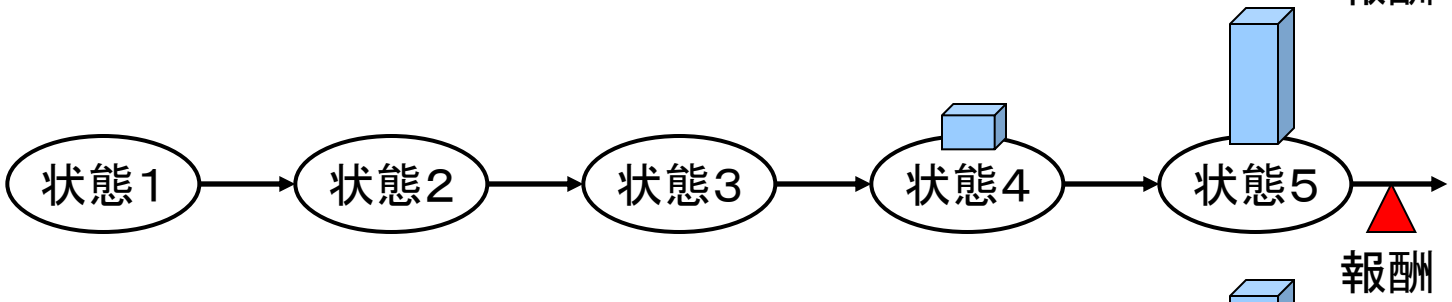
1）学習初期の value



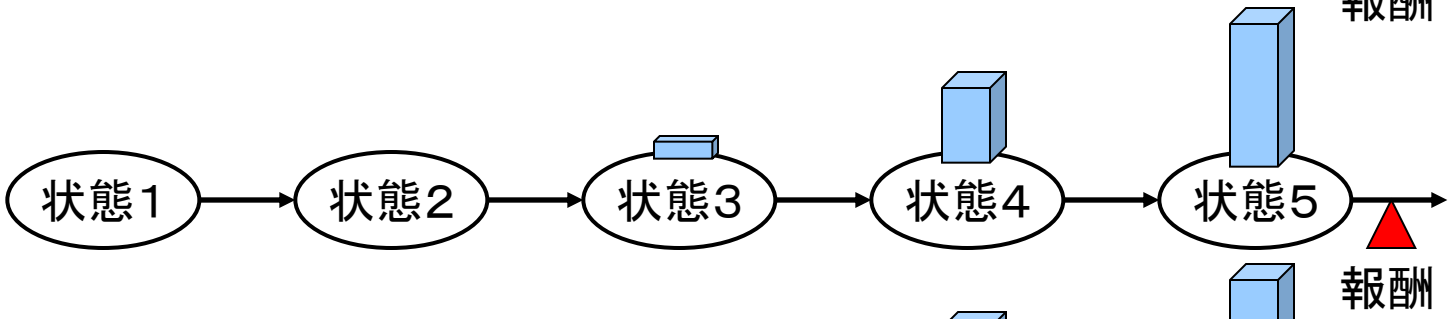
2）エージェントが状態１→５を１回通過後の value



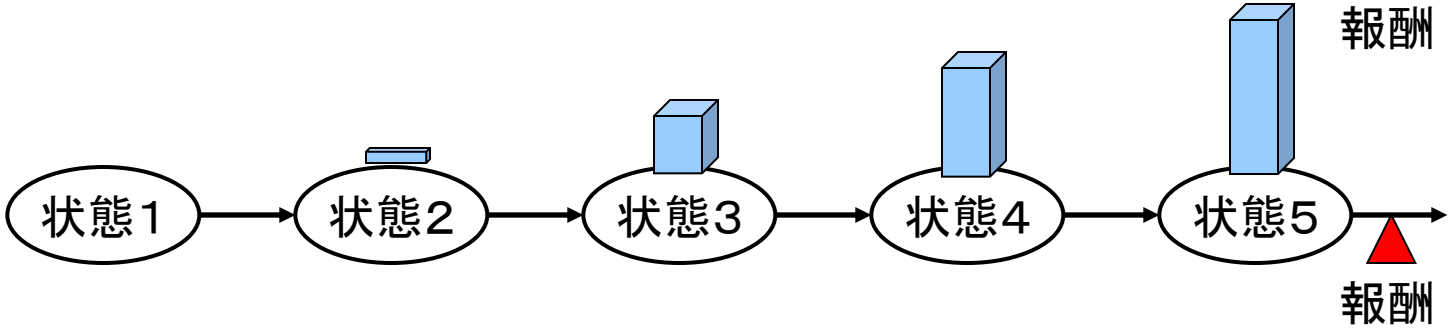
3）状態１→５を２回通過後の value



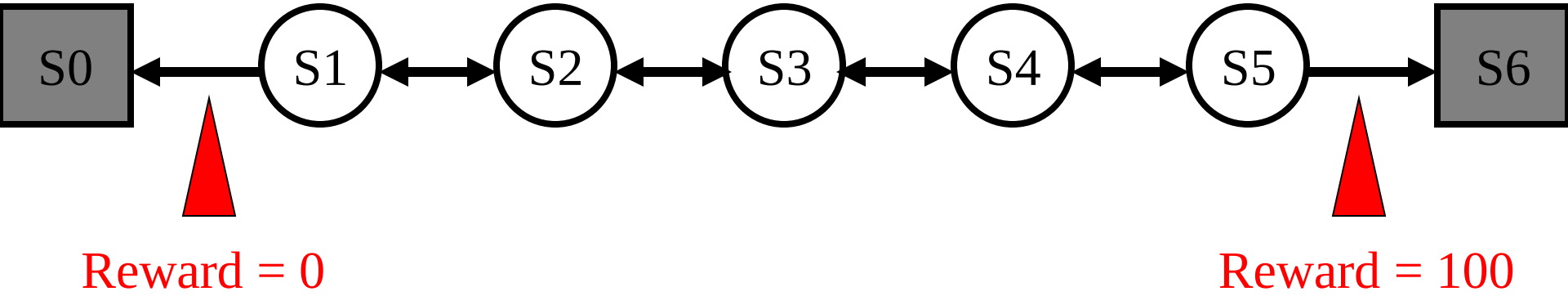
4）3回通過後の value



5）4回通過後の value



TD法による学習例:ランダムウォークのマルコフ過程 (Sutton98)

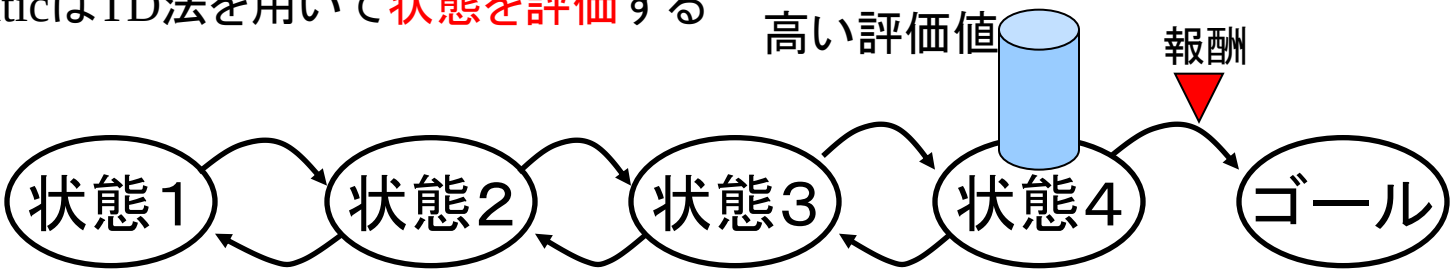


- エージェントはS1からS5のいづれかの状態からスタート
- 上図の環境を右または左へ0.5の確率で遷移
- 状態S0またはS6へ到達すると終了
  - S1からS0へ遷移すると報酬 0
  - S5からS6へ遷移すると報酬 100

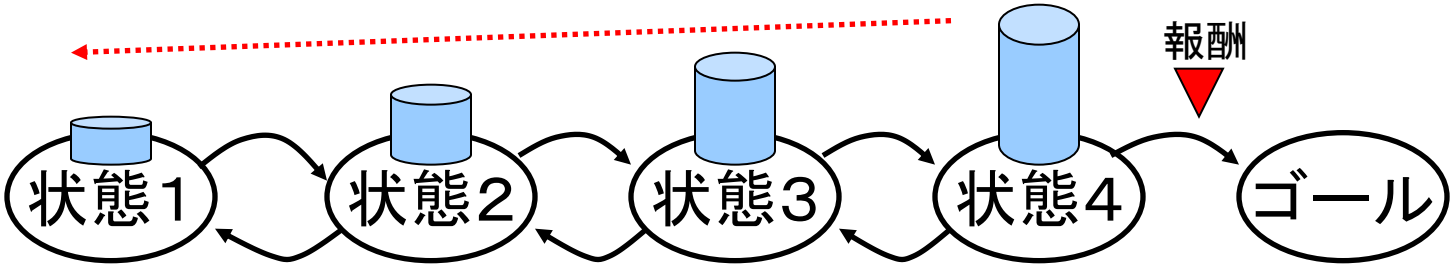
状態の評価関数(value function) V(Si)
状態Siからスタートしたエピソードで得られる報酬の期待値

2）強化学習アルゴリズム：Actor-Critic

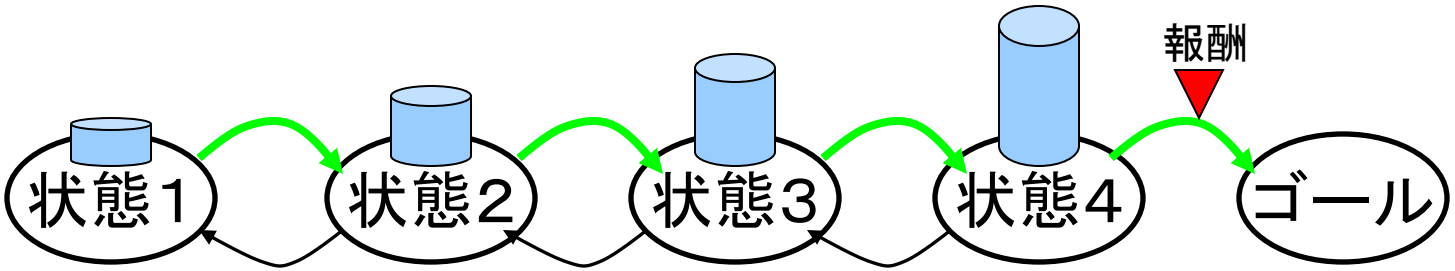
1）CriticはTD法を用いて状態を評価する



状態遷移を繰り返すうちにcriticは状態の評価値を伝播して学習する



2）ActorはTD-errorを手がかりに行動を学習
TD-errorが大きな正の値になる状態へ遷移する行動は良い行動＝選択確率を高く

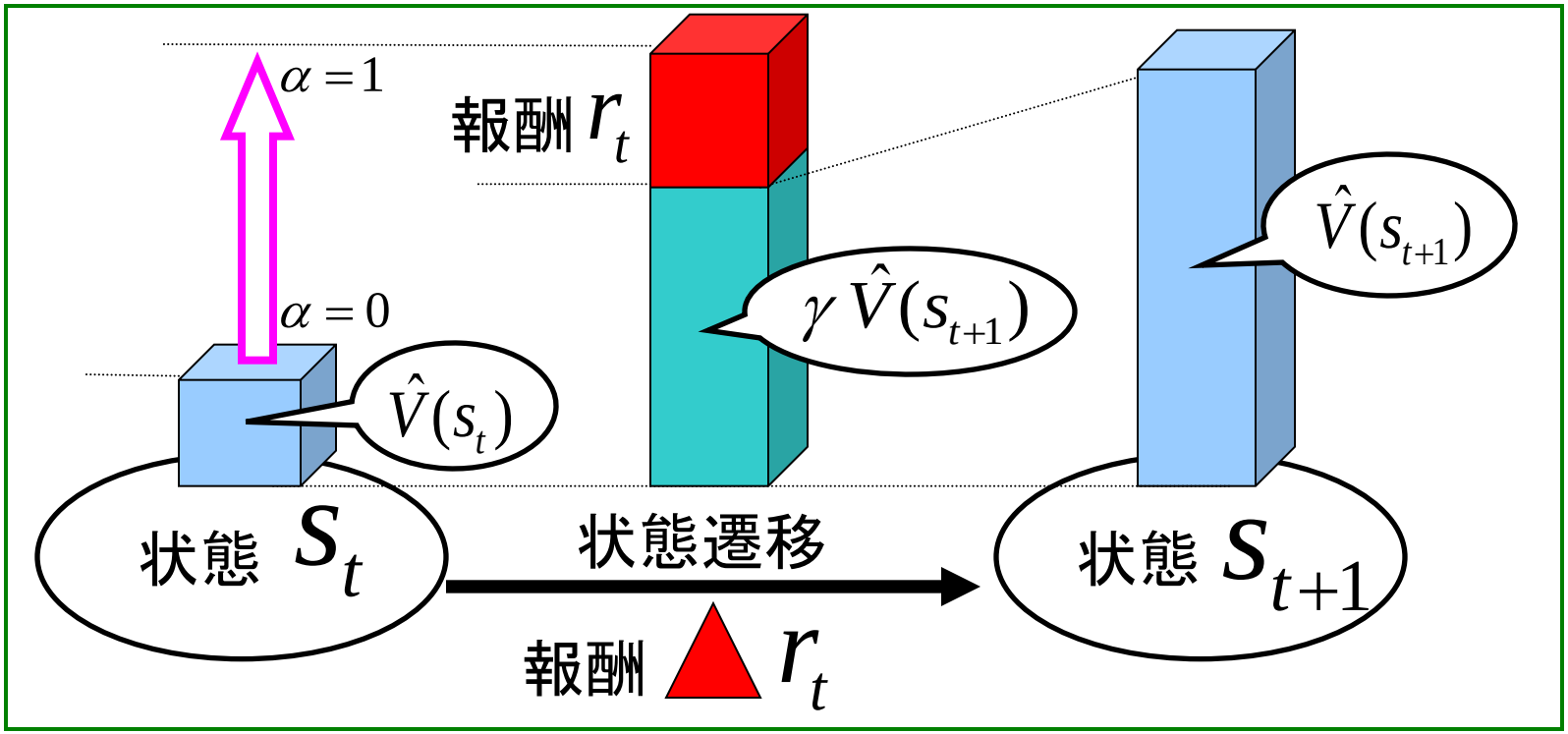


Actor-Criticの処理手順

1） Actorは確率的政策π(a|s,Θ)に従って行動 a を選択

2） （状態遷移）

3） CriticはTD法によってTD-errorを計算し，valueを更新



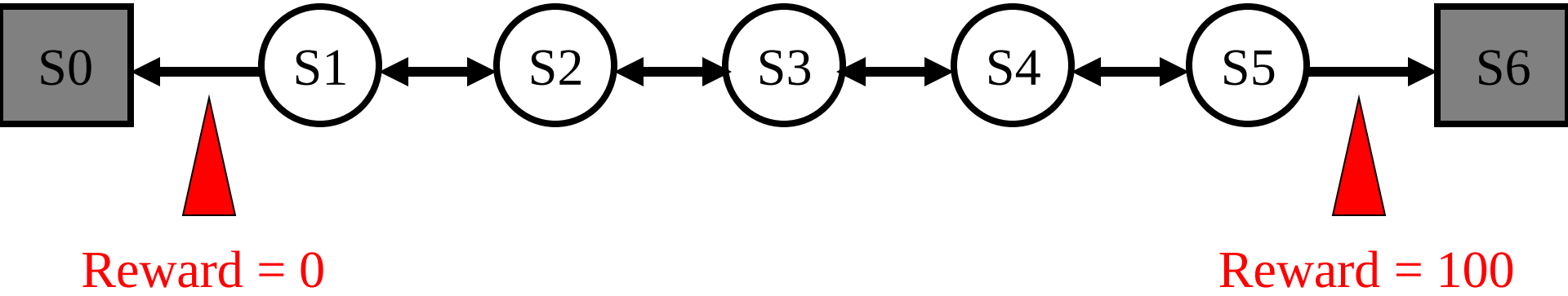
4） (TD-error) > 0 ならば実行した行動 a は良い行動

行動 a の選択確率を増やす

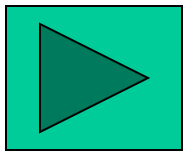
(TD-error) < 0ならば実行した行動 a は悪い行動

行動 a の選択確率を減らす

Actor-Criticによる学習例：Sutton98 に行動を付加



- エージェントはS1からS5のいづれかの状態からスタート
- 行動A0を選択すると確率0.75で左へ移動，確率0.25で右へ移動
- 行動A1を選択すると確率0.25で左へ移動，確率0.75で右へ移動
- 状態S0またはS6へ到達すると終了
  - S1からS0へ遷移すると報酬 0
  - S5からS6へ遷移すると報酬 100



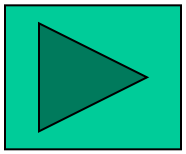
迷路問題

状態の評価関数(value function) V(Si)
状態Siからスタートしたエピソードで得られる報酬の期待値

Actor-Criticの特徴

- 原理や仕組みが非常に簡単で実装が楽
- 状態空間が行動空間が連続値でも実装が簡単
  - 状態評価の推定に関数近似を用いる
  - 行動選択確率（政策関数）にガウス分布などを使う

ロボットなど大規模問題に対して最も有望

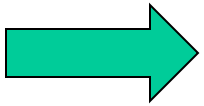


４足ロボット

Actor-Criticの問題点

政策で示される行動選択確率で行動が実行される
行動を選択した後でしか行動の評価ができない

事前に行動の評価値を得られないか？
ランダムな行動選択から最適な政策を得られないか？



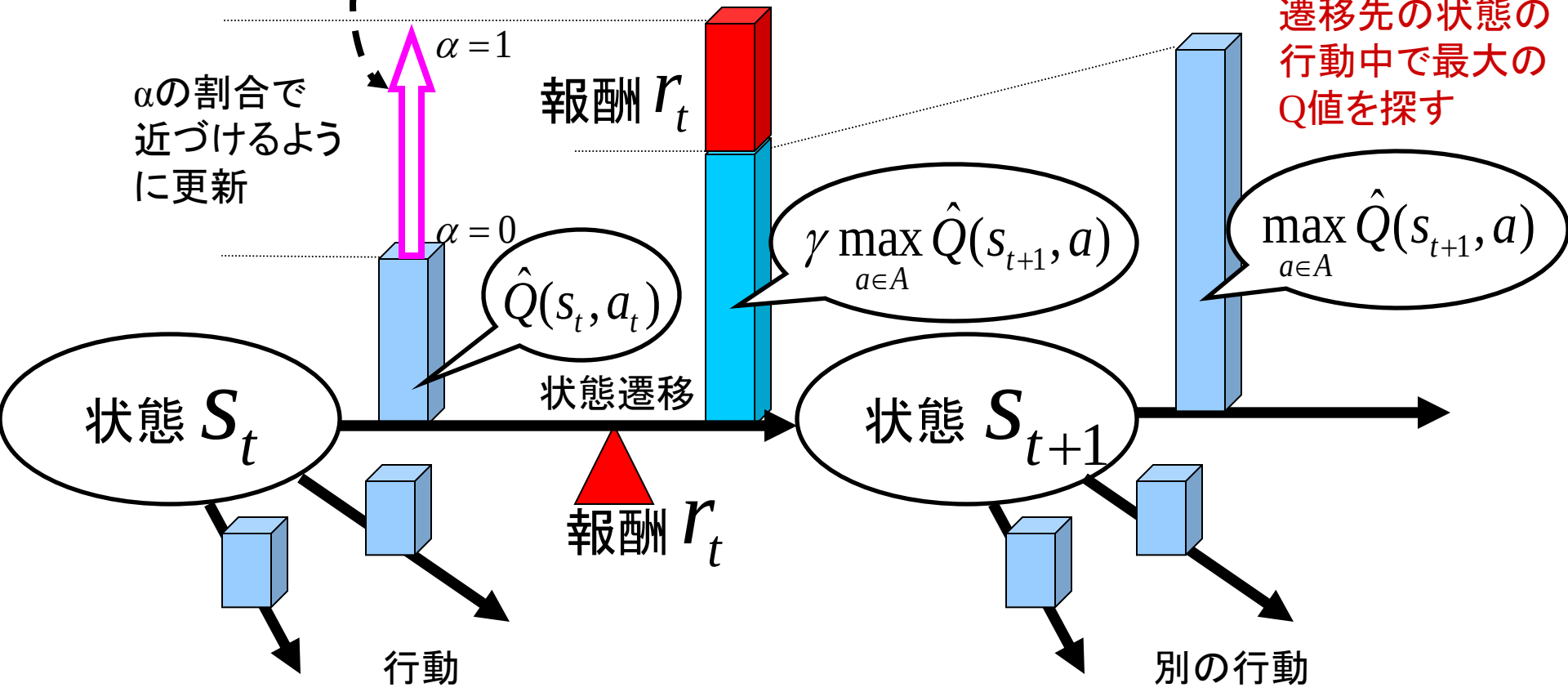
Q-learningアルゴリズム

3) 強化学習アルゴリズムQ-learning:　以下の式で逐次更新

$$\hat{Q}(s_t,a_t) \leftarrow \hat{Q}(s_t,a_t) + \alpha \left[r_t + \gamma \max_{a \in A} \hat{Q}(s_{t+1},a) - \hat{Q}(s_t,a_t) \right]$$

状態Stで実行した
行動のQ値を更新

γは割引率（$0 \leqq \gamma \leqq 1$）
αは学習率（$0 < \alpha \leqq 1$）



Q-learningの収束定理（Watkins92）

行動選択において全行動を十分な回数選択し，かつ学習率αが

$$\sum_{t=0}^{\infty} \alpha'(t) \to \infty \qquad \text{and} \qquad \sum_{t=0}^{\infty} \alpha(t)^2 < \infty$$

を満たす時間 t の関数になっているとき，アルゴリズムで得るQ値は確率１で最適政策の評価値に概収束する。
ただし環境はエルゴート性を有するMDP。

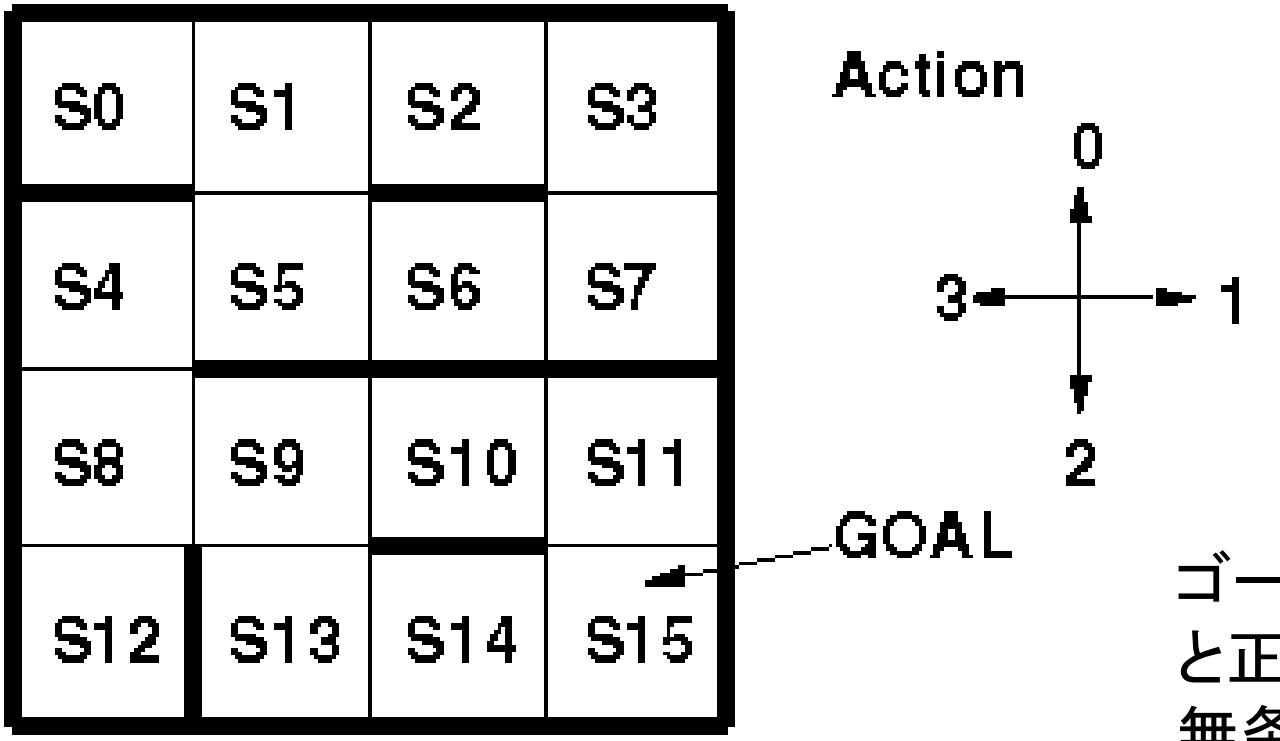
Q-learningの行動選択方法

上記の定理は条件を満たせばどんな行動選択方法でも成り立つ。
学習とともになるべく行動を改善していく行動選択が望ましい

１）ε-greedy選択：
　εの確率でランダム，それ以外は最大Q値の行動

２）ボルツマン選択：
exp(Q(s,a)/T)に比例した割合で行動選択
　ただし温度パラメータTは時間とともにT→0へ

Q-learningの動作例：迷路問題



行動は4種類
上下左右へ
1マス移動

ゴールへ到達する
と正の報酬が与えられ
無条件でS0へジャンプ

問題の説明

学習のデモ

現在取り組まれている強化学習研究

- **連続状態・連続行動空間の扱い**
  - 価値関数の近似表現方法 (Sutton98, Santamaria98, etc.)
  - 政策の表現方法
  - 行動選択方法 (木村2001, 山下2001, etc.)
- 状態観測の不完全性（隠れ状態）の扱い：非マルコフ問題
  - マルコフ決定過程モデルを拡張：POMDP(Singh94, etc.)
  - POMDPでの強化学習法(Jaakkola94,Kimura95, etc.)
- 膨大な状態や行動を扱うための階層化(Dietterich98,Sutton98,99,etc.)
- リスクの回避：報酬の分散を考慮(Neuneier99, Sato2001, etc.)
- マルチエージェント：複数のエージェントが同時に学習・協調／競合
  - (Schneider99, 宮崎2000, etc.)
- エキスパートの知識を利用→試行錯誤を極力減らす
- 環境のモデルを構築(Sutton90,Schneider96,Moore95,Boyan99)
  - 報酬だけが変化する場合でも以前の経験を生かす

参考文献

●Boyan, J. A. and Littman, M. L.: Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach, Advances in Neural Information Processing Systems 6, pp. 671-678 (1993).

●Subramanian, D., Druschel, P., and Chen, J.: Ants and Reinforcement Learning: A Case Study in Routing in Dynamic Networks, Proceedings of the 15th International Joint Conference on Artificial Intelligence, pp.832--838 (1997).

●Kumar, S., and Miikkulainen, R.: Confidence Based Dual Reinforcement Q-Routing: An adaptive online network routing algorithm, Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99), pp.758--763 (1999).

●Singh, S., and Bertsekas, D.: Reinforcement Learning for Dynamic Channel Allocation in Cellular Telephone Systems, Advances in Neural Information Processing Systems 9, pp.974--980 (1997).

●Wang, G., and Mahadevan, S.: Hierarchical Optimization of Policy-Coupled Semi-Markov Decision Processes, Proceedings of the 16th International Conference on Machine Learning, pp.464--473 (1999).

●Crites, R. H., and Barto, A. G.: Improving Elevator Performance Using Reinforcement Learning, Advances in Neural Information Processing Systems 8, pp. 1017--1023 (1995).

●Neuneier, R.: Enhancing Q-Learning for Optimal Asset Allocation, Neural Information Processing Systems 10, pp.936--942 (1998).

●後藤 正徳，木村 元，小林 重信:トランザクション処理におけるタイムアウト間隔の学習，計測自動制御学会 第28回知能システムシンポジウム資料 pp.257--262 (2001).

その他質問や資料等が必要な場合、
Gen@fe.dis.titech.ac.jp または
http://www.fe.dis.titech.ac.jp/~gen/indexj.html