強化学習の基礎

木　村　元*

*九州大学 大学院工学研究院，福岡県福岡市西区元岡 744
*Graduate School of Engineering, Kyushu University, Motooka 744 Nishi-ku Fukuoka city 819-0395, Japan
*E-mail: kimura@nams.kyushu-u.ac.jp

キーワード：強化学習 (reinforcement learning), マルコフ決定過程 (Markov decision process), Q 学習 (Q-learning)
JL 002/02/4202–0086

1. はじめに

強化学習（reinforcement learning）とは，試行錯誤を通じて環境に適応する学習制御の枠組である．教師付き学習 (supervised learning) と異なり，状態入力に対する正しい行動出力を明示的に示す教師が存在しない代わりに，一連の行動に対して結果としての良し悪しを評価する「報酬 (reward)」というスカラーの評価値が与えられ，これを手がかりに学習を行うが，報酬や状態遷移には不確実性や時間的遅れがある．そのため，一般に行動を実行した直後の報酬をみるだけでは，学習主体はその行動が正しかったかどうかを判断できないという困難を伴う．

強化学習の枠組を図 1 に示す．学習の主体「エージェント」と制御対象「環境」は以下のやりとりを行う．

1. エージェントは時刻 t において環境の状態観測 s_t に応じて意思決定を行い，行動 a_t を出力
2. エージェントの行動により，環境は s_{t+1} へ状態遷移し，その遷移に応じた報酬 r_t をエージェントへ与える．
3. 時刻 t を $t+1$ に進めてステップ 1 へ戻る．

エージェントは利得（return: 最も単純な場合，報酬の総計）の最大化を目的として，状態観測から行動出力へのマッピング（政策（policy）と呼ばれる）を獲得する．

環境とエージェントには一般に下記の性質が想定される．

- エージェントはあらかじめ環境に関する知識を持たない．
- 環境の状態遷移は確率的．
- 報酬の与えられ方は確率的．
- 状態遷移を繰返した後，やっと報酬にたどり着くような，段取り的な行動を必要とする環境（報酬の遅れ）．

強化学習では，環境のダイナミクスをマルコフ決定過程 (Markov decision process: MDP) によってモデル化し，学習アルゴリズムを解析するのが一般的である[6)7)]．本稿では，MDP モデルおよび代表的な強化学習法として知られる Actor-Critic 法と Q-learning を中心に理論的な基礎について解説する．

2. 強化学習の基礎理論

2.1 マルコフ決定過程（MDP）モデル

環境のとりうる有限な状態集合を $\mathcal{S} = \{s_1, s_2, \cdots, s_n\}$，エージェントがとる有限な行動集合を $\mathcal{A} = \{a_1, a_2, \cdots, a_\ell\}$

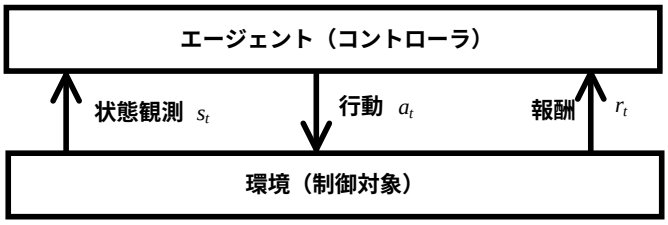

図 1　強化学習の枠組．エージェントは試行錯誤を通じて適切な制御規則を獲得していく．

と表す．時刻 t において環境中のとある状態 $s \in \mathcal{S}$ にあるとき，エージェントが行動 a を選択し実行すると，時刻 $t+1$ において環境は確率的に状態 $s' \in \mathcal{S}$ へ遷移する．その遷移確率は条件付確率 $P(s'|s,a)$ により表される．このとき環境からエージェントへ報酬 r が確率的に与えられるが，その期待値は条件付期待値 $R(s,a,s')$ により表される．すなわち，MDP は状態遷移の条件付確率 $P(s'|s,a)$ と報酬の条件付期待値 $R(s,a,s')$ により表される．ここで，条件付確率 $P(s'|s,a)$ と条件付期待値 $R(s,a,s')$ の条件部が，時刻 t の状態 s およびそのとき選択する行動 a のみであることから，時刻 t 以前の状態や行動の履歴に依存しないことが分かる．この性質はマルコフ性と呼ばれる．

エージェントは各時刻において何らかの規則に従って行動 a を選択するが，強化学習における学習目標は，後述する評価規範を最大化するような行動選択の規則を見出すことである．このとき，環境のマルコフ性により，エージェントが見出すべき行動選択の規則は，状態 s を条件部とした行動 a の条件付確率分布 $P(a|s)$ で表現すれば十分である．この行動選択の規則をここでは政策と呼び π と表す．また，この政策 π のもとで状態 s において行動 a を選択する確率を $\pi(s,a)$ と表す．

2.2 MDP の最適性：割引報酬による評価

ある時間ステップで実行した行動が，その後の報酬獲得にどの程度貢献したのかを評価するため，その後得られる報酬の時系列を考える．報酬の時系列評価は利得 (return) と呼ばれる．エージェントの学習目標は，利得を最大化すること，あるいはその政策を求めることである．強化学習では，以下に示す割引報酬合計による評価を利得とする場合が多い．ある時刻 t の状態，あるいはそのとき実行した行動の利得 V_t を以下で定義する：

$$V_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \quad (1)$$

ただし r_t は時刻 t の報酬，γ は割引率 $(0 \leq \gamma < 1)$ である．この V_t の期待値は，1 ステップあたり $(1-\gamma)$ の確率で停止するエージェントが得る報酬合計の期待値と等価である．MDP において割引報酬合計の評価がよく用いられる理由は以下のとおりである：

- 実環境では環境が変化したり故障等で停止する可能性があるため，時系列上の全ての報酬を同じ重みで考慮するのは妥当ではない．
- 計算処理上，値が発散することがなく都合が良い．
- 割引率 γ を 1 に近づけると，平均報酬の最適政策に一致するような割引報酬評価における最適政策を得る．

MDP においてエージェントが時間とともに変化することのない政策 π（定常政策という）をとるとき，状態 s から s' へ遷移する状態遷移確率 $P^\pi(s'|s)$ は以下で与えられる：

$$P^\pi(s'|s) = \sum_{a \in \mathcal{A}} \pi(s,a) P(s'|s,a) \quad (2)$$

また，状態 s からの 1 ステップの遷移に伴って得る報酬の期待値 $R^\pi(s)$ は以下で与えられる：

$$R^\pi(s) = \sum_{s' \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(s,a) P(s'|s,a) R(s,a,s') \quad (3)$$

エージェントが定常政策をとるとき，式 (1) で表された利得 V_t の期待値は，時刻 t における状態 s だけに依存する．すなわち利得の期待値は状態 s の関数になり，これを状態価値 (state value) 関数と呼び $V^\pi(s)$ と表す．この状態価値 $V^\pi(s)$ は，(1 step 目の遷移で得る報酬の期待値）＋$\gamma\times$（2 step 目の遷移で得る報酬の期待値）＋$\gamma^2\times$（3 step 目の遷移で得る報酬の期待値）＋$\cdots$＋$\gamma^{n-1}\times$（n step 目の遷移で得る報酬の期待値）＋$\cdots$ という具合に式 (4) で与えられ，この無限級数の式から式 (2,3) および $V^\pi(s)$ を用いた連立方程式 (5) を得る：

$$\begin{aligned} V^\pi(s) \;=\; & \\ & R^\pi(s) + \gamma \Big\{ \sum_{s_1 \in \mathcal{S}} P^\pi(s_1|s) R^\pi(s_1) \Big\} \\ & + \gamma^2 \Big\{ \sum_{s_1 \in \mathcal{S}} \sum_{s_2 \in \mathcal{S}} P^\pi(s_1|s) P^\pi(s_2|s_1) R^\pi(s_2) \Big\} \\ & + \gamma^3 \Big\{ \sum_{s_1 \in \mathcal{S}} \sum_{s_2 \in \mathcal{S}} \sum_{s_3 \in \mathcal{S}} P^\pi(s_1|s) P^\pi(s_2|s_1) P^\pi(s_3|s_2) \\ & \quad \times R^\pi(s_3) \Big\} + \cdots \\ & + \gamma^n \Big\{ \sum_{s_1 \in \mathcal{S}} \sum_{s_2 \in \mathcal{S}} \cdots \sum_{s_n \in \mathcal{S}} P^\pi(s_1|s) P^\pi(s_2|s_1) \cdots \\ & \quad \times P^\pi(s_n|s_{n-1}) R^\pi(s_n) \Big\} + \cdots \quad (4) \\ = \; & R^\pi(s) + \gamma \Big\{ \sum_{s_1 \in \mathcal{S}} P^\pi(s_1|s) V^\pi(s_1) \Big\} \quad (5) \end{aligned}$$

2.3 最適政策と状態-行動価値関数

全ての状態 s において $V^\pi(s) \geq V^{\pi'}(s)$ となるとき，政策 π は π' より優れているという．MDP では，他のどんな政策よりも優れた，あるいは同等な政策が少なくとも 1 つ存在する．これを最適政策 π^* という．最適政策は複数存在することもあるが，全ての最適政策は唯一の状態価値関数を共有する．これは最適な状態価値関数 V^* と呼ばれ，以下に定義される：

$$V^*(s) = \max_\pi V^\pi(s) \,, \qquad \forall s \in \mathcal{S}$$

この V^* は以下に示す Bellman の最適方程式を満たす：

$$\begin{aligned} V^*(s) \;&= \max_a \sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V^*(s') \right) \\ & \forall s \in S \end{aligned} \quad (6)$$

また，全ての最適政策は，以下に示す唯一の行動価値 (action value) 関数を共有する．

$$Q^*(s,a) \;=\; \max_\pi Q^\pi(s,a) \,, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$$

$Q^*(s,a)$ は最適な Q 関数と呼ばれ，状態 s で行動 a を選択後，ずっと最適政策をとりつづけるときの利得の期待値を表す．$Q^*(s,a)$ が与えられた場合，状態 s における最大の Q 値は $V^*(s)$ に等しく，またこの Q 値を持つ行動 a が最適な行動である．この $Q^*(s,a)$ は以下に示す Bellman の最適方程式を満たす：

$$\begin{aligned} & Q^*(s,a) \;= \\ & \sum_{s'} P(s'|s,a) \Big(R(s,a,s') + \gamma \max_{a'} Q^*(s',a') \Big) \\ & \forall s \in \mathcal{S}, a \in \mathcal{A} \end{aligned} \quad (7)$$

2.4 割引報酬最大化と平均報酬最大化

割引率 γ を 1 へ近づけると，ある割引率 γ^* 以上の割引率における最適政策は全て同じになり，このときの最適政策は平均報酬評価に関しても最適政策となる．

2.5 MDP の解法：政策反復法と価値反復法

MDP の状態遷移確率 $P(s'|s,a)$ と報酬の期待値 $R(s,a,s')$ が与えられたもとで最適方程式 (6) または (7) を解いて最適政策を求めることは，強化学習に対してプランニングと呼ばれる．上記の最適方程式は max のオペレーションを含む非線形方程式になっており，これを解く方法は 2 種類あるが，これらはダイナミックプログラミング (Dynamic programming: DP) と称される．一つは，政策反復法 (Policy iteration method) と呼ばれ，政策の改善と価値関数の計算を交互に繰り返す方法で，もう一つは価値反復法と呼ばれ，式 (6) または (7) を反復法を用いて数値計算により解く方法である．

1. 【政策評価】ある定常政策 π について以下の状態価値 $\boldsymbol{V}^{\pi}$ を求める：

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

2. 【政策改善】全状態 s において以下の式の値が最大になるよう政策を π から π' へと変更する．この計算で用いる価値関数は政策 π のものを利用する．

$$R^{\pi'}(s) + \gamma \sum_{s' \in S} P^{\pi'}(s'|s) V^{\pi}(s')$$

3. 【打切り判定】$\pi' = \pi$ となり変更すべき行動が無いとき，政策 π は最適政策なので処理を打ち切る．さもなければ $\pi \leftarrow \pi'$ として手順 1 へ戻る．

図 2　政策反復法による MDP の最適政策の求解

(1)　政策反復法

エージェントが定常政策 π をとるときの状態価値関数 $V^{\pi}(s)$ の方程式を式 (5) で示したが，この方程式は線形なので比較的容易に解ける．まず式 (2)，(3) で示した状態遷移確率 $P^{\pi}(s'|s)$ および報酬の期待値 $R^{\pi}(s)$ を以下のように行列で表す：

$$\boldsymbol{P}^{\pi} = \begin{bmatrix} P^{\pi}(s_1|s_1) & P^{\pi}(s_2|s_1) & \cdots & P^{\pi}(s_n|s_1) \\ P^{\pi}(s_1|s_2) & P^{\pi}(s_2|s_2) & \cdots & P^{\pi}(s_n|s_2) \\ \vdots & \vdots & \ddots & \vdots \\ P^{\pi}(s_1|s_n) & P^{\pi}(s_2|s_n) & \cdots & P^{\pi}(s_n|s_n) \end{bmatrix} \quad (8)$$

$$\boldsymbol{R}^{\pi} = \begin{bmatrix} R^{\pi}(s_1) \\ R^{\pi}(s_2) \\ \vdots \\ R^{\pi}(s_n) \end{bmatrix}, \quad \boldsymbol{V}^{\pi} = \begin{bmatrix} V^{\pi}(s_1) \\ V^{\pi}(s_2) \\ \vdots \\ V^{\pi}(s_n) \end{bmatrix} \quad (9)$$

これらの行列を用いて式 (4),(5) を書き直し，さらに $\boldsymbol{V}^{\pi}$ について解くと以下のようになる：

$$\begin{aligned} \boldsymbol{V}^{\pi} &= \boldsymbol{R}^{\pi} + \gamma \boldsymbol{P}^{\pi} \boldsymbol{R}^{\pi} + \gamma^2 (\boldsymbol{P}^{\pi})^2 \boldsymbol{R}^{\pi} + \cdots \\ &= \boldsymbol{R}^{\pi} + \gamma \boldsymbol{P}^{\pi} \boldsymbol{V}^{\pi} \quad = \quad (\boldsymbol{I} - \gamma \boldsymbol{P}^{\pi})^{-1} \boldsymbol{R}^{\pi} \quad (10) \end{aligned}$$

さて，ある政策 π のもとでの価値関数 $V^{\pi}(s)$ が得られると，政策改善の手がかりを得ることができる．ある状態 s においてのみ π とは別の政策 π' に従って行動をとるとき，

$$R^{\pi'}(s) + \gamma \sum_{s' \in S} P^{\pi'}(s'|s) V^{\pi}(s') - V^{\pi}(s) > 0 \quad (11)$$

が成り立つならば，その状態 s だけ政策 π' に従う新しい政策の状態評価値がもとの政策 π よりも改善されることが証明されている．この原理を利用して政策を改善していく．図 2 にその概要を示す．オリジナルの Howard の政策反復法では，政策 π は全て決定論的政策，すなわち各状態においてある行動を選ぶ確率が 1 で，それ以外の行動を選ぶ確率が 0 であるものに限定され，そのため状態と行動が有限な MDP では決定論的政策の個数が有限になるので，政策反復法が有限のステップ数で最適政策を見出すことが証明されている．ここでは強化学習法との関連を述べるため，あえて確率的政策を用いて表現した．

1. 【初期化】全ての状態 s について $Q^*(s,a)$ を適当な値（ゼロなど）に初期化する．
2. 【価値関数値の更新】全ての $Q^*(s,a)$ について以下の式を計算して値を代入：

$$Q^*(s,a) \leftarrow \sum_{s' \in S} P(s'|s,a) \times \left(R(s,a,s') + \gamma \max_{a' \in A} Q^*(s',a') \right)$$

ただし $\leftarrow$ は左辺の変数に右辺の計算結果を代入するオペレーションを表す．右辺に左辺と同じ変数が使われているが，これは更新前の古い値を用いることを意味する．
3. 【打ち切りの判定】処理手順 2 を値が収束するまで繰り返す．

図 3　価値反復法による Bellman 方程式の求解

(2)　価値反復法

ここでは式 (7) の行動価値関数に関する Bellman 最適方程式を数値反復計算で求める方法を紹介する．図 3 にその概要を示す．状態 s における最大の Q 値を持つ行動 a が最適な行動であるので，行動価値関数を求めれば最適政策は容易に求められる．

3.　強化学習アルゴリズム

MDP 環境下での強化学習では，状態遷移確率 $P(s'|s,a)$ や報酬 $R(s,a,s')$ についての知識を予め持たないため，環境とのやりとりを通じてこれらの情報を得なければならない．以下に紹介する強化学習法は，先に紹介した政策反復法や価値反復法を強化学習問題へ拡張したものと考えられ，確率的 DP とも呼ばれている．

3.1　Actor-Critic アルゴリズム

まず政策反復法の強化学習版と考えられる Actor-Critic 法を紹介する．この手法は，確率的政策に従って行動を選択する actor と呼ばれる要素と，状態価値を推定し，その情報から政策改善すべき方向を導く critic と呼ばれる要素より構成される．図 4 にアルゴリズムの概要を示す．図中に示された変数 TD_error は，政策反復法の式 (11) の右辺に相当することに注目してほしい．また，図中の状態価値 $\hat{V}^{\pi}(s)$ の更新式は，確率的価値反復法による状態価値関数の求解に相当する．

Actor-Critic 法は，政策 π すなわち行動選択確率関数において，正規分布を適用することなどにより連続値の行動へ拡張することが容易であったり，自然勾配 (natural gradient)[2] や政策勾配定理[5] などを利用した非常に効率の良

1. エージェントは環境の状態 s_t を観測し，確率的政策 π に従って行動 a_t を選択
2. 環境から報酬 r_t を受取り，状態遷移後の状態 s_{t+1} を観測
3. 【政策改善】以下の TD_error を計算：

$$\text{TD\_error} = \left[r_t + \gamma \hat{V}^{\pi}(s_{t+1})\right] - \hat{V}^{\pi}(s_t)$$

TD_error> 0 ならば，実行した行動 a_t は好ましいものと考えられるので，状態 s_t で行動 a_t を選ぶ確率を増やすよう政策 π を更新する．逆に TD_error< 0 ならば，実行した行動 a_t は好ましくないと考えられるので，状態 s_t で行動 a_t を選ぶ確率を減らすよう政策 π を更新する．

4. 【状態価値の推定値更新】$\hat{V}^{\pi}(s)$ を更新：

$$\hat{V}^{\pi}(s_t) \leftarrow \hat{V}^{\pi}(s_t) + \alpha \text{TD\_error}$$

ただし α は学習率で $0 < \alpha \leq 1$，γ は割引率である．

5. 時間ステップ t を $t+1$ へ進めて手順 1 へ戻る．

図 4　Actor-Critic アルゴリズム

1. エージェントは環境の状態 s_t を観測し，任意の行動選択方法 (探査戦略) に従って行動 a_t を実行
2. 環境から報酬 r_t を受取り，状態遷移後の状態 s_{t+1} を観測
3. 以下の式を用いて行動価値の推定値を更新：

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

ただし α は学習率で $0 < \alpha \leq 1$，γ は割引率である．

4. 時間ステップ t を $t+1$ へ進めて手順 1 へ戻る．

図 5　Q-learning アルゴリズム

い政策改善方法が提案されているなどの理由により，実問題やロボットなどへの適用において最も実績がある[7]．

3.2　Q-learning アルゴリズム

Q-learning[6] は $Q^{*}(s,a)$ の推定値 $\hat{Q}^{*}(s,a)$ を得るための代表的な強化学習法であり，価値反復法の強化学習版と考えられる．図 5 にアルゴリズムの概要を示す．ここで注目すべき点は，Actor-Critic では何らかの政策 π に従って行動が選択され，学習の進行とともに政策が改善されていく（on-policy と呼ばれる）のに対し，Q-learning ではある程度の条件が要求されるものの行動選択方法を規定する政策 π とは無関係に最適行動価値関数 $Q^{*}(s,a)$ や最適政策 π^{*} を求められる点（off-policy と呼ばれる）である[4]．

また，図 3 の価値反復法の更新式と図 5 の Q-learning の更新式の違いに注目すると，価値反復法では Q 値の計算部分に状態遷移確率 $P(s'|s,a)$ を用いているのに対し，Q-learning では状態遷移確率は未知なので用いていない．これは，エージェントが実際に環境での遷移を繰り返して小さな学習率 α によって少しずつ更新していくことが，状態遷移確率で重み付けたサンプル平均をとることになり，図 3 の価値反復法と等価な計算を実現している．これが確率的 DP と呼ばれる所以である．

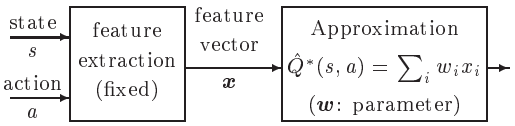

図 6　線形アーキテクチャによる Q 関数の汎化

（1）　Q-learning の収束定理

エージェントの行動選択において全ての行動を十分な回数選択し，かつ学習率 α が $\sum_{t=0}^{\infty} \alpha(t) \rightarrow \infty$ かつ $\sum_{t=0}^{\infty} \alpha(t)^2 < \infty$ を満たす時間 t の関数（例えば $\alpha(t) \propto \frac{1}{t}$ など）となっているとき，Q-learning アルゴリズムで得る Q 値は最適な Q 値に概収束する[6]．ただし環境はエルゴート性を有する離散有限 MDP であることを仮定する．その他，解析については文献[1] を参照されたい．

（2）　行動選択方法 (探査戦略)

上記の収束定理は，全ての行動を十分な回数選択しさえすれば行動選択方法（探査戦略）には依存せずに成り立つ．よって行動選択はランダムでもよい．しかし，強化学習ではまだ Q 値が収束していない学習の途中においてもなるべく多くの報酬を得るような行動選択を求められる．序々に挙動を改善していく行動選択方法として，

- ϵ-greedy 選択：確率 ϵ でランダム，それ以外は最大 Q 値を持つ行動を選択．
- ボルツマン選択：$exp(\hat{Q}(s,a)/T)$ に比例した割合で行動選択，ただし T は時間とともにゼロに近付ける．
- 楽天的初期値 (optimistic initial value: OIV)：Q 関数の初期値を大きめの値に設定しておき，実行していない状態-行動を選択しやすくする．

などの方法が提案されている[4]．

（3）　連続空間での行動価値関数表現

Q-learning に代表される強化学習アルゴリズムの動作の基本は，状態入力から状態・行動価値への写像関数を構築していくことである．状態表現の生成は特徴量抽出 (feature extraction) に関連するが，強化学習ではこの部分については扱わず，予め与えられるものとするのが一般的である．状態入力が連続空間の場合，このテクニックは汎化 (generalization) や関数近似と呼ばれ，強化学習に限らずニューラルネットワークなど幅広い分野で研究されており，強化学習または動的計画法と関数近似法を組み合わせた計算法は neuro-dynamic programming[1] と呼ばれる．Q-learning の最適解への収束が保障される汎化法として線形アーキテクチャ (linear architecture) がある．これは固定された基底関数集合を用いて関数近似を行うもので，ラジアル基底やシグモイド関数を用いたものなど様々だが，状態-行動入力を基底関数によって特徴ベクトル $\boldsymbol{x}$ に変換し，この線形重み付け和で Q 値を表現する点で共通する（図 6 参照）．

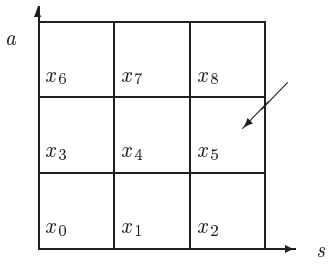

図 7　格子状タイリングによる特徴ベクトル生成例．状態-行動入力 (s, a) が矢印の位置のとき，特徴量ベクトル $(x_0, x_1, \cdots x_8)$ は，該当するタイル要素 $x_5 = 1$，それ以外の要素は全てゼロ．

特徴ベクトルの生成方法は問題に応じて設計者が与える必要があり，図 5 の離散空間 Q-learning との整合性から，特徴ベクトルのノルムは常に 1 となることが望ましい．

図 7 は最も単純な特徴ベクトル生成方法の例を示す．2 次元の入力空間を 1 種類の格子状タイリングによって分割し，各タイル要素に特徴量ベクトルの要素を割り当てる．状態-行動入力があるタイル要素内にあるとき，該当する特徴量ベクトル要素の値は 1 に，それ以外の要素はゼロとする．Q 値は，特徴ベクトル要素と同数のパラメータ変数 w_i を用意し，特徴ベクトルとの線形和で表す：

$$\hat{Q}^*(s, a) = \sum_{i=1}^{m} w_i x_i \tag{12}$$

ただし m は特徴ベクトル要素の個数である．Q 値を変える場合は，パラメータ変数 w_i の値を調節する．線形アーキテクチャによる Q-learning は，図 5 の更新式に対して単純な勾配法に基づくパラメータ w_i の更新で達成される：

$$w_i \leftarrow w_i + \alpha \frac{\partial \hat{Q}^*(s, a)}{\partial w_i} \delta_t \tag{13}$$

式 (12) より $\partial \hat{Q}^*(s, a)/\partial w_i = x_i$ なので，

$$w_i \ \leftarrow \ w_i + \alpha \ x_i \ \delta_t \tag{14}$$

つまり線形アーキテクチャを用いた更新では，更新するパラメータ w_i に対応する特徴量の要素の値 x_i だけを使って簡単に更新できる，図 7 のような単純なグリッド分割による線形アーキテクチャの更新式は，図 5 の離散空間 Q-learning の更新式と全く同じになる．一般に，全ての状態-行動ペアが互いに線形独立な特徴ベクトルで表現されていれば最適行動価値関数への収束が保障される[1]．図 8 に線形アーキテクチャを用いた Q-learning の処理手順をまとめる．

（4） 高次元空間での特徴ベクトル生成

図 9 は格子状タイリングを拡張した CMAC[4] による特徴量生成の例を示す．この例では 2 次元空間に 2 種類のグリッドを重ねているが，一般に多次元空間において多数のグリッドをランダムに重ねて用いる．しかし，CMAC は隙間なく並べられたタイリングを複数用いることが特徴なの

1. エージェントは環境の状態 s_t を観測し，任意の行動選択方法 (探査戦略) に従って行動 a_t を実行
2. 環境から報酬 r_t を受取り，状態遷移後の状態 s_{t+1} を観測
3. 状態 s，行動 a に対する特徴ベクトル $\boldsymbol{x}= (x_1, x_2, \cdots x_m)$ を何らかの方法を用いて生成する．ただし，区別が必要な状態-行動ペア間では必ず互いに線形独立になるよう設定し，ノルムを 1 にしておく．
4. 特徴ベクトル x_i と重み変数 w_i を用いて Q 値を計算：

$$\hat{Q}^*(s, a) = \sum_{i=1}^{m} w_i x_i$$

5. 以下の更新式により重み変数 w_i を更新する：

$$\delta_t = r_t + \gamma \max_a \hat{Q}^*(s_{t+1}, a) - \hat{Q}^*(s_t, a_t)$$
$$w_i \leftarrow w_i + \alpha \delta_t$$

ただし α は学習率で $0 < \alpha \leq 1$，γ は割引率である．

6. 時間ステップ t を $t+1$ へ進めて手順 1 へ戻る．

図 8　線形アーキテクチャを用いた Q-learning

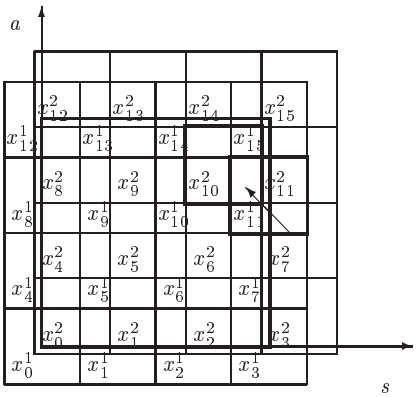

図 9　2 種類の格子状タイリングをずらせて重ねる CMAC による特徴量ベクトル生成．大きな太線の枠は状態-行動の定義域，2 つの小さい太線枠は入力に対応するタイルを表す．状態-行動入力が矢印の位置のとき，特徴量ベクトルに該当するタイル要素 $x^1_{11} = 0.5$，$x^2_{10} = 0.5$，それ以外の全要素はゼロ．

で，扱う次元数が高次元になるとメモリ空間が次元数に対して指数オーダで増加するという「次元の呪い」の問題に直面する．そこで，図 10 のようなランダムタイリングを用いた特徴ベクトル生成方法が提案されている[4)3)]．ある 1 つのランダムタイル $f(x)$ は，入力空間 x を構成する次元のうち，任意の 1 次元以上の複数次元（sensitive element と呼ぶ）で定義される部分空間中の，ある矩形領域を表し，入力された座標 x がその矩形領域である場合 $f(x) = 1$ を出力し，それ以外の場合 $f(x) = 0$ である．矩形領域を構成する次元や矩形の範囲・大きさなどは，タイル毎にランダムに与えるが，その与え方には統計的性質を持たせる[3]．このように，ある 1 つのタイルは空間を構成する次元の一部分で構成される空間の，とある領域として処理するが，こ

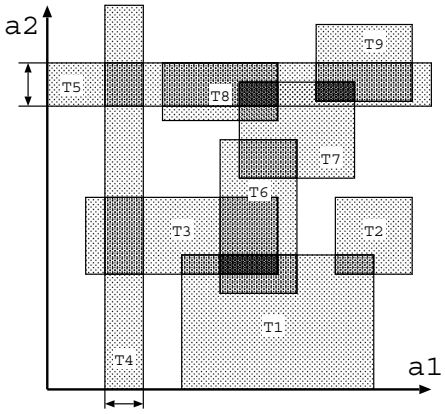

図 10　2 次元空間での 9 個のタイルによるランダムタイリングの例．タイル T4 は部分空間 a1 の矢印で示される区間で定義されるタイルのため，それ以外の空間 a2 では全領域をカバーするタイルになる．タイル T5 も同様．

のタイルを空間全体からみると，定義される部分空間中では範囲の限定された矩形だが，それ以外の空間では全領域をカバーするタイルと同義である．ランダムなタイルを用いることに抵抗を覚える読者が多いと思うが，実験してみると驚くべきことに同数の特徴量を持つ等間隔の格子状タイリングに比べ，比較にならないほど高性能である．図 11 はそれぞれ 2 次元・8 次元の入力空間に対し，ランダムタイリングと格子状タイリングによって特徴ベクトルを生成した場合の任意の近傍点 1000 個における線形独立性の割合を示したグラフである．線形独立の割合が高いほど，任意の 2 点間を区別可能であることを意味する．任意の近傍点のうちの 1 点は，各軸が $[0, 1]$ の範囲に限定された入力空間に一様分布で生成し，もう 1 点は先に生成した点を中心に各軸に標準偏差 0.1 共分散 0 の多次元正規分布で生成した．ロボットの強化学習では，このように空間的に少しだけ離れた 2 点を区別して全く別の制御を行う必要がたびたび生じるが，同じ個数の特徴量を用いた場合，特に高次元入力空間で多数の特徴量を用いた場合においてランダムタイリングは格子状タイリングよりはるかに優れた性能を示している．この理由は，格子状タイリングでは特徴ベクトルの要素のうちただ 1 つが 1 でそれ以外全て 0 という不必要に大きな制約のもとで特徴ベクトルを生成しているのに対し，ランダムタイリングではそのような制約が無いためと考えられる．

（2012 年 9 月 17 日受付）

参　考　文　献

1） Bertsekas, D.P. and Tsitsiklis, J.N.: ”Neuro-Dynamic Programming”, Athena Scientific (1996).
2） Kakade, S.: A Natural Policy Gradient, Advances in Neural Information Processing Systems 14, pp.1531–1538 (2002).
3） 木村 元：ランダムタイリングと Gibbs-sampling を用いた多次元状態-行動空間における強化学習，計測自動制御学会論文集, Vol.42, no.12, pp.1336–1343 (2006).
4） Sutton, R. S. and Barto, A.: ”Reinforcement Learning: An Introduction”, A Bradford Book, The MIT Press (1998).
5） Sutton, R. S., McAllester, D., Singh, S. & Mansour, Y.: Policy Gradient Methods for Reinforcement Learning with Function Approximation, Advances in Neural Information Processing Systems 12 (NIPS12), pp. 1057–1063 (2000).
6） Watkins, C.J.C.H. & Dayan, P.: Technical Note: *Q*-Learning, Machine Learning 8, pp.279–292 (1992).
7） 吉本 潤一郎，銅谷 賢治，石井 信：強化学習の基礎理論と応用，計測と制御, Vol.44, No.5, pp.313–318 (2005).

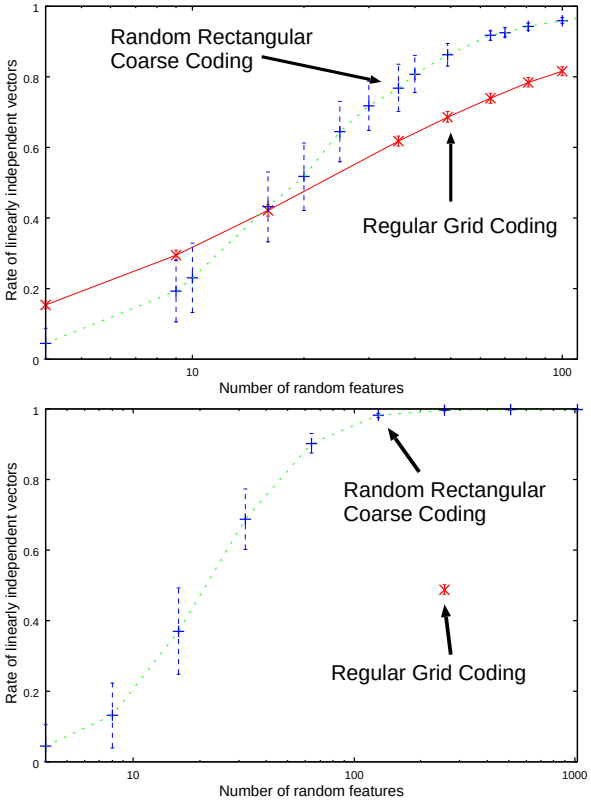

図 11　多次元空間において生成した特徴ベクトルの要素数と任意の近傍点における線形独立性の割合．上が 2 次元，下が 8 次元空間での結果．Random rectangular coarse coding はランダムタイリング，Regular grid coding は等間隔の格子状タイリングを表す．

［著　者　紹　介］

木村 元

1992 年東京工業大学工学部制御工学科卒業．1997 年同大学大学院総合理工学研究科知能科学専攻博士後期課程修了．1998 年 4 月，同大学大学院総合理工学研究科助手．2004 年 4 月，九州大学大学院工学研究院海洋システム工学部門助教授，2007 年 4 月，同部門において准教授へ職名変更，現在に至る．情報技術の船舶海洋分野への応用に関する研究に従事．計測自動制御学会，日本船舶海洋工学会，人工知能学会，日本ロボット学会会員．