岡山理科大学紀要 第47号 A pp. 11-19(2011)

適応性のあるモジュール型強化学習法

浅野 翼・山田 訓*

岡山理科大学大学院工学研究科修士課程知能機械工学専攻

*岡山理科大学工学部知能機械工学科

(2011年9月28日受付、2011年11月7日受理)

1. はじめに

強化学習 (Reinforcement learning : 以下 RL) [1] は，エージェントが環境との試行錯誤により行動の目標に応じた報酬の総和を最大にするような行動則を獲得するための枠組みである．制御結果に対する評価だけを用いて学習し，制御対象に対する事前知識を必要としないため，幅広い制御対象に適用できる可能性がある．しかし，強化学習特に temporal difference learning(TD 学習) では，現在と次の時間ステップの状態における推定値の差を用いて学習するため，CMAC やガウス関数などの局所的な基底関数で構成される関数近似法を用いる必要がある．従って入力次元が増えると必要とするユニット数が指数関数的に増大してしまうという問題がある．CMAC は，この問題を緩和することができるが，根本的な解決法ではない．一方，INGnet(Incremental Normalized Gaussian Network)[2] は局所的な近似を行なうと同時に，基底関数がカバーしている範囲外の状態空間の領域も緩やかな外挿によって汎化する正規化ガウス関数を，逐次的に配置する状態空間の表現法で，課題に適応してユニットを配置できるので，少ないユニット数で処理することができる．しかし，更に入力次元が高次元になると INGnet でも対処できなくなる可能性がある．CMAC でユニット数の増大に対処するために我々は，モジュール型強化学習 (modular reinforcement learning : 以下 modular-RL) を提案し，有効性を検討してきた [3][4][5]．そこで，modular-RL において，CMAC ではなく INGnet を用いることで，より複雑な制御に適用できるようになるのではないかと考え，INGnet を用いたモジュール型強化学習 (modular-RL(INGnet)) の有効性について検討した．ここで，従来の CMAC を用いたモジュール型強化学習を modular-RL(CMAC) と呼ぶこととする．

モジュール型強化学習は，全入力の一部を入力し，制御を学習する制御モジュールと，制御モジュールを状況に合わせて適切に選択するよう学習する選択モジュールで構成される．各モジュールは Q-learning によって学習される．modular-RL の各制御モジュールは，全入力の一部を入力するので，入力次元を少なくでき，単純な制御に分割し個々に学習することで，複雑な制御の学習も可能になる．modular-RL を用いることで，従来の強化学習のままでは困難な制御の学習が可能になることが示されている [3][4][5][6]．しかし，入力次元がさらに大きくなる場合，学習が困難になる場合があった [5]．モジュール型強化学習と INGnet を組み合わせることにより，さらに効率的な学習法になるのではないかと考え，有効性を検討した．本研究では，3 種類の制御課題に RL(INGnet) または modular-RL(INGnet) を適用し，有効性を検討した．特に課題 3 は，3 種類のセンサ情報を用いる必要がある複雑な課題であり，2 種類のセンサ情報の組み合わせ ("AND"条件) により，ターゲットを認識する必要がある．これらの課題を用いて，modular-RL(INGnet) が，課題に対し適切な空間表現法を自律的に学習できるか検討した．

2. 状態空間表現法

2.1 Incremental Normalized Gaussian Network (INGnet)

INGnet は入力空間を柔らかく領域分割する手法で，ある条件に従って正規化ガウス関数を逐次配置する．この手法は，入力空間の必要な場所に基底関数を配置するため，記憶量が少なくできる可能性がある．与えられた n 次元の入力ベクトル $X(t) = (x_1, ..., x_n)$ に対して,k 番目のユニットの活性化関数は，次のように計算される．

$$a_k(X) = e^{-\frac{1}{2}\|M_k(X-c_k)\|^2} \tag{1}$$

ここで,c_k は活性化関数の中心であり,M_k は活性化関数の形状を決定する行列である.

次に, 活性化関数 a_k を各点で総和が 1 なるように正規化したものを基底関数 b_k とする.

$$b_k(X) = \frac{a_k(X)}{\Sigma_{m=1}^{K} a_m(X)} \quad (2)$$

ここで,K は基底関数のユニット数である. INGnet では, 誤差がある基準 e_{max} より大きく, すべての存在するユニットの活性度がある閾値 a_{min} より小さければ, つまり

$$|\hat{y}(x) - y(x)| > e_{max} \; and \; max_k a_k(x) < a_{min} \quad (3)$$

のときに新しいユニットを配置する [2]. 学習初期において荷重の計算が十分でないため, 関数の近似誤差が多いことを考慮して, e_{max} は試行回数に応じて次のようにした.

$$e_{max} = 0.5exp(-T/T_1) \quad (4)$$

ここで, T は現在の Episode 回数, T_1 は 1 試行の Episode 回数とした. 新しいユニットは, $c_k = X$, $M_k = diag(\mu_i)$ で初期化した. ここで μ_i は活性化関数の半径の逆数であり, 本研究では新たに配置されるユニットの μ_i は一定とした.

3. モジュール型強化学習

3.1 学習システム

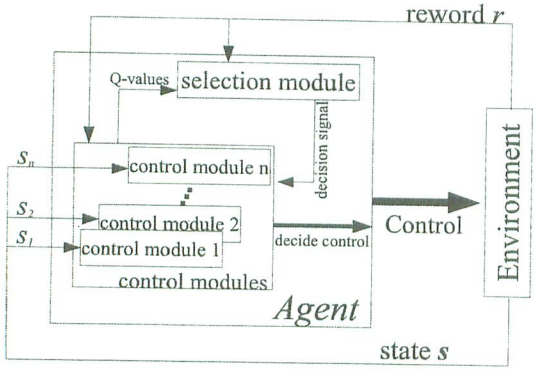

図 1 モジュール型強化学習 (modular-RL) システムの構成

modular-RL は, 図 1 に示すように複数の制御モジュールと選択モジュールから構成されるシステムである. 与えられた制御課題から, 制御できるよう状態空間を分割し, モジュールに割り当てる. 各制御モジュールは, 状態空間の一部を入力とし, 状態に対する適切な制御を学習する. 選択モジュールは, 各制御モジュールの制御結果の予測値 (行動価値関数) を入力とし, 状態に対する適切な制御モジュールを選択するように学習する. 選択モジュールにより, 選択された制御モジュールの選択した制御がシステム全体の制御出力となる. 制御結果に対する報酬はすべてのモジュールに与えられる.

3.2 学習アルゴリズム

各モジュールは Q-learning または Sarsa(λ) で学習し, 各モジュールの状態表現には, INGnet と CMAC[3][4][5][6] を用いる. 以下には状態表現法 INGnet を用いた場合の学習アルゴリズムを示す. 制御モジュール m の状態 s_m, 行動 a に対する近似行動価値関数 $Q_m(s_m, a)$ 及び, 選択モジュールの状態 s_s , 制御モジュール m に対する近似行動価値関数 $Q_s(s_s, m)$ は次式で計算される.

$$Q_m(s^m, a) = \vec{w}_m^T \vec{\phi}_{s^m a} = \sum_{i=1}^{n_m} w_m(i)\phi_{s^m a}(i)$$

$$Q_s(s^s, m) = \vec{w}_s^T \vec{\phi}_{s^s m} = \sum_{i=1}^{n_s} w_s(i)\phi_{s^s m}(i) \quad (5)$$

ここで, $\vec{w}_m$ は制御モジュール m のパラメータベクトル, $\vec{w}_s$ は選択モジュールのパラメータベクトル, $\vec{\Phi}_{s^m a}$ は制御モジュール m の行動 a に対する状態を表す特徴ベクトル, $\vec{\Phi}_{s^s m}$ は選択モジュールのモジュール m に対する状態を表す特徴ベクトルである. n_m, n_s はそれぞれ制御モジュール m の状態・行動のユニット数, 選択モジュールのユニット数である.

選択モジュールは $Q_s(s_s, m)$ に基づき, 制御モジュールを選択する. 選択された制御モジュールは $Q_m(s_m, a)$ に基づき, 行動を選択する. モジュール, 行動は Q 値に依存した確率でランダムに選択する.

制御モジュールのパラメータの更新は, 次式で

行う．

$$\hat{r}_m(t) = \begin{cases} 0 & m_t \neq m \\ r_{t+1} + \gamma_m Q_m(s^m_{t+1}, a_{t+1}) & \\ -Q_m(s^m_t, a_t) & m_t = m, m_{t+1} = m \\ r_{t+1} + \gamma_m Q_s(s^s_{t+1}, m_{t+1}) & \\ -Q_m(s^m_t, a_t) & m_t = m, m_{t+1} \neq m \end{cases}$$

$$w_m(i) = w_m(i) + \alpha_m \hat{r}^m_t e_m(i)$$

$$(i = 1, 2, \ldots, n_m) \qquad (6)$$

ここで，m_t は時刻 t で選択された制御モジュール，a_t は時刻 t で選択された制御，$r(t)$ は時刻 t での報酬である．γ_m，α_m はそれぞれ制御モジュールに対する減衰率，学習率である．e_m は制御モジュール m の適格度トレースである．選択モジュールのパラメータの更新は次式で計算される．

$$\hat{r}^s_t = r_{t+1} + \gamma_s Q_s(s^s_{t+1}, m_{t+1}) - Q_s(s^s_t, m_t) - \hat{r}_{m_t}(t)$$

$$w_s(i) = w_s(i) + \alpha_s \hat{r}^s_t e_s(i)$$

$$(i = 1, 2, \ldots, n_s) \qquad (7)$$

ここで，γ_s，α_s はそれぞれ選択モジュールに対する減衰率，学習率である．$\hat{r}_{m_t}$ は，時刻 t で選択されたモジュール m_t で計算された TD 誤差である．e_s は選択モジュールの適格度トレースである．

適格度トレースは，累積トレースを用いる．次式により計算する．

$$e_m(i) = \begin{cases} \lambda_m e_m(i) + b_{s_m a}(i) & a = a_t, b_{s_m a} = b_{s_m a_t} \\ \lambda_m e_m(i) & otherwise \end{cases}$$

$$(i = 1, 2, \ldots, n_m)$$

$$e_s(i) = \begin{cases} \lambda_s e_s(i) + b_{s_s m}(i) & a = a_t, b_{s_s m} = b_{s_s m_t} \\ \lambda_s e_s(i) & otherwise \end{cases}$$

$$(i = 1, 2, \ldots, n_s) \qquad (8)$$

ここで，λ_m は制御モジュール m に対するトレース減衰率，λ_s は選択モジュールに対するトレース減衰率である．

4. 制御対象と制御課題

4.1 制御対象

本研究では，制御対象として Khepera ロボット (図 2(a)) を用いる．ロボットのモデルは，ニース大学の Olivier Michel が開発した，Khepera シミュレータ [7] を基本とし，各センサの実測データ等を基にプログラムを修正したものを用いた．このシミュレータを用いて制御学習のシミュレーションを行った．Khepera ロボットの各センサのうち，赤外線センサ (図 2(b)) の距離センサモードと，光センサモード，一次元 CCD アレイ (課題 3 のみ) を用いた．

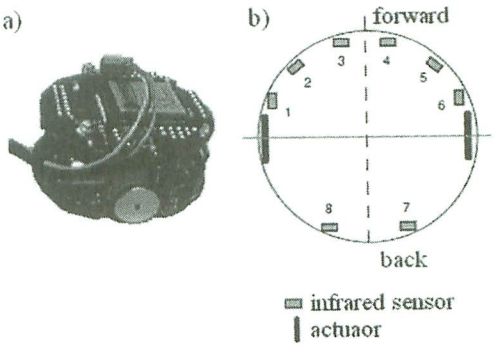

図 2 Khepera ロボット．(a) 概観写真．(b) 赤外線センサ配置図．数字はセンサナンバ．

4.2 制御課題

制御課題として以下の 3 つの課題を行なった．ロボットの行動は，左旋回，前進，右旋回の 3 つとする．ロボットが行動を実行し次の状態を観測するまでを 1Step とし，成功・失敗までを 1Episode とする．

4.2.1 制御課題 1 の報酬関数

図 3(b) のような環境で，壁に衝突せず，壁沿い (距離センサの計測値が一定の範囲内) を移動することが課題である．この課題では距離センサ値のみを入力するので，modular-RL を用いず，RL(INGnet) と RL(CMAC) を比較した．壁沿いを 2000 回連続で走行した場合成功とし，+3 の報酬を与えた．また，壁沿いを走行している場合は +0.1，壁に衝突する，一定回数以上その場で回転する，一定ステップ以内に成功しない場合は，失敗として，-1 の報酬を与えた．その他は 0 を与えた．シミュレーションの条

件は，1000Episode を 1 試行とし 50 試行行なった．制御成功を 10Episode 連続で行なうことができた場合，学習が完了したとした．

4.2.2 制御課題 2 の報酬関数

図 4(b) のような環境で，壁に衝突せずに，ランプに到達することが課題である．ランプに到達した場合は成功とし，+1 の報酬を与えた．失敗は課題 1 と同様である．シミュレーションの条件は，2000Episode を 1 試行とし 50 試行行なった．制御成功を 10Episode 連続で行なうことができた場合，学習が完了したとした．この課題は，距離センサと光センサを用いるので，距離センサを入力とする障害回避モジュール，光センサを入力とする光モジュールと選択モジュールからなるモジュール型強化学習システム (modular-RL) で学習した．RL(INGnet)，modular-RL(INGnet)，modular-RL(CMAC) で学習を行ない，結果を比較した．

4.2.3 制御課題 3 の報酬関数

課題 3 は，図 5(b) のような環境で，壁やダミーを避けて，ターゲットに到達することが課題である．ターゲットに到達した場合は成功とし，+1 の報酬を与えた．失敗及びシミュレーションの条件は課題 1 と同様である．ターゲットは，ランプが点いた黒い棒である．ダミーはランプのみと黒い棒のみである．ターゲットとダミーを区別するためには，光センサと一次元 CCD アレイの"AND"条件を識別する必要がある．この課題は，距離センサ，光センサ，一次元 CCD アレイを用いるので，障害回避モジュール，光モジュール，一次元 CCD アレイデータから得られた情報を入力とする CCD モジュールと選択モジュールからなるモジュール型強化学習システム (modular-RL) を用いた．課題 2 と同様に RL(INGnet)，modular-RL(INGnet)，modular-RL(CMAC) を比較した．CCD モジュールは，黒い棒を認識する左端のピクセル番号と棒の幅の 2 個を入力とする．

5. 学習システム

5.1 制御課題 1

空間表現法 INGnet の有効性を検討するために，CMAC を用いた強化学習 (RL(CMAC)) と INGnet を用いた強化学習 (RL(INGnet)) で制御学習を行なった．制御対象に与える入力は $(\frac{D_{2i-1}+D_{2i}}{2})$, $(i = 1,2,3,4)$ の 4 個の距離センサ値を入力とする（D は距離センサ値，i はセンサナンバを示す（図 2(b))）．CMAC の場合はタイル数を 5 とし，距離センサの最大値 2048 をそれぞれ 10 分割で均等にタイリングした。学習パラメータは $\alpha = 0.2, \gamma = 0.9$, $\lambda = 0.7$, $T = 0.08$ で設定した．INGnet は距離センサの基底関数の形状を $M_d = diag(0.003)$, $M_d = diag(0.005)$, $M_d = diag(0.01)$ の 3 通りで学習を行なった．INGnet の学習パラメータは $\alpha = 0.06$, $\gamma = 0.8, \lambda = 0.8$, $a_{min} = 0.2$, 初期基底関数位置 $c(0) = [200, 200, 200, 200]$ とした．

5.2 制御課題 2

RL(INGnet),modular-RL(INGnet),modular-RL(CMAC) の比較を行なった．障害回避モジュールは課題 1 の入力と同様とした．光モジュールには $min(O_{2i-1}, O_{2i})$, $(i = 1,2,3,4)$ の 4 個を入力とする（O は光センサ値，i はセンサナンバを示す（図 2(b))）．選択モジュールは各制御モジュールの，現在の状態で計算される最大の Q 値 2 個を入力とする．RL(INGnet) の学習パラメータは，$\alpha = 0.7, \gamma = 0.9, \lambda = 0.7$, $T = 0.05$, $a_{min} = 0.3$, 初期基底関数位置 $c(0) = [2000, 2000, 2000, 2000, 500, 500, 500, 500]$ とした．障害回避モジュール，光モジュール，選択モジュールは，CMAC の場合はタイル数を 5 とし，障害回避モジュールは距離センサの最大値 2048 を，光モジュールは光センサの最大値 500 を，選択モジュールは-1 から 1 までを，それぞれ 5 分割し均等にタイリングする．modular-RL(CMAC) の学習パラメータは $\alpha_m = 0.8, \gamma_m = 0.2, \lambda_m = 0.6$, $T_m = 0.01, \alpha_s = 0.4, \gamma_s = 0.2, \lambda_s = 0.8$, $T_s = 0.08$ とした．modular-RL(INGnet) の障害回避，光，選択モジュールの基底関数の形状は $M_d = diag(0.005)$, $M_l = diag(0.03)$, $M_s = diag(10)$ とした．modular-RL(INGnet)

の学習パラメータは，$\alpha_m = 0.5, \gamma_m = 0.8$, $\lambda_m = 0.8$, $T_m = 0.06, \alpha_s = 0.3, \gamma_s = 0.8$, $\lambda_s = 0.8$, $T_s = 0.07$, $a_{min} = 0.3$ とした．(制御モジュールは全て共通) 各手法のモジュール型強化学習の障害回避，光，選択モジュールの初期基底関数位置は $c_d(0) = [2000, 2000, 2000, 2000]$, $c_l(0) = [500, 500, 500, 500]$, $c_s(0) = [0, 0]$ とした．

5.3 制御課題 3

制御課題 3 では，赤外線センサの距離センサモード，光センサモード，一次元 CCD アレーの 3 つのセンサ情報を入力とした 10 入力で制御学習を行なう．RL(INGnet) の学習パラメータは,$\alpha = 0.7, \gamma = 0.8, \lambda = 0.8$, $a_{min} = 0.3$, $T = 0.04$, 初期基底関数位置 $c(0) = [2000, 2000, 2000, 2000, 500, 500, 500, 500, 0, 0]$ とした．障害回避モジュール，光モジュール，選択モジュールは，CMAC の場合のタイル数などは制御課題 2 と同様とする．CCD モジュールのタイル数は 7 とし，入力ピクセル数 64 を 9 分割し均等にタイリングする．modular-RL(CMAC) の学習パラメータは，$\alpha_m = 0.8, \gamma_m = 0.8, \lambda_m = 0.5$, $T_m = 0.01$, $\alpha_s = 0.6, \gamma_s = 0.8$, $T_s = 0.003, \lambda_s = 0.5$ とした．modular-RL(INGnet) の障害回避, 光,CCD, 選択モジュールの基底関数の形状は $M_d = diag(0.005)$, $M_l = diag(0.03)$, $M_c = diag(0.2)$, $M_s = diag(10)$ とした．modular-RL(INGnet) の学習パラメータは，$\alpha_m = 0.2, \gamma_m = 0.8, \lambda_m = 0.8$, $T_m = 0.08, \alpha_s = 0.1, \gamma_s = 0.8, \lambda_s = 0.7$, $T_s = 0.07$, $a_{min} = 0.2$ とした（制御モジュールは全て共通）． 各手法のモジュール型強化学習の障害回避，光，CCD, 選択モジュールの初期基底関数位置は $c_d(0) = [2000, 2000, 2000, 2000]$, $c_l(0) = [500, 500, 500, 500]$, $c_c(0) = [0, 0]$, $c_s(0) = [0, 0, 0]$ とした．

6. 結果と考察

6.1 制御課題 1

図 3(a) は RL(CMAC) と基底関数の幅を変えた RL(INGnet) を比較した結果である．適切な幅の基底関数を用いれば (RL(INGnet)$M(i) = diag(0.005)$), RL(CMAC) より効率よく学習できることが分かった．CMAC は不必要な状態に関しても計算しなければならないので, 最適な制御を学習するのに時間がかかってしまうと考えられる．課題 1 は，壁沿いであることを維持するために，距離センサ値の細かい違いを区別する必要がある．そのため，INGnet の基底関数の幅を適切な値に設定する必要があった．幅が大きすぎる場合 ($M(i) = diag(0.003)$) も細かすぎる場合 ($M(i) = diag(0.01)$) も学習が遅くなった．基底関数の幅が小さすぎる場合は，ユニット数が多くなるため，学習に時間がかかったと考えられる．特に学習初期の成功率が低かった．図 3(b) は学習された制御によるロボットの軌跡を示す．壁沿いを移動できていることが分かる．

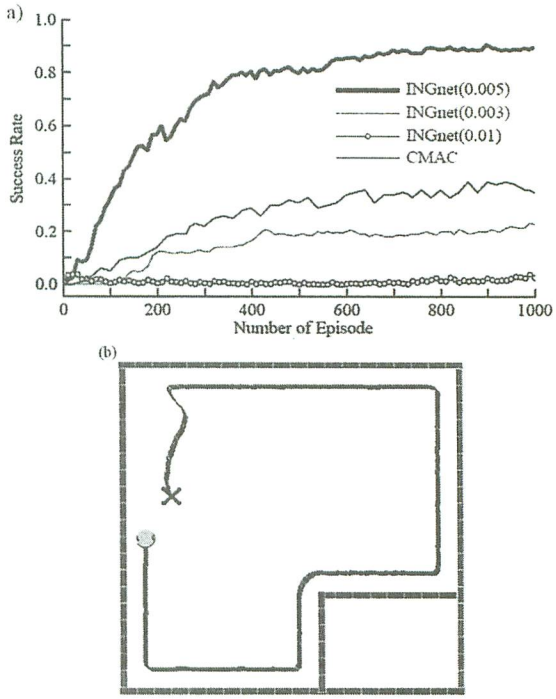

図 3 制御課題 1 の結果．(a) 学習曲線． CMAC は RL(CMAC), INGnet(0.005), INGnet(0.003), INGnet(0.01) は，RL(INGnet) で,$M_i = diag(0.005)$, $M_i = diag(0.003)$, $M_i = diag(0.01)$ を示す．INGnet の平均ユニット数は，INGnet(0.005) は約 54 個，INGnet(0.003) は約 36 個, INGnet(0.01) は約 158 個であった．(b)RL(INGnet,$M_i = diag(0.005)$) で学習された制御によるロボットの軌跡 (×:スタート地点).

表 1 制御課題 2 における RL(INGnet), modular-RL(INGnet), modular-RL(CMAC) のユニット数と計算時間.

	N_{pu}	T_{1u}	T_{all}
modular-RL (INGnet)	108.1	0.676	73.1
RL(INGnet)	110.7	0.817	90.5
modular-RL (CMAC)	6375.0	0.087	555.2

全ユニット数 (N_{pu}),1 ユニットの計算時間 (T_{1u}), 全ユニットの計算時間 (T_{all}).

RL(CMAC) と RL(INGnet) のユニット数を比較すると，CMAC では 5×10^4 個のユニットが必要であるが，$M_k = diag(0.005)$ の RL(INGnet) では，平均 54 個であった．それぞれの計算時間を比較すると，各ユニットの計算時間は,CMAC は INGnet の約 1/10 であるが，ユニット数が大きく異なるので，全体としては RL(INGnet) の約 75 倍であることが分かった.

6.2 制御課題 2

図 4(a) は，課題 2 での学習結果を示す．modular-RL(CMAC) と modular-RL(INGnet), RL(INGnet) の最終的な成功率はあまり変わらないが，modular-RL(CMAC) は学習初期の成功率が低く，学習が遅かった．modular-RL(INGnet) と RL(INGnet) の学習に違いはなかった．図 4(b) は，modular-RL(INGnet) で学習された制御によるロボットの軌跡を示す．ランプ付近では，光モジュールが選択され，適切なモジュール選択が学習できていることがわかる．一方，各手法のユニット数を表 1 に示す．課題 1 と同様，INGnet を用いた場合には，ユニット数は少なくてすむので，全体としては，約 1/10 の計算時間である．課題 2 では，学習効率，ユニット数の両方において，RL(INGnet) と modular-RL(INGnet) の違いはなかった．課題 2 は，入力が 8 次元と小さいので，modular-RL(INGnet) の性能が発揮できなかったのではないかと考えられる．そこで，より入力次元の大きい課題 3 の学習を試みた.

6.3 制御課題 3

図 5(a) は，課題 3 の学習結果を示す．modular-RL(CMAC) は非常に学習が遅く，この Episode

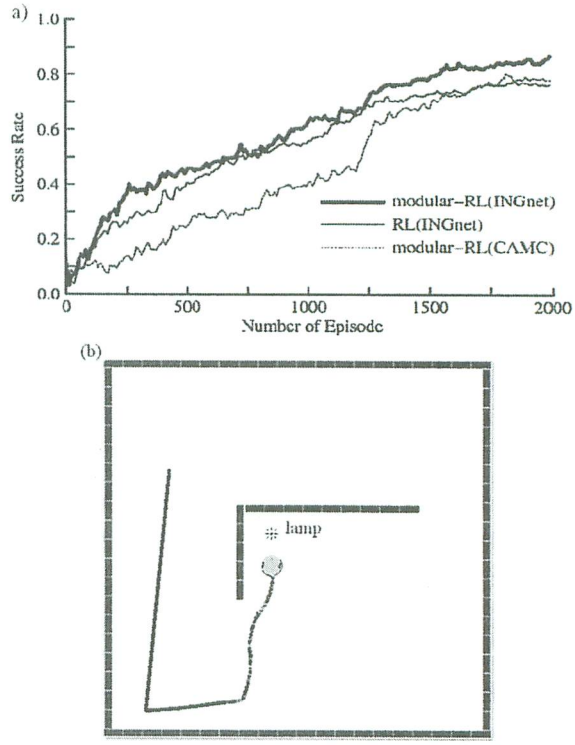

図 4 制御課題 2 の結果．(a)RL(INGnet), modular-RL(INGnet), modular-RL(CMAC) の学習曲線 (b), modular-RL(INGnet) で学習された制御によるロボットの軌跡.

数では，ほとんど学習できなかった．modular-RL(INGnet) は RL(INGnet) より学習速度が速く，最終的な成功率も高かった．また，表 2 に示したユニット数の点でも modular-RL(INGnet) は RL(INGnet) より優れており，約 1/6 のユニット数であった．図 5(b) は，modular-RL(INGnet) で学習された制御によるロボットの軌跡で，ターゲット付近では光モジュールや CCD モジュールが選択され，適切なモジュールが選択され，ターゲットに到達できていることが分かる．以上のことから，高次元入力の制御課題を，modular-RL(INGnet) が効率よく学習できることが分かった.

6.4 パラメータ依存性

新たな制御課題に modular-RL(INGnet) を適用するには，パラメータを適切な値に設定する必要がある．障害回避制御と課題 3 を用いて，パラメータ依存性を調べ，適切なパラメータ設定について，検討

表 2 制御課題 3 における RL(INGnet), modular-RL(INGnet), modular-RL(CMAC) のユニット数

	N_{all}	N_o	N_l	N_c	N_s
modular-RL(INGnet)	111.6	27.0	28.6	11.7	44.3
RL(INGnet)	623.0	—	—	—	—
modular-RL(CMAC)	7442.0	3125.0	3125.0	567.0	625.0

全ユニット数 (N_all), 距離モジュールのユニット数 (N_o), 光モジュールのユニット数 (N_l),CCD モジュールのユニット数 (N_c), 選択モジュールのユニット数 (N_s).

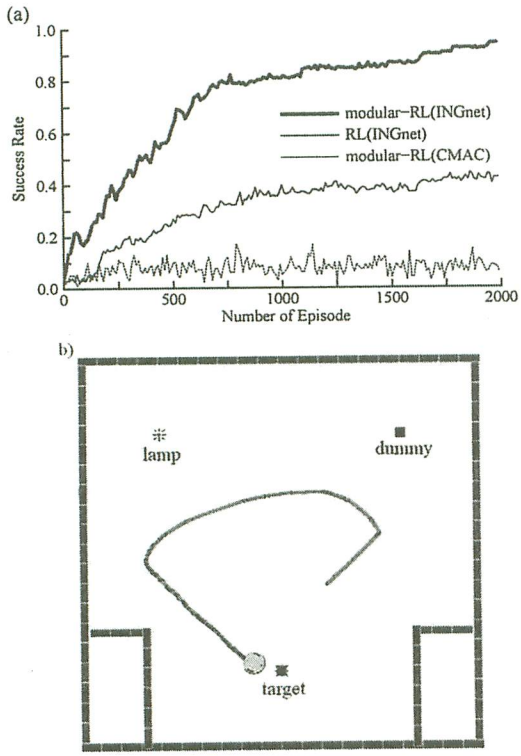

図 5 制御課題 3 の結果. (a)RL(INGnet), modular-RL(INGnet), modular-RL(CMAC) の学習曲線 (b), modular-RL(INGnet) で学習された制御によるロボットの軌跡.

した. 図 6 は基底関数の形状を表す定数 μ_i を変化させた場合の最終平均成功率を表す. 距離センサや光センサを処理するモジュールでは, ガウス関数の相対的な幅が 0.1 ～ 0.2 に相当する μ_i の場合, 成功率が高かった (図 6a). 一方, 1 次元 CCD アレイの情報を処理するモジュールでは, 相対的な幅が 0.03 に相当する μ_i の場合に成功率が高かった (図 6b). セ

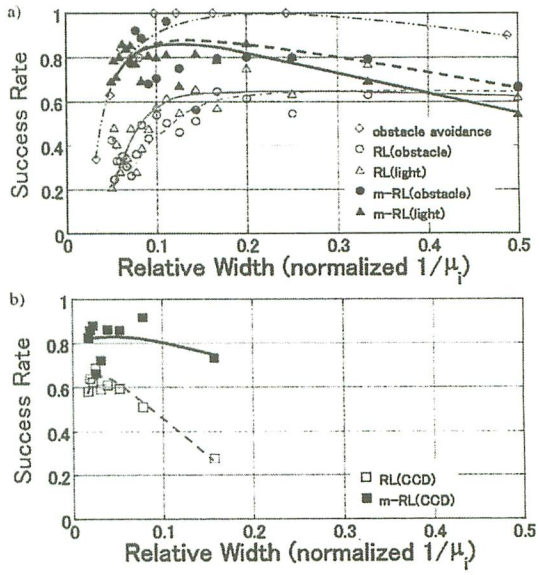

図 6 障害回避制御と制御課題 3 について INGnet を適用させた RL と moudlar-RL でのパラメータ μ_i を変化させた結果. obstacle avoidance は障害回避制御, RL は強化学習, m-RL はモジュール型強化学習, obstacle・light・CCD はそれぞれ障害回避・光・CCD モジュールの基底関数の形状を変化させたことを表す.

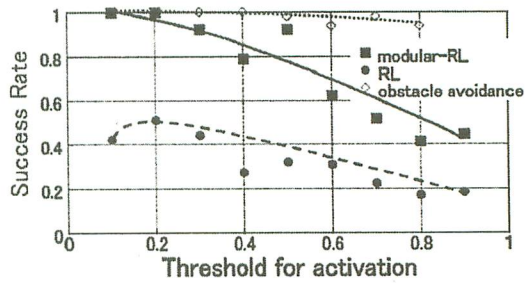

図 7 障害回避制御と制御課題 3 について INGnet を適用させた RL と moudlar-RL でのパラメータ a_{min} を変化させた結果.

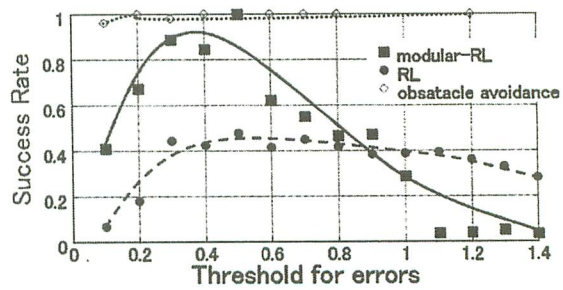

図 8 障害回避制御と制御課題 3 について INGnet を適用させた RL と moudlar-RL でのパラメータ $er = \frac{exp(-T/T_a)}{e_{max}}$ を変化させた結果.

ンサの種類により，適切な μ_i が異なるが，図 6 に示すように μ_i 依存性はいずれも緩やかであるので，μ_i の値を厳密に設定する必要はないと考えられる．図 7 は，a_{min} を変化させた場合の最終平均成功率を表す．$a_{min} = 0.1 \sim 0.3$ のとき，成功率が高いことが分かった．図 8 は，er を変化させた場合の最終平均成功率を表す.$er = 0.5$ のとき，成功率が高いことが分かった．これらの結果より a_{min} と er は，障害回避と課題 3 で最適値がほぼ一致し，同様の傾向を示した．μ_i は入力によって最適値は異なるが，依存性が緩やかであった．従って，本研究で用いたパラメータで多くの課題を学習できるのではないかと考えられる．

7. まとめ

制御課題 1 の結果より，適切な幅の INGnet を用いれば，センサ値の細かい違いを区別する必要のある課題を学習できることが分かった．制御課題 3 の結果より，modular-RL(INGnet) を用いれば，高次元入力の制御課題も効率よく学習できることが分かった．

今後はさらに複雑な制御課題に modular-RL(INGnet) を適用し，modular-RL(INGnet) の可能性を検討したいと考えている．また，基底関数の幅を学習によって適応させる方法や実機での制御学習にも取り組みたいと考えている．

参考文献

[1] Sutton, R. S. and Barto, A. G.: *Reinforcement Learning: An Introduction*, The MIT Press (1998).

[2] 森本淳, 銅谷賢治：強化学習を用いた高次元連続状態空間における系列運動学習—起き上がり運動の獲得—, 電子情報通信学会論文誌, Vol. J82-D-II, No. 11, pp. 2118–2131 (1999).

[3] 山田訓：モジュール型強化学習, 信学技報, pp. 139–146 (1998), NC97–119.

[4] 中間隼人, 田中直樹, 山田訓：3 種類のセンサを持つロボット制御へのモジュール型強化学習の適用, 信学技報, Vol. 108, No. 480, pp. 301–306 (2009), NC2008–154.

[5] Nakama, H. and Yamada, S.: Modular Reinforcement Learning for the Detection of Second Order Correlation of Multi–Sensors, in *Proceedings of the 2010 International Conference on Modeling, Identification and Control*, pp. 41–46 (2010).

[6] Yamada, S., Watanabe, A. and Nakashima, M.: Hybrid reinforcement learning and its application to biped robot control, *Advances in NIPS*, Vol. 10, pp. 1071–1077 (1998).

[7] Michel, O.: *Khepera simulator version 2.0 user manual*, University of Nice (1996).

Adaptive Modular Reinforcement Learning

Tsubasa ASANO and Satoshi YAMADA*

Graduate School of Engineering,

**Department of Inteligent Mechacical Enginnering, Faculty of Engineering,*

Okayama University of Science,

1-1 Ridai—cho, Kita-ku, Okayama 700-0005, Japan

(Received September 28, 2011; accepted November 7, 2011)

The adaptive modular reinforcement learning system was proposed to apply the reinforcement learning into more realistic control problems. The learning system is composed of some control modules and a selection module. All modules of this system are calculated by using the incremental normalized Gaussian networks (INGnet). It learned the task, where the "AND" condition of two types of sensor information should be discriminated, more quickly than the modular reinforcement learning using CMAC, or the reinforcement learning using INGnet. Since the number of the processing units of the adaptive modular reinforcement learning was smaller than that of the reinforcement learning using INGnet, it is considered to have the ability to obtain more appropriate representations for the control.

Keyworlds: reinforcement learning ; incremental normalized Gaussian networks ; modular reinforcement learning.