第７章　時間軸スムージング学習に基づく遅延強化学習

第１章で述べたように、自律学習の範疇に入る代表的な学習アルゴリズムに強化学習がある。本章では、報酬や罰（強化信号）が一連の動作の後に得られる場合に、そこまでの動作をいかに学習するかという遅延強化学習の問題に時間軸スムージング学習が適用できることを示す。そして、ニューラルネットを用いることにより効率的な学習が実現できること、視覚センサ信号を直接入力としても学習を行うことができ、その時、ニューラルネットの中間層に、空間の情報がどのようにコーディングされるかといった点を移動ロボットのシミュレーションを通して示す。そして、これらから、自律学習とニューラルネットの有効性について述べる。

7.1 背景

１.２節で述べたように、遅延強化学習の問題に関しては、1983 年の倒立振子の制御の学習を例題として扱った Barto らの critic-actor アーキテクチャ（ＴＤ学習）[Barto 83] が有名である。ここでは、現在のセンサからの入力を元に、現在から将来にわたって得られる強化信号 r に、現在からその強化信号が得られるまでの時間 t によってディスカウントファクタγ^t　(0≦γ≦1)を掛けたものの総和

$$\bar{r} = \sum_{t=0}^{\infty} \gamma^t \, r(t+1) \qquad （７.１）$$

を予測するように critic 部が学習し、この総和を最大化するように actor 部で動作を学習する。この critic 部の $\bar{r}$ の予測は、ＴＤ(Temporal Difference)学習[Sutton 88]を適用し、逐次的に学習を行う。また、このＴＤ学習を用いて $\bar{r}$ を学習すること、またはこれを用いて強化学習の問題を解くことを単にＴＤ学習と呼ぶ場合もある。元々のＴＤ学習は、予測問題の逐次的学習方法である。例えば、月曜日に次の日曜日に晴れる確率を予測することを学習によって獲得するという問題を考えた時に、実際に日曜日が来るまで学習しないのではなく、火曜日の予測値は月曜日の予測値よりも信頼性が高いという観点から、月曜日の予測値に対し火曜日の予測値を教師信号として学習し、火曜日の予測値を水曜日の予測値から学習するといったように、逐次的に予測の学習を行う方法である。これを $\bar{r}$ の予測値 $P(t)$の学習に適用すると、（７.１）式から

$$P(t) = \sum_{t=0}^{\infty} \gamma^t \, r(t+1)$$
$$= r(t+1) + \gamma \sum_{t=0}^{\infty} \gamma^t \, r(t+2)$$
$$= r(t+1) + \gamma P(t+1) \qquad (7.2)$$

を満たすようになればよいことになる。そこで、$P(t)$ に対して１単位時間経過後に得られる（７.２）式の右辺の値を教師信号として学習を行うことによって実現できる。これによって、倒立振子が倒れた時に罰を与えるだけで、倒立振子が倒れないような制御を獲得することができる。Barto らは、critic 部、actor 部共に、センサの入力を人間が予め決めた方法で場合分けし、それに対してテーブルルックアップ方式で $\bar{r}$ や動作を決定している。その後、Anderson らは、この学習に多層ニューラルネットを用い、学習にＢＰ法を用いることによって倒立振子が倒れるまでの回数が飛躍的に伸びることを示している[Anderson 89]。

本章では、この遅延強化学習の問題に対し、時間という概念を陽に捉えることによって、前述の時間軸スムージング学習の適用を試みた。この時間軸スムージング学習は、前述のように、センサ信号の統合化による空間情報の抽出にも用いることができ、より汎用的な学習則である。そして、これによって、前述の Barto らの方法とほぼ同様な機能を実現できることを示す。さらに、この学習アルゴリズムの問題点を整理し、より適応的な強化学習のアルゴリズムを提案すると共に、視覚センサ信号を直接入力した場合について検討する。

７．２ 学習アルゴリズム

システム（ロボット）の構成は、図７.１のように、強化信号検出部、状態評価部、動作生成部の３つの部分から構成され、状態評価部、動作生成部はニューラルネットによって形成する。そして、システムの状態を評価する状態評価部（評価関数）を遅延強化信号を用いて学習し、さらに、その

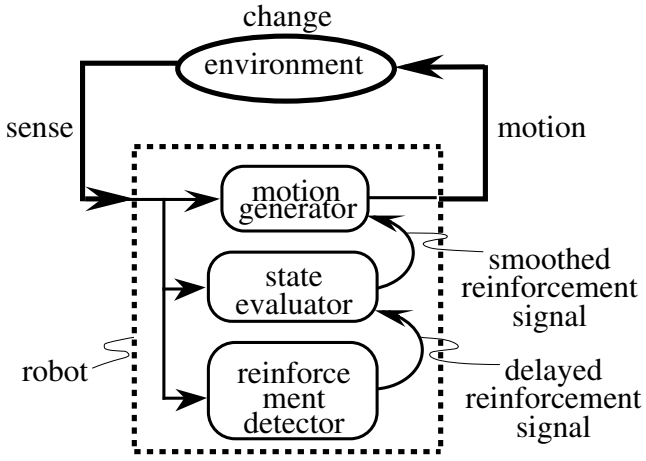


図７.１ 遅延強化学習システムの構成

評価値を用いて動作を学習する。この構成自体は、Barto らと同じものであり、crtic 部がここでは状態評価部 (State Evaluator)、acotor 部が動作生成部 (Motion Generator) に相当する。

7.2.1 動作の学習

ここで述べる動作の学習は、基本的に Williams らの方法 [Williams 88] と同様である。彼らは、確率的動作という観点から説明を行っているが、ここでは、試行錯誤という観点から説明する。簡単のため、ここでは、システムの状態が２つの連続的な変数で表されるものとする。そして、図７.２のように、２個の状態変数（x、y 軸）と評価関数値（z 軸）から形成される３次元空間を考える。ロボットは、A点から動作生成ニューラルネットの出力である指令動作（速度）ベクトル $\boldsymbol{m}$ に乱数ベクトル $\boldsymbol{rnd}$ を加えたベクトルに従って実際の動作を行う。この乱数ベクトル $\boldsymbol{rnd}$ は $\boldsymbol{m}$ に比べて微小でかつ確率分布は状態変数空間上で原点対称とする。これは、状態変数の時間変化を

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{m}) \qquad (7.3)$$

と表すことができ、この $\boldsymbol{f}$ が $\boldsymbol{m}$ によって微分可能であると考えればよい。そして、ある状態Aから微小な単位時間で動作できる領域を、簡単のため図中の斜線で塗りつぶされた円のように表わされるものとする。ここで、動作生成ニューラルネットの出力値に対する教師信号ベクトル $\boldsymbol{m}_s$ を、

$$\boldsymbol{m}_s = \boldsymbol{m} + \zeta \boldsymbol{rnd}\ \Delta\Phi \qquad (7.4)$$

$\triangle\Phi = \Phi(\boldsymbol{x}(t+1)) - \Phi(\boldsymbol{x}(t))$: 単位動作による評価値Φの変化量
$\boldsymbol{x}(t)$: 時刻 t でのシステムの状態、　ζ : 定数

とロボット内部で自動生成し、ニューラルネットを学習させる方法を考える。こうすれば、$\Delta\Phi$が大きい時の $\boldsymbol{rnd}$ の方向に動作がより強化されることになる。ここで、単位時間は十分に短いとし、$\Delta\Phi$が微小であったとすると、

$$\Delta\Phi = (\boldsymbol{m} + \boldsymbol{rnd})\nabla\ \Phi(\boldsymbol{x}) \qquad (7.5)$$

となる。そして、微小な乱数ベクトル $\boldsymbol{rnd}$ を変化させた時の教師信号ベクトル $\boldsymbol{m}_s$ の期待値 $\overline{\boldsymbol{m}_s}$ を求めると、

$$\bar{\boldsymbol{m}}_s = \boldsymbol{m} + \zeta \frac{\iint_{|\boldsymbol{rnd}|<rnd_{\max}} \boldsymbol{rnd}\{(\boldsymbol{m}+\boldsymbol{rnd})\nabla\Phi(\boldsymbol{x})\}\, d\,\boldsymbol{rnd}}{\iint_{|\boldsymbol{rnd}|<rnd_{\max}} d\,\boldsymbol{rnd}}$$

$$= \boldsymbol{m} + \zeta \frac{\iint_{|\boldsymbol{rnd}|<rnd_{\max}} (\boldsymbol{rnd}\cdot\boldsymbol{m}+|\boldsymbol{rnd}|^2)\, d\,\boldsymbol{rnd}}{\pi \cdot rnd_{\max}^2}\nabla\Phi(\boldsymbol{x}) \qquad (7.6)$$

となり、$\boldsymbol{rnd}$ が原点対象であることから、

$$\bar{\boldsymbol{m}}_s = \boldsymbol{m} + \zeta \frac{\iint_{|\boldsymbol{rnd}|<rnd_{\max}} |\boldsymbol{rnd}|^2\, d\,\boldsymbol{rnd}}{\pi \cdot rnd_{\max}^2}\nabla\Phi(\boldsymbol{x}) = \boldsymbol{m} + k\nabla\Phi(\boldsymbol{x}) \qquad (7.7)$$

$$k = \zeta\, rnd_{\max}^2 / 2$$

$\boldsymbol{m}$ から評価関数曲面の最急勾配方向に変化したものであることがわかる。これによって、常に動作生成と学習を区別せずに行うことができる上、動作ベクトルが評価関数の最大傾斜方向に確率的に変化していくことがわかる。ただし、動作可能範囲が状態変数空間上で凹部を持つと、動作がローカルマキシマムに陥る可能性があるが、それがなければ、与えられた評価関数に関して、最適動作へ近づいていくことができる。ただし、この学習には、評価関数曲面が微分可能であることが必要である。また、本論文中では、通常は小さい乱数で、たまに大きな乱数が発生させれるように、一様乱数の３乗を用いた。

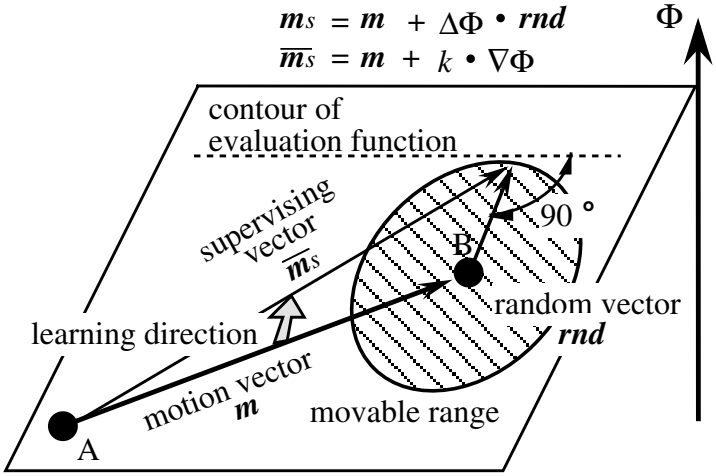


図７.２ 試行錯誤に基づく動作の学習

7.2.2 評価関数の学習

最初に、ロボットの状態が目的達成にいかに近いかを評価する時の規準について考える。まず、図７.３(a)のような２次元空間上の位置関係に、ロボットとターゲットがあるとする。まず、両者間の距離による評価が容易に考えつくが、この２つの次元の間でどう正規化すべきかを考えることは困難であるし、そもそも、距離を知る方法すらロボットは知らない。さらに、図のようにロボットとターゲットとの間に障害物がある等の場合は、目的達成度を単純に距離だけでは表わすことはできない。そこで、目的達成までに要する時間によって現在の状態を評価することを考える。こうすれば、図７.３(b)のように各状態は時間という１次元空間に投射され、正規化の問題も、障害物の問題も解決される。ところが、目的達成までに要する時間は、目的達成後しかわからない。しかし、自分が通った経路すべての場合のセンサ入力等の情報を保持し、経路をさかのぼって評価を学習することは、記憶容量などの面から大変に無駄が多い。

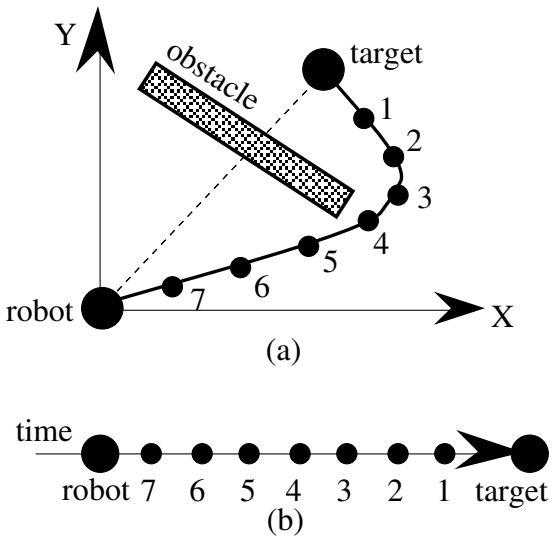


図７.３ 所要時間による評価

図７.４は、時間の変化に対する状態の評価値の変化の様子を示したものである。図中の太い点線のように評価値が初期状態から目的達成状態までの間、時間に対して直線的に単調増加すれば、目的達成までの所要時間で評価を行ったことになる。ところが、実際の評価関数の時間変化が実線のように変化したとすると、状態 A の方が、状態 B より評価が良いことになり、所要時間による状態評価に反する。このような評価の逆転は、評価値の時間変化に凹凸がある場合に起こる。時間変化に対する凹凸は、時間の２階微分値で表わされるため、これを０を近づけるという時間軸スムージング学習を行えば理想の評価関数に近づいていく。ただし、初期状態は評価値を低く、目的達成状態では評価値を高くなるように学習する。そこで、評価値の値域を０以上１以下とした場合、図中の矢印のように、初期状態の時に 0.1、目的を達成した時に 0.9 を教師信号として学習し、かつ、その他の部分では常に教師信号Φ_sを

$$\Phi_s(\boldsymbol{x}(t)) = \Phi(\boldsymbol{x}(t)) + \xi \frac{d^2\Phi(\boldsymbol{x}(t))}{dt^2} \fallingdotseq \frac{\Phi(\boldsymbol{x}(t-1)) + \Phi(\boldsymbol{x}(t+1))}{2} \qquad (7.8)$$

ξ: 定数

と内部で自動生成して学習させる。実際は、１単位時間さかのぼって学習を行うことになる。評価値の時間に対する２階微分値は、動作しながらその都度知るができるため、過去にさかのぼることなくリアルタイムの学習が実現できる。

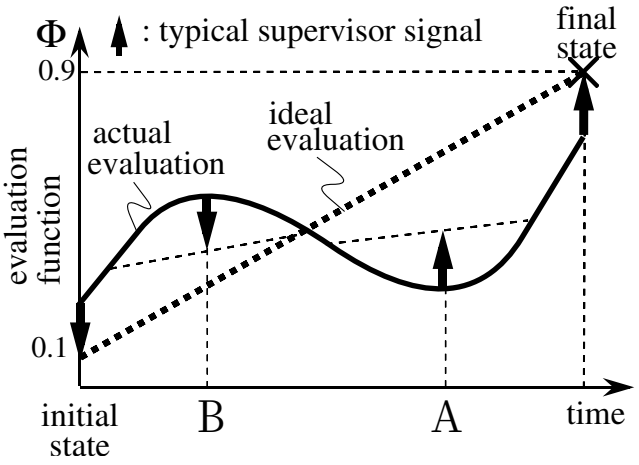


図７.４ 評価関数の学習

7.2.3 ２点間経路最適化の原理

本節では、この評価の学習と動作の学習を行うことによって、経路が最適化されることを示す。ある時、ロボットが、図７.５の経路(b) のように大回りをして点Ａから点Ｂまで行っていたものとする。この場合、点 A から点 B まで経路(a) のように真っ直ぐに進めば８単位時間で到達できるが、経路(b) では１１単位時間かかる。ここで、ロボットは自分の通った経路に対し、評価関数の時間変化を一定になるように学習する。また、評価の値は、ニューラルネットの出力であるため、その汎化能力によって自分の通った経路の近傍も滑らかになる。点 A と点 B の評価値は、どちらの経路を通っても同じ値となるため、この図の領域が微小な領域であるとすると、$d\Phi/dt$ の値は経路 A の方が大きくなる。すると、7.2.1 で説明したように、動作は学習により $d\Phi/dt$ の値が大きい方向へ変化していく。従って、経路は矢印のように、徐々に最適化されていく。

これによって、評価関数曲面に無意味な凹凸ができ、凹のところは避けて通り、凸のところに引き寄せられるようになった場合でも、凸の部分を行ったり来たりしているうちに、凸の部分は次第にへこみ、凹の部分を避けて尾根の部分を通るように学習した場合も、乱数成分によって尾根からはずれ、また尾根に引き戻されるという動作によって凹の部分は徐々に盛り上がり、前述のように

動作の最適化ができることがわかる。ただし、評価関数がターゲットから離れる方向に単調に増加しているようになった場合は、このロボットは単にターゲットから離れていくだけで、うまく学習できない。また、学習の初期では、乱数成分のみによる動作となるため、偶然目的を達成するまで待つことは非常に多くの時間を必要とする。従って、最初は簡単な学習から始め、徐々に難しくしていくという方法によって学習を加速させる。このような方法は逐次接近法と呼ばれ、実際の生物でもその効果が観察される[東 69]。

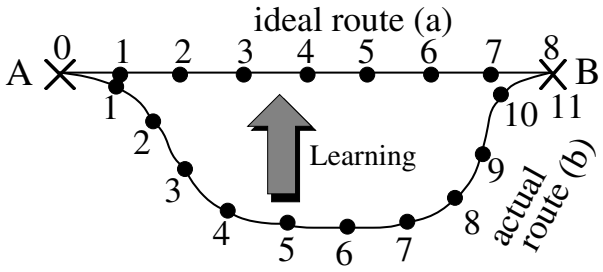


図７.５　２点間経路最適化の原理

7.2.4 ニューラルネットによる学習システムの構成と学習方法

上記のような原理によって動作するニューラルネットの構成と学習方法を示す。図７.６のように、外界からの情報を入力して評価関数を出力するニューラルネット（状態評価ネット）と、同じ入力から動作信号を出力するニューラルネット（動作生成ネット）の２つのネットワークからなる構成とする。ここでは、それぞれのネットワークは通常の階層型とし、出力は０から１の間の値で表わし、それぞれ自動生成された教師信号に近付くようにＢＰ法によって学習を行う。動作ネットの方は、評価ネットの時間変化量、つまり、時間に関する１次微分値を、評価ネット自体には２次微分値を用いて学習する。また、この時の学習は収束するまで繰り返すのではなく、１単位動作につき１回のみ繰り返す。そして、動作生成ネットの出力に乱数成分を加えた値に従って単位動作を行う。動作信号が複数ある場合は、それぞれに対して別々の乱数成分を加える。そして、この動作と学習を目的達成まで繰り返し行う。ただし、両ネットとも、学習を全く行っていない状態では出力層ニューロンへの結合の重み値を全て０とし、入力にかかわらず常に中間の値である 0.5 を出力するようにした。そして、この時の動作を０とし、当初は乱数のみの動作になるようにした。

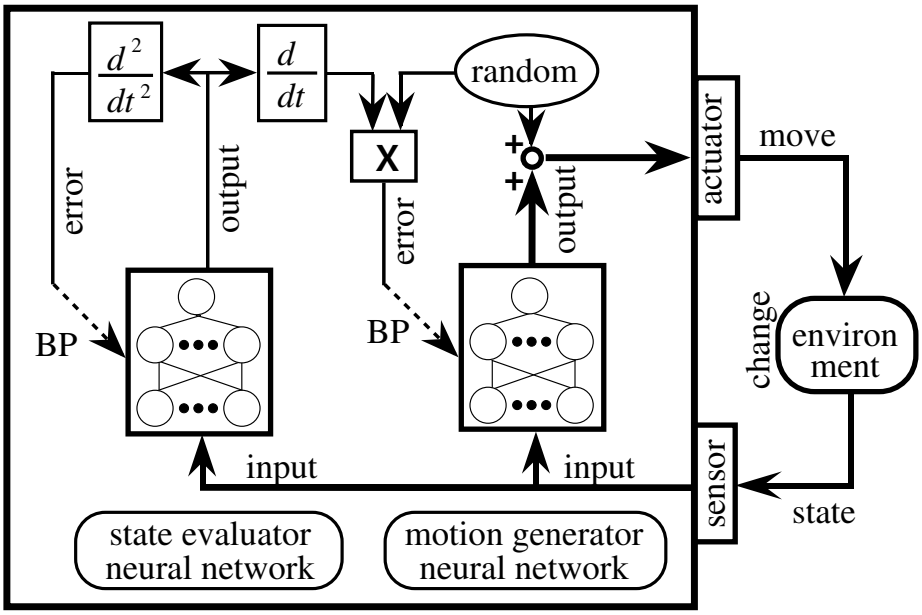


図７.６ 学習システムの構成

７．３ シミュレーション

7.3.1　経路最適化に関する基本シミュレーション

まず始めに、経路が最適化されるかどうかを確かめるための基本的なシミュレーションを典型的な２つの場合について行った。一つは、評価関数に最終目的点以外に凸部ができてそこにトラップされてしまった場合、もう一つは、評価関数に無意味な凹部ができて、そこを避けるように学習してしまった場合である。凸部にトラップされてしまった場合は、図７.７のように、試行錯誤の乱数成分の影響で、凸部を行ったり来たりすることになる。すると、時間軸スムージング学習によって、凸部は徐々にへこみ、最終的には凸部から抜け出すことができると考えられる。

そこで、まず、正方形の平面を考え、この上をロボットが動くような環境を考える。そして、ロボットは、この平面の x 座標と y 座標をニューラルネットに入力し、評価の出力と２つの動作の出力を計算するものとする。そして、２つの動作出力にしたがって、それぞれ x 方向と y 方向に動作するものとする。この時、まず始めに、平面上の中心の評価値が大きく、周辺に行くほど小さくなるように、平面上のいくつかの点に対して評価値の教師信号を与え、単純な教師あり学習で評価の学習を行うと共に、動作は 7.2.1 で述べた動作の学習にしたがって、より評価値が高くなる方向への動作を学習させる。この学習を行わせたものを図７.８(1) に示す。中心に評価関数のピークができ、さらに動作ベクトルはその中心に向かっていることがわかる。この状態でロボットを平面の中心に置き、前述の学習を行う。すると、かなり時間が掛かるが、最終的にロボットは尾根から脱出することができる。脱出した時の評価関数と動作の様子を図７.８(2) に示す。評価関数の尾根が低くなり、動作ベクトルの向きからロボットが抜け出せていることがわかる。この時の、学習の進行による評価値の最大値と最小値の変化の様子を図７.９に示す。この図からも、最大値と最小値の差が徐々に小さくなっていることがわかる。

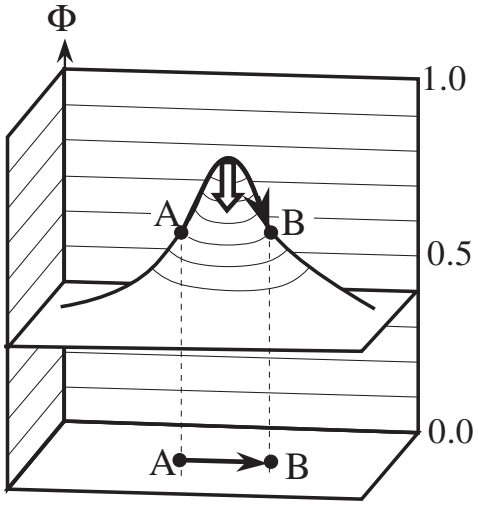


図７.７　評価関数の凸部にトラップされた場合

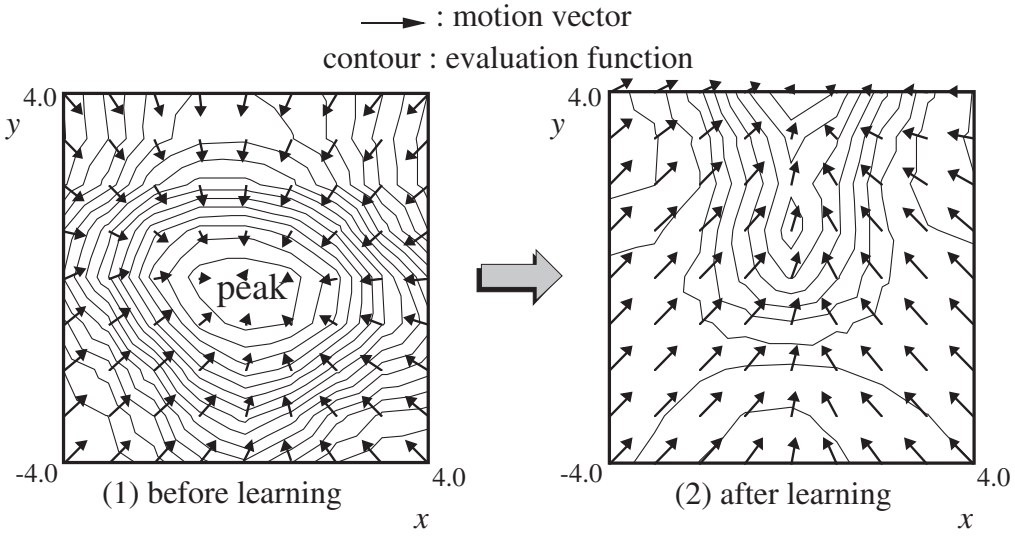


図７.８　評価関数の凸部にトラップされた状態から学習を積んだ場合

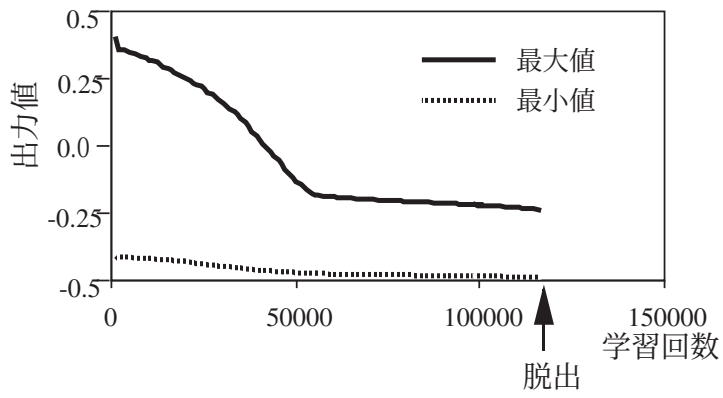


図７.９　学習による評価値の最大値・最小値の変化

次に、図７.１０のように、評価関数に無意味な凹部ができ、そこを避けるような経路を学習してしまった場合、経路の最適化が進むかどうかをシミュレーションした。この場合も、やはり試行錯誤の乱数成分により、ロボットが軌道からはずれるが、軌道からはずれて再び軌道に戻る際に、時間の変化に対して評価値の変化は凹になる。したがって、凹部の評価値は徐々に盛り上がり、最終的には評価関数曲面はフラットに近づき、経路は 7.2.3 で見たように、最適な経路に近づいていくと期待される。

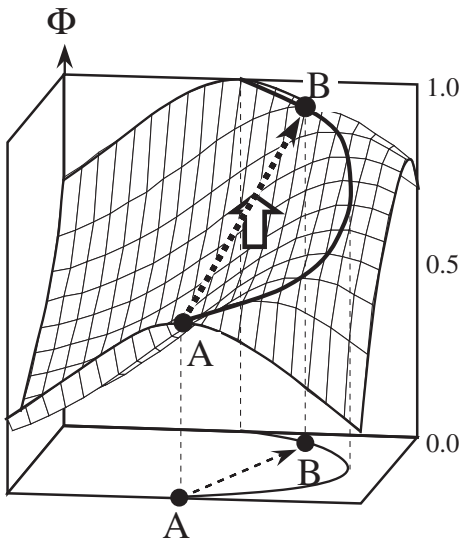


図７.１０ 評価関数に凹部ができた場合

そこで、今度は、まず評価関数の教師信号Φ_sを

$$\Phi_s = 0.4 \exp\left(-\frac{\left(y - \frac{3}{16}x^2 + 3.0\right)^2}{4}\right) + 0.05x - 0.2 \qquad (7.9)$$

として、ニューラルネットに学習させると同時に、動作もニューラルネットの評価出力にしたがって学習させた。この時、単位時間あたりの動作可能範囲を円形にするため、動作出力に対する教師信号ベクトルの大きさが 0.5 を越えた場合には、ベクトルの方向を変えずに、大きさを 0.5 に正規化してから学習に用いた。この時の評価と動作の様子を図７.１１(1) に示す。評価関数の尾根が放物線状になっていることがわかる。また、太い線がサンプルの軌跡を示しているが、ほぼ尾根に沿っていることもわかる。そして、次に、スタート地点を(x, y)=(-4.0, 0.0)とし、これを（７.９）式に代入した値である -0.2 を教師信号として評価の学習をし、その後、前述の学習則で評価と動作を学習し、x=4.0 に到達した時点で再び（７.９）式で求めた値を教師信号として評価を学習させた。これを繰り返した後の評価と動作の様子を図７.１１(2) に示す。この図より、学習を積むことによって、評価関数の凹部がなくなり、ほぼ最適（直進）と思われる軌跡を通ってゴールに到達するよう

になっていることがわかる。また、その際、評価関数の等高線は、経路（図中の太線）に関してほぼ等間隔になっていることがわかる。ただ、経路以外に関しては評価関数が学習されていないので、特に、評価関数が y が正の場合と負の場合で対称になっていない等、所要時間を表すようになっていない。この学習によるスタートからゴールまでの到達時間の変化を図７.１２に示す。所要時間８０がこの場合最適値であるが、学習によって、経路の最適化が行われていることがわかる。

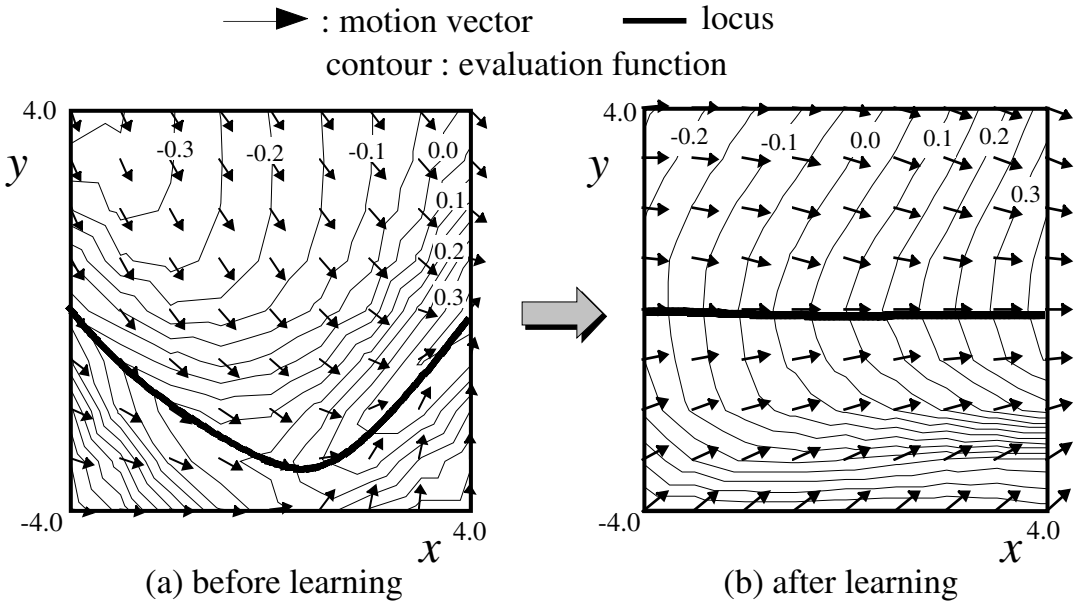


図７.１１　評価関数の凹部ができた状態から学習を積んだ場合

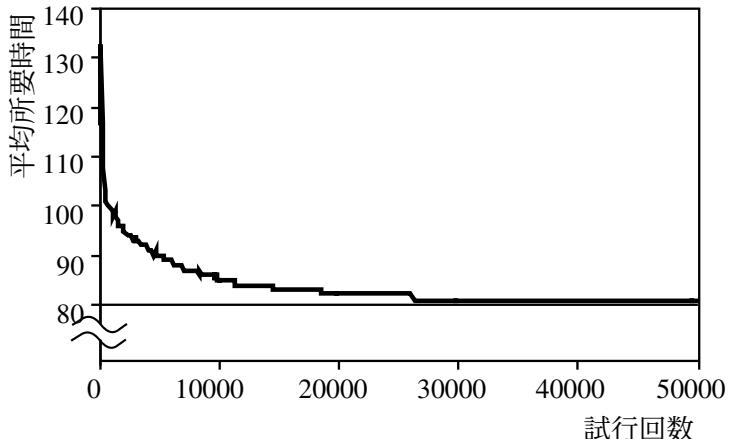


図７.１２　試行回数による平均所要時間の変化

7.3.2 非対称動作特性を持つ移動ロボットのシミュレーション

上記のアルゴリズムを検証するために、図７.１３のような環境で移動ロボットが目標物を取り込むという問題をシミュレーションした。ロボットは、図の位置に置き、目標物(Target) を図中の太線上にランダムに置く。そして、ロボットが動いていき、目標物に到達するまで、動作と学習を繰り返す。ここで、目標物に到達するとは、目標物の中心がロボットに接触した時とする。ただし、目標物が自分より後ろに行ってしまった場合は、失敗として教師信号 0 .1 を与えて学習させる。そして、到達するか失敗したら、そこまでを一試行とし、再びエサの位置をランダムに設定する。ロボットが得られる入力としては、ロボットから目標物を見た時の前方向と横方向の相対距離とし、動作生成ネットからの２つの出力にそれぞれ乱数成分を足した値で示された角度だけロボットの左右の車輪を回す。また、図７.１４のようにロボットの動作特性に非対称性を持たせた。具体的には、右側の車輪は、ニューラルネットの出力と乱数を足したものを３倍にした数に従って回転させ、左側は１倍して回転させた。また、このロボットは学習前は試行錯誤の乱数成分だけでしか動作をできないため、初めから目標物を遠くにおいておくと、いつまで待っても目標物に到達することができない。そこで、前述のように、簡単な問題から徐々に問題を難しくするために、ロボットがある時間経過しても目標物に到達できない場合は、目標物をロボットの近くに移動させるということを行う。

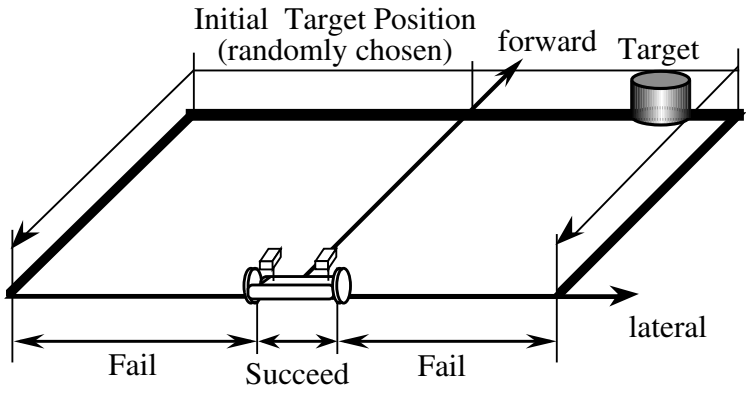


図７.１３ シミュレーションの環境

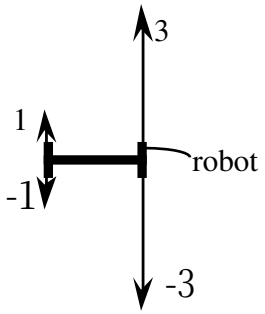


図７.１４ シミュレーションで仮定したロボットにおける非対称な動作特性

学習後の評価と動作の様子を図７.１５に示す。この図は、評価関数の値を表す等高線と、目標物を５カ所に置いたそれぞれの場合のロボットの軌跡を１０単位時間毎にプロットしたものである。ただし、この図は、ロボットを中心にした座標で描いてあるため、目標物の位置によって評価が決まると共に、ロボットの動作によって、相対的に目標物がロボットへ近づいてくる。これより、学習後は、形成された評価関数曲面の尾根が、ロボットの近くから徐々に左前方に伸びていることが

わかる。そして、ロボットは、目標物が遠くにある時には、まず自分の左前方に目標物が見えるように回転している。これは、前述のように、このロボットが右側の車輪をより速く回すことができるため、たとえ目標物がロボットの右遠方にあっても、回転の遅い左側の車輪をめいっぱい回転させて右前方に移動するより、一気に回転して左前方に目標物が見えてから、右の車輪を使って左前方に前進した方が有利であることを学習した結果であると考えられる。さらに、目標物が近くに来ると、逆に、目標物を自分の右側に持ってくる。これも、このロボットの動作特性によるもので、右の車輪をたくさん回転させることができるため、自分の近くにある時は、右側に目標物があった方が早く捕らえられるためであると考えられる。それに対し、比較のため、近い程良いという評価関数を予め与え、動作のみを学習させた場合を図７.１６に示す。この場合は、回転して自分の前方に目標物が見えるようにし、真っ直ぐ前進するという動作を学習している。以上の結果を、絶対座標に直したものを図７.１７に示す。評価関数を学習せずに動作だけ学習した場合は、評価関数と動作を学習させた場合よりも経路が直線に近く、一見よりよい経路に見える。しかし、ロボットが目標物に到達するのに要する時間は、後者の方が短い。また、図７.１８に 5000 試行後と 30000 試行後のロボットの経路を示す。学習を積むことにより経路が最適化が進んでいることがわかる。

図７.１９に試行の数による目標物到達までの所要時間の変化をプロットしたものを示す。縦軸は目標物を１２カ所に置いた時の平均所要時間を示している。また、1000 単位時間経過しても目標物に到達できない場合は、所要時間を 1000 とした。また、図中には、評価関数を与えた時と学習させた時それぞれについてニューラルネットの初期値を変化させた２つの場合をプロットしている。これを見ると、学習の初期は、評価関数を与えた方が早く目標物に到達できるようになっていることがわかる。ところが、試行を積んでいくと、評価関数を学習する方は、環境に適応し、経路の最適化が進むため、最終的には、重み値の初期値によらず、評価関数を学習させた方が速く目標物まで到達できるようになっていることがわかる。

このように、評価関数と動作を共に学習することにより、具体的な知識を与えることがなくても、学習を積むことによって最適に近い動作を獲得できることがわかった。そしてその動作は、時として我々が想像していないものであるが、それがロボットにとっては良い動作であることがわかった。

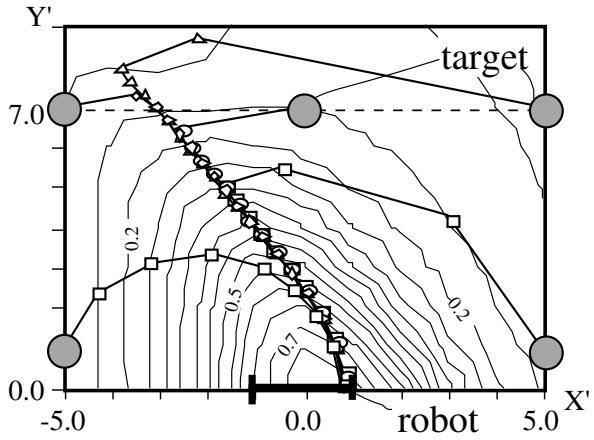


図７.１５ 学習後の評価関数とロボットの経路（ロボット固定の座標）

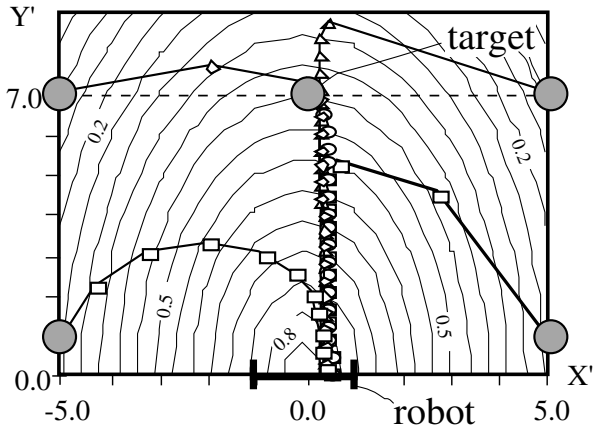


図７.１６ 比較実験（評価関数固定）における評価関数とロボットの経路（ロボット固定座標）

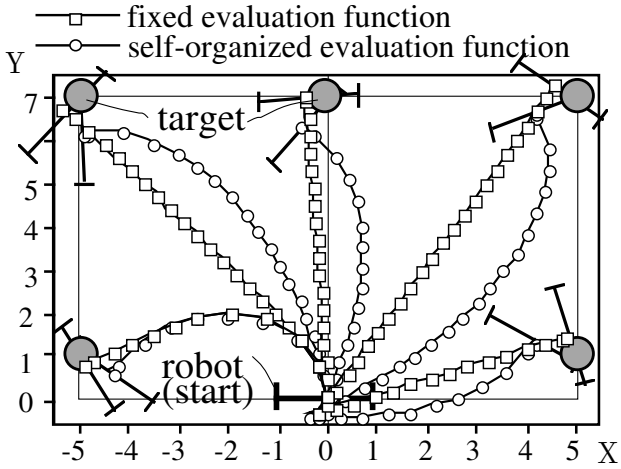


図７.１７ 評価関数を学習させた場合と与えた場合の絶対座標におけるロボットの経路の比較
（絶対座標）

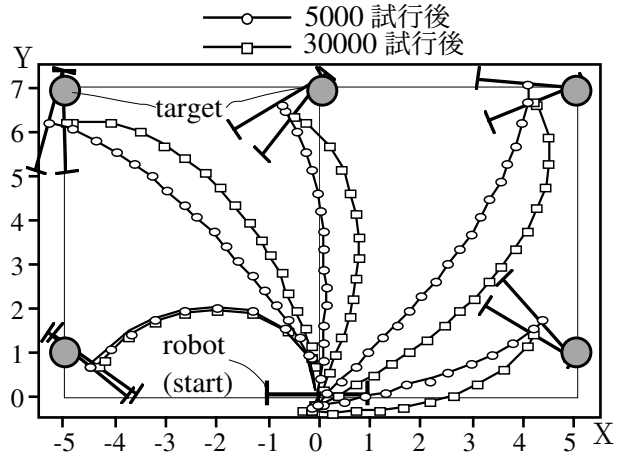


図７.１８ 学習によるロボットの経路の変化（絶対座標）

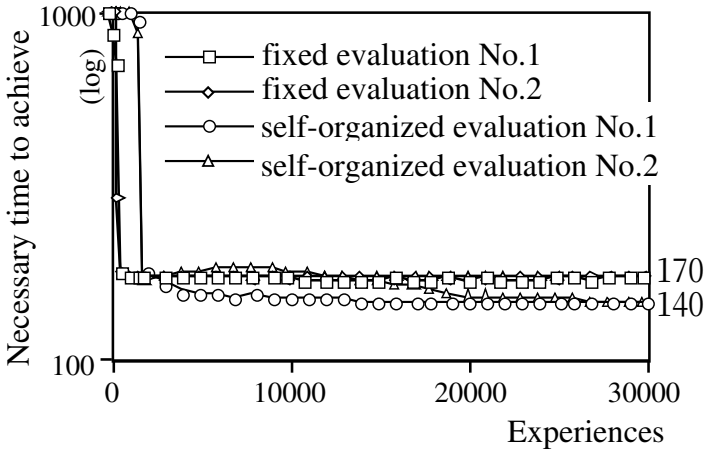


図７.１９ 試行による target 到達までの所要時間の変化

７．４ 試行により所要時間の異なる場合の評価と評価値の時間変化量一定化学習

前述のシミュレーションにおいて、ロボットの動作特性を非対称にする代わりに、ロボットと対象物との相対位置関係によって動作特性が変化するようにした。具体的には、ロボットから見た対象物の横方向の距離 x' によって動作特性を変化させ、ロボットの左側の方ではゆっくりしか進むことができず、右側では速く進むことができるようにした。すると、図７.２０に示すようにロボットの経路は最適化されず、ゆっくりしか進めない場所を通るような経路となった。この原因を探ってみた。図７.２１のように、左右対称の位置に対象物を置き、全く対称的な経路を通っている場合を想定する。この時、route(a) は時間が掛かり、route(b) は短時間で対象物に到達することができる。すると、評価の時間変化を滑らかにするだけでは、図７.２２のように、時間当たりの評価値の変化量が一定にならない。従って、所要時間による評価が正しくできないため、点 A と点 B の評価値を比較すると点 A の評価が高いという逆転現象が起こる。従って、ロボットの経路はより遅くなる方へ学習によって進んでいくことになる。

そこで、評価を時間軸に対して滑らかにするだけでなく、時間に対する評価の傾き $d\Phi/dt$ が一定になるように学習を行うこととした。具体的には、単位時間あたりの理想評価値変化量$\Delta\Phi_{ideal}$ を

$$\Delta\Phi_{ideal} = V / N_{\max} \qquad (7.10)$$

V：理想値域、ここでは 0.9-0.1=0.8、N_{max}：過去の最大所要時間（ただし、１次遅れで減衰させる）

によって求め、これと現在の評価値の変化量とを比較し、１単位時間前の評価値に対し、

$$\Phi_s(t-1) = \Phi(t-1) - \eta(\Delta\Phi_{ideal} - \Delta\Phi(t)) \qquad (7.11)$$

Φ_s：評価値に対する教師信号、　$\Delta\Phi(t) = \Phi(t)\text{-}\Phi(t\text{-}1)$、　η：学習のための定数

という教師信号を生成して学習を行い、さらに、現在の評価値に対し

$$\Phi_s(t) = \Phi(t) + \eta(\Delta\Phi_{ideal} - \Delta\Phi(t)) \qquad (7.12)$$

という教師信号によって学習を行うことによって評価値の時間変化量を一定化することにした。具体的には、現在の評価値の変化量が理想値より大きい場合は、１単位時間前の評価値を上げて現在の評価値を下げ、逆に、現在の評価値の変化量の方が小さい場合は、前の評価値を下げて現在の評価値を上げる学習を行う。

これによって学習した結果を図７.２３に示す。環境は、前述のように、ロボットから見た対象物の位置である X' 座標が大きいほど速く進めるように設定した。この図を見ると、ロボットが速く進める場所を通る経路を学習によって獲得できていることがわかる。

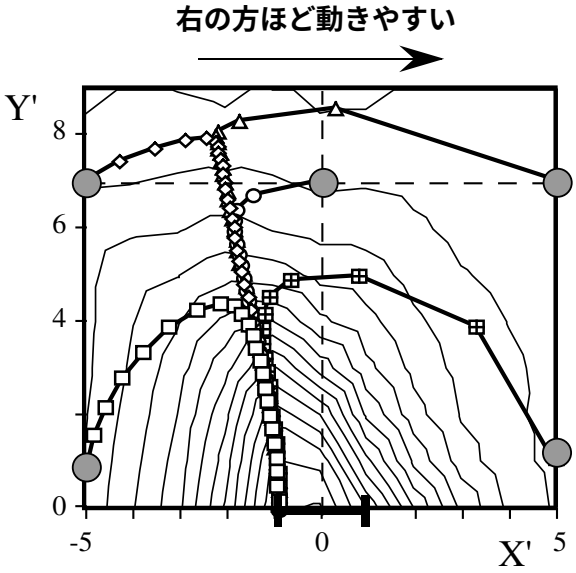


図７.２０ 非対称動作環境での学習後の評価関数とロボットの経路

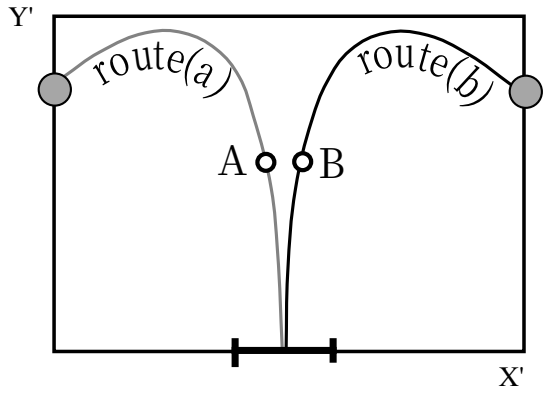


図７.２１　非対称動作環境での対称的な経路

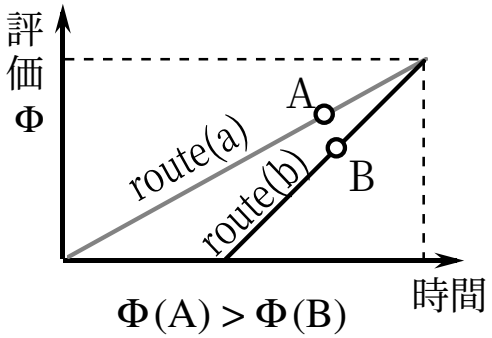


図７.２２　非対称動作環境で、対称的な経路をとった場合の評価関数の時間変化

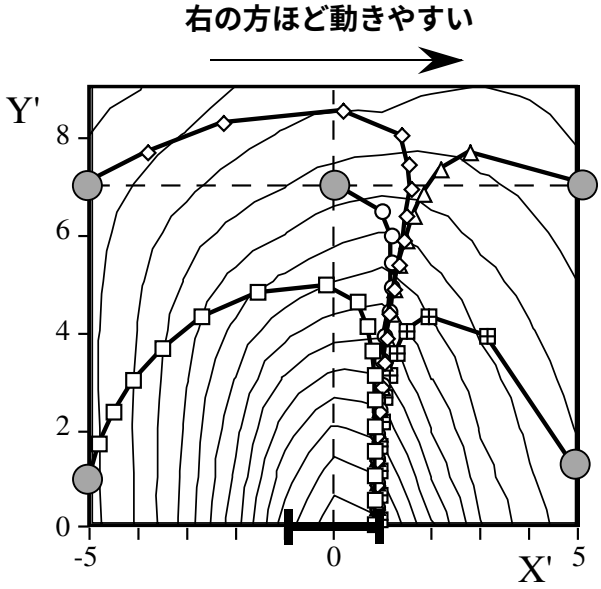


図７.２３　評価値の時間変化量一定化学習の際の非対称動作環境での学習後の評価関数と経路

７．５ 視覚センサ信号を入力とした場合のシミュレーションと評価値の時間変化量一定化学習の学習方法

より現実に近いロボットを想定するということで、前述の移動ロボットが持つセンサとして視覚センサを用いることを考える。視覚センサを強化学習に適用した例としては、１.２節で述べたように、浅田らが行ったサッカーロボットの例がある[浅田 95]。ここでは、視覚センサの信号からボールやゴールの見える大きさ、左右の位置等を検出するプログラムを書くことによって、人間が予め状態空間を分割してやり、その分割された状態と動作のペアに対してQ学習を適用してシュートの動作を学習させている。しかし、遅延強化学習の考え方を利用すれば、視覚センサの出力から直接評価を学習することが可能であると考えられるし、ニューラルネットを用いて学習を行うことによって、より適応的に、また滑らかな状態の評価を行うことができると考えられる。

そこで、図７.２４のように、センサを視覚センサに置き換えて学習を行わせた。すると、図７.２５に示したように、ロボットが目標物にたどりつく直前にループに陥ってしまい対象物までなかなか到達しないという現象が見られた。このループに陥った時には、ロボットへの視覚入力が常に一定になっており、評価値の時間変化量を理想値に近づけようとしても物理的に不可能なってしまう。従って、所要時間での正しい評価が行えないという状況になり、経路の学習がうまく進まないということが判明した。そこで、これを解決するためには、ループやそれに近い状況に陥った時には、周辺の評価値を下げるという学習を行うべきであると考えた。そして、傾き一定化学習を、式（７.１１）および式（７.１２）（図７.２６ の左の図）のように、理想の傾きと実際の傾きの差を求め、１単位時間前の評価値からそれだけ小さくなるように学習し、現在の評価値に対し、その分だけ大きくなるように学習を行う代わりに、図７.２６の右図のように、１単位時間前だけ式（７.１１）に従って学習を行い、現在の評価値に対しては学習を行わないようにした。これによって、経路がループになってしまうと評価の時間変化量が０に近づいていくため、経路全体で評価値が下がるという学習が行われる。また、完全にループにならない場合でもそれに近い状態になると評価値が下がるため、動作の学習によって、経路はループに近づかないように学習によって変化していくことになる。

そこで、この学習アルゴリズムを用いて、図７.２４のような視覚付き移動ロボットのシミュレーションを行った。視覚センサは、２４個の網膜細胞を１次元に配列した視覚センサを２個用意し、各網膜細胞は 7.5 度の受容野を持ち、全体で 180 度の視野を持つものとする。また、各網膜細胞は、受容野中で対象物が占める割合を０から１の範囲の連続値で出力するものとする。また、動作の非対称性はここでは用いていない。図７.２７に 700 回試行後と 10000 回試行後の評価関数とロボットの経路を示す。これを見ると、学習の初期には、個々の視覚センサが小さな受容野しか持っていない影響で、評価関数が放射状に広がっていることがわかるが、学習が進むことによって滑らかな評価関数が形成され、ロボットの経路もほぼ最適な経路になっていることがわかった。

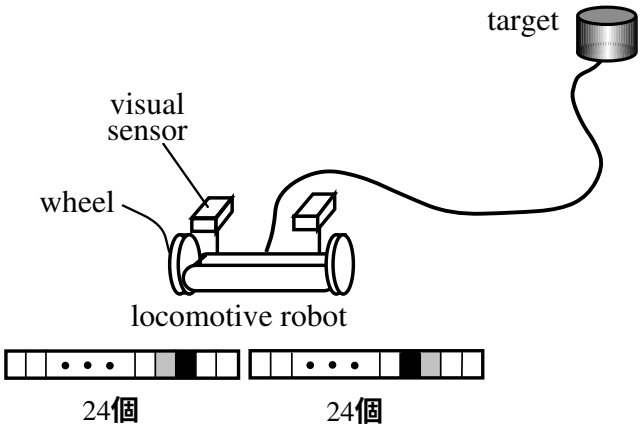


図７.２４ 視覚センサを持った移動ロボット

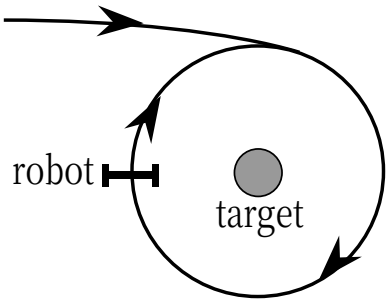


図７.２５ ループに陥った場合

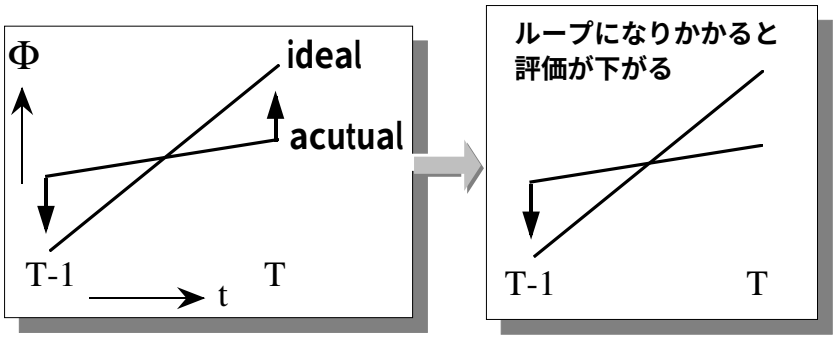


図７.２６ 評価関数の時間軸方向傾き一定化学習の実現方法

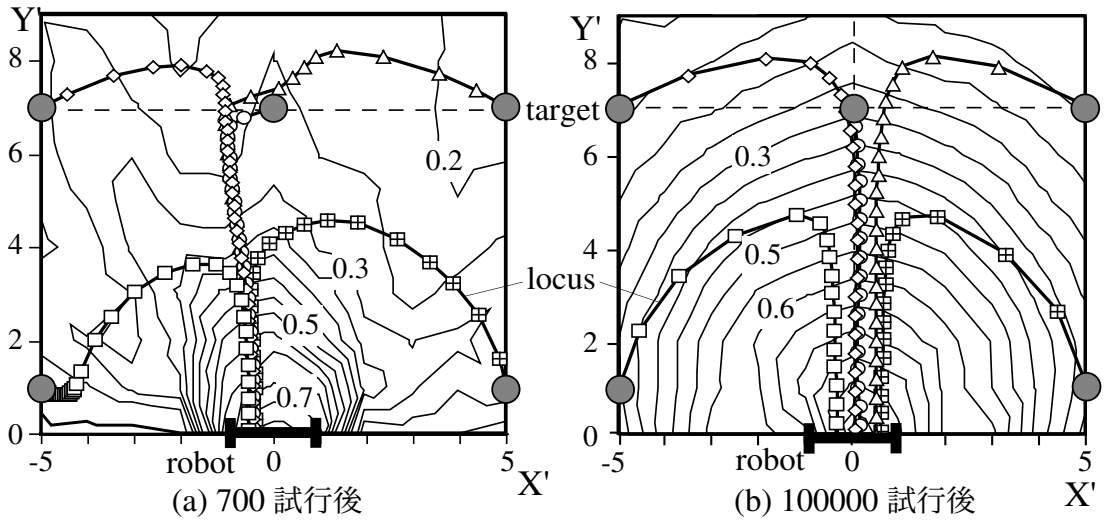


図７.２７ 視覚センサ付きロボットが学習した評価関数と経路

７．６ 中間層ニューロンにおける空間情報のコーディング

次に、このようにして学習したニューラルネットの中間層に意味のある情報がコーディングされているかどうかを調べる実験を行った。図７.２８のように、本来の強化学習を行うニューラルネットにテスト用の出力を加え、ニューラルネットの中の結合の重み値を微小な乱数で決定する。そして、まず、強化学習用の出力（評価用１個、動作用２個）に対し、前述の視覚センサ付き移動ロボット上で強化学習を行わせた。この時、テスト用出力は学習を行わない。そして、その後、今度はテスト用出力に図７.２９に示すような６つの点に関して、そこに物体を置いた時の視覚センサ信号をニューラルネットに入力し、その点の X'座標によって、X'座標が-5.0 の時に 0.1、5.0 の時に 0.9 という教師信号を与えて教師あり学習を行ってみた。学習後の、物体の位置に対する出力の分布を図７.２９(a) に示す。色が黒い部分が出力値が小さく、白い部分は出力値が大きいことを表す。これより、教師信号を与えていない場所でも、出力が X'座標によって滑らかに変化するという傾向が見られた。また、入力層から中間層への重み値を固定し、中間層からテスト用出力への重み値だけを学習させた場合もこれとほぼ同じ結果となった。これに対し、比較のために、強化学習を行わないで、強化学習を適用した場合と同じ結合の重み値を持つ初期ネットワークに対し、教師あり学習だけを行った。すると、図７.２９(b) のように、出力値のきれいな分布は見られず、出力値の尾根と谷がロボットの位置から放射状に広がるようになった。これは、視覚センサの各センサセルの受容野が放射状に広がるため、その受容野の中に物体がある状態で学習をすると、その受容野の中に物体がある場合全てに対して学習が影響を及ぼすためと考えることができる。このことから、強化学習を行うことで、中間層で知識の抽象化が行われ、X' をコーディングするニューロンが生まれていたと考えられる。このことは、知識の抽象化の能力を示しているのみならず、知識の共有の可能性を示しているものであり、非常に興味深い結果であると言える。

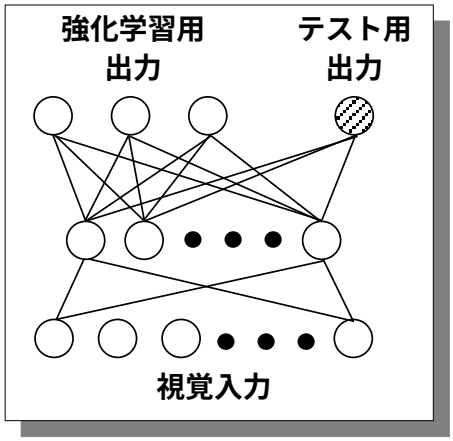


図７.２８ 中間層ニューロンにおけるコーディングのテストのためのニューラルネットワーク

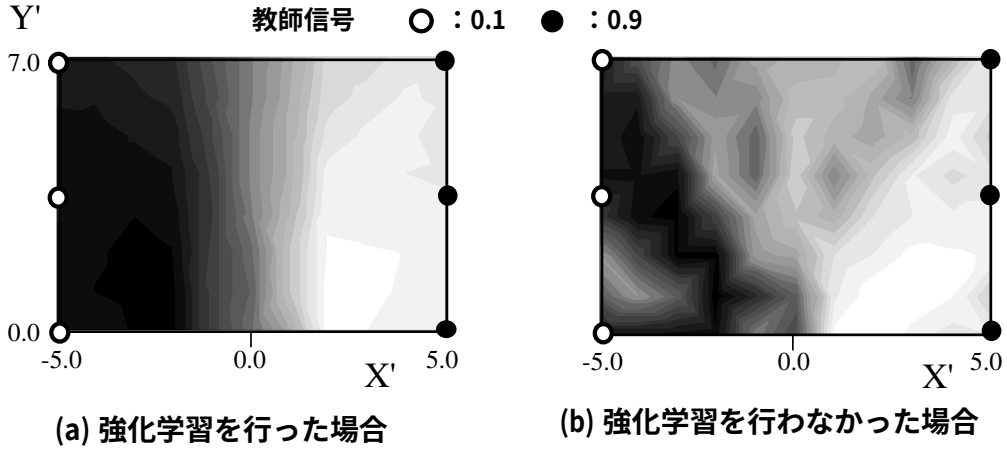


図７.２９ 中間層ニューロンにおけるコーディング

次に、中間層ニューロンが実際に空間の情報をどのようにコーディングしているかを調べた。図７.３０のように、５層のニューラルネットを用意し、恒等写像ニューラルネット（第３章参照）のように、真ん中の中間層ニューロンを２個にしぼり、このニューラルネットに視覚情報を入力し、強化学習を行うことによってこの２つの中間層ニューロンの中で空間の情報がどのように分布しているかを調べた。

そして、何回か学習した後に目標物を図７.３１の格子上に置いた時の視覚信号をニューラルネットに入力した時の中間層ニューロンの値を図７.３２に示す。この図は、横軸に中間層ニューロン１の値、縦軸に中間層ニューロン２の値をとり、図７.３１の格子が中間層ニューロン空間でどのように表されているかをプロットしている。４つの図は、強化学習の進行によってこのコーディングがどのように変化しているかを表している。図７.３２中の１から５までの番号は、図７.３１中の特徴的な目標物の位置５つにつけた番号に相当している。この図より、

（１）学習が進むにつれ、空間の情報を表すために中間層ニューロン空間の広い範囲を使用するよ

うになってくる

（２）空間の情報の変化と共に中間層ニューロンの値が滑らかに変化するようになる

（３）ロボットにとって重要なところ（目標物がロボットに近い状態にいる時）を拡大して表現している

（４）目標物を置いた領域が長方形で、中間層ニューロン空間も正方形の領域であるにもかかわらず、４５度回転した形で中間層ニューロンが目標物の位置を表している。

ということがわかる。（２）は、評価値を時間に対して滑らかにしようとする学習によって、空間情報も滑らかに表現するようになっていると考えられる。これは、第４章での局所センサ信号の統合の際と同じである。ただ、第４章で見た空間情報の表現とは、（３）のように、目的達成のために重要なところが拡大されるという点が大きく違う。また、（４）は、（３）の影響で、ロボットの近くが重要であり、そこを拡大する力が働いた結果であると考えることができる。

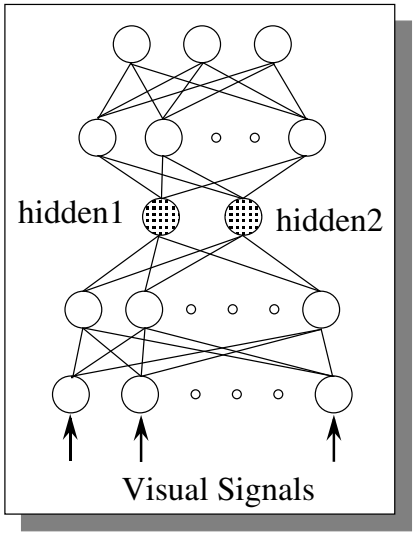


図７.３０ 中間層ニューロンのコーディングを調べるためのニューラルネット

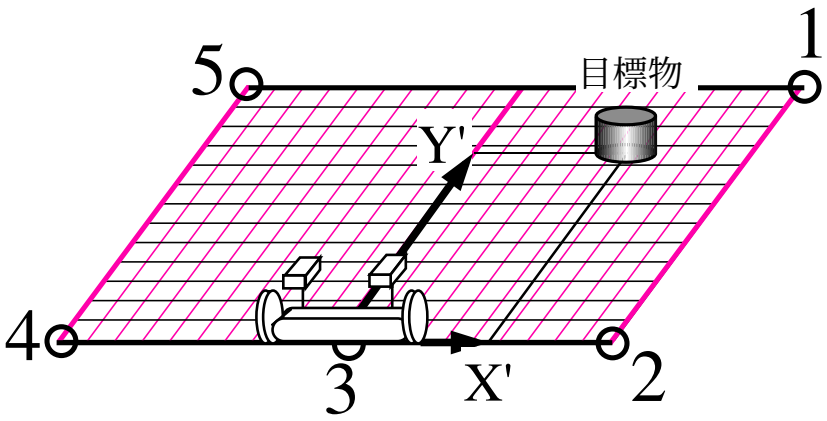


図７.３１　目標物を置いた格子

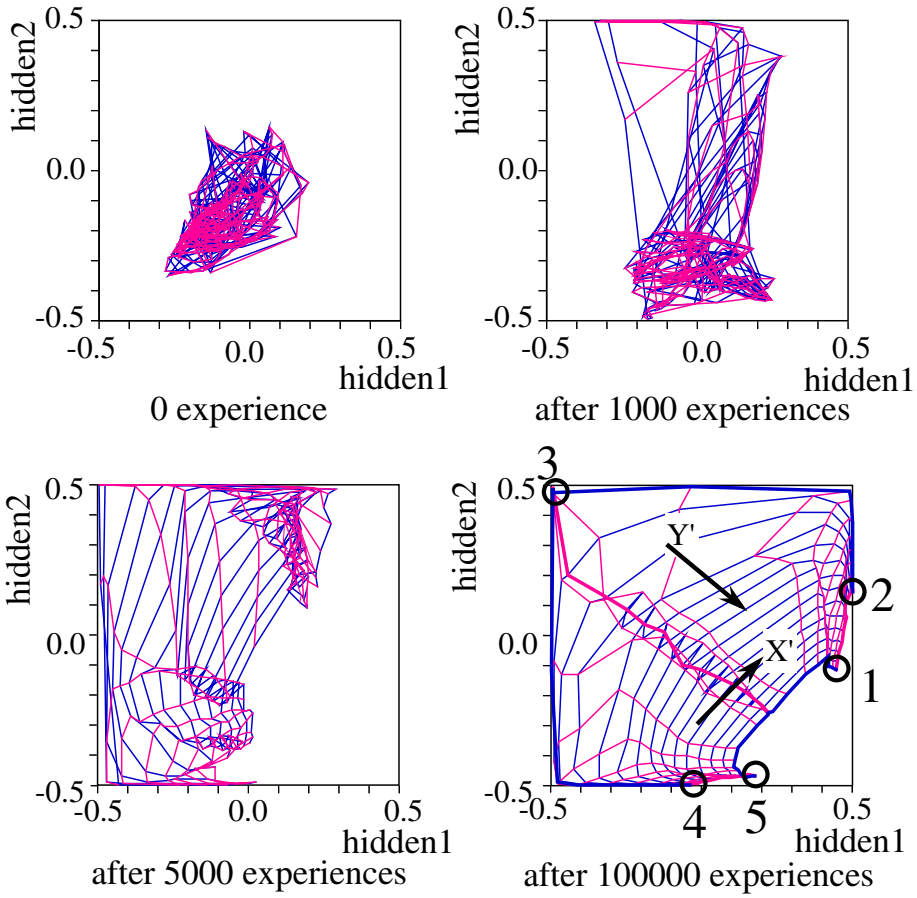


図７.３２　強化学習の試行回数による中間層ニューロンにおける空間情報コーディングの変化

さらに、これを逆に、目標物の位置を軸として、各中間層ニューロンの値の分布を示すと図７.３３に示す。この図より、中間層ニューロン１が X'=1 のあたりで、ニューロン２が X'=-1 のあたりで値の分布が密になっていることがわかる。これは、このロボットが目標物を捕らえるかどうかの境界が X'=-1 および X'=1 にあり、この境界の内側ではロボットは直進すれば良いのに対し、外側では回転して目標物が境界の内側に入るようにしなければならないという動作の違いを必要とする。そのため、中間層の２つのニューロンが分担してその境界の内と外を検出するように学習されたと考えられる。また、２つの図を重ねたものが図７.３３の下の図であるが、中間層ニューロン１と２の分布がほぼ直交していることがわかる。

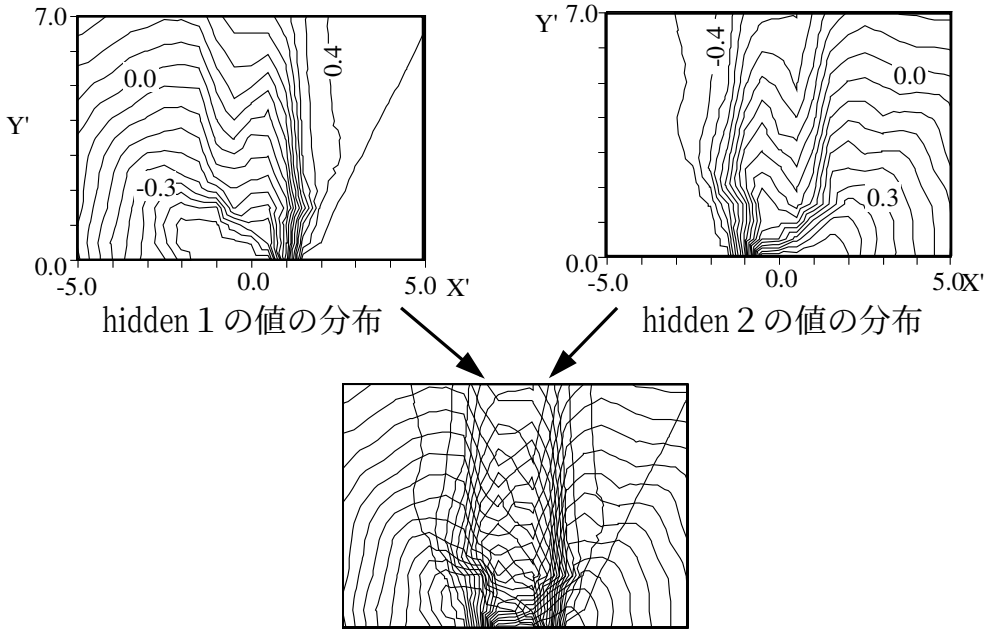


図７.３３ 目標物の位置に対する中間層ニューロンの値の分布

次に、このニューラルネットの入力を７.３節のシミュレーションのように、ロボットと目標物との相対位置を入力して学習させたところ、うまく学習ができなかった。これは、入力が連続値になったことで逆に空間の切りわけを２つの中間層ニューロンでうまく表現できなかったためと考えられる。

次に、図７.１４のような非対称動作特性を持つロボットについて、同様に３層目の中間層が２個の５層のなニューラルネットを用いて学習を行った。この場合、学習があまり安定しないが、学習中で比較的良い経路を通った時（７８０００試行後）の評価関数と経路を図７.３４に示す。５層にした場合でも、３層の場合とほぼ同様に、ロボットは左前方に物体見えるように回転した後に左前方に前進していることがわかる。また、３層目の２つのニューロンにおける物体の位置のコーディングを　図７.３５に示す。この図でロボットの正面を表している線（３から延びている太めの薄い線）を見ると、中間層での表現では、物体がロボットの近くにある場合は物体が右側にある場合を、

物体がロボットから遠くにある時には左前方を拡大されていることがわかる。この表現法は、ロボットに非対称動作特性を持たせない場合と比べて変化しており、かつ、右に動くか左に動くかまたは前進するかの境界のあたりが拡大して表現されていることがわかる。このことから、中間層の表現が、単に視覚センサの受容野が放射状に広がっているという特性だけでなく、動作特性、または動作系列によって適応的に変化することわかった。また、ここで学習が不安定なのは、中間層での物体の位置の表現が非対称になるため、２つの中間層ニューロンの空間（正方形）の中にきれいに収まらないということが一つの要因として考えられる。また、５層の場合でも中間層のニューロン数を増やせば学習は安定した。

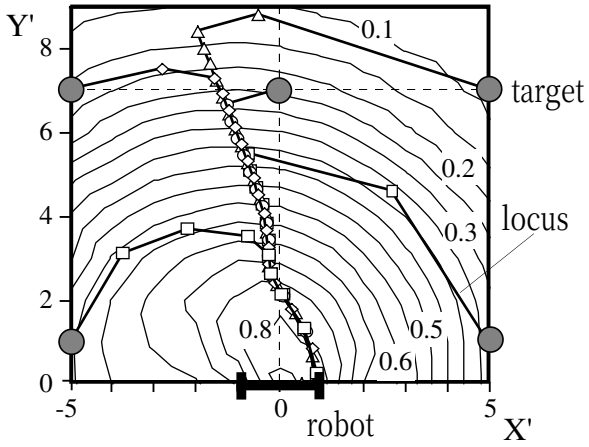


図７.３４　非対称動作特性のロボットが５層のニューラルネットで学習した場合の評価関数と経路

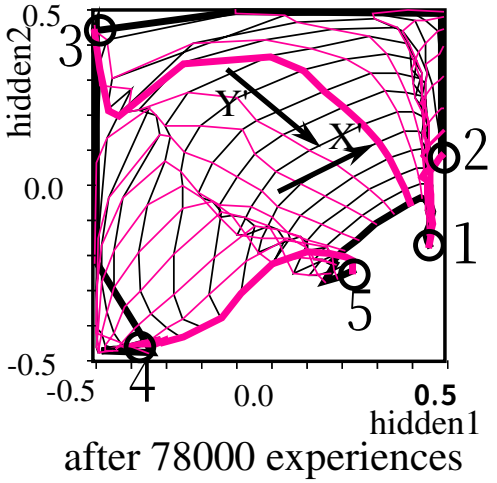


図７.３５　非対称動作特性のロボットが学習した場合の中間層ニューロンにおける物体の位置のコーディング

７．７　障害物回避のシミュレーション

最後に、ここまでのアルゴリズムで障害物回避ができるかどうかをシミュレーションで確認した。ここでの問題は、障害物の前後で状態が大きく変わってしまうが、その不連続さをどのようにニューラルネットで学習させるかにある。例えば、目標物および障害物との相対位置（合計４入力）を入力とし、６.２で述べた時間軸スムージング学習によるアルゴリズムで学習させると、障害物に当たると、前に進まなくなったり、一旦障害物の右側を通って目標物に到達することを学習すると、目標物が障害物より左側にあっても障害物の右側を通ろうとする等学習がうまく進まなかった。これの原因として、入力が障害物との相対距離で表しているため、障害物の前後で入力は連続となってしまう。しかし、ニューラルネットはシグモイド関数という入力に対して出力が滑らかにしか変化しない関数を用いて非線形変換を行うため、入力値の差が小さい時に、出力値の差を大きくする

ことは困難である。ところが、視覚センサを導入することにより、障害物の手前では、視覚センサに障害物が映るが、一旦障害物を越えてしまえばセンサに障害物が映らないということで、障害物の前と後ろで視覚センサの入力が大きく変化するため、ニューラルネットを用いても障害物の前後での空間の分離が可能になることが期待できる。

まず始めに、障害物回避に近い問題として、ロボットから見た目標物がある一定の領域に入った時にロボットの速度が 1/5 に落ちるという状況でシミュレーションを行った。ただし、センサ入力は、物体との前後および左右の相対位置である。その結果を図７.３６に示す。この図も、ロボットを中心にした座標で描いてあり、斜線で囲んだ円形の領域に目標物が入るとロボットの速度が 1/5 になるという設定になっている。外部から何も情報を与えていないにもかかわらず、学習の結果、斜線の領域のあたりで評価値が落ち込んでいることがわかる。また、ロボットの正面に目標物がある場合でも、わざわざロボットが回転して、斜線の領域を避けるような経路をとっていることがわかる。また、目標物が斜線部より右にある場合は、斜線部の右を通って、左にある場合は、左側を通って目標物を捕らえていることがわかる。このことから、空間の分離がうまくできているということができる。さらに、一旦、斜線部を避けたロボットは、目標物をロボットの真ん中で捕らえずに、端の方で捕らえていることもわかる。これは、無駄な動作を避けているということが言える。

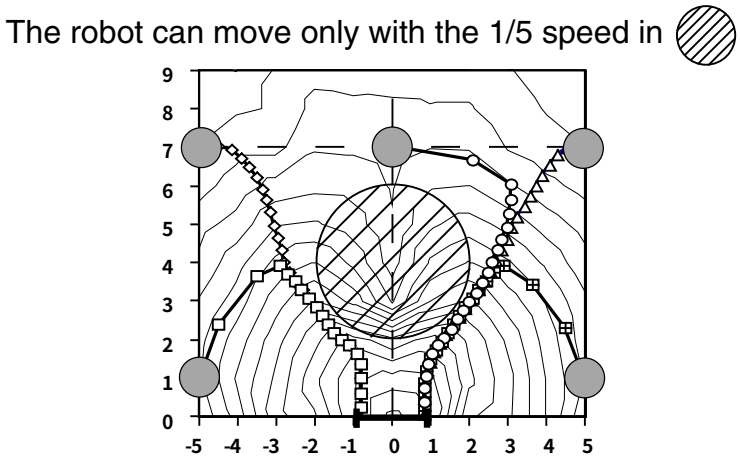


図７.３６ ロボットから見た目標物がある領域にある時にロボットの速度が遅くなるという設定の場合のシミュレーション

次に、実際の障害物を想定したシミュレーションを行った。シミュレーションの環境を図７.３７に示す。ここでは、少し不自然であるが、視覚センサを２種類用意し、片方は目標物のみ見え、もう片方は障害物のみ見えるという設定にした。そして、それぞれ左右１２個のセンサセルを１次元に並べ、合計４８個の信号をニューラルネットに入力した。また、目標物と障害物の位置は、各試行毎に乱数を用いて決定するが、目標物と障害物は最低１以上離すようにした。

結果を図７.３８、図７.３９に示す。両図とも、絶対座標で描いてあり、目標物や障害物は固定でロボットが動いていく。図７.３８は、障害物が視野に現れない場合に、目標物の位置を変化させ

た場合のロボットの経路を表している。ここでは、ロボットは最初に目標物の方に向きを変えて、その後ほぼ直進するという最適に近い経路を通っていることがわかる。一方、図７.３９では、ロボットのスタート地点の目の前に障害物がある場合の各目標物の位置に対するロボットの軌跡を表している。この場合には、図７.３８の場合とは明らかに違う経路をとり、障害物を避けて目標物にたどり着いていることがわかる。ただし、この学習は不安定であり、障害物の右側に目標物が見える時も障害物の左から回っていったりすることもある。また、右回りと左回りのちょうど境界あたりでは、ロボットは前に進めず立ちすくんでしまう。

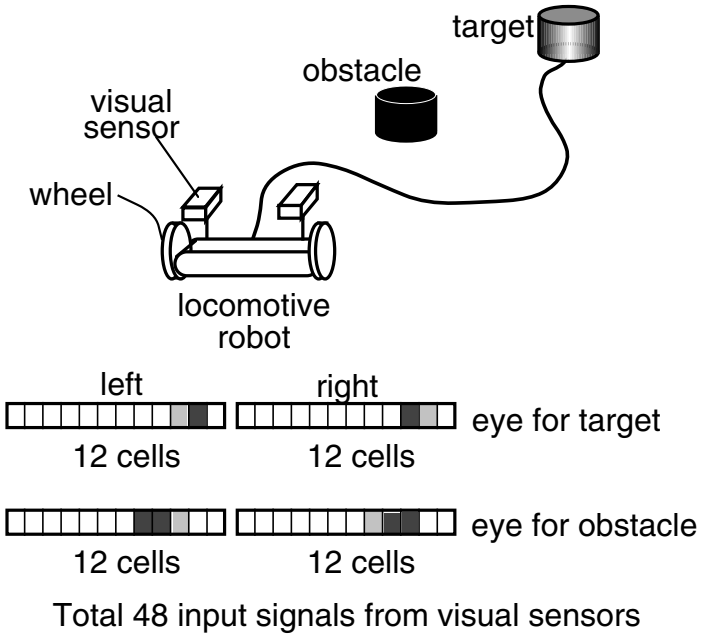


図７.３７　障害物回避のシミュレーションの環境

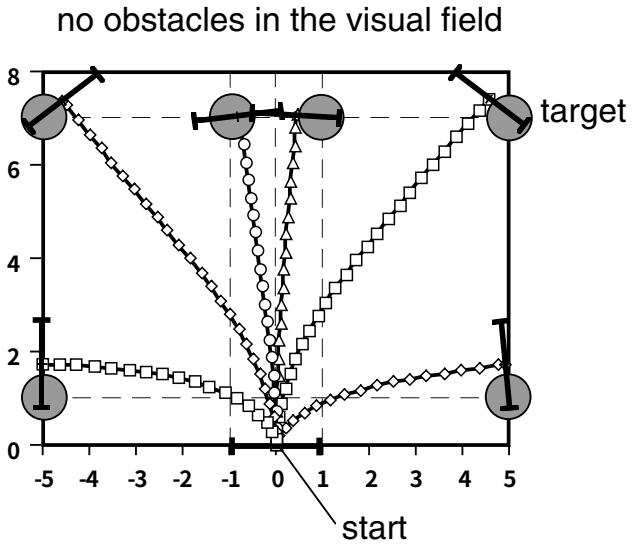


図７.３８ 障害物が見えない場合の各目標物の位置に対するロボットの経路

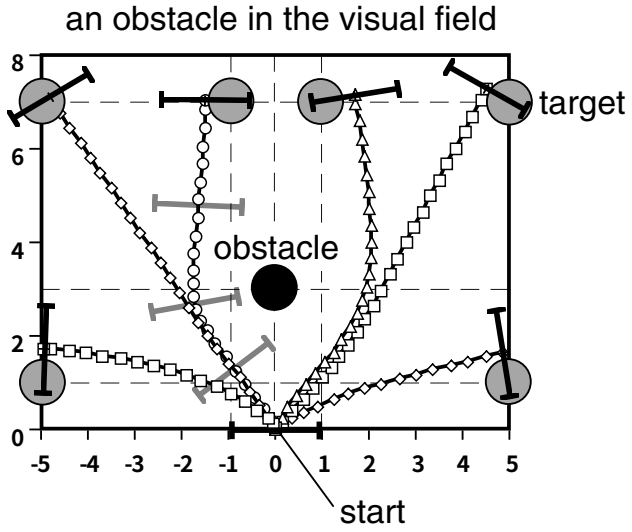


図７.３９ 目の前に障害物が見える場合の各目標物の位置に対するロボットの経路

７．８ ＴＤ学習との比較

ここで、遅延強化学習の評価の学習として従来一般的に用いられてきたＴＤ学習（７.１節参照）との比較を行う。本章では遅延強化学習を目的達成のための所要時間の最短化という問題として捉え、時間軸スムージング学習を拡張した評価値の時間変化量一定化学習を提案した。この学習は、評価値の時間変化量を適応的に変化させるという点を除くと、結果的にＴＤ学習と非常に近い学習則となった。ＴＤ学習は、将来の重み付き報酬和を最大化するという定式化であったが、これを、

単一報酬の場合に適用すると、評価値（（７.２）式の $P(t)$）の時間変化は図７.４０(b) のようになり、図７.４０(a) の本学習の場合と比較すると、ＴＤ学習は直線の代わりに指数関数曲線で所要時間を表現しているという見方ができることがわかる。逆に、本学習を複数の報酬源がある場合に適用しようととすると、傾きが一定のため、近い方を選択する等の機能が働かないことになる。これに関しては、我々人間を振り返ると、何らかの方法で予め目標を一つに絞って行動を起こしているように考えられることから、目標を一つに絞る機構を加えることにより解決できると考えられる。また、本学習とＴＤ学習を別の視点からみると、本学習は式（７.１）に示したＴＤ学習の強化信号の重みづけ総和のディスカウントファクタγを１とし、常時微小な罰を与えていると捉えることも可能である。

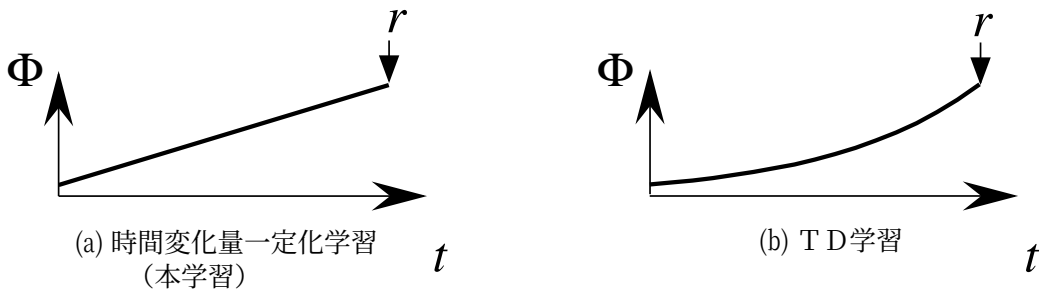


図７.４０　ＴＤ学習との評価値の時間変化の比較

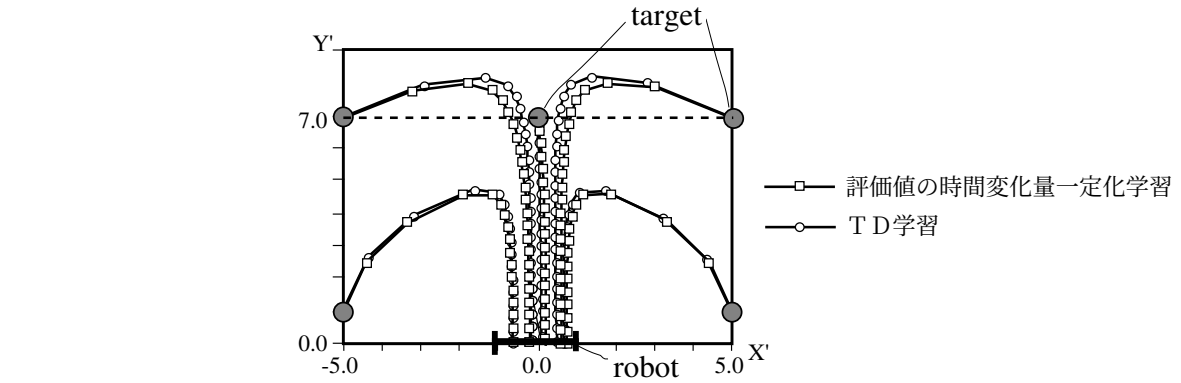
ＴＤ学習を指数関数による所要時間の評価と捉えると、所要時間が正しく学習できたとすれば、ＴＤ学習での評価値と本学習での評価値には１対１の関係付けをすることができる。また、微小線形近似を行うことにより、どちらの場合も評価値の時間に対する勾配が最急な方向に動作を学習させていくことにより経路の所要時間の最短化が実現できる。つまり、学習装置の能力が十分にあり、ローカルミニマにトラップされることなく十分な時間学習したとすれば、その解は等しくなるはずである。ただし、評価関数の形状は、ＴＤ学習ではゴールに近いほど急勾配に、本学習では一定勾配になるため、それが学習経過にどのように影響してくるか、また学習装置の能力が限定されていることがどのように影響するかが問題となる。そこで、前述の視覚センサ付き移動ロボットの問題のシミュレーションを行って両者の比較を行った。

まず、想定した環境は、前述のような図７.１３のような環境で、移動ロボットは図７.２４のような視覚センサを持ち、動作特性に非対称性は持たせなかった。そして、ＴＤ学習のほうも、条件をそろえるため、評価値の最大値を 0.9、最小値を 0.1 となるように、式（７.１０）の代わりに

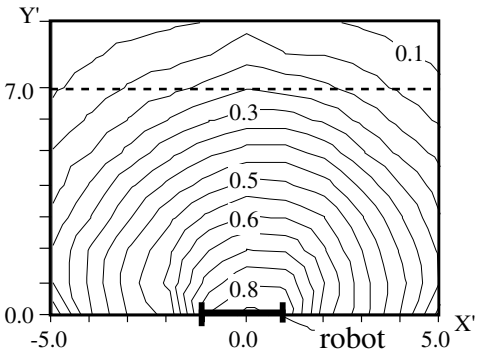
$$\gamma = \left(\frac{0.1}{0.9}\right)^{\frac{1}{N_{\max}}} \qquad (7.13)$$

からγを求めて学習させた。また、動作の学習は全く同様な方法を用いた。学習の結果を図７.４１

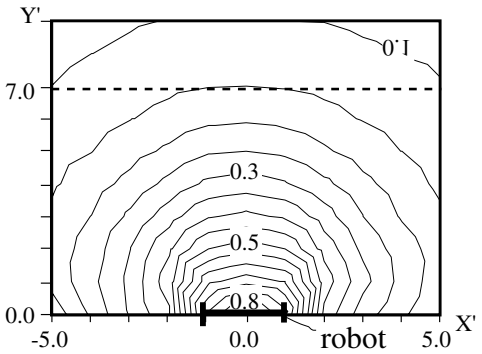
に示す。獲得されたロボットの経路は、図７.４１(a) のように、本学習とＴＤ学習でほとんど差は見られない。一方、評価関数の形状は、図７.４１(b)、(c) を比較してわかるように、ＴＤ学習では、目標物がロボットの近くにある場合は勾配が急に、離れるに従って勾配がゆるやかになっている。図７.４２にロボットが目標物に到達するまでの所要時間の学習による変化を、それぞれの学習を初期値を変えたものを２つずつ示す。ここからも、両者にほとんど差がないことがわかる。



(a) 評価値の時間変化量一定化学習を適用した場合とＴＤ学習を適用した場合のロボットの経路の比較



(b) 評価値の時間変化一定化学習で得られた評価関数



(c) TD学習で得られた評価関数

図７.４１　評価値の時間変化量一定化学習とＴＤ学習による学習結果の比較

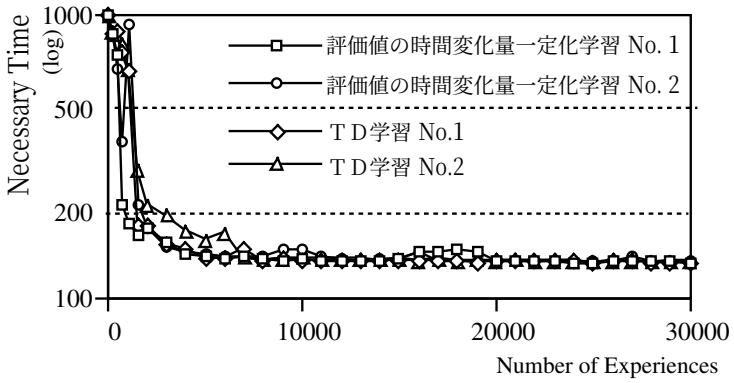


図７.４２　評価値の時間変化量一定化学習とＴＤ学習の学習曲線

７．９ 考察および今後の課題

７.６節では、時間軸スムージング学習に基づく強化学習により、４章で述べた局所センサ信号の統合に近いことがニューラルネットの中間層において実現できることがわかった。ただし、強化学習を行った場合は、評価や動作の変化が大きい状態を拡大して表現しているという特徴が見られる。また、４章の方法では、時間軸スムージング学習に値域拡大学習を併用しなければ、出力の値域に幅をもたせることができない上、第４章の結論でも述べたように、値域拡大学習を適用するためには、時間軸スムージング学習による連続的な学習を中止しなければならず、現時点ではあまり好ましい方法とは言えない。また、第４章のような学習を行うということは、強化学習におけるセンサ信号から動作に至る過程での中間層に対しての学習と捉えることもできる。このような中で、４章のような学習は敢えてする必要がないのか、それとも、学習を高速化させるためにも、４章のような学習が必要なのかという点も検討を要する課題である。それから、ＴＤ学習を行った場合の中間層のコーディングがどのようになっているかも今後調べるべきところである。

さらに、遅延強化学習をさせる時に、探索のために非常に学習に時間がかかるという問題がある。本章で述べたシミュレーションでも、物体に到達するという簡単な学習を行うのに、０から学習させるとは言え、１０００回以上の試行が必要であった。これをさらに効率化させるためには、単純な乱数による探索ではなく、より効率的な探索が必要であると考える。また、そのような探索は、学習を通して実現することが可能ではないかと筆者は考える。また、効率的な探索とは、過去の履歴を踏まえた系統的な探索が必要であると考える。そのためには、リカレント型のニューラルネットの適用がポイントであり、ここでも、リカレントニューラルネットの効率的な学習アルゴリズムへの期待が大きい。また、探索だけでなく、評価および動作自体も過去の履歴を踏まえたものにすることが望ましい。この意味からもリカレントニューラルネットの適用が望まれる。

今後はより複雑な問題への対応が必要となる。複雑なタスクを解こうとすると、目的達成までの所要時間が増大することにより、評価値の時間変化がどんどん小さくなっていってしまう。これには、我々がタスクに慣れると達成時の快感を感じなくなり、さらにその前段階の状態に対して快感を得るといったことが有効に働いていると考えられる。１.２節で述べたように、大脳基底核でこれに近い発火状況をするニューロンが発見されている。また、これは、ＴＤ学習の際の誤差に相当するのではといった見解も出ているが、今後さらに検討していく必要がある。

また、複雑な問題に対応していくためには、障害物を避けるという問題が解けることも重要である。障害物の問題は、単にロボットの行動における問題だけでなく、問題解決に対するあらゆる障害の回避に適用できると考えられる。本論文でもある程度障害回避ができるようになったが、障害物の前で立ち止まってしまうという動作も見られた。ニューラルネットを使うと、状態と動作の関係を不連続にすることが困難であるためと考えられる。このようなことから、状態をある程度シンボリックに表すことも必要と考える。

最後に、これも１.２節で述べたが、このシステムを実用化しようとした時に、所望の行動をさせるためにどのように強化信号を設定すればよいかといった問題がある。例えば、部屋の中のそうじをロボットにさせようとした場合でも、強化信号の設定は難しい。それをしないとロボット自身の生存にかかわるような問題であれば、遺伝的アルゴリズム（ＧＡ）を用いて強化信号を生成させることができるかもしれないが、いずれにせよ難しい問題である。

７．１０ まとめ

本章では、時間軸スムージング学習を基本にして遅延強化学習の評価を学習する方法を提案した。この学習では、所要時間の違う複数の経路での評価を平等に行うためには、時間軸スムージング学習を単純に適用するだけでなく、評価の時間変化量を一定化させることが必要であることがわかった。また、評価の時間変化量一定化学習を行う際に、より将来の評価値の方が正しいという考え方に基づき、将来の評価値から過去の評価値を学習することが重要であることがわかった。

さらに、この学習を用いることによって、ロボットの動作特性を変化させた場合や状態によって動作特性を変化させた場合、予めそれに関する知識を与えることなく状況に適した動作を学習することを示した。そして、時には設計者自身が意図していないような動作を身につけることもわかった。そして、予め知識を与えた場合と比較して、学習に時間はかかるものの、本学習では、経路の最適化が行われることによって最終的には目的達成までの所要時間が短くなった。

また、入力を視覚センサ信号とした場合でもうまく学習ができ、中間層ニューロンに空間の情報を滑らかにコーディングしていることがわかった。この時、評価や動作が大きく変化する状態を中間層では拡大して表現していることがわかった。また、この時の入力層から中間層の部分を他の学習に用いることによって、強化学習で獲得された空間情報を利用できることがわかった。さらに、障害物を設けたシミュレーションでも、ある程度障害物を避けて通る様な経路を学習することができた。