修 士 論 文

高次元連続状態空間での強化学習における クリティカル状態を利用した適応的関数近似

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

二本 真

2007年3月

修 士 論 文

高次元連続状態空間での強化学習における クリティカル状態を利用した適応的関数近似

指導教官　東条 敏 教授

審査委員主査　東条 敏 教授
審査委員　鳥澤 健太郎 助教授
審査委員　島津 明 教授

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

510089 二本 真

提出年月: 2007 年 2 月

概 要

強化学習は機械学習のひとつで，学習主体 (エージェント:agent) が環境との相互作用によって最適な行動系列を習得するための手法であり，幅広い分野で応用されている．強化学習では行動の良し悪しを判断する教師は必要とせず，代わりに，設計者が予め目標の状態に設定する報酬を便りに学習する．これにより，設計者にとって未知な環境でも報酬を設定 (報酬関数の定義) するだけで，エージェントが最適な動作を試行錯誤により自動で獲得できるという利点を持つ．しかし，探索空間が巨大になると，計算機メモリを大量に消費するだけでなく，適切な行動系列を獲得するまでに非常に時間がかかる．また，強化学習は現状態から今後得られるであろう報酬の期待値を状態の評価値とし，評価値に基づいて行動選択を行っているため，報酬までに長い行動系列が必要となる場合，適正な評価値を得るまでに多くの試行錯誤を要せねばならない (報酬の遅れ)．遅れが小さい報酬の設定が理想であるが，対象のタスクについての知識が必要となり，一般に難しい．こうしたなかで，より効率の良い学習アルゴリズムが求められている．従来の強化学習では，どのように状態空間を制限し，学習効率を向上させるかが大きな位置を占めていた．例えば，石井らは 2 足歩行をリズム運動であると仮定し，両手，両足，腰などの運動が同期するとして 状態探索空間を限定し，2 足歩行の強化学習を行った．しかし，この状態空間制 限は 2 足歩行にのみに有効であり，他の問題に応用できるものではない．このように，従来の強化学習では，問題に特化した方法を用いて探索空間を制限しているが，自律的な学習という強化学習本来の利点が損なわれるため，問題の性質に依存した手法は好ましくない．本研究では，問題の性質に依存せずに，より高次元な状態空間を持つ問題に対して適用できる，ゼロベースでの強化学習システムの構築を目指す．

強化学習では，方策関数や状態価値関数を近似し，それらを徐々に変更していくことで学習を行う．方策関数とはエージェントの行動出力を決定する関数で，状態価値関数とは，ある状態から未来においてどれだけの報酬が期待できるかを示す関数である．強化学習でよく用いられる近似手法のひとつに，正規化ガウス関数ネットワーク (normalized Gaussian network:NGnet) があげられる．NGnet は，ユニットと呼ばれる基底関数を状態空間に配置して，その発火量を用いて関数の近似を行う手法で，状態空間に配置されたユニットは，エージェントがユニットの中心に近い場所にあれば強く発火し，中心から遠い場所にあれば弱く発火するので，滑らかな曲線での近似が可能である．しかし，NGnet で状態空間が非常に大きなものの近似を行おうとすると，その状態空間を埋め尽くす数のユニットが必要になるので，大量の計算機メモリが必要になるという問題がある．もし，状態空間の大きさに見合った数のユニットを配置しないで近似を行う場合は，精度の悪い結果しか得られない．これにより，強化学習のにおいて，方策関数，状態価値関数の近似精度は，学習の精度自体に大きく関わる重要な要素なので，学習する問題の状態空間が大きいと，必然的に学習自体の精度も悪くなってしまうのである．そこで，本研究では，ユニットをより効率的に用いることで，状態空間の大きさに対して少ないユニット数で

も，関数近似の精度を悪化させず，安定した学習結果が得られる手法を提案する．本研究では，ユニットを効率的に用いる手段として，ユニットが配置されている状態空間上の位置に着目した．NGnet を用いた関数近似において，近似誤差は状態空間上の全ての場所で一定ではなく，近似誤差が大きい場所や小さい場所があると考えられる．また，状態空間が大きな問題では，エージェントがあまり通ることのない状態が多く存在しており，そのような場所に配置されたユニットは，学習中にほとんど発火することがないと考えられる．それらのことから，状態空間には，ユニットを配置することで学習のために有効である場所とそうでない場所があると考えられ，学習に有効な場所だけにユニットを集中させることができれば，無駄に使われるユニットがなくなり，少ないユニット数でも安定した学習結果が期待できる．

従来の手法を用いてポールバランシング問題を対象に予備実験を行った結果, エージェントの挙動が安定するか不安定になるかのターニングポイントであるクリティカル状態が確認された. 本研究ではこのような場所を, 学習中のサンプルをもとに自動的に検出し, ユニットを配置することで，学習の安定化を図った．また, 実際に強化学習システムを実装し，予備実験より広大な状態空間を持つ二足歩行問題を対象として実験を行った．その結果, 従来手法と比較して, たかだか 44.06 %のユニットで同等以上の学習精度を実現し, 提案手法の有効性を確認した.

目 次

第1章　序論

1.1　本研究の背景と目的

本研究では強化学習を研究対象とする. 強化学習とは教師なし学習の一種であり, 設計者が行動の良否を計る尺度としての報酬を設定しておくだけで, 理論的に制御を得ることが困難な問題においても目標出力を獲得可能な学習手法である [1]. 学習主体であるエージェントは自身が置かれている状況から得られる報酬の総和を最大化する制御則を習得する. この枠組みはゲームの戦略獲得 [2][3][4], ロボットの動作獲得 [5][6], システムの最適化 [7][8] など幅広い分野で応用されている.

強化学習では状態空間の次元数が増すと探索空間が膨大になるため，大量の計算機メモリを消費してしまう．よって，従来の強化学習ではどのように状態空間を制限し，学習効率を向上させるかが大きな位置を占めていた．例えば，石井ら [9] は二足歩行をリズム運動であると仮定し，両手，両足，腰などの運動が同期するとして探索空間を限定し，強化学習を行った．また，石井らは，探索空間を制限する代わりに，予めヒトの関節軌道データを与えることで学習の効率化を図った.

その一方で, ユーザがエージェントの試行錯誤を観察し, その中から望ましい試行を明示し, 報酬関数に作用させる三好らの研究がある [12]. しかし，逐一教示を与えたり, 探索空間を制限したり，予め知識を与えるような問題の性質に依存した手法は，汎用性に乏しく，自律的な学習という強化学習本来の目的から遠ざかってしまう．そこで本研究では，高次元な状態空間をもつ問題の学習を行うことが可能な，ゼロベースの強化学習システムを提案することを目的とする.

強化学習では，方策関数や状態価値関数と呼ばれる関数を近似することで学習をおこなっていて，それらの関数を近似する手法として, 用いられている一手法として NGnet である [9][10][11][12]．NGnet は，状態空間上にユニットと呼ばれる基底関数を配置して，それらの発火量を元に近似を行う．状態空間の次元数が高い問題では，その分の数の基底ユニットを配置しなければならず，もし，状態空間の大きさに見合った分だけの基底ユニットを配置せずに学習に用いた場合は，関数を近似する精度が足りなくなり，学習は安定しない．このような理由から，強化学習では，状態空間の次元数が増すと，大量の計算機メモリが必要になるのである．そこで，本研究では，基底ユニットの数を増やすのではなく，配置できる基底ユニットをできるだけ効率的に活用することで，関数近似の精度を向上させることを試みる.

本研究では，ユニットを効率的に用いる手段として，ユニットが配置されている状態空

間上の位置に着目した．NGnet を用いた関数近似において，近似誤差は状態空間上の全ての場所で一定ではなく，近似誤差が大きい場所や小さい場所があると考えられる．また，状態空間が大きな問題では，エージェントがあまり通ることのない状態が多く存在しており，そのような場所に配置されたユニットは，学習中にほとんど発火することがないと考えられる．それらのことから，状態空間には，ユニットを配置することで学習のために有効である場所とそうでない場所があると考えられ，学習に有効な場所だけにユニットを集中させることができれば，無駄に使われるユニットがなくなり，少ないユニット数でも安定した学習結果が期待できる．具体的には, エージェントの試行から, 制御が不安定になるきっかけであるクリティカルな状態を検出し, 基底の集中化を行う.

これにより，限られた計算機資源の中で，より高次元の問題を学習できるようになることが期待される．この手法は，問題の性質に依存することがないので，他の問題に用意に応用ができる．また，例に挙げた様な，他の手法と併用することができるという利点もある.

1.2 本論文の構成

本論文では，2 章において強化学習の概要を説明した後，NGnet を利用した連続系のアクター・クリティック法を紹介する．3 章では, 提案手法に必要な知見を得るために, 二足歩行問題よりはるかに簡単なポールバランシングでの実験を行う. 4 章では，提案手法の概要と手順の流れを説明する．5 章では高次元連続状態空間での強化学習タスクとして二足歩行問題を取り上げ，実験結果を検証することで従来手法と提案手法の比較を行う．そして，最後に本論文のまとめ，今後の課題を 6 章で示す．また，実際に実験で用いた学習アルゴリズム [10] を付録 A に記した.

第2章　強化学習に関する予備調査

2.1　強化学習とは

強化学習 (Reinforcement learning) とは，ある環境内において，エージェントがその時点での状態を観測し，とるべき行動を決定する問題を扱う機械学習の一種である．強化学習はニューラルネットワーク，パターン認識などで学習されているような教師あり学習 (Supervised learning) とは異なり，教師となるある状態における正しい行動の例などは必要としない．その代わり，ユーザが各状態に対して報酬を定め，エージェントはその報酬を手がかりに学習を行なう．一般に報酬はスカラー値で表され，望ましい状態には高い価，望ましくない状態には低い値を割り当てる (負の値を持つ報酬は罰と呼ぶ)．図 2.1 に強化学習の主な構成要素を示す．図 2.1 に示すように，通常，強化学習には方策 (policy)，報酬関数 (reward function)，評価関数 (valuefunction)，環境モデル (model) の４つの主な構成要素がある．エージェントは方策に基づいて行動を選択し，環境モデルによりその行動の結果となる次状態が決定する．この時，エージェントには報酬関数により導き出された報酬が与えられる．評価関数とは，エージェントがある状態から方策に従って行動した時，将来的に得られる報酬の総和を予測する関数で，評価関数の値が高くなるように方策を改善していくことで学習が行なわれる．

強化学習の最大の利点は学習目標を報酬というスカラー量を与えるだけで，任意の状態から目標状態に至る適切な行動系列が得られるという点である．環境に不確実性や道のパラメータが存在すると，目標達成への行動系列を設計者の手でプログラムするだけでなく，教師付き学習用の教師を用意するのも困難であるが，強化学習ではエージェントが試行錯誤を通して自動で解を得る．また，試行錯誤を通じて学習するため，人間が設計する解よりも優れた解を発見する可能性もある．特に環境がマルコフ性を満たす（マルコフ決定過程）場合，最適方策が得られることが保証されている．

2.2　離散状態・時間における強化学習

本節では, Sutton ら [1] が提唱した離散状態・時間における強化学習について説明する．環境のダイナミクスを有限マルコフ決定過程とした時の，離散状態・時間における強化学習を説明する．実際には厳密にマルコフ性が成り立つことは稀であるが，多くの場合はほぼ成り立つので，ほとんどの離散強化学習の環境は有限マルコフ決定過程として扱われている．

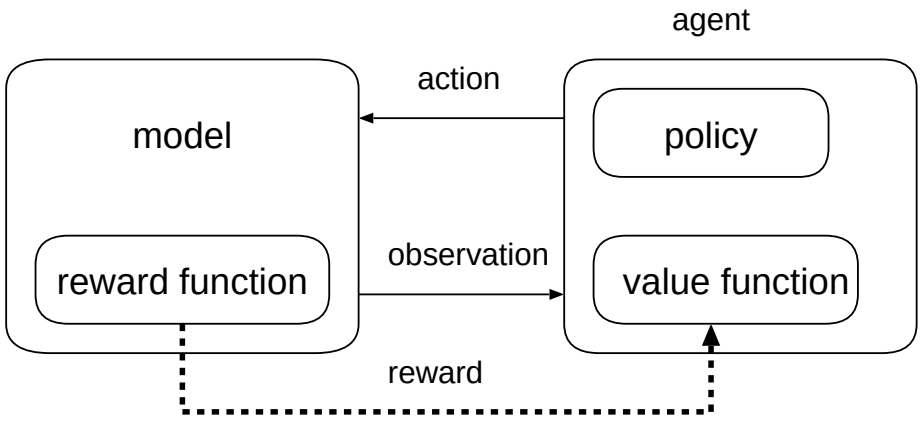


図 2.1: 強化学習の枠組み

マルコフ決定過程

ある環境において，任意の状態が将来を予測する能力を落とすことなく過去を集約しているならば，その環境はマルコフ性を満たすと言う．マルコフ性を満たす強化学習タスクはマルコフ決定過程（Markov decision process：MDP）と呼び，さらに，状態・行動が有限ならば，有限マルコフ決定過程（有限 MDP）と呼ぶ．強化学習では，対象のタスクが MDP であるならば，最適方策が得られることが保証されている．

離散系の強化学習では，時間をステップ単位で区切ることができる．エージェントは 1 ステップ内で，状態観測，行動選択をし，報酬を得て次のステップへ進む．

ある環境でとりうる状態の集合を $\mathbf{S} = \{s_0, s_1, \cdots, s_n\}$，行動の集合を $\mathbf{A} = \{a_0, a_1, \cdots, a_l\}$ とする．時刻ステップ t，状態 $s \in \mathbf{S}$ において，エージェントが行動 $a \in \mathbf{A}$ を選択し，時刻ステップ $t+1$ において状態 $s' \in \mathbf{S}$ へ遷移する確率は

$$Pr\{s_{t+1} = s' | s_t = s, a_t = a\} = \mathcal{P}^a_{ss'} \tag{2.1}$$

と表される．

この時，環境がエージェントへ与える報酬の期待値は

$$E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\} = \mathcal{R}^a_{ss'} \tag{2.2}$$

と表される．これらは時刻 t 以前の状態，行動に依存しない．

状態価値関数

ほとんど全ての強化学習法は状態価値関数（value function）に基づく評価を行なっている．この関数は状態を引数とする関数で，その状態から未来において，どれだけの報酬が期待できるかを表す．将来の報酬は方策 π に依存するため，状態価値関数は特定の方策

に関して次のように定義される．

$$
\begin{aligned}
V_\pi(s) &= E_\pi\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t=s\} \\
&= \sum_a \pi(a,s)\sum_{s'}\mathcal{P}^a_{ss'}(\mathcal{R}^a_{ss'}+\gamma V(s'))
\end{aligned} \tag{2.3}
$$

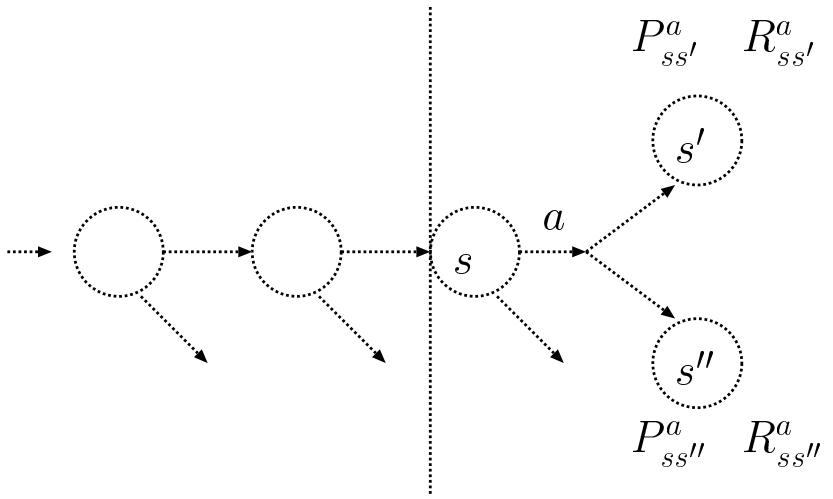


図 2.2: マルコフ決定過程

$\gamma(0 \leq \gamma \leq 1)$ は割引率と呼ばれ，未来の報酬に不確実性があることを考慮し，先読みした報酬をそのままには信じず，値を割り引く．

また，評価関数は，以下のような再帰的な関係を持っている．

$$
\begin{aligned}
V_\pi(s) &= E_\pi\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t=s\} \\
&= E_\pi\{r_{t+1}+\gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}|s_t=s\} \\
&= E_\pi\{r_{t+1}+\gamma V^\pi(s_{t+1})|s_t=s\}
\end{aligned} \tag{2.4}
$$

式 2.4 はベルマン方程式（Bellman equation）と呼ばれる．

方策評価と方策改善

状態価値関数は方策に影響を与える．最も単純な方策関数としてグリーディー方策（greedy policy）π' がある．グリーディー方策では次に示すように，遷移する状態の評

価値の期待値が最も高い行動を選択する．

$$\begin{aligned}\pi' &= \arg\max_a E\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s, a_t = a\} \\ &= \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')] \end{aligned} \tag{2.5}$$

ここで π は任意の方策である．

グリーディー方策 π' で得られる状態価値関数 $V^{\pi'}(x)$ と $V^{\pi}(s)$ は $V^{\pi}(x) \leq V^{\pi'}(x)$ の関係になることが保証されている．つまり，元の方策の評価関数に従ってグリーディーな行動を選択していくことで，その方策を改善する新しい方策を作り出すことができる（方策改善）．そして，新しい方策に従って行動を行い，得られた報酬から状態価値関数を作成する（方策評価）．方策評価・方策改善を繰り返すことで，最適な状態価値関数・方策を得ることができる．

報酬の遅れ

エージェントは報酬を得ることで，状態評価値を適正に更新する．しかし，報酬を得るまでに長い段取りが必要なタスクの場合，適正な評価値を得るために多くの試行を必要とする．これを報酬の遅れという．

例として，迷路の脱出を試みるエージェントを考える．脱出した場合のみに報酬を与えるとすると，脱出が成功するまで評価値の更新はされない．また，脱出が成功したとしても，評価値の更新は表週が伝搬されることによて行われるので，報酬が得られた（脱出に成功した）時間ステップから，時間的に離れた状態の評価値の更新はなかなかされない．

強化学習では報酬の遅れは必然的なものであるが，報酬の遅れが少ない報酬関数の設定で学習の効率化が期待できる．

2.2.1 TD 学習

TD 学習（Temporal Difference Learning）とは強化学習の中心となる新しい考え方の代表的なもので，後述のアクター・クリティック法など，多くの強化学習法の基本となっている．ここでは最も単純な TD(0) 法を取り上げる．

TD 学習とは，自分自身の評価を行い，それを更新するための手法である．具体的には，TD 誤差 (TD error) と呼ばれるものを用いて評価を行う．

TD 誤差 δ_t は

$$\delta_t = r_{t+1} + \gamma V(s_{t+1} - V(s_t)) \tag{2.6}$$

で表される．この式は 1 ステップ後の状態評価値が期待された値より良かったかどうかを評価している．従って，$\delta_t > 0$ の時，a_t を選択する確率が高くなるように方策を更新し，反対に $\delta_t < 0$ の時，a_t を選択する確率が低くなるように方策を更新すればよい．適正な

状態評価値が獲得されると，$\delta_t = 0$ となる．つまり，TD 誤差を 0 にするよう学習が進められる．

状態価値関数の更新

TD 学習では方策 π に従って TD 誤差を得ることで V^π の推定値 V を更新する．TD(0) 法での $V(s_t)$ の更新は 1 ステップごとに次の式によって行われる．

$$\begin{aligned} V(s_t) &\leftarrow V(s_t) + \alpha\delta_t \\ &= V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \end{aligned} \tag{2.7}$$

ただし，$\alpha(0 \leq \alpha \leq 1)$ は学習率と呼ばれる．推定値 V はいかなる方策 π に対しても，V^π に収束することが保証されている．

式が示すように，TD(0) 法では時刻 $t+1$ に観測される r_{t+1} と $\gamma V_t(s_{t+1})$ を目標として更新する．これが良くわかる例を図 2.3 として挙げる．図 2.3 において，1 回目の試行で，エージェントが全ての状態で行動 a を選択し，4 ステップで状態 s_0 から状態 s_4 まで遷移したとすると，報酬が得られる 1 ステップ前の状態 s_3 の評価値が上がる．この様子を表すグラフを図 2.4 に示す．

state :$\mathbf{S} = \{s_0, s_1, s_2, s_3, s_4\}$
action :$\mathbf{A} = \{a, b\}$

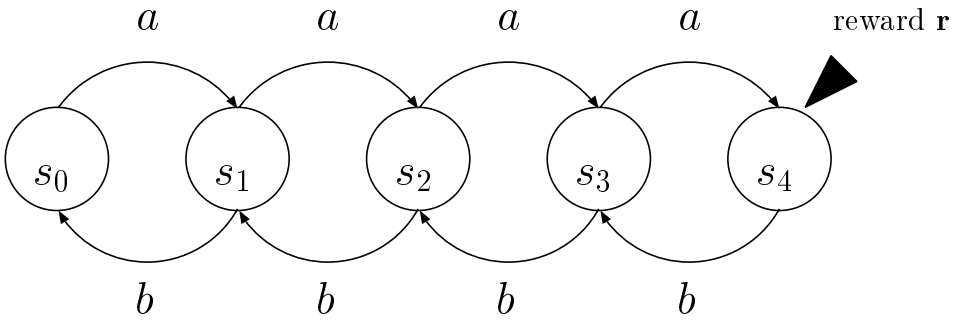


図 2.3: 状態遷移図

2 回目の試行で，再びエージェントが状態 s_0 から状態 s_4 まで遷移したとすると，図 2.5 のように評価値が更新される．このように TD 学習では，状態の評価値は報酬が伝搬することによって更新される．

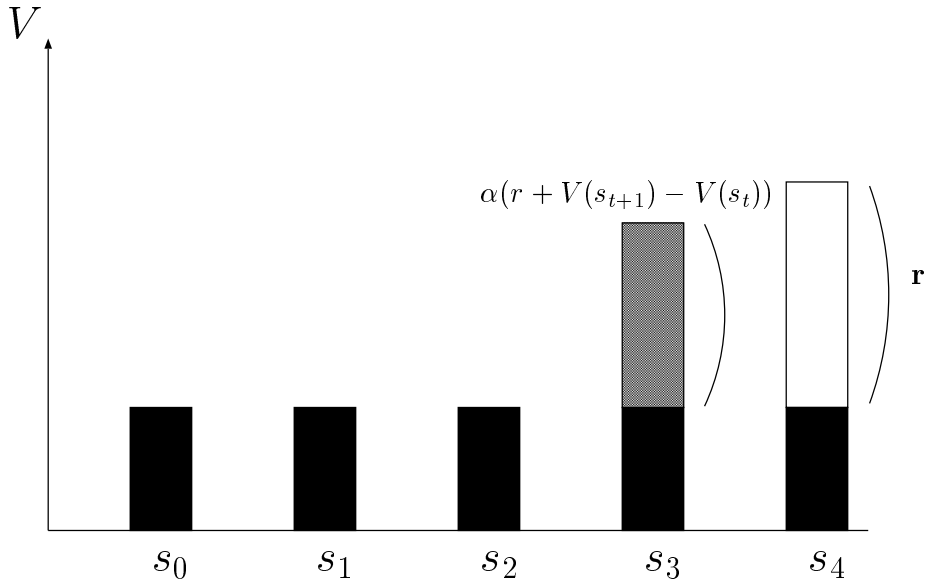


図 2.4: 1 試行後の状態評価値

2.2.2 アクタークリティック法

アクター・クリティック法 (Actor-critic methods) は TD 学習と違い，アクター (Actor) と呼ばれる行動選択部と，クリティック (critic) と呼ばれる状態評価部が独立して存在する強化学習法である．図 2.6 でアクター・クリティック法の構造を示す．

アクターは方策に従って行動を出力し，クリティックは現在アクターが利用している方策に対し評価を行う．この評価がクリティックの唯一の出力であり，アクターおよびクリティックの学習に関係する．アクターには離散系の場合，確率が設定され，連続系の場合は正規分布に基づく方法などが用いられる．これにより，一つの行動を選択するために競合集合の全てを探索する必要がなくなる．また，クリティックの評価は TD 誤差によって表され，TD 学習と同様に TD 誤差が 0 になるように方策を改善する．

2.3 連続状態・時間における強化学習

本節では, 銅谷ら [11] が提唱した連続状態・時間における強化学習について説明する. 離散系の強化学習では全ての状態に対応するテーブル形式の状態価値関数を扱っていた. この手法は単純で理解しやすいが，実際には状態数が小さいタスクでの利用に限られている．これは，単に大きなテーブルを格納するメモリーの問題だけでなく，テーブルを適正な値で埋め尽くすためには非常に多くの時間が必要なためである．

ロボットの制御のような滑らかな動作が求められるタスクでは，細かい状態表現が不可欠である．しかし，前述のように，離散系の強化学習では膨大な数の状態を扱うことは現実的に不可能である．

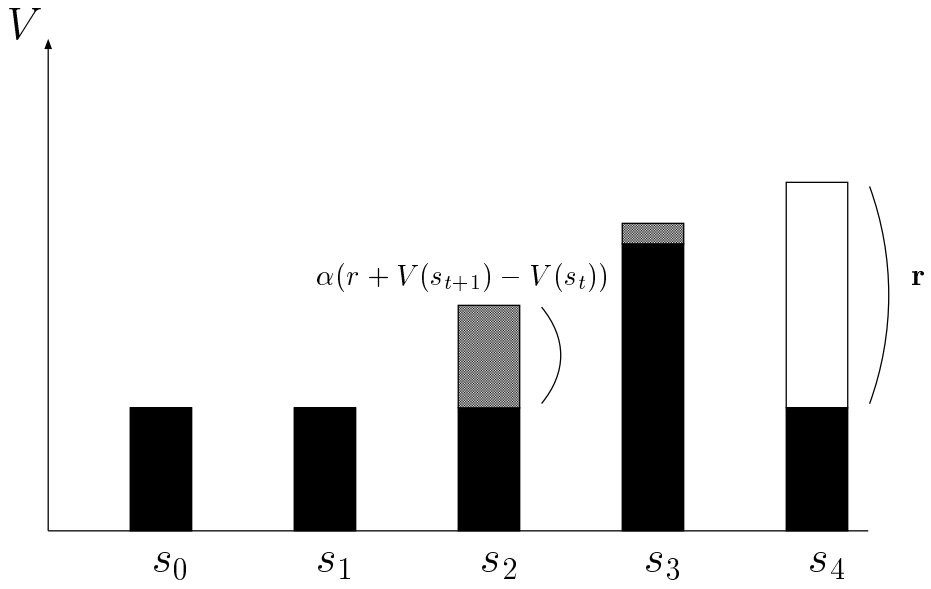


図 2.5: 2 試行後の状態評価値

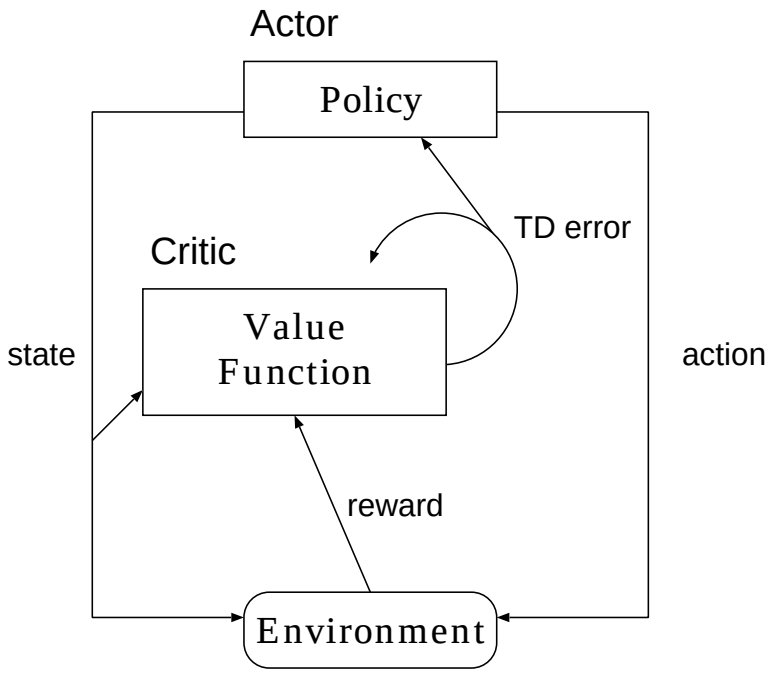


図 2.6: アクター・クリティック法の構造

この問題を解決するために，ニューラルネットワークを用いた連続系の強化学習法を導入する．これにより，離散系強化学習で用いるテーブルのエントリー数と比較して圧倒的に少ない数の基底ユニットで状態価値関数を近似し，表現することができる．また，1つの基底ユニットが，その付近のすべて状態に影響するため，汎化の効果が得られ，学習効率の向上も期待できる．

2.3.1 TD学習

ここでは連続系に拡張したTD学習の説明をする．

連続時間では，ステップごとに時間を区切ることはせず，連続時間・状態系のダイナミクスを

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), \mathbf{a}(t)) \tag{2.8}$$

のように表す．ただし，$\mathbf{x}(t)$，$\mathbf{a}(t)$ は時刻 t の状態，行動出力である．報酬は離散系の場合と同様に，状態を引数とする関数で表される．

$$r(t) = r(\mathbf{x}(t)) \tag{2.9}$$

行動出力 $\mathbf{a}(t)$ が方策関数 $\pi(\mathbf{x}(t))$ で決定されるとすると，連続系での状態価値関数は

$$V^{\pi}(\mathbf{x}(t)) = \int_{t}^{\infty} e^{-\frac{s-t}{\tau}} r(\mathbf{x}(s))ds \tag{2.10}$$

と表される．ここで τ は評価の時定数である．ここで，割引率 γ を $\gamma = 1 - \frac{\Delta t}{\tau}$ と考えると離散系のTD学習と対応づけることができる．

式2.10で定義した評価関数を，ニューラルネットワークを用いた関数近似法によって近似することを考える．$\mathbf{w}$ を関数近似のパラメータのベクトルとすると，近似による評価関数は次のようになる．

$$V^{\pi}(\mathbf{s}(t)) \simeq V(\mathbf{x}(t); \mathbf{w}) \tag{2.11}$$

TD学習では状態価値関数の推定値を用いて学習を進める．状態価値関数 $V^{\pi}(\mathbf{x}(t))$ の推定値を $V(\mathbf{x}(t)) = V(\mathbf{x}(t); \mathbf{w})$ とし，式2.10の両辺を時間微分すると

$$\frac{dV^{\pi}(\mathbf{x}(t))}{dt} = \frac{1}{\tau} V^{\pi}(\mathbf{x}(t)) - r(t) \tag{2.12}$$

となることから，推定値が正しければ，$\dot{V}(\mathbf{x}(t)) = \frac{1}{\tau} V(\mathbf{x}(t))$ を満たす．推定値が正しくない場合は次の式で表される誤差を減らすように学習を行う．

$$\delta(t) \equiv r(t) - \frac{1}{\tau} V(\mathbf{x}(t)) + \frac{dV(\mathbf{x}(t))}{dt} \tag{2.13}$$

これが連続系でのTD誤差である．

2.3.2 正規化ガウス関数ネットワークを用いた関数近似

本研究で用いる連続系の強化学習では [2] で用いられた方法に従い，状態評価関数，方策関数をニューラルネットワークを用いて近似する．本研究では正規化ガウス関数ネットワーク (normalized Gaussian network:NGnet) を用いることにした．基底関数にガウス関数を直接使うのではなく，正規化処理を行っているため，基底ユニットが密に配置される場所では局所的となり，分布の端の部分では大域的な基底関数になる．これにより，少ない基底ユニットでも広範囲をカバーでき，効果的な関数近似が可能である．

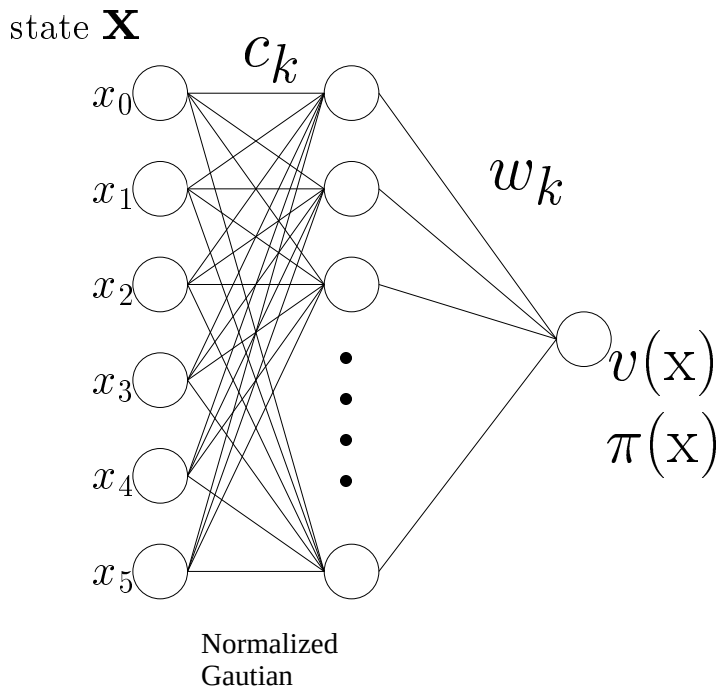


図 2.7: NGnet の構造

NGnet の構造は図 2.7 に示したような，3 層のネットワークで，中間素子は正規化ガウス関数である．

n 次元ベクトル $\mathbf{x} = (x_0, x_1, \cdots, x_n)^T$ を NGnet に入力すると，k 番目のユニットの活性化関数は次のガウス関数で計算される．

$$a_k(\mathbf{x}) = e^{-\frac{1}{2}||M_k(x-c_x)||^2} \tag{2.14}$$

ただし，c_k は活性化関数の中心であり，M_k は活性化関数の経常を決定する行列である．図 2.8 に $a_k(x)$ の様相を示す.

ここで，活性化関数 $a_k(\mathbf{x})$ を各点で総和が 1 になるように正規化したものを，基底関数

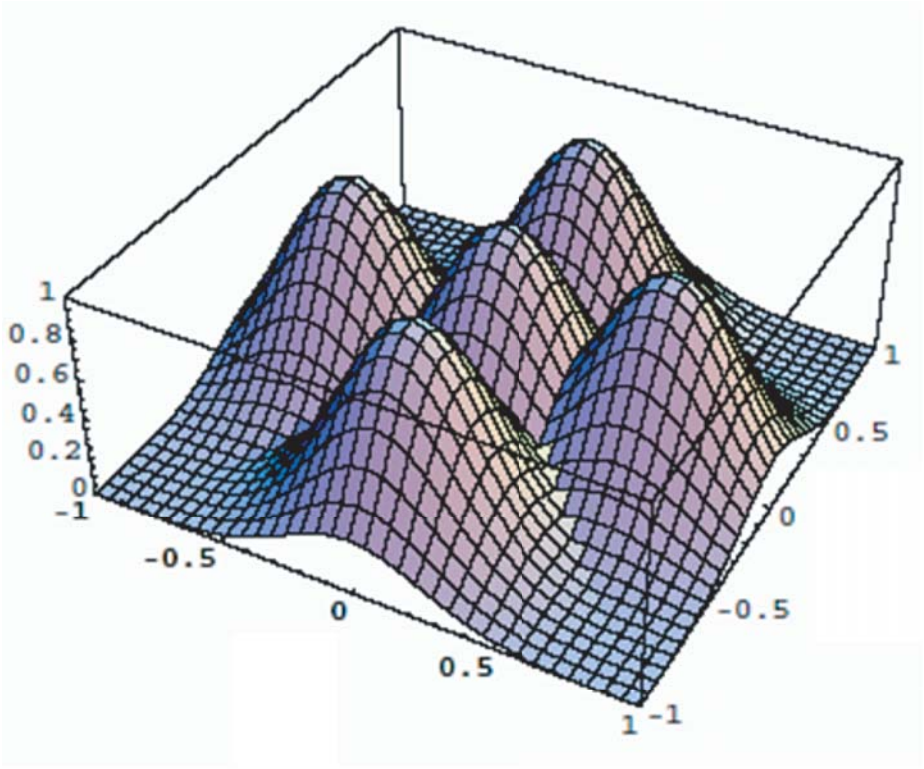


図 2.8: 活性化関数 $a_k(\mathbf{x})$

$b_k(\mathbf{x})$ とする．

$$b_k(\mathbf{x}) = \frac{a_k(\mathbf{x})}{\sum_{l=0}^{K} a_l(\mathbf{x})} \tag{2.15}$$

図 2.9 に示した $b_k(x)$ の様相は, 図 2.8 に対応したものとなっている．NGnet の出力は次の式のように，すべての基底関数と重みの積の総和で表される．

$$y(\mathbf{x}) = \sum_{k=0}^{K} w_k b_k(\mathbf{x}) \tag{2.16}$$

ネットワークの重みの更新は目標出力 $\hat{y}(\mathbf{x})$ が与えられた時,

$$\Delta w_k = \eta(\hat{y}(\mathbf{x}) - y(\mathbf{x}))b_k(\mathbf{x}) \tag{2.17}$$

によってなされる．ただし，η は学習率である．

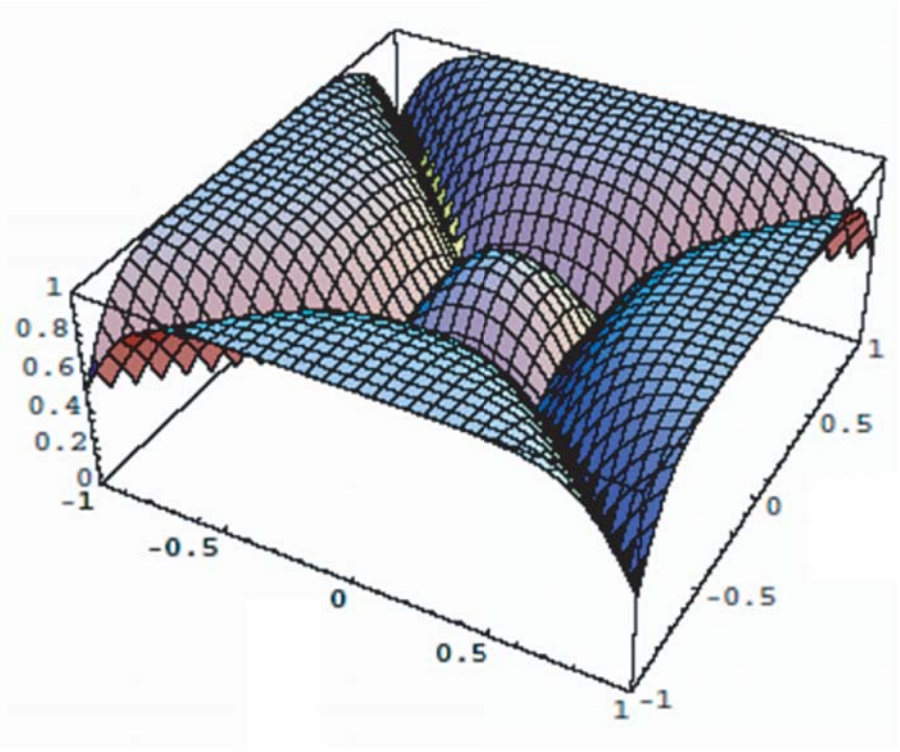


図 2.9: 基底関数 $b_k(\mathbf{x})$

本研究で用いる連続系の強化学習の状態価値関数は，NGnet を用いて

$$V(\mathbf{x}(t)) = \sum_{k=0}^{K} w_k b_k(\mathbf{x}(t)) \tag{2.18}$$

と表す.

2.3.3 適格度トレース

適格度トレース)(eligibility trace)e は状態価値関数を更新する際，報酬を過去の状態へ遡って伝搬させるための重みの役割をする．適格度トレースを用いることで，報酬の直前の状態までのみ遡る TD(0) 法よりも優れた学習効率を得るといわれている．特に時間をステップで区切ることのできない連続系の強化学習では，連続時間に対応した報酬の伝搬法を定式化するためにも適格度トレースの概念は重要となる．

離散系の適格度トレース

離散系の強化学習では各状態に 1 つの適格度トレースが対応する．時刻 t における各状態の適格度トレースは次の式で求められる．

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & (if\, s \neq s_t) \\ \gamma\lambda e_{t-1}(s) + 1 & (if\, s = s_t) \end{cases} \tag{2.19}$$

ここで，$e_t(s)$ は時刻 t での状態 s の適格度トレースを表しており，γ は割引率である．γ は $(0 \leq \gamma \leq 1)$ の値を取るパラメータでトレース減水パラメータ (trace-decay parameter) と呼ばれる．各ステップにおいて，適格度トレースは全ての状態に対して $\gamma\lambda$ だけ，減衰し，そのステップで訪れた状態のみ 1 だけ増加する．この様子を分かりやすく表すために，図 2.10 で，ある適格度トレースの値の推移の例を示す．横軸にある縦線は，その状態へ訪れたことを表している．訪問されると累積され，訪問されないと減衰することから，適格度トレースより，最近訪れた状態，または，頻繁に訪れた状態，つまり強化するべき状態を判断することができる．よって，状態価値関数の更新量は適格度トレースの大きさに比例配分されるべきであり，更新式は次のようになる．

$$\begin{aligned} V(s) &\leftarrow V(s) + \alpha\delta_t e_t(s) \\ &= V(s) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]e_t(s) \end{aligned} \tag{2.20}$$

この式を用いて，適格度トレースが非ゼロである全ての状態の評価値を更新する．式 2.19，式 2.20 より，トレース減衰パラメータの値が大きいほど，過去の状態に遡って，状態評価値を更新することが分かる．特に，$\lambda = 0$ の時，直前の 1 ステップのみ遡ることになり，TD(0) 法と等しくなる．

連続系の適格度トレース

連続系の強化学習では各基底ユニットに 1 つの適格度トレースが対応する．そして，適格度トレースは次の式で表される．

$$\begin{aligned} \dot{e}_i(t) &= -\frac{1}{k}e_i(t) + \frac{\partial V(t)}{\partial w_i} \\ &= -\frac{1}{k}e_i(t) + b_i(\mathbf{x}(t)) \end{aligned} \tag{2.21}$$

次に適格度トレースを用いた状態価値関数の重みの更新式を示す．

$$\begin{aligned} w_i &\leftarrow w_i + \alpha\delta(t)e_i(t) \\ &= w_i + \alpha\left[r_{t+1} - \frac{1}{\tau}V(\mathbf{x}(t)) + \frac{dV(\mathbf{x}(t))}{dt}\right]e_i(t) \end{aligned} \tag{2.22}$$

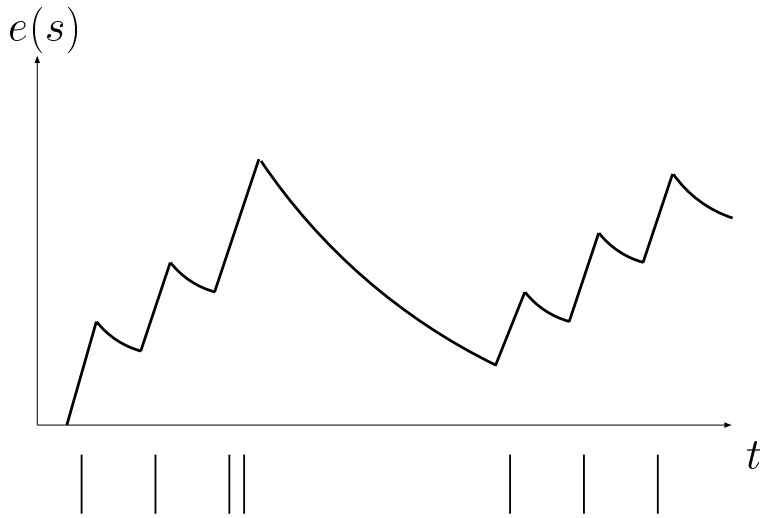


図 2.10: 適格度トレースの推移

2.3.4 アクター・クリティック法

TD 法や，一般によく用いられる Q-learning 法は，行動の離散化が必要であり，滑らかな動作が求められるロボットの学習には不向きである．一方，アクター・クリティック法はアクターが正規分布などに基づいて，行動を決定できるため，無限に存在する行動出力を容易に決定でき，連続系に適していると言える．

クリティックは離散系の強化学習と同様に，TD 誤差を 0 にするよ学習する．クリティックへの入力は状態 $\mathbf{x}(t)$ であり，出力は状態価値関数 $V(\mathbf{x}(t))$ である．状態価値関数の更新には適格度トレースを用いており，状態価値関数の重みの更新は 2.22 で，適格度トレースの更新は式 2.21 でされる．

また，アクターへの入力も状態 $\mathbf{x}(t)$ であり，出力は行動出力のベクトル $\mathbf{a}(\mathbf{x}(t))$ である．行動出力の j 番目の要素 $a_j(\mathbf{x}(t))$ は次のように求める．

$$a_j(\mathbf{x}(t)) = a_j^{\max} g(\sum_i w_{ij} b_i(\mathbf{x}(t)) + \sigma n_j(t)) + a_{bj} \tag{2.23}$$

ただし，w_{ij} は j 番目の行動出力に対する i 番目の基底ユニットの重みを表す．また，$a_j^{\max}$ は j 番目の行動出力の最大値であり，a_{bj} はバイアス出力で，行動出力の中央値が対応する．関数 $g()$ は行動出力を最大出力で飽和させる役割をし，シグモイド関数を用いる．$\sigma n_j(t)$ は，ノイズであり，行動出力の探索のため用意した．

方策関数の重みの更新式を示す.

$$\dot{w}_{ij} = \beta\delta(t)\sigma n_j(t)b_i(\mathbf{x}(t)) \tag{2.24}$$

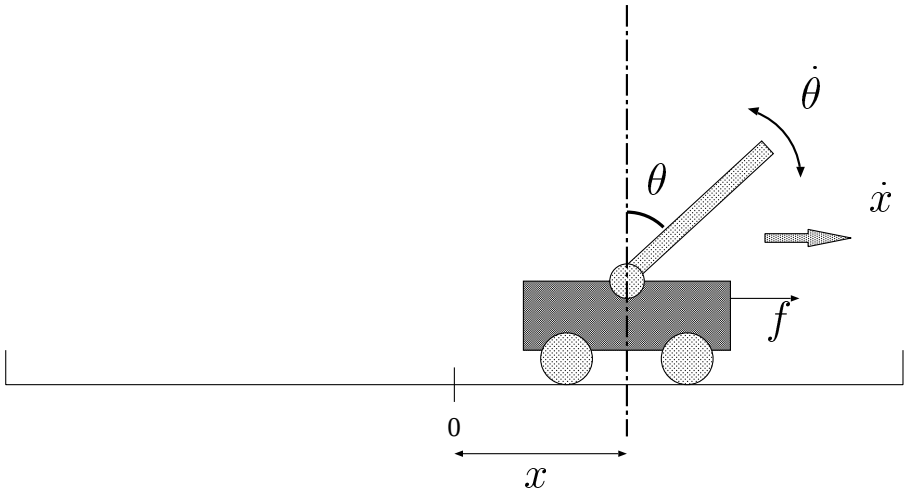
ただし，β は学習率である.

第3章　基底集中化の影響に関する予備実験：ポールバランシング問題

本章で行う予備実験の目的は, 提案手法の検討のために必要な知見を得ることである. 具体的には, 基底ユニットの数と学習精度との関連性, 細やかな制御が必要なクリティカル状態に基底を集中配置したときの影響の検証である.

3.1　ポールバランシング問題

ポールバランシング問題とは図 3.1 のようなポールが付けられた台車をエージェントとし，左右に動くことでポールを振り上げ，バランスをとり垂直状態を保つ動作を獲得させるタスクである．ポールバランシング問題は多くの機械学習の研究で取り上げられており，強化学習のベンチマーク的なタスクと言える．また, 比較的低次元であり, 制御を細やかにすべき位置がポールが立っている状態であることが明らかなため, そこに基底を集中配置したときの影響を図るには適したタスクであるといえる.



x:台車の変位 [m].　$\dot{x}$:台車の速度 [m/s].　θ:ポールの角度 [rad].
$\dot{\theta}$:ポールの角速度 [rad/s].　f:台車への力 [N]

図 3.1: ポールバランシング

ポールバランシング問題の 1 回の試行は

- 壁に衝突する (台車の変位が制限をこえる)
- ポールの角速度が一定値を越える
- 制限時間に達する

の条件のいずれかを満たすとき終了する．試行が終了すると，状態，行動出力，適格度トレースを初期化し，再び新しい試行を始める．

3.2 設定

3.2.1 シミュレーション設定

まず，初めにシミュレーションに用いられるパラメータの設定を表 3.1 に示す．重力加速度，台車・振り子の質量，振り子の長さ，静止・動摩擦係数，時間間隔，制限時間は定数である．

表 3.1: シミュレーションに用いられるパラメータの設定

パラメータ	記号	単位	初期値	範囲
台車の変位	x	m	0	$-3.5 \sim 3.5$
台車の速度	$\dot{x}$	m/s	0	$-8 \sim 8$
台車の加速度	$\ddot{x}$	$\mathrm{m/s^2}$	0	
振り子の角度	θ	rad	3	$-\pi \sim \pi$
振り子の角速度	$\dot{\theta}$	rad/s	0	$-10 \sim 10$
振り子の角加速度	$\ddot{\theta}$	rad	0	
重力の加速度	g	$\mathrm{m/s^2}$	9.8	
台車の質量	M	kg	1	
振り子の質量	m	kg	0.1	
振り子の長さ	l	m	0.5	
台車を押す力	f	N	0	$-15 \sim 15$
静止摩擦係数	μ_c		0.05	
動摩擦係数	μ_p		0.01	
時間感覚	Δt	s	0.02	
制限時間	$T_{\max}$	s	10	

速度，台車・振り子の質量，振り子の長さ，静止・動摩擦係数，時間間隔，制限時間は定数である．

運動方程式

ロボットのダイナミクスは次の運動方程式に従って計算される．

$$\ddot{\theta} = \frac{(M+m)g\sin\theta - (f + ml\dot{\theta}^2\sin\theta)\cos\theta}{\{\frac{4}{3}(M+m) - m\cos^2\theta\}} \tag{3.1}$$

$$\ddot{x} = \frac{\frac{4}{3}(f + ml\dot{\theta}^2\sin\theta) - mg\sin\theta\cos\theta}{\frac{4}{3}(M+m) - m\cos^2\theta} \tag{3.2}$$

学習では連続系の学習アルゴリズムを用いるが，シミュレーションは計算機上で扱うため，実際には時間感覚 Δt で離散化した．付録 A.1 参照．よって，Δt 後の角速度 $\dot{\theta}$，速度 $\dot{x}$，角度 θ，変位 x は式 3.1，式 3.2 を用いて

$$\begin{cases} \dot{\theta} &= \ddot{\theta}\cdot\Delta t \\ \dot{x} &= \ddot{x}\cdot\Delta t \\ \theta &= \dot{\theta}\cdot\Delta t = \frac{1}{2}\ddot{\theta}(\Delta t)^2 \\ x &= \dot{x}\cdot\Delta t + \frac{1}{2}\ddot{x}(\Delta t)^2 \end{cases} \tag{3.3}$$

と計算される．

それぞれの試行は，初期状態 $x(0) = (0, 0, 3, 0)$ から開始する．ポールを真下ではなく，やや傾斜をつけて開始されるのは，振り上げの方向を一定にさせるためである．

試行の終了条件は $x < -3.5$ or $3.5 < x$，$\dot{\theta} < -10$ or $10 < \dot{\theta}$，$t \leq T_{\max}$ のいずれかである．

3.2.2 学習設定

強化学習は連続系のアクター・クリティック法を用いる．これは，2 章で述べたように，ポールバランシングは滑らかな動作が求められており，アクター・クリティックは連続系への応用が用意であるためである．ただし，シミュレーションは離散時間によって行われるので，実際に用いた学習アルゴリズムは 2 章で述べた連続系のアクター・クリティック法とは若干異なっている．実際に用いた学習手順を付録 A.1 に記す．

次に，学習に用いられたパラメータの設定は表 3.2 の通りである．α は状態価値関数，β は方策関数の更新に用いられる学習率である．

速度などの情報が必要になる．また，台車の可動範囲も限定するため，台車の変位情報も必要となる．よって，時間 t での状態 $\mathbf{x(t)}$) は $\mathbf{x(t)} = (x, \dot{x}, \theta, \dot{\theta})$ の 4 次元とした．

エージェントの行動は台車にかける力である．行動は状態を引数とする関数として $\pi(x(t)) = f$ として表す．また行動は 1 次元である．

報酬関数 $R(x(t))$ は

$$R(x(t)) = r(x(t)) + \tilde{r}(x(t)) \tag{3.4}$$

表 3.2: 学習に用いられるパラメータの設定

パラメータ	記号	値
学習率 V	α	5
学習率 π	β	10
学習率 $\tilde{r}$	α_r	5
時定数	τ	1
時定数	κ	0.5
ノイズの標準偏差	σ	0.5
ノイズの拘束期間	$\Delta t_{\mathrm{noize}}$	0.4
$\tilde{r}$ の更新間隔	Δt_{ad}	0.04

$$r(x(t)) = \frac{\cos\theta - 1}{2} \tag{3.5}$$

$$\tilde{r}(x(t)) = \sum_i \tilde{w}_i b_i(x(t)) \tag{3.6}$$

とした．式 3.5 はポールの先端の高さに比例して報酬を与える．また，台車が可動範囲をこえた場合，あるいは，ポールの角速度が制限を越えた場合，タスクの失敗とみなし，$r(x) = -1$ の罰を試行終了時間ばで与え続ける．

状態に対して報酬を設定するため，状態価値関数 $V(x(t))$ と報酬関数 $\tilde{r}(x(t))$ は同一の NGnet を使って表すものとした．また，方策関数 $\pi(x(t))$ は行動出力の次元数分の NGnet が必要となるが，ポールバランシング問題では行動出力が 1 次元であるので，状態価値関数の NGnet と同じとした．

3.2.3 実験結果

ポールバランシング問題の学習を従来の強化学習法で行い，その結果をグラフにしたものを図 3.2-図 3.4 に示す．3 つのグラフは，それぞれ配置する基底ユニットの数が異なったもので，グラフの横軸は試行回数，縦軸はポールの倒立時間を示している．また, 500 回ごとに平均化した結果を折れ線グラフで示している. これらを比較すると，やはり配置された基底ユニットが少なくなるにしたがい学習が安定しなくなり，各試行毎のポールの倒立時間にばらつきが出ることがわかる.

また, ポールの倒立時間が長い試行と, 短い試行について, 挙動を比較したグラフを図 3.5, 図 3.6 に示す. それぞれ, グラフの横軸は経過時間, 縦軸はポールの角度 θ, または角速度 $\dot{\theta}$ である. 長い試行については経過時間 2.3s あたりから $\theta = 0, \dot{\theta} = 0$ の値をとり, 試行終了の 20s まで同じ値をとり続ける. つまりポールが立っている状態に落ち着いている.

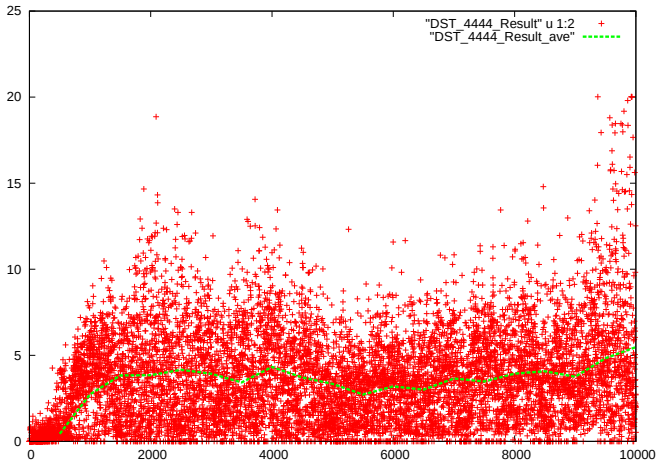


図 3.2: 従来手法でのポールバランシング（ユニット数：4, 4, 4, 4）

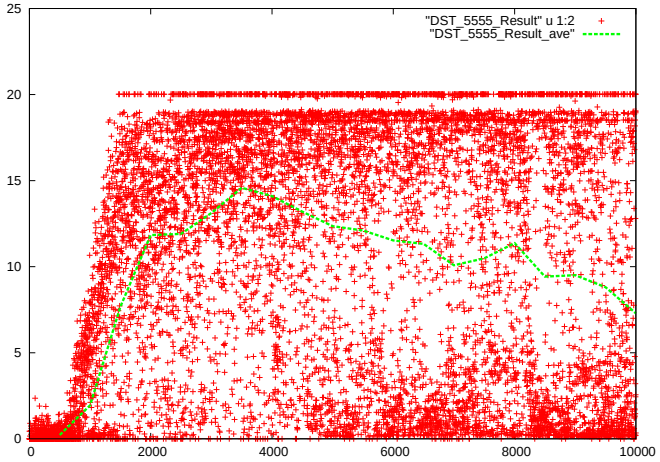


図 3.3: 従来手法でのポールバランシング（ユニット数：5, 5, 5, 5）

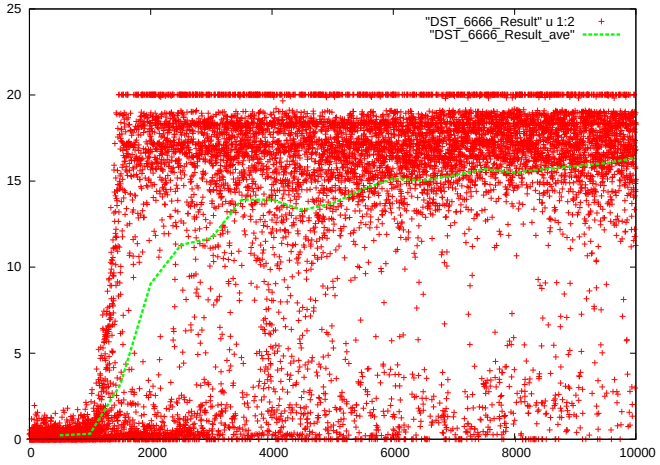


図 3.4: 従来手法でのポールバランシング（ユニット数：6, 6, 6, 6）

い

一方倒立時間が短い試行については, 経過時間 2.3s あたりから $\theta = 0, \dot{\theta} = 0$ の値をとっているが, 6.5s あたりから値が急激に変化する. つまりポールが倒れてしまっている. これは, 短い試行については, $\theta = 0, \dot{\theta} = 0$ 付近の基底ユニットが不足し, 状態に対する制御が粗くなってしまっていることが原因と推測する.

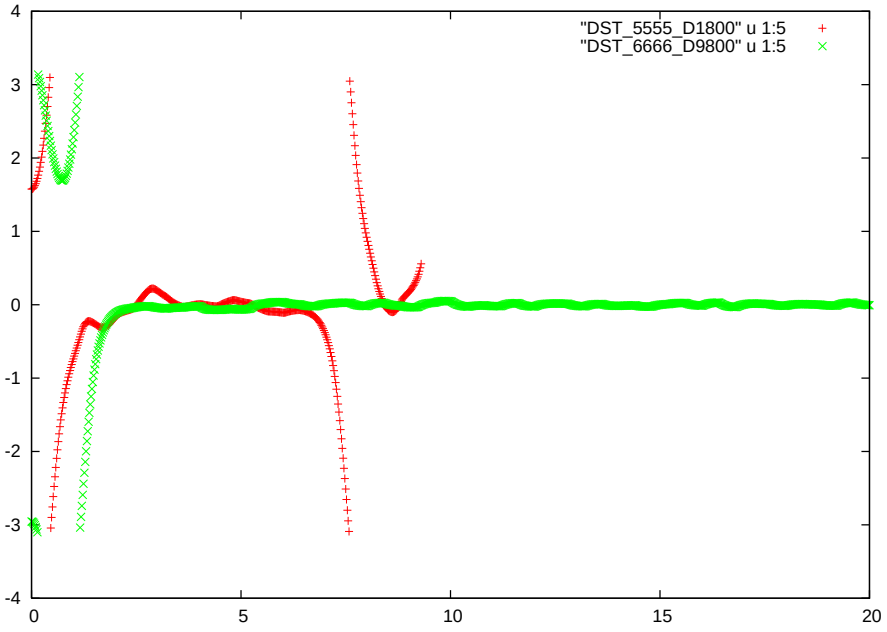


図 3.5: θ に関する挙動

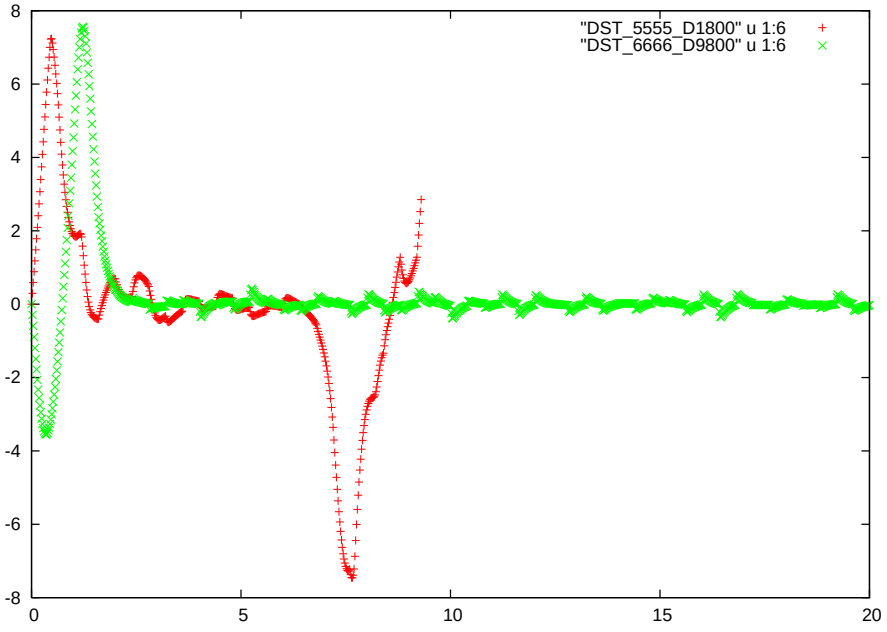


図 3.6: $\dot{\theta}$ に関する挙動

$\theta = 0, \dot{\theta} = 0$ のように挙動が安定するか不安定になるかが, 著しく分れるクリティカルな状態に対しては, 近似精度を向上させる必要があると考えた. そこで, $\theta = 0, \dot{\theta} = 0$ を中心に基底ユニットの配置間隔を従来の配置より 20%程度狭くして, 基底の集中化を試み, 学習への影響を観察する. 結果を図 3.7 に示す.

従来の配置である図 3.3 と比較すると, 集中化した影響で, 学習が安定していることがわかる. これは $\theta = 0, \dot{\theta} = 0$ でポールが立った状態での近似精度が向上し, 細やかな制御が可能になったことが原因と推測する.

以上の予備実験により, 基底ユニットの数に比例して学習精度が向上すること, 細やかな制御が必要であるクリティカルな状態に基底を集中化すると学習精度が向上しうることが示された. ただし, クリティカル状態の検出が目測であるために, クリティカル状態が単にエージェントがよく訪れる状態と同一である可能性がある.

次章では, 制御が不安定になるきっかけであるクリティカルな状態を自動的に検出し, 基底の集中化を施す手法を提案する.

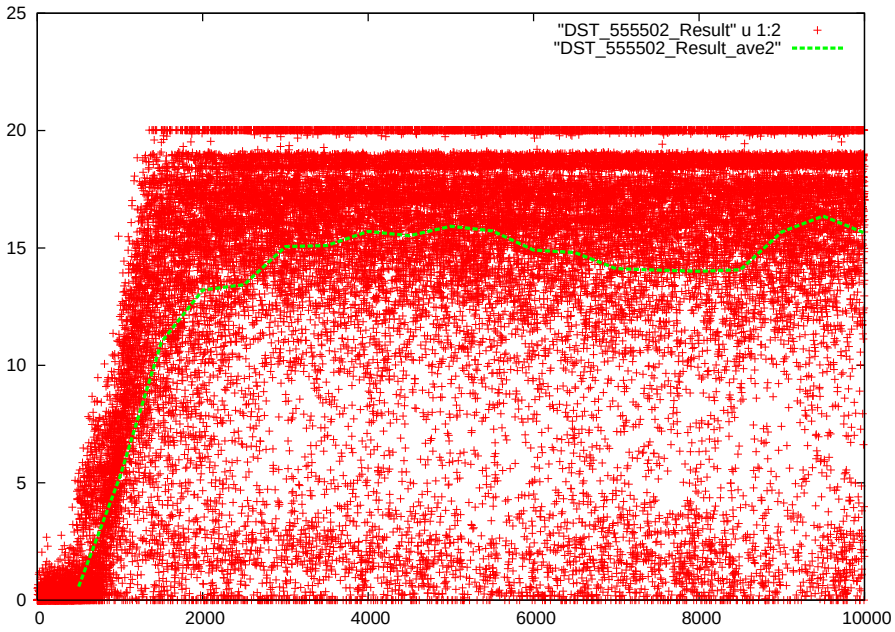


図 3.7: 基底ユニットを $\theta = 0, \dot{\theta} = 0$ に集中化したポールバランシング（ユニット数：5, 5, 5, 5）

第4章　提案手法

本章では, 基底ユニットの配置を, クリティカルな状態に集中化する手法を提案する. そうすることで，無駄に配置された基底ユニットが少なくなり，基底ユニットを効率的に活用でき, 適応的な関数近似が可能になるので，学習精度の向上が期待できる.

以降, はじめに提案手法の概要を説明し, 提案手法の全体像を示す. 次に提案手法の流れを説明した後, 最後に, 詳細において具体的な処理方法について説明する.

4.1　概要

本論文では, NGnet の基底ユニットの配置を, クリティカルな状態に集中化する手法を提案する.

クリティカル状態とは, 動作が不安定になり, エージェントにとって避けるべき終端状態に陥るきっかけとなる状態である. また, このような状態は試行によって終端状態に至るか安定するかが著しく分れている. 例えば, 二足歩行問題において, 成功試行と失敗試行があったとして, 途中まで似た動きを取っているが, ある状態をターニングポイントとして不安定になり転倒に至るか, 安定して歩き続けるかが分れる. 例を図 4.1 に示す.

このような状態の付近ではエージェントの行動に対する細やかなコントロールが要求され, より高精度な価値関数の近似を実現することが必要である.

提案手法では, エージェントの試行をサンプルとして複数保存し, その中で, 最も良い試行との比較によりクリティカル状態を自動的に検出し, 基底の集中化を施す. そうすることで, 学習タスクに応じた最適な近似が可能になると考える. 具体的には, 高精度な近似が必要な部分には基底ユニットを密に, そうでない個所には粗に置き, 少ない基底ユニットで, 学習の安定化をはかる.

4.2　提案手法の流れ

システムでは, 通常の強化学習をある指定した回数 $\Delta_{recovery}$ 回試行が終了するごとに, 提案手法を適用する. 具体的にはクリティカル状態の検出と, 基底の集中化である.

クリティカル状態はエージェントの試行サンプルを比較することにより検出するが, 保存すべき試行は 2 種類ある. 一つはお手本サンプルで, ある一定基準, 例えば報酬総量などにおいて, ベストを更新した時点で保存する. もう一つは, お手本サンプルとの比較用

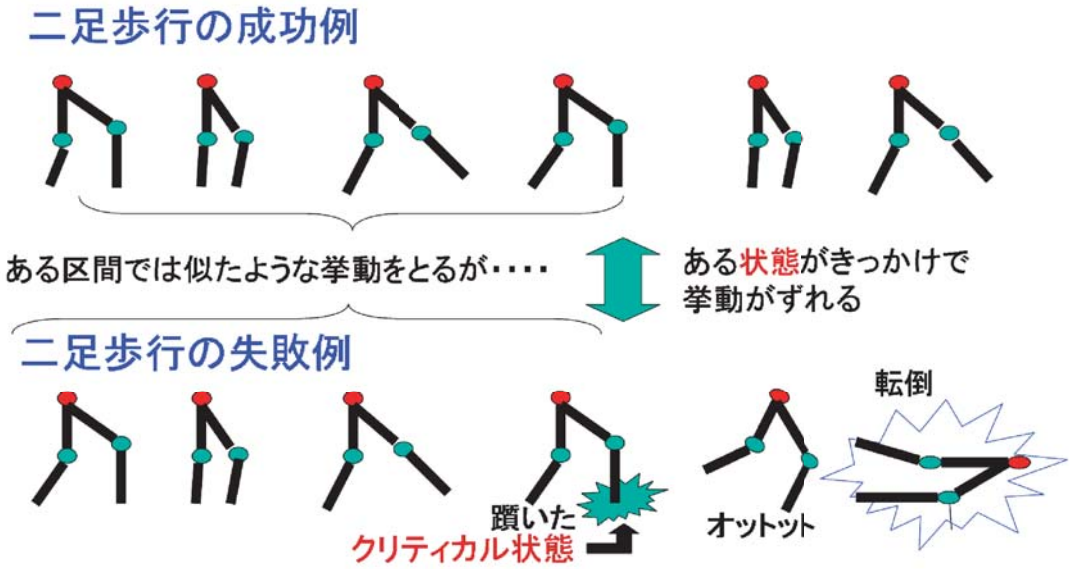


図 4.1: クリティカル状態

サンプルで一定回数ごとに複数の試行を保存する. 提案手法を組み込んだ, システムのフローチャートを図 4.2 に示す.

提案手法の流れを図 4.3 に示す. 提案手法では, 比較用サンプルの終端状態付近の挙動を, お手本サンプルにおいて, 部分的によく似た挙動とマッチングさせた後, 急激に挙動がずれはじめた点をクリティカル状態として抽出する. その後, クリティカル状態を利用して基底の集中化を施す.

STEP1, STEP2, STEP3 はクリティカル状態の検出に対応し, STEP4 は基底の集中化に対応する. 各 STEP について, 以下に概説する.

STEP1: 比較元データ生成

STEP1 では比較用サンプルから, 終端状態における一定時間範囲の部分的試行を抽出する. その時間範囲を ld とする. 本研究では, クリティカル状態を失敗状態に至るきっかけとしており, それは比較用サンプルの失敗状態付近に存在するとの前提から, STEP1 の処理を必要とした.

STEP2: 比較先データ生成

STEP2 ではお手本サンプルから比較元データに, 一定時間範囲において類似する部分的試行を比較先データとして抽出する. その範囲を lm とする. ただし, $(lm \leq ld)$ である.

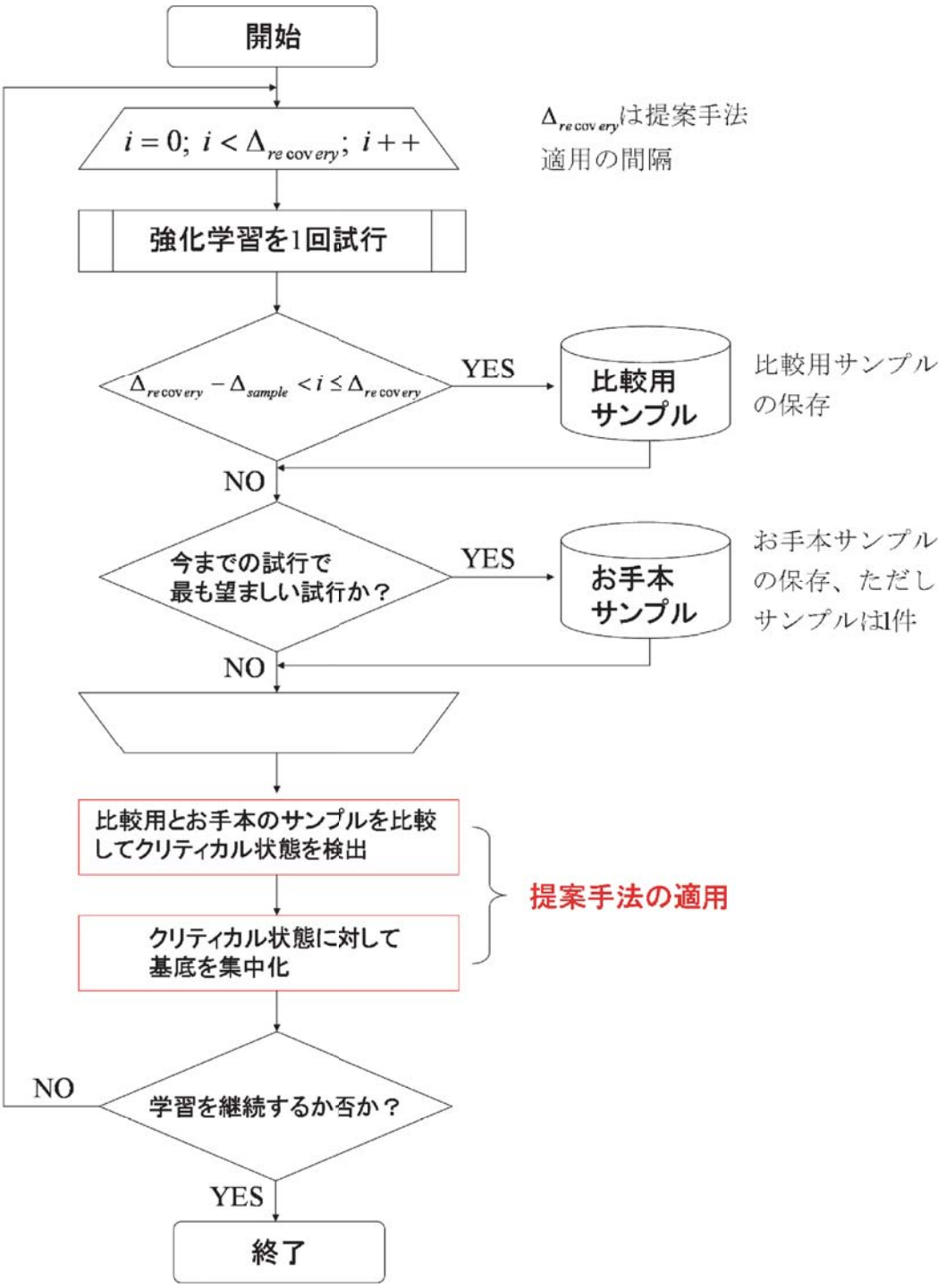


図 4.2: 強化学習システムにおける提案手法のフローチャート

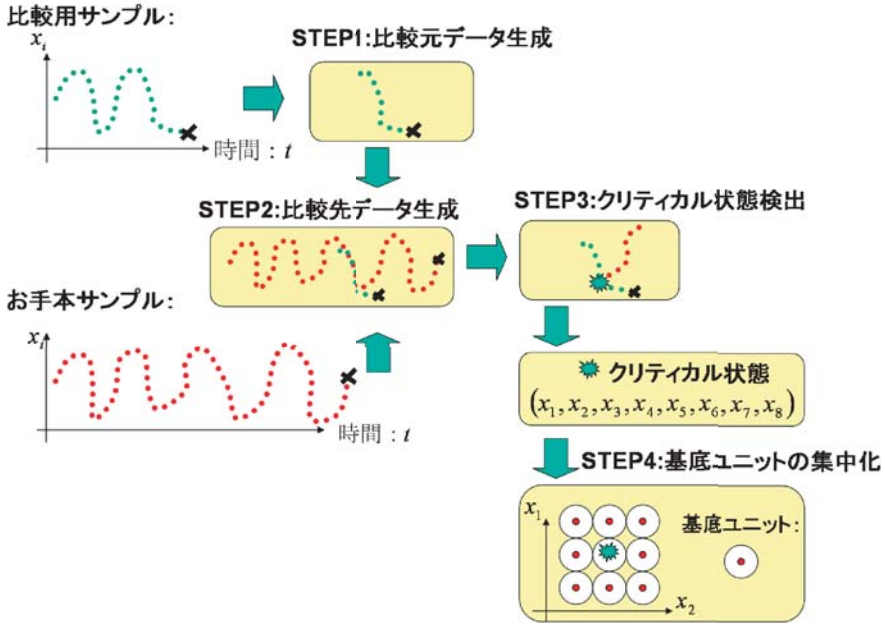


図 4.3: 提案手法の流れ

提案手法では, 比較用サンプル, 比較元サンプルともに, クリティカル状態に至るまで 2 つのサンプルが途中まで類似した動きを取っていることを前提としている. そのため, 比較元データと比較先データは一定時間範囲 lm において類似している必要があり, STEP2 の処理を必要とした.

STEP3: クリティカル状態検出

STEP3 では比較元データと比較先データが急激にずれ始めている点をクリティカル状態として抽出する. ずれ始めている点とは, ある一定時間範囲内 lp において 2 データの誤差の変化量が急激に変化している点である. 本研究において, クリティカル状態は, 2 データが途中まで類似した動きをとっていることに加えて, 急激にずれ始めている点が存在する. また, そのずれは局所的なものではなく, 失敗状態に至るまでずれていることが必要である. そのため, 比較元データと比較先データにおける一定範囲での変化量を考慮する必要があり, STEP3 の処理を必要とした.

STEP4: 基底ユニットの集中化

STEP4 ではクリティカル状態を利用して基底ユニットの集中化を施す. ただし, クリティカル状態に対してのみ, 直接的に基底を配置するのではなく, クリティカル状態から

一定範囲前の, 比較元データの挙動に対しても基底ユニットを複数配置し集中化を施す. これは, クリティカル状態より以前から制御を細やかにすることで, 失敗状態を回避しようとする試みである. それゆえに STEP4 の処理を必要とした.

STEP1, STEP2, STEP3, STEP4 の具体的な処理については, 次節で詳述する.

4.3　提案手法の詳細

本節では, クリティカル状態検出と, 基底集中化の 2 つに分けて説明する. STEP1, STEP2, STEP3 はクリティカル状態の検出に対応し, STEP4 は基底の集中化に対応する. 各 STEP について, 以下に詳しく説明する.

4.3.1　クリティカル状態検出

データ形式

まず, 強化学習タスクにおける試行 E を以下に定義する.

$$E = \{X(0), \cdots, X(t), \cdots, X(T)\} \tag{4.1}$$

$$X(t) = (x_{(1,t)}, \cdots, x_{(n,t)}) \tag{4.2}$$

$X(0)$ は試行における開始状態, $X(T)$ はその終端状態である. 状態はそれぞれ n 次元の要素を持つ. 例として, 二足歩行学習におけるエージェントの様相とデータ形式との対応を図 4.4 に示す. 以後, クリティカル状態検出に対応する STEP1, STEP2, STEP3 では, このデータ形式での処理が中心となる.

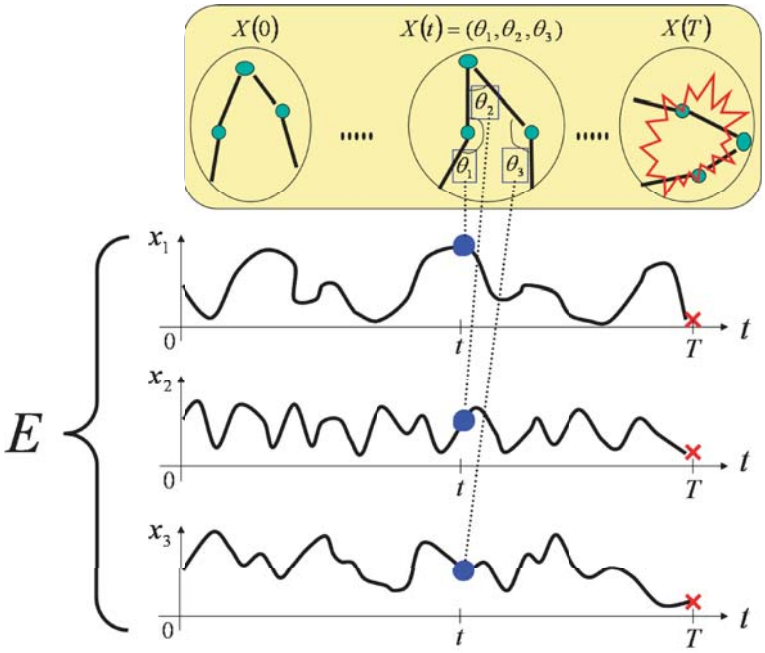


図 4.4: データ形式とエージェントの様相との対応

STEP1: 比較元データの生成

STEP1 では比較用サンプル E から終端状態から直前の部分的試行を, ld を指定することで, 比較元データとして抽出する. 比較元データは E_{smp} とし, 以下に定義する.

$$E_{smp} = \{X_{smp}(T_{smp} - ld), \cdots, X_{smp}(t), \cdots, X_{smp}(T_{smp})\} \tag{4.3}$$

$$X_{smp}(t) = (x^{smp}_{(1,t)}, \cdots, x^{smp}_{(n,t)}) \tag{4.4}$$

具体的には, 終端状態からの区間 $[T_{smp} - ld, T_{smp}]$ を設定することで, E_{smp} を生成する. その様相を図 4.5 に示す.

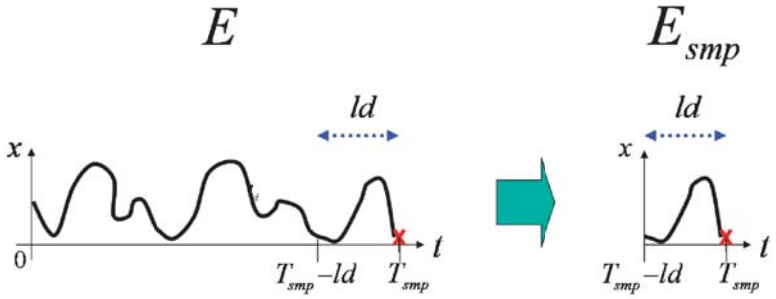


図 4.5: 比較元データの生成

からを, 比較先データとして抽出する.

STEP2: 比較先データの生成

STEP2 では, まずお手本サンプルを E_{bst} とし, 以下に定義する.

$$E_{bst} = \{X_{bst}(0), \cdots, X_{bst}(t), \cdots, X_{bst}(T_{bst})\} \tag{4.5}$$

$$X_{bst}(t) = (x^{bst}_{(1,t)}, \cdots, x^{bst}_{(n,t)}) \tag{4.6}$$

比較先データは学習中のお手本サンプル（例：報酬獲得総量が最大）の部分的試行であり, E_{smp} に, 一定時間範囲 lm において類似する部分的試行である. 比較先データを $E_{sub-bst}$ とし, 以下に定義する.

$$E_{sub-bst} = \{X_{bst}(t_{match}), \cdots, X_{bst}(t_{match} + ld)\} \tag{4.7}$$

t_{match} は, E_{smp} を E_{bst} の $[0, T_{bst} - ld]$ 全てについて, 範囲 ld において比較し, 最も類似した比較の開始点である. なお, 比較の E_{bst} における開始点を t_{start} とする.

図 4.6 にその様相を示すとおり, $lm \leq ld$ である. これは, クリティカル状態は, 2 つの試行は全て類似している必要はなく, 途中まで類似した動きをとっていることを前提にしていることから, $lm \leq ld$ とした. 類似性は, 主に 2 つの試行の誤差を基準に決定する. 次元 i における誤差を $\Delta x_{(i,t)}$ とし, 以下に定義する.

$$\Delta x_{(i,t_{start})} = |x^{smp}_{(i,T_{smp}-ld)} - x^{bst}_{(i,t_{start})}| \tag{4.8}$$

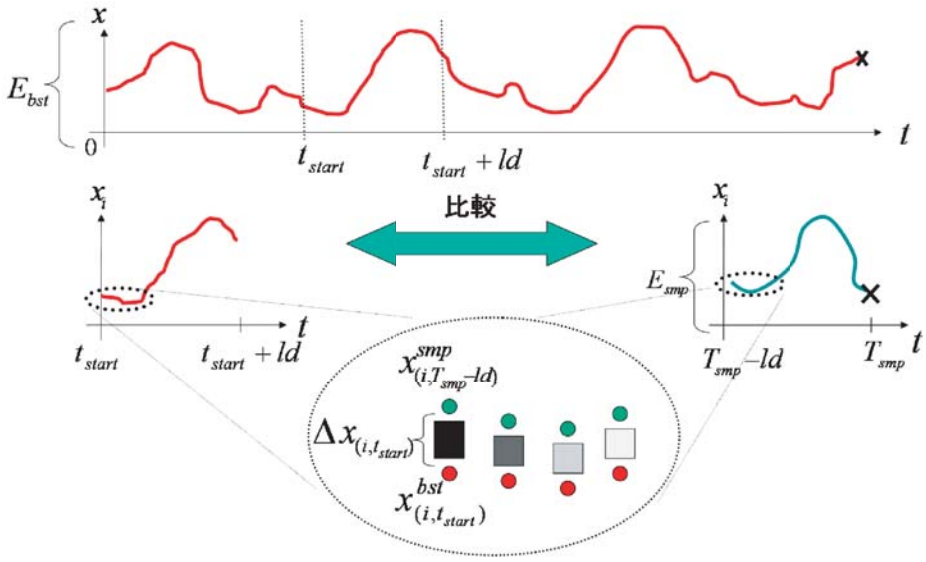


図 4.6: 比較先データ生成のための比較

具体的には, 検出に考慮する次元を設定するためのパラメータベクトル

$$D = (d_1, \cdots, d_n) \tag{4.9}$$

を, 考慮する場合は各 d は 1, 考慮しない場合 0 に値を設定し, すべての t_{start} $(0 \leq t_{start} \leq T_{bst} - ld)$ に次式を適用することで t_{match} を決定する.

$$t_{match} = \min_{t_{start}} \sqrt{\sum_{i=1}^{n} d_i (\sum_{t=t_{start}}^{t_{start}+lm} \alpha \gamma^{(t-t_{start})} \sigma^2(\Delta x_i) \Delta x_{(i,t)})^2} \tag{4.10}$$

ただし, α は減衰パラメータ, γ は減衰率, $\sigma^2(\Delta x_i)$ は次元 i における $\Delta x_{(i,j)}$ の分散値である.

(4.10) 式について説明する. E_{bst} から, E_{smp} の区間 lm における (4.10) 式右辺の値がが最小になる, 部分的試行の始点 t_{start} を抽出する. 類似性は, 4.7 に示すとおり, E_{smp} と E_{bst} の範囲 lm における誤差 $\Delta x_{(i,t)}$ の総和を基準に計算する.

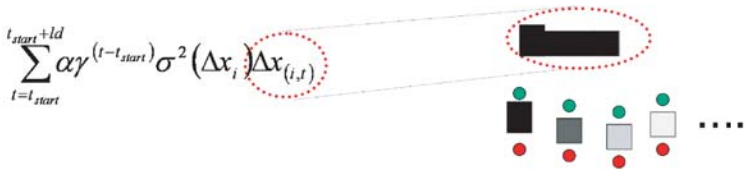


図 4.7: 類似性に対する考慮基準 1-誤差 $\Delta x_{(i,t)}$ の総和

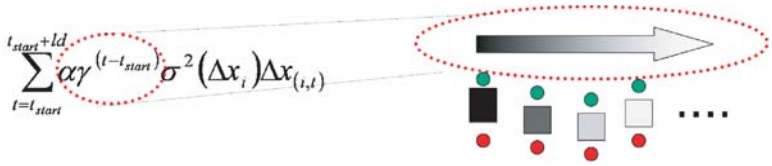


図 4.8: 類似性に対する考慮基準 2-減衰パラメータ α と減衰率 γ

α, γ は, 図 4.8 に示すとおり, 比較に際して, 最初の方の誤差ほど重視するために用いる. これは, クリティカル状態は途中まで 2 つの試行が類似していることが重要で, 最初からずれて, 途中から一致する部分的試行を抽出しないことを意図している.

$\sigma^2(\Delta x_i)$ は, 図 4.9 に示すとおり誤差 $\Delta x_{(i,t)}$ が一定であれば, E_{smp} と類似であるみなすために用いる. これは, 2 つの試行について絶対的な値の差があっても, 相対的に変化量がおなじであれば, 似たような挙動であると見なし, 抽出することを意図している.

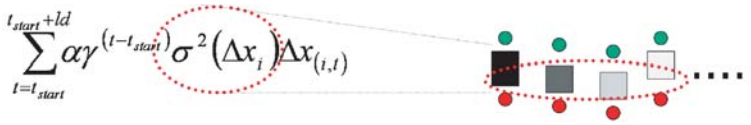


図 4.9: 類似性に対する考慮基準 3-誤差の分散

STEP3: クリティカル状態検出処理

STEP3 では比較元データと比較先データの距離が, ある一定区間において急激にずれ始めている点 $x^{smp}_{(i,t)}$ をクリティカル状態として抽出する. クリティカル状態 $X_{critical}$ は以下に定義する.

$$X_{critical} = (x^{smp}_{(1,t_{critical})}, \cdots, x^{smp}_{(n,t_{critical})}) \tag{4.11}$$

$t_{critical}$ の検出に考慮する区間を lp $(0 < lp < ld)$ とし, すべての t_{start} $(t_{match} \leq t_{start} \leq t_{match} + ld - lp)$, すべての t_{sub} $(t_{start} \leq t_{sub} \leq t_{start} + lp)$ に次式を適用することで $t_{critical}$

を決定する.

$$t_{critical} = T_{smp} - ld + \max_{t_{start}} \sum_{i=1}^{n} d_i \tan^{-1}\left(\frac{|\Delta x_{(i,t_{sub}+1)} - \Delta x_{(i,t_{sub})}|}{|\Delta x_{(i,t_{start}+1)} - \Delta x_{(i,t_{start})}|}\right) - t_{match} \quad (4.12)$$

(4.12) 式での様相を図 4.10 に示す. 図は左から, 2 つのデータの比較, 変化量, 変化量の比を表している. (4.12) 式では範囲 lp における誤差 $\Delta x_{(i,t_{start})}$ と $\Delta x_{(i,t_{sub})}$ の 1 離散時間 STEP における変化量の比を基準に $t_{critical}$ を決定する. $|\Delta x_{(i,t_{start}+1)} - \Delta x_{(i,t_{start})}|$ に対して, $|\Delta x_{(i,t_{sub}+1)} - \Delta x_{(i,t_{sub})}|$ の比が範囲 lp 内で大きければ当然角度も大きくなり, 範囲 lp 内において激しく挙動がずれているとして, クリティカル状態と見なす.

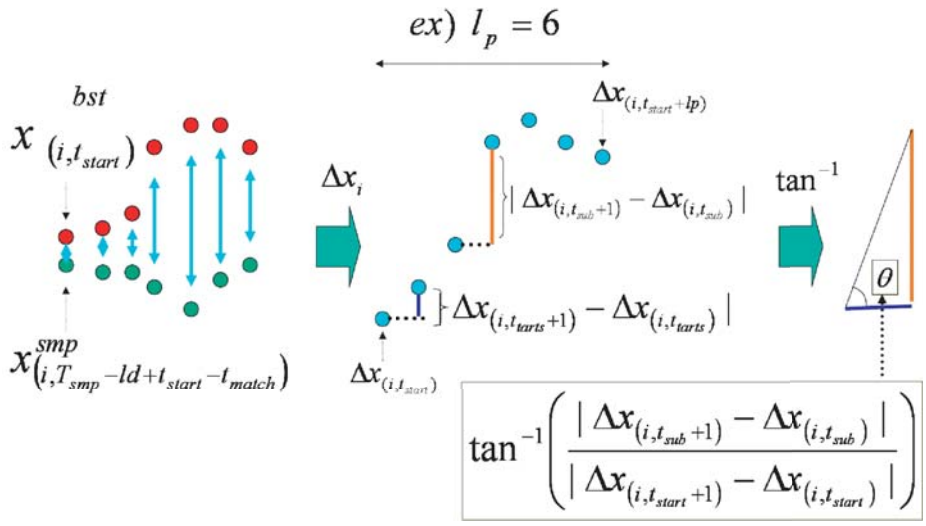


図 4.10: (4.12) 式の様相

4.3.2 STEP4:基底ユニットの集中化

配置単位

提案手法では基底ユニットをボックス単位で, 一括配置する. そうすることでクリティカル状態, 周辺における近似精度を高める. ボックスは, 以下のように構成する. 概念図を図 4.11 に示す.

$$BOX = < Ct, Fl, Bs > \quad (4.13)$$

Ct はボックスの中心座標であり, 以下のように定義する.

$$Ct = (ct_1, \cdots, ct_n) \quad (4.14)$$

Fl はボックスの構成領域の範囲であり, 以下のように定義する.

$$Fl = (fl_1, \cdots, fl_n) \tag{4.15}$$

Bs はボックスの各次元における, 基底の配置個数であり, 以下のように定義する.

$$Bs = (bs_1, \cdots, bs_n) \tag{4.16}$$

ボックスの基底は各次元, 構成領域 $[ct_i - fl_i, ct_i + fl_i]$ の範囲で格子状に配置する.

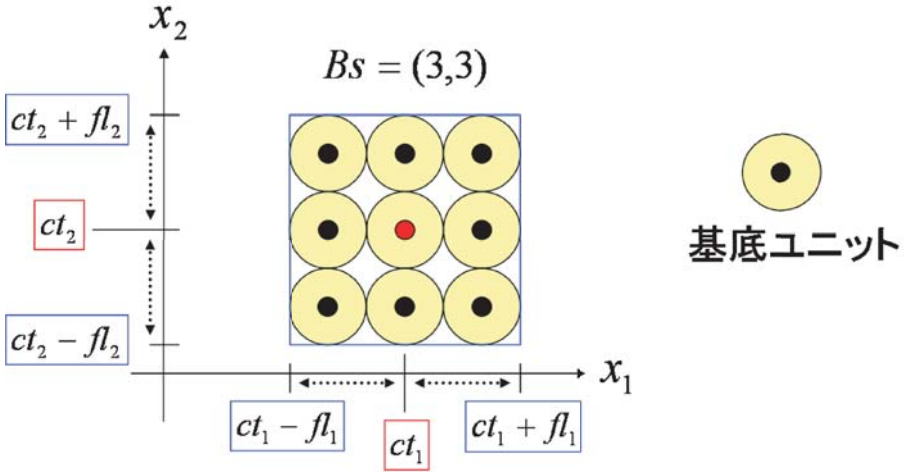


図 4.11: ボックスの概念図

STEP4: 基底ユニットの集中化

提案手法では, 以下 2 つの条件式を満たした場合のみ, ボックスを配置することで基底ユニットの集中化を行う. μ_{match} は比較先データと比較元データの類似性に関する基準値であり, $\mu_{critical}$ はクリティカル状態のずれ度合いに関する基準値である. それぞれ, STEP2 の (4.10) 式, STEP3 の (4.12) 式を条件式として利用する.

$$\sqrt{\sum_{i=1}^{n} d_i \Big(\sum_{t=t_{match}}^{t_{match}+lm} \alpha \gamma^{(t-t_{match})} \sigma^2(\Delta x_i) \Delta x_{(i,t)} \Big)^2} \quad < \quad \mu_{match} \tag{4.17}$$

$$\sum_{i=1}^{n} d_i \tan^{-1}\Big(\frac{|\Delta x_{(i,t_{sub}+1)} - \Delta x_{(i,t_{sub})}|}{|\Delta x_{(i,t_{critical}+1)} - \Delta x_{(i,t_{critical})}|} \Big) \quad > \quad \mu_{critical} \tag{4.18}$$

ボックスは, 条件式 (4.17), (4.18) 式を満たすクリティカル状態を発見するたびに配置する. ただし, 総配置数が一定数 b_{max} に達したならば図 4.12 に示すとおり, 発火量がすくな

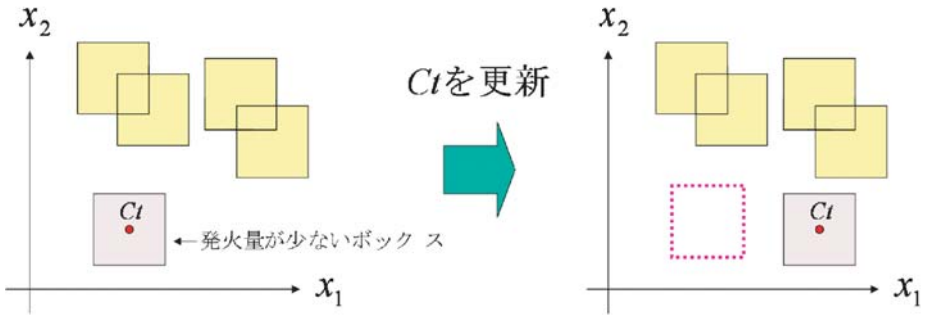


図 4.12: ボックスの移動

いボックスの中心を書き換える形で, ボックスを移動させる. そうすることで, 数限りある基底を有効に利用することができる.

なお, ボックスの中心はクリティカル状態ではない. 図 4.13 に示すとおりクリティカル状態から範囲 lp の部分的な試行のうち, 最小値と最大値の中点を, ボックスの中心としてとる. 具体的には, 以下の条件式を満たすボックスにおける Ct の全要素 $ct_1, \cdots, ct_n$ を次式で更新することで, ボックスの配置, または移動をを行う.

$$ct_i = \frac{max(x^{smp}_{(i,t_{critical}-lp)}, \cdots, x^{smp}_{(i,t_{critical})}) + min(x^{smp}_{(i,t_{critical}-lp)}, \cdots, x^{smp}_{(i,t_{critical})})}{2} \tag{4.19}$$

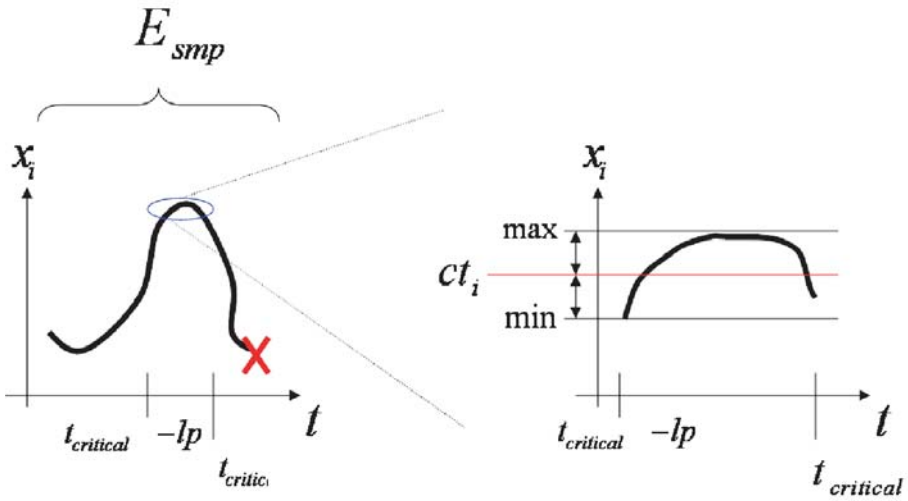


図 4.13: ct_i の決定

第5章　本実験：二足歩行問題

5.1　二足歩行問題

提案手法のタスクとして二足歩行問題を取り上げる. 図 5.1 に示すような, 四つのリンクからなる二足ロボットをエージェントとし, 二次元空間での歩行を学習させる.

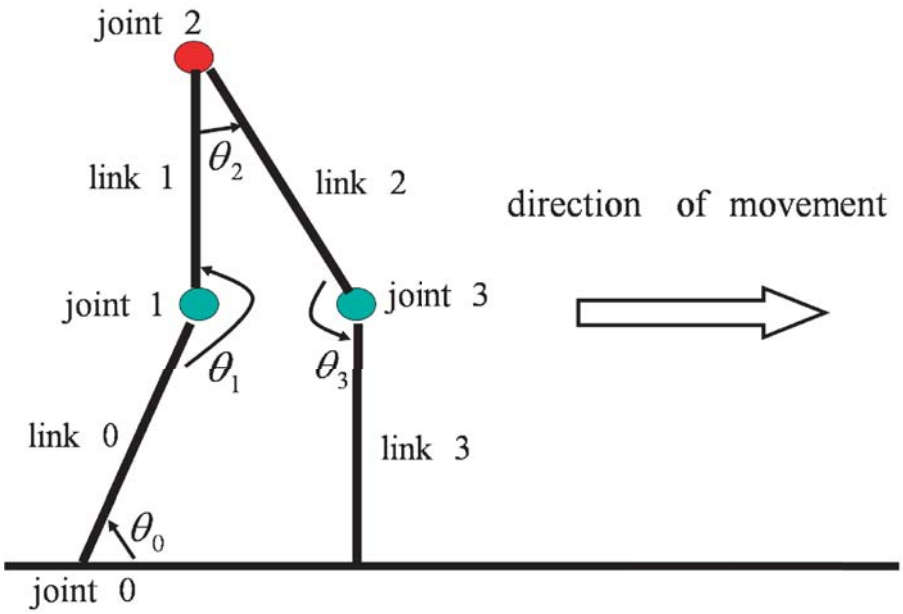


図 5.1: 二足歩行問題

エージェントは, joint1, joint2, joint3 でトルクをかけることができる. エージェントは各関節の角度や角速度に応じて歩行に適したトルクのかけかたを学習する.

二本の足で安定して体を支えるためには, 細やかなトルク制御が必要となる. また, ポールバランシング問題と比較して, 状態, 行動の次元数が増え, 探索空間が巨大になることから, 二足歩行問題は難しいタスクであると言える.

二足歩行学習の一回の試行は

- エージェントが転倒する
- リンクの角速度が一定値を越える

- 制限時間に達する

の条件のいずれかを満たすときに終了する. 試行が終了すると, 状態, 行動出力, 適格度トレースを初期化し, 再び新しい試行を始める.

5.2 設定

5.2.1 シミュレーション設定

まず, シミュレーションに用いられるパラメータの設定を表 5.1 に示す.

表 5.1: シミュレーションに用いられるパラメータの設定

パラメータ	記号	単位	初期値	範囲
joint j の変位	$\mathbf{x}_j = (x_j, y_j)$	m m	$x_0 = (0, 0)$	
joint 0 の角度	θ_0	rad	$\frac{\pi}{3}$	$\left[\frac{-\pi}{6}, \pi\right]$
joint 1 の角度	θ_1	rad	$\frac{7\pi}{6}$	$\left[\frac{\pi}{2}, \frac{3\pi}{2}\right]$
joint 2 の角度	θ_2	rad	$\frac{\pi}{6}$	$\left[\frac{-3\pi}{10}, \frac{3\pi}{10}\right]$
joint 3 の角度	θ_3	rad	$\frac{5\pi}{6}$	$\left[\frac{\pi}{2}, \pi\right]$
joint j の速度	$\dot{\mathbf{x}}_j = (\dot{x}_j, \dot{y}_j)$	m/s m/s	全ての joint で (0, 0)	
joint j の角速度	$\dot{\theta}_j$	rad/s	全ての joint で 0	$[-4\pi, 4\pi]$
joint j の加速度	$\ddot{\mathbf{x}}_j$	m/s^2 m/s^2	全ての joint で (0, 0)	
joint j の角加速度	$\ddot{\theta}_j$	rad/s^2	全ての joint で 0	
link i の長さ	l_i	m	l_0, l_3 は 0.9, l_1, l_2 は 1.1	
link i の重さ	m_i	kg	m_0, m_3 は 0.9, m_1, m_2 は 1.1	
重力加速度	g	m/s^2	9.8	
joint j にかかる力	F_j	N N	全ての joint で (0, 0)	
joint 1 にかかるトルク	τ_1	N· m	0	[-15, 15]
joint 2 にかかるトルク	τ_2	N· m	0	[-20, 20]
joint 3 にかかるトルク	τ_3	N· m	0	[-15, 15]
静止摩擦係数	μ_c		1.0	
動摩擦係数	μ_p		0.7	
時間間隔	Δt	s	0.01	
制限時間	T_{max}	s	10	

ただし, i, j はリンク, 関節のインデックスで i は 0 から 3 の整数, j は 0 から 4 の整数が対応する.

運動方程式

ここでは, リンクの総数が N である N-link ロボットの運動方程式を示す. $N = 4$ とすることで, 本研究で用いる 4-link の二足歩行ロボットとなる.

図 5.2 に示すような link$i$ を考える. ただし, i はリンクのインデックスを表している. $i = (0, 1, \cdots, N-1)$. 図 5.2 の r_i は link$i$ の方向の単位ベクトルを表し, $r_i = (\cos\theta_i, \sin\theta_i)^T$

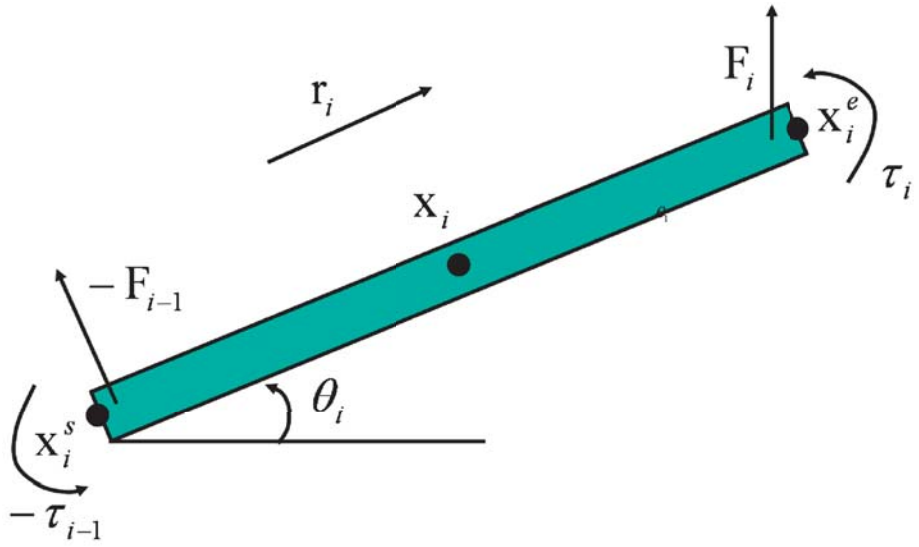


図 5.2: i 番目のリンク

で求められる. リンクの始点 $\mathbf{x}_i^s$ には反作用として, $i-1$ のリンクの終点 $\mathbf{x}_{i-1}^e$ にかかる力 F_{i-1} の逆向きの力がかかる. 同様に, 反作用として, $\mathbf{x}_i^s$ において $-\tau_{i-1}$ のトルクが作用する.

次に運動方程式を示す.

$$m_i\ddot{x}_i = -F_{i-1} + F_i \tag{5.1}$$

$$\frac{1}{12}m_i l_i^2\ddot{\theta}_i = \frac{1}{2}l_i r_i \cdot F_{i-1} + \frac{1}{2}l_i r_i \cdot F_i + (-\tau_{i-1} + \tau_i) \tag{5.2}$$

また, X_i^s, X_i^e は次の式から求められる.

$$\mathbf{x}_i^s = \mathbf{x}_i - \frac{1}{2}l_i r_i \tag{5.3}$$

$$\mathbf{x}_i^s = \mathbf{x}_i + \frac{1}{2}l_i r_i \tag{5.4}$$

式 (5.3), 式 (5.4) を二回微分することで, 次の式が得られる.

$$\ddot{\mathbf{x}}_i^s = \ddot{\mathbf{x}}_i + \frac{1}{2}l_i \begin{pmatrix} \sin\theta \\ -\cos\theta \end{pmatrix} \ddot{\theta}_i + \frac{1}{2}l_i \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \dot{\theta}_i^2 \tag{5.5}$$

$$\ddot{\mathbf{x}}_i^e = \ddot{\mathbf{x}}_i - \frac{1}{2}l_i \begin{pmatrix} \sin\theta \\ -\cos\theta \end{pmatrix} \ddot{\theta}_i + \frac{1}{2}l_i \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \dot{\theta}_i^2 \tag{5.6}$$

そして, x_i^s と x_{i-1}^e は常に同じ座標をとることから,

$$\ddot{\mathbf{x}}_i^e = \ddot{\mathbf{x}}_{i+1}^s \tag{5.7}$$

が成り立つ.

今, τ_i が与えられたとすると, 式 (5.8) の条件の下で, 式 (5.1), 式 (5.2), 式 (5.6), 式 (5.7) の連立方程式を時, $F_0, F_1, \cdots, F_N$ を求めることができる. ただし, $\mathbf{x}_0^s$ が床から離れていた場合, $F_0 = 0$, 同様に, $\mathbf{x}_N^e$ が床から離れていた場合, $F_N = 0$ となる. また, $\mathbf{x}_0^s$ が接地し, 摩擦が十分であれば, $\ddot{\mathbf{x}}_0$ となるように F_0 を決定し, 摩擦が足りなく, 滑る場合は, 動摩擦係数に従い, $\mathbf{x}_0^s$ が床を滑るように F_0 を決定した. F_N についても同様である.

リンクと床の衝突は完全非弾性衝突とした.

角度の制限

人間の関節が, ある一定の角度以上曲がらないのと同様に, 本実験で用いる 4-link ロボットにも稼働する角度の制限を設ける. 制限角度を越える方向に大きな力が作用しても, それを打ち消すようなトルク τ_{spring} をかける.

ある関節 joint$i$ が θ_{min} から θ_{max} まで可動であるとするなら, τ_{spring} は

$$\tau_{spring} = \begin{cases} \frac{-2\dot{\theta}_i}{max[0.05, \theta_{max} - \theta_i]} & (if \quad \theta_{max} - \theta_i \leq \frac{\pi}{6} \quad and \quad \dot{\theta}_i > 0) \\ \frac{2\dot{\theta}_i}{max[-0.05, \theta_{min} - \theta_i]} & (if \quad \theta_{min} - \theta_i \geq \frac{\pi}{6} \quad and \quad \dot{\theta}_i < 0) \\ 0 & (else) \end{cases} \tag{5.8}$$

で求める. $\dot{\theta}_i$ が θ_{min}, あるいは θ_{max} から遠ざかる角速度 (θ_{max} と θ_{min} の中心に向かう角速度) であるならば, τ_{spring} の出力は必要がないと考え, 0 とした. 試行では, エージェントの行動出力である τ_i と τ_{spring} を合成したトルクが 4-link ロボットにかけられることになる.

5.2.2 学習設定

強化学習は連続系のアクター・クリティック法を用いる. これは, 2 章で述べたように, 二足歩行はなめらかな動作が求められており, アクター・クリティックは連続系への応用

が容易であるためである. ただし, シミュレーションは離散時間によって行われるので, 実際に用いた学習アルゴリズムは 2 章で述べた連続系のアクター・クリティック法とは若干異なっている. 実際に用いた学習手順を付録 A.1 に記す.

次に学習に用いられたパラメータの設定は表 5.2 の通りである. α は状態評価関数, β は方策関数の更新に用いられる学習率である.

表 5.2: 学習に用いられるパラメータの設定

パラメータ	記号	値
学習率 V	α	5
学習率 π	β	10
時定数	τ	1
時定数	κ	0.5
ノイズの標準偏差	σ	0.5
ノイズの拘束期間	Δt_{noize}	10

歩行には, 自分自身の姿勢, 移動速度の情報が必要であることから, エージェントが観測する状態 $\mathbf{x}(t)$ は $(\theta_0, \theta_1, \theta_2, \theta_3, \dot{\theta_0}, \dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3})$ の 8 次元とした. 行動 $\pi(t)$ は joint 1, joint 2, joint 3 でかけられるトルク (τ_1, τ_2, τ_3) の 3 次元である.

報酬関数 $r(\mathbf{x}(t))$ は

$$r(\mathbf{x}(t)) \quad = \quad \dot{x}_2 + 0.5y_2 - 2.0 \tag{5.9}$$

とした. $r(\mathbf{x})$ は joint 2 の速度と高さに応じた値を返す. つまり, ロボットの腰の水平方向の速度が大きく, 腰が高い位置にあるほど, 大きい報酬を与える. また, y_2 が 0.5 を下回る, あるいは, あるリンクの始点と終点が床面に接地したとき, 転倒したと判断し, -2 のばつを試行終了時間まで与え続ける.

1 回の試行はエージェントが転倒する. または, 制限時間 10 秒に達したとき, 終了する. 1 回の実験では 20 万回の試行を重ねることにした.

5.2.3 実験設定

ここでは, 従来手法・提案手法それぞれ固有の設定について記す.

従来手法

従来手法では, ボックスの追加は行わず. NGnet の基底ユニットは全て固定である. ユニットは, $\theta_0, \theta_1, \theta_2, \theta_3$ に 8 個ずつ, $\dot{\theta}_0, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3$ に 6 個ずつ, 格子状に配置した. ユニットの総数は 5308416 個となった.

提案手法

提案手法では, 従来手法の固定ユニットは, 固定ユニットは $\theta_0, \theta_1, \theta_2, \theta_3$ に 7 個ずつ, $\dot{\theta}_0, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3$ に 5 個ずつ, 格子状に配置した. ユニットの総数は 1500625 個となった. さらに, クリティカル状態を検出するたびにボックスを追加, もしくは再配置する. それぞれのボックスは $\theta_0, \theta_1, \theta_2, \theta_3$ に 3 個ずつ, $\dot{\theta}_0, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3$ に 2 個ずつ, 計 1296 個のユニットで構成される.

また, 表 5.3 に提案手法に関するパラメータ設定を記す.

表 5.3: 提案手法で用いられるパラメータの設定

パラメータ	記号	値	単位
提案手法を適用する間隔	$\Delta_{recovery}$	10000	回
検出に用いる比較用サンプル	Δ_{sample}	1000	試行
比較元データの切取区間	ld	1.0	s
比較先データとの比較区間	lm	0.5	s
検出のための比較区間	lp	0.1	s
検出に考慮する次元設定ベクトル	D	$(1,1,1,1,0,0,0,0)$	
割引パラメータ	α	3.0	
割引率	γ	0.9	
ボックスの各次元における基底配置範囲	Fl	$(\frac{1}{6}\pi, \frac{1}{7}\pi, \frac{3}{35}\pi, \frac{1}{14}\pi, \frac{4}{3}\pi, \frac{4}{3}\pi, \frac{4}{3}\pi, \frac{4}{3}\pi)$	
ボックスの各次元における基底配置個数	Bs	$(3,3,3,3,2,2,2,2)$	
比較先データ生成に対する基準値	μ_{match}	0.01	
検出に対する基準値	$\mu_{critical}$	6.0	
ボックスの発火量に対する基準値	μ_{fire}	0.0	

ボックスの基底配置範囲は, 原則固定ユニットの格子状配置間隔と等しくした. また, お手本サンプル E_{bst} は 1 つである. 全試行中, 最も長く歩いた試行をお手本サンプル E_{bst} とする.

5.3 実験結果

5.3.1 従来手法による二足歩行の学習

従来手法による二足歩行の学習を行った. 図 5.3 に結果のグラフを示す. ただし, グラフの縦軸は試行回数, 横軸は歩行距離である.

図 5.3 から, 試行を重ねることで, 緩やかに歩行距離がのびていることが分かる. 学習が

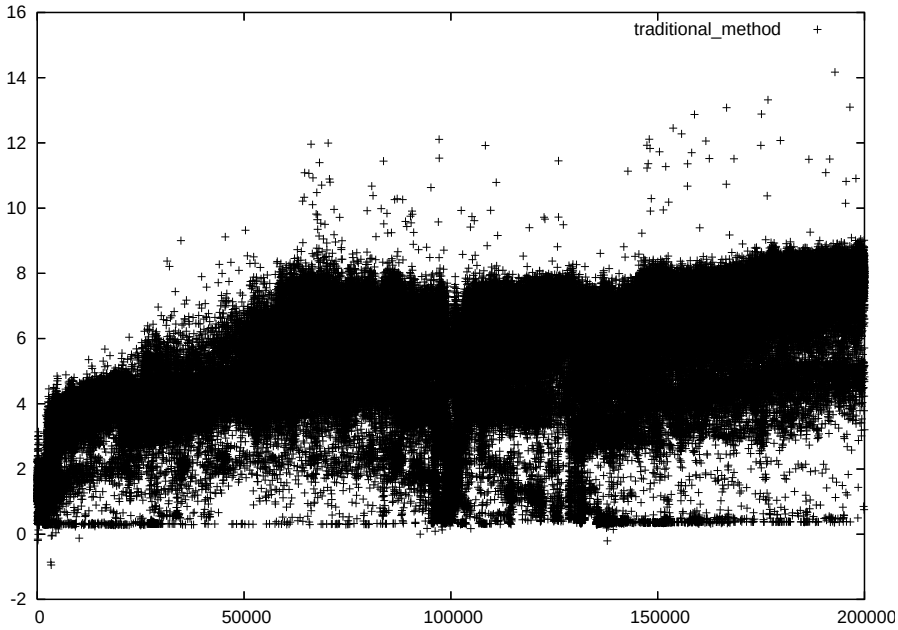


図 5.3: 従来手法を用いた二足歩行の学習

収束せず, 揺れ幅が大きいのは, 行動の次元数が 3 であり, その各々に対してノイズが影響するため, バランスをとるのが難しく転倒してしまうためだと推測している.

5.3.2 提案手法による二足歩行の学習

提案手法による二足歩行の学習を行った. 提案手法の適用間隔は 10, 000 回の試行とした. お手本サンプルは, その実験でなされた最も歩行距離が長い試行である. この結果を図 5.4 に示す. 図 5.4 から, 回数を重ねることで, 歩行距離は急激にのびるが, ある回数では急激に落ちていることがわかる. 図 5.4 と図 5.5 を見比べるとボックスをあたらしく配置した試行で, 歩行距離が急激に落ちている個所がある一方, 急激に歩行距離がのびている区間があることもわかる. これは, クリティカル状態に基底を集中化することで, 少ない学習期間で近似精度が上がりやすくなる反面, あらたに基底を配置することで元の学習が進んでいる状態価値関数, 方策関数の形を変えてしまい一時的に制御が不安定になるためであると推測している.

提案手法において, 10000 試行ごとに検出されたクリティカル状態の数を, 図 5.5, その分布を x 軸を θ_0, y 軸を θ_1, z 軸を θ_2 として, 図 5.6-図 5.18 に示す.

図 5.6-図 5.18 を見ると, クリティカル状態は試行によって様々な個所に散らばっていることがわかる. これは, 学習がすすむにつれてお手本サンプルや全体の比較元サンプルの歩き方が変わることによると推測している. また, クリティカル状態付近のエージェント

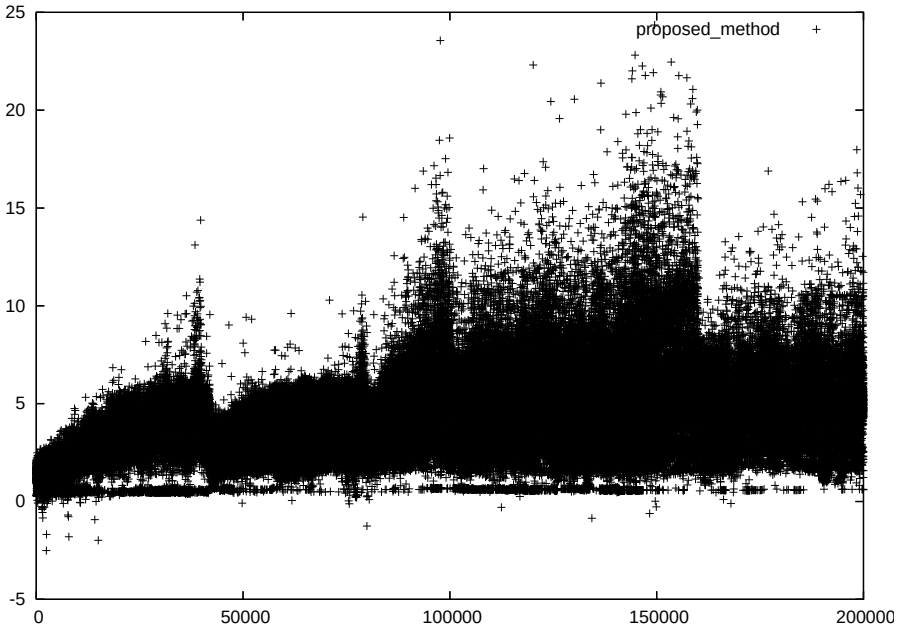


図 5.4: 提案手法を用いた二足歩行の学習

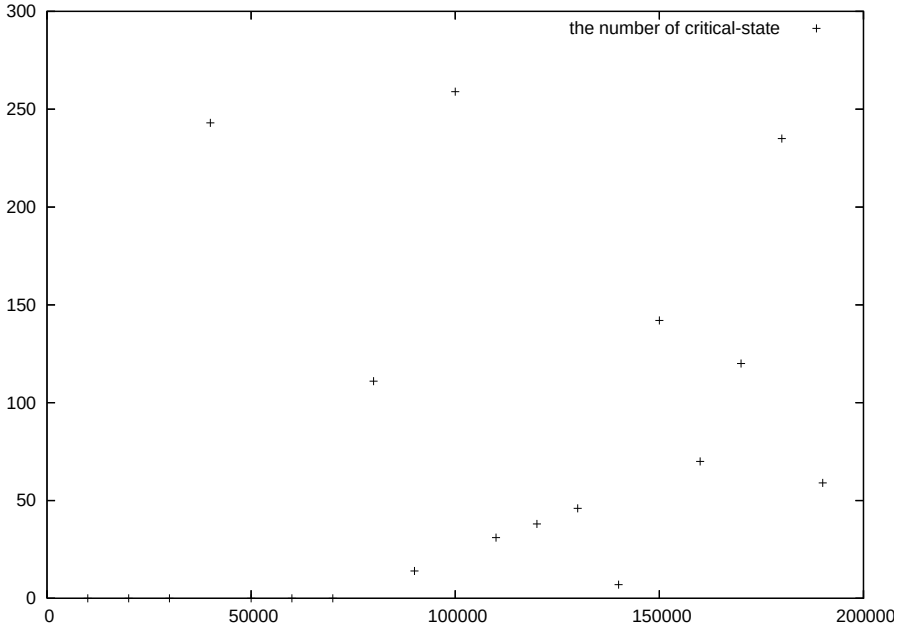


図 5.5: 検出されたクリティカル状態の数

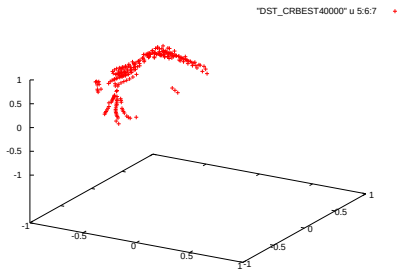


図 5.6: 40000 試行目で検出されたクリティカル状態

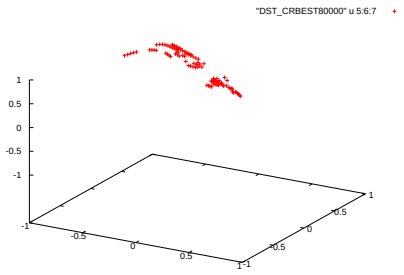


図 5.7: 80000 試行目で検出されたクリティカル状態

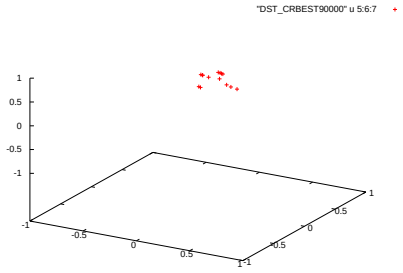


図 5.8: 90000 試行目で検出されたクリティカル状態

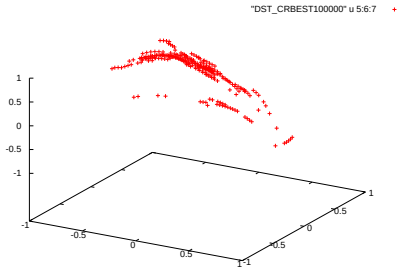


図 5.9: 100000 試行目で検出されたクリティカル状態

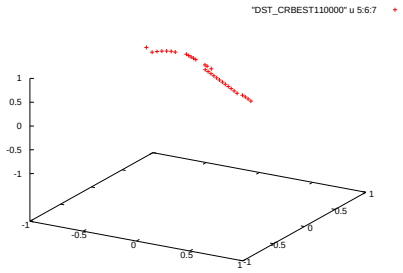


図 5.10: 110000 試行目で検出されたクリティカル状態

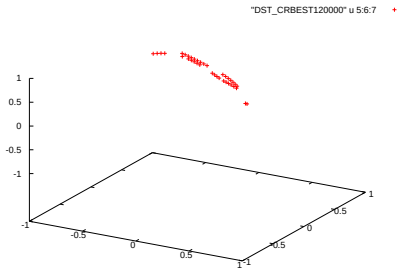


図 5.11: 120000 試行目で検出されたクリティカル状態

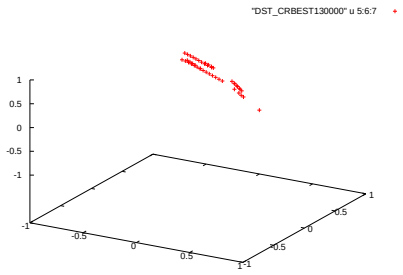


図 5.12: 130000 試行目で検出されたクリティカル状態

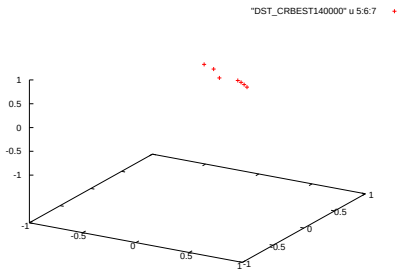


図 5.13: 140000 試行目で検出されたクリティカル状態

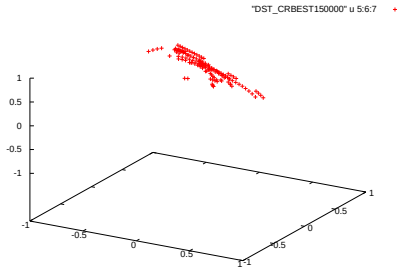


図 5.14: 150000 試行目で検出されたクリティカル状態

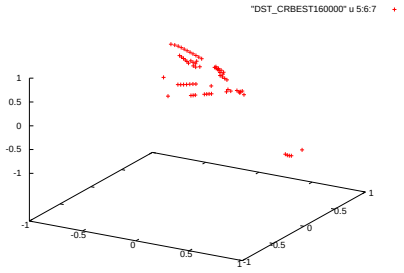


図 5.15: 160000 試行目で検出されたクリティカル状態

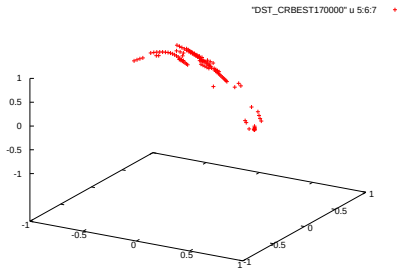


図 5.16: 170000 試行目で検出されたクリティカル状態

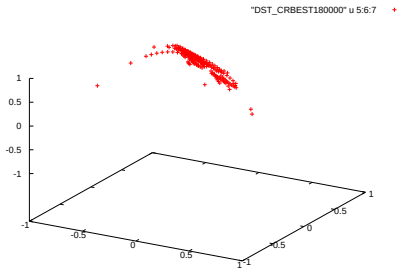


図 5.17: 180000 試行目で検出されたクリティカル状態

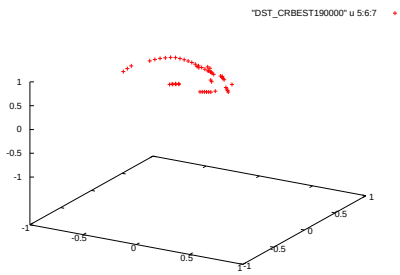


図 5.18: 190000 試行目で検出されたクリティカル状態

の試行をアニメーションで観察すると, クリティカル状態は大きく足を振り上げる時と着地する時に, 集中して現れることが多い. これは, 二足歩行問題において, おおよそ目測で検出するクリティカル状態と一致する.

この結果により, 学習が進むにしたがって, 近似すべき位置が移り変り, 適応的な近似が実現できていることがわかる. また, クリティカル状態に対しては適切に検出できていることがわかる. 具体的に挙動データで説明する. 比較元サンプルの終端状態付近の挙動と, お手本サンプルの挙動データとの比較の結果, 提案手法で検出されたクリティカル状態の一つを, 図 5.19, 図 5.20 に示す. それぞれ, 横軸は時間 s, 縦軸はエージェントが取っている関節の角度 θ_1, θ_2 である. それぞれの図を見ると, お手本サンプルと途中まで一致しているが, 途中からずれ始めている点を抽出していることがわかる.

5.3.3 従来手法と提案手法の比較

従来手法と提案手法, それぞれの実験を 10 回行い平均値をとった結果を示す. 図 5.21 は単なる平均値のプロットである. 図 5.22 は平均値のプロットを 500 回ごとに平均化してある. 図 5.24 は使用しているユニットの個数である.

図 5.21 を見ると, 提案手法は従来手法にくらべて振れ幅が広がっている. これは, 2 点原因があると推測する. 第一に, 図 5.3, 図 5.4 を見比べると, 提案手法の方が, 歩行距離の最高値について, 10 程度の差があること, 第二に, 基底を集中化する際に, 重みをリセットする必要があるため一時的に制御が不安定になり, 歩行距離が急激に低下することにある.

各手法（図 5.3, 図 5.4）で最も歩いた挙動 (θ_2) に関して, 図 5.23 で比較すると, 提案手法は従来手法に比べて, 周期性や周期の形が一定であり, よりなめらかな動作が獲得できていることがわかる. これは, エージェントが躓いている個所をクリティカル状態として検出し, 基底の集中化を施すことでその付近の近似精度が向上し, 躓きや転倒を回避する適切な制御が獲得でき, なめらかな動作が獲得できたことがわかる.

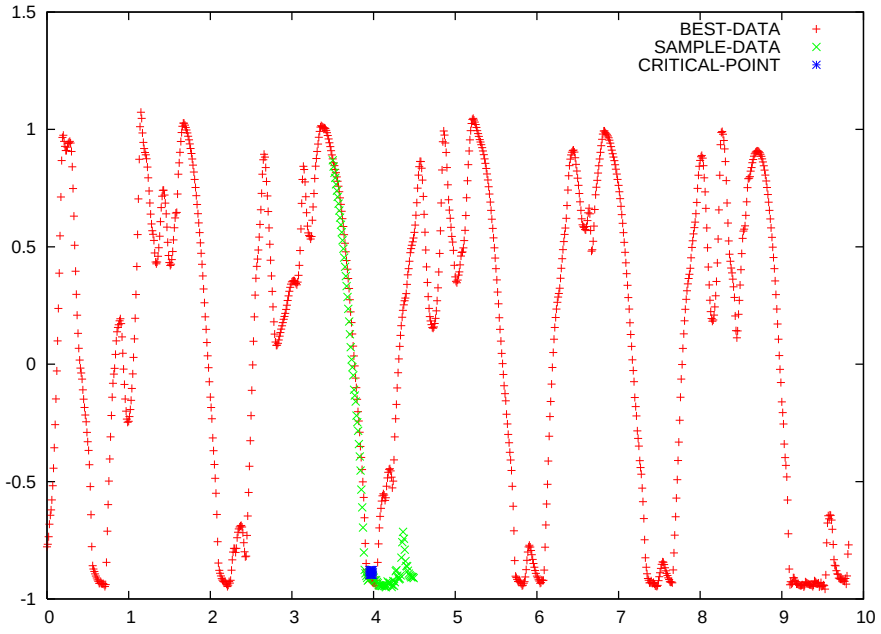


図 5.19: θ_1 の挙動でのクリティカルポイント

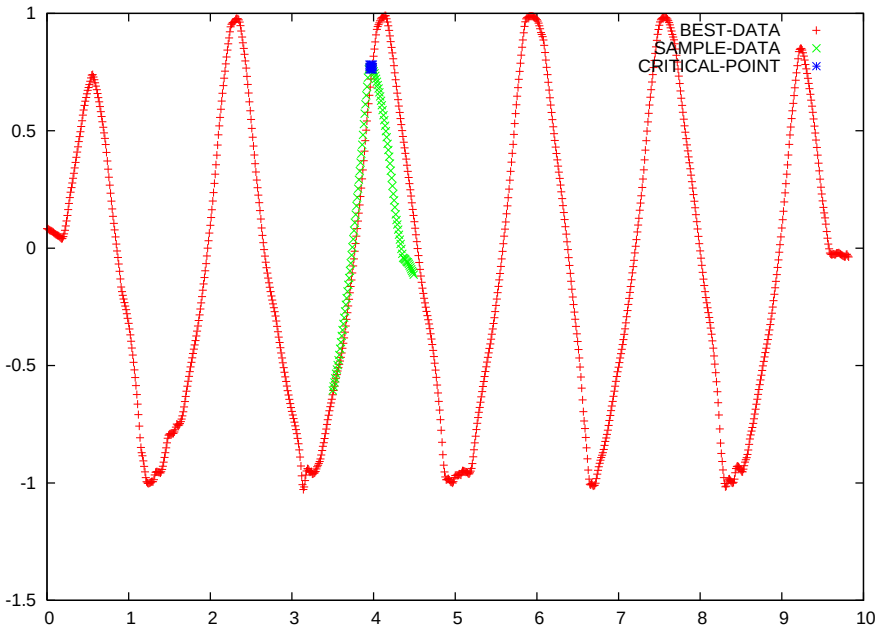


図 5.20: θ_2 の挙動でのクリティカルポイント

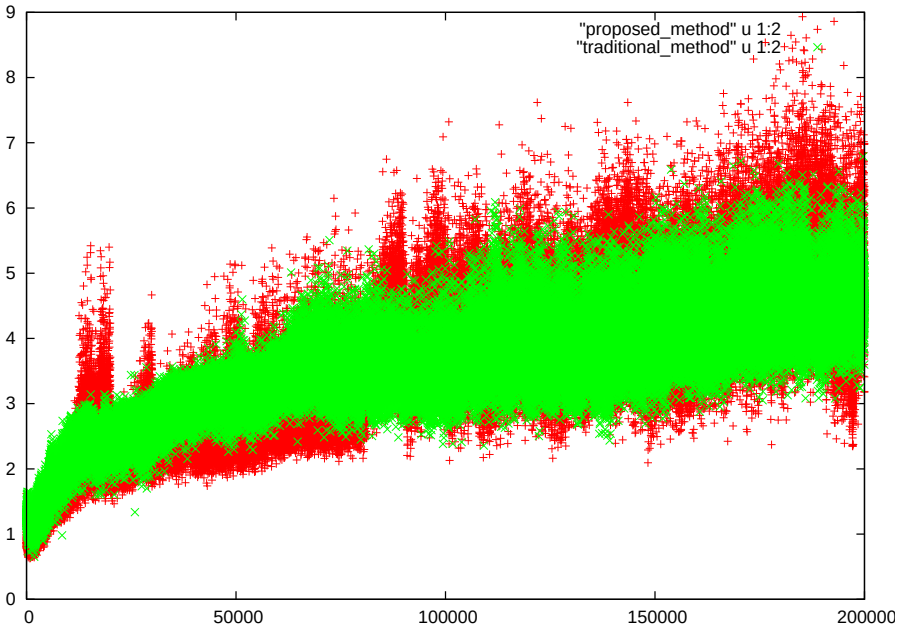


図 5.21: 歩行距離に関する従来手法と提案手法の比較

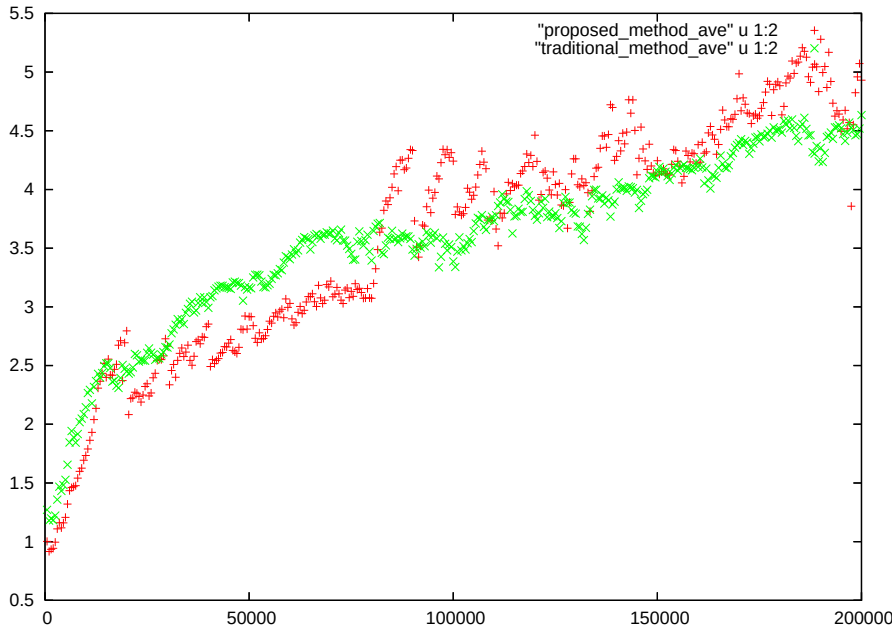


図 5.22: 歩行距離に関する従来手法と提案手法の比較（500 試行平均）

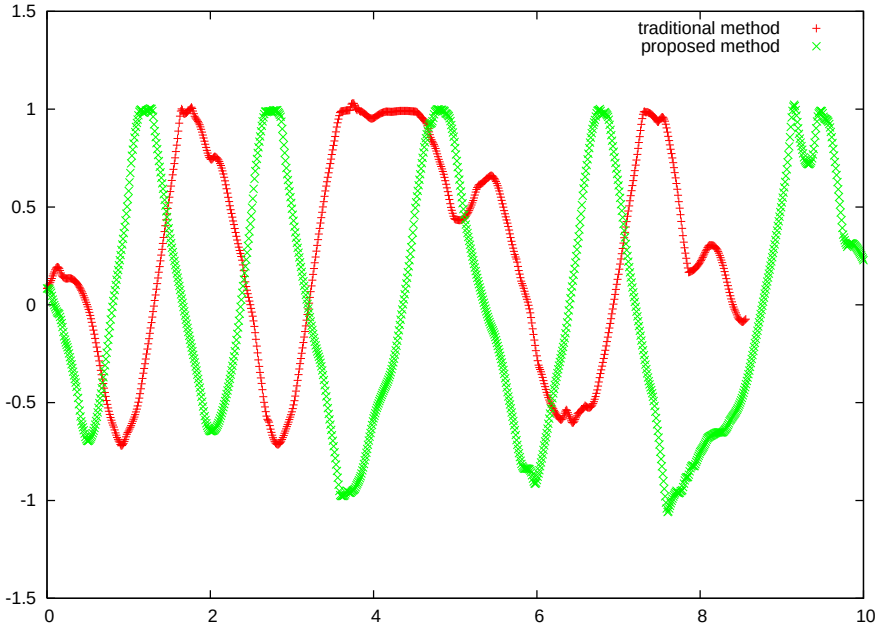


図 5.23: 挙動に関する従来手法と提案手法の比較

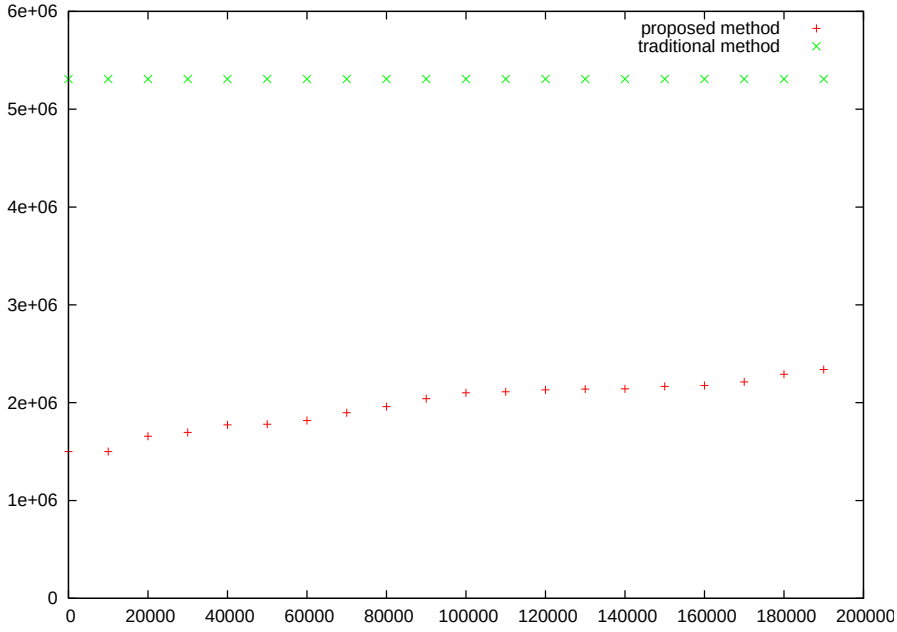


図 5.24: 使用したユニット数に関する従来手法と提案手法の比較

図 5.22 を見ると, 提案手法は学習の初期段階においては従来手法より歩行距離が劣るが, 学習が進むにつれて, 学習精度が同等以上になることがわかる. また, 図 5.24 と見比べるとユニットを配置した個数が増えるにつれて, 学習が促進されていることがわかる. さらに学習に使用しているユニットの個数について, 提案手法は, 従来手法のたかだか 44.06 %のユニットしか使用していないにも関わらず, 同等以上の学習精度を実現している. これは, ユニットをクリティカル状態に集中化することは, 学習の加速という点において有効性があることを示している.

第6章　結論と今後の課題

6.1　結論

本研究では, クリティカル状態を利用した適応的関数近似による強化学習手法を提案した.

予備実験においてポールバランシング問題で, 基底ユニットの数と学習精度の影響, および挙動が不安定になる特定の状態をクリティカル状態とし, 基底の集中化を試みた. その結果, クリティカル状態に基底を集中化することで精度が向上することを確認した. ただし, ポールバランシングの問題上クリティカル状態とエージェントがよく訪れる状態が同一の可能性がある.

クリティカル状態を「エージェントの動作が不安定になり終端状態に陥るきっかけとなる状態」とし, 学習の進行状況に応じて適応的に自動検出し, 基底集中化した. そうすることで, すくないユニットで, 方策関数, 状態価値関数の近似精度の向上, および学習の効率化を図った.

実験では, 高次元である二足歩行を学習対象とし, 従来手法（ユニットの格子状固定配置）と比較して, 従来手法の 44.06 %のユニットで同等以上の学習精度を実現可能なことを示すことができた. また, 従来手法より比較的なめらかな制御が獲得できた. この結果により, 提案手法の有効性を示すことができたと言える.

6.2　今後の課題

提案手法ではユニットを集中化する際に重みの値を一定値に初期化している. これは, 学習が進んだ関数の形を局所的にが変えてしまい, 学習が不安定になる容易になっている. 今後はユニットを集中化する際, 基底の重みの初期値を, 関数の形を変えないよう, 最適化する機構を組み込むことが必要であると考える. また, クリティカル状態の性質に関する調査も必要であると考える. 具体的にはエージェントの訪問回数と, TD 誤差との関連性を調べてみる必要がある.

将来的な試みとしては, 政策を共有した上で, マルチエージェントで学習を行い, クリティカル状態検出に対して, 互いのお手本サンプルと比較用サンプルを利用しあう試みがあげられる.

参考文献

[1] Richard S. Sutton and Andrew G.Barto, 強化学習, 森北出版株式会社.

[2] G.Tesauro, TD-gammon, a self-teaching backgammon program, achieves master-level play Neural Computation, 6, pp.215-219, 1994

[3] J.Hu and M.P.Wellman, Nash Q-learning for general-sum stochastic games Journal of Machine Learning Research 4, pp.1039-1069, 2003

[4] 藤田 肇, 石井信, 部分観測カードゲームのためのモデル同定型強化学習 電子情報通信学会論文誌, D Vol.J88-D-II, No.11, pp2277-2287, 2005

[5] 木村 元, 小林重信, ロボットアームのほふく行動の強化学習：確率的傾斜法による接近人工知能学会誌, Vol.14, No.1, pp122-130, 1999

[6] 吉本 潤一郎, 石井 信, 佐藤 雅昭, オンライン EM アルゴリズムによる強化学習法の acrobot 制御への応用, 電子情報通信学会論文誌, D-II Vol.j83-D-II, No.3, pp.1024-1033, 2000.

[7] Zhang.W and Dietterich.T, A reinforcement learning approach to job-shop scheduling In Proceedings of the 14th International Joint Conference on Artificial Intelligence, pp.1114-1120, 1995

[8] 大久保隆晴, 亀谷由隆, 佐藤泰介, 事例に基づく関係的な強化学習のエレベータ制御問題への適用 第 19 回人工知能学会全国大会予稿集, 2005

[9] 石井 信, 佐藤 雅昭, 正規化ガウス関数ネットワーク, Mixture of experts と EM アルゴリズム, 日本神経回路学会誌, vol.6, No.1, pp.30-40, 1999.

[10] 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:起き上がり動作の獲得, 電子情報通信学会論文誌, D-II Vol.j79-D-II pp 1-13, 1996.

[11] Kenji Doya, Reinforcement Learning In Continuous Time and Space, Neural Computation, 12(1), 219-245 2000.

[12] 三好 正敏, インタラクティブにユーザが教示を与える強化学習の研究, 北陸先端科学技術大学院大学, 修士論文, 2005.

付 録A

A.1 連続時間強化学習の計算機上での実現 [10]

本研究では連続時間の強化学習を扱うが，計算機でのシミュレーションでは時間を有限の時間間隔 Δt で離散化しなけれあならない．しかし，連続時間での学習アルゴリズムを用いることで，時間間隔 Δt にとらわれない評価関数などの表現が可能になるという利点を持つ．

連続時間で定義された TD 誤差を求める際には，状態価値関数の時間微分が必要となる．本研究では，ある時刻 t での状態価値関数の時間微分 $\dot{V}(t)$ を

$$\dot{V}(t) = \frac{V(t) - V(t - \Delta t)}{\Delta t} \tag{A.1}$$

とし，TD 誤差は

$$\delta(t) = r(\mathbf{x}(t)) - \frac{1}{\tau}V(t) + \dot{V}(t) \tag{A.2}$$

と求める．

これを含めた計算機上での強化学習アルゴリズムを以下に示す．

1. クリティック，アクターを初期化する．
2. 適格度トレースを初期化，時刻を 0 とする．(a) から (b) を繰り返す．
   (a) 現在の時刻を t とする．式 2.23 を用いて状態 $\mathbf{x}(t)$ における行動出力 $a(t)$ を決定する．
   (b) 式 2.21 のように，適格度トレースを更新する．

$$e_i(t + \Delta t) = (1 - \frac{\Delta t}{\kappa})e_i(t) + b_i^C(t)\Delta t \tag{A.3}$$

   ただし，$b_i^C(t)$ はクリティックの基底関数である．
   (c) (b) でもとめた行動出力 $a(t)$ を用いてエージェントのダイナミクスを計算する．
   (d) 時刻は $t + \Delta t$ となる．エージェントは状態 $x(t + \Delta t)$ を観測し，環境から $r(t + \Delta t)$ を得る．状態 $x(t + \Delta t)$ が終了条件んを満たすなら 3. へ．
   (e) 次の式を用いて TD 誤差 $\delta(t + \Delta t)$ を計算する．

$$\delta(t + \Delta t) = r(t + \Delta t) - \frac{1}{\tau}V(\mathbf{x}(t + \Delta t)) + \frac{V(\mathbf{x}(t + \Delta t)) - V(t)}{\Delta t} \tag{A.4}$$

(f) 式 2.22 のように，状態価値関数を更新する．

$$v_i \leftarrow v_i + \alpha\beta(t+\Delta t)e_i(t+\Delta t)\Delta t \tag{A.5}$$

(g) 式 2.24 のように，方策関数を更新する．

$$w_i \leftarrow w_i + \beta\delta(t+\Delta t)\sigma n_j(t)b_i^A(t)\Delta t \tag{A.6}$$

ただし，$b_i^A(t)$ はアクターの基底関数である．

3. 次の式を用いて全ての $e_i(t) > 0$ となるユニットの重みを更新し，(2) へ戻る．

$$v_i \leftarrow v_i + \alpha \cdot e_i(t+\Delta t)(\tau \cdot r(\mathbf{x}(t)) - V(t+\Delta t)) \tag{A.7}$$