修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院 情報理工学研究科 情報・ネットワーク工学専攻 博士前期課程		
氏　　名	吉田 直人	学籍番号	1731167
論 文 題 目	チェス人工知能が提示する複数の選択肢から着手する 人工知能の強化学習		

要　旨

人工知能が活躍する場面が増えつつある現代において、人工知能は大量の情報を適切に集約し意思決定を行わなければならない。多数の知識が集まれば全体としてより良い意思決定を行うことができることが一般には知られており、このような性質を持つ知能を集合知と呼ぶ。近年大きな成果を上げているゲーム人工知能の分野で集合知に関連する研究が行われており、その例として Althöfer らの Multiple Choice System の研究などがある。Multiple Choice System は人工知能がゲームの候補手を提示し、ボスと呼ばれる人間がそれらの中から一つを選択するシステムである。Althöfer らはチェスにおいて Multiple Choice System の Elo レーティングがベースとなるゲーム人工知能の Elo レーティングより高くなる可能性を示した。

本研究の目的は知識を適切に集約し意思決定を行う人工知能、ボス人工知能を強化学習やニューラルネットワークを用いて作成し、その性能を調査することである。題材はチェスとし、Multiple Choice System のボスをボス人工知能に置き換える。強化学習法は Watkins の Q(λ)と方策オフ型モンテカルロ法を用いる。ニューラルネットワークは畳み込み層を用いた様々な構成を用いる。実験の結果、Watkins の Q(λ)と一部のニューラルネットワークの構成の組み合わせで、単純にチェスの指し手を選択する方法より良い選択方法を学習したボス人工知能が作成できた。一番性能の良い強化学習法は Q(0.9)であった。ニューラルネットワークの構成について、各構成要素がどのように性能に関係しているかは明らかにならなかった。明らかにならなかった原因として学習が収束していないことが考えられ、その理由として重み更新回数が足りなかった、訓練サンプルを再利用すべきだった、訓練サンプルが独立でなかった、学習係数を段階的に小さくしていく必要があった、などの事項が考えられる。

電気通信大学 情報理工学研究科
情報・ネットワーク工学専攻 修士学位論文

チェス人工知能が提示する複数の選択肢から着手する人工知能の強化学習

平成 31 年 3 月 5 日

学籍番号 1731167

吉田直人

指導教員 保木邦仁

目次

1 はじめに

近年、ゲームをプレイする人工知能の活躍が目覚ましい。囲碁では 2017 年に人工知能 AlphaGo が中国棋士ランキング 1 位のカ・ケツ九段に勝利し[*1]、将棋では同年に人工知能 Ponanza が佐藤天彦名人に勝利した[*2]。チェスでは 1997 年において既に人工知能 DeepBlue が当時の FIDE 世界ランキング 1 位であるガルリ・カスパロフに勝利している[*3]。成果を上げた先行事例が多く存在することから、人工知能の研究においてゲームを題材にすることは妥当と考えられる。

ゲーム以外の分野においても人工知能が活躍する機会は増えている。ネットコンテンツでは人工知能が利用者の情報を分析して、コンテンツのレコメンドを行うことが当たり前になっている[*4]。このような分析を行うため人工知能は大量のデータを適切に集約し、意思決定を行わければならない。人工知能の分野が発展していくために、人工知能に高度な意思決定能力を実装する様々な手法を探すことは重要ではないかと考える。

三人寄れば文殊の知恵という慣用句がある通り、多数の知識が集まれば全体としてより良い意思決定を行うことができることが一般には知られている。このような性質を持つ知能は集合知と呼ばれ研究されており、状況により集合知が機能する場合と機能しない場合があることがわかっている。1999 年に行われたカスパロフ対ワールドというチェスのイベントでは、世界各地からネットを介して参加した約 5 万人ほどのアマチュアチェスプレイヤの連合と、当時の FIDE 世界ランキング 1 位であるガルリ・カスパロフが対局し、連合側は負けてしまったがカスパロフ相手に善戦した [1]。このとき連合側には 1 手 24 時間の持ち時間が与えられ、ネットのフォーラムにてプロ級のチェスプレイヤのアドバイスを含めた熟議の末電子投票による多数決で着手が決定された。しかし 1996 年に開催されたカルポフ対ワールドというチェスのイベントでは、アナトリー・カルポフというカスパロフに劣らないプロチェスプレイヤと、カスパロフ対ワールドと同様にして集まったチェスプレイヤ連合が対局したが、こちらはカルポフの圧勝で終了した [1]。このとき連合側は 1 手 10 分という少ない持ち時間が与えられ、プレイヤ同士の議論はほぼ無しに電子投票による多数決で着手が決定された。このように知識の集約方法が多数決のみなどの単純な方法だと集合知は上手く機能しない場合がある。

多数の知識を集約して一つの意思決定を行うシステムを、ゲームをプレイする人工知能を用いて構成した研究が複数存在し、その例として Althöfer らによる Multiple Choice System [2] や伊藤らによる合議アルゴリズム [3] の研究がある。

Multiple Choice System は人工知能がゲームの候補手を提示し、ボスと呼ばれる人間がそれらの中から一つを選択するシステムである。Althöfer らはチェスにおいて、Multiple Choice System の Elo レーティングがベースとなるゲーム人工知能の Elo レーティングより高くなる可能性を示

[*1] https://www.asahi.com/and_M/articles/SDI2017062384131.html
[*2] https://www.nikkei.com/article/DGXLASDG01HCT_R00C17A4CC1000
[*3] https://www.nikkei.com/article/DGXKZO26472440S8A200C1MY7000
[*4] http://www.asahi.com/digital/mediareport/TKY201003100152.html

した。

合議アルゴリズムは複数の独立した人工知能が個々に導いたゲームの候補手の中から、何らかの方法で一つの手を選択するアルゴリズムである。伊藤らは合議アルゴリズムを用いたシステムが、ベースとした各々のゲーム人工知能に有意に勝ち越すことを示した。

本研究の目的は知識を適切に集約し意思決定を行う人工知能、ボス人工知能を作成することである。題材はチェスとし、Multiple Choice System のボスをボス人工知能に置き換える。

2　基礎知識

本章では本研究に関連する基礎知識について述べる。

2.1　ニューラルネットワークとその学習方法

ニューラルネットワークは脳機能の一部を模した数学モデルである。シナプスの結合により形成されたネットワークが学習によりその結合の強度を変化させ、問題解決能力を持つようなモデル全般のことを指す。

本節の内容は [4] を参考にしている。

2.1.1　順伝播型ニューラルネットワーク

順伝播型ニューラルネットワークは、ニューラルネットワークの一種であり、層状に並んだユニットとユニット間の結合から構成される。各ユニットは結合している一つ前の層のユニットから値を受け取り、その値から計算された別の値を次の層の結合しているユニットに出力する。同じ層のユニット同士は結合せず、隣り合う層のユニット同士が結合する。順に並んだ層の中で、最初の層を入力層、最後の層を出力層、入力層と出力層の間の層を中間層と呼ぶ。入力層のユニットに入力を与えることで、各ユニットの入力と出力が計算される。図 1 に順伝播型ニューラルネットワークの例を示す。

ユニットが受け取る値はそのユニットに結合する一つ前の層のユニットの出力にそれぞれ異なる重みをかけた値の和にバイアスと呼ばれる定数を足した値になる。重みとバイアスを合わせてパラメータと呼ぶ。第 $l+1 (l = 1, 2, ..., L)$ 層の j 番目のユニットが受け取る値を $u_j^{(l+1)}$、第 l 層の i 番目のユニットの出力を $z_i^{(l)}$、第 $l+1$ 層の j 番目のユニットと第 l 層の i 番目のユニット間の重みを

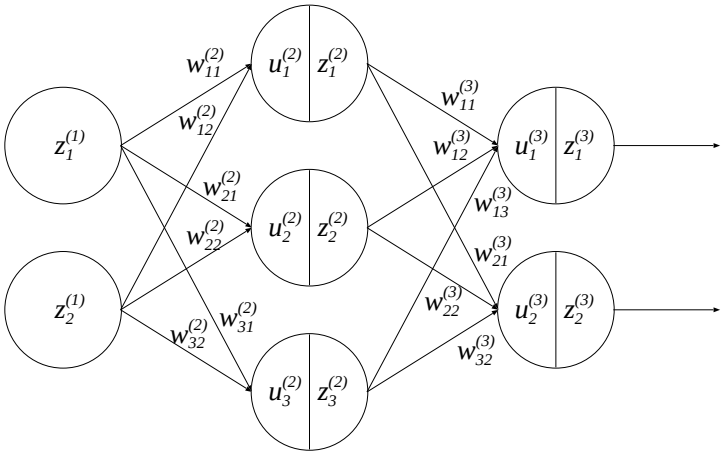


図 1　順伝播型ニューラルネットワークの例

$w_{ji}^{(l+1)}$、第 l+1 層の j 番目のユニットのバイアスを $b_j^{(l+1)}$ と表すと、全結合層と呼ばれる一つ前の層の全てのユニットがその層のそれぞれのユニットに結合を持つような層では

$$u_j^{(l+1)} = \sum_i w_{ji}^{(l+1)} z_i^{(l)} + b_j^{(l+1)} \tag{1}$$

となる。

ユニットが出力する値はそのユニットが受けとった値に活性化関数と呼ばれる関数をかけたものになる. 活性化関数を f とすると

$$z_j^{(l+1)} = f(u_j^{(l+1)}) \tag{2}$$

となる。活性化関数として ReLU

$$f(u_j^{(l)}) = \max(0, u_j^{(l)}) \tag{3}$$

がよく用いられる。

順伝播型ニューラルネットワークに入力 x_j を与えるときは

$$z_j^{(1)} = x_j \tag{4}$$

とする。$z_j^{(1)}$ から各ユニットの入力、出力の値が定まる。

2.1.2 畳込み層

隣接層間の特定のユニットのみが結合を持つ特別な層を畳込み層と呼ぶ。畳み込み層は入出力が主に画像となる。畳込み層を用いた順伝播型ニューラルネットワークは画像認識などで頻繁に用いられる。

濃淡値を各画素に格納したグレースケールの画像を考える。画像サイズを $W \times W$ とし、画素をインデックス $(i, j)(i = 0, ..., W-1, j = 0, ..., W-1)$ で表す。画像の左上角、右下角の画素のインデックスはそれぞれ $(0, 0)$、$(W-1, W-1)$ である。画素 (i, j) の画素値を x_{ij} と表し、実数値をとるとする。またフィルタと呼ぶサイズの小さい画像を考え、そのサイズを $H \times H$ とする。フィルタの画素はインデックス $(p, q)(p = 0, ..., H-1, q = 0, ..., H-1)$ で表し、画素値を h_{pq} と表す。h_{pq} は x_{ij} と同様に実数値とする。画像の畳込みとは、入力画像とフィルタ間で定義される積和計算

$$u_{ij} = \sum_{p=0}^{H-1} \sum_{q=0}^{H-1} x_{i+p, j+q} h_{pq} \tag{5}$$

である。ただし出力画像の画素 u_{ij} は $i + p \leq W - 1$、$j + q \leq W - 1$ となるインデックス (i, j) の範囲で計算されるとする。

画像内にフィルタ全体が収まる範囲でフィルタを動かすと、畳込みの出力画像サイズは入力画像サイズよりも小さくなる。このときそのサイズは

$$(W - 2\lfloor H/2 \rfloor) \times (W - 2\lfloor H/2 \rfloor)$$

となる。$\lfloor\cdot\rfloor$ は小数点以下を切り上げて整数化する演算子とする。例えば、8×8 の入力画像に 3×3 のフィルタを用いて畳込みを行うと、出力画像サイズは式 2.1.2 より $(8-2\lfloor 3/2\rfloor)\times(8-2\lfloor 3/2\rfloor) = 6\times 6$ となる。

畳込み層の出力画像が入力画像と同サイズとなると都合が良い場合がある。その場合には入力画像の外周に新たに幅 $\lfloor H/2\rfloor$ で画素を加え、出力画像のサイズが入力画像のサイズと同一となるようにする。新たに加えた画素の画素値は未定であり、何らかの方法で決める必要があるが、最も一般的であるのが画素値を 0 とすることである。このように入力画像の外周に画素値 0 の画素を加えることをゼロパディングと呼ぶ。

実用的な畳込み層ではグレースケールの画像 1 枚だけでなく、多チャネルの画像に対し、複数個のフィルタを用いて畳込みを行う。多チャネルの画像とは各画素が複数の値を持つ画像であり、チャネル数が K の画像の各画素は K 個の画素値を持つ。画像の縦横サイズが $W\times W$ で、チャネル数が K のとき、この画像のサイズを $W\times W\times K$ と表すことにする。

複数の畳込み層を用いた順伝播型ニューラルネットワークを考える。第 l 層の畳込み層は直前の第 $l-1$ 層から幅 $\lfloor H/2\rfloor$ でゼロパディングされたサイズ $(W+2\lfloor H/2\rfloor)\times(W+2\lfloor H/2\rfloor)\times K$ の入力画像 $z^{l-1}_{ijk}(k=0,...,K-1)$ を受け取り、これに M 種類のフィルタ $h_{pqkm}(m=0,...,M-1)$ を適用する。ゼロパディングされた入力画像の左上角、右下角の画素のインデックスはそれぞれ $(0,0)$、$(W+2\lfloor H/2\rfloor-1, W+2\lfloor H/2\rfloor-1)$ とする。各フィルタは入力画像と同じチャネル数 K を持ち、そのサイズを $H\times H\times K$ とする。$m=0,1,...,M-1$ の各フィルタ m について畳込みが実行され、それぞれ 1 チャネルの u_{ijm} が出力される。その計算は、各チャネル $k(=0,...,K-1)$ について画像とフィルタの畳込みを行った後、結果を画素ごとに全チャネルにわたって加算するもので

$$u_{ijm} = \sum_{k=0}^{K-1}\sum_{p=0}^{H-1}\sum_{q=0}^{H-1} z^{(l-1)}_{i+p,j+q,k} h_{pqkm} + b_m \tag{6}$$

と表される。ただし出力画像の画素 u_{ijm} は $i+p \le W+2\lfloor H/2\rfloor$、$j+q \le W+2\lfloor H/2\rfloor$ となるインデックス (i,j) の範囲で計算されるとする。b_m はバイアスである。このように、入力画像のチャネル数によらず、一つのフィルタからの出力は常に 1 チャネルとなる。最終的な畳込み層の出力 z_{ijm} は活性化関数 f を適用し

$$z_{ijm} = f(u_{ijm}) \tag{7}$$

となる。

2.1.3 学習の枠組み

L 層からなる順伝播型ニューラルネットワークの入力をベクトル $\boldsymbol{x} = [x_0, x_1, ...]^T$、出力をベクトル $\boldsymbol{y} = [z^L_0, z^L_1, ...]^T$ と表し、また全てのパラメータを要素に持つベクトルを $\boldsymbol{w}$ とする。式 1、6 などからわかるように、順伝播型ニューラルネットワークが表す関数 $\boldsymbol{y}(\boldsymbol{x};\boldsymbol{w})$ はパラメータ $\boldsymbol{w}$ を変えると変化する。よい $\boldsymbol{w}$ を選ぶことにより順伝播型ネットワークが望みの関数を与えるようになる。

ここで望みの関数を表現する具体的な $\boldsymbol{w}$ はわからないが、入出力のペアが複数与えられている

状況を考える。一つの入力 $\boldsymbol{x}$ に対する望ましい出力を $\boldsymbol{d}$ とすると、そのような入出力のペアが

$$\{(\boldsymbol{x}_1, \boldsymbol{d}_1), (\boldsymbol{x}_2, \boldsymbol{d}_2), ..., (\boldsymbol{x}_N, \boldsymbol{d}_N)\}$$

と N 個与えられている。これらのペア $(\boldsymbol{x}, \boldsymbol{d})$ 一つ一つを訓練サンプルと呼び、その集合を訓練データと呼ぶ。このとき $\boldsymbol{w}$ を調整することでこれらの入出力のペアを再現すること、すなわちどの $n(= 1, ..., N)$ のペア $(\boldsymbol{x}_n, \boldsymbol{d}_n)$ に対しても入力 $\boldsymbol{x}_n$ を順伝播型ニューラルネットワークに与えたときの出力 $\boldsymbol{y}(\boldsymbol{x}_n; \boldsymbol{w})$ がなるべく $\boldsymbol{d}_n$ に近くなるように $\boldsymbol{w}$ を調整する。これを学習と呼ぶ。このとき順伝播型ニューラルネットワークが表す関数と訓練データとの近さ $(\boldsymbol{y}(\boldsymbol{x}_n; \boldsymbol{w}) \approx \boldsymbol{d}_n)$ をどのように測るか、すなわちそれらの近さの尺度が重要となる。この尺度を表す関数を誤差関数と呼ぶ。誤差関数 $E(\boldsymbol{w})$ として二乗誤差

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \|\boldsymbol{d}_n - \boldsymbol{y}(\boldsymbol{x}_n; \boldsymbol{w})\|^2 \tag{8}$$

がよく用いられる。訓練データについて $E(\boldsymbol{w})$ を最も小さくするような $\boldsymbol{w}$ を選ぶことにより、順伝播型ニューラルネットワークで望みの関数を実現する。

2.1.4 確率的勾配降下法

ある順伝播型ニューラルネットワークの全てのパラメータを要素に持つベクトルを $\boldsymbol{w}$、誤差関数を $E(\boldsymbol{w})$ とすると、勾配 ∇E は

$$\nabla E \equiv \frac{\partial E}{\partial \boldsymbol{w}} = \left[\frac{\partial E}{\partial w_1} ... \frac{\partial E}{\partial w_M} \right]^T \tag{9}$$

と定義される。また、次の更新式

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \epsilon \nabla E \tag{10}$$

を定義する。ϵ は学習係数と呼ばれる定数 $(\epsilon > 0)$ であり、左向きの矢印は代入を表す。式 9、10 を反復計算することにより、パラメータ $\boldsymbol{w}$ について $E(\boldsymbol{w})$ の値を $E(\boldsymbol{w})$ の極小点の一つに向かって減少させることができる。これを勾配降下法と呼ぶ。

誤差関数 $E(\boldsymbol{w})$ は訓練サンプル 1 個だけについて計算される誤差 $E_n(\boldsymbol{w})$ の和として

$$E(\boldsymbol{w}) = \sum_{n=1}^{N} E_n(\boldsymbol{w}) \tag{11}$$

と表すことができる。ここで二つの式

$$\nabla E_n \equiv \frac{\partial E_n}{\partial \boldsymbol{w}} = \left[\frac{\partial E_n}{\partial w_1} ... \frac{\partial E_n}{\partial w_M} \right]^T \tag{12}$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \epsilon \nabla E_n \tag{13}$$

を反復毎に n を変更して反復計算することで $E(\boldsymbol{w})$ を最小化する方法がある。これを確率的勾配降下法と呼ぶ。確率的勾配降下法は訓練データに冗長性がある場合に最急降下法に比べて学習が速い、また反復毎に異なる誤差 E_n を計算するため、反復計算が望まない局所解にトラップされにくいという利点がある。

2.1.5 ミニバッチ

複数の訓練データの集合をミニバッチと呼ぶ。適切な要素数のミニバッチと確率的勾配降下法を合わせて用いると、確率的勾配降下法の利点を生かし、さらに並列計算により訓練データを効率良く処理することができる。ミニバッチを用いて確率的勾配降下法を計算するとき、ミニバッチを $D_t(t = 1, 2...)$ とすると

$$E_t = \frac{1}{|D_t|} \sum_{n \in D_t} E_n \tag{14}$$

と表される誤差 E_t を用いて二つの式

$$\nabla E_t \equiv \frac{\partial E_t}{\partial \boldsymbol{w}} = \left[\frac{\partial E_t}{\partial w_1} ... \frac{\partial E_t}{\partial w_M} \right]^T \tag{15}$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \epsilon \nabla E_t \tag{16}$$

を反復毎に異なる t を用いて反復計算する。

2.2 強化学習

本節の内容は [5] を参考にしている。

強化学習とは、相互作用から学習して目標を達成する問題の枠組みである。学習と意思決定を行う者はエージェントと呼ばれる。エージェントが相互作用を行う対象は環境と呼ばれる。両者は継続的に相互作用を行い、エージェントは行動を選択し、環境はその行動に応答し、エージェントに新しい状況を提示する。環境は報酬の発生源でもあり、この報酬はエージェントが時間の経過の中で最大化しようとする特別な数値である。

エージェントと環境は離散的な時間ステップ $t = 0, 1, 2, 3, ...$ の各々において相互作用を行う。各時間ステップ t において、エージェントは何らかの環境の状態の表現 $s_t \in \mathcal{S}$ ($\mathcal{S}$ は可能な状態の集合) を受け取り、これに基づいて行動 $a_t \in \mathcal{A}(s_t)$ を選択する ($\mathcal{A}(s_t)$ は状態 s_t において選択することのできる行動の集合である)。1 時間ステップ後に、エージェントはその行動の結果として数値化された報酬 $r_{t+1} \in \mathcal{R}$ を受け取り、新しい状態 s_{t+1} にいることを知る。

各時間ステップにおいて、状態から可能な行動を選択する確率の写像を方策と呼び、π_t で表す。ここで、$\pi_t(s, a)$ はもし $s_t = s$ ならば $a_t = a$ となる確率である。強化学習法の目的は、エージェントが方策を改善し、期待収益を最大化することである。時間ステップ t の後に受け取った報酬の系列を $r_{t+1}, r_{t+2}, r_{t+3}, ...$ と表すと、収益 R_t は

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + ... + r_T \tag{17}$$

と表される。ここで、T は最終時間ステップである。このアプローチはエージェントと環境の相互作用が、エピソードと呼ばれる部分系列に分解されるときに意味をなす。エピソードは終端状態と呼ばれる特殊な状態で終わり、この終端状態に続いて、標準的な開始状態へのリセットが行われる。この種のエピソードを伴うタスクはエピソード的タスクと呼ばれる。

2.2.1 マルコフ決定過程

時刻 t で取られた行動に対して、環境が時刻 $t+1$ においてどのように応答するかを考える。最も一般的な場合では、この応答は以前に起こったあらゆることに依存する。この場合、ダイナミクスは、全ての s', r と、過去の事象全ての可能な値 $s_t, a_t, r_t, ..., r_1, s_0, a_0$ に対して、完全な確率分布

$$P_r\{s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, ..., r_1, a_0, r_0\} \tag{18}$$

を指定することによってのみ定義される。

他方、状態信号がマルコフ性を持つなら、$t+1$ における環境の応答は t における状態と行動のみに依存することになり、このときには全ての s', r, s_t と a_t に対して

$$P_r\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\} \tag{19}$$

のみを指定することで環境のダイナミクスを定義することができる。

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、MDP と呼ばれる。状態と行動の空間が有限であるなら、それは有限 MDP と呼ばれる。

特定の有限 MDP は状態と行動の集合と、環境の 1 ステップダイナミクスから定義される。任意の状態と行動、s と a が与えられたとして、次に可能な各状態 s' の確率は

$$\mathcal{P}^a_{ss'} = P_r\{s_{t+1} = s' | s_t = s, a_t = a\} \tag{20}$$

となる。これらの量は遷移確率と呼ばれる。また、次の報酬の期待値は

$$\mathcal{R}^a_{ss'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\} \tag{21}$$

となる。

2.2.2 価値関数

ほとんど全ての強化学習アルゴリズムは価値関数に基づく評価を行っている。この関数は状態 (あるいは状態行動対) の関数で、期待収益に関して定義される。エージェントが将来受け取る報酬は、エージェントがどのような行動を取るかに依存するため、価値関数は特定の方策に関して定義される。

方策 π のもとでの状態 s の価値 $V^\pi(s)$ は、

$$V^\pi(s) = E_\pi\{R_t | s_t = s\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\} \tag{22}$$

となる。$E_\pi\{\}$ は、エージェントが π に従うとしたときの期待値を表す。終端状態の価値は常に 0 である。関数 V^π を方策 π に対する状態価値関数と呼ぶ。

方策 π のもとで状態 s において行動 a を取ることの価値 $Q^\pi(s, a)$ は、

$$Q^\pi(s, a) = E_\pi\{R_t | s_t = s, a_t = a\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\} \tag{23}$$

となる。Q^π を方策 π に対する行動価値関数と呼ぶ。

任意の方策 π と状態 s に対して、s の価値と可能な後続状態群の価値との間に整合性条件

$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^\pi(s')\right] \tag{24}$$

が成り立つ。式 (24) は V^π に対する Bellman 方程式である。この式は、開始状態の価値は、期待される次状態の価値と途中で得られる期待報酬との和に等しい、ということを述べている。

2.2.3 最適価値関数

価値関数は方策に関して半順序を定義する。全ての $s \in \mathcal{S}$ 対して、$V^\pi(s) \geq V^{\pi'}(s)$ であるなら、そのときに限り $\pi \geq \pi'$ である。他の方策より良いか、それに等しい方策が常に少なくとも一つ存在し、これが一つの最適方策 π^* である。最適方策が持つ状態価値関数は最適状態価値関数と呼ばれ、全ての $s \in \mathcal{S}$ に対して

$$V^*(s) = \max_\pi V^\pi(s) \tag{25}$$

と定義される。

最適方策は、Q^* と表される最適行動価値関数も持つ。最適行動価値関数は、全ての $s \in \mathcal{S}$ と $a \in \mathcal{A}(s)$ に対して

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) \tag{26}$$

と定義される。

V^* は方策に対する価値関数であるから、式 (24) の状態価値に対する Bellman 方程式で与えられる自己整合性条件を満たす必要がある。これは最適価値関数であるので、V^* の整合性条件は特定の方策を参照しない特別な形として

$$V^*(s) = \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^*(s')\right] \tag{27}$$

と書くことができる。これが V^* に対する Bellman 最適方程式である。Q^* に対する Bellman 最適方程式は、

$$Q^*(s,a) = \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^*(s',a')\right] \tag{28}$$

と表される。

2.2.4 方策改善

いま任意の決定論的な方策 π に対して、その価値関数 V^π が求まったとする。我々が知りたいことは、ある状態 s に対して、ある行動 $a \neq \pi(s)$ を決定論的に選択するように方策を変更すべきかという点である。新しい方策に変えることがより良いのか、悪いのかを知る一つの方法は、状態 s で行動 a を選択し、その後は既存の方策 π に従うことである。この価値は

$$\begin{aligned} Q^\pi(s,a) &= E_\pi \{r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s, a_t = a\} \\ &= \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^\pi(s')\right] \end{aligned} \tag{29}$$

と計算される。もしこの値が $V^{\pi}(s)$ より大きい場合、すなわち s で一度 a を選んで、その後 π に従うことが常に π に従う場合よりも良いならば、状態 s に対して行動 a を常に選ぶことが良く、したがってこの新しい方策は全体的に改善されることが期待される。ここで、全ての $s \in \mathcal{S}$ について

$$Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s) \tag{30}$$

が成り立つ決定論的な方策 π、π' が存在するとき、全ての $s \in \mathcal{S}$ について

$$V^{\pi'}(s) \geq V^{\pi}(s) \tag{31}$$

が成り立つ。これを方策改善定理と呼ぶ。元の決定論的方策を π、$\pi'(s) = a$ であること以外は π と同じである方策を π' とすると 、方策改善定理より、$\pi' \geq \pi$ が成り立つ。

全ての状態で、全ての可能な行動に関して、各状態で $Q^{\pi}(s, a)$ が最大となる行動をとる方策をグリーディ方策と呼ぶ。グリーディ方策 π'' は

$$\pi''(s) = \arg\max_{a} Q^{\pi}(s, a) \tag{32}$$

と定義される。π'' は方策改善定理により、$\pi'' \geq \pi$ であることが保証される。元の方策の価値関数が最大となる行動を選択していくことで、その方策を改善するような新しい方策を作り出す過程を方策改善と呼ぶ。

2.2.5 一般化方策反復

V^{π} を使って方策 π を改善し、より優れた方策 π' を得ることができたならば、続いて π' から $V^{\pi'}$ を計算して改善を行うことで、さらに優れた方策 π'' を得ることができる。このように方策と価値関数を次々と改善する系列

$$\pi_0 \rightarrow^{E} V^{\pi_0} \rightarrow^{I} \pi_1 \rightarrow^{E} V^{\pi_1} \rightarrow^{I} \pi_2 \rightarrow^{E} \ldots \rightarrow^{I} \pi^* \rightarrow^{E} V^* \tag{33}$$

を考えることができる。ここで $\rightarrow^{E}$ は方策に対する価値関数を計算すること (方策評価) を意味し、$\rightarrow^{I}$ は方策改善を表す。各方策は直前の方策に対し、厳密な改善となっていることが保証される。有限 MDP の方策は有限個であるから、この過程は有限回の繰り返しで最適価値関数に収束する。最適方策を発見するこのような手法を方策反復と呼ぶ。

方策反復では方策評価と方策改善が交互に行われたが、必ずしも二つの過程を完全、また交互に実行する必要はない。両方の過程が共に全ての状態について更新し続ける限り、どのように過程を適用してもたいがい最終的に価値関数と方策は最適価値関数と最適方策へ収束する。方策評価と方策改善の二つの過程の相互作用を一般化方策反復 (GPI) と呼ぶ。ほとんど全ての強化学習法は GPI として十分に説明できる。

2.2.6 方策オフ型モンテカルロ法

モンテカルロ法は状態の系列から得られる収益を価値の推定に用いて強化学習タスクを解く強化学習法の一種である。状態の系列を生成する方策を挙動方策と呼ぶ。また、挙動方策とは別の、評

価され改善される方策を推定方策と呼ぶ。このように制御に用いる方策と推定する方策を分ける学習制御手法を方策オフ型手法と呼ぶ。図 2 にテーブル形式の行動価値関数を用いた方策オフ型モンテカルロ法のアルゴリズムの一例を示す。

w は収益 R_t の重みで、挙動方策 π' に対する推定方策 π の元での系列の相対的生起確率であり、偏りのない収益の平均を求めるために用いられる。π' は $\pi(s,a) > 0$ ならば $\pi'(s,a) > 0$ となるような方策でなければならない。

全ての $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$ に対して初期化：
$Q(s,a) \leftarrow$ 任意の値
$N(s,a) \leftarrow 0$ ($Q(s,a)$ の分子)
$D(s,a) \leftarrow 0$ ($Q(s,a)$ の分母)
$\pi \leftarrow$ 任意の決定論的方策

永久に繰り返し：
(a) π' を選択し、それを用いて 1 個のエピソードを生成する：
$s_0, a_0, r_1, s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T, s_T$
(b) $\tau \leftarrow$ 最も最近 $a_\tau \neq \pi(s_\tau)$ となった時刻
(c) 時刻 τ あるいはそれ以降にエピソード中に出現した各 s, a の対について：
$t \leftarrow t \geq \tau$ であるような s, a が最初に発生した時刻
$w \leftarrow \Pi_{k=t+1}^{T-1} \frac{1}{\pi'(s_k,a_k)}$
$N(s,a) \leftarrow N(s,a) + wR_t$
$D(s,a) \leftarrow D(s,a) + w$
$Q(s,a) \leftarrow \frac{N(s,a)}{D(s,a)}$
(d) 各 $s \in \mathcal{S}$ について：
$\pi(s) \leftarrow \arg\max_a Q(s,a)$

図 2　方策オフ型モンテカルロ制御アルゴリズム

2.2.7 λ 収益

方策オフ型モンテカルロ法では収益 R_t を用いて価値の推定を行ったが、値

$$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+n}) \tag{34}$$

を価値の推定に使うこともできる。$R_t^{(n)}$ を n ステップ収益と呼ぶ。エピソードの終了時刻を T とすると、$T - t \leq n$ のときは $R_t^{(n)} = R_t$ となり、今まで通りの収益となる。

ある状態の推定価値をある目標値に近づける操作をバックアップと呼ぶ。バックアップでは推定価値の増分が計算され、増分を用いて推定価値が更新される。また、目標値を n ステップ収益としたバックアップを n ステップ・バックアップと呼ぶ。n ステップ・バックアップによる、$V_t(s_t)$ の増分 $\Delta V_t(s_t)$ は

$$\Delta V_t(s_t) = \alpha[R_t^{(n)} - V_t(s_t)] \tag{35}$$

と定義される。α は正の定数であり、ステップサイズ・パラメータと呼ばれる。また、全ての $s \neq s_t$ について $\Delta V_t(s) = 0$ である。推定価値の更新は全ての $s \in \mathcal{S}$ について

$$V_{t+1}(s) = V_t(s) + \Delta V_t(s) \tag{36}$$

と定義される。n ステップ・バックアップは推定価値関数を真の価値関数に近づけることが保証されている。

バックアップの目標値に n ステップ収益の平均値を用いることもできる。例えば、2 ステップ収益の半分と 4 ステップ収益の半分である、$\frac{1}{2}R_t^{(2)} + \frac{1}{2}R_t^{(4)}$ を用いてバックアップを行うことができる。要素となる n ステップ収益の重み値が正で、合計が 1 になっているのなら、いかなる n ステップ収益の集合も平均化できる。ここで、全ての n ステップ収益を平均化した収益は

$$R_t^{\lambda} = (1 - \lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)} + \lambda^{T-t-1} R_t \tag{37}$$

となる。これを λ 収益と呼ぶ。λ は重み値と呼ばれ、$0 \leq \lambda \leq 1$ である。λ 収益についても n ステップ・バックアップと同様に増分

$$\Delta V_t(s_t) = \alpha[R_t^{\lambda} - V_t(s_t)] \tag{38}$$

を計算し、式 36 を用いてバックアップを行うことができる。

2.2.8 Watkins の $Q(\lambda)$

Watkins の $Q(\lambda)$ は方策オフ型強化学習法の一種である。図 3 にテーブル形式の行動価値関数を用いた Watkins の $Q(\lambda)$ のアルゴリズムを示す。

$e(s, a)$ は適格度トレースと呼ばれる。各ステップにおいて、適格度トレースは訪問された 1 個の状態行動対に対して 1 だけ増加し、Q を更新後、全ての状態行動対に対して $\gamma\lambda$ だけ減衰する。ただし $a' \neq a^*$ となる場合は全ての状態行動対 s, a について $e(s, a) = 0$ となる。図 3 のアルゴリズムは適格度トレースを用いて逐次的に行動価値関数 Q を更新するが、エピソードが終了するまで更

$Q(s,a)$ を任意に初期化し、全ての s,a に対して $e(s,a)=0$ とする
各エピソードに対して繰り返し：
  s,a を初期化
  エピソードの各ステップに対して繰り返し：
    行動 a を取り、r,s' を観測する
    Q から導かれる方策 (例えば ϵ グリーディ方策) を用いて
      s' で取る行動 a' を選択する
    $a^* \leftarrow \arg\max_b Q(s',b)$
      (a' の場合と最大値が等しいならば、$a^* \leftarrow a'$)
    $\delta \leftarrow r + \gamma Q(s',a^*) - Q(s,a)$
    $e(s,a) \leftarrow e(s,a) + 1$
    全ての s,a について：
      $Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
      もし $a' = a^*$ ならば、$e(s,a) \leftarrow \gamma\lambda e(s,a)$
        それ以外 $e(s,a) \leftarrow 0$
    $s \leftarrow s'; a \leftarrow a'$
  s が終端状態であれば繰り返しを終了

図 3　Watkins の $Q(\lambda)$(テーブル形式版)

新を待ち、状態、行動及び報酬の系列が得られた後に λ 収益を用いてほぼ同様の更新を行うこともできる。エピソードの系列を $\{(s_0,a_0,r_1),(s_1,a_1,r_2),...,(s_{T-1},a_{T-1},r_T)\}$、状態 s_t での行動集合を $\mathcal{A}(s_t)$ とし、時刻 t の後、直近で $Q(s_\tau,a_\tau) \neq \max_{a\in\mathcal{A}(s_\tau)} Q(s_\tau,a)$ となった時刻を τ、存在しなければ $\tau = T$ とすると更新は

$$R_t^\tau = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^{\tau-t-2} r_{\tau-1} + \gamma^{\tau-t-1} r_\tau \tag{39}$$

$$R_t^{(n)} = \begin{cases} r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^{n-1} r_{t+n} + \gamma^n \max_{a\in\mathcal{A}(s_{t+n})} Q_{\boldsymbol{\theta}}(s_{t+n},a) & (\tau - t > n) \\ R_t^\tau & (\tau - t \leq n) \end{cases} \tag{40}$$

$$R_t^\lambda = (1-\lambda) \sum_{n=1}^{\tau-t-1} \lambda^{n-1} R_t^{(n)} + \lambda^{\tau-t-1} R_t^\tau \tag{41}$$

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[R_t^\lambda - Q(s_t,a_t)] \tag{42}$$

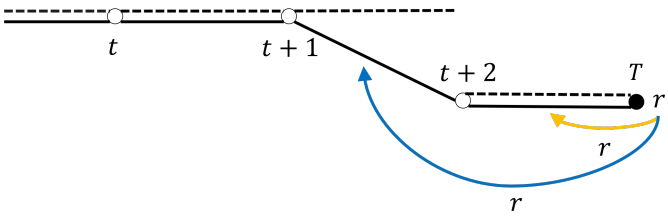
を計算することで行う。

2.2.9　Watkins の $Q(\lambda)$ と方策オフ型モンテカルロ法

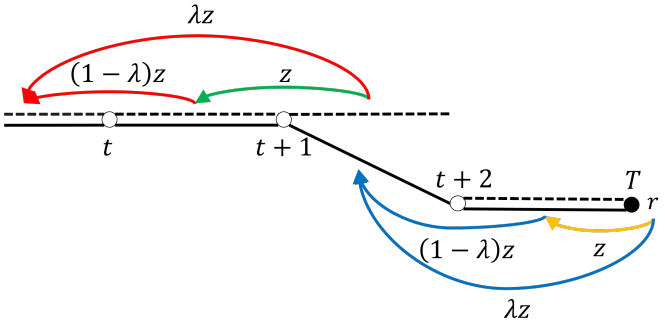
エピソード的なタスクにおいて行動価値関数をオフラインで更新するとき、Watkins の $Q(\lambda)$ と方策オフ型モンテカルロ法ではバックアップ方法に違いが存在する。Watkins の $Q(\lambda)$ ではエピソー

ド中に観測されたほぼ全ての状態行動対について行動価値関数を更新するのに対し、方策オフ型モンテカルロ法では最も最近、挙動方策と推定方策それぞれから導かれた行動が異なった時刻以降の状態行動対についてのみ行動価値関数を更新する。図 4 に Watkins の $Q(\lambda)$ と方策オフ型モンテカルロ法のバックアップの一例を示す。

図 4 は同じエピソードに対する Watkins の $Q(\lambda)$ と方策オフ型モンテカルロ法のバックアップ方法の違いを表している。時刻は t と表され、$0 \leq t \leq T$ である。図 4 の丸は状態 s を表し、白丸は非終端状態、黒丸は終端状態である。丸同士を繋ぐ線は行動 a を表し、点線は推定方策から導かれた行動、実線は挙動方策から導かれた行動である。このエピソードでは時刻 $t+1$ において挙動方策と推定方策それぞれから導かれた行動が異なっている。また状態 s_T に遷移した際に報酬 r が与えられており、それ以外の報酬は 0 である。矢印は矢印の根本の価値や報酬を用いた、矢印が指す状態行動対の行動価値に対するバックアップを表す。図 4(a) は方策オフ型モンテカルロ法のバックアップを表しており、矢印では図 2 中の式 $Q(s,a) \leftarrow \frac{N(s,a)}{D(s,a)}$ を計算する。図 4(b) は Watkins の $Q(\lambda)$ のバックアップを表しており、矢印では式 42 を計算する。Watkins の $Q(\lambda)$ では時刻 $t+1$ 以前の状態行動対についてバックアップを行うが、方策オフ型モンテカルロ法では行わない。このような違いが 2 手法の収束速度の差に関わってくると考えられる。



(a) 方策オフ型モンテカルロ法



(b) Watkins の $Q(\lambda)$

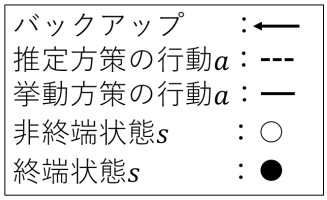


図 4　方策オフ型モンテカルロ法と Watkins の $Q(\lambda)$ のバックアップ

3　先行研究

本章では、本研究に関わる先行研究について述べる。

Multiple Choice System

Multiple Choice System は人工知能がゲームの候補手を複数提示し、ボスと呼ばれる人間がそれらの中から一つを選択するシステムである。Althöfer らは Multiple Choice System を提案し、チェスと囲碁における Multiple Choice System の性能を調査した [2]。

Althöfer らは 3-Hirn、Double-Fritz with Boss、List-3-Hirn と呼ばれる Multiple Choice System のバリエーションを構成した。3-Hirn は二つの独立したチェス人工知能がそれぞれ一つの候補手を提示し、ボスがその中から一つを選択するシステムである。Double-Fritz with Boss はチェス人工知能 Fritz に最善手と次善手の二つの候補手を提示させ、ボスがこれらの中から一つを選択するシステムである。List-3-Hirn は二つの独立したチェス人工知能がそれぞれ候補手を複数提示し、ボスがそれらの中から一つを選択するシステムである。Althöfer らはこれらのバリエーションを用いて対戦実験を行い、そのほとんどにおいて Multiple Choice System の棋力がベースとしたゲーム人工知能の棋力を上回る可能性を示した。

合議アルゴリズム

合議アルゴリズムは複数の独立した人工知能が個々に導いたゲームの候補手の中から、何らかの方法で一つの手を選択するアルゴリズムである。伊藤らは合議アルゴリズムを提案し、合議アルゴリズムの性能を調査した [3, 6]。

伊藤らは将棋人工知能の局面の評価値に乱数を加えることにより、一つの将棋人工知能から複数の将棋人工知能を擬似的に作る手法を提案した。伊藤らはこの手法により作成した複数の将棋人工知能の多数決や評価値最大の候補手を選択する合議アルゴリズムのシステムを構築し、このシステムが乱数を加えていない元の将棋人工知能及び別の独立した将棋人工知能 YSS に有意に勝ち越すことを示した。また伊藤らは、独立したそれぞれ異なる将棋人工知能、Bonanza、GPS 将棋、YSS の多数決合議アルゴリズムのシステムが、ベースとしたそれぞれの人工知能に有意に勝ち越すことを示した。

AlphaZero

AlphaZero はチェス、将棋、囲碁などを学習しプレイする人工知能である。AlphaZero はゲームのルール以外の知識を与えられることなく、自己対戦によりゲームでの勝利方法を学習する。Silver らは AlphaZero のアルゴリズムを提案し、AlphaZero の性能を調査した [7]。

AlphaZero はモンテカルロ木探索を用いて着手の吟味を行う。価値関数は状態の勝点期待値とし

てニューラルネットワークにより近似される。またニューラルネットワークは着手確率を出力し、これを用いて選択的に探索を行う。ニューラルネットワークは自己対戦により生成されたゲームプレイのデータを用いて学習を行い、勝点期待値と着手確率を改善する。Silver らはチェス、将棋、囲碁それぞれについて学習した三つの AlphaZero と人間のプロフェッショナルに勝る棋力を持つチェス人工知能 Stockfish、将棋人工知能 Elmo、囲碁人工知能 AlphaGo Zero との対戦実験を行い、各ゲームにおいて AlphaZero が勝ち越す (AlphaGo Zero については同等程度である) ことを示した。チェスにおいて AlphaZero は 8 手数前から現在までの駒配置、千日手、自色、手数、キャスリング可否、最後に駒を取ったまたは取られてからの手数の情報を符号化し、ニューラルネットワークの入力とした。

TD-Gammon

TD-Gammon はバックギャモンを学習しプレイする人工知能である。TD-Gammon はゲームのルール以外の知識を与えられることなく、自己対戦によりバックギャモンでの勝利方法を学習する。Tesauro は TD-Gammon のアルゴリズムを提案し、TD-Gammon の性能を調査した [8]。

TD-Gammon は強化学習法の一種である TD(λ) を用いてバックギャモンを学習する。価値関数は勝点期待値であり、ニューラルネットワークを用いて近似される。状態は事後状態と呼ばれる、行動後のバックギャモンの局面が用いられた。ニューラルネットワークは自己対戦により生成されたゲームプレイのデータを用いて学習を行い、勝点期待値を改善する。Tesauro は十分に学習した TD-Gammon がバックギャモンにおいてトップクラスのレーティングを持つ Rovertie とほぼ互角に対戦できることを示した。

菅原らの研究

菅原らは Multiple Choice System のバリエーションの一つである Double-Fritz with Boss をニューラルネットワークとチェス人工知能 Stockfish 7 を用いて構成し、その性能を調査した [9]。

菅原らは Stockfish 7 の自己対戦の結果をニューラルネットワークに予測させた。Stockfish 7 の自己対戦では次善手が低確率で選ばれ、最後に次善手が選ばれてから対局終了までのデータが訓練データとして用いられる。ニューラルネットワークの構成は複数試され、そのほとんどが単純な予測方法よりも良い予測精度を示した。また菅原らはニューラルネットワークの予測が良い候補手を選択するボスと、常に最善手を選択するボスを対局させたが、前者が後者に統計的に有意に勝ち越すことはないことを示した。

4　目的と方法

本章では本研究の目的及びそれを達成するための方法について述べる。

本研究の目的

本研究の目的は Multiple Choice System のボスを強化学習及びニューラルネットワークを用いた人工知能に置き換え、その性能を調査することである。強化学習法は基本的な方策オフ型強化学習手法であるモンテカルロ法と Watkins の $Q(\lambda)$ を用いる。ニューラルネットワークは AlphaZero でも用いられた畳み込み層を用いた構成を複数用いる。

強化学習問題の設定

ここでは本研究で設定した強化学習問題について述べる。

エージェントはボス人工知能であり、環境は相手プレイヤを含めたチェスのゲーム及びボス人工知能に候補手を提示するゲーム人工知能である。状態は自手番においてゲーム人工知能の出力から観測される情報とチェスのゲーム状況であり、行動は複数の候補手の中から一つを選択することである。報酬はゲーム終了時にのみ発生する勝点と呼ばれる値であり、価値関数は勝点期待値を表す。勝点は勝ちなら 1, 引き分けなら 0、負けなら −1 とする。一つのエピソードは一つのチェスのゲーム開始から終了までとする。ボス人工知能はチェスをプレイしながら種々の強化学習手法に基づいて一般化方策反復を行い、勝点期待値を最大にするような方策を探す。

チェスの駒配置の組み合わせはおよそ 10^{47} と言われており [*5]、ゲーム人工知能が全てのチェスのゲーム状況に対し決定論的に出力を行うとすると、本強化学習問題の状態集合及び行動集合は有限と考えられる。また状態遷移規則は有限 MDP により記述されると仮定し、種々の強化学習法を適用する。

4.1　λ 収益、関数近似を用いた Watkins の $Q(\lambda)$

2.2.8 で紹介した Watkins の $Q(\lambda)$ は行動価値関数がテーブル形式であり、チェスのゲーム状況を状態として扱うにはテーブル形式の行動価値関数ではメモリやデータ数、テーブルの計算時間などの観点から困難が生じる。そこで価値関数をニューラルネットワークを用いて関数近似する。また適格度トレースを用いた逐次的な更新方法はニューラルネットワークを用いて関数近似を行う場合実装が困難となるので、更新量が同じとなるような λ 収益を用いたオフライン更新形式に変更する。図 5 に関数近似、λ 収益を用いた Watkins の $Q(\lambda)$ のアルゴリズムを示す。

同様に 2.2.6 で紹介した方策オフ型モンテカルロ法もテーブル形式であるため行動価値関数を関数近似する。図 6 に関数近似を用いた方策オフ型モンテカルロ法のアルゴリズムを示す。

[*5] John's Chess Playground, https://tromp.github.io/chess/chess.html

$\lambda \leftarrow$ 任意の値 $(0 \leq \lambda \leq 1)$

$\boldsymbol{\theta} \leftarrow$ 任意のパラメータベクトル

$Q_{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}$ によって近似された行動価値関数

$\mathcal{D} \leftarrow \phi$

$\pi \leftarrow \arg\max_a Q_{\boldsymbol{\theta}}(\cdot, a)$

永久に繰り返し：

s_0 を初期化

ある挙動方策 (例えば ϵ グリーディ方策) を用いて 1 個のエピソードを生成する：

$s_0, a_0, r_1, s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T, s_T$

各時刻 $t(= 0, 1, ..., T-1)$ について繰り返し：

$\tau \leftarrow t < \tau$ かつ $Q_{\boldsymbol{\theta}}(s_\tau, a_\tau) \neq \max_{a \in \mathcal{A}(s_\tau)} Q_{\boldsymbol{\theta}}(s_\tau, a)$ となるような最小の τ、

存在しなければ終時刻 T

$R_t^\tau \leftarrow r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^{\tau-t-2} r_{\tau-1} + \gamma^{\tau-t-1} r_\tau$

$R_t^\lambda \leftarrow (1-\lambda) \sum_{n=1}^{\tau-t-1} \lambda^{n-1} R_t^{(n)} + \lambda^{\tau-t-1} R_t^\tau$

$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^{n-1} r_{t+n} + \gamma^n \max_{a \in \mathcal{A}(s_{t+n})} Q_{\boldsymbol{\theta}}(s_{t+n}, a)$

(ただし、$\tau - t \leq n$ ならば $R_t^{(n)} = R_t^\tau$)

$\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, R_t^\lambda)\}$

$\mathcal{D}$ の要素数が M 未満になるまで繰り返し：

$\mathcal{D}_p \leftarrow \mathcal{D}_p \subset \mathcal{D}$ かつ $\left|\mathcal{D}_p\right| = M$ となる集合

誤差関数 $\sum_{(s,a,R^\lambda) \in \mathcal{D}_p} E(s, a, R^\lambda, \theta)$ で θ を調整

$\mathcal{D} \leftarrow \mathcal{D} - \mathcal{D}_p$

$\pi \leftarrow \arg\max_a Q_{\boldsymbol{\theta}}(\cdot, a)$

図 5　λ 収益、関数近似を用いた Watkins の $Q(\lambda)$ のアルゴリズム

$\boldsymbol{\theta} \leftarrow$ 任意のパラメータベクトル
$\mathcal{D} \leftarrow \phi$
$\pi \leftarrow \arg\max_a Q_{\boldsymbol{\theta}}(\cdot, a)$

永久に繰り返し：
  s_0 を初期化
  ある挙動方策 (例えば ϵ グリーディ方策) を用いて 1 個のエピソードを生成する：
    $s_0, a_0, r_1, s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T, s_T$
  $\tau \leftarrow a_\tau \neq \pi(s_\tau)$ となるような最大の τ、存在しなければ 0
  各時刻 $t(= \tau, \tau+1, ..., T-1)$ について繰り返し：
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, R_t)\}$
  $\mathcal{D}$ の要素数が M 未満になるまで繰り返し：
    $\mathcal{D}_p \leftarrow \mathcal{D}_p \subset \mathcal{D}$ かつ $\left|\mathcal{D}_p\right| = M$ となる集合
    誤差関数 $\sum_{(s,a,R^\lambda)\in\mathcal{D}_p} E(s, a, R^\lambda, \theta)$ で θ を調整
    $\mathcal{D} \leftarrow \mathcal{D} - \mathcal{D}_p$
  $\pi \leftarrow \arg\max_a Q_{\boldsymbol{\theta}}(\cdot, a)$

図 6　関数近似を用いた方策オフ型モンテカルロ法のアルゴリズム

4.2 事後状態の符号化

チェスのゲーム状況及びゲーム人工知能の出力からなる状態は符号化により 8×8 の多チャネル画像として表現する。符号化する状態は TD-gammon でも用いられた事後状態と呼ばれる、その状態で行動をとった直後の状態とする。チェス盤は縦横 8 マスの正方形であり、画素がチェス盤のマスに対応する。あるチャネルにはある駒種の配置が符号化される。駒が存在するマスには 1、それ以外のマスには 0 の値が格納される。画像は常に手前側が自陣となるように適宜上下反転される。キャスリングの可否や評価値はチャネル毎に全ての画素に 0 や 1、0 から 1 の値を格納することにより表現する。図 7 に初期局面で一番左のポーンを前に 1 マス動かしたときの事後状態を符号化した画像の自陣ポーンを表すチャネルの例を示す。

AlphaZero が符号化した特徴を参考にし状態の特徴をいくつか符号化する。表 1 に本研究と AlphaZero の符号化する特徴とそのチャネル数を示す。

表 1(a)(b) いづれも左列が各特徴を表し、右列が各特徴のチャネル数を表す。AlphaZero でも用いられた現在の駒配置、過去の駒配置の系列、キャスリング可否の特徴を本研究でも簡易な形で符号化し更に相手着手予想、評価値などの本研究独自の特徴も符号化する。表 1(a) の現在の駒配置は着手後のチェス盤上の駒の配置であり、チャネル数は駒種 6 とプロモーションした駒種 4 で 10 枚、さらに自陣と相手陣で計 20 枚のチャネルで表現する。過去の駒配置の系列は相手の手番を含む 2 手数前までの駒配置で表される。相手着手予想はゲーム人工知能が予想する相手の次の着手後の駒配置で表される。キャスリング必要条件はキャスリングの必要条件であるキングとルークが動いていないことを表したものであり、全ての画素値が 0 か 1 のチャネルで表現される。チャネル数は自陣と相手陣のクイーンに近い側のルークとキング、キングに近い側のルークとキングが動いたかを表すチャネルの計 4 枚である。評価値はゲーム人工知能が出力する現在の着手に対する評価の値である。評価値は tanh 関数を用いて −1 から 1 の値に変換し、その値をチャネル全ての画素値とする。

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0

図 7　初期局面の自陣ポーンの符号化

表 1　符号化する特徴とチャネル数

(a) 本研究の符号化

特徴	チャネル数
現在の駒配置	20
過去の駒配置の系列	20×2
相手着手予想	20
キャスリング必要条件	4
評価値	1

(b) AlphaZero の符号化

特徴	チャネル数
現在の駒配置	12
過去の駒配置の系列	12×7
千日手	2×8
キャスリング可否	4
自色	1
手数	1
キャスリング可否	4
進捗無し手数	1

4.3　ニューラルネットワークの構成

価値関数をニューラルネットワークで近似することを考える。ニューラルネットワークは符号化された状態を受け取り、勝点期待値を出力する。本研究では AlphaZero でも用いられた畳込み層を用いる。表 2 に用いるニューラルネットワーク各層の構成を示す。

表 2 の左列はニューラルネットワークの各層を表し、右列は各層の詳細を表している。層は順に入力層、畳み込み層 A(N 個)、畳み込み層 B、全結合層と並ぶ。N 及び表 2 中の M, L は適宜変更する。N は畳込み層 A の数であり、$N = 1$ か $N = 7$ である。M は入力する特徴によって変更し、表 1(a) 中の現在の駒配置とキャスリング必要条件と評価値を入力する場合の $M = 25$、全ての特徴を入力する場合の $M = 85$ の 2 種類が存在する。L は畳込み層 A のフィルタ数であり、$L = 16$、$L = 32$、$L = 64$ のいずれかである。ニューラルネットワークの出力は最後の全結合層の活性化関数である tanh 関数によって -1 から 1 の値となる。

表 2　ニューラルネットワークの各層の構成

層	詳細
入力層	画像サイズ：$8 \times 8 \times M$
畳込み層 A	フィルタ数：L フィルタ縦横サイズ：3×3 ゼロパディング：1 活性化関数：ReLU
畳込み層 B	フィルタ数：1 フィルタ縦横サイズ：1×1 ゼロパディングなし 活性化関数なし
全結合層	ユニット数：1 活性化関数：tanh

5 実験

本章では本研究で行った実験について述べる。

共通の実験設定

ここではいづれの実験にも共通する実験設定について述べる。

候補手はオープンソースのチェス人工知能 Stockfish 8[*6]の MultiPV 機能を用いて生成する。MultiPV 機能は Stockfish 8 に任意の個数、次の着手を探索させる機能である。Stockfish 8 はチェスの駒配置などを入力として着手の探索を行う。探索節点数は 1 万とし、候補手数は 7 とする。

探索節点数を一定にした Stockfish 8 は一つ一つのチェスのゲーム状況に対して決定論的に着手を出力する。そのため探索節点数を一定にした Stockfish 8 同士の対局は全て同じ棋譜を生成する。Stockfish 8 を用いたボス人工知能が非決定論的な行動を行わなければボス人工知能についても同様である。そこで実際のチェスであり得るような様々な局面を生成するため、相手の着手の決定にはオープニングブックを用いる。オープニングブックは人間の対局で頻繁に用いられる、様々なチェスの着手が登録されているデータベースである。オープニングブックはあるチェスのゲーム状況に対応する着手とその着手の頻度が登録されている。オープニングブックを利用する際、そのときのチェスのゲーム状況に対して登録されている着手が複数存在するときは頻度に応じた確率で一つの着手を選ぶ。今回はインターネットサイト PGN Mentor[*7]から得られたチェスの上級者プレイヤ同士の対局の棋譜およそ 27 万局から 12 局以上で選択されている着手を元にオープニングブックを構成した。オープニングブックは Polyglot[*8]と呼ばれるチェスプログラムを用いて作成及び利用した。

Stockfish 8 は UCI protocol[*9]と呼ばれるプロトコルを用いてチェス環境を提供するサーバと通信を行う。今回サーバは Xboard[*10]を用いた。Xboard では連続対局が可能であり、各対局の後は先手と後手が入れ替わる。

ニューラルネットワークの実装にはオープンソースの深層学習フレームワークである Caffe[*11]を用いた。ニューラルネットワークの学習は確率的勾配降下法を用いて行った。ミニバッチ数は 30 とし、学習係数は一律で 0.01 とした。

挙動方策は ϵ グリーディ方策、推定方策はグリーディ方策を用いた。ϵ は実験により変更する。

300 手数以上続いたゲームは引き分けとした。

[*6] https://stockfishchess.org
[*7] https://www.pgnmentor.com/files.html
[*8] http://wbec-ridderkerk.nl/html/details1/PolyGlot.html
[*9] http://wbec-ridderkerk.nl/html/UCIProtocol.html
[*10] https://www.gnu.org/software/xboard/
[*11] http://caffe.berkeleyvision.org

5.1 実験 1

方策オフ型モンテカルロ法と Watkins の $Q(\lambda)$ のエピソード中の各手数における訓練サンプル採取数を調査した。訓練サンプル採取数はニューラルネットワークの訓練サンプルとして採取できた (s_t, a_t, r) の組の数である。ここで r は Watkins の $Q(\lambda)$ を用いるときは R_t^λ、方策オフ型モンテカルロ法を用いるときは R_t である。ϵ は 1 手目のみ 1.0 とし、以降は一律とした。対局相手はオープニングブックを利用し、オープンニングブックに着手が登録されていない場合は候補手の中から評価値最大のものを選択するボス人工知能とした。ニューラルネットワークは重み 1 個、バイアス 1 個の全結合層一つを用いた 1 入力 1 出力の単純なものとし、入力は Stockfish 8 が出力する評価値 1 個とした。重みの初期値は −1、バイアスは 0 とした。このニューラルネットワークを用いたボス人工知能は学習が進むと、常に評価値最大の候補手を最善手と認識するようになった。訓練サンプル採取数の計測はボス人工知能が常に評価値最大の候補手を最善手と認識するようになってから行った。図 8 に実験の結果を示す。

図 8 の横軸は初期状態からの手数、縦軸は訓練サンプル採取数を表す。図 8 中の青色、オレンジ色、灰色の点はそれぞれ ϵ をエピソード中一手目以外一律で 0.01、0.04、0.16 としたモンテカルロ法、黄色、水色、緑色の点はそれぞれ ϵ をエピソード中一手目以外一律で 0.01、0.04、0.16 とした Watkins の $Q(\lambda)$ を適用しているときの訓練サンプル採取数を表す。初期局面から対局を 5000 回行い、訓練サンプルを採取した。

図 8 の各点を各強化学習法について比較すると、序盤の訓練サンプル採取数に大きく違いがあることがわかる。手数 0 から 30 付近について、Watkins の $Q(\lambda)$ はいずれの ϵ についてもほとんど違いなく、比較的多くの訓練サンプルを採取できているが、方策オフ型モンテカルロ法は ϵ が増加するにつれ訓練サンプル採取数が低下している。Watkins の $Q(\lambda)$ の方が序盤の訓練サンプルを効率

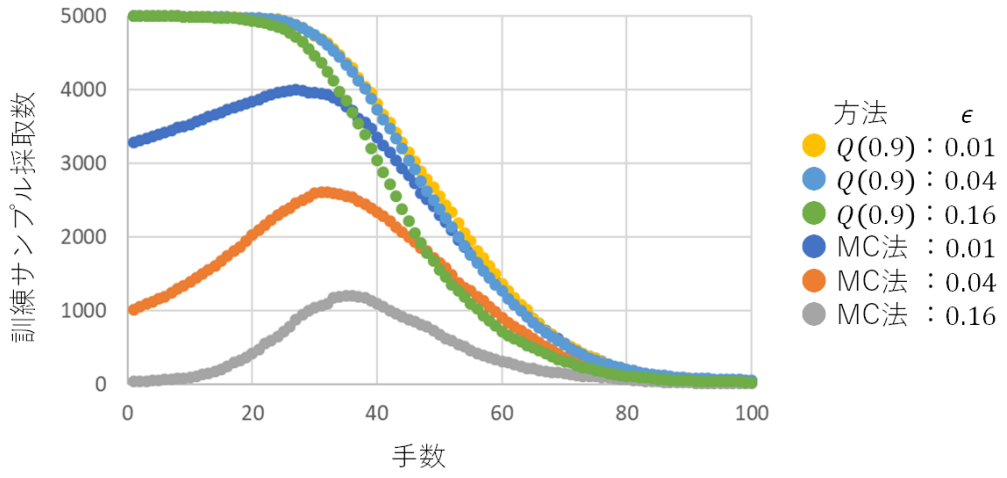


図 8　訓練サンプル採取数

よく採取でき、またいづれの ϵ の値でも多くの訓練サンプルを採取できることがわかる。すなわち Watkins の $Q(\lambda)$ は方策オフ型モンテカルロ法と比べて序盤の状態について効率良く学習することができると考えられる。Watkins の $Q(\lambda)$ は方策オフ型モンテカルロ法と比べて ϵ の値を多様に決めることができると考えられる。

5.2 実験 2

様々なニューラルネットワークの構成で、方策オフ型モンテカルロ法と Watkins の $Q(\lambda)$ を用いてボス人工知能に学習を行わせた。ニューラルネットワークの全ての重みは Xavier Initialization [10] を用いて初期化し、全てのバイアスは初期値を 0 とした。ϵ は一手目のみ $\epsilon = 1$ とし、以降は $\epsilon = 0.04$ とした。対局相手はオープニングブックを利用し、オープンニングブックに着手が登録されていない場合は確率 ϵ' で候補手の中から評価値最大のものを選択し、$1 - \epsilon'$ でランダム行動するボス人工知能とした。

対局開始後、重み更新回数が 10000 回になるまで $\epsilon' = 0.3$ とし、以降は $\epsilon' = 0$ とした。重み更新回数が 0 回から 100000 回の間は対局相手がオープニングブックを利用するのを止めた。ニューラルネットワークの重み更新を合計で 200000 回行った後 $\epsilon = 0$ として対局を 1000 回行い、学習を行ったボス人工知能の勝点平均を計算した。実験の結果を表 3 に示す。

表 3 は 5 つに分割されており、いずれも左列が学習に用いた強化学習法とニューラルネットワークの各構成、中央列が各構成の下での勝点平均とその 95% 信頼区間、右列がボス人工知能が評価値最大の候補手を選択した割合を表す。構成は (強化学習法)-(畳込み層数 N)-(入力画像のチャネル数 M)-(畳み込み層のフィルタ数 L) と各要素がハイフンで区切られて表される。評価値最大選択の構成は単に評価値最大の候補手を選択するような構成である。勝点平均を計測するために生成した各棋譜の重複を調べたところ平均して 5 割程度が重複していた。勝点平均はユニークな棋譜のみを用いて算出した。

一番勝点平均が高くなった構成は $Q(0.9)$-1-85-16 であり、ベースラインとなる評価値最大選択の構成の勝点平均を越えた。しかし統計的に有意な差はなく更なる検証が必要と考えられる。各強化学習法を比較すると $Q(0.9)$ が概ね一番勝点平均が高くなっていることがわかる。バックアップ方法が似ている $Q(1)$ と MC 法を比較すると、概ね $Q(1)$ の方が勝点平均が高くなっていることがわかる。すなわち序盤の訓練サンプルを多く採取できることはボス人工知能の性能の向上に関連があると考えられる。どの構成においても評価値最大を選択する割合は 0.5 以上であり、ボス人工知能は評価値が高いほど良いという基本的傾向を認識したと考えられる。勝点平均と評価値最大を選択する割合の比例関係は見受けられない。

畳込み層数、入力画像のチャネル数、畳み込み層のフィルタ数のいずれにも、比例して勝点平均が単調に増加もしくは減少する傾向は見受けられない。これには学習が最後まで進んでいない可能性が考えられる。学習が最後まで進んでいない原因の一つとして、重み数に対して訓練サンプル数が足りていないことが考えられる。表 4 に各ニューラルネットワークの構成の重み数を示す。

表 4 の左列は各ニューラルネットワークの構成、右列は各構成の重み数である。本実験の訓練サ

表 3　学習後勝点平均

構成	勝点平均	評価値最大選択割合
評価値最大選択	-0.05 ± 0.09	1.00

構成	勝点平均	評価値最大選択割合
$Q(0)$-1-25-16	-0.56 ± 0.09	0.49
$Q(0)$-1-25-32	-0.12 ± 0.08	0.54
$Q(0)$-1-25-64	-0.27 ± 0.08	0.60
$Q(0)$-1-85-16	-0.38 ± 0.08	0.55
$Q(0)$-1-85-32	-0.14 ± 0.08	0.54
$Q(0)$-1-85-64	-0.36 ± 0.09	0.56
$Q(0)$-7-25-16	-0.08 ± 0.08	0.55
$Q(0)$-7-25-32	-0.37 ± 0.08	0.57
$Q(0)$-7-25-64	-0.19 ± 0.07	0.58
$Q(0)$-7-85-16	-0.10 ± 0.08	0.60
$Q(0)$-7-85-32	-0.24 ± 0.09	0.57
$Q(0)$-7-85-64	-0.23 ± 0.08	0.56

構成	勝点平均	評価値最大選択割合
$Q(0.9)$-1-25-16	-0.06 ± 0.08	0.61
$Q(0.9)$-1-25-32	-0.09 ± 0.08	0.62
$Q(0.9)$-1-25-64	-0.03 ± 0.08	0.68
$Q(0.9)$-1-85-16	0.07 ± 0.08	0.74
$Q(0.9)$-1-85-32	0.00 ± 0.08	0.68
$Q(0.9)$-1-85-64	0.04 ± 0.09	0.70
$Q(0.9)$-7-25-16	-0.10 ± 0.09	0.57
$Q(0.9)$-7-25-32	-0.10 ± 0.08	0.53
$Q(0.9)$-7-25-64	-0.01 ± 0.10	0.53
$Q(0.9)$-7-85-16	-0.04 ± 0.09	0.61
$Q(0.9)$-7-85-32	-0.08 ± 0.08	0.62
$Q(0.9)$-7-85-64	0.01 ± 0.08	0.53

構成	勝点平均	評価値最大 選択割合
Q(1)-1-25-16	-0.15 ± 0.08	0.61
Q(1)-1-25-32	-0.11 ± 0.08	0.63
Q(1)-1-25-64	-0.05 ± 0.08	0.72
Q(1)-1-85-16	-0.01 ± 0.08	0.68
Q(1)-1-85-32	-0.04 ± 0.08	0.63
Q(1)-1-85-64	-0.11 ± 0.08	0.65
Q(1)-7-25-16	-0.14 ± 0.09	0.62
Q(1)-7-25-32	-0.14 ± 0.09	0.54
Q(1)-7-25-64	-0.17 ± 0.09	0.69
Q(1)-7-85-16	-0.11 ± 0.09	0.70
Q(1)-7-85-32	-0.17 ± 0.10	0.66
Q(1)-7-85-64	-0.01 ± 0.08	0.66

構成	勝点平均	評価値最大 選択割合
MC 法-1-25-16	-0.22 ± 0.08	0.69
MC 法-1-25-32	-0.20 ± 0.09	0.61
MC 法-1-25-64	-0.18 ± 0.08	0.64
MC 法-1-85-16	-0.30 ± 0.09	0.64
MC 法-1-85-32	-0.24 ± 0.08	0.66
MC 法-1-85-64	-0.20 ± 0.09	0.66
MC 法-7-25-16	-0.33 ± 0.08	0.63
MC 法-7-25-32	-0.24 ± 0.08	0.66
MC 法-7-25-64	-0.27 ± 0.08	0.62
MC 法-7-85-16	-0.40 ± 0.11	0.63
MC 法-7-85-32	-0.08 ± 0.08	0.63
MC 法-7-85-64	-0.35 ± 0.09	0.67

表 4　各ニューラルネットワークの構成の重み数

NN の構成	重み数
1-25-16	3680
1-25-32	7296
1-25-64	14528
1-85-16	12320
1-85-32	24576
1-85-64	49088
7-25-16	17504
7-25-32	62592
7-25-64	235712
7-85-16	26144
7-85-32	79872
7-85-64	270272

ンプル数は重み更新回数 200000 とミニバッチ数 30 から 600 百万個あったと考えられる。本実験のニューラルネットワークの最大の重み数は 27 万程度であり、その 10 倍以上であることから訓練サンプル数は十分であったと考えられる。しかしニューラルネットワークは一般に同じ訓練サンプルで何回も学習することが想定されているが [4]、本実験では一度重みの更新に利用した訓練サンプルは棄却するので、訓練サンプルを再利用するべきだった、もしくは重み更新回数が十分ではなかったなどの可能性が考えられる。また本実験の訓練サンプルは一回の対局から複数採取されており、訓練サンプル同士に相関があり独立でない。更に訓練サンプルは入手順に学習に利用されるため、ミニバッチの各要素のほとんどは他の要素に相関があると考えられる。このことは確率的勾配降下法による重み更新の量に偏りを生じさせる要因となると考えられる。訓練サンプルそれぞれまたはミニバッチの要素を独立にする処理が必要だった可能性が考えられる。また学習の収束には式

$$\sum_{k=1}^{\infty} \alpha_k = \infty \qquad \text{かつ} \qquad \sum_{k=1}^{\infty} \alpha_k^2 < \infty \tag{43}$$

が満たされなければならないことが強化学習の分野では知られている [5]。ここで α_k は重み更新回数 k のときの学習係数である。本実験では学習係数を固定としたので、式 43 の右の式が満たされない。学習の途中で学習係数を小さくする過程が必要だった可能性が考えられる。

6 総括

本研究では Multiple Choice System のボスを強化学習及びニューラルネットワークを用いた人工知能に置き換え、その性能を調査した。強化学習法は方策オフ型モンテカルロ法と Watkins の $Q(\lambda)$ を用い、また様々なニューラルネットワークの構成を用いた。$Q(0)$、$Q(0.9)$、$Q(1)$ 及び方策オフ型モンテカルロ法間では性能差が明らかに認められた。一番性能が良いボス人工知能を作成できた強化学習法は $Q(0.9)$ であり、一部のニューラルネットワークの構成との組み合わせの下で、単純に選択を行うボス人工知能の性能を上回っている可能性があるボス人工知能を作成できた。ニューラルネットワークの構成による性能の変化は明らかにならなかった。明らかにならなかった原因として学習が収束していないことが考えられ、その理由として重み更新回数が足りなかった、訓練サンプルを再利用すべきだった、訓練サンプルが独立ではなかった、学習係数を段階的に小さくしていく必要があった、などの事項が考えられる。

謝辞

本研究を進めるにあたり研究の方針、修士論文の推敲などの様々な点でアドバイス、ご指導を頂きました保木邦仁先生に深くお礼申し上げます。また輪読やゼミでご指摘をくださった西野順二先生及び西野研究室の皆様には深く感謝いたします。査読をして頂いた村松正和先生に深く感謝申し上げます。共に研究を進めていく上で協力して頂いた保木、村松、高橋研究室の生徒に感謝いたします。

参考文献

[1] 西垣通 (監修). 角川インターネット講座 (6) ユーザーがつくる知のかたち 集合知の深化. 角川学芸出版, 2015.

[2] Althöfer Ingo and Raymond G. Snatzke. Playing games with multiple choice systems. In *International Conference on Computers and Games*, pp. 142–153. Springer, 2002.

[3] 伊藤毅志, 小幡拓弥, 杉山卓弥, 保木邦仁. 将棋における合議アルゴリズム—多数決による手の選択. 情報処理学会論文誌, Vol. 51, No. 5, pp. 3030–3037, 2011.

[4] 岡谷貴之. 深層学習. 株式会社講談社, 2017.

[5] 三上貞芳, 皆川雅章. 強化学習. 森北出版株式会社, 2000.

[6] 杉山卓弥, 小幡拓弥, 斎藤博昭, 保木邦仁, 伊藤毅志ほか. 将棋における合議アルゴリズム—局面評価値に基づいた指し手の選択. 情報処理学会論文誌, Vol. 51, No. 11, pp. 2048–2054, 2010.

[7] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, Vol. 362, pp. 1140–1144, 2018.

[8] Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, Vol. 38, No. 3, pp. 58–68, 1995.

[9] 菅原真, 保木邦仁. ”double-fritz with boss”のボスをニューラルネットワークに置き換える研究. 情報処理学会研究報告, Vol. 2018-GI-39, No. 3, 2018.

[10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.