Vol.2019-GI-41 No.6
2019/3/8

# WatkinsのQ(λ)法に基づくMultiple Choice Systemのボスの強化学習

吉田 直人[1,a)]　保木 邦仁[1,b)]

**概要**：Multiple Choice System は囲碁やチェスなどの既存人工知能が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．本研究ではボスを強化学習及びニューラルネットワークを用いたボス人工知能に置き換え，その性能を調査する．題材としたゲームはチェス，既存人工知能は Stockfish 8 である．候補手は探索節点数 10000 の Stockfish 8 の MultiPV 機能を用いて生成した．MultiPV 機能とは指定した数の次の着手を探索する機能である．適切な強化学習法とニューラルネットワークの構成の下で，学習した対局相手に対するボス人工知能の勝点平均が，評価値最大の手を単純に選択し続けるボスの勝点平均よりも有意に高いことが示された．

# Reinforcement Learning of the Boss in Multiple Choice System Based on Watkins's Q(λ)

## 1. はじめに

現代社会において人工知能が活躍する機会が増えている．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 [*1]．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．人工知能分野が発展していくためには，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られる．

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おり，このような性質を持つ知能は集合知と呼ばれる．近年大きな成果を挙げているゲーム人工知能と集合知に関連する研究が複数存在しており，その例として Althöfer らによる Multiple Choice System の研究がある [1].

Multiple Choice System は囲碁やチェスなどの人工知能が候補手を提示し，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．図 1 に Multiple Choice System の概念図を示す．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チェス人工知能の Elo レーティングより高くなる可能性を示した．

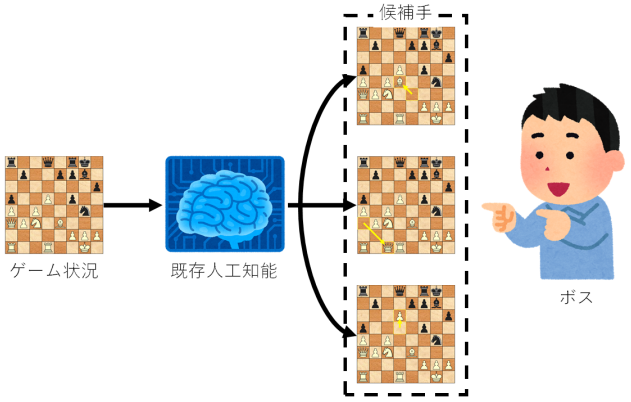


**図 1**　Multiple Choice System の概念図

本研究の目的は知識を適切に集約し意思決定を行うボス人工知能を，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ことを試みる．

[1] 電気通信大学
The University of Electro-Communications, Chofu, Tokyo, 182-8585, Japan
[a)] aerodyne0176@gmail.com
[b)] k.hoki@uec.ac.jp

[*1] http://www.asahi.com/digital/mediareport/TKY201003100152.html

## 2. 強化学習

本章では本研究と関連する強化学習の基礎知識を述べる．本章の内容は Sutton らの書籍を参考にしている [2].

### 2.1 エピソードと行動価値

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．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 $t = 0, 1, ..., T$ の各々において相互作用を行う．各 $t$ 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（$\mathcal{A}(s)$ は状態 $s \in \mathcal{S}$ において選択することのできる行動の集合）．各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 $r_{t+1} \in \mathcal{R}$（可能な報酬の集合）を受け取り，新しい状態 $s_{t+1}$ にいることを知る．$s_T$ を終端状態と呼ぶ．$T$ が有限であるような相互作用の系列をエピソードと呼ぶ．

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，$\pi$ で表す．ここで，$\pi(s, a)$ は条件 $s_t = s$ の下での $a_t = a$ 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

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

と表される．

いくつかの強化学習法は行動価値に基づく方策評価を行っている．行動価値は $Q^\pi(s, a)$ と表され，状態 $s$ で行動 $a$ をとった後，方策 $\pi$ に従うことで得られる期待収益である．実際の強化学習問題において行動価値が既知であることは少なく，これの推定を行わなければならない．

### 2.2 方策オフ型モンテカルロ法

方策オフ型モンテカルロ法は強化学習法の一種であり，状態の系列から得られる収益を行動価値の推定に用いて強化学習問題を解く．状態の系列を生成する方策を挙動方策と呼ぶ．また挙動方策とは異なる，評価され改善される決定論的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 $\pi'$ に対する推定方策 $\pi$ の下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．$\pi'$ は $\pi(s, a) > 0$ ならば $\pi'(s, a) > 0$ となるような方策であることが望ましい．

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) $\pi'$ 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 $\tau$ 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法のアルゴリズム（書籍 [2] より抜粋）

### 2.3 $\lambda$ 収益

方策オフ型モンテカルロ法では収益 $R_t$ を用いて行動価値の推定を行ったが，$n$ ステップ収益

$$R_t^{(n)} = r_{t+1} + r_{t+2} + r_{t+3} + ... + r_{t+n} + \max_{a \in \mathcal{A}(s_{t+n})} Q(s_{t+n}, a) \tag{2}$$

を $R_t$ の代わりに用いて推定を行うこともできる．ただし，$T - t \leq n$ のときは $R_t^{(n)} = R_t$ とする．

ある状態の推定行動価値を未来の系列から計算され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行動価値の増分が計算され，増分を用いて推定行動価値が更新される．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 $Q(s_t, a_t)$ の増分 $\Delta Q(s_t, a_t)$ は

$$\Delta Q(s_t, a_t) = \alpha[R_t^{(n)} - Q(s_t, a_t)] \tag{3}$$

と定義される．$\alpha$ 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．

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．$\lambda$ 収益

$$R_t^\lambda = (1 - \lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)} + \lambda^{T-t-1} R_t \tag{4}$$

$Q(s,a)$ 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 $\epsilon$ 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 $Q(\lambda)$ のアルゴリズム（書籍 [2] より抜粋）

は終端状態までの $n$ ステップ収益を平均化した値である．$\lambda$ は重み値と呼ばれ，$0 \le \lambda \le 1$ である．$\lambda$ 収益についても $n$ ステップ・バックアップと同様に増分

$$\Delta Q(s_t,a_t) = \alpha[R_t^\lambda - Q(s_t,a_t)] \tag{5}$$

を計算し，バックアップを行うことができる．

### 2.4　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 $\lambda$ 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，エピソードが終端するまで更新を待ち，状態，行動及び報酬の系列が得られた後に $\lambda$ 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 $(s_0,a_0,r_1),(s_1,a_1,r_2),...,(s_{T-1},a_{T-1},r_T)$ とし，時間ステップ $t$ の後，直近で $Q(s_\tau,a_\tau) \neq \max_{a\in\mathcal{A}(s_\tau)} Q(s_\tau,a)$ となった時間ステップを $\tau$，存在しなければ $\tau = T$ とすると更新は

$$R_t^\tau = r_{t+1} + r_{t+2} + ... + r_{\tau-1} + r_\tau \tag{6}$$

$$R_t^{(n)} = \begin{cases} r_{t+1} + r_{t+2} + ... + r_{t+n} \\ \quad + \max\limits_{a\in\mathcal{A}(s_{t+n})} Q(s_{t+n},a) & (\tau - t > n) \\ R_t^\tau & (\tau - t \le n) \end{cases} \tag{7}$$

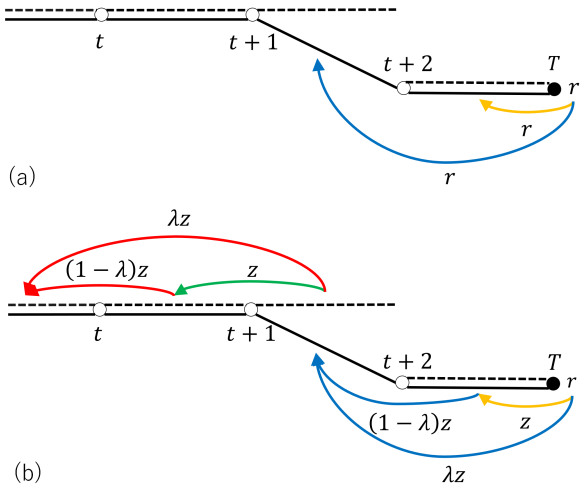


**図 4**　(a) 方策オフ型モンテカルロ法と (b)Watkins の $Q(\lambda)$ のバックアップ

$$R_t^\lambda = (1-\lambda)\sum_{n=1}^{\tau-t-1} \lambda^{n-1} R_t^{(n)} + \lambda^{\tau-t-1} R_t^\tau \tag{8}$$

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[R_t^\lambda - Q(s_t,a_t)] \tag{9}$$

を計算することで行われる．

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

エピソード的な強化学習問題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間ステップ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 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．白丸は非終端状態，黒丸は終端状態を表す．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間ステップ $t+1$ 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行動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 4a 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 4b は Watkins の $Q(\lambda)$ のバックアップを表しており，矢印では式 9 を計算する．Watkins の $Q(\lambda)$ では時間ステップ $t+1$ 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 手法の収束速度に影響を与えると考えられる．

## 3. 先行研究

Multiple Choice System はチェスや囲碁の人工知能が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 [1].

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．

合議アルゴリズムは複数の将棋などの人工知能が個々に導いたの候補手の中から，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], [4].

伊藤らはまた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も提案した．また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 の多数決が，ベースとなっ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 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 [5].

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も出力し，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を用いて学習を行い，推定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 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 [6].

TD-Gammon は強化学習法の一種である TD($\lambda$) 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，行動直後のバックギャモンのゲーム状況が用いられた．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 [7].

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ゲーム状況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という結果は得ることはなかった．

## 4. 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ボス人工知能に置き換え，その性能を調査することである．強化学習法は強化学習手法の一種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には AlphaZero でも用いられた畳み込み層からなる構成を用いる．

本研究の強化学習問題におけ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

$\lambda \leftarrow$ 任意の値 $(0 \le \lambda \le 1)$
$\boldsymbol{\theta} \leftarrow$ 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 $\epsilon$ 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 $\tau$，存在しなければ終時刻 $T$
$R_t^\tau \leftarrow r_{t+1} + \gamma r_{t+2} + ... + \gamma^{\tau-t-2} r_{\tau-1} + \gamma^{\tau-t-1} r_\tau$
$R_t^\lambda \leftarrow (1-\lambda) \sum_{n=1}^{\tau-t-1} \lambda^{n-1} R_t^{(n)} + \lambda^{\tau-t-1} R_t^\tau$
$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + ... + \gamma^{n-1} r_{t+n} + \gamma^n \max_{a \in \mathcal{A}(s_{t+n})} Q_{\boldsymbol{\theta}}(s_{t+n}, a)$
(ただし，$\tau - t \le n$ 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 $N$ 未満になるまで繰り返し：
$\mathcal{D}_p \leftarrow \mathcal{D}_p \subset \mathcal{D}$ かつ $\left|\mathcal{D}_p\right| = M\ (\le N)$ となる集合 $\mathcal{D}_p$
誤差関数 $\sum_{(s,a,R^\lambda) \in \mathcal{D}_p} E(s, a, R^\lambda, \theta)$ で $\theta$ を少しだけ更新
$\mathcal{D} \leftarrow \mathcal{D} - \mathcal{D}_p$
$\pi \leftarrow \arg\max_a Q_{\boldsymbol{\theta}}(\cdot, a)$

**図 5** 関数近似と $\lambda$ 収益を用いた Watkins の $Q(\lambda)$ のアルゴリズム

了時にのみ発生する勝点であり，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．ボス人工知能は種々の強化学習法に基づいて，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 [*2]，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マルコフ決定過程により記述されると仮定する．

## 5. 実験方法

本章では実験に用いた方法について述べる．

### 5.1 関数近似と $\lambda$ 収益を用いた Watkins の $Q(\lambda)$

2.4 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計算機のメモリやデータ数，テーブルの計算時間などの観点から困難が生じる．そこで行動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に $\lambda$ 収益を用いたオフライン更新形式に変更する．図 5 に関数近似と $\lambda$ 収益を用いた Watkins の $Q(\lambda)$ のアルゴリズムを示す．

$N$ はニューラルネットワークの訓練サンプルのプール数であり，$M$ はミニバッチ数である．$\mathcal{D}_p$ の要素は $\mathcal{D}$ の要素からランダムに選択する．$\mathcal{D}_p$ の要素を独立にするため，$N$ は大きな値が望ましい．

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

**表 1** 符号化する特徴とチャネル数

| 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 |

### 5.2 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 $(s, a)$ 直後のゲーム状況を表す事後状態とする．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，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．キャスリングの必要条件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．

AlphaZero が符号化した特徴を参考にし事後状態の特徴をいくつか符号化する．表 1 は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 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．相手着手予想は事後状態後に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．表 1 キャスリング必要条件とはキングとルークが動いていないことであり，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．

### 5.3 ニューラルネットワークの構成

行動価値関数をニューラルネットワークで近似すること

**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：8×8×85 |
| 畳込み層 A | フィルタ数：64<br>フィルタ縦横サイズ：3×3<br>ゼロパディング：1<br>活性化関数：ReLU |
| 畳込み層 B | フィルタ数：1<br>フィルタ縦横サイズ：1×1<br>ゼロパディングなし<br>活性化関数なし |
| 全結合層 | ユニット数：1<br>活性化関数：tanh |

を考える．ニューラルネットワークは符号化された事後状態を受け取り，勝点期待値を出力する．

表 2 にニューラルネットワークの各層の構成を示す．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．$N$ は畳込み層 A の数であり，$N = 3$ と $N = 7$ の 2 種類を用いる．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．

## 6. 実験

本章では本研究で行った実験とその結果について述べる．

### 6.1 実験設定

候補手はオープンソースのチェス人工知能 Stockfish 8[*3] 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．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．対局が同じ棋譜を生成することを避け，様々な対 $(s, a)$ を生成するために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に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．現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．本実験ではインターネットサイト PGN Mentor[*4] 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 3 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[*5] 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 や他のチェス思考エンジンの殆どは UCI protocol[*6] と呼ばれるプロトコルを用いてチェス対局環境を提供するサーバと通信を行う．今回サーバは Xboard[*7] 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[*8] 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．訓練サンプルのプール数は 3 万，ミニバッチ数は 30 とし，学習係数は 0.001 とした．

挙動方策は $\epsilon$ グリーディ方策，推定方策はグリーディ方策を用いた．$\epsilon$ 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．

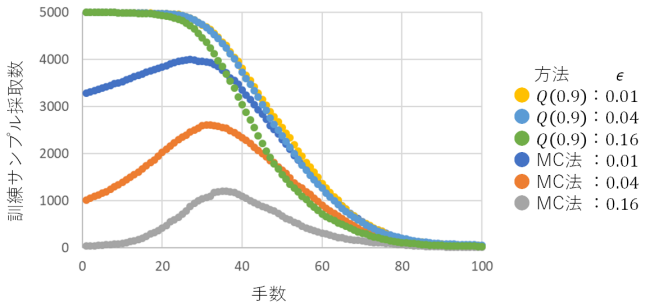


**図 6** 訓練サンプル採取数

### 6.2 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 $R_t^\lambda$，方策オフ型モンテカルロ法を用いるときは $R_t$ である．$\epsilon$ は 1 手目のみ 1.0 とした．対局相手はオープニングブックを利用し，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 6 に実験の結果を示す．

図 6 の横軸は初期状態からのボス人工知能の候補手選択数，縦軸は訓練サンプル採取数を表す．図 6 中の青色，オレンジ色，灰色の点はそれぞれ $\epsilon$ 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 $\epsilon$ 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．対局は 5000 回行い，

*3 https://stockfishchess.org
*4 https://www.pgnmentor.com/files.html
*5 http://wbec-ridderkerk.nl/html/details1/PolyGlot.html
*6 http://wbec-ridderkerk.nl/html/UCIProtocol.html
*7 https://www.gnu.org/software/xboard/
*8 http://caffe.berkeleyvision.org

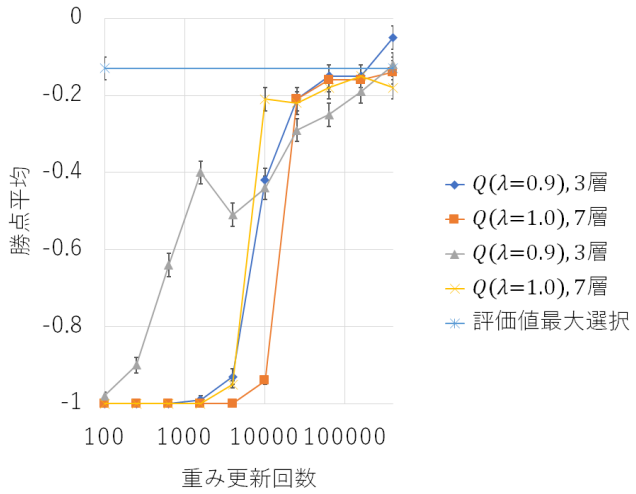


**図 7** 勝点平均

訓練サンプルを採取した．

図 6 の各強化学習法の比較から，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 $\epsilon$ 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 $\epsilon$ 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よく採取でき，またいづれの $\epsilon$ の値でも多くの訓練サンプルを採取できることが確認された．Watkins の $Q(\lambda)$ は方策オフ型モンテカルロ法と比べて多様な $\epsilon$ の値を利用可能であると考えられる．

### 6.3 実験 2

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重みは Xavier Initialization [8] を用いて初期化し，バイアスは初期値を 0 とした．$\epsilon$ 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，着手が登録されていない場合は $\epsilon'$ グリーディ方策のボスとした．

対局開始後，重み更新回数が 15000 回になるまで $\epsilon' = 0.3$ とし，以降は $\epsilon' = 0$ とした．また適当な重み更新回数毎に推定方策を用いてテスト対局を 5000 回行い，ボス人工知能の勝点平均を計算した．テスト対局の結果を表 7 に示す．

図 7 の横軸は重み更新回数であり，縦軸はテスト対局の勝点平均である．各点のエラーバーは 95% 信頼区間を表す．青色の点は畳込み層 A を 3 層，オレンジ色の点は 7 層として，$\lambda = 0.9$ としたボス人工知能の勝点平均を表す．灰色の点は畳込み層 A を 3 層，黄色の点は 7 層として，$\lambda = 1.0$ としたボス人工知能の勝点平均を表す．水色の点は評価値最大の候補手を単純に選択するボス人工知能の勝点平均を表す．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．勝点平均と 95% 信頼区間はユニークな棋譜のみからなる標本を用いて算出した．

図 7 から，畳込み層 A を 3 層，$\lambda = 0.9$ としたボス人工知能の勝点平均が最も高くなり，ベースラインとなる評価値最大を選択するボス人工知能の勝点平均より有意に高くなった．しかし他のボス人工知能はベースラインを有意に超えることはなかった．多くの $\lambda$ と畳み込み層 A 数の組合せにおいて，勝点平均は重み更新回数に対して上昇傾向であるため，更に重み更新を行うことで勝点平均が高くなる可能性がある．畳込み層 A の同一の数において $\lambda$ の値が異なる結果を比較すると，現状では $\lambda = 0.9$ の方が勝点平均が高くなる傾向が見られる．

## 7. 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て作成したボス人工知能に置き換え，その性能を調査した．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 $Q(\lambda)$ であり，複数のニューラルネットワークの構成を検討した．実験 1 では Watkins の $Q(\lambda)$ が方策オフ型モンテカルロ法よりも効率良く訓練データを採取することを示した．実験 2 では $\lambda = 0.9$，畳み込み層 A を 3 層としたボス人工知能の，学習した対局相手に対する勝点平均が評価値最大の候補手を選択するボスの勝点平均よりも有意に高いことを示した．学習アルゴリズムの繰り返しの数は十分ではないと考えられ，重み更新を更に行うことで勝点平均が上昇する可能性がある．

## 参考文献

[1] Althöfer, I. and Snatzke, G. R.: Playing games with multiple choice systems, *International Conference on Computers and Games*, Springer, pp. 142–153 (2002).
[2] Sutton, S. R., Barto, G. A., 三上貞芳，皆川雅章：強化学習，森北出版株式会社 (2000).
[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] 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).
[5] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I. et al.: A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play, *Science*, Vol. 362, pp. 1140–1144 (2018).
[6] Tesauro, G.: Temporal difference learning and TD-Gammon, *Communications of the ACM*, Vol. 38, No. 3, pp. 58–68 (1995).
[7] 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).
[8] Glorot, X. and Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks, *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256 (2010).