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	森 寛毅	学籍番号	2031141
論 文 題 目	4 人でプレイするキャントストップを題材とした強化学習の事例研究		

要　　旨

近年、将棋やチェスなどの二人零和ゲームにおいて、人間の強さを超えるような人工知能が強化学習を用いて開発されてきた。さらに、カードの排出順などの、プレイヤの意思決定とは無関係に決定されるような偶然的要因がある不確定多人数ゲームに関しても近年、手札などの非公開情報が在る不完全情報ゲームにおいて顕著な成果が報告されている。しかし、該当するようなゲームが現実社会においてはさほどプレイされていないためか、不確定性を伴う完全情報ゲームに関する研究は少ない。

本研究では不確定多人数完全情報ゲームの1つであるキャントストップを題材として、ゲームをプレイする人工知能をいくつかの手法で開発し、各手法の性能を比較する。各手法の性能を比較することで二人零和ゲームや一人ゲームで適用されてきた手法「TD(λ)」や「Q学習」が不確定多人数完全情報ゲームであるキャントストップにおいてどの程度有効であるか探る。適用・調査する強化学習に関連するいくつかの方法は、ε-グリーディ法、3層NNによる関数近似、オンポリシー型のTD(λ)、オフポリシー型のQ学習、経験リプレイ、先読み探索である。性能の測定には、簡易なルールを利用して実装したプレイヤ3人と対戦させ、その勝率を計測した。

実験結果から、次のことがわかった。NNの中間層は性能に大きく寄与しなかった。また、TD(λ)の方策ε-グリーディ法の探査の度合いを指定するパラメタεは0.2程度が適切であり、価値バックアップの深さを指定するパラメタλは0.8程度が適切であった。そして、TD(0)をオフポリシー型で行うQ学習の性能は、λの値が適切に設定されたTD(λ)には及ばなかった。さらに経験リプレイは、これを適用しない場合でも性能が劣化せず、登録されたデータの利用頻度を極端に大きくした場合性能が劣化した。最後に、先読み探索は訪問節点数が増えるほど性能が改善した。

電気通信大学大学院情報理工学研究科
修士論文

４人でプレイするキャントストップを題材とした強化学習の事例研究

2022 年 3 月 22 日

情報・ネットワーク工学専攻

学籍番号 2031141

森 寛毅

指導教員
保木 邦仁
高橋 里司

目 次

1　はじめに

近年、将棋やチェスなどの二人零和ゲームにおいて、人間の強さを超えるような人工知能が強化学習を用いて開発されてきた。さらに、カードの排出順などの、プレイヤの意思決定とは無関係に決定されるような偶然的要因がある不確定多人数ゲームに関しても近年、手札などの非公開情報が在る不完全情報ゲームにおいて顕著な成果が報告されている。特に目覚ましい成果を挙げた人工知能として、6 人制ポーカーの Pluribus [2] や 4 人麻雀の SuperPhenix [5] などが有名である。しかし、該当するようなゲームが現実社会においてはさほどプレイされていないためか、不確定性を伴う完全情報ゲームに関する研究は少ない。

本研究では不確定多人数完全情報ゲームの 1 つであるキャントストップを題材として、ゲームをプレイする人工知能をいくつかの手法で開発し、各手法の性能を比較する。各手法の性能を比較することで二人零和ゲームや一人ゲームで適用されてきた手法「TD(λ)」や「Q 学習」が不確定多人数完全情報ゲームであるキャントストップにおいてどの程度有効であるか探る。

本論文の構成は次のようである。1 章では、本研究の題材であるキャントストップのルールについて説明する。2 章では、本論文に必要な知識のうち NN と強化学習に関するものを説明する。3 章では、本研究の目的を基礎知識で説明した用語などを用いて具体的に説明する。4 章では、本研究で構成したゲーム木や、NN と強化学習の適用方法などの各種実験方法について説明する。5 章では、4 章の実験方法で述べた設定にそった実験結果を図にまとめてその結果を考察する。

2　キャントストップ

キャントストップ (CAN'T STOP) は 1980 年に Sid Sackson が考案したゲームである。このゲームは 2〜4 人でプレイするすごろくの一種であり、本研究では 4 人でプレイするキャントストップに着目する。このゲームはダイスを用いてプレイする不確定ゲームであり、どのプレイヤにも手札などの非公開情報が存在しないため完全情報ゲームでもある。この章ではまずキャントストップのルールについて述べる。

2.1 ルール

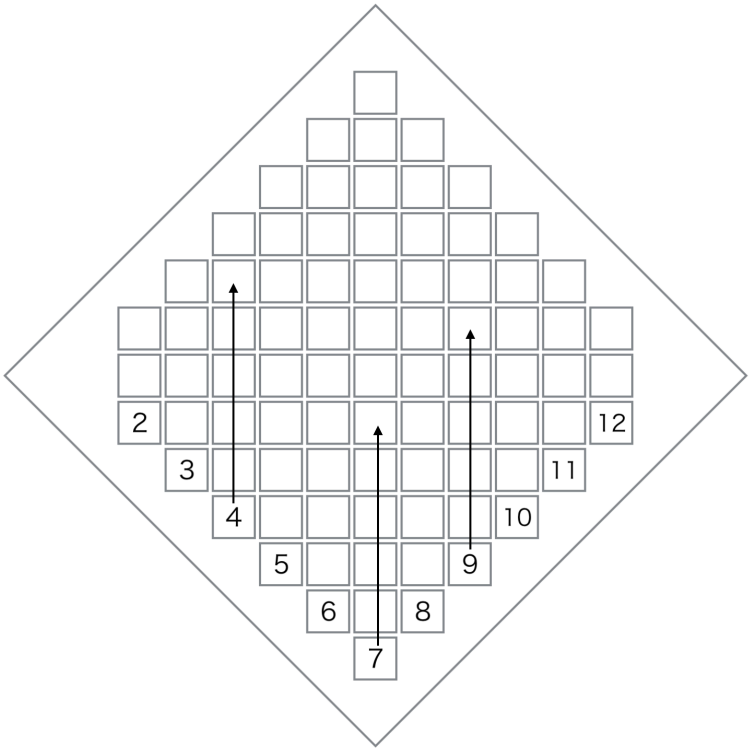

図 1: ゲーム盤

以下の 4 つの道具を用いてこのゲームはプレイされる。

- 2 から 12 までの番号が割り振られた 11 本のレーンが描かれたゲーム盤
- 4 つの 6 面ダイス
- 各プレイヤに 11 個ずつ与えられるマーカー
- 全プレイヤ共有の 3 個のポーン

ゲーム開始時は図 1 のように盤上にマーカーやポーンがない。任意のレーン（数字が割り振られた縦の列）のマーカーを最上部のマスに到達させたプレイヤがそのレーンを獲得する。ゲームの勝利条件は、ゲーム開始前に決定した本数だけレーンを獲得することである。以下のようなステップに従いゲームをプレイする。

1. ダイスを振るプレイヤの順番と開始プレイヤを決定する
2. 手番プレイヤが 4 つのダイスを振る

3. 4 つのダイスの出目を 2 つずつ 2 組に分け、それぞれの和を計算する。その組合せから得られる対の集合を C とする。たとえば 4 つの出目が 2,2,3,4 ならば、$C = \{(4,7),(7,4),(5,6),(6,5)\}$ となる。C から対を 1 つ選択することはポーンの操作を行うレーンの選択に対応する。まず対の先頭の要素に対応するレーンに対して以下のポーンの操作を行う。

   (a) そのレーンが、あるプレイヤに獲得されている。もしくはポーンがそのレーンのゴールの位置にある
      - 何もしない

   (b) (a) に該当せず、そのレーンにポーンが配置されている
      - そのポーンを進める

   (c) (a) か (b) に該当しない
      i. そのレーンに手番プレイヤのマーカーが配置されている
         - そのマーカーの 1 つ上のマスにポーンを配置する
      ii. (1) 以外
         A. 盤外にポーンがある
            - そのレーンのスタート地点にポーンを配置する
         B. 盤外にポーンがない
            - 何もしない

   次に対の末尾の要素に対応するレーンに対しても同様のポーンの操作を行う。ポーンを進められるもしくは配置できる対が C に 1 つも存在しない場合はステップ 6 に行く。そうでない場合、C の中からポーンを進められるもしくは配置できる対を 1 つ選びポーンを操作する。

4. ステップ 2 に行くことができる。この再度ダイスを振りポーンを進めることをストップと呼び、これがキャントストップの名前の由来である。

5. 盤内のポーンの位置全てに手番プレイヤの同レーンのマーカーを移動させる。ただし、手番プレイヤのマーカーが同レーンに存在しない場合は盤外のマーカーを新たにポーンの位置に配置する。

6. ポーンを盤上から除去する。

7. 手番プレイヤをステップ 1 で決定した順序に従い変更し、ステップ 2 に行く。

2.2 Keller の 28 ルール

キャントストップの戦略においてストップするタイミングが重要である。これを適切に決めるヒューリスティックな手法の 1 つとして Keller の 28 のルール[1]がある。この方法では、各レーンに対してポーンの配置及び前進をするたびに加点していき、レーン全ての点数が 28 以上になるまでストップしない。点数の初期値は 0 である。そして、プレイヤがレーン l を選択するたびに、次のように l に加点する。

- ポーンを置いた場合 $2 \times (1 + |7 - l|)$ 点増
- ポーンを進めた場合 $1 + |7 - l|$ 点増

さらに、レーン全ての点数は、レーンの点数の総和に次のように加点もしくは減点し得られる。ポーンが置かれたレーンが 3 つあり、それぞれどのレーン l も

- 奇数の場合 2 点増
- 偶数の場合 2 点減
- 7 以下の場合 4 点増
- 7 以上の場合 4 点増

このルールは、相手と比べて自分のマーカーの進みぐあいが非常に悪い場合やあと少しでレーンを占領できそうな場合では、適切ではないと Keller は述べている。

3 基礎知識

本章では、本研究で使用した強化学習の用語などを 3.1 節で、ニューラルネットの用語などを 3.2 節で、統計的検定を 3.3 節で説明する。

3.1 強化学習

本節では、本研究で使用した強化学習の用語などを書籍 [8] の内容を参考にして説明する。本研究では、キャントストップがほとんど確実に有限な時間ステップで終了するため、有限時間ステップで終了するタスクの強化学習に特化した説明を行う。また、強化学習の主要な構成要素である状態集合、行動集合および報酬集合は有限集合とする。

[1]Can't Stop? Try The Rule of 28: http://www.solitairelaboratory.com/cantstop.html (last access, 2020)

3.1.1 エピソードと報酬列

強化学習とは、エージェントが受け取る数値化された報酬信号を最大化するために、エージェントがどのように状況と行動を結びつけのが適切なのかを学習すること及び学習する方法を指す。強化学習問題は、エージェントにとって未知のマルコフ決定過程（MDP）の最適制御として定式化できる。学習と意思決定を行うエージェントと、それ以外の全てから構成される環境は相互作用を行う。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、生起する報酬の総和をエージェントは時間の経過の中で最大化しようとする。

エージェントと環境は離散的時間ステップ $t = 0, 1, 2, 3, \ldots$ 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 $s_t \in \mathcal{S}$（$\mathcal{S}$ は可能な状態の有限集合）を受け取り、これに基づいて行動 $a_t \in \mathcal{A}(s_t)$（$\mathcal{A}(s)$ は状態 s において選択することのできる行動の有限集合）を選択する。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 $r_{t+1} \in \mathcal{R} \subset \mathbb{R}$（$\mathcal{R}$ は報酬の有限集合）を受け取り、新しい状態 s_{t+1} にいることを知る。この相互作用により発生した、MDP が開始してから有限時間ステップで終了するまでの系列 $s_0, a_0, r_1, s_1, \cdots, s_{T-1}, a_{T-1}, r_T, s_T$ をエピソードと呼ぶ。

状態から可能な行動を選択する確率を与える関数 $\mathcal{S} \times \mathcal{A} \to [0, 1]$ をエージェントの確率的方策と呼ぶ（$\mathcal{A}$ は行動の有限集合）。任意の状態において、ある行動を確率 1 で取るような方策 $\mathcal{S} \to \mathcal{A}$ を決定論的な方策と呼ぶ。このような方策全体を有限集合 $\Pi^{\mathrm{D}} = \{\pi : \mathcal{S} \to \mathcal{A}\}$ で表す。

強化学習の目的は、エージェントが経験から方策を改善することで、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 $r_{t+1}, r_{t+2}, r_{t+3}, \ldots, r_T$ と表すと、収益 G_t は

$$G_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T \tag{1}$$

と表せる。ここで、T はこの系列の最終時間ステップである。ここで、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時系列に自然に分解されると仮定した。各エピソードは終端状態と呼ばれる特殊な状態で終わる。この終端状態に続いて、ある特定の開始状態あるいはある確率分布に従って生起した開始状態へのリセットが行われる。

3.1.2 MDP

時間ステップ t で取られた行動に対して、環境が時間ステップ $t + 1$ においてどのように応答するかを考える。MDP は確率過程の一種であり、エピソードはとある確率で生起

する。そして、この確率過程はマルコフ性をもち、$t+1$ における環境の応答は t における状態と行動表現のみに依存することになり、時間ステップ t の状態 s_t が状態 $s \in \mathcal{S}$ かつ行動 a_t が行動 $a \in \mathcal{A}$ であるとき、時間ステップ $t+1$ の状態 s_{t+1} が $s' \in \mathcal{S}$ かつ報酬 r_{t+1} が $r \in \mathcal{R}$ になる確率

$$\Pr\{s_{t+1} = s', r_{t+1} = r | s_t = s, a_t = a\} \tag{2}$$

を指定することで環境の 1 ステップダイナミクスを定めることができる。ただし条件付き確率 $\Pr\{A|B\}$ は、条件 B が成り立つとき条件 A が成り立つ確率である。

MDP は状態と行動の有限集合と、環境の 1 ステップダイナミクスから定義される。時間ステップ t の任意の状態 $s \in \mathcal{S}$ と任意の行動 $a \in \mathcal{A}$ から、次の時間ステップに可能な任意の状態 $s' \in \mathcal{S}$ への遷移確率 $\mathcal{P}^a_{ss'}$ は

$$\begin{aligned} \mathcal{P}^a_{ss'} &= \Pr\{s_{t+1} = s' | s_t = s, a_t = a\} & (3) \\ &= \sum_{r \in \mathcal{R}} \Pr\{s_{t+1} = s', r_{t+1} = r | s_t = s, a_t = a\} & (4) \end{aligned}$$

となる。同様に、次の時間ステップの報酬 r_{t+1} の期待値 $\mathcal{R}^a_{ss'}$ は

$$\begin{aligned} \mathcal{R}^a_{ss'} &= \mathbb{E}[r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'] & (5) \\ &= \sum_{r \in \mathcal{R}} r \Pr\{s_{t+1} = s', r_{t+1} = r | s_t = s, a_t = a\} & (6) \end{aligned}$$

である。ただし、$\mathbb{E}^\pi[x|A]$ は、条件 A が成り立ちエージェントが方策 π に従ったときの x の期待値である。

3.1.3 価値関数

価値関数は状態の関数で、エージェントがある状態にいることで将来得られる期待収益を評価する。エージェントが将来受け取ることを期待できる報酬は、エージェントの行動列に依存する。したがって、価値関数は特定の方策 π に関して定義される。有限 MDP の価値 $V^\pi(s)$ の形式的な定義は次のようになる。

$$\begin{aligned} V^\pi(s) &= \mathbb{E}^\pi[G_t | s_t = s] & (7) \\ &= \mathbb{E}^\pi\left[\sum_{k=0}^{T-t-1} r_{t+k+1} \middle| s_t = s\right] & (8) \end{aligned}$$

終端状態の価値は方策によらず 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。

同様に、方策 π のもとで状態 s において行動 a を取ることの価値を $Q^\pi(s,a)$ で表し、状態 s で行動 a をとり、その後に方策 π に従った期待報酬として定義する。

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}^\pi \left[G_t | s_t = s, a_t = a \right] && (9) \\ &= \mathbb{E}^\pi \left[\sum_{k=0}^{T-t-1} r_{t+k+1} \middle| s_t = s, a_t = a \right] && (10) \end{aligned}$$

3.1.4 最適価値関数

価値関数を使って確率的方策 π, π' に半順序を定義する。

$$\pi \geq \pi' \Longleftrightarrow \forall s \in \mathcal{S},\ V^\pi(s) \geq V^{\pi'}(s) \tag{11}$$

また、次の条件を満たす方策を最適方策と呼び、π^* と表す。

$$\pi^*\text{が最適方策} \Longleftrightarrow \forall \pi \in \Pi^{\mathrm{D}},\ \pi^* \geq \pi \tag{12}$$

最適状態価値関数は、次のように定義される。

$$V^*\text{が最適状態価値関数} \Longleftrightarrow \forall s \in \mathcal{S},\ V^*(s) = \max_{\pi \in \Pi^{\mathrm{D}}} V^\pi(s) \tag{13}$$

最適方策群は、Q^* で記される最適行動価値関数も共通して持つ。最適行動価値関数は、次のように定義される。

$$Q^*\text{が最適状態行動価値関数} \Longleftrightarrow \forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s),\ Q^*(s,a) = \max_{\pi \in \Pi^{\mathrm{D}}} Q^\pi(s,a) \tag{14}$$

3.1.5 方策改善・方策反復

ある方策の状態価値関数を求めることを方策評価という。いま、ある決定論的な方策 π に対して、方策評価を行なったとする。そして、ある状態 s のみに対して、ある行動 $a \neq \pi(s)$ を選択するようにその方策を変更するべきかを考える。新しい方策に変えることが良いのか知る 1 つの方法は、状態 s で行動 a を一度だけ選択し、その後は既存の方策 π に従うことである。この方策の下では状態行動対 (s,a) の価値は次のように計算される。

$$Q^\pi(s,a) = \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + V^\pi(s') \right] \tag{15}$$

この価値と変更する前の価値 $V^\pi(s)$ の比較が重要な判断基準を提供する。もし、$Q^\pi(s,a)$ の方が $V^\pi(s)$ より大きい場合、つまり s で一度 a を選んで、その後 π に従うことが、常に

π に従う場合より良いならば、状態 s に対して a を常に選ぶことが良い。したがって、変更前の方策は全体的に改善されるだろうと期待できる。これが成り立つことは方策改善定理と呼ばれる定理の特別な場合に相当する。

方策改善定理の特殊形

任意の決定論的な方策 $\pi, \pi' \in \Pi^{\mathrm{D}}$ に対して、

$$\forall s \in \mathcal{S},\ Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s) \tag{16}$$

ならば

$$\forall s \in \mathcal{S},\ V^{\pi'}(s) \geq V^{\pi}(s) \tag{17}$$

が成り立つ。

次の条件を満たす方策 π' を、π に関するグリーディ方策と呼ぶ。

$$\forall s \in \mathcal{S},\ \pi'(s) = \underset{a \in \mathcal{A}(s)}{\operatorname{argmax}}\, Q^{\pi}(s, a) \tag{18}$$

この方策は方策改善定理の条件を満たすため、元の方策と同等かそれ以上であることが保証される。元の方策の価値関数に従ってグリーディな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は方策改善と呼ばれている。

Q^{π} を使って方策 π を改善し、より優れた方策 π' を得る操作を繰り返し、最適方策を見つける手法を方策反復と呼ぶ。

$$\pi_0 \xrightarrow{E} Q^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} Q^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi^* \xrightarrow{E} Q^* \tag{19}$$

ここで $\xrightarrow{E}$ は方策評価を意味し、$\xrightarrow{I}$ は方策改善をあらわす。各方策は直前の方策に対して、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（$|\Pi^D| \leq |\mathcal{A}|^{|\mathcal{S}|}$）であるため、方策反復は有限回の繰り返しで最適価値関数に収束する。

3.1.6 方策評価法：関数近似を利用した TD(λ)

方策 π に従って状態 s で行動 a を行うことによって得られるエピソードの標本がいくつか与えられたとして、π のもとで状態行動対 (s, a) の価値 $Q^{\pi}(s, a)$ を推定したい。以降、エピソード中において状態行動対 (s, a) が発生することを (s, a) への訪問と呼ぶ。

モンテカルロ法では各訪問に対する収益がわかるまで待ち、その値を目標値として推定価値 $Q(s, a)$ を更新し精度を上げる。強化学習では、NN を使った関数近似と呼ばれる一般化手法がよく利用される。関数近似とは、目標関数から実例を採取し、その関数の近似

を作り出す手法である。価値関数をパラメータベクトル ω を用いた関数 $Q(s,a;\omega)$ で近似しモンテカルロ法を行う場合は、状態行動対 (s_t,a_t) の推定価値 $Q(s_t,a_t)$ が目標値 G_t に近づくようにパラメータを更新する。つまり、誤差関数

$$E(\omega) \quad = \quad \frac{1}{2}\left(G_t - Q(s_t,a_t;\omega)\right)^2 \tag{20}$$

の値が減少するようにパラメータを更新する。

TD(0) では時間ステップ $t+1$ の報酬 r_{t+1} と推定価値 $Q(s_{t+1},a_{t+1})$ を用いて、時間ステップ t の価値 $Q(s,a)$ の推定を行う。価値関数をパラメータベクトル ω を用いた関数 $Q(s,a;\omega)$ で近似し TD(0) を行う場合は、状態行動対 (s_t,a_t) の推定価値 $Q(s_t,a_t)$ が目標値 $r_{t+1}+Q(s_{t+1},a_{t+1})$ に近づくようにパラメータを更新する。つまり、次の誤差関数

$$E(\omega) \quad = \quad \frac{1}{2}\left(r_{t+1} + Q(s_{t+1},a_{t+1};\omega) - Q(s_t,a_t;\omega)\right)^2 \tag{21}$$

の値が減少するようにパラメータを更新する。

モンテカルロ法では最終時間ステップ T で得られる収益 G_t を用いて価値を推定した。一方、TD(0) の価値推定においては当該状態から 1 時間ステップ後の報酬 r_{t+1} だけを用い、それ以降の報酬和は 1 時間ステップ後の推定価値 $Q(s_{t+1},a_{t+1})$ で代用している。これらの手法を一般化し、n ステップ後までの報酬和と n ステップ後の推定価値を用いて価値を推定することもできる。一般化された目標値

$$G_t^{(n)} = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_{t+n} + Q(s_{t+n},a_{t+1}) \tag{22}$$

は n ステップ収益と呼ばれる。特に、$n \geq T-t$ のとき $G_t^{(n)} = G_t$ である。

推定価値の目標値に n ステップ収益の平均値を用いることもできる。ここで以下のような $G_t^{(1)}, G_t^{(2)}, \cdots, G_t^{(T-t)}$ の加重平均を定義する。

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{T-t-1} \lambda^{n-1} G_t^{(n)} + \lambda^{T-t-1} G_t^{(T-t)} \tag{23}$$

G_t^{λ} を λ 収益と呼ぶ。λ は減衰パラメータと呼ばれ、$0 \leq \lambda \leq 1$ であり、λ が 0 のとき λ 収益は TD(0) の目標値と一致し、1 のときモンテカルロ法の目標値と一致する。価値関数をパラメータベクトル ω を用いた関数 $Q(s,a;\omega)$ で近似し TD(λ) を行う場合、状態行動対 (s_t,a_t) の推定価値 $Q(s_t,a_t)$ が G_t^{λ} に近づくようにパラメータを更新する。つまり、$Q(s_t,a_t)$ と G_t^{λ} との誤差を利用し、誤差関数

$$E(\omega) \quad = \quad \frac{1}{2}\left(G_t^{\lambda} - Q(s_t,a_t;\omega)\right)^2 \tag{24}$$

の値が減少するようにパラメータを更新する。

3.1.7　状態行動空間の探査とオンポリシー型学習

状態行動空間における探査と知識利用はトレードオフの関係にある。知識利用だけを行うと、すなわち推定価値から得られたグリーディ方策に従いエピソードを生成すると、いつまでも訪問されず方策改善に寄与できない状態・行動対がありえて、最適方策付近の方策を見つけることに失敗する可能性がある。探査と知識利用のバランスをとる単純な方法として ε-グリーディ方策がある。ε-グリーディ方策は、確率 ε で選択可能な行動の中から一様ランダムに行動を選択し、確率 $1-\varepsilon$ でグリーディ方策に従う。

エージェントがエピソードを生成する際に従う方策は挙動方策と呼ばれ、各状態行動対の価値を推定する際に従う方策は推定方策と呼ばれる。TD 法では、推定価値 $Q(s,a;\omega)$ を挙動方策に従って得られた目標値に近づけるため、推定方策は挙動方策と一致する。このように挙動方策と推定方策が一致している強化学習の手法をオンポリシー型と呼び、挙動方策と推定方策が異なる手法はオフポリシー型と呼ぶ。

3.1.8　Q 学習

TD(0) をオフポリシー型に修正した手法が Q 学習である。Q 学習では、まず挙動方策 b に従ってエピソードを生成する。次に、時間ステップ $t+1$ で観測した報酬 r_{t+1} と推定価値の推定量の最大値 $\max_{a\in\mathcal{A}(s)} Q(s_{t+1},a)$ を用いて、時間ステップ t におけるグリーディ方策の価値の良い推定となるように $Q(s_t,a_t)$ を更新する。

ここで、推定方策に従った場合の価値関数を、パラメータベクトル ω を用いた関数 $Q(s,a;\omega)$ で近似し Q 学習を行う場合を考える。エピソードの生成には挙動方策 b を用い、状態・行動対 (s_t,a_t) を訪問して 1 時間ステップ後の状態 s_{t+1} と報酬 r_{t+1} が得られるたびに、次の誤差関数

$$E(\omega) \;=\; \frac{1}{2}\left(r_{t+1}+\max_{a\in\mathcal{A}(s_{t+1})} Q(s_{t+1},a;\omega)-Q(s_t,a_t;\omega)\right)^2 \tag{25}$$

の値が減少するようにパラメータを更新する。

3.1.9　事後状態

状態価値関数は通常、状態（エージェントが行動を選択するときに環境が提示する状況）の価値で方策を評価する。これに対し、エージェントがこなすタスクの性質によっては時として、事後状態（エージェントが行動を選択した直後に環境が提示する状況）を導入し、これの価値で方策を評価する方が都合が良い。

チェスのように行動した直後の盤面がわかるような場合で、このような事後状態をもって評価する手法は通常の手法に比べて有効である。なぜなら、チェスの場合では異なる盤面から異なる行動を選択した場合でも同じ盤面に到達することがありうるからである。この場合、状態・行動対が異なるため、行動価値関数もそれぞれ異なったものとして推定されなければならないが、事後状態を利用する場合は 1 つの事後状態の価値を推定するだけで良い。

3.2 ニューラルネットワーク

本節では、本研究で使用したニューラルネットワークの用語などを書籍 [11] の内容を参考にして説明する。

3.2.1 順伝播型 NN

ニューラルネットワーク (NN) は生物の神経回路網を模した数理モデルの 1 種である。回路網は神経細胞を模倣したユニットを結合し形成され、学習によりユニット間の信号強度を変化させることで問題解決を行う。NN は複数のユニットからなり、j 番目のユニットは 1 つの入力ベクトル x を受け取り、次の式で計算された 1 つの値 z_j を出力する。

$$u = \sum_i w_i x_i \tag{26}$$

$$z_j = f_j(u + b) \tag{27}$$

u は活性と呼ばれ入力の重み付き和である。これにバイアス b を加え、さらに活性化関数 f を作用させて出力を得る。

順伝播型 NN は、ユニットを層状に結合し形成されていて、入力信号は出力側へ一方向にのみ伝播する。入力側から出力側へ並べたときの先頭の層を入力層、終端の層を出力層、それ以外の層を中間層と呼ぶ。以降、それぞれの層で入力から出力を得る過程を述べる。

入力層は NN 全体への入力を複数受け取り、それらを各成分とするベクトルを出力する。$z^{(1)}$ を第 1 層の出力ベクトル、x を NN 全体への入力ベクトルとするとき、$z^{(1)}$ と x は次の条件を満たす。

$$z^{(1)} = x \tag{28}$$

中間層の各ユニットは、直前の層のユニットの出力を入力として受け取り、直後の層のユニットへ出力する。$w_{ji}^{(l)}$ を第 $l-1$ 層のユニット i と第 l 層のユニット j の間の結合の重

み、$W^{(l)}$ を第 l 層の重み行列、$u^{(l)}$ を第 l 層の活性ベクトル、$f^{(l)}$ を第 l 層の活性化関数、$z^{(l)}$ を第 l 層の出力ベクトルとすると、中間層で行う演算は以下のようになる。

$$u^{(l)} = W^{(l)} z^{(l-1)} \tag{29}$$

$$z^{(l)} = f^{(l)}(u^{(l)} + b^{(l)}) \tag{30}$$

ただし、

$$W^{(l)} = \begin{pmatrix} w_{11}^{(l)} & w_{12}^{(l)} & \cdots \\ w_{21}^{(l)} & w_{22}^{(l)} & \\ \vdots & & \ddots \end{pmatrix} \tag{31}$$

である。出力層は直前の中間層の出力を受け取り、NN 全体の出力である推定ベクトル $\hat{y}$ を出力する。$W^{(L)}$ を出力層の重み行列、$u^{(L)}$ を出力層の活性ベクトル、$f^{(L)}$ を出力層の活性化関数、$z^{(l)}$ を出力層の出力ベクトルとすると、出力層で行う演算は以下のようになる。

$$\hat{y} = z^{(L)} = f^{(L)}(u^{(L)} + b^{(L)}) \tag{32}$$

$$u^{(L)} = W^{(L)} z^{(L-1)} \tag{33}$$

3.2.2 学習の枠組み

順伝播型 NN は入力ベクトル x を受け取ると、$\hat{y}(x; W^{(2)}, \ldots, W^{(L)}, b^{(2)}, \ldots, b^{(L)})$ を出力する。以降重みやバイアスの値をまとめて ω と表記する。ここで入力ベクトルと出力の目標ベクトルとの対が複数与えられている状況を考える。このような対を訓練データ点と呼ぶ。NN の出力は ω を変えることにより変化するため、訓練データ集合 D になるべく適合するように ω を調整する。これを学習と呼ぶ。このとき、訓練データ点 x_n を NN に与えたときの出力 $\hat{y}(x_n; \omega)$ と目標ベクトル y_n がどれだけ近いかを測る尺度が必要となる。この尺度を表す関数を誤差関数と呼ぶ。誤差関数 $E(w)$ として、二乗誤差

$$E(\omega) = \sum_{(x,y)\in D} E_B(x, y, \omega) \tag{34}$$

$$E_B(x, y, \omega) = \frac{1}{2}(\hat{y}(x; \omega) - y)^2 \tag{35}$$

がよく用いられる。

L2 正則化は、NN のパラメータが訓練データ集合に対して過適合することを防ぐ手法の 1 つである。式 (34) に正則化項を加えることで NN のパラメータがスパース化しやす

くなり過適合を防ぐ。L2 正則化項を加えた二乗誤差は次のようである。

$$E(\omega) = \sum_{(x,y)\in D} E_B(x, y, \omega) + \lambda ||\omega||_2^2 \tag{36}$$

λ は重み減衰パラメータであり、$||\cdot||_2$ は L2 ノルムを表す。

3.2.3 確率的勾配降下法とその拡張

ある順伝播型 NN の M 個の要素をもつベクトルを ω、誤差関数を $E(\omega)$ とする。ω についての勾配 ∇E は

$$\nabla E = \left[\begin{array}{ccc} \frac{\partial E}{\partial \omega_1}, & \ldots, & \frac{\partial E}{\partial \omega_M} \end{array} \right]^{\mathrm{T}} \tag{37}$$

のようなベクトルとなる。式 (38) を用いて反復計算をすることにより、パラメータ ω について $E(\omega)$ の値を局所的極小解に向けて減少させることができる。これを勾配降下法と呼ぶ。

$$\omega_{\mathrm{new}} = \omega_{\mathrm{old}} - \eta \nabla E(\omega_{\mathrm{old}}) \tag{38}$$

$\eta\ (> 0)$ は学習率と呼ばれる定数である。

確率的勾配降下法 (SGD) は、式 (38) の右辺の勾配でランダムに選択した訓練データ集合の標本 1 つを使い、次のように ω を更新して $E(\omega)$ の極小値を求める方法である。

$$\nabla E_B = \left[\begin{array}{ccc} \frac{\partial E_B}{\partial \omega_1}, & \ldots, & \frac{\partial E_B}{\partial \omega_M} \end{array} \right]^{\mathrm{T}} \tag{39}$$

$$\omega_{\mathrm{new}} = \omega_{\mathrm{old}} - \eta \nabla E_B(x, y, \omega_{\mathrm{old}}) \tag{40}$$

SGD では、ランダムに訓練データ点を選択することが望まれる。これにより、パラメータ ω が望ましくない停留点にとどまり続ける確率が下がる。また、訓練データ集合に似たような訓練データ点が含まれている場合、1 回の更新ステップで似たような訓練データを重複して使用する無駄がなくなる。

モーメンタム法は勾配降下法における降下方向の振動を抑制する手法であり、過去の勾配と現在の勾配の加重平均をとることで急激に勾配が変化することを防ぐ。更新を繰り返して得られるパラメータ ω の系列を $\omega^{(0)}, \omega^{(1)}), \ldots$ とすると、t 回目の更新で得るパラメータは

$$\omega^{(t)} = \omega^{(t-1)} + \Delta\omega^{(t)} \tag{41}$$

$$\Delta\omega^{(t)} = \rho\Delta\omega^{(t-1)} + (1-\rho)\eta\nabla E(\omega) \tag{42}$$

である。ただし、$0 \leq \rho \leq 1$、$\Delta\omega^{(0)} = \vec{0}$ である。

3.2.4 誤差逆伝播法

誤差逆伝播法とは、勾配降下法の更新式に用いる勾配 ∇E を効率よく求めるための手法である。誤差関数 E の重み $\omega_{ji}^{(l)}$ についての偏微分 $\frac{\partial E}{\partial \omega_{ji}^{(l)}}$ は、第 l 層の j 番目のユニットの活性 $u_j^{(l)}$ を用いて連鎖律から

$$\frac{\partial E_n}{\partial \omega_{ji}^{(l)}} = \frac{\partial E_n}{\partial u_j^{(l)}} \frac{\partial u_j^{(l)}}{\partial \omega_{ji}^{(l)}} \tag{43}$$

$$= \frac{\partial E_n}{\partial u_j^{(l)}} z_i^{(l-1)} \left(\because u_j^{(l)} = \sum_i \omega_{ji}^{(l)} z_i^{(l-1)} \right) \tag{44}$$

$$= \delta_j^{(l)} z_i^{(l-1)} \tag{45}$$

と表せる。ただし、$\delta_j^{(l)} = \frac{\partial E_n}{\partial u_j^{(l)}}$ である。$z_i^{(l-1)}$ は順伝播によって求まる値であるため、$\frac{\partial E}{\partial \omega_{ji}^{(l)}}$ を効率よく求めるためには、$\delta_j^{(l)}$ を効率よく求めれば良い。$\delta_j^{(l)}$ は第 $l+1$ 層の k 番目のユニットの活性 $u_k^{(l+1)}$ を用いて連鎖律から

$$\delta_j^{(l)} = \frac{\partial E}{\partial u_j^{(l)}} = \sum_k \frac{\partial E}{\partial u_k^{(l+1)}} \frac{\partial u_k^{(l+1)}}{\partial u_j^{(l)}} \tag{46}$$

$$= \sum_k \delta_k^{(l+1)} \omega_{kj}^{(l+1)} f'(u_j^{(l)}) \left(\because u_k^{(l+1)} = \sum_{j'} \omega_{kj'}^{(l+1)} f(u_{j'}^{(l)}) \right) \tag{47}$$

と表すことができる。

訓練データ点を入力して順伝播させた後、出力層から入力層に向けて $\delta_j^{(l)}$ を順に求める。さらに、式 (45) を用いて $\frac{\partial E}{\partial \omega_{ji}^{(l)}}$ を求めることができる。

3.3 標本平均の信頼区間

本節では、書籍 [1] と [13] の定義を用いて標本平均の信頼区間の説明を行う平均が μ の確率分布に従い生起する値 x を n 回観測し求まる標本平均 X_n は、確率分布 $f(x)$ に従い得られるとしよう。ここで値 $x_1, x_2, \ldots, x_n$ が得られたとき、各観測結果の標本平均 X_n と標本分散 σ_n^2 は

$$X_n = (x_1 + x_2 + \cdots + x_n)/n \tag{48}$$

$$\sigma_n^2 = ((x_1 - X_n)^2 + (x_2 - X_n)^2 + \cdots + (x_n - X_n)^2)/n \tag{49}$$

である。X_n の標準偏差は

$$s(X_n) = (\sigma^2(x_1)/n^2 + \sigma^2(x_2)/n^2 + \cdots + \sigma^2(x_n)/n^2)^{1/2} \tag{50}$$

$$= \sigma^2(x)/n^{1/2} \tag{51}$$

として得られ、これを標準誤差と呼ぶ。ただし、$\sigma(x)$ は x の標準偏差を表す。

2 つの統計量

$$\hat{\mu_U}(x_1, x_2, \ldots, x_n), \hat{\mu_L}(x_1, x_2, \ldots, x_n)$$

に対し、

$$P(\hat{\mu_L} < \mu < \hat{\mu_U}) = 0.95$$

となるならば、区間 $(\hat{\mu_U}(x_1, x_2, \ldots, x_n), \hat{\mu_L}(x_1, x_2, \ldots, x_n))$ を信頼係数 0.95 の信頼区間と呼ぶ。

特に標本平均の分布が正規分布かつ分散 σ^2 が既知のとき、信頼係数 0.95 の信頼区間 $(\hat{\mu_L}, \hat{\mu_U})$ は

$$\hat{\mu_L} = X_n - u(0.05/2)\frac{\sigma}{\sqrt{n}} \tag{52}$$

$$\hat{\mu_U} = X_n + u(0.05/2)\frac{\sigma}{\sqrt{n}} \tag{53}$$

で与えられる。ただし、$u(\varepsilon/2)$ は正規分布 $N(0,1)$ の上側 $\varepsilon/2$ 点である。

信頼区間の幅の見積もりは観測回数 n が十分大きいとき、標準偏差 σ の代わりに観測値の標準誤差 s を用いて精度良く近似できる。そして n が十分大きくないと、標本平均の分布 $f(x)$ を正規分布で精度良く近似できず、s も σ を精度良く推定できない。n がどの程度ならば十分に大きいと見做せるかは一般的には分からないが、本論文では $n \geq 32$ ならば区間の幅が有効数字 1 桁の精度で近似できると仮定する。

求めたい確率変数が観測結果から推定できる 2 つの確率変数の差で表せるとき、その標準誤差を考える。それぞれ n 回の測定結果から推定できる確率変数を x, y、求めたい確率変数を

$$z = x - y$$

とする。このとき、確率変数 z の平均値は

$$Z_n = X_n - Y_n$$

となる。

Z_n の標準誤差は

$$s(Z_n) = \left[s(X_n)^2 + s(Y_n)^2\right]^{1/2} \leq s(X_n) + s(Y_n)$$

となる。よって、2 つの確率変数それぞれの信頼区間をエラーバーを用いて表したとき、2 つのエラーバーの範囲に重複がなければ 95%の確率で 2 つの確率変数の期待値は一致しない。

4　先行研究

表 1: いくつかの先行研究との比較

	バックギャモン	Atari 2600	囲碁など	キャントストップ
文献	Tesauro [10] [9]	Minh ら [6]	Silver ら [7]	本研究
不確実性	✓	✓		✓
プレイヤ	2 人	1 人	2 人	4 人
行動集合	10^1	10^1	10^2	10^1
行動列	10^2	10^3	10^2	10^2
方策	オン	オフ	オン	オン・オフ両方
バックアップ	TD(λ)	TD(0)	TD(1)	TD(λ)
事後状態	✓			✓
経験リプレイ		✓	✓	✓

表 1 に、顕著な成果が報告されているいくつかの強化学習法の適用事例をまとめる。本章では、これらの事例と本研究との類似点と相違点を説明する。

表中でキャントストップと最も類似するゲームはバックギャモンであろう。まず、どちらも双六のようなゲームであり、不確実性がダイスの目によってもたらされるという点が同じである。つぎに、行動集合の大きさやエピソード長も同程度である。そして、顕著に異なる点はプレイ人数である。

文献 [10] で用いられた強化学習法はオンポリシー型であり、TD(λ) による価値のバックアップと NN による価値の関数近似を行う。価値関数は、事後状態の価値を推定するものである。NN の規模は近年の深層ニューラルネットワーク（NN）より小さくて、入力層が 198 ユニット、中間層が 40〜160 ユニット、出力層が 4 ユニットの 3 層 NN である。入力は盤面のコマの配置を符号化した数値列である。挙動方策はグリーディ法である。

このようにして作られた TD-Gammon は、価値推定と 2 人ゲームの先読み探索を組合せることで、人間の熟練プレイヤに匹敵する強さを獲得した。TD-Gammon 3.1 の性能を測るにあたってバックギャモンのグランドマスターレベルの実力者と対戦をさせた結果、彼らと遜色ない強さを持っていることが報告されている。

Atari 2600 は複数のビデオゲームのタイトルからなるコレクションである。行動集合の大きさは同程度ではあるが、キャントストップよりもエピソード長は長い。文献 [6] で提案された手法「DQN」は、Q 学習を発展させたものである。深層 NN で行動価値関数を

近似し、ビデオゲームの画面のピクセルの輝度を入力とする。挙動方策はε-グリーディ法である。強化学習中になされるNNの更新を安定化させるために経験リプレイを利用し、同文献によりこれの有効性が広く知られることとなった。経験リプレイは、プレイヤが経験した状態や行動をリプレイメモリに一定数貯めて、NNのパラメタ更新に用いる訓練データ点の並びを乱雑にする手法である。ターゲットネットワークは、価値の更新に用いる目標値を計算するNNの更新を遅延させ、学習を安定化させる手法である。

同文献の実験では、DQNで構成された人工知能がAtari 2600内のいくつかのゲームで人間の最高スコアを上回った。

5 目的

4人不確定ゲームであるキャントストップのゲームプレイにおいて、二人ゲームを題材とした先行研究で用いられている先読み探索法と強化学習法を適用して人工知能を構築し、性能を調査する。適用・調査する強化学習に関連するいくつかの方法は、ε-グリーディ法、3層NN(中間層1層)による関数近似、オンポリシー型のTD(λ)、オフポリシー型のQ学習、経験リプレイ、先読み探索である。

ε-グリーディ法の調査では、実験で採用したキャントストップの強化学習の枠組みにおいて適切なεの値を探る。オンポリシー型の価値のバックアップでは、εの値が大きすぎると価値の更新の際にランダムに選択された行動の影響を強く受け、価値の推定精度は悪くなる。

オンポリシー型の学習に加えて、本研究ではオフポリシー型のQ学習も用いる。オフポリシー型の価値のバックアップでは、εの値による価値の更新の影響はオンポリシー型より比較的小さいと思われる。オン・オフどちらもεが大きいと試行的な探索が増え、最適状態価値への収束により多くのステップ数を要するだろう。しかしεの値が小さすぎると、どちらも状態・行動空間の探査が十分にはなされないであろう。

3層NNに関しては、中間層の有無が性能に与える影響を探る。バックギャモンの先行研究 [10] では、中間層のユニット数が160程度の中間層が有効であった。本研究ではキャントストップにおいて、同程度の規模のNNの性能も計測する。

TD(λ)のλにおいても、先行研究で適切だとされた値0.7が、キャントストップにおいても適切かどうか調べる。

経験リプレイに関しては、リプレイメモリの大きさが性能に与える影響を探る。リプレイメモリが大きくなるほど、訓練データ点の並びはより乱雑になり、これはSGDやこれに類するパラメータの更新法の効率化に寄与することになるであろう。その一方で、更新

回数が少ないNNのパラメタで生成された訓練データ点もリプレイメモリに含まれることになり、これは効率化を妨げるであろう。

経験リプレイに関してはまた、リプレイメモリに登録した状態・行動対の利用頻度が性能に与える影響を探る。状態・行動対の利用頻度が高くなるほど、NNのパラメタ更新回数に対するゲームプレイ生成数が少なくなり、これは強化学習の効率化に寄与することになるであろう。その一方で、同じ訓練データ点を何度も利用することになるため、NNが訓練データ集合に過適合しやすくなり、これは効率化を妨げるであろう。

強化学習法を用いて作成した人工知能が対戦するときには、先読み探索を利用し手を選択する。本研究の先読み探索に関しては、節点から手番プレイヤが変わらない範囲で探索の深さが性能に与える影響を探る。本研究の探索は、自プレイヤの手番が終了する、もしくはゲーム状況の実現確率がある一定の値に到達した場合に打ち切る。本研究ではこの先読み探索の手法を、先読み探索と呼ぶ。先読み探索を深く行うほど価値のバックアップを正確に行うことになり、状態価値関数の推定の精度向上に寄与することになるであろう。

6 実験方法

6.1 ゲーム木と状態行動空間

キャントストップのゲームプレイ全てを、展開形ゲーム（書籍 [12] を参照）のゲーム木で表現する。内部節点を、プレイヤの意思決定とダイスの出目によって生じるゲームプレイの分岐点に、終端節点はゲームの勝敗が決定した状況に対応づける。そして、内部節点は偶然節点（ダイスの出目による分岐点）かプレイヤ節点（プレイヤの意思決定による分岐点）に分類される。

本家研究で設計したゲーム木の節点は、表 7 で示されるように 5 つの種類に分類される。これらの節点は図 2 で示す親子関係を満たす。そして、本実験で行われる強化学習の状態は、ダイスの出目が決定した直後のプレイヤの意思決定点 (B_i の子の C_i と、B_i の子の D_i) に対応する。行動は、ダイスを振るか、勝敗がつくかするまでのマーカー及びポーンの操作 (状態に対応する節点から D_i の子節点までの枝の列) に対応する。したがって、行動 1 つは枝 1〜2 本に対応する。事後状態は、ストップ直後のダイスを振る状況、もしくは勝敗が着いた状況 (D_i の子節点) に対応する。

表 2: 節点の分類

種類	該当する節点
A	開始プレイヤを決定する偶然節点
B_i	プレイヤ i のダイスの 出目を決定する偶然節点
C_i	プレイヤ i が 4 つのダイスから 動かすポーンを決定するプレイヤ節点
D_i	プレイヤ i が次のプレイヤに 手番を渡すか決定するプレイヤ節点
T_i	プレイヤ i が勝った終端節点

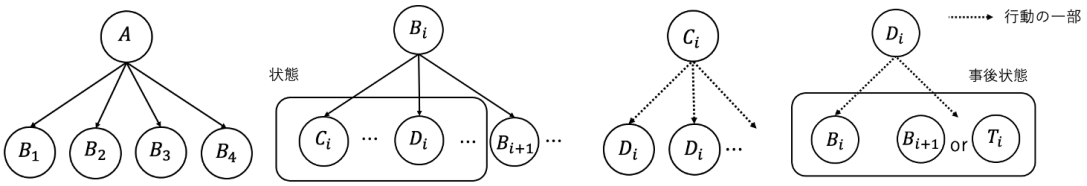

図 2: 各種節点の親子関係。円は節点、円中のラベルは節点の種類を表す。中点 3 つは、すぐ左にある種類の節点が 0 個以上続くことを表す。

6.2 先読み探索と性能の評価方法

まず、先読み探索するプレイヤについて述べる。強化学習の結果として得られた推定価値を元に先読み探索を行うプレイヤを作成し、3 人の K28+と対戦させ勝率を計測する。この先読みは、状態行動空間の探索により行われる。この探索の開始点は、対戦実験で意思決定する時の状態である。探索する経路は、開始点から始まって手番プレイヤが変わった時点で終わり、最後の状態遷移の直前の事後状態の推定価値で評価される。この経路は、終端状態に達しても終わり、手番プレイヤが勝つと評価値は 1、負けると 0 になる。また、この開始点から続く経路上の遷移確率 $\mathcal{P}^a_{ss'}$(式 3 参照) の積が閾値以下になったときにも、最後の遷移をする直前の事後状態の推定価値で評価する。全ての経路の推定価値が求まったら、最も評価値の期待値が大きい事後状態に対応する行動を選択する。遷移確率の積の閾値を 0 にすると、行動選択はグリーディ方策と一致する。

次に、強化学習法の性能評価の方法を述べる。性能評価は、強化学習の結果として得られた推定価値を使って先読み探索を行うプレイヤと 3 人の K28+とを対戦させて行う。K28+と強化学習によって得られたプレイヤの性能が同じ場合、勝率の理論値は 0.25 である。

最後に、本実験で採用した簡易的なヒューリスティックプレイヤ K28+の実装を説明する。レーンの選択は、Keller の 28 ルールで使用するスコアがなるべく小さくなるようにレーンの選択を行う。ストップの判断は、基本的には Keller の 28 ルールに従うが、ストップをすれば少なくとも 1 つレーンを獲得できるような盤面では必ずストップする。ただし、レーンの選択とストップの判断どちらも、最小のスコアが複数あった場合には一様ランダムに 1 つ最小スコアを選ぶ。K28+はランダムプレイヤ 3 人と対戦させた結果、97.0%の確率で勝利することが確認された。

レーンの選択とストップの判断に使用するルールを K28+から一様ランダムな選択や判断に差し替えた場合の性能を調査した。4 通りのプレイヤー同士を 1 万回対戦させ、その勝率を表 3 にまとめた。

表 3: ヒューリスティックプレイヤの性能

レーン選択	ストップの判断	勝率 (%)
ランダム	ランダム	0.4
K28+と同様	ランダム	0.8
ランダム	K28+と同様	34.2
K28+と同様	K28+と同様	64.5
合計		100.0

6.3 NN の学習とゲーム状況の符号化

事後状態の価値関数を 3 層 NN（中間層 1 つ）で近似する。NN はある手番プレイヤの符号化されたゲーム状況を表すベクトルを受け取り、そのプレイヤの推定勝率を出力する。隠れ層の基本的なユニット数は 16 で、活性化関数として ReL 関数を使用する。出力層のユニット数は 1 で、活性化関数にシグモイド関数を使用することで NN の出力が $(0, 1)$ の範囲に収まるようになる。

NN の学習には、深層学習フレームワーク「Caffe」 [4] を利用する。NN のパラメータの初期化には、このフレームワークに実装されている Xavier の初期化 [3] を使用する。さらに、NN のパラメータの更新は、リプレイメモリから重複を許して一様ランダムに選択

した訓練データ点をつかって行う。このリプレイメモリは、ゲームプレイを 1 つ生成するたびに、各強化学習法に沿った方法で採取された訓練データ点で最新の状態に保たれる。L2 正則化項は Caffe の重み減衰の機能（weight decay）を用いて行い、減衰の係数は 10^{-7} とした。各学習の学習率は 10^{-n} $(n = 1, 2, \cdots)$ の中から最適な値を設定した。

状態 s で行動 a を選択した直後の事後状態は、長さ 416 のビット列 $\Phi(s, a)$ で符号化する。ビット列の内訳は次のようである。ゲーム盤上のあるマス b のコマの配置を、次のような長さ 5 のビット列 x_b で表現する。

$$x_b = \begin{pmatrix} f(b,i) \\ f(b,i+1) \\ f(b,i+2) \\ f(b,i+3) \\ g(b) \end{pmatrix} \tag{54}$$

$$f(b,i') = \begin{cases} 1 & (b \text{ に } i' \text{のマーカーが存在}) \\ 0 & \text{otherwise} \end{cases} \tag{55}$$

$$g(b) = \begin{cases} 1 & (b \text{ にポーンが存在}) \\ 0 & \text{otherwise} \end{cases} \tag{56}$$

ここで整数 i は行動 a を選択したプレイヤ、$i' + 1$ は整数 i' で表されるプレイヤの次にダイスを振るプレイヤを表す。ゲーム盤上にマスは 83 個あるため、ビット列の長さは $83 \times 5 = 415$ となる。最後の 1 ビットは、この偶然節点でサイコロを振るプレイヤも i の場合は 1、そうでない場合は 0 である。

ビット列 $\Phi(s, a)$ は、行動 a を選択した直後のコマのみで表現されることとなる。したがって、異なる状態・行動対 2 つ $(s, a) \neq (s', a')$ が同じビット列 $\Phi(s, a) = \Phi(s', a')$ に対応付けられることがある。これにより、NN の定義域の大きさが状態・行動空間の大きさよりも大幅に小さくなり、各ビット列が訓練データ点に含まれる頻度が高くなることが期待できる。

6.4 方策反復の手順

本実験では、自己対戦形式でゲームプレイを生成して強化学習を行う。この方策改善の手順は次のようである。

まず、NN の推定価値に基づき ε-グリーディで挙動するプレイヤ 4 人で対戦を行い、1 つのゲームプレイを生成する。ゲームプレイ開始時は盤上にどのプレイヤのマーカーも存在しないゲーム状況で、ゲームプレイ終了時はいづれか 1 人のプレイヤが勝利したゲーム状況である。

次に、ゲームプレイから 8 つのエピソードを採取する。各エピソードは、あるプレイヤの手番の事後状態の系列であり、終端状態はいづれかのプレイヤが勝利したゲーム状況である。報酬は、勝利したプレイヤがそのプレイヤであれば 1、さもなくば 0 である。さらに、ゲームルールは 7 のレーンを軸にして左右対称になっているため、1 つのゲームプレイからそのプレイヤに着目したエピソードが 2 つ採取できる。このように採取されたエピソードから訓練データ点を採取する。TD(λ) では、エピソード内の事後状態とその後続の事後状態列の組を訓練データ点 1 つとする。Q 学習では、エピソード内の事後状態とその直後の事後状態の組を訓練データ点 1 つとする。

経験リプレイでは、訓練データ点をリプレイメモリに一定数だけ保存する。新しい訓練データ点 1 つを追加したとき、もし訓練データ点の数が一定数を超えていればバッファ内に保存された最も古い訓練データ点を削除する。また、リプレイメモリから訓練データ点を取り出すとき、リプレイメモリ内から訓練データ点を一様ランダムに取得する。

最後に、リプレイメモリを使ってある程度 NN を学習した後、この学習結果を ε-グリーディプレイヤに反映させる。

7 実験結果

実験で用いた強化学習法は、特に指摘がなければ

- 中間層のユニット数 $N = 16$
- 挙動方策は ε-グリーディ ($\varepsilon = 0.2$)
- 価値バックアップ法の TD($\lambda = 0.8$)
- リプレイメモリのサイズ $M = 10000$
- リプレイメモリ内の訓練データ点の使用頻度 $F_{\mathrm{ratio}} = 1.0$

である。NN の更新は、L2 正則化と係数 0.9 のモーメンタムを用いた SGD で行なった。また、NN の重み更新回数は 10 億回とした。

図 3 に、幾つかの異なる ε の値で強化学習を行い得られたグリーディ方策の値を変え、TD(λ) によって学習した強化学習プレイヤの勝率の推移を示す。縦軸は 3 つの K28+と対戦したグリーディ方策の勝率、横軸はリプレイメモリ内の訓練データ集合を使って NN の重みを更新した回数である。ただし、η は学習率を表す。この結果から、NN の更新回数が 10^9 のとき $\varepsilon = 0.2$ 付近の勝率が最も高いことがわかった。このことから ε は、0.1 以下では探査が十分行われず良い方策を発見することが難しくなり、0.4 以上では探査しすぎてグリーディ方策を評価しているとは見做し難しくなっていることが伺われる。他にも、$\varepsilon = 0.0$ では状態行動空間の探索が十分行われず、性能が落ちていることがわかる。そして、$\varepsilon = 0.5$ で性能がそれほど落ちていないことからグリーディな行動が勝率に寄与しないような局面が存在することがわかる。

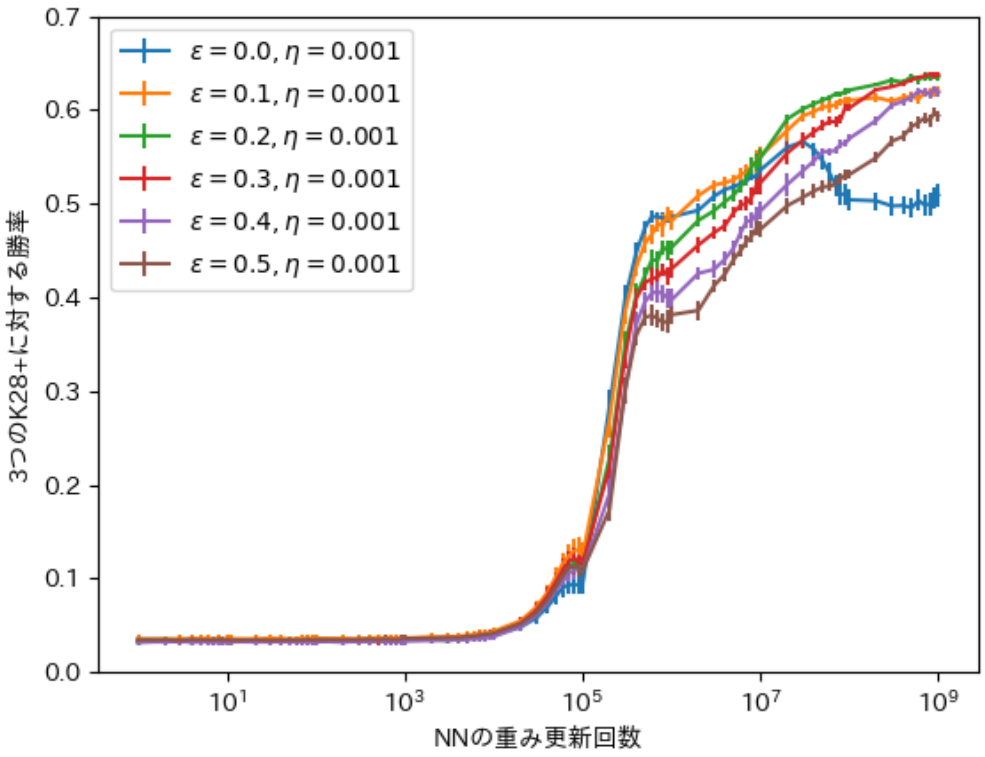

図 3: 勝率の ε 依存性（オンポリシー型）。勝率は強化学習実験を 32 回行い平均をとったもの。エラーバーは標準誤差から見積もった 95% 信頼区間を表す。

表 4: 最も平均勝率が高かった時点での重みを使い再度勝率を測定した。勝率は強化学習実験を 32 回行い平均をとったもの。信頼区間の信頼係数は 0.95 。

ε	平均勝率と標準誤差 (%)
0.0	52.3 ± 0.5
0.1	61.9 ± 0.4
0.2	62.0 ± 0.3
0.3	60.4 ± 0.3
0.4	57.1 ± 0.5
0.5	53.0 ± 0.9

図 4 に、幾つかの異なる ε の値で Q 学習を行い得られたグリーディプレイヤの勝率の推移を示す。ターゲットネットワーク (TN) の重みの更新は、価値を推定する NN の重みを 1000 回更新するごとに行う。この結果から、NN の更新回数が 10^9 のとき $\varepsilon = 0.2$ 付近が有意に性能が良いことがわかった。また、ターゲットネットワーク有無による、勝率の差はなかった。

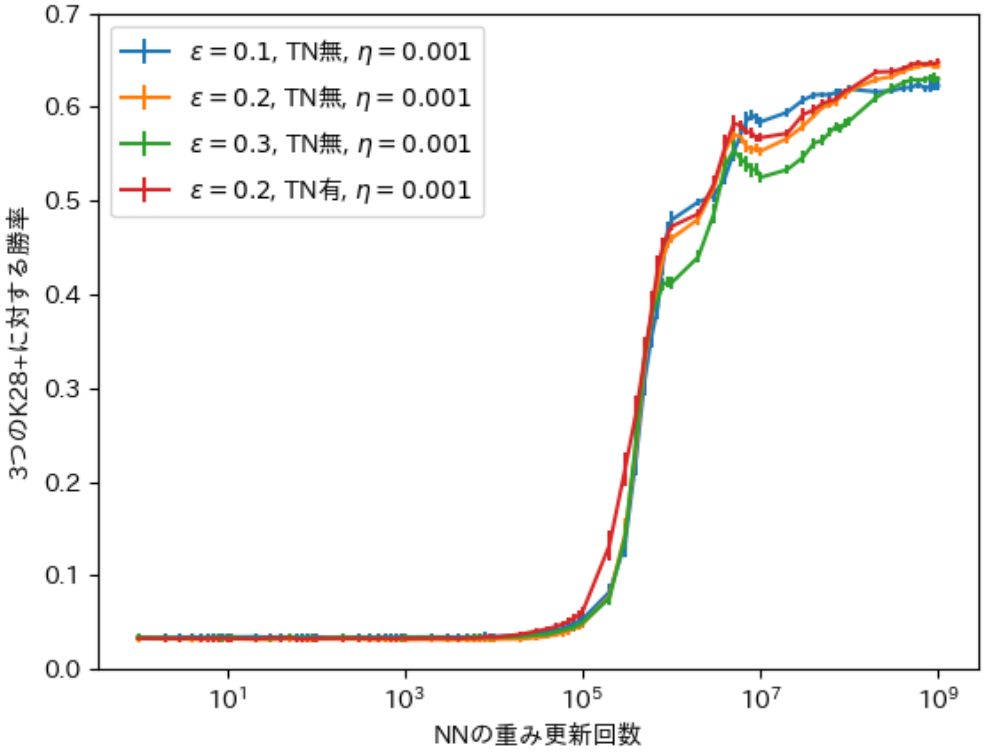

図 4: 勝率の ε 依存性（オフポリシー型）。勝率は強化学習実験を 32 回行い平均をとったもの。エラーバーは標準誤差から見積もった 95% 信頼区間を表す。

表 5: 最も平均勝率が高かった時点での重みを使い再度勝率を測定した。勝率は強化学習実験を 32 回行い平均をとったもの。信頼区間の信頼係数は 0.95 。

ε	平均勝率と信頼区間 (%)
0.1 (TN 無)	61.8 ± 0.4
0.2 (TN 無)	63.4 ± 0.6
0.2 (TN 有)	64.7 ± 0.3
0.3 (TN 無)	57.5 ± 0.5

図 5 に、幾つかの異なる中間層のユニット数 N で強化学習を行い得られたグリーディ方策の勝率の推移を示す。この結果からは、NN の更新回数が 10^9 のとき $N = 16, 64$ や中間層なしの間に有意な差が見られた。結果を詳しく見ると、10^9 付近の $N = 64$ の 3 層 NN

が最も良い勝率 6.52×10^{-1} を与えていて、これは 10^9 付近で得られた中間層なしの勝率の最大値 5.57×10^{-1} よりも有意に高い。このことから、中間層は一定の役割を果たしているとも考えられる。

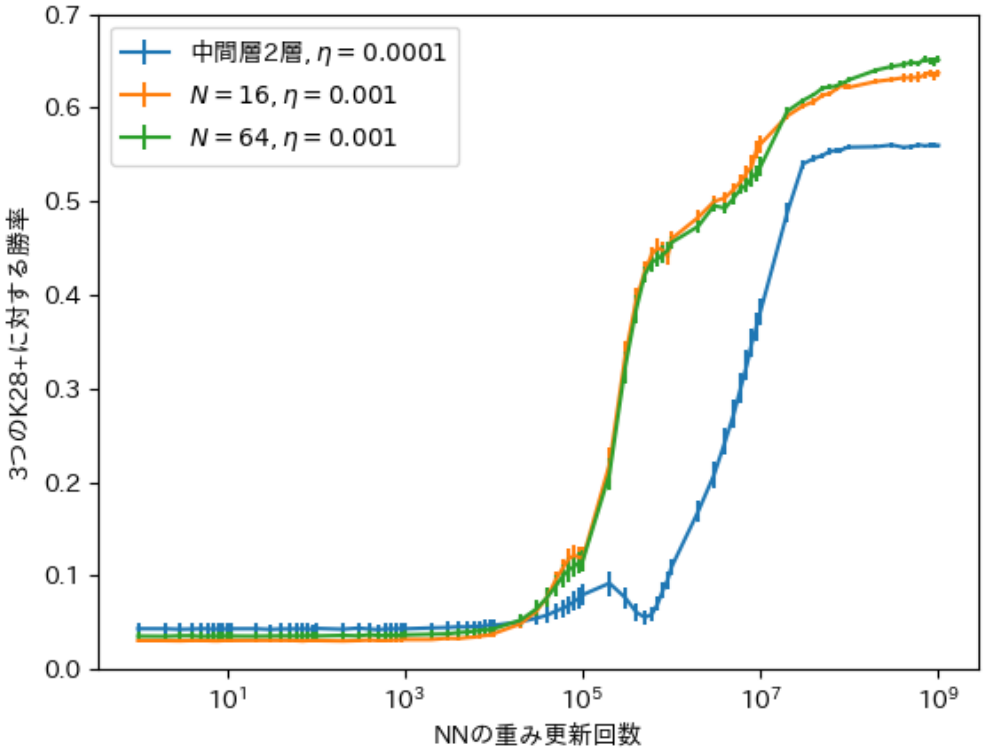

図 5: 勝率の中間層ユニット数依存性。勝率は強化学習実験を 32 回行い平均をとったもの。エラーバーは標準誤差から見積もった 95% 信頼区間を表す。

表 6: 最も平均勝率が高かった時点での重みを使い再度勝率を測定した。勝率は強化学習実験を 32 回行い平均をとったもの。信頼区間の信頼係数は 0.95 。

N	平均勝率と信頼区間 (%)
中間層 2 層	55.7 ± 0.3
16	62.3 ± 0.3
64	65.2 ± 0.4

図 6 に、幾つかの異なる λ の値で強化学習を行い得られたグリーディ方策の勝率の推移

を示す。この結果から、NN の更新回数が 10^9 のとき $\lambda = 0.8$ 付近で勝率が最も高くなることがわかった。これはバックギャモンの先行研究で提案された $\lambda = 0.7$ と概ね一致する。

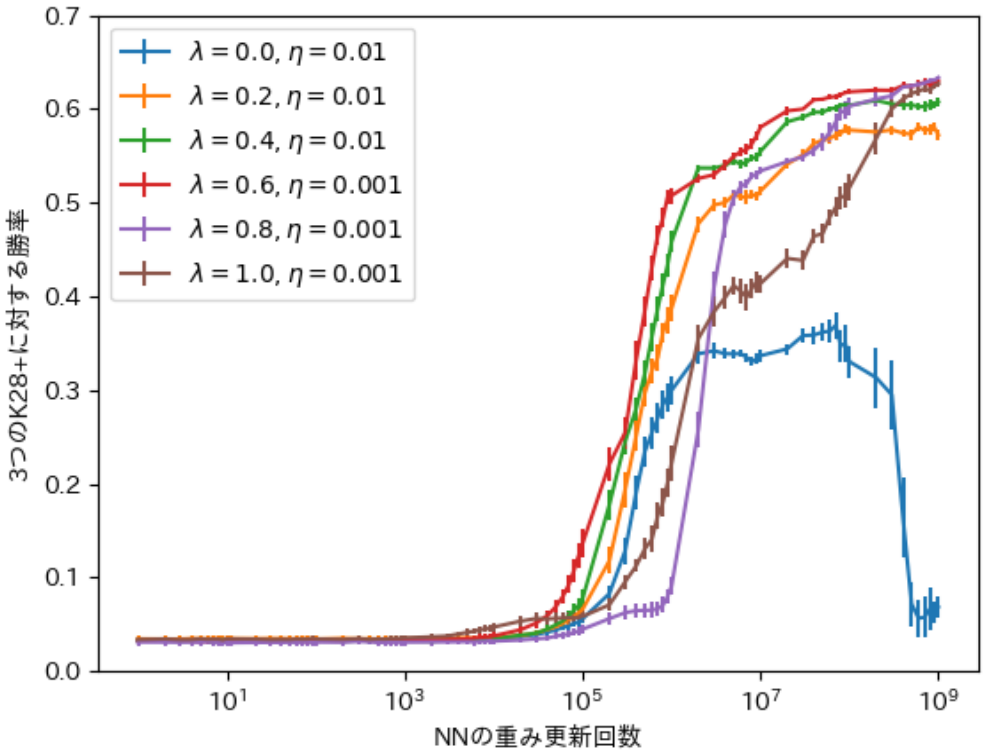

図 6: 勝率の λ 依存性。勝率は強化学習実験を 32 回行い平均をとったもの。エラーバーは標準誤差から見積もった 95% 信頼区間を表す。

表 7: 最も平均勝率が高かった時点での重みを使い再度勝率を測定した。勝率は強化学習実験を 32 回行い平均をとったもの。信頼区間の信頼係数は 0.95 。

λ	平均勝率と信頼区間 (%)
0.0	33.7 ± 0.0
0.2	57.5 ± 0.5
0.4	60.4 ± 0.3
0.6	62.9 ± 0.4
0.8	63.3 ± 0.2
1.0	62.6 ± 0.3

図 7 に、幾つかの異なるリプレイメモリのサイズ M で強化学習を行い得られたグリーディ方策の勝率の推移を示す。$M = 10000$ の勝率が $M = 1$ の間に有意な差は見られなかった。このことから、リプレイメモリは性能の向上に寄与しないことがわかった。

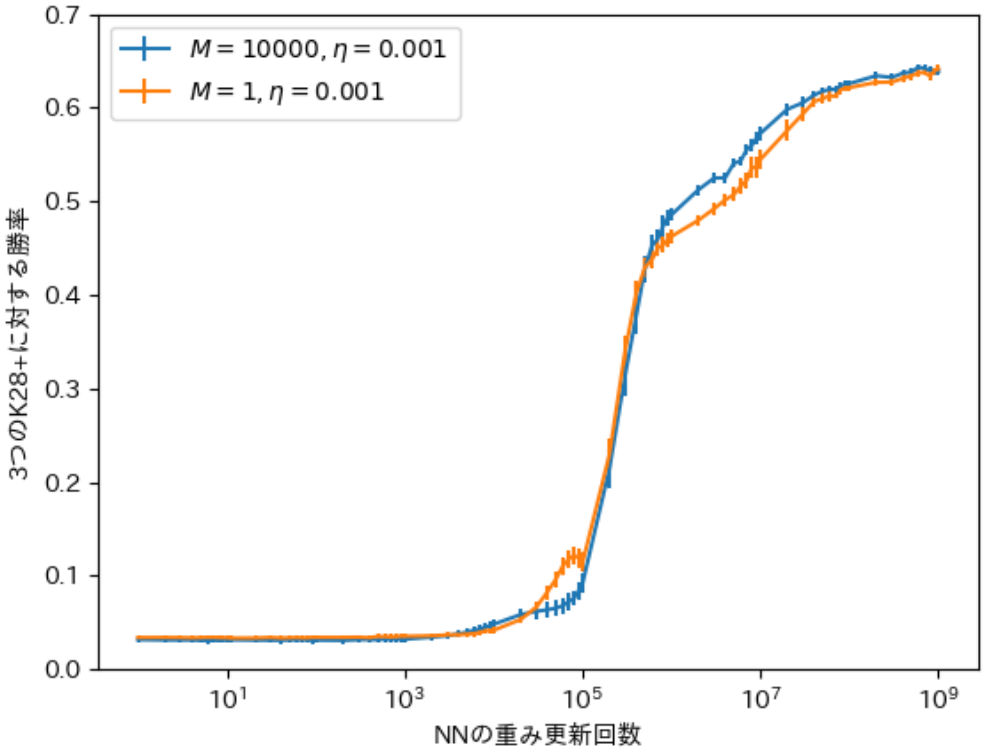

図 7: 勝率のリプレイメモリサイズ依存性。勝率は強化学習実験を 32 回行い平均をとったもの。

表 8: 最も平均勝率が高かった時点での重みを使い再度勝率を測定した。勝率は強化学習実験を 32 回行い平均をとったもの。信頼区間の信頼係数は 0.95 。

M	平均勝率と信頼区間 (%)
1	64.0 ± 0.4
10000	64.1 ± 0.4

図 8 に、幾つかの異なる F_{ratio} で強化学習を行い得られたグリーディ方策の勝率の推移を示す。結果には、NN の更新回数が 10^9 のとき F_{ratio} が 1 よりも高いと勝率が低くなる

現象が見られる。

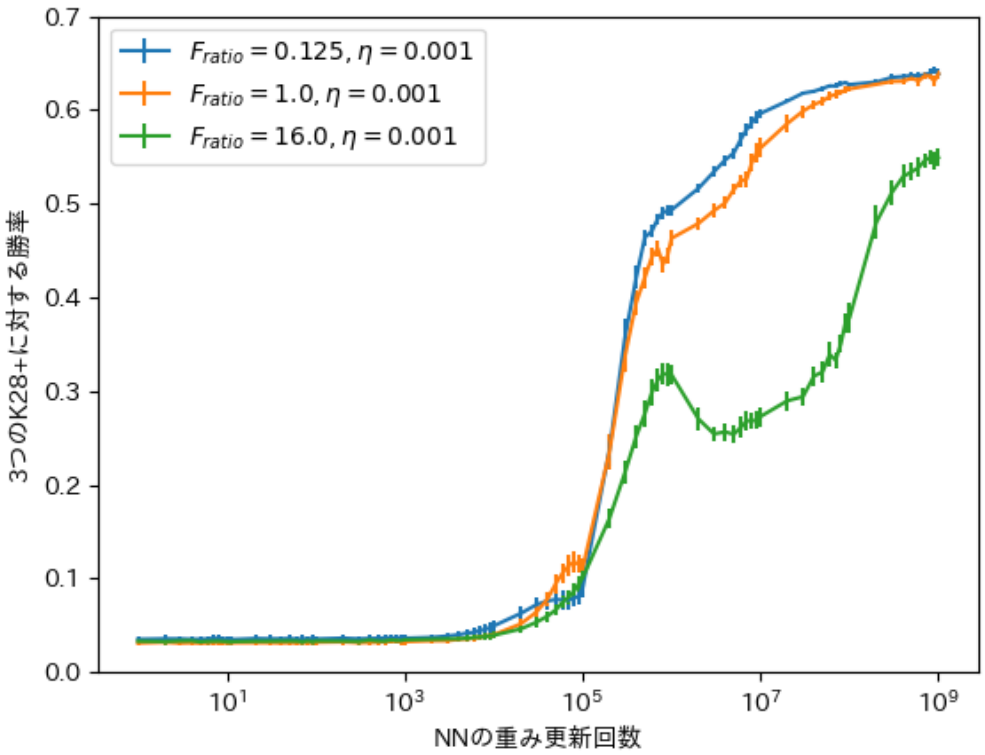

図 8: 勝率のリプレイメモリ内のデータ使用頻度依存性。勝率は強化学習実験を 32 回行い平均をとったもの。

表 9: 最も平均勝率が高かった時点での重みを使い再度勝率を測定した。勝率は強化学習実験を 32 回行い平均をとったもの。信頼区間の信頼係数は 0.95 。

F_{ratio}	平均勝率と信頼区間 (%)
0.125	63.9 ± 0.4
1.0	63.8 ± 0.4
16	55.0 ± 1.0

$\lambda = 0.8, \varepsilon = 0.2, N = 16, M = 10000, F_{\text{ratio}} = 1.0$ の設定で TD(λ) の強化学習により得られた価値推定を使用して、先読み探索を行い性能を評価した（表 10)。この結果から、価値推定に基づき先読み探索をすることで、先読み探索を行うよりも性能が改善すること

がわかった。訪問節点数が増えるほど勝率が増加しているため、さらに閾値を下げて探索を行うことで性能が改善する見込みがある。

表 10: 先読み探索の探索範囲依存性。勝率の平均値は強化学習実験を 32 回行い得られた。勝率は対戦回数 1 万回で推定。

閾値	事後状態数の平均	勝率の平均値 (%)
0	3.5×10^0	67.0 ± 1.0
1.0×10^{-2}	3.6×10	67.3 ± 1.0
1.0×10^{-3}	2.7×10^2	69.2 ± 1.0
1.0×10^{-4}	2.9×10^2	69.0 ± 1.0

8 おわりに

4 人不確定ゲームであるキャントストップに、先行研究で用いられている強化学習法と先読み探索を適用し、プレイヤの性能を調査した。適用・調査した手法は、ε-グリーディ法、3 層 NN による関数近似、オンポリシー型の TD(λ) による価値のバックアップ、オフポリシー型の Q 学習による価値のバックアップ、経験リプレイ、自プレイヤ先読み探索である。

実験結果から、次のことがわかった。まず、NN の中間層は性能に大きくは寄与しなかった。また、TD(λ) の ε-グリーディ方策のパラメタ ε は 0.2 程度が適切であり、パラメタ λ は 0.8 程度が適切であった。そして、オフポリシー型の Q 学習の性能は λ の値が適切に設定された TD(λ) と同程度の性能を得た。さらに経験リプレイは、これを適用しない場合でも性能が劣化せず、登録されたデータの利用頻度を極端に大きくした場合にのみ性能が劣化した。最後に、先読み探索は探索範囲が広がるほど性能が改善した。

先行研究の TD-Gammon と同様に、$\lambda = 0.7$ 付近で性能の向上が見られて先読み探索による性能の向上も見られた。その一方で、ε-グリーディ法のような探査を促す方法の有効性は、TD-Gammon の論文では論じられてはいない。バックギャモンとキャントストップどちらのゲームプレイにもダイスによってもたらされる不確実性があるが、キャントストップの強化学習においては特に探査的方策の利用が重要となる可能性がある。

参考文献

[1] N. C. Barford, 英行酒井. 実験精度と誤差：測定の確からしさとは何か. 丸善, 1997.

[2] Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, Vol. 365, No. 6456, pp. 885–890, 2019.

[3] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Vol. 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[4] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[5] Junjie Li, Sotetsu Koyamada, Qiwei Ye, Guoqing Liu, Chao Wang, Ruihan Yang, Li Zhao, Tao Qin, Tie-Yan Liu, and Hsiao-Wuen Hon. Suphx: Mastering mahjong with deep reinforcement learning. *ArXiv*, Vol. abs/2003.13590, , 2020.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, Vol. 518, No. 7540, pp. 529–533, 2015.

[7] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, Vol. 362, No. 6419, pp. 1140–1144, 2018.

[8] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An introduction second edition.* 2018.

[9] Gerald Tesauro. Practical issues in temporal difference learning. *Machine Learning*, Vol. 8, No. 3, pp. 257–277, 1992.

[10] Gerald Tesauro. Programming backgammon using self-teaching neural nets. *Artificial Intelligence*, Vol. 134, No. 1, pp. 181–199, 2002.

[11] 瀧雅人, 講談社サイエンティフィク. これならわかる深層学習入門 = Introduction to deep learning. MLS 機械学習スタートアップシリーズ. 講談社, 2017.

[12] 岡田章. ゲーム理論. 有斐閣, 2011.

[13] 国沢清典. 確率統計演習 2 統計. 培風館, 1996.