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・通信工学専攻　博士前期課程		
氏　　名	佐藤佑史	学籍番号	1431053
論 文 題 目	ガイスターにおける自己対戦による行動価値関数の学習		

要　旨

ガイスター(Geister)とは，Alex Randolph によって開発された二人不完全情報ゲームである．相手の駒の色が分からないチェスのようなゲームとなっている．ガイスターにおいて，2 種類ある駒の推測やブラフなどの心理戦と将棋のような先読みに基づいた駒の動きが重要となる．ガイスターにおけるＡＩは現在非常に弱い．

本研究では，機械学習の一種である強化学習法のモンテカルロ法や TD(0)，Sarsa(λ)学習を用いて，ある局面における手を指したときの勝率の見積もりを計算する行動価値関数をＡＩ同士での対戦を行った結果を用いて学習する．通常のガイスターだけでなく，より盤面の小さい Minimum-Geister を定義し，このガイスターにおいても行動価値関数の学習を行うことで既知である最善戦略を求めることが出来るかも検証する．なお，通常のガイスターにおける局面数は膨大となるため，行動価値関数は 3 層ニューラルネットワークという神経回路を模した数学モデルを用いて近似する．このニューラルネットワークの入力として，駒に対する推測を全く用いない盤面の情報のみにより構成される入力，Prototype-Based Learning を使用した相手の駒に対する推測と盤面の情報で構成される入力，同じく Prototype-Based Learning を使用し両プレイヤの駒に対する推測と盤面の情報で構成される入力の三種類を用意し，学習を行う．さらに，この入力に出口と呼ばれるマスへの最短距離や隣接している駒の位置などのゲーム上で重要と思われる特徴を加えるなどの改良を行った入力でも同様の学習を行い，学習により獲得した各入力での行動価値関数を用いたＡＩプレイヤの性能比較を行う．行動価値関数の学習において，通常のガイスターのルールに加えて，ルールに変更を加えた様々なルール上での学習や着手に制限を加えた上での学習を行う。

さらに，学習によって得られた行動価値関数に基づき手を選択するＡＩプレイヤを作成し，ランダムプレイヤや既存手法であるモンテカルロ木探索を利用したＡＩとの対局実験を行う．

平成 27 年度 電気通信大学 情報・通信工学専攻 修士論文

ガイスターにおける自己対戦による行動価値関数の学習

指導教員　村松正和　教授

平成 28 年 1 月 29 日

電気通信大学大学院 情報理工学研究科 情報・通信工学専攻
1431053
佐藤佑史

目 次

第1章　はじめに

本章では，研究背景および研究目的について説明する．

1.1　研究背景

人工知能の研究において，知識の理解だけでなく知識の構築についても取り扱っている．現在の人工知能研究では，学習や推論といった一般的な問題から，チェスや定理の証明と言った特定の分野に至るまで広大な領域を扱う [1]．とりわけ，ゲームはルールが明確で，勝ち負けにより良し悪しを判断することができるとため，人工知能の性能評価に用いられてきた．そのうえ，ゲームによっては強い人間プレイヤがいるため，人知を超えるという目標も設定することが可能である．

完全情報ゲームではゲーム木探索や評価関数の自動生成といった有効な手段が知られている．それらを適用した AI プレイヤはトップレベルの人間プレイヤの強さを持っているものがある．例として，オセロでは 1997 年に Logistello が世界チャンピオンの村上健に 6 戦全勝で勝利した [2]．また，チェスにおいては 1997 年に Deep Blue が世界チャンピオンの Garry Kimovich Kasparov に 2 勝 1 敗 3 引で勝利した [3]．将棋では 2014 年にドワンゴ・日本将棋連盟主催 第 3 回 将棋電王戦において，コンピュータがプロ棋士に対して 4 勝 1 敗で勝利した [4]．囲碁においては，2014 年に第 2 回 電聖戦において Crazy Stone が 19 路盤 4 子局で依田紀基九段に対して 2 目半勝ちした [5]．

このように完全情報ゲームにおいて研究は進んでいるものの，不完全情報ゲームでは完全情報ゲームほど研究は進んでいない．不完全情報ゲームではプレイヤが得られるゲームの状態に関する情報が部分的であり，完全情報ゲームなどで使われる探索手法を直接適用することはできない．

不完全情報ゲームの一種にガイスターがある．2013 年，Ghosts Challenge 2013 において，PrototypeBasedLearning を用いた推測により，駒色を断定し，完全情報ゲームとした上で，モンテカルロ木探索を用いる AI プレイヤ BLISS が勝利を収めた [6]．それ以降，同様のモンテカルロ木探索は主流の方法となっている．しかし，モンテカルロ木探索を用いた AI プレイヤは人間に比べて非常に弱い．

1.2 研究目的

本研究では，機械学習の一種である強化学習をガイスターに適用する．強化学習は不確定完全情報ゲームであるバックギャモンにおいて，TD-Gammon というプログラムの開発に寄与した [7]．TD-Gammon はバックギャモンについての予備知識をほとんど必要とせずに，グランドマスターに近いレベルの手を指せる AI プレイヤである．

バックギャモンは不確定完全情報ゲームであり，不確定ゲームはサイコロやコインなどの確率的な要素を含む．TD-Gammon はサイコロによる確率を考慮した，ある局面における勝率の見積もりを計算する状態価値関数を獲得することにより，強い AI となった．

一方，ガイスターはバックギャモンとは異なり，運の要素を一切含まない確定不完全情報ゲームであるが，ガイスターにおける駒の色が推測に紐付いた確率により決定されるものと仮定するのならば，ある局面の手における勝率の見積もりを計算する行動価値関数を獲得することにより，強い AI を作ることができる可能性がある．

ターン t となる局面 s_t での手 a における勝率の見積もりを計算する行動価値関数 $Q(s_t, a)$ では，着手より相手の駒の推測を行なうことが一般的な戦略であるガイスターにおいて，これまでの着手が含まれていないため相手の駒に対する推測を考慮した行動価値関数を求めることが出来ない．初期局面 s_0 から t 手指された後の局面 s_t までの全ての局面と局面 s_t における着手 a により定義される行動価値関数 $Q(s_0, s_1, s_2, \ldots, s_t, a)$ を求めることができれば，相手の手を考慮したよい戦略が求まる可能性がある．

しかし，通常のガイスターではゲーム中に現れる局面の組合せが膨大であるため，行動価値関数 $Q(s_0, s_1, s_2, \ldots, s_t, a)$ を求めることは現実的ではない．よって，$s_0, \ldots, s_{t-1}$ を駒の動きなどの特徴から求められる推測値を用いて近似し，特徴 f とすることによって，行動価値関数 $Q(f, s_t, a)$ を求め，行動価値関数を利用した AI プレイヤを作成する．

第2章　基礎知識

本章では，本研究の実験内容に関連する基礎知識について説明する．まず，研究対象であるガイスターについて説明する．そして，評価関数の学習を行うために必要となる強化学習とニューラルネットワークについて説明する．

2.1　ガイスター (Geister)

ガイスター (Geister)[8] は Alex Randolph によって開発された二人確定不完全情報ゲームである．「Geister」はドイツ語で「オバケ」の意味であり，白い布を被ったオバケをモチーフにした背中に赤もしくは青色の印の付いた駒を用い，互いに相手の色がわからない駒を取り合う心理的要素の強いボードゲームとなっている．

また，販売されている国により，ゲームの名前が異なり，イタリアでは「Fantasmi」，英語圏では「Ghosts」といった名前を持つ．1982 年に発売され，同年のドイツ年間ゲーム大賞にノミネートされた．発売以降，出版社や名前を変えて再版を繰り返されており，シュミット社，ノリス社，ヴェニスコネクション社，ドライ・マギア社などから販売されている．現在，日本では KOD KOD INTERNATIONAL GAMES により出版された「ガイスター」をメビウスゲームが販売している．

ルールが非常に簡単であり，プレイ時間も 10 ～ 20 分，平均終了ターン数が 45 [1] と短く，対象年齢も 8 歳以上であるため非常に手軽なゲームとなっている．そのうえ，子供や初心者でも相手に勝つことが可能である．相手の駒の動きを元に相手の駒の色を推測しながらゲームを進行させなくてはならないが，論理によって全ての駒を推測することはとても難しく，ハッタリや勘を駆使する必要のあるゲームでもある．特に，将棋のような先を読みながら駒を移動させる能力と相手に対する推測や自分の駒の色を悟られないようにする，さらには，ブラフで相手を騙すような心理戦における能力がガイスターにおいて勝利するために必要となる．

2013 年，2014 年にはガイスターの AI 同士を競わせるコンペティションである GhostsChallenge[9] が開かれた．

2.1.1　ルール

盤のサイズは 6×6 のマスにより構成されており，四隅には矢印が描かれた出口と呼ばれるマスが存在する．駒には背中に青い印が付いた「良いオバケ」の駒と赤い印の付いた

[1] 同じ対局者による 100 戦の平均

「悪いオバケ」の駒がある．各プレイヤはこれら２種類の駒を４つずつ持ち，ゲーム開始時に自陣の８マスに自由に配置することができる[2]．盤上に配置された駒は背中に印があるため，両プレイヤは互いの盤上の駒の色を確認することは出来ない．駒の配置後，各プレイヤは交互に駒を動かす．駒は前後左右に１マス動かすことができる．ただし，すでに自分の駒があるマスに駒を動かすことは出来ない．将棋のように駒を相手の駒のいるマスに動かした時には，相手の駒を取ることができ，そのとき，相手の駒の印の色を確認できる．手番をパスすることは出来ず，必ずいずれかの駒を動かさなくてはならない．相手プレイヤ側にある矢印の描かれた出口のマスに「良いオバケ」の駒が乗っているとき，プレイヤは自分のターンにこの駒を脱出させる手を指すことができる．なお,「悪いオバケ」は出口から脱出させることはできない．

ゲームの勝利条件は，次の３つの条件の内１つを満たすことである．

1. 相手の「良いオバケ」４つ全てを取る．
2. 自分の「悪いオバケ」４つを全て相手に取らせる．
3. 自分の「良いオバケ」１つを相手側の出口から脱出させる．

GhostsChallengeではこれらのルールに，両プレイヤがそれぞれ50手を指した時点で勝敗がついていない場合に引き分けとするルールを加えられている．今回は，このGhostsChallengeのルールを採用する．

図2.1はPlayer Aから見たガイスターにおける初期局面の例であり，この場合，相手プレイヤであるPlayer Bのコマの印は確認できないようになっている．Player Aの出口は左上と右上の矢印の付いたマスとなっている．

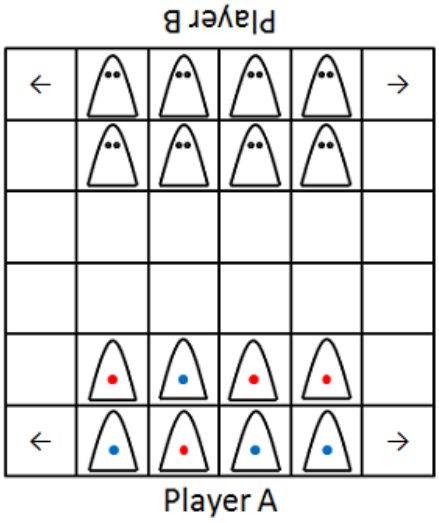


図 2.1: ガイスターにおける初期局面の例

[2] ${}_8C_4 = 70$ 通りの組み合わせがある

2.2 強化学習

強化学習では，数値化された報酬を最大にするために，何をすべきかを学習する．教師あり学習のように学習者がどの行動を取るべきかは教えられず，どの行動を取ればより報酬を得られるかを見つけ出す必要がある．

強化学習には３つの主な構成要素がある．それは方策 (policy)，報酬 (reward)，価値関数 (value function) である．方策はある時点での学習エージェントがどのような行動をするかを定義する．エージェントとは振る舞う者のことを意味する人工知能における基本的な用語である．報酬とはその状態に備わった望ましさを表している．価値関数はその状態を起点として将来にわたって得られる報酬の総量となる．

ゲームにおいて，方策はある局面においてどういった手を選択するか，報酬はゲームの勝敗結果，価値関数は勝率の見積もりに対応する．

図 2.2 は事後状態の概念図である．

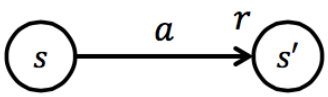


図 2.2: 事後状態の概念図

ある状態 s で行動 a を選択した後の状態は事後状態 s' となり，そのとき得られる報酬が r となる．ゲームにおける状態は局面，行動は手，事後状態は手を指したあとの局面に対応する．報酬は，状態 s' 勝敗の結果に対応し，ゲームに勝利した時 1，敗北した時 0，引き分けた時 0.5，ゲームが終了していない場合には 0 とする．

状態 s の価値関数を状態価値関数 $V(s)$ とし，状態 s における行動 a を取った際の価値関数を行動価値関数 $Q(s,a)$ とする．つまり，ゲームにおいて，状態価値関数 $V(s)$ は局面 s での勝率の見積もり，行動価値関数 $Q(s,a)$ は局面 s で手 a を取った際の勝率の見積もりを意味する．今後，局面の評価関数として状態価値関数，局面における手の評価関数として行動価値関数を用いる．

本論文におけるエピソードとは状態と状態行動対が交互に現れる図 2.3 のような系列を指すこととする．ゲームにおいて，エピソードは 1 つのゲームの開始から終了までの局面遷移となり，状態は局面，状態行動対は局面とその局面における手の組となる．

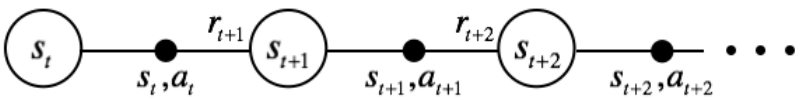


図 2.3: エピソード

2.2.1 強化学習におけるモンテカルロ法 (MC)

今回用いるモンテカルロ法は機械学習の枠組みにおけるものであり，強化学習の一種である．このモンテカルロ法を以下 MC と書く．

MC は数値解析の分野におけるシミュレーションや数値計算を乱数を用いて行うモンテカルロ法とは異なる．数値解析の分野におけるモンテカルロ法は乱数を用い，n 回のシミュレーションを行い，ある事象が m 回起これば，その事象の起きる確率を m/n で近似するものである．一方，機械学習の分野における MC は行動によって得られた報酬だけを頼りに状態に対する価値もしくは行動に対する価値を推定する方法を指す．機械学習分野における MC は数値解析分野におけるモンテカルロ法とは，必ずしも乱数とは関連付けられない点で大きく異なる．つまり，機械学習分野の MC は乱数を用いない場合がある点で注意が必要である．

この MC はある状態 s から得られる報酬を予測し，状態の価値と次に行う行動を決定する．このとき，MC は以下の式で状態価値関数および行動価値関数を更新する:

$$V(s) \leftarrow V(s) + \alpha[R_t - V(s)] \tag{2.1}$$

$$Q(s,a) \leftarrow Q(s,t) + \alpha[R_t - Q(s,t)]. \tag{2.2}$$

ここで，T はエピソードの終了時刻であり，α は学習率を表し，$0 < \alpha < 1$ である．また R_t はシミュレーションによって得られる報酬を未来に得られる分割り引いたものの総和であり，以下の式で表される:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T. \tag{2.3}$$

ここで r_t は時刻 t で得られた報酬であり，γ は割引率 $(0 \leq \gamma \leq 1)$ である．MC はある状態 s から何らかの方策で次の行動を選択し，エピソードが終了し R_t が収束するまでそれを繰り返した後，$V(s)$ と $Q(s,t)$ を更新するという行動を繰り返すことによって状態および行動の価値を学習する．

ゲーム終了時の勝敗結果のみを報酬とし，それ以外での報酬を 0 とすると，(2.3) 式は以下のように表せる:

$$R_t = \gamma^{T-t-1} r_T. \tag{2.4}$$

式 (2.4)，(2.1)，(2.2) より，ゲームにおける MC は以下の式で表せる:

$$V(s) \leftarrow V(s) + \alpha[\gamma^{T-t-1} r_T - V(s)]$$

$$Q(s,a) \leftarrow Q(s,t) + \alpha[\gamma^{T-t-1} r_T - Q(s,t)].$$

よって，ゲームでは，状態 s における状態価値関数 $V(s)$ と状態 s での行動 a における行動価値関数 $Q(s,a)$ はゲーム終了時の勝敗結果を用いて更新することとなる．

2.2.2 TD 学習 (TD)

強化学習の中心となる新しい考え方の代表的なものとして TD 学習 (Temporal Difference Learning) がある．この TD 学習を以下 TD と呼ぶ．TD では MC と同様に経験から直接

学習することが可能である．しかし，最終結果を待たずに，他の推定値の学習結果を一部利用し，推定値を更新する点でMCとは異なる．

MCではR_tがエピソードの終了までわからない．よって，$V(s_t)$の増分量を決定するためにエピソードの終わりまで待たなくてはならない．しかし，TDでは次の時間ステップを待つだけで良い．TDでは時刻$t+1$で直ちに目標値を作り，観測した報酬r_{t+1}と推定量$V(s_{t+1})$を用いて更新を行う．次状態に対して着目したTD法をTD(0)と呼び，以下の式で表す:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]. \tag{2.5}$$

式(2.1)，(2.5)より，MCがR_tを目標として更新するのに対し，TD(0)は$r_{t+1}+\gamma V(s_{t+1})$を目標とし更新することがわかる．

本研究において，ゲーム終了時の報酬を勝敗結果とし，それ以外での報酬を0とするため，式(2.5)は以下のように表せる:

$$\Delta V(s_t) = \begin{cases} \alpha[\gamma V(s_{t+1}) - V(s_t)] & (s_{t+1}\text{が非終端状態}) \\ \alpha[r_T - V(s_t)] & (s_{t+1}\text{が終端状態}). \end{cases} \tag{2.6}$$

次にTD(0)のアルゴリズムを示す．

$V(s)$を任意に初期化する
各エピソードに対して繰り返し：
　sを初期化
　エピソードの各ステップに対して繰り返し：
　　$a \leftarrow s$に対して方策により与えられる行動
　　行動aを取り，報酬rと次状態s'を観測する
　　$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$
　　$s \leftarrow s'$
　　sが終端状態なら繰り返しを終了

2.2.3 TD(λ)

TD(λ)とは，MCとTD(0)の中間に当たる手法である．ここでは，MCとTD(0)の収益の違いからnステップ収益，λ収益，およびTD(λ)を説明する．

nステップ収益

一般的にMCにおいて，$V(s_t)$はr_{t+1}から$\gamma^{T-r-1}r_T$までの収益で更新される:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T. \tag{2.7}$$

R_tを目標値と呼ぶことにする．MCでは，目標値は完全な形の収益となる．しかし，TD(0)においては，目標値は最初の報酬に次の状態の割引された推定価値$V(s_{t+1})$を加えたもの

となる:

$$R_t^{(1)} = r_{t+1} + \gamma V(s_{t+1}). \tag{2.8}$$

1 つ次の状態の報酬を利用するため，$R_t^{(1)}$ は 1 ステップ収益と呼ばれる．この $\gamma V(s_{t+1})$ は，(2.7) 式の残りの項 $\gamma r_{r+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T$ の代わりになっている．2 ステップ後も，同様に考えることができる:

$$R_t^{(2)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 V(s_{t+2}). \tag{2.9}$$

ここで，$\gamma^2 V(s_{t+2})$ は $\gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots + \gamma^{T-t-1} r_T$ の代わりとなっている．一般的に，n ステップ収益は以下のようになる:

$$R_t^{(n)} = r_{t+1} + \gamma r_{r+2} + \gamma^2 r_{r+3} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+n}). \tag{2.10}$$

この量は，n ステップ後を打ち切り，n 番目の次状態の価値を加えて，打ち切りを近似的に修正しているため,「修正型 n ステップ打ち切り収益」と呼ばれることもある．n ステップ収益を用いた n ステップ TD 法と呼ぶ.

もし，n ステップに至る前にエピソードが終了するなら，この n ステップ収益の打ち切りはエピソードの終わりで発生し，$t+1$ から T までの収益となる．つまり，もし $t+n \geq T$ であれば，$R_t^{(n)} = R_t^{(T-t)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T$ となる．式 (2.3) より $R_t^{(n)} = R_t^{(T-t)} = R_t$ が成り立つ.

λ 収益

状態価値関数および行動価値関数の更新は任意の n ステップ収益に対してのみでなく，n ステップ収益の平均値に対しても行うことができる．例として，2 ステップ収益の半分と 4 ステップ収益の半分 (つまり，$R_t^{\text{ave}} = \frac{1}{2} R_t^{(2)} + \frac{1}{2} R_t^{(4)}$) に対して更新を行うこともできる．要素となる収益の重みが正で，その合計が 1 になっているならば，いかなる収益の集合もこのように平均化可能となる.

TD(λ) アルゴリズムは，n ステップ収益を平均化する手法の 1 つである．各収益は $\lambda^{n-1} (0 \leq \lambda \leq 1)$ に比例して重み付けされる．結果として得られる収益は λ 収益と呼ばれ，次のように定義される:

$$R_t^{\lambda} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)}. \tag{2.11}$$

1 ステップ収益には最大重み値 $1-\lambda$ が与えられ，2 ステップ収益にはその次に大きな重み値 $(1-\lambda)\lambda$，3 ステップ収益には $(1-\lambda)\lambda^2$ が与えられ，以下同様となる．1 ステップごとに重みの値は λ の値の分だけ減少する．ここで無限等比級数の和の公式より $\sum_{n=1}^{\infty} \lambda^{n-1} = \frac{1}{1-\lambda}$ が成り立つ．つまり，正規化係数 $1-\lambda$ により，重みの合計が 1 になることが保証される.

終端状態に達した後，後続の n ステップ収益は全て R_t に等しくなる．エピソードが $T-t$ で終わり，その報酬が R_t となることと無限等比級数の和の公式より，$n \geq T-t$ の総和に

関して以下のことが言える:

$$\begin{aligned}\sum_{n=T-t}^{\infty} \lambda^{n-1} R_t^{(n)} &= \lim_{n\to\infty} \sum_{k=T-t}^{n} R_t \lambda^k \\ &= \lim_{n\to\infty} \frac{R_t(1-\lambda^{n+1})}{1-\lambda} \\ &= \lim_{n\to\infty} \frac{R_t}{1-\lambda} + \lim_{n\to\infty} \frac{R_t \lambda^{n+1}}{1-\lambda} \\ &= \frac{R_t}{1-\lambda}.\end{aligned}$$

よって，式 (2.11) は以下のように書くことができる:

$$\begin{aligned}R_t^{\lambda} &= (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)} \\ &= (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)} + \lambda^{T-t-1} R_t. \end{aligned} \tag{2.12}$$

$\lambda = 1$ のとき，式 (2.12) は以下の式となる:

$$R_t^{\lambda} = R_t. \tag{2.13}$$

つまり，$\lambda = 1$ の場合には MC と同じ収益を利用することになる．一方，$\lambda = 0$ の場合には，λ 収益は以下の式になる:

$$R_t^{\lambda} = R_t^{(1)}. \tag{2.14}$$

これは 1 ステップ収益と同じである．よって，$\lambda = 0$ の場合には λ 収益は TD(0) と同じになる．

TD(λ) の前方観測的な見方

λ 収益を利用し，状態価値関数を更新する手法が TD(λ) である．その更新式は以下のように表される:

$$V(s) \leftarrow V(s) + \alpha[R_t^{\lambda} - V(s)] \tag{2.15}$$

$$R_t^{\lambda} = \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)} + \lambda^{T-t-1} R_t \tag{2.16}$$

本研究において、報酬が終端状態でしかないことより，このように書くことができる:

$$R_t^{\lambda} = \sum_{n=1}^{T-t-1} \lambda^{n-1} V(s_n) + \lambda^{T-t-1} r_T. \tag{2.17}$$

訪問した各状態に対して，将来起こりうる全ての報酬を眺め，最良の組合せを決定する．ある状態から前方を眺めて更新を行った後，次の状態に移動し，以前の状態を参照する必要はない．これに対して，将来の状態はそれに先行する位置から観察され，処理が行われる．そのため，このアプローチは前方観察的な見方と呼ばれる．

TD(λ) の後方観察的見方

TD(λ) の後方観察的見方は概念的にも計算的にも単純であると理由から有用である．後方観察的見方は，前方観察的見方を近似するための，因果関係のある漸進的メカニズムを提供する．

TD(λ) の後方観察的見方においては，強化学習の基本的なメカニズムの 1 つである適格度トレース (eligibility trace) を用いる．適格度トレースは各状態に対応する．時刻 t における状態 s の適格度トレースは $e_t(s) \in \mathcal{R}^+$ と表される．各ステップにおいて，この適格度トレースは全ての状態に対して $\gamma\lambda$ だけ減衰し，そのステップで訪問された 1 つの状態の適格度トレースは，全ての $s \in \mathcal{S}$ に対して次式のように 1 だけ増加する:

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & (s \neq s_t) \\ \gamma\lambda e_{t-1}(s) + 1 & (s = s_t). \end{cases} \tag{2.18}$$

これ以降，λ を**トレース減衰パラメータ** (trace-decay parameter) と呼ぶ．この適格度トレースは特に**累積トレース** (accumulating trace) と呼ばれる．

適格度トレースは，価値関数の更新が発生したとき，各状態が学習上の変化を受けることが「適格 (eligible)」であることの度合いを示している．TD(λ) の後方観察的見方においては, TD 誤差は, 最近訪問した, 非ゼロの適格度トレースを持つ全ての状態に対して，比例配分的な更新を生じさせる．つまり，全ての $s \in \mathcal{S}$ に対して，

$$\Delta V_t(s) = \alpha\delta_t e_t(s). \tag{2.19}$$

TD(λ) の後方観察は，時間を遡る向きに行われる．各時点において，現在の TD 誤差を観察し，その時点での適格度トレースに従って，各先行状態に対してその誤差を割り当てる．

オンラインで学習を行う TD(λ) の擬似コードは以下のようになる．

$V(s)$ を任意に初期化し，全ての $s \in \mathcal{S}$ に対して $e(s) = 0$ とする
各エピソードに対して繰り返し：
    s を初期化
    エピソードの各ステップに対して繰り返し：
        $a \leftarrow s$ に対して方策により与えられる行動
        行動 a を取り，報酬 r と次状態 s' を観測する
        $\delta \leftarrow r + \gamma V(s') - V(s)$
        $e(s) \leftarrow e(s) + 1$
        すべての s について：
            $V(s) \leftarrow V(s) + \alpha\delta e(s)$
            $e(s) \leftarrow \gamma\lambda e(s)$
        $s \leftarrow s'$
        s が終端状態なら繰り返しを終了

Sarsa(λ)

これまでは状態価値関数を学習する TD(λ) を扱ってきた．行動価値関数を学習するためには TD(λ) に変わる新たな手法が必要となる．行動価値関数を学習するため，TD(λ) に変更を加えたものが Sarsa(λ) である．これまでは状態間の遷移に着目して状態価値を学習する方法について考えてきたが，ここでは状態行動対の間の遷移に着目し，状態行動対に対する価値をする方法を示す．

TD(0) での状態価値関数の更新式 (2.5) の $V(s)$ を $Q(s,a)$ で置き換えたものが Sarsa(0) における更新式となる

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)]. \tag{2.20}$$

この更新式は非終端状態 s_t から状態遷移を行うたびに毎回適用される．s_{t+1} が終端状態の場合，上式で $Q(s_{t+1},a_{t+1})$ は 0 と定義する．この規則は，ある状態行動対から次の状態行動対への遷移を定義する 5 項組 $(s_t,a_t,r_{r+1},s_{t+1},a_{t+1})$ の全てを利用している．Sarsa という名前は，この 5 項組の頭文字から取ったものである．

Sarsa(0) アルゴリズムの一般形は次のようになる．

$Q(s,a)$ を任意に初期化
各エピソードに対して繰り返し：
　s を初期化
　Q から導かれる方策を用いて s で取る行動 a を選択
　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 から導かれる方策を用いて s' で取る行動 a' を選択
　　$Q(s,a) \leftarrow Q(s,a) + \alpha[r_{t+1} + \gamma Q(s',a') - Q(s,a)]$
　　$s \leftarrow s';\ a \leftarrow a';$
　　s が終端状態なら繰り返しを終了

同様に，Sarsa(λ) は TD(λ) での状態価値関数を行動価値関数に置き換えたものとなる．オンライン Sarsa(λ) の擬似コードは以下のようになる．

$Q(s,a)$ を任意に初期化し, すべての 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 を初期化
　Q から導かれる方策を用いて s で取る行動 a を選択
　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 から導かれる方策を用いて s' で取る行動 a' を選択
　　$\delta \leftarrow r + \gamma Q(s',a') - Q(s,a)$
　　$e(s,a) \leftarrow e(s,a) + 1$
　　全ての s,a に対して：
　　　$Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
　　　$e(s,a) \leftarrow \gamma\lambda e(s,a)$
　　$s \leftarrow s';\ a \leftarrow a';$
　　s が終端状態なら繰り返しを終了

2.3　ニューラルネットワーク

ニューラルネットワークとは脳の神経を数学的に記述することを目指して作られた数学的モデルの一種である [14]．図 2.4 は 3 層ニューラルネットワークの構成となっている．3 層ニューラルネットワークは入力層，中間層，出力層の 3 つの層より構成される．ここでは入力層が n 個のユニット，中間層が m 個のユニット，出力層が 1 個のユニットからなる 3 層ニューラルネットワークを考える．x_1 から x_n は入力層の各ユニットの出力値，y_1 から y_m は中間層の各ユニットの出力値，z は出力層のユニットの出力値である．v_{ij} は入力層の i 番目のユニットと中間層の j 番目のユニットの結合の重みであり，w_j は中間層の j 番目のユニットと出力層のユニットの結合の重みである．本研究では x_n，y_m をバイアス項として常に -1 の値を出力させることとし，バイアス項を除いた中間層の数はバイアス項を除いた入力層の数を 2 で割った値 $+1$ とした．

入力として入力層に $x_1, x_2, \ldots, x_n$ が与えられた際の中間層の j 番目のユニットにおける出力は以下の式となる:

$$y_j = f(\sum_{i=0}^{n} x_i v_{ij}). \tag{2.21}$$

さらに，出力層のユニットにおける出力は以下の式となる:

$$z = f(\sum_{k=0}^{m} y_k w_k). \tag{2.22}$$

このようにして，入力が与えられた際の出力 z を求めること計算処理をフォワードプロパゲーションと呼ぶ．式 (2.21),(2.22) における関数 f は活性化関数と呼ばれており, 以下の

式で示される標準シグモイド関数がよく用いられる:

$$f(x) = \frac{1}{1+e^{-x}}. \tag{2.23}$$

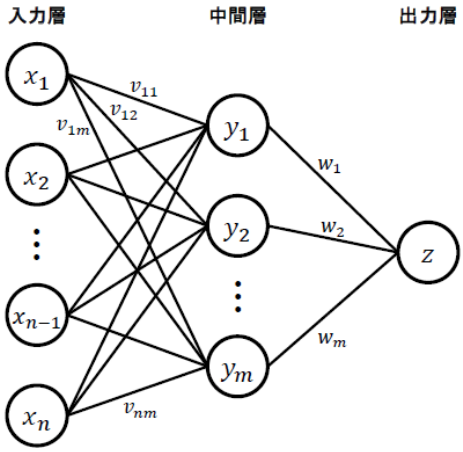


図 2.4: 3 層ニューラルネットワーク

2.3.1 バックプロパゲーション

バックプロパゲーション (誤差逆伝播法) はニューラルネットワークの重みや閾値を調整する教師付き学習アルゴリズムの一種である. バックプロパゲーションで学習を行うには学習パターンが必要となる. 学習パターンはニューラルネットワークの入力層に与える入力 $\boldsymbol{x}$ と入力に対応する出力である理想値 z_d の組からなる.

3 層ニューラルネットワークにおけるバックプロパゲーションのアルゴリズムは以下のようになっている.

入力 $x_1, x_2, \ldots, x_n$ が与えられた時，フォワードプロパゲーションにより得た出力を計算値 z_c，理想値を z_d とする．なお，学習係数 η には適当な値を与え，学習率 $\alpha = \frac{1}{n}, \beta = \frac{1}{m}$ とする．

Step1 重み w_j を更新するのに使う，出力層のユニットにおける学習信号 σ を以下の式とする:

$$\sigma \leftarrow z_c(1 - z_c)(z_d - z_c). \tag{2.24}$$

Step2 中間層の j 番目のユニットと出力層のユニットの間における重みの修正量 Δw_j を次の式で更新する:

$$\Delta w_j \leftarrow \eta\sigma y_j + \beta\Delta w_j. \tag{2.25}$$

Step3 中間層の j 番目のユニットと出力層のユニットの間における重み w_j を次の式で修正する:

$$w_j \leftarrow w_j + \Delta w_j. \tag{2.26}$$

Step4 重み v_{ij} を更新するのに使う，中間層の j 番目のユニットにおける学習信号 τ_j を次の式で求める:

$$\tau_j \leftarrow y_j(1 - y_j)\sigma w_j. \tag{2.27}$$

Step5 入力層の i 番目のユニットと中間層の j 番目のユニットの間における重みの修正量 Δv_{ij} を次の式で求める:

$$\Delta v_{ij} \leftarrow \eta\tau_j x_i + \alpha\Delta v_{ij}. \tag{2.28}$$

Step5 入力層の i 番目のユニットと中間層の j 番目のユニットの間における重み v_{ij} を次の式で修正する:

$$v_{ij} \leftarrow v_{ij} + \Delta v_{ij}. \tag{2.29}$$

学習パターンを N 個用意したとする．バックプロパゲーションにおける学習では N 個の学習パターンそれぞれに対してフォワードプロパゲーションで得た値を用いてバックプロパゲーションを行う．バックプロパゲーションではニューラルネットワークと理想の値 z_c の平均二乗誤差を小さくする．平均二乗誤差は以下の式で表される

$$\frac{\sum_{i=1}^{N}(F(\boldsymbol{x}) - z_d)^2}{N}. \tag{2.30}$$

平均二乗誤差は N 個の学習パターンそれぞれに対して学習パターンの出力 z_d と学習パターンの入力 $\boldsymbol{x}$ をニューラルネットワークに与えた時の出力 $F(\boldsymbol{x})$ の二乗誤差の総和を学習パターン数で割ったものである．平均二乗誤差が小さくなることは，N 個の学習パター

ンにわたり，z_d と $F(\boldsymbol{x})$ が接近することを意味する．

バックプロパゲーションを繰り返すことにより，学習パターンの入力を与えた際のニューラルネットワークの出力である計算値 z_c と学習パターンの出力の理想値 z_d の誤差の二乗が小さくなり，ある入力を与えることにより対応すべき出力を返す関数を近似的に得ることができる．

2.3.2 ニューラルネットワークを用いた Sarsa(λ)

実際のガイスターにおいて，取りうる状態数は非常に大きな値となり，各状態行動対に対する行動価値関数を配列に保存することは現実的ではない．よって，行動価値関数をニューラルネットワークを用いて近似する．

Sarsa(λ) とは行動価値観数を学習させるため TD(λ) における状態価値関数を行動価値関数に置き換えたものであった．Sarsa(λ) におけるバックプロパゲーションは以下のようになる．

Step1 現状態における行動価値 $Q(s,a)$ および次状態における行動価値 $Q(s',a')$，次状態での報酬 r を用いて，TD 誤差 δ を以下の式で求める

$$\delta \leftarrow r + \gamma Q(s',a') - Q(s,a). \tag{2.31}$$

Step2 TD 誤差 δ，中間層の j 番目のユニットから出力層への適格度トレース e_{w_j} から，中間層の j 番目のユニットと出力層の間における重み w_j の修正量 Δw_j を求める

$$\Delta w_j \leftarrow \beta\delta e_{w_j}.$$

Step3 中間層の j 番目のユニットと出力層のユニットの間における重み w_j を修正する

$$w_j \leftarrow w_j + \Delta w_j.$$

Step4 TD 誤差 δ，入力層の i 番目のユニットから中間層の j 番目のユニットを通り出力層へ至る適格度トレース $e_{v_{ij}}$ から，入力層の i 番目のユニットと中間層の j 番目のユニットの間における重み v_{ij} の修正量 Δv_{ij} を求める

$$\Delta v_{ij} \leftarrow \alpha\delta e_{v_{ij}}.$$

Step5 入力層の i 番目のユニットと中間層の j 番目のユニットの間における重み v_{ij} を修正する

$$v_{ij} \leftarrow v_{ij} + \Delta v_{ij}.$$

バックプロパゲーションによるニューラルネットワークの重み更新後，再度，フォワードプロパゲーションを行い $Q(s,a)$ を求める．その後，次の適格度トレースの更新を行う．

Step1 フォワードプロパゲーションにより求めた行動価値 $Q(s,a)$ の計算値 z を用いてシグモイド関数の導関数値 σ を求める

$$\sigma \leftarrow z(1-z). \tag{2.32}$$

Step2 σ と中間層の j 番目のユニットの出力値 y_j を用いて中間層の j 番目のユニットから出力層への適格度トレース e_{w_j} を以下の式で更新する

$$e_{w_j} \leftarrow \lambda e_{w_j} + \sigma y_j.$$

Step3 中間層の j 番目のユニットの出力値 y_j を用いてシグモイド関数の導関数値 τ_j を求める

$$\tau_j \leftarrow y_j(1-y_j).$$

Step4 σ，τ と入力層の i 番目のユニットの出力値 x_i を用いて入力層の i 番目のユニットから中間層の j 番目のユニットを通り出力層へ至る適格度トレース $e_{v_{ij}}$ を以下の式で更新する．

$$e_{v_{ij}} \leftarrow \lambda e_{v_{ij}} + \sigma w_j \tau_j x_i.$$

以下に，本研究で用いるガイスターにおける自己対戦での学習アルゴリズムを示す．

1: ニューラルネットワークの重みを初期化し，適格度トレース e_v, e_w を 0 とする
2: 各対局に対して繰り返し：
3: 　両方の初期配置をランダムに決定
4: 　方策を用いて局面 s で手 a を選択
5: 　対局における各ターンに対して繰り返し：
6: 　　手 a を取り,r, s' を観測する
7: 　　方策を用いて s' で手 a' を選択
8: 　　$\delta \leftarrow r + \gamma(1 - Q(s', a')) - Q(s, a)$
9: 　　δ を用いてバックプロパゲーションによる重みの更新
10: 　　フォワードプロパゲーションにより計算値 z を算出
11: 　　z を用いた適格度トレースの更新
12: 　　$s \leftarrow s';\ a \leftarrow a';$
13: 　　局面 s にてゲームが終了しているならば繰り返しを終了

式 (2.31) における一般的な TD 誤差と自己対戦における学習アルゴリズムの 8 行目での

TD 誤差 δ は違う目標値を取っている．前者における目標値は $r + \gamma Q(s', a')$ であり，後者における目標値は $r + \gamma(1 - Q(s', a'))$ となる．後者における TD 誤差はガイスターが二人対戦ゲームであるためにこのような値を取る．後者では，ゲームの途中において $r = 0$ とするために，$\gamma(1 - Q(s', a'))$ を目標とし，行動価値関数を更新する．s' は手を指した後の相手プレイヤから見た局面となっており，a' は s' における相手の手となっている．$Q(s', a')$ は相手プレイヤが局面 s' において手 a' を指した時の相手プレイヤの勝率の見積もりとなる．つまり，$1 - Q(s', a')$ は相手プレイヤが局面 s' において手 a' を指した時の自分の勝率の見積もりを表す．つまり，自己対戦における学習アルゴリズムでは，相手プレイヤが手を指した後の勝率の見積もりを目標として状態価値関数を更新する．

2.4 df-pn アルゴリズムによる必勝手探索

df-pn アルゴリズム [15] とは，深さ優先探索と証明数および反証数を組み合せた AND/OR 木探索である．df-pn アルゴリズムは，証明数と反証数を閾値として利用し，最有力ノードを常に展開するという証明数探索と同じ特徴を持つ．df-pn アルゴリズムは詰将棋を解くプログラムなどへの応用でよく用いられる．ガイスターは不完全情報ゲームであるため，AND/OR 木探索を行なうことができない．

2.4.1 ガイスターの完全情報ゲーム化

ガイスターにおいて，AND/OR 木探索を行なうことが出来ないため，相手の駒を紫色の駒とすることにより，完全情報ゲームとする．紫色の駒は以下のように扱う．

- 紫色の駒は青駒と同様に出口から脱出することが可能である
- 紫色の駒を取った場合，赤駒を取ったものとして扱う

ガイスターにおいて，両プレイヤは常に相手の盤上にある青駒と赤駒の数がわかった状態にある．さらに，相手の各駒が取りうる色の組合せを求めることができる．例えば，相手の駒が盤上に 2 つある場合，相手の駒は青駒 1 つと赤駒 1 つとなる．2 つの駒を駒 A と駒 B と区別すると，駒 A が青で駒 B が赤であるパターンと駒 A が赤で駒 B が青であるパターンの 2 つが考えられる．以後，このような各駒に対する整合性のある色の割り振りを配色パターンと呼ぶ．

相手の駒を紫色とした完全情報ガイスターは考えうる全ての配色パターンとなる完全情報ゲームよりもプレイヤにとって不利なゲームとなる．よって，相手の駒を紫色とした完全情報ガイスターにおいて必勝となる手はどのような配色パターンにおいても必勝となる．よって，相手の駒を紫色とした完全情報ガイスターにおいて df-pn アルゴリズムによる必勝手を見つける探索を行うことは元のガイスターにおける必勝手探索となる．

第3章　既存研究

本章では，ガイスターで用いられている駒の推測手法である PrototypeBasedLearning と探索手法であるモンテカルロ木探索，TD-Gammon について説明する．

3.1 Prototype-Based Learning

概念記述 (concept description) とは，与えられたデータの集合の簡潔な要略を行う特徴付けと 2 つ以上のデータの集合から記述を比べる比較からなる．Prototype-Based Learning(PBL)[10] はこの概念記述を小さな学習データからでさえ学習可能で, 不適当なデータさえ耐えうる機械学習手法であり，認知心理学上の理論であるプロトタイプ理論に基づいている．プロトタイプ理論とは，人間が実際にもつカテゴリは，必要十分条件によって規定される古典的カテゴリではなく，典型事例とそれとの類似性によって特徴づけられるという考え方から成る．

既存手法では, 青駒と赤駒の 2 つの駒に対して複数の特徴からなるプロトタイプの特徴ベクトルを求め，その 2 つのプロトタイプベクトルとの類似性によって青駒か赤駒かの推測を行っており [11]，本研究でも同様の方法を用いる．ゲームの棋譜中に現れる全ての青駒と赤駒に対する初期配置やどういった動きをしたかなどの特徴で構成される特徴ベクトルを求め，それを平均化することにより青駒と赤駒のプロトタイプとする．

ゲーム中に現れる青駒の特徴ベクトルの集合が $B = \{b_1, \cdots, b_n\}$，赤駒の特徴ベクトルの集合が $R = \{r_1, \cdots, r_n\}$ と与えられるとする．

この時の青駒，赤駒のプロトタイプベクトルは以下のように計算される

$$P_B = \frac{1}{n}\sum_{i=1}^{n} b_i, P_R = \frac{1}{n}\sum_{i=1}^{n} r_i.$$

今，色がわからない駒の特徴ベクトル f が与えられたとき，その駒に対する推測値は以下のように定義する

$$s(f) = \frac{d(f, P_B) - d(f, P_R)}{d(f, P_B) + d(f, P_R)}.$$

この式における $d(x, y)$ はベクトル x，y のユークリッド距離となる．この式によって求められる推測値は f が P_B に近いほど -1，P_R に近いほど 1 に近い値を取る．つまり，ある駒の推測値の値が P_B に近いほど青らしく，P_R に近いほど赤らしいと推測をする．

本論文において，表3.1に示される18個の特徴から特徴ベクトルを構成した．なお，太字で表される特徴が既存手法から新たに追加した特徴となっている．

表 3.1: プロトタイプベクトルを構成する特徴

特徴	説明	取りうる値
初期配置 (8 個)	ある駒の初期配置が 8 マスの内どの位置にいるかを表す特徴．8 個の特徴がそれぞれのマスにに対応しているため，駒のいるマスに対応している特徴には 1 を，それ以外には 0 を与える．	0,1
一手目の駒	一手目に動かした駒を表す特徴．一手目に動かした駒である場合 1，そうでない場合 0 となる．	0,1
二手目の駒	二手目に動かした駒を表す特徴．二手目に動かした駒である場合 1，そうでない場合 0 となる．	0,1
前進した回数	駒を前に進めた回数を表す特徴.	0 以上の整数
後退した回数	駒を後ろに下げた回数を表す特徴.	0 以上の整数
横に移動した回数	駒を左右に移動した回数を表す特徴.	0 以上の整数
駒を取った回数	この駒を動かすことで相手の駒を取った回数を表す特徴.	0 以上の整数
相手の駒との隣接状態から抜けた回数	相手の駒と上下左右で隣り合っている状態からに手を指すことより，隣接しない状態にした回数	0 以上の整数
相手の駒との隣接状態を保持した回数	相手の駒と上下左右で隣り合っている状態から手を指さずに，隣接状態を維持した回数を表す特徴.	0 以上の整数
隣接状態から動かし別の隣接状態にした回数	相手の駒と上下左右で隣り合っている状態から手を指すことにより，違った相手の駒と隣接する状態にした回数を表す特徴	0 以上の整数
非隣接状態から隣接状態にした回数	相手の駒との非隣接状態から手を指すことにより，相手の駒と隣接させた回数を表す特徴.	0 以上の整数

本研究において，GhostsChallenge における上位の AI プレイヤ同士の対戦における棋譜データを利用し，これらの特徴を持つ青駒と赤駒のプロトタイプベクトルを学習させる．なお，学習にあたりゲームの結果が引き分けになる棋譜データは千日手によって駒の移動の回数が大きくなりすぎる．よって，引き分けを除く 839 試合を学習用のデータとする．

このプロトタイプを用いた駒の色に対する推測値を入力の一部とし，自己対戦による Sarsa(λ) 学習を行う．また，モンテカルロ木探索にも同様の推測値を用いる．

3.2　モンテカルロ法

数値計算におけるモンテカルロ法とは，乱数を用いて n 回のシミュレーションを行い，ある事象が n 回のシミュレーションを行い，ある事象が m 回起これば，その事象が起きる確率を m/n で近似するものである．ゲームにおいては，ある局面における最も勝率が高い手を求めるため，手を指した際の勝率の見積もりをシミュレーションを行うことが多い．このシミュレーションにおいて，両プレイヤにゲームが終了するまでランダムに手を指させる．このゲーム終了までの 1 回のシミュレーションをプレイアウトと呼ぶ．沢山のプレイアウトを行うことで勝率の見積もりはより正確になる．モンテカルロ法は局面の優位性を表現する評価関数を正確に求めることが出来ないゲームにおいてよく用いられる．

3.3　モンテカルロ木探索

モンテカルロ木探索とは，モンテカルロ法に木探索を組合せた手法である．評価関数を用いずプレイアウトによってゲーム木の探索を行う．各合法手に対してプレイアウトを何度か行い，得られたスコアを集計することにより，有望でありそうな手に対して多くのプレイアウトを割り当て，各節点でプレイアウトが行われた回数がある閾値を超えた場合にはその手を展開し，プレイアウトを開始する節点を 1 つ深くする．その一連の流れか図 3.1，3.2 となっている．

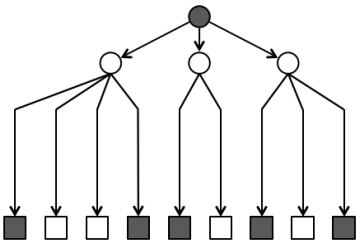


図 3.1: プレイアウトの割り当て

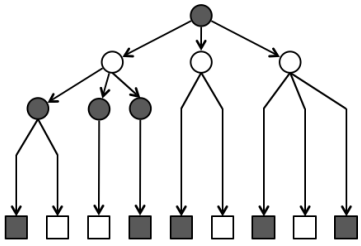


図 3.2: 節点を 1 つ深く読む様子

図 3.1 は将来有望な手に多くの多くのプレイアウトを割り当てている様子であり，ルートか深さ 2 の最も左にある節点へ遷移する手に最も多くのプレイアウトが割り当てられている．図 3.2 では深さ 2 の最も左にある節点でのプレイアウト回数が閾値を超えたため手が展開されている．

モンテカルロ木探索は，プレイアウトにゲーム特有の知識を導入したり，プレイアウト回数を増やすことで性能が向上するという特徴がある．

3.3.1　UCT アルゴリズム

モンテカルロ木探索を行うとき，有望な手に多くのプレイアウトを割り当てる選択基準として UCB1 値を採用したものが UCT(Upper Confidence bound applied to Trees) アル

ゴリズム [12] である．このアルゴリズムにより，勝率の高いノードにプレイアウトを集中させながらも，その他の勝率が実際よりも低く見積もられた可能性があるプレイアウトの少ない手を探索することが可能となる．

UCB1 値は，手 i の勝率 $\overline{x_i}$，手 i に対して行われたプレイアウト数 n_i，その局面の総プレイアウト数 n が与えられた時に以下の式で求められる

$$\mathrm{UCB}(i) = \overline{x_i} + \sqrt{\frac{2\log n}{n_i}}. \tag{3.1}$$

式 3.1 の第一項は手の勝率であり，これにより勝率の高い候補手が選択されやすくなる．第二項はバイアス項であり，着手の回数が少ないほど大きくなるため，プレイアウト数の少ない手がより選択されやすくなる．よって，有望とされる手に多くプレイアウトを割り振りながらも，プレイアウトの少ない手に対しても探索を行うことができる．

UCT アルゴリズムは，適切な仮定の下において，プレイアウト数が無限大になれば最善手が発見できることが証明されている [13]．

既存手法において，相手の全ての駒に対し，特徴ベクトルから推測値を算出し，推測値の高い駒から順に赤，それ以外を青と判断することで本来不完全情報ゲームであるガイスターを完全情報ゲームとして扱い UCT アルゴリズムを用いたモンテカルロ木探索を行う AI プレイヤが開発されており，現在最強のガイスター AI プレイヤとなっている．

しかし，この手法を用いた AI プレイヤには欠点が 2 つある．一つは推測値を用いて駒の色の推定を行う際に各駒の推測値に対する大小関係のみを利用しており，推測値自体の大きさを活かすことが出来ていない点である．相手の駒 1 つの推測値が最も高く赤だと推測され，他のいくつかの駒の推測値がこれよりも低い値でほとんど差がなく，相手の赤駒が 2 つ残っている局面を考える．このとき，まず推測値が最も高い駒を赤だと推測し，推測値がほとんど差がない駒の中から 1 つが赤駒だと推測される．推測された 2 つの赤駒は推測値が大きく離れているが，全く同じ赤駒として扱われることとなる．さらに，推測値にほぼ差がない駒なのにもかかわらず赤駒と青駒に区別される．どれほど赤らしいのか青らしいのかというのは非常にガイスターをプレイする上で非常に重要な情報であるが，推測値から完全情報ゲームとし，モンテカルロ木探索を行なう手法では推測値の大きさを活かせていないのが問題となる．

二つ目は不完全情報ゲームを完全情報ゲームとして扱うことが問題だということである．不完全情報ゲームであるガイスターにおいて駒の色を推定し，全ての駒の色が完全に的中していたとしても，完全情報ゲームとして扱ったガイスターにおける探索では最善手を指すことが出来ない．例えば，推測が完全に当たっているが非常に劣勢となっている局面を考える．この場合のモンテカルロ木探索では，勝率は低いものの中から最も勝率の高い手を選択する．しかし，ガイスターでは，ある手によって相手を騙すことに成功し，大幅に勝率を上げる手が存在する可能性がある．よって，ガイスターにおいての最善手は相手の正確な心理モデルがなければ求めることが出来ない．

3.4 TD-Gammon

Tesauro はバックギャモンを学習するプログラム TD-Gammon の開発を目的とした研究成果を報告した．TD-Gammon とはバックギャモンにおいて自分自身と対戦し，その結果からバックギャモンにおける局面評価関数となるようなニューラルネットワークを用いた AI プレイヤである．局面評価関数は TD(λ) により学習している．TD-Gammon は三層ニューラルネットワークを用いており，入力層のユニット数は 198，中間層のユニット数は TD-Gammon のバージョンにより異なり 40 ～ 160，出力層のユニット数は 1 となっている．

TD-Gammon の学習にはバックプロパゲーションが用いられている．TD-Gammon を学習させるためには学習パターンが必要となる．学習パターンの入力はゲームの状態に対して合法手を指した後の事後状態であり，学習パターンの出力は事後状態からの勝敗となる．

Tesauro は学習パターンを生成するために TD-Gammon に自己対戦を行わせた．自己対戦の開始時において，ニューラルネットワークの重みや閾値はランダムな小さい値に設定された. したがって，自己対戦の初期において，TD-Gammon が計算した評価値はランダムな値となる．この評価値により手を選択するため，自己対戦の初期においては弱い手しか指すことができない．しかし，数十回ゲームを行うことで性能は急激に向上した．

自己対戦を 300,000 回行った TD-Gammon0.0 は当時最強のバックギャモン AI プレイヤと互角に対戦できるほどに強化された．当時最強のバックギャモン AI プレイヤはバックギャモンにおける膨大な知識を利用していたので，バックギャモンについての知識を与えていない TD-Gammon が同等の強さになったことは驚くべき結果であった．

そして，TD-Gammon0.0 にバックギャモン固有の知識を導入した TD-Gammon1.0 が開発された．この TD-Gammon1.0 は従来のバックギャモン AI プレイヤ全てに対して，圧倒的な強さを見せ，人間のエキスパートプレイヤと互角に戦うことができた．

TD-Gammon2.0 では選択的 2 段階探索が導入された．選択的 2 段階探索とは選んだ手の直後のゲーム状態だけではなく，相手の出すサイコロの目と，それぞれに対応する合法手についても先読みを行うことである．その際，相手は最善手を選択すると仮定した．計算時間を節約するため，高く評価された 4 手から 5 手のみについて選択的 2 段階探索を行った．その結果，TD-Gammon2.0 はグランドマスターレベルに至った．

TD-Gammon3.0 では中間層のユニットを 160 個とし，選択的 3 段階探索が導入された．TD-Gammon3.0 は世界最高の人間プレイヤと同程度の強さを持っていると思われており，すでに世界チャンピオンとなっている可能性もある．

TD-Gammon の登場により，世界最高クラスの人間プレイヤも知らなかった定石が発見され，その定石に基づいた手を人間の最高クラスのプレイヤが使うようになった．このように，TD-Gammon は人間プレイヤのプレイの仕方を変えるなどの影響を与え，バックギャモンの文化の発展に貢献した．

第4章　Minimum-Geisterにおける実験

本章では，MC と TD(0) がガイスターに対して有効かどうかを調べるため，小さなサイズのガイスターにおいて両手法での学習を行う．MC と TD(0) で最善戦略を求めることができるかを確認するため，予め最善戦略がわかり，ゲームの履歴を持つことができる2つの小さなガイスターを定義する．さらに，最善戦略はわからないがゲームの履歴を持つことができる小さなガイスター1つを定義する．ここでは，それら3つのガイスターを用いて，状態価値関数および行動価値関数を求める実験を行い，MC と TD(0) で最善戦略を求めることができるかを確認する．また，局面の履歴を用いた行動価値関数と用いていない行動価値関数を用意し，比較を行うことで履歴の有用性についても確認する．

4.1　3つの Minimum-Geister

ガイスターの基本ルールに基づき以下のような **Minimum-Geister1(以下 MG1)** というゲームを定義する．

- 盤のサイズは 2×3
- 出口は 1 つずつ
- 両プレイヤ共に青駒 1 つずつを持つ
- 駒の初期配置は固定
- 先手番は固定
- 両プレイヤの合計手数が 8 手になった時点で勝敗がついていない場合, 引き分け

お互いの駒の色は青となっている．よって，常に相手の駒の色がわかる状態となる．つまり，このゲームは完全情報ゲームである．図 4.1 は MG1 における初期局面である．

さらに，MG1 のルールを 1 つ変更した **Minimum-Geister2(以下 MG2)** を定義した．MG2 を定義するため，MG1 において変更するルールは「両プレイヤ共に青駒 1 つずつを持つ」であり，これを「**両プレイヤはゲーム開始時にゲームで用いる駒 1 つの色を青か赤から選択する**」とする．よって，以下の様な条件となる．

- 盤のサイズは 2×3
- 出口は 1 つずつ

- 両プレイヤはゲーム開始時にゲームで用いる駒 **1** つの色を青か赤から選択する
- 駒の初期配置は固定
- 先手番は固定
- 両プレイヤの合計手数が 8 手になった時点で勝敗がついていない場合，引き分け

MG2 において，相手の駒の色がわからないため不完全情報ゲームとなる．図 4.2 は MG2 における先手番から見た初期局面の例である．不完全情報ゲームであるため，後手番の駒の色は確認できない状態となる．

MG2 における合法手にパス手を加えた **Minimum-Geister3(以下 MG3)** というゲームも定義する．

- 盤のサイズは 2×3
- 出口は 1 つずつ
- 両プレイヤはゲーム開始時にゲームで用いる駒 1 つの色を青か赤から選択
- 相手の駒の色は駒を取るまたは駒が出口から出るまでわからない
- 先手番は固定
- 両プレイヤの合計手数が 8 手になった時点で勝敗がついていない場合, 引き分け
- 合法手としてパス (駒を動かさない) が可能

図 4.1: MG1 における初期局面

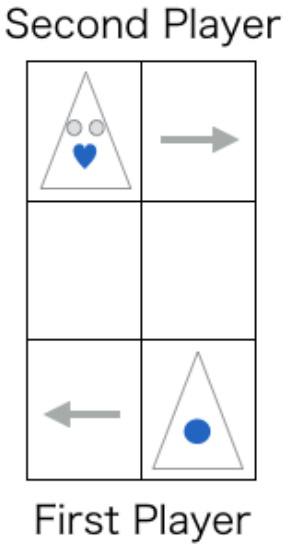
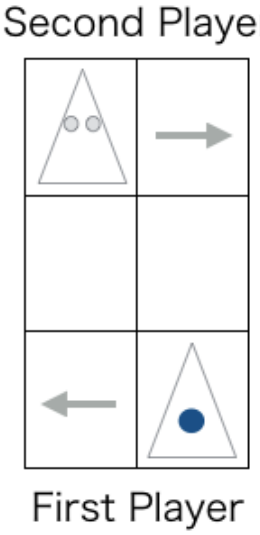


図 4.2: MG2 における先手番から見た局面の例



4.2 Minimum-Geisterにおける最善戦略

MG1 および MG2 においての最善戦略を求める上でこれらのゲームにおける重要な法則が存在する．それは２つの駒のマンハッタン距離 (以下 MD) の偶奇性により, そのゲームにおける，プレイヤの相手への駒取りを行えるかが決まるという法則である．具体的には以下の定理となる．

定理 1. 手番をパスすることができない MG1 および MG2 において, ２つの駒のマンハッタン距離 MD により，そのゲーム中に相手の駒を取ることが行えないプレイヤが決定する．

もし MD が奇数ならば,現在手番であるプレイヤはゲームを通して駒取りを行えない
もし MD が偶数ならば,現在手番でないプレイヤはゲームを通して駒取りを行えない

証明. MD = 奇数 の局面において，手番プレイヤが手を指すと次局面では必ず MD = 偶数となる．また，MD = 偶数 の局面においても同様に，手番プレイヤが手を指すと次局面では必ず MD = 奇数 となる．よって，常に片側のプレイヤの手番における MD の偶奇はゲームを通して一致する．駒が取られた局面において MD = 0 であり，MD = 1 の局面における手番のみが駒取りを行うことができる．つまり，手番で MD が奇数にならないプレイヤはそのゲームを通して必ず駒取りを行えない．さらに，非手番で MD が偶数にならないプレイヤはそのゲームを通して必ず駒取りを行えない． □

Minimum-Geister において，初期局面で MD = 3 となるため，MD の偶奇が保たれる MG1，MG2 では初期局面の後手番プレイヤは必ず駒を取ることが出来ない．

MG1 において，先手番プレイヤは前に駒を進めることによって，後手番プレイヤは前もしくは横に駒を進めなくてはならない．しかし，どちらに駒を進めようとも，進めた後に駒取りによって負けることになる．つまり，MG1 は先手勝ちのゲームとなる．

MG2 において，後手番プレイヤが青駒を選択した場合，MG1 と同様の方法で負けてしまう．よって，後手番プレイヤが赤駒を選択することがよりよい戦略だと考えられる．一方，先手番プレイヤは定理 1 より，自分の駒を取られない．つまり，相手に青駒を取られて負けることもなければ，相手に赤駒を取らせて勝つことも出来ない．表 4.1 は MG2 における先手番プレイヤが青駒を選択した時と赤駒を選択した時のそれぞれにおける勝利条件と敗北条件である．

表 4.1: MG2 における先手番プレイヤの駒の色に対する勝利条件と敗北条件

	勝利条件	敗北条件
青駒を選ぶ	・自分の青駒を脱出させる ・相手の青駒を取る	・相手の青駒に脱出される ・相手の赤駒を取る
赤駒を選ぶ	・相手の青駒を取る	・相手の青駒に脱出される ・相手の赤駒を取る

この表より，赤駒を選ぶことは勝利条件を少なくするデメリットしかないことが明らかであるため，先手番プレイヤは青駒を選択することがより良い戦略となる．先手番プレイヤは青駒を選択して初めに駒を前進させたとき，後手番プレイヤが駒を前進させてしまうと先手番プレイヤは駒を脱出させて勝つことができる．しかし，後手番プレイヤは赤駒を出口の上に乗せることで脱出を阻止することができる．つまり，先手番プレイヤはこの赤駒を取ってしまうと負けてしまうため，このまま先手番プレイヤが脱出をするための隙を覗い，後手番プレイヤは出口上に駒を乗せて脱出を防ぎながら，引き分けを迎えるのが互いにとっての最善戦略となる．

MG3 においては，パスが可能になっているため，定理 1 は使えず，最善戦略も事前にはわからない．

4.3 MG1 における状態価値関数の学習

使う駒が青駒のみで完全情報ゲームとなっている MG1 において，ランダム方策とグリーディー方策を用いた際の MC，TD(0) での状態価値関数 $V(s)$ を求める．ランダム方策とは可能な手の中からランダムに着手を選択するという方策であり，グリーディー方策とは次状態の状態価値関数が最も高い着手を選択する方策である．

実験

まずゲーム中に発生する局面を事前に全列挙する (全 62 局面)．その各々に対して方策を用いて，状態価値関数の更新を一度行う．この更新をランダム方策を用いた MC，TD(0) の両方で $1,000,000$ 回，グリーディー方策を用いた MC，TD(0) の両方で 1,000 回繰り返し，状態価値関数の学習を行う．

結果

ランダム方策とグリーディー方策を用いた際の MC と TD(0) における初期局面の先手番に対する勝率は以下のようになる．

表 4.2: 各方策を用いた際の MC，TD(0) における初期局面での先手番の勝率

	MC	TD(0)
ランダム方策	0.84302	0.83174
グリーディー方策	0.9999	0.9995

ランダム方策では MC，TD(0) の両方において勝率が約 8 割となった．これは MG1 が定理 1 により先手有利となっていることを考えると妥当である．一方，グリーディー方策では勝率が約 10 割となり．先手必勝であることと先手が初手で青駒を前進させる手が必勝手であることが求められた．

4.4 MG2における状態価値関数と行動価値関数の学習

初期局面において駒の色を選択できる不完全情報ゲームである MG2 において，ランダム方策を用いた際の MC，TD(0) での状態価値関数 $V(s_t)$ とグリーディー方策を用いた際の MC，TD(0) での行動価値関数 $Q(s_t, a)$ および $Q(s_0, s_1, s_2, \cdots, s_t, a)$ を求めた．

MG1 における実験と同様にゲーム中に発生する全局面を列挙し，その各々に対して一度ずつ状態価値関数の更新を行っていく．具体的には，互いに色がわかっている状態におけるゲーム木の全局面を列挙する．その全ての局面において，手番プレイヤが相手の駒の色を認識できない局面 s_t もしくはルートからの局面系列 $s_0, s_1, \cdots, s_t$ を生成し，局面における $V(s_t)$ もしくは $Q(s_t, a)$，$Q(s_0, s_1, s_2, \cdots, s_t, a)$ を更新する．

状態価値関数 $V(s_t)$ における s_t のとりうる状態数は履歴を考慮しない片側不完全情報となる局面の数と一致し，259 となる．更新を開始するために用いる両側完全情報ゲームとした際の履歴を考慮した局面数は 860 となる．

片側不完全情報ゲームとした際の履歴込みの局面と手の可能な組合せは計 2037 となる．つまり，これは行動価値関数 $Q(s_0, s_1, s_2, \cdots, s_t, a)$ の引数の組合せの数となる．

実験

ランダム方策を用いた MC において 1 つの局面に対して状態価値関数の更新を 100,000 回，TD(0) では 1,000 回行う．

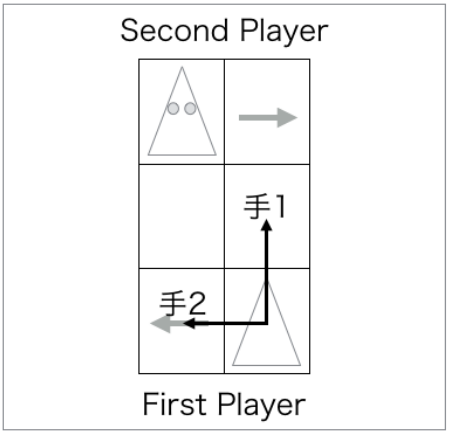
さらに，グリーディー方策を用いた MC での $Q(s_t, a)$ の更新では 3,000 回，$Q(s_0, s_1, s_2, \cdots, s_t, a)$ では 1,000 回の更新を行う．また，グリーディー方策を用いた TD(0) での $Q(s_t, a)$ と $Q(s_0, s_1, s_2, \cdots, s_t, a)$ の更新では 1,000 回の更新を行う．

結果

全ての局面，手における状態価値関数および行動価値関数を求めるが，特に図 4.3 における局面および図 4.4 の手に対しての結果を見る．

図 4.3: 局面 1,2,3,4

図 4.4: 手 1,2



ランダム方策を用いた際の MC と TD(0) における片側不完全情報とした 4 つの初期局面の先手番に対する状態価値関数 $V(s_t)$ は表 4.3 となる．

表 4.3: ランダム方策を用いた際の MC,TD(0) における 4 つの初期局面での先手番の勝率

	MC	TD(0)
局面 1	0.57316	0.50597
局面 2	0.47536	0.49885
局面 3	0.78338	0.82746
局面 4	0.32598	0.16268

MC と TD(0) の方策の両方で局面 1 と局面 2 の両方で状態価値関数 $V(s_t)$ はほぼ 5 となり，状態価値関数の値が完全に一致しないのは乱数と更新回数の差と MC と TD(0) の収束性に違いがあるからだと考えられる．

グリーディー方策を用いた際の MC と TD(0) における片側不完全情報とした 4 つの初期局面の先手番に対する $Q(s_t, a)$ は表 4.4 となる．

表 4.4: グリーディー方策を用いた際の MC,TD(0) における 2 つの初期局面での先手番の手に対する $Q(s_t, a)$

	MC	TD(0)
局面 1+手 1	0.7513	0.5000
局面 1+手 2	0.2487	0.4737
局面 3+手 1	0.5000	0.5000
局面 3+手 2	0.5000	0.4737

グリーディー方策を用いた際の MC と TD(0) における片側不完全情報とした 4 つの初期局面の先手番の手に対する $Q(s_0, s_1, s_2, \cdots, s_t, a)$ は表 4.5 となる．

表 4.5: グリーディー方策を用いた際の MC,TD(0) における 2 つの初期局面での先手番の手に対する $Q(s_0, s_1, s_2, \cdots, s_t, a)$

	MC	TD(0)
局面 1+手 1	0.7632	0.5000
局面 1+手 2	0.2368	0.4737
局面 3+手 1	0.5000	0.5000
局面 3+手 2	0.5000	0.4737

$Q(s_t, a)$，$Q(s_0, s_1, s_2, \cdots, s_t, a)$ において，ほぼ同様の結果となった．両者の MC における局面 1+手 1 と局面 1+手 2 の結果に差があることは更新回数による違いによるものである．

MC と TD(0) において, 結果が異なることとなったが, それぞれの次局面を末端局面まで辿っていった所，両者における勝率の計算は妥当であった. 完全情報でのガイスターにおけるゲーム木の全ての局面を用いて，行動価値関数の更新を行っているせいであると考えているが，なぜこのような結果になるのかを解き明かすには更なる考察が必要である.

4.5 MG2 での ϵ-グリーディー方策による行動価値関数の学習

MG2 において ϵ-グリーディー方策を用いた時の MC と TD(0) における行動価値関数 $Q(s_t, a)$ および $Q(s_0, s_1, s_2, \cdots, s_t, a)$ を求める．ϵ-グリーディー方策とは, 確率 ϵ でランダム，確率 $1-\epsilon$ でグリーディーに行動 a を選ぶ方策である．今回は $\epsilon = 0.15$ として実験を行う．

実験

10^9 回の対戦により，行動価値関数の更新を行う．

ϵ-グリーディー方策を使うことによって，全ての状態行動対に対して行動価値関数の更新を行なうことができる．対戦回数が非常に多いのも，出現回数が非常に少ない局面と手に対しても，行動価値を求めることができるようにする目的がある．

結果

表 4.6: ϵ-グリーディー方策を用いた際の MC,TD(0) における 2 つの初期局面での先手番の手に対する $Q(s_t, a)$

	MC	TD(0)
局面 1+手 1	0.35232	0.45047
局面 1+手 2	0.40377	0.36868
局面 3+手 1	0.64785	0.52311
局面 3+手 2	0.14268	0.35744

表 4.7: ϵ-グリーディー方策を用いた際の MC,TD(0) における 2 つの初期局面での先手番の手に対する $Q(s_0, s_1, s_2, \cdots, s_t, a)$

	MC	TD(0)
局面 1+手 1	0.54157	0.46019
局面 1+手 2	0.35675	0.40768
局面 3+手 1	0.56487	0.53142
局面 3+手 2	0.10454	0.45404

それぞれの行動価値関数に基づき，最もプレイヤに取って行動価値が高くなるように手を指しゲームを進行させた時，ゲームが引き分けになることを確認した．両プレイヤは，

相手の駒を取ることは無く，互いの駒を出口に向かわせないように進路妨害をしながら，ゲームを引き分けに終えている．

$Q(s_t, a)$ での MC と $Q(s_0, s_1, s_2, \cdots, s_t, a)$ での TD では両プレイヤが赤駒を選択している．一方，$Q(s_t, a)$ での TD と $Q(s_0, s_1, s_2, \cdots, s_t, a)$ での MC では先手番プレイヤが青駒を，後手番プレイヤが赤駒を選択している．初手番プレイヤが赤駒を選択しているのは，常に進路を妨害される上，相手の駒が赤駒であるため取ることが出来ず，赤駒を選択しても結果が引き分けとなるためだと考えられる．最善戦略とは異なる駒の色となったが，引き分けになるという結果は導くことが出来た．

4.6 MG3でのϵ-グリーディー方策による価値関数の学習

実験

手番をパスすることが可能であるMinimum3においてランダム方策を用いた際のMCとTD(0)における状態価値関数$V(s_t)$を求める．さらに，Minimum3において，グリーディー方策およびϵ-グリーディー方策を用いた際の行動価値関数$Q(s_t, a)$，$Q(s_0, s_1, s_2, \cdots, s_t, a)$を求める．

結果

表4.8はMG3においてランダム方策を用いた際のMCおよびTD(0)における4つの初期局面での先手番プレイヤに対する$V(s_t)$を表す．

表 4.8: ランダム方策を用いた際のMC,TD(0)における4つの初期局面での先手番プレイヤに対する$V(s_t)$

	MC	TD(0)
局面1	0.3787	0.4182
局面2	0.6971	0.5808
局面3	0.5759	0.6495
局面4	0.3493	0.3657

表4.9はMG3においてϵ-グリーディー方策を用いた際のMCおよびTD(0)における2つの初期局面での先手番プレイヤの手に対する$Q(s_t, a)$を表す．

表 4.9: ϵ-グリーディー方策を用いた際のMC,TD(0)における2つの初期局面での先手番の手に対する$Q(s_t, a)$

	MC	TD(0)
局面1+手1	0.3959	0.5129
局面1+手2	0.3698	0.4301
局面3+手1	0.5466	0.5211
局面3+手2	0.4921	0.4499

表??はMG3においてϵ-グリーディー方策を用いた際のMCおよびTD(0)における2つの初期局面での先手番プレイヤの手に対する$Q(s_0, s_1, s_2, \cdots, s_t, a)$を表す．

ϵ-グリーディー方策を用いて各学習を行った$Q(s_t, a)$，$Q(s_0, s_1, s_2, \cdots, s_t, a)$を使い，手番プレイヤにとって最も行動価値が高くなるように手を指してゲームを進行させたところ，相手の駒を取らず，相手の駒の進路を防ぐ形で全て対局で引き分けとなった．

表 4.10: ϵ-グリーディー方策を用いた際の MC,TD(0) における 2 つの初期局面での先手番の手に対する $Q(s_0, s_1, s_2, \cdots, s_t, a)$

	MC	TD(0)
局面 1+手 1	0.4423	0.5208
局面 1+手 2	0.4806	0.4573
局面 3+手 1	0.4922	0.5597
局面 3+手 2	0.5000	0.4695

この実験からは MG3 における最善戦略があるのかはわからなかった．ゲーム自体に大きく偏りがあり，どちらかに非常に有利となる場合には，必勝手順などが求まるが，そうでない場合においては多くは引き分けに持ち込む戦術に落ち着いている．

第5章　ガイスターにおける学習

本章では，通常のガイスターにおける自己対戦によるニューラルネットワークを利用したSarsa(λ) による学習を行う．(1) 全く推測を用いない学習，(2) 相手の駒に対する推測を入力に加えることで推測状況に応じた行動価値関数を求めることを目標とした学習，(3) 相手の駒の推測と同様に自分の駒に対する推測も用い，相手から推測をされていることを考慮した状況に応じた行動価値関数を求めることを目標とした学習の 3 種類の学習を行う．その上で，学習を円滑に行うため，様々なルールのガイスターにおいて自己対戦による学習を行う．様々なルールとして，通常のガイスターに加え，判定勝ち等を加えた特殊なルールや着手制限を加えたルールなどのルールを考案する．さらに入力として与える特徴の種類を増やし，様々な入力での学習を行う．なお，駒の初期配置は完全ランダムで行なうこととした．

Sarsa(λ) 学習におけるパラメータは本章を通じて $\lambda = 0.7, \gamma = 0.95$ とする．方策は ϵ-グリーディー方策を用い，$\epsilon = 0.2$ とする．ニューラルネットワークの重みの初期化には非常に小さな正の実数である乱数値を用いる．乱数により初期化されたニューラルネットワークの重みが大きいと，初期のニューラルネットワークの出力が 1 に限りなく近づき，式 (2.32) におけるシグモイド関数の導関数値 σ が 0 となり，適格度トレースが更新されないため，ニューラルネットワークの重みも更新されない．実際には入力層のユニットの数に応じて変えており，表 5.1 のようにした．

表 5.1: ニューラルネットワークにおける重みの初期値

入力層のユニット数	重みの初期値
100 代	0 ～ 0.20 までの乱数
200 代	0 ～ 0.10 までの乱数
300 代	0 ～ 0.05 までの乱数

5.1 通常のガイスターにおける学習

自己対戦によるニューラルネットワークを利用した Sarsa(λ) 学習での行動価値関数の学習を行う．ニューラルネットワークの入力として，PBL による推測を全く用いないもの，相手の駒の推測値を用いるもの，相手の駒の推測値に加えて自分の駒の推測値を用いたものの 3 種類を用意する．以後，推測を全く用いないものを **NO-EST**，相手の駒の推測値を用いるものを **OP-EST**，両プレイヤの推測値を用いるものを **BOTH-EST** とする．そして，これらをまとめて **NORMAL** と呼ぶ．

NO-EST におけるニューラルネットワークの入力は以下とする．

- 自分の青駒の配置 (36 ユニット)
  自分の青駒がいるマスを表す．36 ユニットのそれぞれが盤上のマスに対応しており，駒がある場合 1，ない場合は 0 とする．

- 自分の赤駒の配置 (36 ユニット)
  自分の青駒の配置と同様に，自分の赤駒がいるマスを表す．

- 相手の駒の配置 (36 ユニット)
  上と同様に，相手の駒がいるマスを表す．

- 自分が取った相手の青駒の数 (3 ユニット)
  自分が取った相手の青駒の数を 2 ビットで表現した際の各ビットの値を各ユニットに割り当てる．

- 自分が取った相手の赤駒の数 (3 ユニット)
  上と同様に，自分が取った相手の赤駒の数を各ユニットに割り当てる．

- 合法手 (37 ユニット)
  移動させる駒のマスと移動先のマスを駒の配置と同様の方法で表す．なお，駒を脱出させる手を表現するために脱出時に 1 とする 37 番目のユニットを用意する．よって，駒を脱出する手が指される際には移動元のマスに対応するユニットと 37 番目のユニットが 1 となり，それ以外のユニットは 0 となる．

相手の駒に対する推測値を用いる **OP-EST** におけるニューラルネットワークの入力には NO-EST の入力に以下のユニットを追加する．

- 相手の駒に対する推測値 (5×8 ユニット)
  相手の駒に対する PBL によって求めた推測値を表す．具体的には，1 つの駒の推測値が $-1 \sim 1$ の値を取るため，$-1 \sim 1$ の領域を等間隔な 32 の領域に分割し，小さい方から $0 \sim 31$ の値を割り振る．この割り振った値を抽象化した推測値として 2 ビットで表現し，各ビットの値を 5 つのユニットに与える．これを図 5.1 に従った駒の位置の順番に 8 駒全てで行う．駒を取ってしまったために駒が盤上に存在しない場合はその駒に対応するユニットの値を全て 0 としておく．

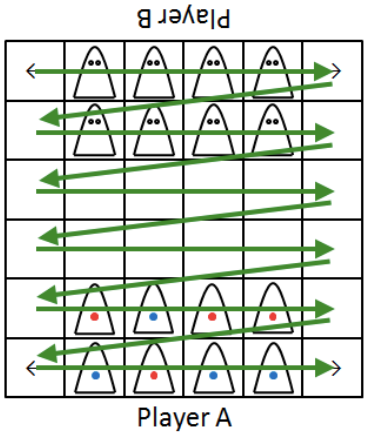


図 5.1: 推測値を入れる駒の順番

BOTH-EST におけるニューラルネットワークの入力には OP-EST の入力に以下のユニットを追加する．

- 自分の駒に対する推測値 (40 ユニット)
  相手の駒に対する推測値と同様に，自分の駒に対する PBL によって求めた推測値を表す．

NO-EST，OP-EST，BOTH-EST のそれぞれにおけるバイアス項を含めた入力層と中間層のユニットの数は表 5.2 のようになった．

表 5.2: NO-EST，OP-EST，BOTH-EST における入力層と中間層のユニット数

	入力層のユニット数	中間層のユニット数
NO-EST	152	77
OP-EST	192	97
BOTH-EST	232	117

実験

NO-EST，OP-EST，BOTH-EST のそれぞれで 100,000 回の自己対戦を行い，Sarsa(λ) による行動価値関数の学習を行う．さらに学習によって得られた行動価値関数を用いた AI プレイヤとランダムプレイヤとの対局実験も行う．

ランダムプレイヤとの対戦結果

学習によって得られた行動価値関数を用いて，最も行動価値関数の高い手を選択する AI プレイヤと合法手の中からランダムに手を選択するランダムプレイヤとの対局実験を

行う．先手番を交換し，1500 回ずつ合計 3000 回の対戦を行う．100,000 回の自己対戦により学習した NORMAL とランダムプレイヤにおける対戦結果は表 5.3 となる．

表 5.3: NORMAL とランダムプレイヤとの 3000 戦における結果

	NORMAL 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
NO-EST	10	61	1429	61	9	1430
OP-EST	11	53	1436	66	11	1423
BOTH-EST	35	53	1412	65	20	1415

この対戦において，全ての AI プレイヤは初期配置から駒をほとんど前進させず，手前で駒を動かし，同形反復を繰り返す手を指す．ランダムに手を指すため AI プレイヤと同様に勝利条件を達成し難いランダムプレイヤと対戦すると殆どの対局において引き分けとなってしまう．AI プレイヤが初期配置から同形反復を繰り返すようになるのはガイスターというゲームが適当に手を指すと引き分けになりやすいという性質を持つためである．つまり，ゲームの初期において，適当に手を指し，ほとんどが引き分けになるため，引き分けの手のみを評価し，勝敗をつける手が評価されなくなることによる．

よって，次節ではガイスターのルールの変更を行い，より勝敗がつきやすく，駒を取る手や駒を前進させる手がより評価され易いルールに変更する．

5.2 勝敗をつきやすくするための勝利条件の変更

通常のルールを，ゲームに偏りが生じ積極的にプレイせざるを得ない状況を生み出すルールや比較的ゲームに勝敗がつきやすいような判定勝ちのルールに変えることで攻撃的な手の頻度を上げゲームに勝敗をつきやすくする工夫を導入する．なお，工夫を入れていない通常のルールを以降 **Rule:Normal** と呼ぶ．以下が変更した勝利条件となっている．

- **Rule:Turn200**
  両プレイヤが 100 手，合計 200 手を指すまでゲームが行われるルール．両プレイヤが合計 200 手を指した時点で，勝敗がついていない場合，引き分けとなる．

- **Rule:SecondWinAfterTurnLimit**
  両プレイヤが 50 手を指した時点で勝敗がついていない場合，後手番プレイヤが勝ちとなるルール．

- **Rule:Decision1**
  両プレイヤが 50 手を指した時点で勝敗がついていない場合，取った青駒の数が多いプレイヤが勝ちとなる．それでも勝敗がつかない場合，両プレイヤの青駒と出口までの最短マンハッタン距離を求め，距離が短いプレイヤを勝ちとし，さらに，勝敗がつかない場合は引き分けとする．

- **Rule:Decision2**
  両プレイヤが50手を指した時点で勝敗がついていない場合，両プレイヤの青駒と出口までの最短マンハッタン距離を求め，距離が短いプレイヤが勝ちとする．それでも勝敗がつかない場合，取った青駒の数が多いプレイヤを勝ちとし，さらに，勝敗がつかない場合は引き分けとする．

- **Rule:Decision3**
  両プレイヤが50手を指した時点で勝敗がついていない場合，両プレイヤのスコアを計算し，スコアが高いプレイヤを勝ちとする．スコアが同じ場合には引き分けとする．スコアは次の式で求める．

$$\text{score} = \text{取った青駒の数} - \text{最短である青駒から出口までのマンハッタン距離}$$

Rule:Turn200はターン数が増えることにより，ゲームがより進行することを狙い導入する．Rule:SecondWinAfterTurnLimitはゲームに後手番プレイヤが大幅に有利であるという大きな偏りを生じさせることにより，先手番プレイヤがより積極的に動かなければ負けてしまう状況を作り出す．Rule:Decision1,2,3は青駒を出口に進ませる手および青駒を取る手を評価することにより，駒の前進や駒取りを行う手がよりよく評価されることを目標とする．

実験

それぞれの勝利条件に変更を加えたガイスターにおいて100,000回の自己対戦により学習し，得られた行動価値関数を用いるAIプレイヤとランダムプレイヤとの3000試合の対局実験を行う．

結果

表5.4，5.5，5.6，5.7，5.8は各ルールでの学習を行ったAIプレイヤとランダムプレイヤとの対戦結果となっている．前回の実験から全くAIプレイヤは改善されず，手前で駒を動かし続け，多くの試合で引き分けになる．これは，別の特殊なルールにおいて勝敗を決する手が元のゲームでのルールにおいては引き分けになる手となっており．特殊なルールにおいて評価されてる手が通常のルールでは決着が付く手となっていないことによる．よって，次節ではさらに影響の大きい着手における制約を課し，よりゲームを進行させる工夫とする．

表 5.4: Rule:Turn200 で学習を行った NORMAL とランダムプレイヤとの 3000 戦における結果

	NORMAL 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
NO-EST	26	61	1413	57	22	1421
OP-EST	23	60	1417	48	26	1426
BOTH-EST	32	73	1395	79	27	1394

表 5.5: Rule:TurnLimitSecondWin で学習を行った NORMAL とランダムプレイヤとの 3000 戦における結果

	NORMAL 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
NO-EST	28	48	1424	57	22	1421
OP-EST	22	81	1397	66	23	1411
BOTH-EST	9	57	1434	59	15	1426

表 5.6: Rule:Decision1 で学習を行った NORMAL とランダムプレイヤとの 3000 戦における結果

	NORMAL 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
NO-EST	24	58	1418	53	36	1411
OP-EST	17	54	1429	54	12	1434
BOTH-EST	25	66	1409	54	23	1423

表 5.7: Rule:Decision2 で学習を行った NORMAL とランダムプレイヤとの 3000 戦における結果

	NORMAL 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
NO-EST	34	81	1385	66	30	1404
OP-EST	12	54	1434	54	29	1417
BOTH-EST	6	69	1425	39	9	1452

表 5.8: Rule:Decision3 で学習を行った NORMAL とランダムプレイヤとの 3000 戦における結果

	NORMAL 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
NO-EST	8	60	1432	59	14	1427
OP-EST	14	73	1413	69	17	1414
BOTH-EST	20	64	1416	74	25	1401

5.3　合法手の制限

自己対戦におけるゲームを強制的に進行させ，人間の対局に近づけるため，合法手の制限を行う．さらに、青駒が相手側の出口のマスに乗っており，脱出できる状況なら脱出する手を必ず指すようにする．着手を全く制限しないものを以降 **Hand-All** と呼ぶ．

以下が合法手の制限となっている．

- **Hand-LimitSamePosition1**
  過去の手番における自分の青駒と赤駒の配置を再現する手を指すことを禁止するという制限を導入する．

- **Hand-LimitSamePosition2**
  過去の手番における自分の駒の配置を再現する手を指すことを禁止するという制限を導入する．Hand-LimitSamePosition1 において，駒の配置は同じでも駒の色が違うといった配置になる手は認められるが，Hand-LimitSamePosition2 の場合には，駒の色に関係なく配置が同じになる手ならば，その手を指すことは出来ない．よって，Hand-LimitSamePosition1 よりも厳しい制限となる．

- **Hand-SuperLimit**
  Hand-LimitSamePosition2 の合法手制限に加えて，合法手から駒を後退させる手を除く．

- **Hand-SuperLimit2**
  Hand-LimitSamePosition2 の合法手制限に加えて，手前 4 行のマスにある駒を後退する手を合法手から除く．つまり，図 5.2 において赤い点線の枠に囲まれたマスにある駒は駒は前進と左右への移動しか出来ない．これは Hand-SuperLimit の制限を緩和させたものとなる．

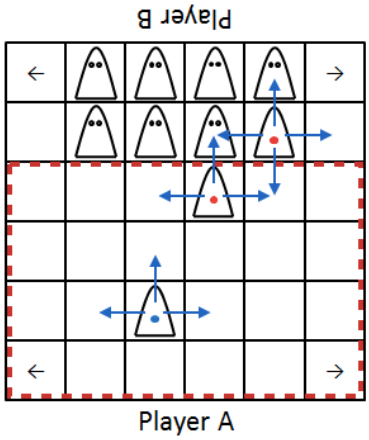


図 5.2: Hand-SuperLimit2 における PlayerA の合法手の例

ただし，これらの制限により，合法手が無くなった場合には，過去の手番における駒配置を考慮しない手の中からランダムに手を指すこととする．つまり，制限により合法手が無くなった場合，Hand-LimitSamePosition1,2 では全ての合法手から，Hand-SuperLimit1 では全ての合法手より後退する手を除いたものから，Hand-SuperLimit2 では全ての合法手より手前 4 行のマスにある駒の後退する手を除いたものからランダムに手を選ぶ．

実験

Rule:Normal に各着手制限を加えたルールで入力を NORMAL とし，100,000 回の自己対戦による学習を行う．各着手制限を加えたルールで学習させた AI プレイヤとランダムプレイヤとの対戦を 3000 回行う．さらに学習時と同様の着手制限を加えた AI プレイヤとランダムプレイヤとの対戦を 3000 回行う．

結果

各着手制限を加えたルールで学習させた AI プレイヤとランダムプレイヤとの対戦を 3000 回行う．

表 5.9～ 5.12 は各着手制限下で学習を行った AI プレイヤとランダムプレイヤとの対局結果である．

表 5.9: Hand-LimitSamePosition1 学習 NORMAL とランダムプレイヤとの 3000 戦における結果

	NORMAL 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
NO-EST	14	69	1417	68	29	1403
OP-EST	31	73	1396	56	30	1414
BOTH-EST	34	56	1410	44	12	1444

表 5.10: Hand-LimitSamePosition2 学習 NORMAL とランダムプレイヤとの 3000 戦における結果

	NORMAL 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
NO-EST	29	71	1400	67	13	1420
OP-EST	31	62	1407	84	21	1395
BOTH-EST	38	94	1368	108	43	1349

表 5.11: Hand-SuperLimit1 学習 NORMAL とランダムプレイヤとの 3000 戦における結果

	NORMAL 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
NO-EST	27	59	1414	53	16	1431
OP-EST	7	50	1443	33	8	1459
BOTH-EST	16	58	1426	76	22	1402

表 5.12: Hand-SuperLimit2 学習 NORMAL とランダムプレイヤとの 3000 戦における結果

	NORMAL 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
NO-EST	16	51	1433	69	16	1415
OP-EST	36	76	1388	71	29	1400
BOTH-EST	12	48	1442	50	21	1429

その結果，同様に手前で駒を動かし，多くの試合で引き分けとなる．これは，着手制限によって指すことが出来なかった手には行動価値関数の更新が行われないことによる．つまり，一度も行動価値関数が更新されていない手に対して，AI プレイヤが行動価値を求めることになる．これにより，行動価値関数がどんな値を返すかは未定義となる．今回の行動価値関数はゲーム開始時の手が最も高く，初期配置を保持する手がよいという結果を出力していることを確認した．よって，AI プレイヤは初期配置を保持する手を指し，ランダムプレイヤとの対局において引き分けが非常に多くなる．

そこで，AI プレイヤに学習した際の着手制限と同様の着手制限を導入し，ランダムプレイヤと対戦させた．

表 5.13∼ 5.16 は学習時と同じ着手制限を課した AI プレイヤとランダムプレイヤとの 3000 戦における結果である．

表 5.13: Hand-LimitSamePosition1 学習着手制限付き NORMAL とランダムプレイヤとの 3000 戦における結果

	NORMAL 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
NO-EST	27	63	1410	70	37	1393
OP-EST	25	53	1422	42	21	1437
BOTH-EST	103	75	1322	72	94	1334

表 5.14: Hand-LimitSamePosition2 学習着手制限付き NORMAL とランダムプレイヤとの 3000 戦における結果

	NORMAL 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
NO-EST	95	131	1274	149	108	1243
OP-EST	24	78	1398	87	39	1374
BOTH-EST	40	76	1384	84	42	1374

表 5.15: Hand-SuperLimit1 学習着手制限付き NORMAL とランダムプレイヤとの 3000 戦における結果

	NORMAL 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
NO-EST	54	86	1360	84	48	1368
OP-EST	34	68	1398	62	23	1415
BOTH-EST	26	43	1431	50	27	1423

表 5.16: Hand-SuperLimit2 学習着手制限付き NORMAL とランダムプレイヤとの 3000 戦における結果

	NORMAL 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
NO-EST	37	87	1376	100	43	1357
OP-EST	174	206	1120	198	195	1107
BOTH-EST	302	204	994	94	118	1288

Hand-LimitSamePosition1,2 および Hand-SuperLimit1 では，引き分け数はあまり減っていない．しかし，Hand-SuperLimit2 での OP-EST と BOTH-EST において，引き分け数はより少なくなっている．特に BOTH-EST ではランダムプレイヤに対して勝ち越している．

Hand-LimitSamePosition1,2 は着手制限が弱く，より引き分けになりやすいため，自己対戦において多くの試合で勝敗がついていないと考えられる．実際に，自己対戦の初期を確認した所，相手の駒を取る手や出口に進む手はあまり指されることはなく，勝敗がつかない対局が多く含まれた．Hand-SuperLimit1 では赤駒が出口のマス上にあり，その隣に青駒が並んでいる場合，赤駒を交代させ，青駒を最短で脱出させる手が使うことが出来ない．よって，Hand-SuperLimit1 ではこのような状況下での行動価値関数を誤って

学習する．一方，Hand-SuperLimit2 では合法手の制約が弱くなっているため，このような手を指すことができる．これらのことから Hand-SuperLimit1 より Hand-SuperLimit2 は行動価値関数をより正確に学習できる．よって，これ以降の実験では着手制限として Hand-SuperLimit2 を用いる．

Hand-SuperLimit2 において，学習の初期における自己対戦の内容を確認したところ，AI プレイヤが脱出手や相手の青駒を取る手により勝敗がつく試合が多いことを確認した．これは脱出手や駒取りを行う手を学習していることによるものだと考えられる．自己対戦の初期において，より積極的な脱出手や駒取りを行う手を指せるようになっていたものの，100,000 回学習を行った AI プレイヤはランダムプレイヤ相手に大きく勝ち越すことは出来ていない．

次節では，実際の自己対戦がどのように進展しているのかを確認するため自己対戦 500 戦ごとの対戦結果を確認する．

5.4　500回ごとの勝敗結果の出力

Hand-SuperLimit2における自己対戦による学習の初期において，青駒を脱出させる手や相手の駒を取る手を指すことが可能であった．しかし，100,000回の自己対戦により学習した行動価値関数を利用したプレイヤは非常に弱い．自己対戦による学習の過程を観察するため，500回ごとに500戦における勝敗と引き分けの数をカウントする．

実験を行うにあたり，ニューラルネットワークの入力を改める．ルールは最も脱出手や駒取りを行う手を学習していたRule:SuperLimit2とし，NORMALに変わる新しい入力**NEW**，NEWに更なる特徴を追加した**OPTION**，手に関する特徴への重みを大きくした**EXTRA**の3つの入力群を定義し，これらの入力を用いて学習を行う．

改良後の推測値を含まない入力を**NEW-NO-EST**，相手の駒の推測値を用いた入力を**NEW-OP-EST**，両プレイヤの推測値を用いた入力を**NEW-BOTH-EST**とする．これらをまとめて**NEW**と呼ぶ．

以下が**NEW-NO-EST**の入力である．

- 自分の青駒の配置 (36ユニット)
  自分の青駒がいるマスを表す．
- 自分の赤駒の配置 (36ユニット)
  自分の赤駒がいるマスを表す．
- 相手の駒の配置 (36ユニット)
  相手の駒がいるマスを表す．
- 自分が取った相手の青駒の数 (3ユニット)
  自分が取った相手の青駒の数を表5.17のように表現する．なお，取った駒が4つの時にはゲームがすでに終了しており，ニューラルネットワークの入力を生成する必要がないため，この場合は考えない．
- 自分が取った相手の赤駒の数 (3ユニット)
  上と同様に，自分が取った相手の赤駒の数を表現する．
- 合法手 (37ユニット)
  移動させる駒のマスと移動先のマスを駒の配置と同様の方法で表す．

表 5.17: 取った駒の数とそれに対応するビット列

取った駒の数	ビット列
0	000
1	001
2	010
3	100

相手の駒に対する推測値を用いる **NEW-OP-EST** におけるニューラルネットワークの入力には NEW-NO-EST の入力に以下のユニットを追加する．

- 相手の駒に対する推測値 (12×8 ユニット)
  相手の駒に対する PBL によって求めた推測値を表す．1 つの駒に対する推測値を 12bit で表現する．具体的には，推測値が -0.19 以下である場合 100000000000 の各ビットをユニットに与える．推測値が 0.19 以上である場合には 000000000001 の各ビットを 12 個のユニットに与える．推測値 s が $-0.19 < s < 0.19$ である場合である場合，$-0.19 \sim 0.19$ の領域を等間隔な 10 の領域に分割し，小さい方から 01000000000, 00100000000，000100000000, $\cdots$，00000000010 とし，各ビットの値をユニットに与える．

NEW-BOTH-EST におけるニューラルネットワークの入力には NEW-NO-EST の入力に以下のユニットを追加した．

- 相手の駒に対する推測値 (10×8 ユニット)
  NEW-OP-EST における，1 つの駒に対する推測値を 12bit から 10bit にし，$-0.19 \sim 0.19$ の領域の分割を 8 としたもの．この場合，NEW-OP-EST よりも推測値が抽象的な表現になる．

- 自分の駒に対する推測値 (10×8 ユニット)
  上と同様に，自分の駒に対する PBL によって求めた推測値を表す．

NEW-NO-EST，NEW-OP-EST，NEW-BOTH-EST のそれぞれにおけるバイアス項を含めた入力層と中間層のユニットの数は表 5.18 のようになる．

表 5.18: NEW-NO-EST，NEW-OP-EST，NEW-BOTH-EST における入力層と中間層のユニット数

	入力層のユニット数	中間層のユニット数
NEW-NO-EST	152	77
NEW-OP-EST	248	125
NEW-BOTH-EST	312	157

さらに NEW-NO-EST，NEW-OP-EST，NEW-BOTH-EST に新たな特徴ベクトルを加えた **OPTION-NO-EST**，**OPTION-OP-EST**，**OPTION-BOTH-EST** を構成する．これらをまとめて OPTION と呼ぶ．

それぞれに追加した特徴は以下のようになる．

- 自分の青駒の出口のマスまでの最短距離 (7 ユニット)
  自分の青駒から出口のマスまでの最短のマンハッタン距離を 7bit で表 5.19 のように表現する．各ビットの値をユニットに入れる．また，駒から出口のマスまでのマンハッタン距離の最大値は 7 となり，その時の駒位置は最も手前の行の真ん中の 2 マスにある時となる．

- 相手の駒と隣り合う自分の駒の位置 (36 ユニット)
  相手の駒と前後左右で隣接する自分の駒の位置を自分の青駒の配置などと同様の方法で表現する．

- 自分の駒と隣り合う相手の駒の位置 (36 ユニット)
  上と同様の方法で相手の駒と上下左右で隣接する自分の駒の位置を表現する．

表 5.19: マンハッタン距離とそれに対応するビット列

マンハッタン距離	ビット列
0	0000001
1	0000010
2	0000100
3	0001000
4	0010000
5	0100000
6	1000000
7	0000000

自分の青駒の出口のマスまでの最短距離は出口からの脱出を目指す上で非常に重要な特徴であるため追加する．相手の駒と隣り合う自分の駒の位置と自分の駒と隣り合う相手の駒の位置は相手が手を指すことによって取られる駒の位置と自分が手を指すことによって取ることができる駒の位置であり，相手の駒取りから逃れる，もしくは，相手に駒を取らせる，そして，相手の駒を取る上で非常に重要な情報となるため追加する．なお，Rule:SuperLimit2 では，駒が後退出来ないことがあるため，隣接しているからといって必ず駒を取ることができるまたは駒が取られる可能性があるとは限らない．

OPTION-NO-EST，OPTION-OP-EST，OPTION-BOTH-EST のそれぞれにおけるバイアス項を含めた入力層と中間層のユニットの数は表 5.20 となる．

表 5.20: OPTION-NO-EST，OPTION-OP-EST，OPTION-BOTH-EST における入力層と中間層のユニット数

	入力層のユニット数	中間層のユニット数
OPTION-NO-EST	231	117
OPTION-OP-EST	327	165
OPTION-BOTH-EST	390	196

最後に，着手後の状態の表現に重点を置いた **EXTRA-NO-EST**，**EXTRA-OP-EST**，**EXTRA-BOTH-EST** を考えた．これらをまとめて EXTRA と呼ぶ．

EXTRA-NO-EST の入力は以下のようになっている．

- 着手後の自分の青駒の配置 (37 ユニット)
  自分の青駒がいるマスを表す．
- 着手後の自分の赤駒の配置 (36 ユニット)
  自分の赤駒がいるマスを表す．
- 着手前の相手の駒の配置 (36 ユニット)
  相手の駒がいるマスを表す．
- 自分が取った相手の青駒の数 (3 ユニット)
  自分が取った相手の青駒の数を表現する．
- 自分が取った相手の赤駒の数 (3 ユニット)
  自分が取った相手の赤駒の数を表現する．
- 駒を取る着手か否か (1 ユニット)
  相手の駒を取る着手ならばユニットに 1，そうでなければ 0 を与える．
- 着手後の出口までの最短距離 (8 ユニット)
  脱出させた駒の距離を 0，出口のマスの上に乗っている駒の距離を 1 として数える．つまり，盤上にある駒の着手後の出口までの最短距離はマンハッタン距離に 1 加えたものとなる．この最短距離に対応するビット列は表 5.21 において示す．

- 着手後に相手の駒と隣り合う自分の駒の位置 (36 ユニット)
  自分が着手をした後に，相手の駒と上下左右で隣接する自分の駒の位置を自分の青駒の配置などと同様の方法で表現する．脱出した駒には隣接する駒はないものとして考える．

- 着手後の自分の駒と隣り合う相手の駒の位置 (36 ユニット)
  上と同様の方法で，自分が着手をした後に，自分の駒と上下左右で隣接する相手の駒の位置を表現する．

表 5.21: 着手後の出口までの距離とそれに対応するビット列

出口までの距離	ビット列
0	00000001
1	00000010
2	00000100
3	00001000
4	00010000
5	00100000
6	01000000
7	10000000
8	00000000

この EXTRA-NO-EST に NEW-OP-EST や OPTION-OP-EST と同様の相手の駒に対する推測値 12×8 ユニットを加えたものを **EXTRA-OP-EST** とする．

さらに，EXTRA-NO-EST に NEW-BOTH-EST や OPTION-BOTH-EST と同様の相手の駒に対する推測値 10×8 ユニットと自分の駒に対する推測値 10×8 ユニットを加えた入力を **EXTRA-BOTH-EST** とする．

EXTRA-NO-EST および EXTRA-OP-EST，EXTRA-BOTH-EST におけるバイアス項を含めたニューラルネットワークの入力層と中間層のユニット数は表 5.22 となる．

表 5.22: EXTRA-NO-EST，EXTRA-OP-EST，EXTRA-BOTH-EST における入力層と中間層のユニット数

	入力層のユニット数	中間層のユニット数
EXTRA-NO-EST	197	100
EXTRA-OP-EST	293	148
EXTRA-BOTH-EST	357	180

実験

これらの入力を用いた AI プレイヤで 100,000 回の自己対戦をさせ，自己対戦 500 回ごとに直前 500 戦中の先手勝ちの回数および後手勝ちの回数，引き分けの回数，とその時のニューラルネットワークの重みを出力させる．

結果 1

表 5.3~5.11 が NEW および OPTION，EXTRA を入力として自己対戦により学習させた際の 500 試合ごとの先手勝ちおよび後手勝ち，引き分けの回数のグラフである．

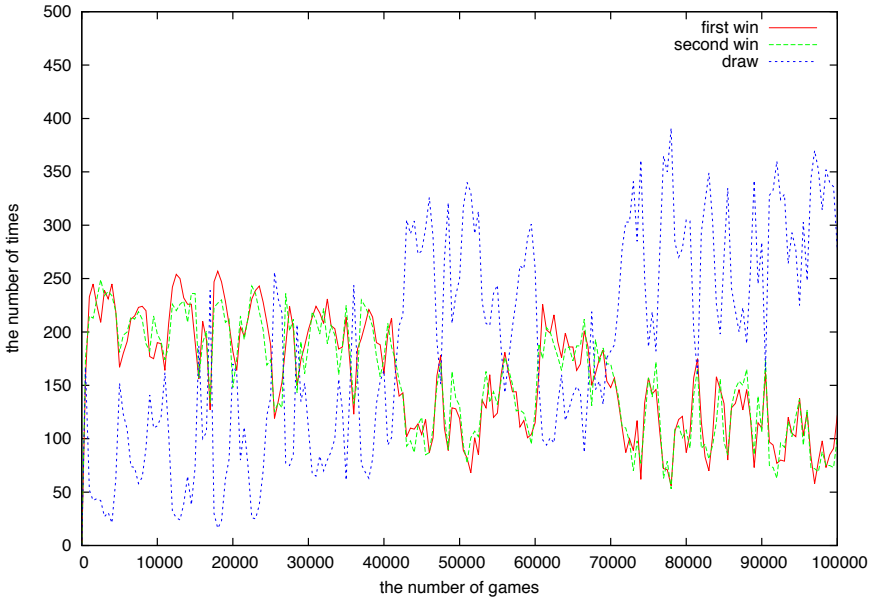


図 5.3: NEW-NO-EST における 500 戦ごとの各リザルトの回数

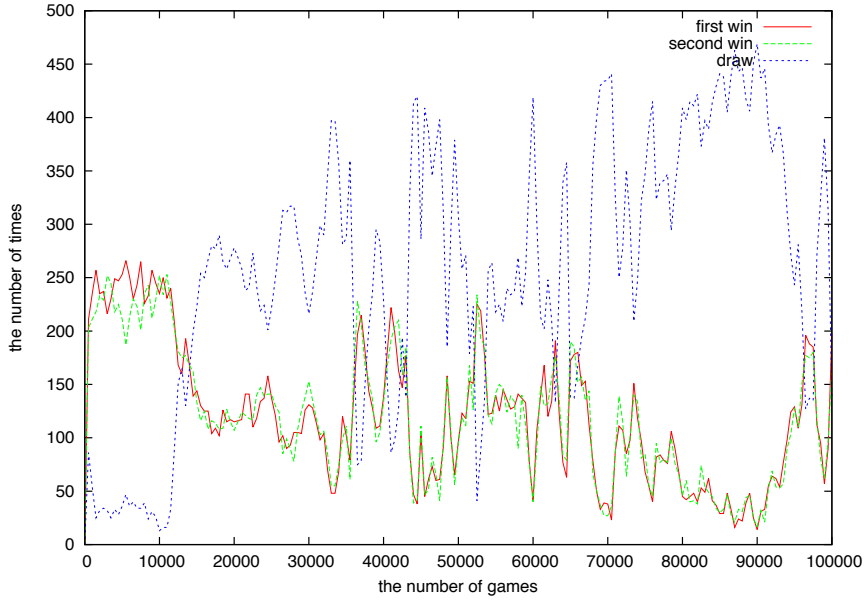


図 5.4: NEW-OP-EST における 500 戦ごとの各リザルトの回数

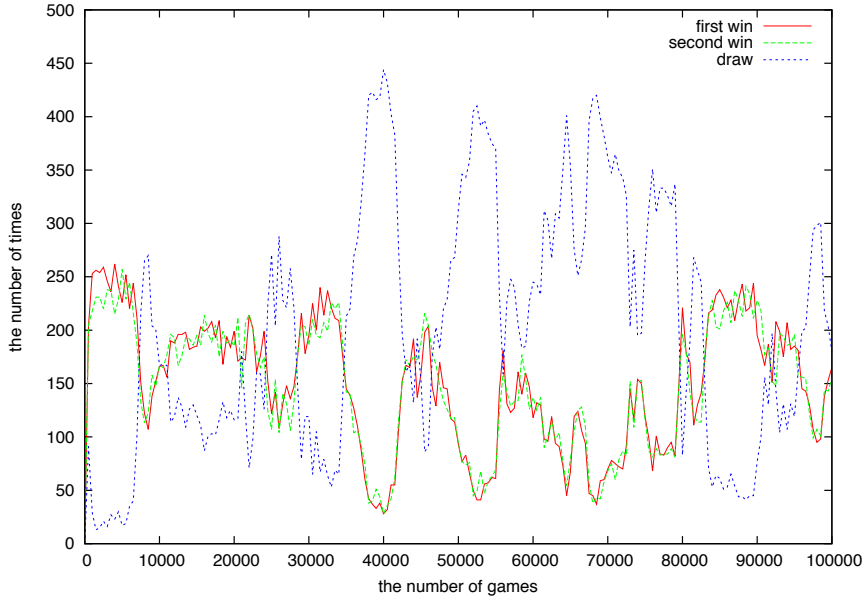


図 5.5: NEW-BOTH-EST における 500 戦ごとの各リザルトの回数

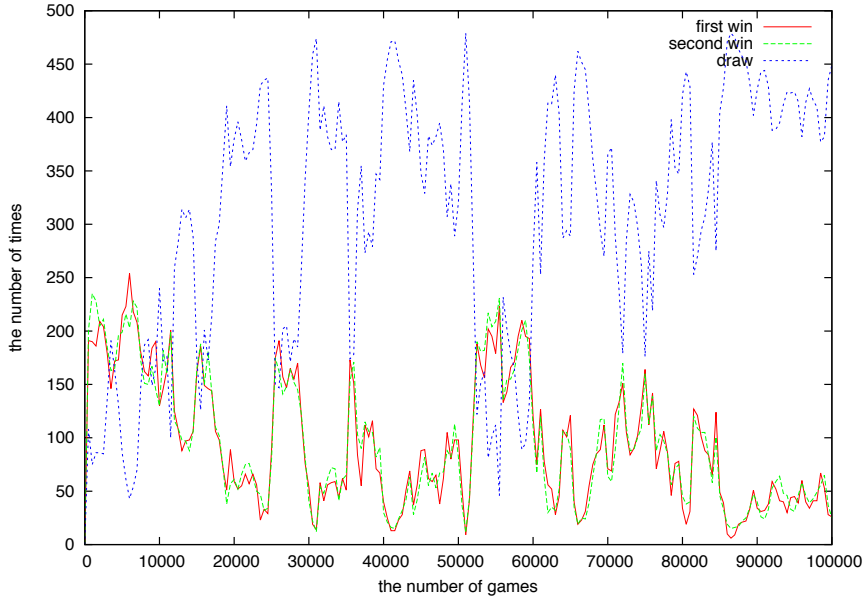


図 5.6: OPTION-NO-EST における 500 戦ごとの各リザルトの回数

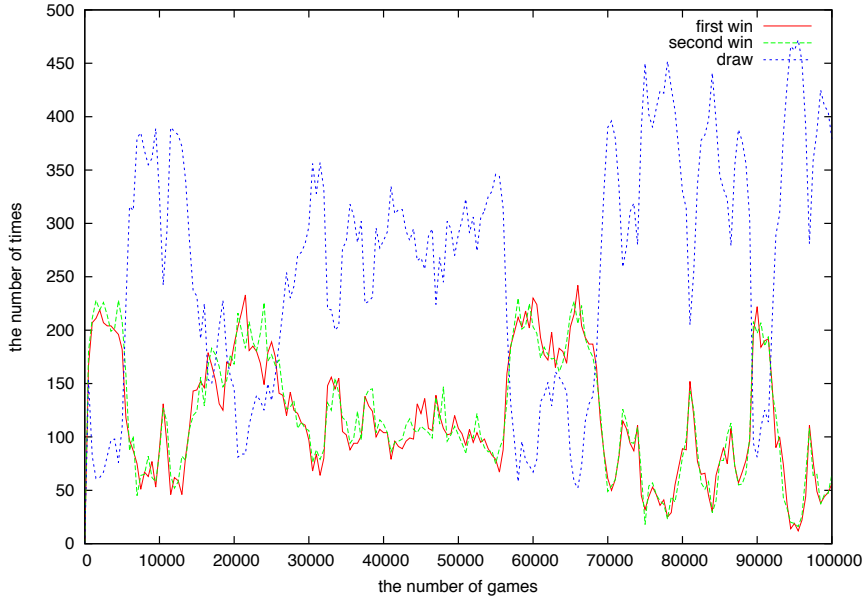


図 5.7: OPTION-OP-EST における 500 戦ごとの各リザルトの回数

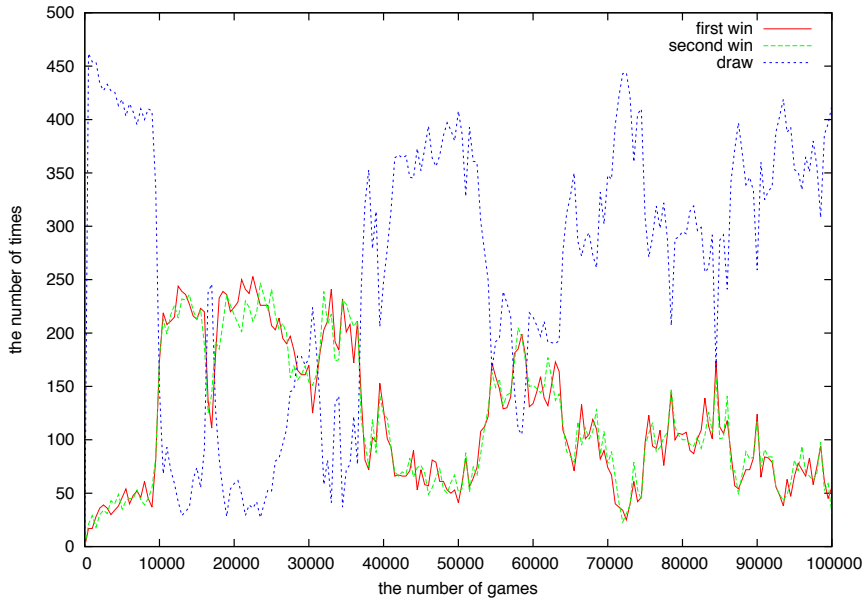


図 5.8: OPTION-BOTH-EST における 500 戦ごとの各リザルトの回数

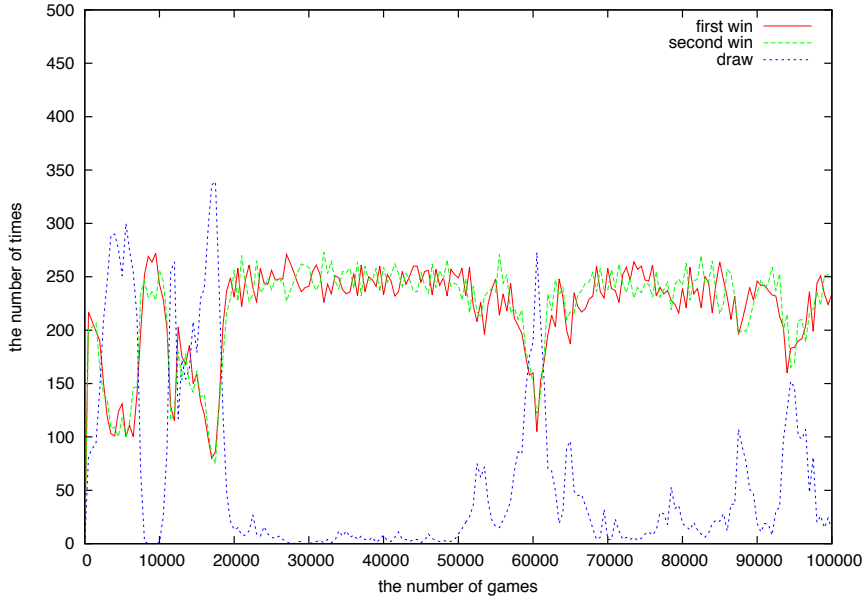


図 5.9: EXTRA-NO-EST における 500 戦ごとの各リザルトの回数

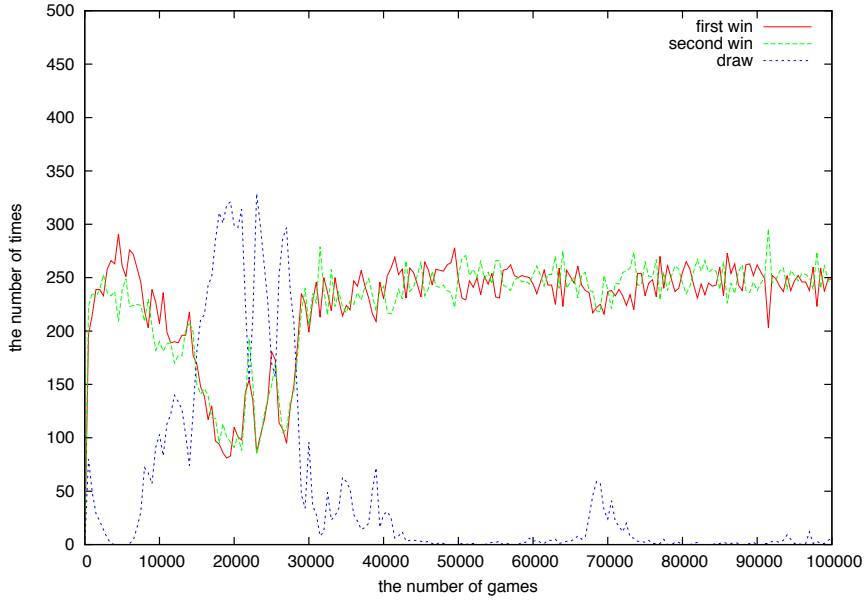


図 5.10: EXTRA-OP-EST における 500 戦ごとの各リザルトの回数

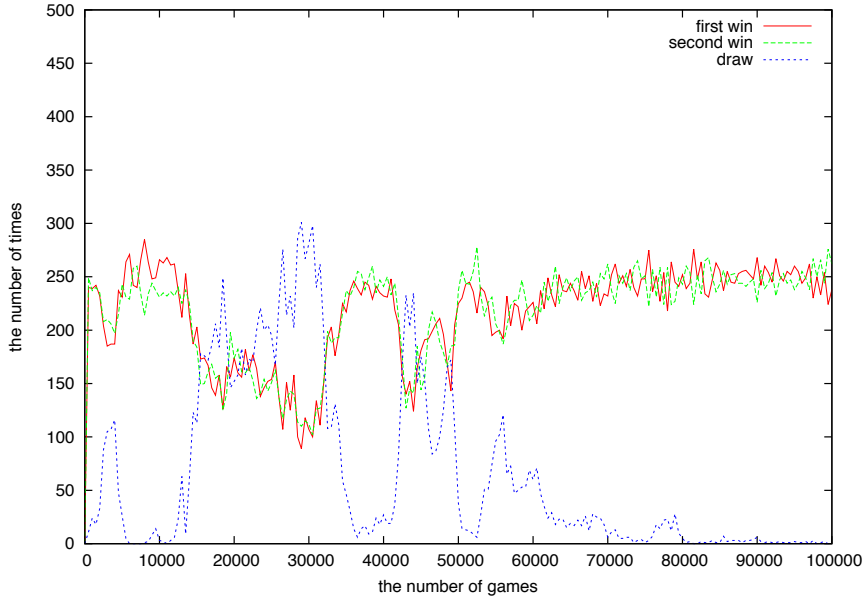


図 5.11: EXTRA-BOTH-EST における 500 戦ごとの各リザルトの回数

全ての入力での結果において先手勝ちと後手勝ちの回数は常に近い値を取り続けている．勝敗がついた対戦の回数と引き分けの回数は振動しており，NEW-NO-EST，NEW-OP-EST，OPTION において引き分け回数は次第に多くなる傾向がある．NEW-NO-EST では先手勝ち数と後手勝ち数が緩やかに少なくなり，引き分け数が緩やかに多くなる傾向

が見られる．NEW-OP-EST では自己対戦約 12,000 回から引き分け数が非常に大きくなり，その後，引き分け数が大きな値で推移している．NEW-BOTH-EST では，先手勝ち数および後手勝ち数，引き分け数が周期が大きい波となっているのが見られる．

OPTION の全てで学習の後半にかけて引き分け数が非常に高い値で推移している傾向が見られる．

EXTRA においては，NEW や OPTION と違い，先手勝ち，後手勝ち数は学習の前半において急激に低くなる箇所が見られるがその後上昇し 250 付近で落ち着いていることがわかる．一方，引き分け数は低い値で落ち着いている．EXTRA は手を指した後の自駒の配置を入力としており，各合法手における入力が大きく変わりやすい．よって，NEW や OPTION と大きく異る結果になったと考えられる．

先手勝ち数および後手勝ち数が多いということは，勝利条件を満たすための脱出手や駒取りを行なう手が多く行われるということである．つまり，自己対戦において積極的に勝利条件を満たす手をより評価していると考えられる．一方，引き分け数が多いということは，積極的に勝利条件を満たす手があまり評価されていないということだと考えられる．

先手勝ち数および後手勝ち数が振動しているのは，何らかの勝利条件を満たし勝つ手を学習したあと，それに対抗する着手を学習し，防御的になるためだと考えられる．防御的になり，引き分け数があまりに多くなってしまうと，引き分けにする手ばかりを学習してしまい，自己対戦であることより，両プレイヤがゲームを引き分けに持ち込むようになる．ガイスターでは両プレイヤがゲームを引き分けに持ち込むことが容易である．よって，勝敗がつく手を学習するまでの長い間，引き分け数が多い状態が続くと思われる．

結果２

NEW-NO-EST における自己対戦 500 回ごとの先手勝ち数，後手勝ち数，引き分け数に特徴があるものをサンプリングし，その時のニューラルネットワークの重みを利用して，ランダムプレイヤと対戦させた．

表 5.23 は NEW-NO-EST における自己対戦回数とその直前 500 戦での試合結果である．

表 5.23: NEW-NO-EST における学習対戦数と直前の 500 戦の結果

学習対戦数	先手勝ち数	後手勝ち数	引き分け数
12000	242	226	32
15500	156	157	187
17000	127	134	239
78000	56	53	391
90500	162	174	164

表 5.24 は NEW-NO-EST におけるそれぞれの自己対戦回数での行動価値関数を用いた AI プレイヤとランダムプレイヤとの対戦結果である．

表 5.24: 各自己対戦回数での入力を NEW-NO-EST とする行動価値関数を用いた AI プレイヤとランダムプレイヤとの 3000 戦における結果

	NEW-NO-EST 先手			ランダム先手		
学習対戦数	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
12000	1217	179	104	182	1217	101
15500	634	85	781	115	625	760
17000	673	148	679	123	704	673
78000	63	120	1317	116	92	1292
90500	147	211	1142	199	138	1163

対戦回数 12,000 回ではランダムプレイヤに対して大きく勝ち越している．対戦内容を確認した所，この AI プレイヤは脱出手を学習しており，相手の駒をあまり考慮せず青駒を出口に向かわせる手を指す．ランダムプレイヤでは出口に向かって動く相手の青駒を防ぐ手を指すことは非常に難しい．よって，勝利数は多くなる．直前の 500 戦の結果において引き分けが多い対戦回数 78000 回と 90500 回の場合はランダムプレイヤに対しても引き分けが多く弱い．この場合，AI は消極的な手を指し，引き分けにもつれることが多い．よって，直前の 500 戦における引き分けの回数が多い場合には AI プレイヤは弱く，多くの試合で引き分けると考えられる．

表 5.25 は OPTION-NO-EST における自己対戦回数とその直前 500 戦での試合結果である．

表 5.25: OPTION-NO-EST における学習対戦数と直前の 500 戦の結果

学習対戦数	先手勝ち数	後手勝ち数	引き分け数
13000	88	98	314
15000	162	165	173
19000	51	38	411
58500	210	201	89
66000	19	19	462

表 5.26 は OPTION-NO-EST におけるそれぞれの自己対戦回数での行動価値関数を用いた AI プレイヤとランダムプレイヤとの対戦結果である.

表 5.26: 各自己対戦回数での入力を OPTION-NO-EST とする行動価値関数を用いた AI プレイヤとランダムプレイヤとの 3000 戦における結果

	OPTION-NO-EST 先手			ランダム先手		
学習対戦数	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
13000	157	191	1152	196	158	1146
15000	209	206	1085	198	195	1107
19000	85	72	1343	93	91	1316
58500	252	262	986	242	240	1018
66000	36	52	1412	59	24	1417

引き分けの回数が少ない対戦回数 58500 回の OPTION-NO-EST を用いた AI プレイヤは少ない引き分け回数になるもののランダムプレイヤと近い勝敗数となっている.

表 5.27 は EXTRA-NO-EST における自己対戦回数とその直前 500 戦での試合結果である.

表 5.27: EXTRA-NO-EST における学習対戦数と直前の 500 戦の結果

学習対戦数	先手勝ち数	後手勝ち数	引き分け数
9000	264	236	0
12000	115	121	264
17000	80	84	336
22000	261	227	12
80000	239	242	19

表 5.28 は OPTION-NO-EST におけるそれぞれの自己対戦回数での行動価値関数を用いた AI プレイヤとランダムプレイヤとの対戦結果である．

表 5.28: EXTRA-NO-EST 自己対戦回数とランダムプレイヤとの 3000 戦における結果

	Normal 先手			ランダム先手		
学習対戦数	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
9000	1472	26	2	21	1479	0
12000	188	61	1251	57	190	1253
17000	106	98	1296	106	110	1284
22000	550	610	340	649	527	324
80000	247	284	969	279	280	941

EXTRA-NO-EST での自己対戦 9000 回を行った AI プレイヤは出口を狙う戦略を学習しており，非常に高い勝率となる．自己対戦 12000 回と 17000 回を行った AI プレイヤは学習での直前の 500 戦における引き分けの割合も多く，ランダムプレイヤの対局においても引き分けが非常に多くなる．自己対戦 22000 回を行った AI プレイヤとランダムの対局では引き分け数が少なくなるが，同じく直前 500 戦中の引き分け回数が少ない自己対戦 80000 回の AI プレイヤではランダムプレイヤとの対局による引き分け数が非常に多くなる．80000 回の自己対戦を経た AI プレイヤは相手に対して，にじり寄るような戦略をとる．慎重に攻める戦略はランダムプレイヤとの相性が悪く，対局に時間がかかるため，引き分け数が非常に増えたと思われる．

引き分けとなる試合の学習による影響を調べるため，次節では，引き分けを少なくする工夫を導入した学習を行なう．

5.5　引き分けを回避する自己対戦

引き分けが多くなると互いが引き分けになるような手を指し続けるようになり，対局が人間プレイヤが行うガイスターのゲーム内容から乖離するを避けるため，引き分け数が増え過ぎなくなる工夫を導入する．

NEW，OPTION，EXTRA のそれぞれにおいて，100,000 回の自己対戦を行い，500 回ごとに先手勝ち，後手勝ち，引き分けの回数とニューラルネットワークの重みを出力させる．もし，500 回の自己対戦における引き分けになった対局の回数が 260 を超えていた場合，ニューラルネットワークの重みを 500 戦を行う前に戻し，500 戦の学習をなかったことにし，自己対戦を継続させる．また，この 500 戦は自己対戦回数としてカウントしない．つまり，500 戦中引き分けが 260 戦以下になる自己対戦を 200 回行わなくてはならない．学習をなかったことにするため閾値は，500 戦中の引き分け数が全体の三分の二よりも大きい数を取ると長い間引き分けが多くなることが頻繁に起こったため，それ以前の段階で引き分けが多くなることを食い止めるために 260 という値とした．

NEW の各入力を持つプレイヤの自己対戦における引き分けの数は図 5.12 となる．

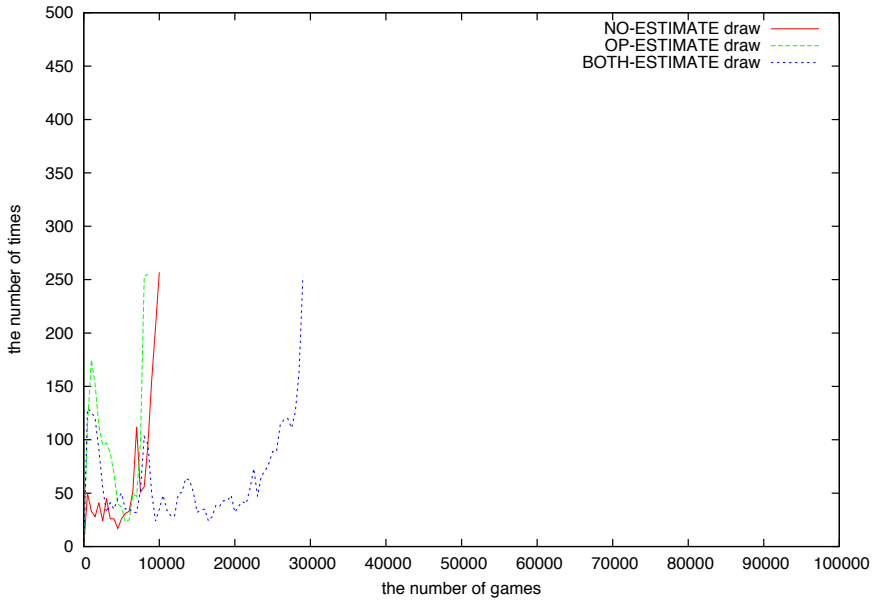


図 5.12: NEW の各 AI における 500 戦ごとの引き分け数

NEW の全てのプレイヤは学習の途中において，引き分けの数が増え，常に引き分けの数が 260 を超えるようになった．その後，数十回も 500 戦中の引き分け数が 260 を超えたため学習を中止した．

OPTION の各入力を持つプレイヤの自己対戦における引き分けの数は図 5.13 となる.

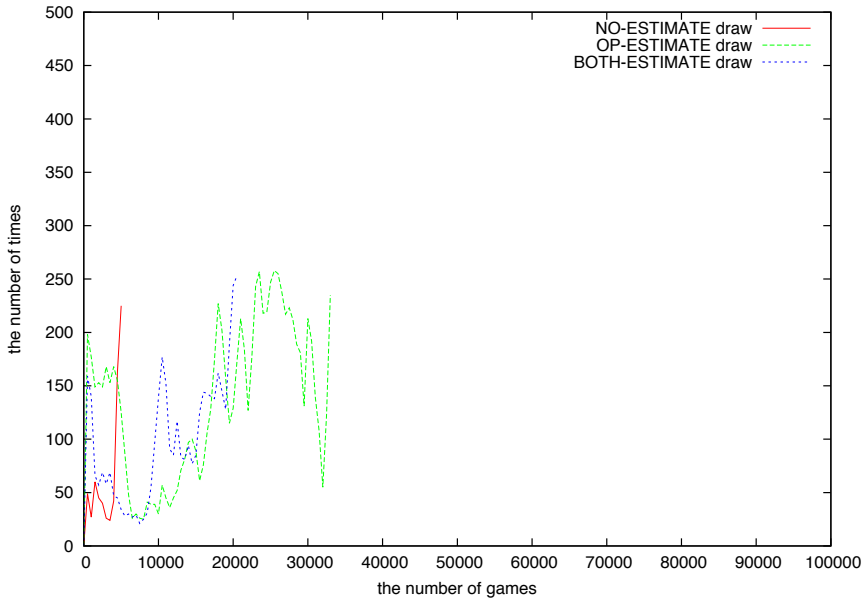


図 5.13: OPTION の各 AI における 500 戦ごとの引き分け数

EXTRA の各入力を持つプレイヤの自己対戦における引き分けの数は図 5.14 となった.

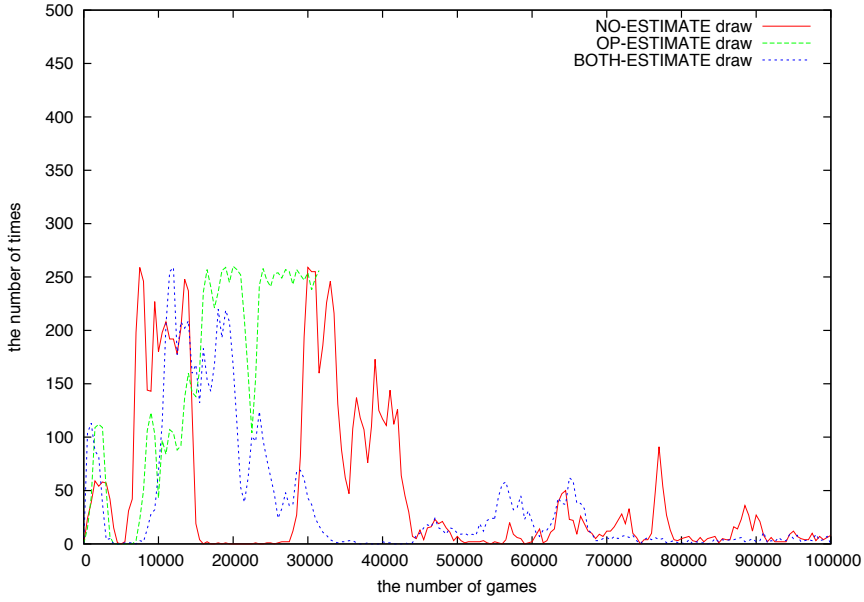


図 5.14: EXTRA の各 AI における 500 戦ごとの引き分け数

表 5.29: EXTRA-NO-EST における学習対戦数と直前の 500 戦の結果

学習対戦数	先手勝ち数	後手勝ち数	引き分け数
22500	249	251	0
31500	162	178	160
44000	235	258	7
69500	264	229	7
100000	258	234	8

表 5.30: 自己対戦回数での EXTRA-NO-EST を入力とする行動価値関数を持つ AI プレイヤとランダムプレイヤとの 3000 戦における結果

	EXTRA-NO-EST 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
15500	724	663	113	650	720	130
22500	879	619	2	619	877	4
31500	306	344	850	364	269	867
44000	283	367	850	361	283	856
69500	672	695	133	694	658	148
100000	424	802	274	789	424	287

表 5.31: EXTRA-BOTH-EST における学習対戦数と直前の 500 戦の結果

学習対戦数	先手勝ち数	後手勝ち数	引き分け数
9000	234	238	28
15500	182	186	132
33000	212	284	4
100000	246	254	0

表 5.32: 自己対戦回数での EXTRA-BOTH-EST を入力とする行動価値関数を持つ AI プレイヤとランダムプレイヤとの 3000 戦における結果

	EXTRA-BOTH-EST 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
9000	1278	27	195	24	1260	216
15500	150	129	1221	153	123	1224
33000	685	682	133	696	671	133
100000	640	783	77	765	646	89

NEW と同様に EXTRA-NO-EST と EXTRA-BOTH-EST 以外において，学習の途中で引き分け数が 260 以上になる続けることを避けられなかったため，学習を打ち切った．これは，攻撃的な手を学習した後，それに対応するための守備的な手を急激に学習している時に学習結果をなかったことにするため，AI プレイヤが守備的な手を指すような流れを止めること，及び，そこから攻撃的な手を学習することが非常に難しいからだと考えられる．

学習した AI プレイヤのランダムとの対局結果から，引き分けが少ない状態で学習を続けたからといって，より AI が強くなっていくとは限らないことがわかる．

次節では，全く引き分けから学習しない場合において AI がどのようになるかを実験により調べる．

5.6　引き分けとなるゲームを除いた学習

勝敗がついた対戦のみから学習させることで積極的な AI プレイヤを作ることができるのかを確認する．ゲームが引き分けになった場合，ニューラルネットワークの重みをそのゲームが行われる前の重みに戻し，そのゲームにおける学習をなかったことにすることにより，勝敗が決するゲームのみで自己対戦による学習を行う．自己対戦の回数のカウントは勝敗が決したゲームのみに対して行う．

NEW，OPTION，EXTRA の各プレイヤの自己対戦における引き分けの数は図 5.15,**??**, 5.17 のようになる．

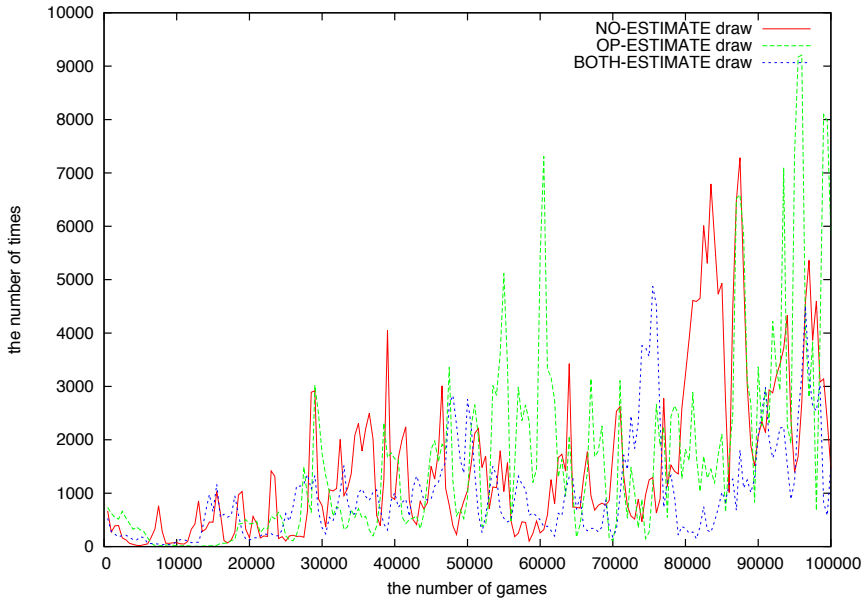


図 5.15: NEW の各 AI における 500 戦ごとの引き分けの回数

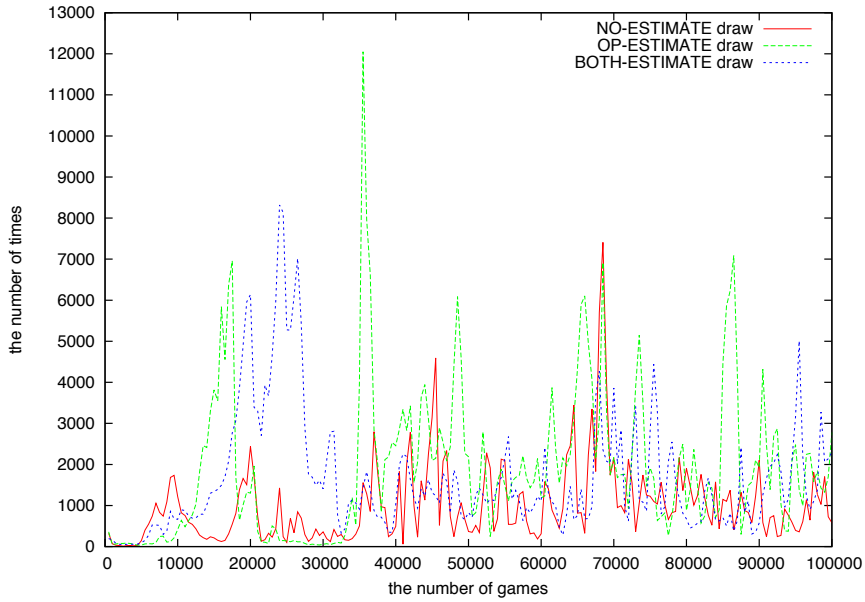


図 5.16: OPTION の各 AI における 500 戦ごとの引き分けの回数

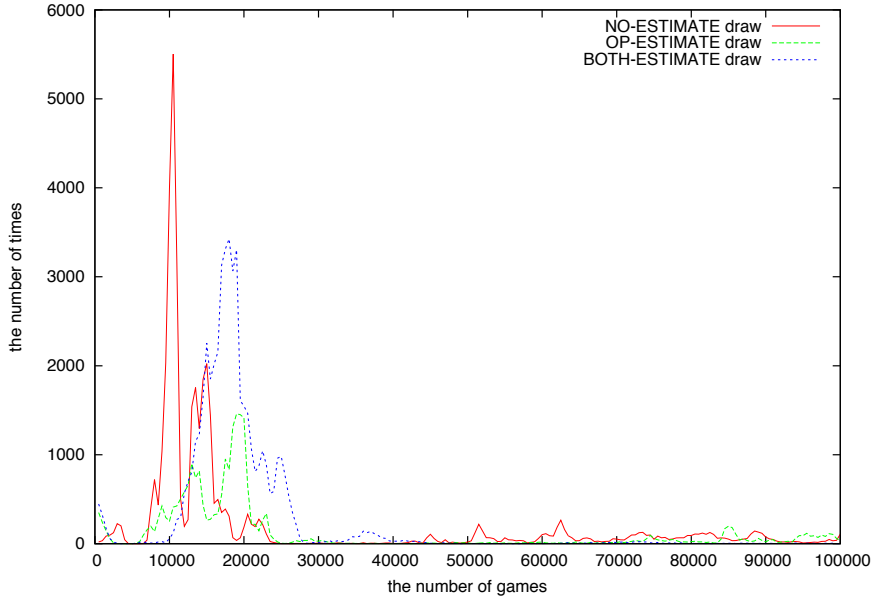


図 5.17: EXTRA の各 AI における 500 戦ごとの引き分けの回数

NEW において，引き分けとなるゲームを取り除くと勝敗がついた対局 500 戦ごとの引き分け対局数が対局回数を重ねるごとに大きくなっていく．NEW では，引き分けを取り除いた学習を行うことで学習における引き分け数を少なくすることは難しい．

NEW は徐々に引き分けが多くなっているように見えるが，OP では引き分けが多くなっているようには見えない．

EXTRA では対戦回数 10000〜26000 において引き分け数が高くなるが，それ以降の引き分け数は非常に小さい値に推移している．これは，EXTRA が引き分けの少ない入力となっているためだと考えられる．

表 5.33 は引き分けとなるゲームを除いた自己対戦による学習を行った EXTRA-BOTH-EST とランダムプレイヤとの 3000 戦における結果である．

表 5.33: 自己対戦回数ごとの EXTRA-BOTH-EST での AI プレイヤとランダムプレイヤとの 3000 戦における結果

	EXTRA-BOTH-EST 先手			ランダム先手		
学習対戦数	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
20000	75	81	1344	87	66	1347
40000	519	620	361	621	538	341
60000	652	821	27	807	669	24
80000	694	754	52	772	670	58
100000	696	690	114	733	676	91

引き分けが多くなる 20000 回の学習では，ほとんどの対局において引き分けとなっているが，引き分けが少ない 40000 回の学習以降では，引き分けは少ない値となっているが，100000 回の学習における引き分け回数は微増の傾向にある．

先手勝ちと後手勝ちの数の差は学習を重ねるごとに少なくなっているが，大きく違うわけでもない．

よって，引き分けを除く方法はあまりいいとはいえない．

次節では，自己対戦 500 回ごとの各勝利条件を満たした回数を分析し，AI がどのような手を学習しているのかを調べる．

5.7　500戦ごとの各勝利条件を満たした回数の解析

新たな入力に対して，自己対戦での学習を行い，自己対戦500回ごとの各勝利条件を満たした回数を出力させ，分析し，AIがどのような手を学習し，指すことができるのかを調べる．

着手後の状態の表現に重点を置いたEXTRAに先手後手を表すユニットと推測値を利用する場合には取る相手の駒の推測値を表すユニットを付加した**LAST-NO-EST**，**LAST-OP-EST**，**LAST-BOTH-EST**を考えた．これらをまとめて**LAST**と呼ぶ．

LAST-NO-ESTの入力は以下のようになっている．

- 先手後手 (1ユニット)
  先手後手を表すユニットとなり，先手なら0，後手なら1を与える．
- 着手後の自分の青駒の配置 (37ユニット)
  自分の青駒がいるマスを表す．
- 着手後の自分の赤駒の配置 (36ユニット)
  自分の赤駒がいるマスを表す．
- 着手前の相手の駒の配置 (36ユニット)
  相手の駒がいるマスを表す．
- 自分が取った相手の青駒の数 (3ユニット)
  自分が取った相手の青駒の数を表現する．
- 自分が取った相手の赤駒の数 (3ユニット)
  自分が取った相手の赤駒の数を表現する．
- 駒を取る着手か否か (1ユニット)
  相手の駒を取る着手ならばユニットに1，そうでなければ0を与える．
- 着手後の出口までの最短距離 (8ユニット)
  脱出させた駒の距離を0，出口のマスの上に乗っている駒の距離を1として数える．つまり，盤上にある駒の着手後の出口までの最短距離はマンハッタン距離に1加えたものとなる．この最短距離に対応するビット列は表5.21において示す．
- 着手後に相手の駒と隣り合う自分の駒の位置 (36ユニット)
  自分が着手をした後に，相手の駒と上下左右で隣接する自分の駒の位置を自分の青駒の配置などと同様の方法で表現する．脱出した駒には隣接する駒はないものとして考える．
- 着手後の自分の駒と隣り合う相手の駒の位置 (36ユニット)
  上と同様の方法で，自分が着手をした後に，自分の駒と上下左右で隣接する相手の駒の位置を表現する．

LAST-OP-EST と LAST-BOTH-EST に加える取る相手の駒の推測値のユニットは以下となる．

- 取る相手の駒の推測値 (**12** ユニット)
  手によって取る相手の駒の推測値を表現する．取る駒がない場合には全てのユニットに 0 を与える．取る駒がある場合には EXTRA-OP-ESTIMATE などで使われる推測値のユニットと同様に 12 ユニットで駒の推測値を表現する．

この LAST-NO-EST に EXTRA-OP-EST と同様の相手の駒に対する推測値 12×8 ユニットと取る相手の駒の推測値のユニットを加えたものを **LAST-OP-EST** とする．

さらに，LAST-NO-EST に EXTRA-BOTH-EST と同様の相手の駒に対する推測値 10×8 ユニットと自分の駒に対する推測値 10×8 ユニット，取る相手の駒の推測値のユニットを加えた入力を **LAST-BOTH-EST** とする．

LAST-NO-EST および LAST-OP-EST，LAST-BOTH-EST におけるバイアス項を含めたニューラルネットワークの入力層と中間層のユニット数は表 5.34 となる．

表 5.34: LAST-NO-EST，LAST-OP-EST，LAST-BOTH-EST における入力層と中間層のユニット数

	入力層のユニット数	中間層のユニット数
LAST-NO-EST	198	101
LAST-OP-EST	306	158
LAST-BOTH-EST	370	187

実験

自己対戦による 1,500,000 回の学習を行い，500 戦ごとに直前の 500 における先手勝ち数および後手勝ち数，引き分け数，青駒を取ることによりゲームが終了した回数，赤駒を取ることによりゲームが終了した回数，脱出によりゲームが終了した回数とそのときのニューラルネットワークの重みを出力させる．

結果

図 5.18,5.19, 5.20 は LAST での自己対戦における 500 戦ごとの各勝利条件を満たした回数となっている．

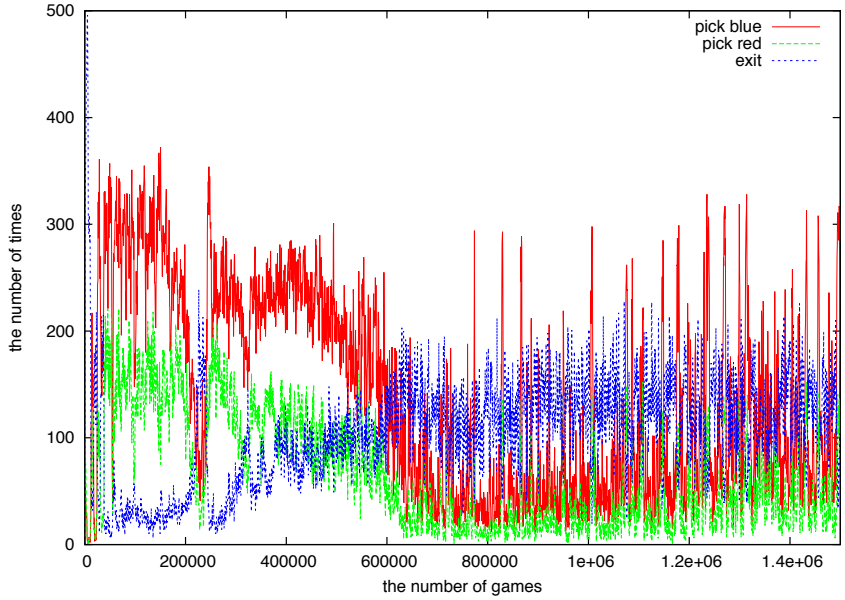


図 5.18: LAST-NO-EST の 500 戦ごとの各勝利条件を満たした回数

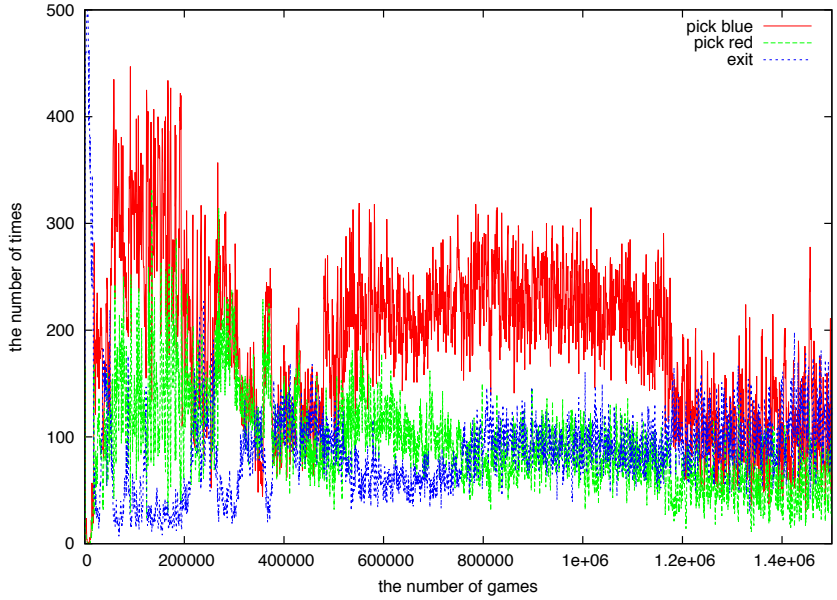


図 5.19: LAST-OP-EST の 500 戦ごとの各勝利条件を満たした回数

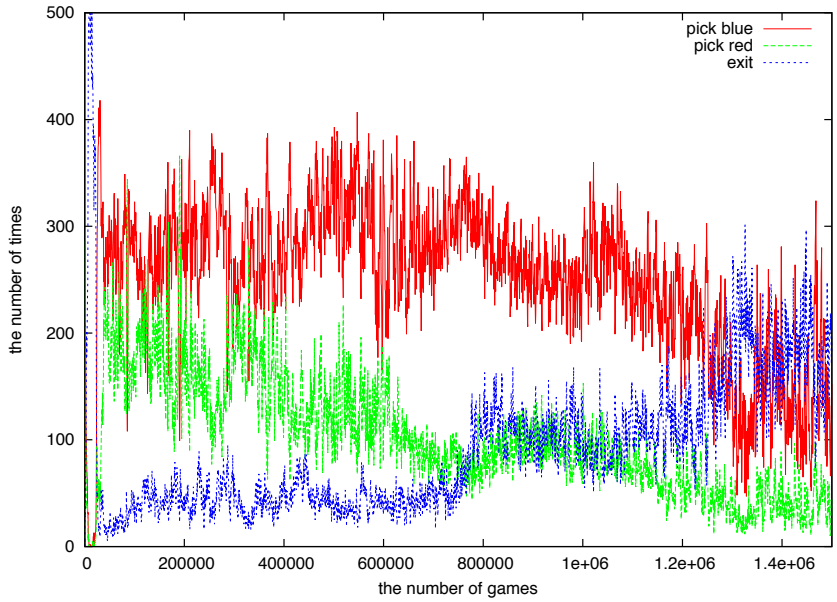


図 5.20: LAST-BOTH-EST の 500 戦ごとの各勝利条件を満たした回数

それぞれの勝利条件を満たした回数は大きく振動しながら，変化している．LAST のそれぞれにおいて，学習の初期では青駒を脱出させることによる決着が 500 に近い数値をとる．その後，青駒を取ることにより決着する回数と赤駒を取ることにより決着する回数が増える．その後，脱出により決着する回数が徐々に増え，駒取りによる決着の回数は低く なる．人間プレイヤにおいては，青駒を取ることにより決着する回数は非常に少ない．それには，人間プレイヤが駒を配置するときに赤駒をより多く前列に並べることが多く，より赤駒を取りやすいため，すぐに全ての赤駒を集めてしまうことや，青駒を脱出させる必勝手が生まれることが多いことが挙げられる．しかし，LAST を入力とした自己対戦において，青駒を取ることによる決着数が非常に多くなった．

より細かく見ると，脱出による決着数が増えた後に，青駒を取ることによる決着数が増える現象や，青駒を取ることによる決着数が増えた後に，赤駒を取ることによる決着数が増える現象などが観察できた．これは，ある勝利条件を満した回数が増えた後にそれを対策する手が増え，別の勝利条件を満たした回数が増えていると考えられる．

各勝利条件を満たした回数が興味深い数値となっているニューラルネットワークの重みをサンプリングし，どのような自己対戦を行っているかを確認した．LAST-BOTH-EST における自己対戦 192500 回により得られた行動価値関数を用いた AI プレイヤでは，人間プレイヤがよく用いる序盤の定石や赤駒を相手の出口に進めようとすることで青駒だと相手に思わせて取らせようとする手が多く見られた．

表 5.35 は自己対戦回数 192500 回の直前 500 試合における結果である．

表 5.35: LAST-BOTH-EST における自己対戦の結果

学習対局数	青駒取りでの試合終了数	赤駒取りでの試合終了数	脱出での試合終了数
192500	171	279	41

その序盤の定石は図 5.21 となっている．

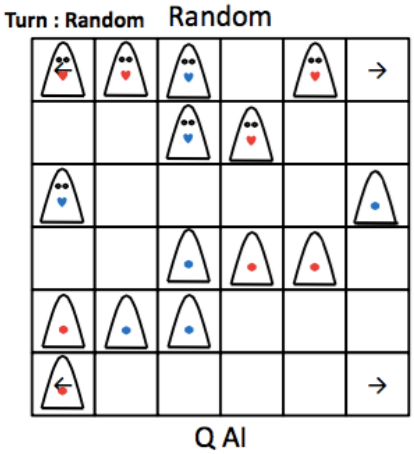


図 5.21: LAST-BOTH-EST とランダムプレイヤでの対局に見られる序盤定石

これは，相手との盤上の駒数の差がつきにくくなる陣形であり，相手が駒に隣接させてきた場合にはすぐに駒を取ることができ，相手に駒を取られた場合には取られた駒と隣接状態にあった駒で相手の駒を取ることができるという手になっている．盤上の駒数に差がついてしまうと，駒がいない間を縫って青駒を脱出させやすくなる．よって，盤上の駒数に差がつきにくくする陣形を序盤で敷くことは非常に重要となる．

さらに，自己対戦においてブラフとなる手をどのように指しているのかを確認した所，図 5.22 の赤駒を出口に近づけようとする手が見られた．

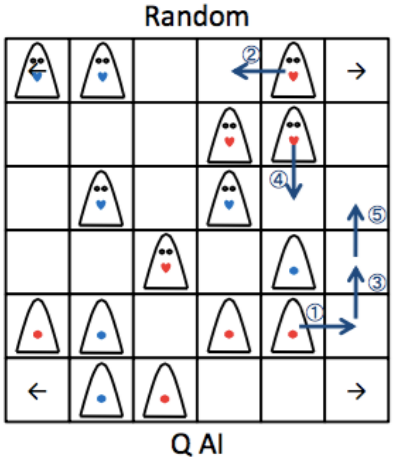


図 5.22: LAST-BOTH-EST とランダムプレイヤでの対局に見られる序盤定石

これは，相手の駒との隣接を避け，赤駒を出口に近づけ，相手に赤駒を取らせようという手であり，人間プレイヤがよく使うブラフとなる手である．人間プレイヤ同士の対局において，このような手が指された場合，駒が青であったときを考え，駒を取ることが非常に多い．モンテカルロ木探索では，このようなブラフとなる手を指すことは出来なかった．図 5.23 は自己対戦 192,500 回付近での対局におけるブラフとなる手である．

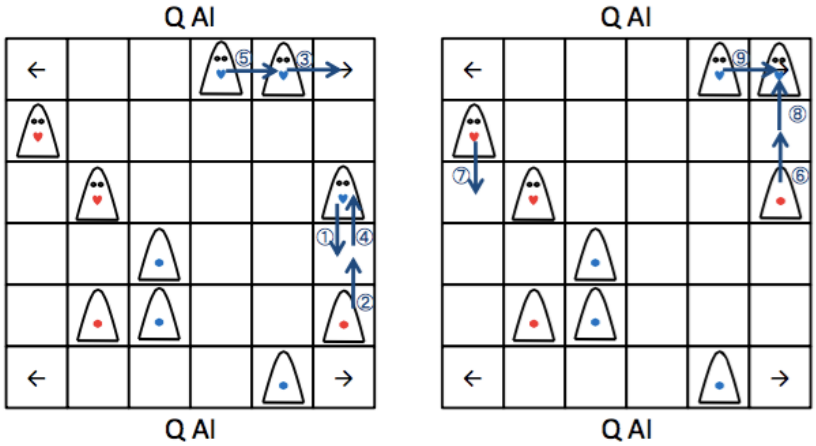


図 5.23: 自己対戦 192,500 回付近での対局におけるブラフとなる手

実験

これらの特殊な手を指すことができるようになった自己対戦を 192500 回行った LAST-BOTH-EST を入力とする行動価値関数を利用した AI プレイヤとランダムプレイヤとの 3000 戦の試合を行う．

結果

表 5.36 は 192500 回の学習を行った LAST-BOTH-EST を用いた AI プレイヤとランダムプレイヤにおける 3000 戦の試合の結果となっている．

表 5.36: LAST-BOTH-EST 自己対戦回数とランダムプレイヤとの 3000 戦における結果

	LAST-BOTH-EST 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
192500	678	711	111	716	688	96

ランダムプレイヤに対して，LAST-BOTH-EST を用いた AI プレイヤは負け越している．LAST-BOTH-EST により学習を行った行動価値関数を用いた AI プレイヤとランダムプレイヤの対局において，AI プレイヤは 2 手以上の必勝となる脱出手を理解することが出来ず，ランダムプレイヤに負ける場面が度々見られた．これは直前の 500 戦における脱出による勝敗の割合が非常に低く，脱出手が評価されておらず，より駒取りによる決着での更新回数の多かったため，脱出よりも駒取りにより決着する手の行動価値を高く見積もっていることが考えられる．

よって，数手先までの脱出手を探索し，必ず脱出することができる局面を認識することができれば，ランダムプレイヤに勝ち越すことが可能だと考えられる．また，相手がランダムに手を指すため，推測値が乱されることもランダムプレイヤに負け越した原因として考えられる．

実験

UCT を利用したモンテカルロ木探索を用いた AI プレイヤとの対局を行なう．対局を行なうにあたり，モンテカルロ木探索におけるプレイアウト数は 100,000 とし，プレイアウトは完全ランダムとする．初期局面における 100,000 プレイアウトを行なうモンテカルロ木探索を用いた AI プレイヤの思考時間は約 3.7 秒程度である．

結果

表 5.37: LAST-BOTH-EST 自己対戦回数と MCT プレイヤとの 1000 戦における結果

	LAST-BOTH-EST 先手			MCT 先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
192500	195	287	18	328	152	20

モンテカルロ木探索を用いたプレイヤに対して，LAST-BOTH-EST を入力とした行動価値関数を利用した AI プレイヤは負け越す結果となった．

対局の内容を確認した所，モンテカルロ木探索を用いた AI プレイヤは消極的な自陣でまとまる手を指し，駒をあまり取らずに，自分の赤駒を取らせようとする．一方，行動価値関数を利用する AI プレイヤは定石である自駒が隣接する形を形成しながら，積極的に駒取りを行い攻めていき，最後は赤駒を取り負ける場面が非常に多かった．自己対戦 500 戦中における赤駒取りによるゲーム終了の回数が多いということは，相手に赤駒を取らせて勝つ回数と自分が赤駒を取って負ける回数が多いことを意味している．よって，自己対戦 192,500 回で得られた行動価値関数を用いた AI プレイヤはモンテカルロ木探索を用いたプレイヤに対して相性が非常に悪いことが考えられる．

別の戦略を獲得している行動価値関数を利用する AI プレイヤでモンテカルロ木探索を用いたプレイヤに対して勝つことができる可能性は否めない．

5.8 必勝手探索を加えた AI プレイヤ

前節において自己対戦回数 192,500 回で得られた行動価値関数を用いた AI プレイヤは必勝となる局面において，必勝手がわからないために多くの対局を落としていた．よって，必勝手探索を用い，簡単な必勝手を見逃さない工夫を加える．

前節における自己対戦回数 192,500 回で得られた行動価値関数を用いた AI プレイヤに Df-pn アルゴリズムによる必勝手探索を加えた AI プレイヤを作成した．以降，この AI を Q-AI-Dfpn と呼ぶ．Q-AI-Dfpn は 150msec の必勝手探索を行い，必勝手が見つかった場合には必勝手を，必勝手が見つからなかった場合には行動価値関数の値が最大となる手を指す．

実験

Q-AI-Dfpn とランダムプレイヤおよびモンテカルロ木探索を用いた AI との対局実験を行なう．モンテカルロ木探索は前節と同様の手法にて行なう．

結果

表 5.38: Q-AI-Dfpn とランダムプレイヤとの 3000 戦における結果

	Q-AI-Dfpn 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
Q-AI-Dfpn	967	476	57	440	1004	56

表 5.39: Q-AI-Dfpn と MCT プレイヤとの 2000 戦における結果

	Q-AI-Dfpn 先手			MCT 先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
Q-AI-Dfpn	622	353	25	216	556	28

Q-AI-Dfpn がランダムプレイヤおよび MCT プレイヤの両プレイヤに対して勝ち越すという結果となった．これは今まで多く見逃していた必勝手を指すことが可能になり，勝つことができる対局をみすみす逃さなくなったためである．

5.9 必勝手探索を加えた自己対戦による学習

自己対戦による Sarsa(λ) を行ってきたが，自己対戦中において，必勝手探索により必勝手が見つかった場合，必ず必勝手を指すようにし，Sarsa(λ) 学習を行なう．

なお，学習を行なうにあたり，LAST-BOTH-EST の入力に変更を加えた．変更後の入力を用いる AI プレイヤを **WIN-HAND** と呼ぶ．

WIN-HAND の入力は以下のようになっており，LAST-BOTH-EST から変更があった特徴は太字で示されている．

- 先手後手 (1 ユニット)
  先手後手を表す．
- 着手後の自分の青駒の配置 (37 ユニット)
  自分の青駒がいるマスを表す．
- 着手後の自分の赤駒の配置 (36 ユニット)
  自分の赤駒がいるマスを表す．

- 着手前の相手の駒の配置 (36 ユニット)
  相手の駒がいるマスを表す.

- 自分が取った相手の青駒の数 (3 ユニット)
  自分が取った相手の青駒の数を表現する.

- 自分が取った相手の赤駒の数 (3 ユニット)
  自分が取った相手の赤駒の数を表現する.

- 駒を取る着手か否か (1 ユニット)
  相手の駒を取る着手を表現する.

- 取る相手の駒の推測値 (12 ユニット)
  手によって取る相手の駒の推測値を表現する.

- 着手後の出口までの最短距離 (8 ユニット)
  着手後の青駒の出口までの最短距離を表現する.

- 着手後の敵駒の脱出阻止の可否 (1 ユニット)
  相手の全ての駒に青駒の可能性があると考えた際，相手の駒を取ることにより脱出を防ぐことができるかどうかを表現する．脱出を防げる場合には 0，脱出を防ぐことが出来ない場合には 1 とする.

- 着手後の相手の駒と隣り合う自分の青駒の位置 (36 ユニット)
  自分が着手をした後に，相手の駒と上下左右で隣接する自分の青駒の位置を自分の青駒の配置などと同様の方法で表現する.

- 着手後の相手の駒と隣り合う自分の赤駒の位置 (36 ユニット)
  同様に，相手の駒と上下に隣接する自分の赤駒の位置を表現する.

- 着手後の自分の駒と隣り合う相手の駒の位置 (36 ユニット)
  上と同様の方法で，自分が着手をした後に，自分の駒と上下左右で隣接する相手の駒の位置を表現する.

- 相手の駒に対する推測値 (10×8 ユニット)
  相手の駒に対する PBL によって求めた推測値を表す.

- 自分の駒に対する推測値 (10×8 ユニット)
  自分の駒に対する PBL によって求めた推測値を表す.

WIN-HAND におけるバイアス項を含めたニューラルネットワークの入力層と中間層のユニット数は表 5.40 となる.

表 5.40: WIN-HAND における入力層と中間層のユニット数

	入力層のユニット数	中間層のユニット数
WIN-HAND	414	209

実験

自己対戦による必勝手探索を組み込んだ Sarsa(λ) 学習を行う．前節と同様の必勝手探索を 150msec 行い，必勝手が見つかった場合には行動価値関数に依らず必勝手を指すようにする．

結果

図 5.24 は WIN-HAND での自己対戦における 500 戦ごとの各勝利条件を満たした回数となっている．

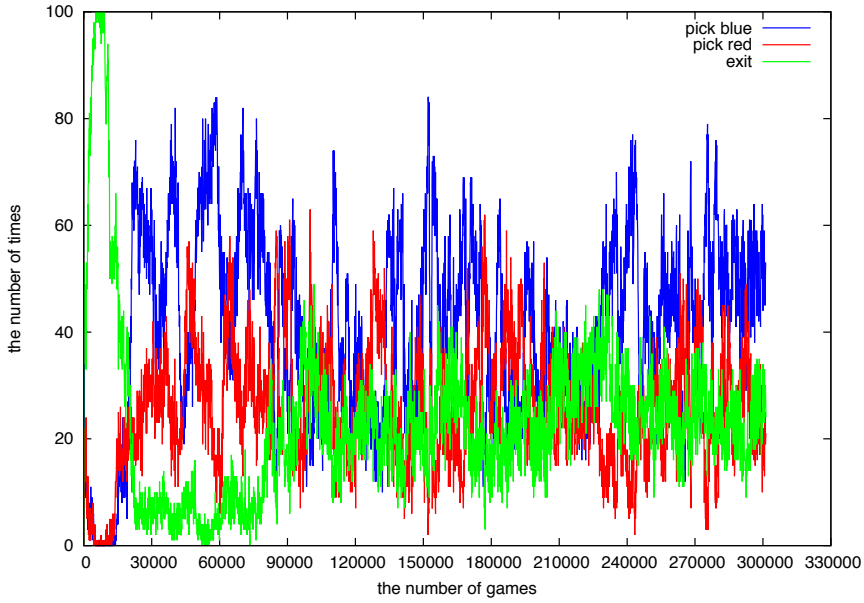


図 5.24: WIN-HAND の 500 戦ごとの各勝利条件を満たした回数

図 5.24 から各勝利条件を満たした回数は収束していないことわかる．しかし，必勝手探索を加えたことにより，各勝利条件を満たした回数は大幅に変化している．

実験

130,100 回の自己対戦によって得られた行動価値関数を用いる AI プレイヤを作成し，これに Df-pn アルゴリズムによる必勝手探索を加えた AI プレイヤでの対局実験を行なう．

以降，この AI プレイヤを Q-AI2-Dfpn とする．なお，必勝手探索は 150msec とする．

結果

表 5.41: Q-AI2-Dfpn とランダムプレイヤとの 1000 戦における結果

	Q-AI2-Dfpn 先手			ランダム先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
Q-AI2-Dfpn	248	134	118	105	268	127

表 5.42: Q-AI2-Dfpn と MCT プレイヤとの 1000 戦における結果

	Q-AI2-Dfpn 先手			MCT 先手		
	先手勝ち	先手負け	引き分け	先手勝ち	先手負け	引き分け
Q-AI2-Dfpn	274	202	28	245	236	19

Q-AI2-Dfpn がランダムプレイヤおよび MCT プレイヤの両プレイヤに対して勝ち越すという結果になった．しかし，MCT プレイヤに対しては勝数にそれほど大きな開きがない．

第6章　おわりに

6.1　まとめ

適当に手を指すとき，ゲーム自体が引き分けとなりやすいガイスターにおいて，着手制限 Hand-SuperLimit2 を AI に課した状態で学習を行い，AI に同様の着手制限をかけることにより，強化学習を行った AI がまともにゲームをプレイし，序盤の定石やブラフとなる手を指すことができるようになった．しかし，AI の強さは非常に弱く，ランダムプレイヤにも勝てない強さである．ランダムプレイヤとの対局においては必勝手の見逃しが非常に多く見られた．必勝手の見逃しを防ぐため，Df-pn アルゴリズムを用いた必勝手探索を提案し，AI に導入したところ，ランダムプレイヤおよび MCT プレイヤに勝ち越すことができた．必勝手探索を組み込んだ AI での Sarsa(λ) 学習を行ったが，学習によって得られた AI である Q-AI2-Dfpn では，MCT プレイヤに対して大きく勝ち越すことはできなかった．現状では，最も強いと思われる学習段階を知ることは出来ないという問題により，よりよい行動価値関数を手に入れることが不可能をなっている．この問題を解決することにより，より強い AI を作成することができると考えられる．

6.2　今後の課題

今回は LAST-BOTH-EST における自己対戦 192500 回での行動価値関数を利用したが別の学習回数での行動価値関数を用いた AI プレイヤがランダムプレイヤやモンテカルロ木探索を用いたプレイヤに対して強くなる可能性があるため，どの自己対戦回数において最もよい行動価値関数が得られるのかを調べる手法が必要となる．

今回，着手に制限を課す手法を用いたが，自己対戦での引き分けが多くなり過ぎないように着手制限を緩めることで，AI がより多様な着手を行なうことができる．よって，学習の効率と着手制限の最適なバランスを見つけることも非常に重要となる．ガイスターにおいては勝利条件が 3 つある．よって，それぞれの勝利条件を満たす確率の見積もりを計算する行動価値関数を獲得することで，様々な局面に対して，より柔軟に対応できると考えられる．ニューラルネットワークの出力を 3 つに増やし，各出力で 3 つの勝利条件を満たす確率の見積もりを計算することが今後の課題である．

自己対戦により学習を行うため，様々な戦略に対応できる AI プレイヤを作成することが出来ないことも今後の課題となっている．様々な学習段階での AI プレイヤを作成し，作成した AI プレイヤ同士の複数の組合せで対戦させ，学習を行なうことで，この問題を解決できる可能性がある．

必勝手探索により，必勝手が求められるようになったことで，相手のある駒の色を青だと仮定すると相手の青駒を脱出させる必勝手がある局面において，相手がその必勝手を指さなかったことにより，その駒の色を赤だと特定することが可能となる．この赤特定のアルゴリズムを AI に組み込むことにより，より正確に敵駒の色を判断することが可能となり，より AI が強くなることが可能となると思われる．

6.3 謝辞

本研究を進めるにあたり，非常に多くの先生方および研究室の皆様に深く御礼申し上げます．村松正和教授には，私のわがままで囲碁の研究から本研究に移ることを許可して頂き，さらにはご指導まで頂き，有難うございました．保木邦仁准教授，西野順二助教には本研究において，御指導，御協力を賜り，大変参考になりました．有難うございました．

岡本吉央准教授，高橋里司助教には学生としての生活面での様々なアドバイスを頂きました．研究者として生活する上で非常に役に立ちました．有難うございました．

川瀬先輩には，研究室生活のあれこれや悩んでいるときに付き合って頂きました．この恩は一生忘れません．祐成先輩には，必勝となる局面の条件や後退解析に関する議論やゲーム終了までの平均手数の調査に付き合って頂き，非常に感謝しております．

荒木先輩と荒井先輩には，研究において困ったときにしつこく質問や議論をさせていただきました．非常に勉強になり，研究における知識を深めることが出来ました．ただただ感謝の気持ちでいっぱいです．

他の研究室の方々にも多大な助けを頂きました．本研究は皆さんのお力添えによって，進めることが出来たものだと，感謝しております．

最後に，このガイスターという非常に面白く，奥の深いゲームを生み出してくださった Alex Randolph への尊敬と感謝の意を表することで謝辞とさせていただきます．

参考文献

[1] Stuart Russell, Peter Norvig, *Artificial Intelligence: A Modern Approach*, Pearson, (2010).

[2] Michael Buro, "The Othello Match of the Year: Takeshi Murakami vs. Logistello", ICCA Journal, 20.3, (1997).

[3] Campbell, Murray, A. Joseph Hoane Jr and Feng-hsiung Hsu, "Deep blue", Artificial Intelligence, 134.1, (2002).

[4] 第 3 回 将棋電王戦 HUMAN VS COMPUTER - niconico, http://ex.nicovideo.jp/denou/3rd/match.html visited on 2016/01/22.

[5] 第 2 回 電聖戦, http://entcog.c.ooco.jp/entcog/densei/denseisen-2nd.html visited on 2015/01/22.

[6] https://ghosts-challenge.math.unipd.it/public/docs/2013/bliss.pdf

[7] Gerald Tesauro, "Temporal difference learning and TD-Gammon"，Communications of the ACM 38.3,(1995).

[8] http://www.mobius-games.co.jp/Gester.htm

[9] https://ghosts-challenge.math.unipd.it

[10] Kuniaki Uehara, Masayuki Tanizawa, Sadao Maekawa, "PBL: Prototype-Based Learning Algorithm".

[11] F.Aiolli, C.E.Palazzi. "Enhancing Artificial Intelligence in Games by Learning the Opponent's Playing Style". Proceedings of the IFIP-ECS Conference,(2008).

[12] 松原仁, 美添一樹, 山下宏. コンピュータ囲碁　モンテカルロ法の理論と実践, 共立出版株式会社 2012, pp.210.

[13] L.Kocss acd C.Szepesvari. Bandi based Monte-Carlo planning. In 17th European Conf. on Machine Learning(ECML 2016). pp.282-293, (2006).

[14] 平野 廣美,C でつくるニューラルネットワーク, パーソナルメディア,(1991).

[15] A. Nagai, "Df-pn Algorithm for Searching AND/OR Trees and Its Applications", Ph. D. thesis, Department of Information Science, University of Tokyo,(2002).