博　　士　　論　　文

報酬が疎なゲームにおける
強化学習を用いたプレイヤの構築

指導教員　　鶴岡 慶雅 准教授

東京大学工学系研究科
電気系工学専攻

氏 名　　**37-157079** 水上直紀

提 出 日　　平成 **29** 年 **12** 月 **1** 日

概 要

コンピュータゲームの研究は人工知能のテストベッドとして発展してきた．近年では将棋や囲碁，ポーカーなどでは計算機が人間のトップと同等以上の性能を達成している．ゲーム AI はルールとその特性から現実世界の問題を研究する方法の一つと考えれている．政治や経済などといった現実世界の問題ではゲームにおいて報酬がめったに観測されない，すなわち報酬が疎な環境であるとみなすことができる．そこで本研究では報酬が疎であるゲームを対象に強いコンピュータプレイヤを開発するための手法を提案する．ゲームとしては Atari と麻雀を対象とした，強化学習や教師あり学習の手法を組み合わせることで提案手法の多くについてその有効性を示した．

多くの Atari のゲームでは最近の深層強化学習の手法は人間を超える性能を出している．しかしながら本研究が対象とする報酬が疎のゲームではその性能は人間にはるかに及ばない．一つの解決方法はエージェントの学習時に明示的で賢い探索戦略を組み込むことである．本研究では学習の進行具合を明示的に考慮した Upper Confidence Bounds (UCB) を用いた探索ボーナスを用いる効果的な探索戦略を提案する．さらに提案手法では探索ボーナスを報酬から分離することで本来の目的関数の学習を妨害を回避する仕組みを導入する．これにより報酬が疎なゲームにおいてベースラインとなる asynchronous advantage actor-critic よりも性能が向上した．

麻雀は Atari と異なる性質を持つゲームではあるが，Atari と同様に報酬が疎なゲームである．本研究では麻雀の繰り返しゲームの性質に着目し，一ゲームをそのまま考慮する代わりに一局単位を最適化する方法を提案する．そのため最終順位を考慮したコンピュータ麻雀プレイヤの構築法について述べる．Atari と異なり大量に取得可能な牌譜中に現れた点数状況から最終順位を予測するモデルの学習を行う．モンテカルロ法のシミュレーションでの報酬を予測モデルの結果を用いることで最終順位に基づく手をプログラムは選択する．実験により局収支を最大化するプレイヤと比較して最終順位を考慮したプレイヤの性能が高いことを示した．

次に上記の麻雀プレイヤを利用し強化学習を用いた麻雀プレイヤを構築する方法について述べる．初めに手牌から和了点数を予測するモデルを生成した牌譜から学習する．このモデルの結果と期待最終順位を用いて効率的な和了を行う手をプログラムは選択する．得られたプログラムは高い点数を和了する技術を獲得したものの，自己対戦による評価実験では元のプログラムに勝ち越すことはできなかった．

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　背景

人工知能とは人間と同様の知能を計算機で実現しようとする試みである．ゲームはルールが明確に定義されており，勝敗による評価や人間との比較が容易であることから，人工知能のテストベッドとして用いられている．2 人零和確定完全情報ゲームであるチェス，オセロ，将棋といったゲームでは人間のトッププレイヤと同等かそれ以上の実力を持つコンピュータプレイヤが実現されている [11, 13, 70]．これらの思考エンジンを構築するための手段としては，プロ棋士の棋譜による教師あり学習が有効であった．しかしながら棋譜生成はプロ棋士が長い時間をかけて生成する必要があるためコストが高く十分な量の棋譜を集めることは難しい．また，トッププレイヤを超える実力となったこれらのゲームではこれまで人間の常識として行っていたことが局所的な解である可能性が高く，棋譜の価値も相対的に下がっている．ゲーム AI の実力向上のためにはこれらの常識にとらわれないようにするため，人の知識を使わないまたは人の影響が小さいプレイヤが求められる．

そこで人間の知識ではなくゲームの「報酬」を用いてエージェントを賢くする強化学習が注目されている．強化学習とは，エージェントがある環境において経験から累積される報酬を最大にする行動を自動的に学習する機械学習の一種である．この学習の枠組みは汎用的であるため様々な分野において応用が期待されている．

ゲームの分野において古くはバックギャモンが強化学習を取り入れて成功を収めた [53]．バックギャモンはサイコロを振り，出た目によってコマを進めすべてのコマをゴールに進めたほうが勝つ双六のようなゲームである．バックギャモンが強化学習を取り入れ成功したのは二つの条件が成立していたからだと考えられる．その条件は

1. 行動と報酬の結びつきが容易であること
2. 局面状態が複雑でないこと

である．バックギャモンはランダム指し手を進めても勝敗という報酬が得られるため，そのため行動と報酬の結びつきが容易である．またバックギャモンではコマが配置できる箇所は 26 箇所と囲碁や将棋と比べて少なく，またコマの進む方向も一方向だけであり，局面状態が囲碁将棋と比較して簡単である．そのためバックギャモンの強化学習が成功したと考えられている．

局面状態の複雑さを適切に表現する方法として，近年では深層学習を用いて盤面を表現する方法が用いられている．AlphaZero は囲碁，将棋，チェスの 3 種類ゲームの自己対戦の結果を用いて従来手法による最強プログラムに勝つまでなった [46]．AlphaZero では最低限の知識のみから学習されており，そのゲーム固有の知識を一切使用していない．

これらの結果から，強化学習の成功する理由のうち局面状態の複雑さに関しては深層学習により対応することが可能になったと考えられる．バックギャモン，囲碁，将棋，チェスといった古典的なゲームは完全情報ゲームかつ局面遷移が既知な環境である．完全情報ゲームとはすべてのプレイヤから見た情報が等しいゲームである．また局面遷移が既知な環境とはプレイヤの行動を選択した際の次の局面が自明なゲームである．これらのゲームではゲーム木の探索を行うことでゲームの終端ノードの情報を利用することが容易である．そのため局面状態と報酬が容易に結びつきやすく強化学習が効率的に働いたと考えられる．

そのため今後取り組むべき課題は行動と報酬の関係性を理解することである．行動と報酬との関係性を理解することが困難であるとは，ある行動によって報酬を得るまでに時間がかかることである．言い換えればある程度長期的な戦略をとらなければ報酬が得られないことである．このような事態が起きるゲームとは，報酬を得られる機会が非常に少ない報酬が疎なゲームである．

この報酬が疎なゲームに対して疑似的な報酬を設計することで対応した研究がある．そしてこれらのゲームは不完全情報ゲームや局面遷移が未知な環境のゲームである．ここではビデオゲームに対象とした研究を説明する．Pathak らは局面遷移の予測の難しさを「好奇心」とすることで報酬の自動生成を行った [40]．この研究では観測された局面と次局面から行動を予測するモデルと局面と行動から次局面を予想するモデルを用いて，その誤差を疑似的な報酬としていえる．また Burda らは状態を入力とする予測問題の二乗誤差を疑似的な報酬を設計した [10]．これらの手法により VizDoom と Super Mario Bros.，Montezuma’s Revenge といったゲームにおいて成功を収めた．

これまでの研究は人間の知識を一切使用しない研究であったが，ゲーム自体が複雑になると人間の知識を利用する研究もある．例えば StarCraft は不完全情報ゲーム，リアルタイム性があり長期的な戦略を求められるゲームである．StarCraft を対象にした研究では人間のプレイを使用した教師あり学習を行い，さらに強化学習を行うことで人間のトッププレイヤに勝つまでになった [37]．

1.2 本研究の提案

本研究では報酬が疎なゲームに対して強化学習を用いたプレイヤの構築を提案する．図 1.1 はこの博士論文の全体像を表している．本研究の提案は報酬が疎なゲームに対してゲームから得られる真の報酬だけでなく，疑似的な報酬を設計し，利用することで学習を効率的に行うことである．本研究で対象となるのは報酬が疎なゲームであり，かつ局面遷移が未知な環境のゲームや不完全情報ゲームである．これらのゲーム性を同時に扱うのは困難であるため，局面遷移が未知な環境のゲームと不完全情報ゲームの二つに分けて研究を行うことにした．そしてこれらの性質を満たすゲームとして本研究では Atari と麻雀を対象とした．本研究はゲームにより二つに構成されており第 2 章で Atari，第 3 章，第 4 章，第 5 章，第 6 章で麻雀について実験を行っている．

Atari はビデオゲームであり行動によりどのようにゲームの局面がどのように遷移するかわからない未知の環境のゲームである．すなわち古典的なゲームで行われている探索は不可能であり，しかもゲームの状態を画像のように扱うためゲームの状態数が非常に大きい．そのため背景の研究でも触れたように未知の局面に対する疑似的な報酬を設計することが重要になる．強化学習では報酬が得られた局面を探索することも重要であるがそうでなくても探索する価値のある局面を探索する必

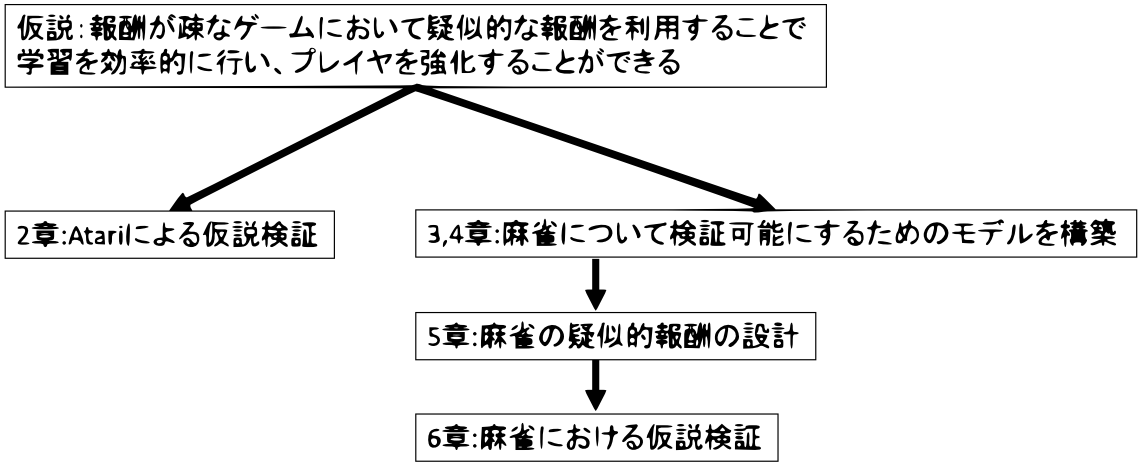

図 1.1. 博士論文の全体像

要がある．これを実現するため実際の報酬が与えられていない局面であっても，探索する価値を疑似的な報酬として設計し，それをエージェントに与えることで未知の局面を効率的に探索することが可能になると考えられる．

麻雀は Atari とは異なるゲーム性を持ったゲームである．ゲームの遷移自体は既知である環境ではあるが，不完全情報ゲームや多人数，繰り返しゲームという性質がある．これらのゲーム性により麻雀は状態の理解が非常に難しいゲームであるといえる．そのため人間の知識や牌譜を使用せず，強化学習を行ったとしても学習が効率的に行われないと考えられる．そこで麻雀では StarCraft の研究のように，まず人間の牌譜や知識を用いてある程度強い麻雀 AI を構築しそこから強化学習を用いることで更なる実力の向上を図る．図 1.1 で説明している通り，第 3 章と第 4 章は不完全情報ゲームや多人数に対応した麻雀 AI を構築する．次に第 5 で疑似的な報酬の設計とそれを用いた麻雀 AI を構築する．最後に第 6 章でこれまでの麻雀 AI を構築するために用いたモデルを利用して，強化学習を用いる．

1.3 本研究の貢献

本研究では以下の貢献が挙げられる．

1. Atari の報酬が疎なゲームにおいても効率的な学習に成功した
   疑似的な報酬を用いることで中長期的な行動をとることが可能になり，特定のゲームにおいて比較した手法の中で最高の成績を出すことに成功した．

2. 麻雀の有効な疑似的な報酬を設計することに成功した
   麻雀が繰り返しゲームであるということに着目し，点数状況から疑似的な報酬を設計した．強化学習を用いていないため本研究の副次的な貢献ではあるが，報酬が疎なゲーム麻雀において

この疑似的な報酬を利用することでトッププレイヤ並みの実力を得ることに成功した．この結果から麻雀における有効な疑似的な報酬を設計することに成功したといえる．

3. 強化学習を行うことで麻雀の序盤において状況に応じた戦略をとれる可能性を示した
麻雀における疑似的な報酬を利用した強化学習を行った．結果として元の AI を超える実力は得られなかった．しかしながら序盤において状況に応じた適切な戦略をとれる局面も確認できており，疑似的な報酬を用いた強化学習は一定の成果を収めた．

1.4 本論文の構成

本論文の構成は以下のようになっている．第 2 章で Atari について研究結果を述べる．第 3 章と第 4 章では不完全情報や多人数性に対応するための AI を人間の牌譜をもとに構築した．第 5 章では麻雀の繰り返しゲームに着目し麻雀における点数を最終的な順位に変換する方法を考え，それを用いることでより強くなった AI について説明する．第 6 章ではこれまでに構築した AI を利用して，自己対戦を用いることで局面を生成し序盤戦術の改善を試みた．第 7 章でこれまでのまとめを行う．

第2章　報酬が疎な環境におけるUCBを用いた探索報酬の提案

2.1　はじめに

ロボット，自動車，ドローンといった実世界の機械を制御する方法を自動的に獲得する技術として，深層強化学習技術が広く注目されるようになった．しかし，実環境での実証実験には費用や時間のコストが非常にかかるという大きな課題があるため，基盤となる深層強化学習技術の開発は，将棋／囲碁／ビデオゲームといった明確なルールと仮想的な環境が存在するゲームを用いて行われることが多い．例えば，Arcade Learning Environment (ALE) [4] に実装されている Atari 2600 のゲームは，深層強化学習法のベンチマークとしてよく用いられている．こういった仮想環境において，事前にそのゲームのルールを与えられていない状態から[1]，ゲームの画面（まはた局面）とスコア（または勝敗）のみの情報を用い，実際にプレーしながらその環境（ゲーム）に適したエージェントの操作戦略を自動で構築する，という最終的には実環境への適用へつながるタスク設定がしばしば深層強化学習の研究に用いられている．

一般論として，Deep Q-Network (DQN) [31] といった近年開発された深層強化学習法を用いて構築された Atari 2600 中の各ゲームをプレーするエージェントは，ゲーム画面とスコアの情報のみから学習を行い，人間の平均的なプレーヤーのレベルを超えていると言われる．しかし実際は，強化学習によって構築されたエージェントがうまく機能するかどうかは，ゲームの性質によって大きく異なる．Bellemare らは，文献 [5] の中で，強化学習のアルゴリズムにとって,「良い報酬が得られる局面を探索するのが容易か」という観点で各ゲームの難易度を分類している．具体的には，探索が容易と考えられるゲームを**探索容易（easy exploration）ゲーム**，困難と考えられるゲームを**探索困難（hard exploration）ゲーム**と呼ぶ．また,「報酬が得られる局面が多いか」という観点で探索困難ゲームをさらに二種類に分類している．探索困難ゲームの中で，報酬が得られる局面が多いゲームを**密な報酬環境（dense reward）ゲーム**，報酬が得られる局面が少ないゲームを**疎な報酬環境（sparse reward）ゲーム**と呼んでいる．

探索容易ゲームでは，エージェントに ϵ-greedy といった簡単な探索戦略を適用するだけで，高いスコアを比較的容易に獲得できる．一方，探索困難ゲームでは，ϵ-greedy といったランダム性に基づく方法では不十分であり，何かしらの高度な探索戦略が必要と考えられる．ただし，探索困難ゲームでも密な報酬環境ゲームに関しては，最近の深層強化学習技術の発展に伴い，平均的な人間レベルに到達している [32, 38, 56, 58]．しかしながら，疎な報酬環境ゲームにおいては，最新の深層強化

[1] 合法手／非合法手程度の最低限の情報は与えられると仮定

学習の成果を適用しても平均的な人間レベルには至っていない．その理由として考えられることとして，例えば強化学習を行う際に，報酬が得られる局面を見つけられなかった場合，強化学習の方法がどんなに優秀だとしても学習を行うことが出来ないのは明らかである．つまり，疎な報酬環境ゲームでは，何かしらの方法を用いて報酬が得られる局面を見つけ出さなくてはいけない．しかし，ゲーム画面とスコアのみからエージェントを構築する設定では，Atari 2600 のゲームは列挙困難なほど大きい局面空間を持つため，その探索は容易ではない．

そこで本論文では，巨大な局面空間をもつ疎な報酬環境ゲームでも有効に機能する方法論について議論し，こういった状況に適したを考案する．まず提案手法は，ベースの深層強化学習法として，汎用的かつ学習性能の高い Asynchronous Advantage Actor-critic (A3C) [29] アルゴリズムを採用する．そして，ベースとなる A3C に対して疎な報酬環境ゲームに適した改良をおこなうことで提案手法を構築する．また，提案手法の効果を検証するため，A3C が人間と比較し苦戦している疎な報酬環境ゲームに着目し，それらを用いて実証実験を行い，提案手法が A3C と比較して疎な報酬環境ゲームのスコアは大きく上昇可能であること，さらに一部のゲームにおいて最高スコアを達成可能であることを示す．

2.2 問題設定

強化学習の枠組みでは，エージェントが行動し，その行動に対する報酬に基づいてエージェントの学習が行われる．報酬が疎とは，ある局面 (状態) に対する行動に与えられる報酬が大半の場合でゼロ[2]であることをさす．$\mathcal{S}$ を，すべての可能な局面の集合とする[3]．また，$\mathcal{A}$ を全ての可能な行動の集合とする．次に，t を時刻を表す変数とし，0 および正の整数を取ることとする．この時，$s_t \in \mathcal{S}$ を時刻 t の局面，$r_t \in \mathbb{R}$ を時刻 t の報酬，$a_t \in \mathcal{A}$ を時刻 t の行動を表す記号とする．

対象とする環境がマルコフ決定過程に従うと仮定した場合，報酬 r_{t+1} は報酬確率 $p(r_{t+1} = r \mid s_t, a_t, s_{t+1})$ に基づいて与えられる．ここで，全ての可能な (s_t, a_t, s_{t+1}) の組み合わせの集合を $\mathcal{T}$ とおく．この時，本論文では,「報酬が疎な環境」という用語を以下のように定義する．

定義 1 (報酬が疎な環境) ある特定の (s_t, a_t, s_{t+1}) の組み合わせの集合を $\mathcal{T}' \subset \mathcal{T}$ とする．この時 $|\mathcal{T}'| \ll |\mathcal{T}|$ と仮定する．すなわち，特定の組み合わせ $\mathcal{T}'$ は，全体に対して要素数が非常に少ない集合である．この時，全ての $(s_t, a_t, s_{t+1}) \in \mathcal{T} \backslash \mathcal{T}'$ に対して，$r_{t+1} = 0$ となる確率が 1 となる環境，すなわち

$$p(r_{t+1} = 0 \mid s_t, a_t, s_{t+1}) = 1 \quad \forall (s_t, a_t, s_{t+1}) \in \mathcal{T} \backslash \mathcal{T}' \tag{2.1}$$

を，報酬が疎な環境と呼ぶ．

この環境の意味するところは，どのような局面から次の局面へ遷移する行動をとっても，報酬を得られる確率が限りなく低いことを表している．このような環境の場合，報酬が得られる行動の系列が得られることが少ないため，エージェントの学習を進めることが困難であることを意味している．

表 2.1 に本論文で用いる記号の一覧を示す．

[2]報酬は一般的に正の報酬のみを用いることが多いが，負の報酬がある場合も含むこととする．
[3]ここでの「局面」とは,「状態」とも呼ばれるが，一般名詞としての「状態」との混同を避けるために本論文では局面という用語を用いる．

表 2.1. 本論文で用いられる記号

t	時刻．$t \in \{1, 2, \dots\}$
$\mathcal{S}$	全ての可能な局面の集合
$\mathcal{A}$	全ての可能な行動の集合
a または a_t	行動，または，時刻 t での行動．$a_t \in \mathcal{A}$
s または s_t	局面，または，時刻 t の局面．$s_t \in \mathcal{S}$
r または r_t	報酬，または，時刻 t での報酬，探索報酬との区別のために「真の報酬」という記載も用いる
e または e_t	探索報酬，または，時刻 t での探索報酬
$p(r_{t+1} \mid s_t, a_t, s_{t+1})$	局面 s_t から s_{t+1} へ行動 a_t により遷移した際の $t+1$ での報酬 r_{t+1} の確率
$\mathcal{T}$	(s_t, a_t, s_{t+1}) の組み合わせの集合
π または $\pi(a_t \mid s_t; \theta'_\pi)$	戦略，または，パラメタ θ'_π のときに局面 s_t で行動 a_t を選択する確率
v または $v(s_t; \theta'_v)$	価値関数，または，パラメタ θ'_v のときに局面 s_t の価値を表す値
θ，または，θ_π，θ_v	パラメタ，または，戦略 π に関連するパラメタ，および，価値関数 v に関連するパラメタ
θ'，または，θ'_π，θ'_v	パラメタ，または，戦略 π に関連するパラメタ，および，価値関数 v に関連するパラメタ（θ と比較した場合は，各スレッド毎のローカルなパラメタを表す）．
$H(\theta'_\pi)$	エントロピー項
$\mathcal{Z}$	任意の有限の整数空間．$\mathcal{Z} = \{1, \dots, N\}$
N	局面のハッシュ値の上限
$\phi(s_t)$	ハッシュ関数 $\phi : \mathcal{S} \to \mathcal{Z}$ 例：$\phi(s_t) = z$，ただし $s_t \in \mathcal{S}$ および $z \in \mathcal{Z}$．
z_t	時刻 t の局面 s_t からハッシュ関数 $\phi : \mathcal{S} \to \mathcal{Z}$ により変換したハッシュ値
$Z_{1:t}$	時刻 1 から時刻 t の局面 s_t からハッシュ関数 $\phi : \mathcal{S} \to \mathcal{Z}$ により変換したハッシュ値
$\psi(Z_{1:t}, z_t)$	整数のリスト $Z_{1:t}$ に対して整数 z_t の出現回数を返す関数．
$\tilde{\psi}(s_t)$	現在の局面 s_t の擬似出現回数を計算する関数
β	項の重み係数，β^{a3c}，β^{hash}，β^{psc}，β^{ucb} など
γ	時刻ごとの報酬の割引率

2.3 関連研究

2.3.1 深層強化学習

Mnih ら [31] はディープニューラルネットワークを利用して，Q 関数を学習する DQN と呼ばれる手法を提案した．この手法により人間が観測するゲームの画面情報と現在のスコアのみから Q 関数を効果的に直接学習することに成功した．

Mnih ら [29] は，さらに Actor-Critic 法に基づく A3C アルゴリズムを提案した．A3C は，Atari 2600 を用いた実験において高い性能を示した代表的な深層強化学習法であり，DQN よりも探索困難ゲームを除けば優れている．また，A3C は，DQN のように Q 関数を学習するのではなく，行動戦略と価値関数を分離して同時に学習を行う方式を採用している．このような学習方式は行動空間の高度な表現力を持つ一方，表現力の向上による過学習になりやすい傾向があり，学習時の探索が局所的になる可能性が高い．この問題を解決するため，A3C では目的関数にエントロピー正則化項を加え過学習を軽減する仕組みを導入している．より具体的には，エントロピー正則化項は，全ての行動を同じ確率で行うとき，すなわち，エージェントがランダムに行動する際に最も値が高くなる．よって，エントロピー正則化項を追加することで，エージェントは報酬が得られた行動のみを取るように学習が進むといった過学習を軽減することができる．

Algorithm 2.1 に A3C の学習手順の概要を示す．R を特定の周期で観測された報酬の加重和とする．

Algorithm 2.1 各 actor-learner スレッドごとの A3C の学習アルゴリズム

// T: 共有するグローバル変数 (全体で $T = 0$ で初期化)
// θ_π, θ_v: 共有するグローバル変数, θ'_π, θ'_v: スレッド固有の変数
// t_{interval}: 更新間隔パラメータ, $T_{\max}$: 最大繰り返し回数
初期化 $t \leftarrow 1$
repeat
  $d\theta_\pi \leftarrow 0, d\theta_v \leftarrow 0, \theta'_v \leftarrow \theta_v, \theta'_\pi \leftarrow \theta_\pi$　　//毎回初期化
  $t_s \leftarrow t$
  repeat
    戦略 $\pi(a_t|s_t;\theta'_\pi)$ から選択された行動 a_t
    得られる報酬 r_t と次局面 s_{t+1}
    $t \leftarrow t + 1$
    $T \leftarrow T + 1$　　//全てのスレッドについて非同期に更新
  until $s_t =$ 終端局面 **Or** $t - t_s = t_{\mathrm{interval}}$
  if $s_t =$ 終端局面 **then**
    $R \leftarrow 0$
  else
    $R \leftarrow v(s_t;\theta'_v)$
  end if
  foreach $i \in \{t-1, \ldots, t_s\}$ **do**
    $R \leftarrow \gamma R + r_i$
    $d\theta_\pi \leftarrow d\theta_\pi + d\theta'_\pi$ 式 (2.2) より θ'_π の累積勾配 $d\theta_\pi$
    $d\theta_v \leftarrow d\theta_v + d\theta'_v$ 式 (2.3) より θ'_v の累積勾配 $d\theta_v$
  end for
  $d\theta_\pi$ と $d\theta_v$ を用いた θ_π と θ_v の非同期更新
until $T \geq T_{\max}$

π と v はそれぞれ、戦略と価値関数の出力と表す．θ_π と θ_v はそれぞれ π と v のパラメタである．同様に θ'_π と θ'_v はそれぞれ π と v のパラメタであり，各スレッド毎に管理されるスレッド毎に独立なパラメタとする．さらに $H(\cdot)$ はエントロピー項，β^{a3c} はエントロピー項の強さを決定する係数である．この時，A3C は各スレッドの独立なパラメタに対して勾配 $d\theta'_\pi$ と $d\theta'_v$ を次の計算によって求める．

$$d\theta'_\pi = \nabla_{\theta'_\pi} \log(\pi(a_i|s_i;\theta'_\pi))(R - v(s_i;\theta'_v)) + \beta^{\mathrm{a3c}} \nabla_{\theta'_\pi} H(\theta'_\pi), \tag{2.2}$$

$$d\theta'_v = \frac{\partial (R - v(s_i;\theta'_v))^2}{\partial \theta'_v} \tag{2.3}$$

これらの値を用いて最終的にはグローバルなパラメタ θ_π と θ_v を更新する．

2.3.2 強化学習における探索と探索報酬

報酬が疎な環境に対応する方法の一つとして MBIE-EB アルゴリズム [49] が提案されている．MBIE-EB アルゴリズムは，局面の訪問回数を記録し，その局面の訪問回数に基づく探索報酬を報酬の一部として利用することで，未知の局面へより遷移しやすくなるように学習を進めて報酬が得られる局面を探索する方法となっている．この方法は，局面空間が列挙可能な程度に小さいマルコフ決定過程の設定では有効であると実証されている．しかし，近年盛んに取り組まれている深層強化学習の枠組みでは，ゲーム画面のピクセル情報をそのまま入力情報として用いている．これは，全ての可能なゲーム画面の集合により局面空間 $\mathcal{S}$ が定義されることを意味する．例えば，文献 [31] では，各ゲーム画面は 84×84 ピクセルの 256 階調で構成され，さらに 4 つの連続する時刻の画面を一つの

入力として利用している．つまり，単純計算で最大 $256^{84\times84\times4}$ 種類の局面が存在する環境と解釈できる．このように膨大な局面数となる環境下では，同一の局面に到達する確率は非常に小さい．このため，単純に各局面の出現回数を手がかりとして探索する MBIE-EB アルゴリズムのような方法は機能しないことが容易に推測できる．

この状況に対応するため，複数の局面の抽象化手法が深層強化学習の分野において提案されている．例えば，Tang ら [52] は，ハッシュ関数による局面の抽象化手法を用いて，以下のような探索報酬 e^{hash} を提案している．

$$e_t^{\mathrm{hash}} = \frac{\beta^{\mathrm{hash}}}{\sqrt{\psi(Z_{1:t}, z_t)}}, \tag{2.4}$$

ここで $\beta^{\mathrm{hash}} \in \mathbb{R}_{\geq 0}$ は探索報酬の係数である．局面の集合 $\mathcal{S}$ から有限の整数空間 $\mathcal{Z}$, つまり $\mathcal{Z} = \{1, \ldots, N\}$ に写像するハッシュ関数を $\phi : \mathcal{S} \to \mathcal{Z}$ で定義する．例えば $z_t \in \mathcal{Z}$ を，ハッシュ関数 ϕ によって時刻 t の局面 $s_t \in \mathcal{S}$ から ϕ によって変換された整数とする．次に，$Z_{1:t}$ を，z_1 から時刻 z_t までを並べた整数のリストとする．つまり，$Z_{1:t} = (z_1, \ldots, z_t)$ である．例えば，$Z_{1:6} = (1, 2, 1, 3, 3, 1)$ および $z_t = 1$ の時，$Z_{1:t}$ 中に 1 が 3 回出現するので $\psi(Z_{1:t}, z_t) = 3$ となる．$\psi(Z_{1:t}, z_t)$ は Z_t の長さに対して，単調非減少関数なので，時刻 t が大きくなれば e^{hash} の値は徐々に小さくなっていく．また，z_t と同じ整数が出現すればするほど値が小さくなる関数なので，式 2.4 は，同じ局面を経験するたびに探索報酬が減少する関数と解釈することができる．

別の局面の抽象化手法として，Bellemare ら [5] は，局面の疑似出現回数を用いた探索手法を提案した．探索報酬 e^{psc} は次の式で定義される．

$$e_t^{\mathrm{psc}} = \frac{\beta^{\mathrm{psc}}}{\sqrt{\tilde{\psi}(s_t) + 0.01}}, \tag{2.5}$$

この時，β^{psc} は探索報酬の係数である．局面 s_t の疑似出現回数 $\tilde{\psi}(s_t)$ は，現在の局面 s_t に対して，局面の疑似回数の出現確率 $\rho(s_t)$ と，次に同じ局面に遭遇したと仮定した際の出現確率 $\rho'(s_t)$ を用いて以下の式で計算する．

$$\tilde{\psi}(s_t) = \frac{\rho(s_t)(1 - \rho'(s_t))}{\rho'(s_t) - \rho(s_t)}, \tag{2.6}$$

この手法は特に “Montezuma’s revenge” という Atari 2600 の中でも深層強化学習アルゴリズムにとって最も難しいゲームの一つの成績を大きく向上した．

Tang らや Bellemare らの手法は局面の未知さを報酬とするものであるが，Intrinsic Curiosity Module (ICM) [40] では，局面遷移の予測の難しさを「好奇心」とすることで報酬の自動生成を行う．この報酬の生成には逆モデルと順モデルの二つのモデルが関係している．逆モデルでは観測された局面 s_t, s_{t+1} からエージェントの行動 a_t の予測を行う．順モデルでは局面 s_t と行動 a_t から次の局面 s_{t+1} の予測を行う．次にそれぞれのモデルの予測と実際の行動や局面から誤差を計算する．その誤差が大きいということはエージェントが理解していない行動や局面であり，好奇心の大きさを表現している．すなわちその誤差を報酬とすることでエージェントは好奇心に基づいた未知の局面を探索する行動をとることができる．この手法により VizDoom と Super Mario Bros. において外部の報酬を用いることなく，効率的な探索を行うことに成功した．

2.3.3 探索における選択基準

Lai と Robbins [25] は，Multi Armed Bandit（MAB）問題に対して Upper Confidence Bounds (UCB) スコアが最大となるよう行動を選択するというアルゴリズムを提案した．このアルゴリズムは異なる行動同士の関係を考慮するために全体の試行回数を用いて行動を決定する．UCB スコア r^{ucb} は以下の式で定義される．

$$r_t^{\mathrm{ucb}} = r(a_t) + \sqrt{\frac{2\log(t)}{\psi(A_{1:t}, a_t)}}, \tag{2.7}$$

a_t は時間 t で選択された行動，$A_{1:t}$ は時刻 1 から t までの行動の履歴，$r(a_t)$ は行動 a_t をとった際の平均報酬，$\psi(A_{1:t}, a_t)$ は行動 a_t がこれまでに選択された回数を表すとする．UCB スコアの第二項は各行動を選択する情報の価値を表している．UCB アルゴリズムは無限回の試行で最良の行動を選択できると保証されている．

2.3.4 報酬が疎な環境における学習

本論文が対象としているビデオゲームの自動操作エージェントを学習するタスク以外でも，例えばロボットアームの研究分野においても同様に報酬が疎な環境を対象にした研究が行われている．ロボットアームの課題として箱を開け，中にブロックを配置する課題などは，報酬を得るための適切な行動は複雑であり，ランダムな行動で報酬を得ることは非常に困難である．そのためロボットアームの分野においても報酬が疎な環境において強化学習を行うことは挑戦的な課題である．

Andrychowics ら [2] は Hindsight Experience Replay と呼ばれる疑似ゴールを設定することで効率的な学習に成功した．この疑似ゴールは報酬の得られなかったエピソードから生成される．そしてこの疑似ゴールに対する行動と局面をセットにして価値関数を学習する．

Riedmiller ら [47] は意図戦略を定義することで本来のゴールを学習する Scheduled Auxiliary Control を提案した．意図戦略はゴールからの距離に応じて与えられる報酬をもとに学習が行われる．またスケジューラが個別の意図戦略を選択し実行することでより効率的な学習が行われる．

これらの論文と本論文では問題設定が異なる．ロボットアームの分野では例えばゴールからの距離といった単純で直感的な疑似報酬を設計することができる．一方，Atari2600 のゲームでは，どのような状態のときに報酬が与えられると考えていいのかは自明ではない．そのため，提案手法では「未知の局面」に対して報酬を設計しているという点が大きく異なる．

2.4 報酬が疎な環境に適した深層強化学習法

ここでは提案手法について説明する．図 2.1 に，提案手法における各モジュールとモジュール間の関係を示す．

提案手法は，前節で説明した A3C をベースの深層強化学習法として採用する．その上で，ベースの A3C から報酬が疎な環境でも学習を効率的に行うために以下の 3 点に関して改良を行う．

- UCB に基づく探索報酬を計算するモジュールの追加（2.4.1 節）

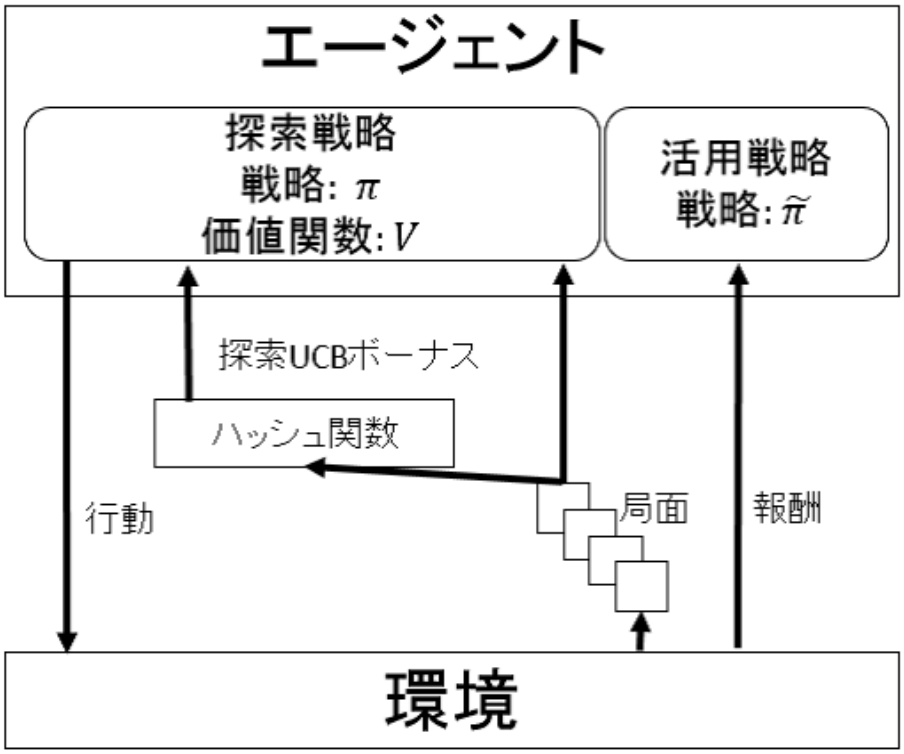

図 2.1. 提案手法における各モジュールの関係

- 探索用の戦略（探索戦略）と実行用の戦略（活用戦略）という二つの異なる目的で使用される戦略の組み込み（2.4.2 節）
- 報酬が得られた行動の再利用（2.4.3 節）

2.4.1　UCB に基づく探索報酬

式 2.4 や式 2.5 に示した従来法の探索報酬は，同一の局面に遭遇する度に値が単調に減少する関数である．つまり，未知の局面をより優先的に選択する戦略と言える．しかし，報酬が疎な環境では，未知の局面を優先する戦略，逆にいえば，既知の局面をなるべく選択しないようにする戦略が良いとは必ずしも言えない．これは，報酬が疎な環境では，エージェントが報酬を得られる局面に仮に近づいていたとしても，報酬が得られる局面まで必ず到達できる保証がないため，同じ局面をある程度の試行を繰り返す必要があるからである．この仮説に基づき，我々は式 2.7 で紹介した UCB に基づく探索報酬を提案する．

時刻 t における探索報酬を e_t とする．この時，e_t を以下のように定義する．

$$e_t = \beta^{\mathrm{ucb}} \sqrt{\frac{\log(t)}{\psi(Z_{1:t}, z_t)}}, \tag{2.8}$$

ただし，β^{ucb} は e_t の重みを調整する係数とする．

e_t は，式 2.7 に示した UCB の計算式の右辺第二項を元に定義されている．しかし，提案手法の状況に合うように修正を行なっている．具体的には，本来の UCB は行動回数 $\psi(A_{1:t}, a_t)$ に基づいて値を計算するが，e_t は，その代わりに局面の訪問回数 $\psi(Z_{1:t}, z_t)$ に基づいて値を計算する点が異なる．

e_t の直観的な解釈としては以下のようになる．まず基本的に，ある時刻 t を固定した場合，ある局面 s_t に対するハッシュ関数 $\phi(s_t)$ の値 z_t の出現回数が小さければ e_t は大きい値になり，反対に，z_t の出現回数が大きければ e_t の値は小さくなる．つまり，頻度に基づく方法と同様に，基本的には未知の局面を優先的に探索する方法となっている．一方で，z を固定して考えた場合，時刻 t が進むに連れ e_t の値は対数スケールで徐々に大きくなる．つまり，長期間 z の局面を訪問しなかった場合，改めて z の局面を選択するようになる可能性が高まることを意味する．このように，式 2.8 に基づく探索報酬は，訪問回数に基づいて未知局面を優先して選択するだけでなく，時間変化による探索の進み具合も考慮した計算式になっていると言える．これ以降，本論文で探索報酬と記述する際には式 2.8 で定義された UCB に基づく探索報酬のことを指すこととする．

2.4.2 二種類の戦略の学習

この章では，提案手法の全体的な学習手順について説明する．Algorithm 2.2 は提案手法によるエージェントの学習方法を示しており，基本的にはベースラインの A3C と同じ手法である．最初にエージェントは，現在の戦略に基づいて行動を選択し，次の局面，報酬，および，探索報酬を受け取る．提案手法が，A3C や他の同様の探索方法を用いる手法と大きく異なる点は，真の報酬と探索報酬を区別して二つの戦略を独立に学習することである．

以下，二つの戦略を用いる理由を述べる．一般的に，これまでの関連研究では探索報酬 e は真の報酬 r に追加して使用される．すなわち，$R = r + e$ とおき，R はエージェントのネットワークを更新するために r の代わりに用いられる．これらの方法では，戦略と価値関数の適切な訓練という観点で問題を引き起こす可能性がある．これは，R は r と e の和で表されるため，エージェントはどちらの報酬に基づいて現在の学習が行われているか明確にわかっていない状態で学習が行われることを意味する．つまり，報酬が疎な環境では，エージェントは探索報酬に従って探索を行うのみの学習となっている可能性が高く，実際に報酬が得られた際の価値を適切に判断していない可能性がある．また，学習時の探索報酬 e は，時刻 t が無限になるまで学習を行った場合，理論的には 0 に近似するので，最終的に探索報酬は真の報酬の学習には悪影響を与えない．しかし，学習時の更新回数は有限であるため，現実的な状況では探索報酬 e_t は 0 にはならない．更に，真の報酬 r はほぼ 0 であるため，学習される戦略は，結局探索報酬の影響を強く反映した戦略となる可能性が高い．

上記の推測に基づいて，報酬と探索報酬用の 2 つの異なる戦略を導入する．一つ目の戦略は**探索戦略** π と呼び，トレーニング時に探索報酬を用いて学習する．二つ目の戦略は**活用戦略** $\tilde{\pi}$ と呼び，環

Algorithm 2.2 各 actor-learner スレッドごとの提案手法 (A3C+UCB) の学習アルゴリズム

// T: 共有するグローバル変数 (全体で $T = 0$ で初期化)
// $\theta_{\tilde{\pi}}, \theta_\pi, \theta_v$: 共有するグローバル変数
// $\theta'_{\tilde{\pi}}, \theta'_\pi, \theta'_v$: スレッド固有の変数
// t_{interval}: 更新間隔パラメータ, $T_{\max}$: 最大繰り返し回数
初期化: $t \leftarrow 1$
repeat
  $d\theta_{\tilde{\pi}} \leftarrow 0, d\theta_\pi \leftarrow 0, d\theta_v \leftarrow 0, \theta'_{\tilde{\pi}} \leftarrow \theta_{\tilde{\pi}}, \theta'_\pi \leftarrow \theta_\pi, \theta'_v \leftarrow \theta_v$
  $t_s \leftarrow t$
  repeat
    $\tilde{\pi}(a_t|s_t;\theta'_{\tilde{\pi}})$ に基づく行動 a_t.
    観測された r_t と次局面 s_{t+1}
    $t \leftarrow t + 1$
    $T \leftarrow T + 1$　//全てのスレッドについて非同期に更新
  until s_t 終端局面 **Or** $t - t_s = t_{\mathrm{interval}}$
  if $s_t =$ 終端局面 **then**
    $R \leftarrow 0$
  else
    $R \leftarrow v(s_t;\theta'_v)$
  end if
  foreach $i \in \{t-1, \ldots, t_s\}$ **do**
    $R \leftarrow \gamma R + e_i$
    $R' \leftarrow \gamma R' + r_i$
    $\tilde{R} \leftarrow clip(R', -1, 1)$ //R' を $[-1, 1]$ の範囲でクリップ
    $d\theta_{\tilde{\pi}} \leftarrow d\theta_{\tilde{\pi}} + d\theta'_{\tilde{\pi}}$ 式 (2.9) より $\theta'_{\tilde{\pi}}$ の累積勾配 $d\theta_{\tilde{\pi}}$
    $d\theta_\pi \leftarrow d\theta_\pi + d\theta'_\pi$ 式 (2.2) より θ'_π の累積勾配 $d\theta_\pi$
    $d\theta_v \leftarrow d\theta_v + d\theta'_v$ 式 (2.3) より θ'_v の累積勾配 $d\theta_v$
  end for
  $d\theta_{\tilde{\pi}}$ と $d\theta_\pi$ と $d\theta_v$ を用いた $\theta_{\tilde{\pi}}$ と θ_π と θ_v の非同期更新
until $T \geq T_{\max}$

境からの真の報酬のみで学習し，主にテスト時に使用する．これらの 2 つの戦略は別々の報酬で学習を行うが，報酬や探索報酬に関する情報を共有させるため単一の畳み込みニューラルネットワークを使用する．

$\theta'_{\tilde{\pi}}$ を $\tilde{\pi}$ を計算する際に利用されるパラメタの集合とする．式 2.2 および式 2.3 と同様に，$d\theta'_{\tilde{\pi}}$ は以下の式を用いて計算する．

$$d\theta'_{\tilde{\pi}} = \nabla_{\theta'_{\tilde{\pi}}} \log(\tilde{\pi}(a_i|s_i;\theta'_{\tilde{\pi}}))\tilde{R} \tag{2.9}$$

2.4.3 経験再生

これまで述べてきたように，報酬が疎な環境では，真の報酬が得られる局面に到達する可能性が非常に小さいという問題がある．この問題から，エージェントの学習途中で運よく報酬が得られる局面に到達したとしても，報酬が得られた局面を再現することが難しいということが容易に想像できる．これは，エージェントは報酬を一度受け取っても，パラメータの変動幅は小さいため，同じ行動を再現できるようになるまでには繰り返しの学習が必要となる．この課題に対応するため以前の研究では経験再生と呼ばれる手法が用いられている [31]．本研究ではエージェントが真の報酬を得た際の局面と行動を履歴として記録し，それを再利用する枠組みを導入する．

この履歴をトレーニング時に ϵ の確率で用いる．これによりエージェントに何度も同じ行動を強制的にとらせることで学習を効果的に行う．本研究では ϵ の値を 0.1 とした．

表 2.2. トレーニングとテストに用いられる構成の概要

学習時	トレーニングステップ	2 億ステップ (8 億フレーム)
	スレッド数	56
	最適化手法	RMSProp, 減衰率 0.99
	割引率 γ	0.99
	エントロピー項の係数 β^{a3c}	0.01
評価時	テスト間隔	100 万ステップごと
	テスト回数	それぞれ 30 回

2.5 実験

本論文では，ALE [4] に実装されている Atari 2600 のゲームを用いて実験を行った．

2.5.1 疎な報酬環境ゲーム

文献 [5] で疎な報酬環境ゲームに分類されているゲームを対象に実証実験を行う．具体的には，“Freeway”，“Gravitar”，“Montezuma revenge (Montezuma)”，“Private Eye (Private)”，“Solaris”，“Venture” の 6 種類のゲームである．また，文献 [5] では報酬が疎なゲームに分類されていないが，本研究には一章で述べたように A3C の改良という観点もあるため，ベースラインの A3C のスコアが 0 であった “Enduro” も実験に追加し，報酬が疎なゲームと同様に，探索によって報酬が容易に得られるようになるか効果を検証する．[4] よって，合計 7 種類のゲームを用いて実験を行った．

2.5.2 ベースライン手法の設定

本論文では，文献 [29] 中の A3C の実験で使用された実験設定に従って実験を行った[5]．例えば，ネットワークアーキテクチャは既存研究 [30] と同一である．具体的には，二層の畳み込みニューラルネットワーク (スライド 4 で 8×8 の 16 枚のフィルタとスライド 2 で 4×4 の 32 枚のフィルタ) と 256 の隠れユニットを持つ全結合層の 3 層からなる構成である．隠れユニットの活性化関数として，ReLU [33] を使用した．また，最終出力層は，価値関数と各行動ごとの確率を出力するソフトマックス層の二種類で構成される．

表 2.2 に，学習および評価時の設定の概要を示す．まず，既存の研究と一致させるために，エージェントの学習ステップ数を 2 億ステップとした．次にエージェントの評価時には，“no-op performance measure” [56] という方法でさまざまな初期ランダム条件で 30 回評価を行った．最終的な評価は，30 回の平均スコアを最終スコアとして用いた．

[4] Enduro は，いわゆる「車のレースゲーム」であり，車を抜かすたびにスコアが加算されるので，その観点では疎な報酬環境ゲームではない．しかし，基本操作として,「車のアクセル」に相当する操作を選び続けるという行為が学習できるまでは，報酬を一度も得られないまま試行が終了することがほとんどとなる．よって，学習がうまくいっていない段階では，報酬が疎なゲームで解消したい状態と同じとみなせる．実際，ベースラインの A3C では，2 億ステップの学習が終わった状況でも「車のアクセル」に相当する操作を選択し続ける．ということを学習できていないため，一度も報酬が得られず，スコアが 0 点のままとなる．

[5] スレッド数に関しては，元文献 [29] では 16 を用いているが，学習時間を短縮するため 56 を用いた．スレッド数を増加させてもトレーニングのステップは固定であるため，結果は大きく変わらないと考えている．

Algorithm 2.3 局面 s からハッシュキー z を計算するアルゴリズム

Input: 局面 s
// k: ハッシュの粒度を表す変数
// $\boldsymbol{A} \in [-1, 1]^{k \times D}$: ランダム写像行列
// $g(\cdot)$: 局面 s を D 次元ベクトルに写像する関数
// p_r: インデックスの最大値を決定する素数
$z \leftarrow 0$ // z: ハッシュキーを計算するための変数
$\tilde{\boldsymbol{b}} \leftarrow g(s)$ // $\tilde{\boldsymbol{b}}$: D 次元ベクトル
$\boldsymbol{b} \leftarrow \boldsymbol{A}\tilde{\boldsymbol{b}}$ // $\boldsymbol{b}$: k 次元ベクトル
for $i = 1$ to k **do**
    $z \leftarrow z \times 2$
    if $b_i > 0$ **then** // $\boldsymbol{b} = (b_1, \ldots, b_k)$
        $z \leftarrow z + 1$
    end if
end for
$z \leftarrow z \% p_r$
Output: z

2.5.3 ハッシュ値の計算方法

ここで，提案手法における局面 s からハッシュ値 z を計算する具体的な方法を述べる．実際に用いた処理手順を Algorithm 2.3 に示す．Algorithm 2.3 は，基本的に Locality-Sensitive Hashing (LSH) [1] と呼ばれる計算法であり，本質的には Tang ら [52] の研究で用いられている計算方法と同じである．

まず初期化処理として，各要素を $[-1, 1]$ の乱数で初期化した $k \times D$ 行列 $\boldsymbol{A}$ を用意する．次に前処理として，局面 s を関数 $g(\cdot)$ を用いて D 次元ベクトル $\tilde{\boldsymbol{b}}$ に変換する．前述の通り文献 [31] に従うと，各ゲーム画面は 84×84 ピクセルで構成され，さらに 4 つの連続する時刻の画面を一つの入力として利用する．提案手法では，関数 $g(\cdot)$ を，4 つの連続する画面の最後の画面と一番目と四番目の画面の差分という二つ分の画面を考え，それを一次元のベクトルに連結して返す関数と定義する．つまり，$D = 14,112\,(= 84 \times 84 \times 2)$ である．次にベクトル $\tilde{\boldsymbol{b}}$ を，変換行列 A によって k 次元ベクトル $\boldsymbol{b}$ に写像する．最後に，$\boldsymbol{b}$ の各要素に対して，正の値の場合は 1，それ以外の場合は 0 として，$\boldsymbol{b}$ からハッシュキー（バイナリコード）z を得る．

LSH において k の値は粒度を決定する．値が大きいほど衝突が少なくなるため，局面を一意に区別することができる．理想的には現在とその後の局面のハッシュキーは常に異なることが望ましい．そのためには k を大きな値にする必要がある．しかしながら，メモリ要求は k の値に比例して増加する．そこで必要なメモリ所要量を減らすため，インデックスの最大数を表す素数 p_r を使用する．ここでは Tang ら [52] の研究を参考にして $p_r = 999983$，$k = 128$ とした．

2.6 結果および考察

2.6.1 結果

表 2.3 に実験結果を示す．議論の都合上，実験結果を 4 つのカテゴリー (a), (b), (c), (d) に分類する．カテゴリー (a) は本論文で実際に実装して実験を行った結果である．このカテゴリーの主な目的は，ベースライン手法（A3C），ベースラインに既存の探索法を加えたもの（A3C + psc），および提案手法（A3C + UCB）を公平な条件で比較することである．カテゴリー (b) は，カテゴリー (a) にお

表 2.3. 提案手法の結果，ベースラインおよび他の強化学習アルゴリズムとの比較．太字の数字は各ゲームの最良の結果を示す．

Cat.	method	Enduro	Freeway	Gravitar	Montezuma	Private	Solaris	Venture
(a)	A3C (ベースライン：再実験)	0.0	0.0	283.3	3.3	160.1	3287.3	0.0
	A3C+psc (再実験)	181.7	22.1	311.7	0.0	1855.7	2612.0	0.0
	A3C+UCB (提案手法)	28.5	23.0	406.7	126.7	**7643.5**	**4622.7**	163.3
(b)	A3C (ベースライン：文献 [5] 報告値)	0.0	0.0	201.3	0.2	97.4	2102.1	0.0
	A3C+psc [5]	694.8	30.4	238.7	273.7	99.3	2270.2	0.0
(c)	DDQN+psc (文献 [52] 報告値)	–	29.2	–	**3439**	–	–	369
	TRPO+picel-SimHash [52]	–	31.6	468	0	–	2897	263
	TRPO+BASS-SimHash [52]	–	28.4	604	238	–	1201	616
	TRPO+AE-SimHash [52]	–	33.5	482	75	–	4467	445
(d)	DQN [31]	301.8	30.3	306.7	0.0	1788.0	–	380.0
	DDQN [56]	319.5	31.8	170.5	0.0	670.1	–	93.0
	Gorila [32]	114.9	11.7	**1054.5**	4.2	748.6	–	**1245.3**
	Bootstrapped DQN [38]	1591.0	**33.9**	286.1	100.0	1812.5	–	212.5
	Dueling network [58]	**2258.2**	0.0	588.0	0.0	–	2250.8	497.0

表 2.4. 提案手法とベースラインの各ゲームにおける結果の四分位点

	method	第 1 四分位点	第 2 四分位点	第 3 四分位点
Enduro	A3C	0.0	0.0	0.0
	A3C+psc	142.0	165.5	186.5
	A3C+UCB	22.5	29.5	35.5
Freeway	A3C	0.0	0.0	0.0
	A3C+psc	21.0	22.0	23.0
	A3C+UCB	22.0	23.0	24.0
Graviter	A3C	0.0	225.0	425.0
	A3C+psc	100.0	300.0	500.0
	A3C+UCB	250.0	500.0	550.0
Montezuma	A3C	0.0	0.0	0.0
	A3C+psc	0.0	0.0	0.0
	A3C+UCB	0.0	0.0	400.0
Private	A3C	-661.0	0.0	0.0
	A3C+psc	-999.0	3983.0	4000.5
	A3C+UCB	0.0	9081.0	14600.5
Solaris	A3C	1400.0	1920.0	3330.0
	A3C+psc	1240.0	1840.0	3100.0
	A3C+UCB	3100.0	5000.0	6400.0
Venture	A3C	0.0	0.0	0.0
	A3C+psc	0.0	0.0	0.0
	A3C+UCB	0.0	100.0	250.0

いて実装して実験した方法論の元文献で報告されているスコアを示している [5]．カテゴリー (c) は，最近提案された psc と SimHash ベースの探索手法を取り入れた文献の結果を示す[6]．最後にカテゴリー (d) は，現在のトップスコアを獲得した方法論の結果である．これらは，DQN [31], double DQN (DDQN) [56], Gorila [32], Bootstrapped DQN [38], Dueling network [58] のもっとも高いスコアを掲示している．これらの手法は基本的には特別な探索戦略は実装されていない．

表 2.3 の結果は，従来研究に合わせてスコアの平均値で評価を行なっている．しかしながら平均値だけでは手法に有意な差があるのかを判断することは困難である．そこで，本論文で再実験を行なったカテゴリー (a) に関して，詳細な実験結果として四分位点を表 2.4 に示す．

次に，「A3C+UCB（提案法）と A3C（ベースライン: 再実験）」および「A3C+UCB（提案法）と A3C+psc（再実験）」の間で，それぞれ独立に「2 つの手法間で差がない」という帰無仮説のもと検定を行なった．検定には，マン・ホイットニーの U 検定を用いた．その結果を表 2.5 に示す．

表 2.3，表 2.4，表 2.5 のから次のことがいえる．

[6]DDQN+psc の詳細はスコアは元論文 [5] では報告されていないので，文献 [52] の報告値を用いた．

表 2.5. マン・ホイットニーの U 検定を用いた有意差検定の結果：∗ は A3C+UCB と A3C の結果に対して有意水準 1%で有意な差が認められたゲーム．† は A3C+UCB と A3C+psc の結果に対して有意水準 1%でスコアに有意な差が認められたゲーム．

method	Enduro	Freeway	Gravitar	Montezuma	Private	Solaris	Venture
A3C	∗	∗	∗	∗	∗	∗	∗
A3C+psc	†			†	†	†	†

表 2.6. Private Eye の画面 19 まで到達した回数

method	到達回数
A3C	1
A3C+psc	8
A3C+UCB	17

1. ベースラインの A3C に関して，文献 [5] の報告値と本論文の実験結果はほぼ同じとなった．これは本論文で用いている A3C の実装および実験設定が，文献 [5] で行われたものを再現できていることを示す一つの証拠になると考えられる．同様に，本論文での実験の報告値の信頼性を担保する結果となっている．

2. A3C+UCB は一貫してベースラインの A3C より良い結果を出している．この結果から UCB ベースの探索戦略はベースラインの A3C と比較して高い報酬が得られる環境の探索に成功している．

3. A3C+UCB (提案手法) の “Private Eye” と “Solaris” のそれぞれの平均スコアは **7643.5**，および，**4622.7** に到達した．これは比較した手法の中では最高の成績であった．

4. A3C+UCB と A3C を比較すると，7 種類すべてのゲームにおいて有意水準 1%で有意な差が認められた．

5. A3C+UCB と A3C+psc を比較すると，“Enduro”,“Montezuma”,“Solaris”,“Venture” の 5 ゲームに関して有意水準 1%で有意な差が認められた．ただし “Enduro” に関しては A3C+psc の方が平均スコアが高いので，A3C+psc の方が有意によい．また “Freeway” ,“Gravitar” に関しては有意水準 1%で有意な差は認められなかった．

2.6.2 提案手法の Private Eye に対する考察

ここでは Atari 2600 の中で最も難しいゲームの一つである Private Eye で，提案手法である A3C+UCB により獲得した戦略を調べ，比較した手法の中で最高記録を達成した本質的な利点を考察する．このゲームでは，メインキャラクターが車を運転しながら町中にあるアイテムを集め，特定の場所に運ぶことを目的としている．アイテムを取得したりアイテムを特定の場所まで運ぶことで正の報酬（得点）が得られる．また，それ以外では，正の報酬を得ることはできない．さらにアイテムの数は

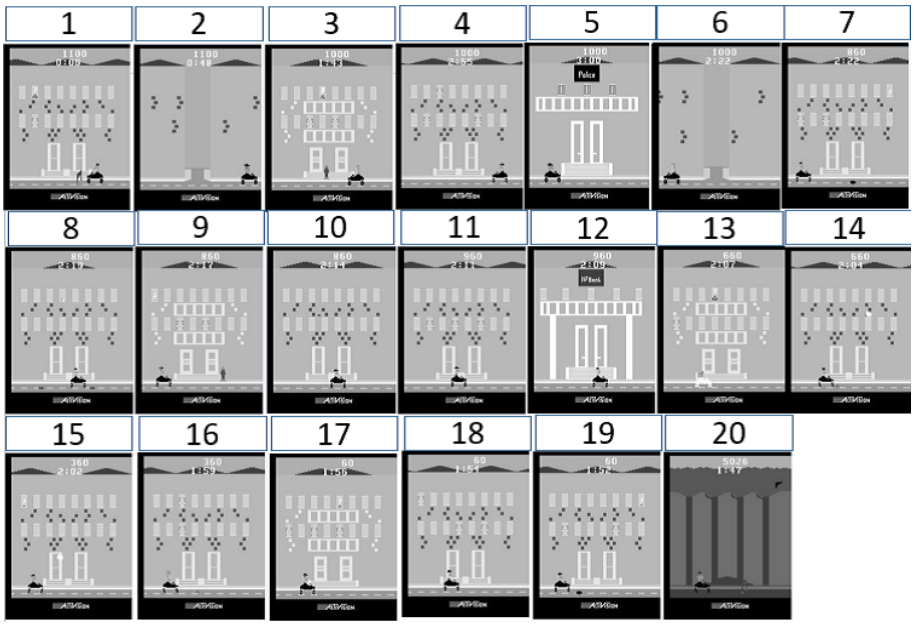

図 2.2. Private Eye のマップ

とても少ないため，正の報酬を獲得できる局面は非常に限定的であり，負の報酬を獲得できる局面も同様であるため疎な報酬環境ゲームに分類されている．

Private Eye には，エージェントが高い報酬を得る戦略の獲得を妨害するいくつかの要因があると考えられる．具体的には，敵や障害物にぶつかると負の報酬（スコアが減る）となる．また，高い正の報酬はスタート地点から，遠い局面に配置されている．図 2.2 に，Private Eye のマップ全体の一部を示す．このマップは 20 もの建物で構成されており，番号の 7 と 8，14 と 15 はそれぞれ繋がっている．メインキャラクターの初期位置は 4 番である．正の報酬を得られる場所は 3, 7, 9, 11, 13, 15, 17 番目の建物で相対的に小さい 100 点の報酬，そして 19 番目の建物で相対的に大きい 5000 点の報酬が得られる．敵などは 7, 8, 10, 13, 14, 15, 16, 19 番目の建物に存在する．

このゲームにおいて高得点を獲得するには，一番奥の建物までキャラクターを動かし，大きい報酬を得る必要があるが，そのためには負の報酬を得ても遠くまで探索を進める戦略が必須となる．従来法では，学習が進むに連れて，エージェントが負の報酬を獲得しないようにこれらの局面を回避し，初期位置近くの小さな報酬を獲得できる局面を重点に探索する傾向があると容易に推測できる．つまり，大きな正の報酬が得られる局面に到達するためには，長期的な探索戦略をもっている必要がある．提案手法は，初期位置近くの小さい報酬を獲得しても，探索報酬によって，未知の局面であるより奥の建物にエージェントを進ませようとする．

具体的な検証として，それぞれの手法のそれぞれの試行で，画面 19 まで到達できた回数をカウントした．その結果を表 2.6 に示す．この結果からもわかるように，提案法では 30 回のうち半数以上の 17 回は高い報酬がある建物の場所まで到達できた．一方，従来法は 1 回のみであった．

表 2.7. 各提案モジュール個別の効果検証

method	Enduro	Freeway	Gravitar	Montezuma	Private	Solaris	Venture
提案手法 (単一の戦略)	32.0	22.1	313.0	0.0	1409.0	170.0	0.0
提案手法 w/ psc	4.0	19.6	100.0	20.0	967.0	2798.0	0.0
提案手法	28.5	24.0	406.7	126.7	7643.5	4622.7	163.3

結果として，提案手法においてエージェントは未知の局面を有効に探索し，大きな報酬を得ることに成功したと考えられる．このように，報酬が得られる状態が非常に少ないゲームにおいても，提案手法の探索が有効に機能していることがわかった．

2.6.3 各提案モジュールの効果

提案手法の各モジュールの効果を検証するために次の設定で実験を行った．

- 提案手法 (単一の戦略): 戦略 π' を用いず，従来法で用いられるように，戦略 π の更新に探索報酬 e を加算して使用．
- 提案手法 w/ psc: 式 2.8 を式 2.5 で置き換え

表 2.7 結果を示す．なお，提案手法の結果は，表 2.3 で示した結果を再掲したものである．

まず，提案手法 (単一の戦略) と提案手法の結果を比べることで，二種類の戦略を別に学習することの効果を見ることができる．この結果から，明らかに二種類の戦略を用いることの効果が確認できる．

また，提案手法 w/ psc と提案手法の結果を比較することで，UCB に基づく探索基準を用いることの効果を見ることができる．psc を用いることでも一定の効果があるが，UCB に基づく探索基準の方が効果が高いことが確認できる．

このように，提案手法は複数のモジュールを複合した結果としてうまく効果を発揮していることがみてとれる．

2.7 まとめ

本論文では，報酬が疎な環境に対する深層強化学習に関して議論を行い，報酬が疎な環境に適した深層強化学習の枠組みを一つ提案した．提案手法の効果を検証するため，深層強化学習法のベンチマークとしてよく用いられている Arcade Learning Environment (ALE) に実装されている Atari 2600 のゲームの中で，疎な報酬環境ゲームに分類されるゲームを用いて実験を行った．提案法は，ベースラインとなる A3C と比べて疎な報酬環境ゲームのスコアを大きく向上できることを示した．また，疎な報酬環境ゲームの中で，“Private Eye” と “Solaris” においては比較した手法の中で最高の成績であった．

提案法の活用戦略の行動は，探索戦略という異なる戦略からサンプリングされたエピソードを利用しているため，活用戦略の方策勾配は探索戦略によるバイアスがかかる．つまり提案手法はヒューリスティックな方法論であり，活用戦略の理論的な最適性の保証を与えられるかは現状不明である．学習の収束性や理論的な解析に関しては，今後の課題として取り組みたいと思っている．

第3章　牌譜からの教師付き学習による コンピュータ麻雀プレイヤの構築

3.1　背景

第 1 章で述べたように麻雀に対して人間の知識や牌譜を用いることなく，強化学習を行ってもそのゲーム性から効率的な学習は行われないと考えれれる．そこでこの章では麻雀の多人数性に着目し，対応することを試みる．

不完全情報多人数ゲームは，多くのプレイヤがお互いに異なる情報を持って，個人の利益を最大化する目的であるため，政治や経済といった現実世界の問題をモデル化したものと言える．世界中で楽しまれているあるポーカーは不完全情報ゲームであり，プレイヤ人数が 2 人から 10 人程度であっても成立するゲームである．コンピュータポーカーは 1990 年代から研究がおこなわれてきた [41]. 現在では AAAI や NIPS といった国際会議でもポーカーを題材とした研究が取り上げられる．また 2 人ポーカーを解いた論文はサイエンス誌に載るなど不完全情報ゲームが現実の世界に与える影響は大きい [7].

不完全情報ゲームの戦略として，ナッシュ均衡 [35] がある．これはプレイヤが独立で合理的である場合，どのプレイヤも一人だけ戦略を変更しても現在の戦略よりも利益を増やすことが出来ない状態である．コンピュータポーカーは情報集合を抽象化することによって，これを近似的に求めることで成功を収めた．情報集合とは特定のプレイヤの視点で起こる可能性のある集合を指す．ナッシュ均衡戦略を求める方法としては Counter Factual Regret minimization (CFR) である [8]．これはオンライン学習であり，$Regret$ の最小化手法を用いてナッシュ均衡戦略を求めることである．

一方，2007 年と 2010 年の 4 人零和不確定不完全情報ゲームである麻雀の研究は平均レベルのプレイヤにすら満たないレベルである [64, 72]．しかしながらここ数年で麻雀に関する結果がいくつか報告され，トッププレイヤに近いレベルになっている． [67, 71, 57].

麻雀の難しさの原因としては多人数であるため状況によってプレイヤの行動の目的が違う，不完全な情報がある，確率的な要素があるといったことが原因で探索空間が膨大であることがあげられる．先に説明した CFR は情報集合がポーカーより膨大で適応が現実的でないため，ほかの手法が求められる．

3.2 本章での提案

多人数ゲームでのゲーム木の探索は色々提案されている [26, 43, 50]．しかし麻雀においては相手の手牌が不完全情報であることと，手に入れた牌も不明であることから，相手が捨てる牌を予測することが難しく，探索があまり効果的でないことが予想される．

本章では情報集合や計算量が膨大で現実的に解くとこが出来ない麻雀に対し，多人数性を排除しながらも最適な戦略を求めるという点で考察を行った．提案手法として二つの方法を述べる．

一つの方法は以下の通りである．相手を考慮しない，一人麻雀を定義する．麻雀を行う時の人間の思考は一人麻雀をもとにしているため，一人麻雀から四人麻雀への拡張は容易だと考えられる．次にその一人麻雀を行う一人麻雀プレイヤを作り，一人麻雀と四人麻雀の差を解析する．その結果から降りる局面を認識と鳴きの機能を追加によって四人麻雀へ適応する手法である．

もう一つは以下の通りである．プレイヤのモデル化を行う．モデル化とは相手の状態や行動を三種類の要素の組み合わせである．これをもとに四人麻雀を仮想的な一人麻雀のようにとらえ，モンテカルロ法により手を決定する方法である．

この章と次章の本論文から見た位置付けはまずこの複雑なゲーム性を持つ麻雀というゲームに対して人間の知識や牌譜を利用してある程度のレベルの AI を作成することである．そのため強化学習や報酬が疎であるといった話題はこの章では触れない．

3.3 麻雀

3.3.1 麻雀のルール

この章では麻雀のルールや用語について解説する．中国や台湾でも同じような牌を使った各国の麻雀があるが，この論文で扱うのは日本の麻雀と呼ばれるルールである．日本の麻雀のルールでもプレイ人数やゲーム終了条件の異なるローカルルールがある．この論文で扱うのは一般的に普及しているリーチ麻雀である．

麻雀は 4 人のプレイヤが相手の得点をやり取りするゲームである．各プレイヤは 13 枚の牌を持っており，1 枚牌を引いて （**ツモ**） 1 枚牌を切る行為と**鳴き**または**副露（フーロ）**と呼ぶ，捨てた牌を利用する行為を各プレイヤごとに基本的には順番に行いながら，14 枚の牌を組み合わせて役を作り得点を得る．役を完成させることを**和了（ホーラ）**と呼ぶ．またツモして和了することもツモと呼び，相手の捨てた牌で和了することを**ロン**と呼ぶ．またロンされることを**放銃**と呼ぶ．ツモの場合，全員から得点をもらい，ロンの場合，放銃したプレイヤだけが得点を払う．基本的には誰かが和了して一局が終了する．これを特定の回数行い最終的な点数の大きさを競うものである．

牌は図 3.1 で表される通り，全部で 34 種類あり，それぞれが 4 枚あるため合計は 136 枚となる．牌には数字の 1〜9 のいずれかが書かれた**数牌**と文字の書かれた**字牌**がある．数牌は 3 種類 (万子，索子，筒子) あり優劣はない．字牌は 7 種類あり，東，南，西，北の総称である**風牌**と白，発，中の総称である**三元牌**に分けられる．

図 3.1. 牌の種類

各プレイヤには東，南，西，北のいずれかが割り当てられ，その牌をメンツにすることで役になる，この方角を**自風**と呼ぶ．さらに親を全員が 1 回行うまでは東，その後は南が**場風**として決められ，メンツにすることで役になる．すなわち字風と場風が東の時に，東をメンツにすると二つの役が完成することになる．

和了に必要な 14 枚の組み合わせを完成させるためには，**面子（メンツ）**と呼ばれる特定の 3 枚の組み合わせが 4 組と，**対子（トイツ）**と呼ばれる 2 枚の同一牌が 1 組必要である．メンツは同じ牌を 3 つ集める**刻子（コウツ）**と数字の連続する 3 つ牌を集める**順子（シュンツ）**がある．鳴きには**ポン**と**チー**と**カン**がある．ポンは捨てられた牌と同じ牌を手牌から二つ表にして刻子にする．チーは捨てられた牌と手牌の二つを表にして順子にする．カンは捨てられた牌と同じ牌を手牌から三つ表にして刻子にする．そして一枚，牌を引いて補充する．一度も鳴きを行わないことを**面前**と呼び，得点が高い．また，あと 1 枚で和了という状態を**聴牌**と呼ぶ．和了には前述の 4 つの面子と 1 つの対子を作るほかにも 7 つの対子を作る七対子と数牌の 19 と字牌を全種類集める国士無双がある．

麻雀はプレイヤは親と子に分かれ，その数は親が 1 人で子が 3 人である．親と子の異なる点は，得られる得点である．役が同じ場合，親の得点は子の 1.5 倍である．しかしツモの得点を払う場合，親は子の 2 倍払う．親は一局ごとに順番に交代する．また親が和了した場合はもう一度同じプレイヤが親を行う．ゲームの終了は，一般的には親を全員が 1 ないし 2 回行った場合である．親を全員が 1 回行うのを東風戦，2 回行うのを半荘と呼ぶ．

一般的なゲームの流れを説明する．各プレイヤは 1 局開始時に**配牌**と呼ばれる，13 枚の牌を山から持ってくる．また**ドラ**を決める．これは和了した時にドラを持っていた場合，得点が加算される牌のことである．当然プレイヤはドラを使いきれるようにゲームを進めていく．また**赤ドラ**と呼ばれる各数牌の 5 の内，1 枚は赤色でありドラと同等の価値を持つ．ゲームが開始すると親から順に，東，南，西，北と順番に牌をツモって切る行為を繰り返し，和了を目指す．途中鳴きが入った場合，鳴いた次のプレイヤからツモを行う．例えば，西家が切った牌を東家が鳴いた場合，次からは南家から始まり，北家の手番が飛ばされた格好になる．チーはひとつ前のプレイヤからしかできないが，ポン，カンはどのプレイヤからでもできる．局の終了は大きく分けて二つある．一つは誰かが役を完成させて和了した場合である．もう一つは誰も和了せずに，ツモる牌がなくなった場合である．この場合を**流局**と呼ぶ．この時に聴牌であることを**流局聴牌**や**形式聴牌**と呼び，ほかのプレイヤの聴牌している数に応じて点数を得る．

麻雀は四人のうち誰かが役を構成すると 1 局が終了し，これを定められた回数の局数を行う（通常は 4 または 8 回）．最初の持ち点は 25,000 から開始し，最終局（**オーラス**）を終了した時の得点の多さに応じて順位が決まる．同点の場合は，ゲーム開始時の親に近いほうが優先される．途中で誰かが持ち点が 0 未満になっても試合は終了する．オーラスが終了しても誰も 30,000 点を超えていない場合は**サドンデス**と呼ばれ，誰かが 30,000 点を超えるまでゲームが続行される．例外的に西 4 局が終了しても 30,000 を超えない場合はそこでゲームが終了となる．

リーチを行うには 1,000 点を支払う必要がある．この 1,000 点を**リーチ棒**と呼ぶ．このリーチ棒は和了したプレイヤの和了点に追加される．流局した場合はこのリーチ棒は次の局に持ち越される．

点数を決める追加要素に**本場**がある．最初は 0 本場から始まり，親が和了するまたは流局した際に，本場が増える．誰かが和了した時に，本場の数 ×300 点が和了点数に追加される．

流局時には聴牌していたプレイヤは点数を得て，聴牌していないプレイヤは点数を払う．その点数は，聴牌していたプレイヤの数によって決まる．

3.3.2 麻雀の役と用語

以下に一般的な麻雀用語を説明する．これらは麻雀の戦略を語る上で必要になる．

リーチ
: 鳴きを一度も行わず，あと 1 枚で和了できる手牌になった時に宣言出来る行為である．リーチ宣言の後は手牌を変更することはできない．すなわち，リーチ後にツモで入手した牌は，和了できる場合を除きすべてそのまま捨てなくてはならない．

ダマ
: リーチを出来る状態であっても，リーチしないこと

門前
: 鳴かないこと

聴牌（テンパイ）
和了に必要な牌の枚数が 1 枚の状態

降り
自分の和了を諦め，相手の和了牌を捨てないようにする行為

差し込み
相手の和了しそうな牌を切ること．

回し打ち
和了を目指した最善手を選ぶのではなく，放銃を避けつつ和了や聴牌を狙う行為

ターツ
あと一枚あればメンツになる組み合わせ．

両面待ち
2 と 3 といった連続した二つの数字のターツ．面子の完成に 2 種類の牌がある

カンチャン待ち
2 と 4 といった一つ数字の空いた二つの数字のターツ．

ペンチャン待ち
1 と 2 といった最小または最大の数字が二つ並ぶ数字のターツ．

単騎待ち
和了に対子が必要な場合

シャンポン待ち
対子が 2 種類ある聴牌

愚形
両面待ち以外のこと

流局
誰も和了することなく，ツモれる牌がなくなること

現物
相手がすでに捨てた牌．ルール上，現物をロンされることはない

筋
相手の捨てた数牌の 3 つ外側の牌

壁
2,3,7,8 が 4 枚見えた時のその牌の内側の牌

向聴数
　和了に必要な牌の枚数.

完全一向聴数
　聴牌にあと一枚必要な状態ですべての牌が受け入れ枚数にかかわる状態

ブロック数
　メンツやメンツ候補の数.

受け入れ枚数
　シャンテン数を減らす枚数.

タンヤオ牌
　数字の 2 から 8 の牌.

浮き牌
　前後の数字がない牌

明刻
　ポンして同じ牌を 3 枚集めた面子

暗刻
　ポンせず同じ牌を 3 枚集めた面子

明槓
　鳴きをして同じ牌を 4 枚集めた面子

暗槓
　鳴かずに同じ牌を 4 枚集めた面子

ツモ切り
　ツモった牌を切ること

手出し
　ツモ切り以外の牌を切ること

　また今回の論文で役を表 3.1 にまとめる．複数の役が成立した場合は，それぞれを足し合わせる．役満は 13 翻に相当する．
　麻雀の点数は役の数だけでなく符にも影響を受ける．符は各面子の状態や和了の仕方によりそれぞれ符が決定されるそれらを足し合わせることで決まる．

基本符
　すべて和了につく:20 符

門前加符
　面前でロン和了した場合:10 符

ツモ符
　ツモでの和了した場合:2 符

面子加符
　順子，明刻，暗刻，明槓，暗槓に対してそれぞれ 0，2，4，8，16 符．さらにその牌が 19 字牌であれば 2 倍

対子加符
　対子が役牌であれば 2，字風かつ場風である場合は 4 符

和了形加符
　愚形待ちに 2 符

これらを足し合わせ一桁の数字が 0 でなければ繰り上げる．例外として七対子は 25 符，平和のツモ和了は 20 符，またロン時の最小符は 30 符とする．翻数と符を計算して表 3.2，表 3.3，表 3.4 から点数が決定される．カッコ内の数字はツモ時の子と親の払う点数である．自分の手牌によっては複数の解釈が可能な場合が存在するが，その場合は高い点数を申告する．

表 3.1. 麻雀の役

役名	翻数	説明
リーチ	1	面前で聴牌した時に宣言することで成立．リーチ宣言の後は手牌を変更することはできない．すなわち，リーチ後にツモで入手した牌は，和了できる場合を除きすべてそのまま捨てなくてはならない．
一発	1	リーチしてから次の自分の順番までに和了すると成立．ただしほかのプレイヤの鳴きがあった場合，成立しない．
門前清模和	1	面前かつツモで和了したとき成立．
役牌	1	字風，場風，三元牌のいずれかをメンツにすると成立．
平和	1	面前かつメンツが 4 つとも順子で，対子が役牌でなく、両面待ちの時に成立．
タンヤオ	1	2〜8 の数牌のみの場合に成立．
一盃口	1	面前かつ同じ順子が 2 組ある場合に成立．
河底	1	最後の牌で和了．
嶺上開花	1	カンをした時のツモ牌で和了．
槍槓	1	ほかのプレイヤがカンした牌で和了．
ダブルリーチ	2	自分の最初の捨牌でリーチ宣言すると成立．
チャンタ	2 食い下がり 1	メンツと対子のすべてに 1，9，字牌を含んでいると成立．
混老頭	2	すべての牌が 1，9，字牌であると成立．
三色同順	2 食い下がり 1	同じ数字の順子が三種類あると成立．
三色同刻	2	、同じ数字の刻子が三種類あると成立．
一通	2 食い下がり 1	一種類の数牌が 1 から 9 まであると成立．
対々和	2	メンツが全て刻子であると成立．
三暗刻	2	ポンせず作ったが刻子が 3 つあると成立．
三槓子	2	カンを 3 回すると成立．
小三元	2	三元牌のうち 2 つを刻子，1 つを対子にすると成立．
七対子	2	対子が 7 組ある場合に成立．
二盃口	3	一盃口が二つある時に成立．
純全帯	3 食い下がり 2	メンツと対子のすべてに 1，9 を含んでいると成立．
混一色	3 食い下がり 2	一種類の数牌と字牌のみで成立．
清一色	6 食い下がり 5	一種類の数牌のみで成立．
四暗刻	役満	ポンせず作ったが刻子が 4 つあると成立．
大三元	役満	三元牌すべてを刻子にすると成立．
字一色	役満	すべての牌が字牌である場合に成立．
四喜和	役満	風牌のすべてを刻子か対子にすると成立．
緑一色	役満	索子の 23468 と発のみで成立．
九蓮宝燈	役満	面前かつ 1112345678999 の牌とさらに 1〜9 の牌を追加した手牌である場合に成立
清老頭	役満	すべての牌が 1，9 であると成立．
四槓子	役満	カンを 4 回すると成立．
国士無双	役満	1，9，字牌を全種類とさらにその中からもう一枚で成立．
天和，地和	役満	最初のツモで和了する．

表 3.2. 子の点数表

	1 翻	2 翻	3 翻	4 翻
20 符		(400,700)	(700,1300)	(1300,2600)
25 符		1600	3200 (800,1600)	6400 (1600,3200)
30 符	1000 (300,500)	2000 (500,1000)	3900 (1000,2000)	7700 (2000,3900)
40 符	1300 (400,700)	2600 (700,1300)	5200 (1300,2600)	8000 (2000,4000)
50 符	1600 (400,800)	3200 (800,1600)	6400 (1600,3200)	8000 (2000,4000)
60 符	2000 (500,1000)	3900 (1000,2000)	7700 (2000,3900)	8000 (2000,4000)
70 符	2300 (600,1200)	4500 (1200,2300)	8000 (2000,4000)	8000 (2000,4000)
80 符	2600 (700,1300)	5200 (1300,2600)	8000 (2000,4000)	8000 (2000,4000)
90 符	2900 (800,1500)	5800 (1500,2900)	8000 (2000,4000)	8000 (2000,4000)
100 符	3200 (800,1600)	6400 (1600,3200)	8000 (2000,4000)	8000 (2000,4000)
110 符	3600 (900,1800)	7100 (1800,3600)	8000 (2000,4000)	8000 (2000,4000)

ゲームが途中で流局する場合が正式には存在する．以下の条件が成立すると流局扱いになる．

四人リーチ
　四人のプレイヤが全員リーチを宣言する．

三家ロン
　あるプレイヤの牌で残りの全員がロンする．

四槓流れ
　一局の間で四回カンが起きること

四風連打
　最初の一巡で全員が同じ風牌を切ること

九種九牌
　最初の配牌で 19 字牌が 9 種類あること

本研究ではこれらは成立する可能性が低いため，自己対戦のルールや思考部分では考慮しない．

3.4　ポーカーの関連研究

ここでは麻雀に近いゲームとして同じ不完全情報ゲームのポーカーの研究について述べる．ポーカーでも人数によって用いられる手法が違う．ここでは二人と多人数で分けて説明する．

表 3.3. 親の点数表

	1 翻	2 翻	3 翻	4 翻
20 符		(700)	(1300)	(2600)
25 符		2400	4800 (1600)	9600 (3200)
30 符	1500 (500)	2900 (1000)	5800 (2000)	11600 (3900)
40 符	2000 (700)	3900 (1300)	7700 (2600)	12000 (4000)
50 符	2400 (800)	4800 (1600)	9600 (3200)	12000 (4000)
60 符	2900 (1000)	5800 (2000)	11600 (3900)	12000 (4000)
70 符	3400 (1200)	6800 (2300)	12000 (4000)	12000 (4000)
80 符	3900 (1300)	7700 (2600)	12000 (4000)	12000 (4000)
90 符	4400 (1500)	8700 (2900)	12000 (4000)	12000 (4000)
100 符	4800 (1600)	9600 (3200)	12000 (4000)	12000 (4000)
110 符	5300 (1800)	10600 (3600)	12000 (4000)	12000 (4000)

表 3.4. 4 翻以上点数表

	5 翻	6,7 翻	8,9,10 翻	11,12 翻	13 翻以上
親	12000 (4000)	18000 (6000)	24000 (8000)	36000 (12000)	48000 (16000)
子	8000 (2000,4000)	12000 (3000,6000)	16000 (4000,8000)	24000 (6000,12000)	32000 (8000,16000)

3.4.1 二人ポーカー

3.4.1.1 Counter Factual Regret minimization

ポーカーが成功した手法としては Counter Factual Regret minimization (CFR) が挙げられる．CFR はゲームを各情報集合毎に集約したゲーム木と表現し，自己対戦を行うことで各ゲームでの $Regret$ を求め，それをもとに各情報集合での戦略を更新していくアルゴリズムである．$Regret$ を R として式 (3.1) に定義する．

$$R_i^T(I,a) = \frac{1}{T}\sum_{t=1}^{T} \pi_{-i}^{\sigma^t}(I)(u_i(\sigma^t|_{I\to a,I}) - u_i(\sigma^t, I)) \tag{3.1}$$

ここで T はノードの訪問回数，i は I での手番プレイヤ，$\pi_{-i}^{\sigma^t}(I)$ は戦略 σ^t を i 以外のプレイヤが利用した時の情報集合 I へ到達する確率である．$u_i(\sigma^t, I)$ は情報集合 I でのプレイヤ i の平均効用であり，$u_i(\sigma^t|_{I\to a,I})$ は I で行動 a をとった時の平均効用である．この計算式の正の値を式 (3.2) で抽出する．

$$R_i^{T+}(I,a) = max(R_i^T(I,a), 0) \tag{3.2}$$

この値をもとに，次の自己対戦での戦略を式 3.3 で変更する．

$$\sigma_i^{t+1}(I,a) = \begin{cases} \frac{R_i^{T+}(I,a)}{\sum_{a\in A(I)} R_i^{T+}(I,a)} & if \sum_{a\in A(I)} R_i^{T+}(I,a) > 0 \\ \frac{1}{|A(I)|} & otherwise \end{cases} \tag{3.3}$$

$A(I)$ は合法手の集合であり，$|A(I)|$ は合法手数である．このように自己対戦を行い式 (3.1)，(3.2) と (3.3) を何度も更新し，最終的には戦略の平均をとることで近似のナッシュ均衡戦略に収束することが保障されている [6]．CFR は実装上，多人数ゲームであっても適応可能ではあるが，ナッシュ均衡戦略に収束するかは保証されていない．また多人数になるほどゲームの状態数が増え，計算量的に適応が難しい．

CFR をそのまま適応しようとするとゲーム木が莫大になるため，CFR のバリエーションとして，チャンスノードまたはアクションノードを一つだけサンプリングする方法がある．チャンスノードとは確率的な事象が起こるノードであり，ポーカーでは場に新たなカードが配られることに相当する．またアクションノードとはプレイヤが行動を選択するノードであり，ポーカーではベッド，コール，フォールドがこれに相当する．この手法により世界チャンピオンに勝つ [8] コンピュータポーカーの作成に成功した．

Bowling らは “CFR+” と呼ばれる手法を提案し，2 人ポーカーを解いたと主張した [7]．これは負の値でない “regret matching+” を用いてゲーム木全体に対してサンプリングを行う．行動としては良さそうな手の後に悪そうな手を選択する．これにより戦略の開拓の可能性がなくなることが経験的に知られている．これにより現在の戦略を CFR+での解とすることで平均戦略を計算する必要がなく計算量の削減が可能になった．CFR+によって戦略を計算するため，200 台の計算機を 68.5 日使用した．実力は自分にとって最悪な戦略をとり続けるプレイヤに対して人間が一生プレイできるゲーム数を行っても統計的有意に負けない強さになった．

3.4.1.2 搾取戦略

CFR は負けない戦略ではあるものの，相手が弱いプレイヤであっても 1 ゲームあたりの報酬は低い．そこで相手をモデル化し 1 ゲームあたりの報酬を増やす研究がある．

あらかじめモデルの状態数と状況においての行動の頻度を計算しておく方法として，ゲームの棋譜を利用する方法が [55] が提案されている．今までの行動履歴や頻度，場の状況から複数の相手モデルを作る．相手モデルとは行動の確率をいくつかのクラスタに分類することである．また相手の手のカードを予測し，手が開かれた時の勝率を計算し，モンテカルロ木探索を用いることで搾取を試みる．

モンテカルロ木探索とはシミュレーションによる平均報酬とノードの探索回数を考慮して，見込みのある手に対してほかの手より多くのシミュレーションと深いゲーム木の探索を行う手法である．モンテカルロ木探索が成功した例として，コンピュータ囲碁の世界では Crazy Stone [17] がヒューリスティックに基づいた確率分布による手をシミュレーション中に選択することで 9 路盤であれば，プロ並みの実力を得ている．

ここではモンテカルロ木探索の一つである UCT (UCB applied to Trees) について説明する [24]．UCT には四つのステップがある．

子ノードの選択

UCT では式 (3.4) で求められる UCB (Upper Confidence Bounds) 値が最大となる子ノードを選択する．

$$\overline{X_i} + C\sqrt{\frac{lnT}{T_i}} \tag{3.4}$$

$\overline{X_i}$ は子ノード i の平均報酬，C は定数，T は親ノードの探索回数，T_i は子ノードの探索回数である．子ノードが一度も選択されていない場合は最優先で選択される．UCB は第一項である子ノードの平均報酬の高さと第二項の探索の回数が少ないことによる報酬の高さの期待を C でバランスをとった値である．

子ノードの展開

子ノードが存在しない場合，合法手を生成し，局面を更新する．

プレイアウト

子ノードから終端ノードまでシミュレートを行う．シミュレート時の行動として一番単純なのはランダムだが，一回のシミュレートの質を向上させる方法として，ヒューリスティック，確率分布などを用いて行動が選択されることもある．

更新

終端ノードの結果をもとに探索経路中のノードの情報を更新する．

この手順を終了条件を満たすまで行う．最終的に平均報酬の高いノードや探索回数の多いノードなどを選択する．モンテカルロ木探索は合法手の生成が出来れば適応が可能であるため，色々なゲームに用いられる．

棋譜を用いずにゲーム中に相手モデルを更新して搾取する方法 [18] もある．局面の抽象化のため，手の強さを勝率からいくつかの種類に分類しておく．手の強さの初期値として CFR を用いてある情報集合における手の強さ分布を求める．そしてゲームプレイ中に相手の手が開かれるたびに，そのゲームでの経過した情報集合における手の強さの頻度分布を更新する．またレイズの数やタイミングが近い情報集合も同時に更新を行う．次のゲームでは更新した頻度分布をもとに Expectimax 探索行うことで自分の手の決定する．これにより CFR よりも 1 ゲームあたりの報酬を増やすことに成功した．

3.4.2 多人数ポーカー

ここでは 3 人以上のポーカー研究について述べる．Risk らは CFR を 3 人ポーカーに適応した．当然 3 人ポーカーは状態数が膨大なため CFR を直接使うことはできない．そこでゲームをなるべく抽象化する必要がある．抽象化の方法としては手の強さを 2 つに分ける “PR2” と過去の状態を記憶しない代わりに手の強さの数を 16 に増やした “IR16” を提案した．この手法は 2009 AAAI/IJCAI Computer Poker Competition で優勝している．

古井ら [63] はゲームのプレイヤ人数が増えても，ゲームの本質は変わらないと考え，より人数の少ない戦略も有効に使えるのではないかと考えた．そこで相手の行動を抽象化や削減することで本来よりも人数の少ないゲームとして考慮した．ゲームとしてはテキサスホールデムではなく，より単純なルダックホールデムを用いる．計算量が困難なプレイヤ人数のゲームに対して相手行動の抽象化と削減により CFR より少ない計算量で近似のナッシュ均衡戦略を求めた．またナッシュ均衡戦略から大きく離れた特定のプレイヤに対してナッシュ均衡戦略よりも多く搾取することも可能になった．

これらのポーカーの研究は相手が固定されているが，麻雀の場合，インターネット麻雀サイトなどでは不特定の相手と対戦であり，対戦数も 1 ゲームのみであるため一人一人モデル化を行うのは困難である．また特定の相手と連続して対戦する場合にしても相手の手牌を具体的に予測することが人間にも難しいため，行動を予測するのは困難である．また人間であっても一人の相手に着目して搾取を行うなどは戦術として語られていないことからも現実的とは言えない．

3.5 麻雀の関連研究

ここでは麻雀の関連研究について述べる．

北川らは 3 層ニューラルネットワークを用いた学習を提案した [72]．牌譜をもとに何を切るかという局面と鳴きの場面を自分の手番の合法手のみ探索し，3 層ニューラルネットワークで学習する方法である．いわゆる牌効率の悪さや降りるときの安全牌の切り方に問題があった．

三木らはモンテカルロ木探索を用いた麻雀プレイヤを提案している [64]．この手法では，麻雀のゲーム木の探索は探索の分岐数が膨大であり，正確に行うのは困難なため，相手の手牌や行動をランダムでシミュレートするモンテカルロ木探索を用いている．麻雀の知識をほとんど使わないにもかかわらず，和了に必要な牌の枚数を下げるように打つという単純なルールに基づいたプレイヤよりも成績が上回る結果となった．しかし，相手はシミュレート時にほとんど和了できないため，役を作ることが困難になる鳴きをするというように，人間が行うプレイとは大きく異なる挙動を示した．これら二つの関連研究も平均レベルのプレイヤに達していない．

四人麻雀を対象とするのではなく，より単純な一人麻雀を考えた研究がある．小松ら [66] は一人麻雀のルールとして 13 枚の配牌と 18 回のツモからより点数の高い手を作るというルールである．小松らの研究の提案手法は次にツモる牌がわかる完全情報ゲームとして現在の手牌にツモ牌をすべて追加しその中で最も点数の高い手牌を構成し，点数を牌に追加し必要でない牌を選択するという方法である．これを何度も行うことで捨てるべき牌を選択する．この結果，ランダムに牌を捨てるモンテカルロ法に比べ，効果的に役を作ることに成功した．

捨て牌の危険度の推定を推定する研究として我妻らは SVR を用いた方法を提案している [62]．これは牌譜に記録された局面データから特徴ベクトルを抽出して危険度として期待損失点の予測を行うものである．しかし教師データに聴牌している局面が少なく学習が上手くいかない問題や人間との一致度という主観的な方法で推定の精度を求めなければならない問題があった．

3.6 機械学習

ここでは本研究で用いた機械学習の手法について述べる．使用した機械学習の手法は平均化パーセプトロンと Support Vector Machine (SVM) である．どちらも教師あり学習であり，データには正解として正例と負例のどちらかのついたラベルがついている．

3.6.1 平均化パーセプトロン

パーセプトロンはニューラルネットの一種である．教師データから出力 y，特徴ベクトル $\mathbf{x}$，重みベクトル $\mathbf{w}$，バイアス項 b とするとこれらの関係は以下の式 (3.5) で表される．

$$y = f(\mathbf{x} \cdot \mathbf{w} + b) \tag{3.5}$$

$f(x)$ は以下の式 (3.6) で表される．

$$f(x) = \begin{cases} 1 & if \quad x \geq 0 \\ -1 & otherwise \end{cases} \tag{3.6}$$

学習では適切な重みベクトルを得ることが目的である．そのため学習中に教師データの正解ラベル t と今の重みベクトルで得られた出力が異なっていた場合，重みベクトルを以下の式 (3.7) で更新する．

$$\mathbf{w} \leftarrow \mathbf{w} + (t - y)\mathbf{x} \tag{3.7}$$

パーセプトロンではこの更新を繰り返すことでデータを分類する重みベクトルを学習する．パーセプトロンは正例と負例が線形分離可能である場合は収束することが証明されている．

しかし，線形分離が不可能である場合，学習を止めるタイミングによっては重みベクトルが大きく異なる．平均化パーセプトロンとは最終的な重みベクトルとしてこの学習中の過去の重みベクトルの平均をとることである．これにより安定した重みベクトルを求めることが出来る．

3.6.2 Support Vector Machine

SVM は，Vapnik らによって提案された 2 値分類の手法である．SVM は，超平面とそれに最も近い教師データとのマージン を最大化するように，教師データを線形分離する超平面を求める．教師データのラベルを y，特徴ベクトルを $\mathbf{x}$，分離超平面の法線ベクトルを $\mathbf{w}$ とし，識別関数を $g(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ とする．$g(\mathbf{x}) = 0$ と $\mathbf{x}$ の距離は $|g(\mathbf{x})|/||\mathbf{w}||$ となる．また $g(\mathbf{w} \cdot \mathbf{x} + b) \leq 1$ の条件の下でパラメータを調整することから，マージン最大化するには $\frac{2}{|\mathbf{w}|}$ を最大化すればよいということになる．

ラグランジュの未定乗数法を用いることで式 (3.8) を最小化する問題になる．

$$L(\mathbf{w}, \boldsymbol{\alpha}, b) = \frac{1}{2}|\mathbf{w}|^2 - \sum_{i=1}^{l} \alpha_i (y_i(\mathbf{x_i}\mathbf{w} + b) - 1) \tag{3.8}$$

ここで $\alpha_i \leq 0$ はラグランジュ乗数である．この α_i について最大化し，$\mathbf{w}$ について最小化する．L を偏微分することで式 (3.9) と式 (3.10) を得る．

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{3.9}$$

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x_i} \tag{3.10}$$

この三つの式から双対問題である式 (3.11) を最大化する問題に置き換えることが出来る．

$$\begin{gathered} W(\alpha_i) = \alpha_i - \tfrac{1}{2} \textstyle\sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j} \\ \text{subject to } \forall \alpha_i \geqq 0 \\ \textstyle\sum_{i=1}^{l} \alpha_i y_i = 0 \end{gathered} \tag{3.11}$$

今までの説明は教師データが超平面で分離可能という前提を置いていた．しかし実際には超平面ではなく超曲面である問題も多い．その場合には教師データを高次元空間に写像してからその次元上で線形分離を行う．$\mathbf{x}$ を非線形写像 $\phi(\mathbf{x})$ とすると式 (3.11) は式 (3.12) と書き換えられる．

$$W(\alpha_i) = \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x_i}) \phi(\mathbf{x_j}) \tag{3.12}$$

ここで内積の $\phi(\mathbf{x_i})\phi(\mathbf{x_j})$ をカーネル関数 $K(\mathbf{x_i}, \mathbf{x_j})$ とすることで高次元ベクトル演の必要がなく，計算量が大きく減る．本研究ではカーネル関数を以下の式 (3.13) とする．

$$K(\mathbf{x_i}, \mathbf{x_j}) = exp(-\frac{|\mathbf{x_i} - \mathbf{x_j}|^2}{\sigma^2}) \tag{3.13}$$

これはガウシアンカーネルとして知られている．σ はハイパーパラメータであり，超局面の複雑さを決定する．

3.7 最適化手法

機械学習においては目的関数の最適化が必要になる．目的関数の最適化の方法として FOBOS と Adagrad を説明する．

3.7.1 FOBOS

FOBOS [20] は劣勾配法に似たオンライン学習アルゴリズムであり，微分不可能な点を持つ目的関数でも学習が可能になる方法である．FOBOS は二つの手順を繰り返すことで重みベクトルを得る．一つ目は勾配による重みの更新であり，二つ目は正則化のための重みの更新である．

$\mathbf{w_t}$ を重みベクトル，η_t を学習率，g_t を勾配とする．添え字の t は t 回目の更新時のそれぞれの値とする．勾配の重み更新は以下の式 (3.14) で計算する．

$$w_{t+1/2} = w_t - \eta_t g_t \tag{3.14}$$

次に正則化のための重み更新であるが，正則化の関数 $R(w)$ とすると，勾配の重み更新は以下の式 (3.15) で計算する．

$$w_{t+1} = \arg\ min_w(|w - w_{t+1/2}|^2 + \eta_t R(w)) \tag{3.15}$$

この式は $R(w)$ によっては閉じた式になり，計算を容易に行うことが可能である．λ を正則化項とすると $R(w)$ が L1 正則化すなわち $R(w) = \lambda|w|$ の場合，式 (3.15) は式 (3.16) になる．

$$w_{t+1} = sign(w_t)max(0, w_{t+1/2} - \lambda\eta_t g_t) \tag{3.16}$$

また L2 正則化すなわち $R(w) = \lambda\|w\|_2^2$ の場合，式 (3.15) は式 (3.17) になる．

$$w_{t+1} = \frac{w_t - \eta_t g_t}{1 + \lambda} \tag{3.17}$$

以上の二つの手順を繰り返すことで更新は可能になるが，特徴ベクトルの次元が大きい場合，正則化の更新に時間がかかる．そこで一回の学習で更新される特徴ベクトルだけ正則化の計算を行う．Λ を前回の更新から学習が行われた数とする．正則化による勾配の更新は λ を Λ に置換した式になる．

3.7.2 Adagrad

一般的に機械学習では学習率は次第に小さくしていく必要がある．Adagrad [19] は重みベクトルが更新されるたびに学習率が小さくなっていくアルゴリズムであり，式 (3.18) で表現される．

添え字の i は特徴ベクトルの i 番目の要素を示している．$g_{t,i}$ は t 回目に更新された特徴ベクトルの i 番目の勾配である．

$$w_{t+1,i} = w_{t,i} - \frac{\eta g_{t,i}}{\sqrt{1 + \sum_{k=1}^{t} g_{t,i}^2}} \tag{3.18}$$

3.8 一人麻雀プレイヤの作成

この節では四人麻雀に拡張するための土台となる一人麻雀プレイヤを説明する．また，先行研究で提案された一人麻雀での一人麻雀プレイヤとの性能比較も行った．

3.8.1 一人麻雀

ここでは一人麻雀のルールについて述べる．使う牌の種類は変更しないが，以下の三つのルールが追加される．

- 相手がいないため，相手の捨て牌はない
- 配牌とツモのみでプレイする
- 鳴きとリーチはない

評価について述べる．この一人麻雀を最終的には四人麻雀に適応させるため，この一人麻雀の性能の良さが四人麻雀にも生きなければならない．そこで上手いプレイヤの特徴を反映させる必要がある．特徴として四人麻雀では成績の良いプレイヤほど，平均和了点が低く和了率が高いことが統計で明らかになっている [61]．上手いプレイヤの平均和了点が低くなる最も大きな理由は，手役を無理に狙った打ち方をしないためである．麻雀の点数の特性上，ある一定以上の点数の役は難易度の割には点数が高くないため，比較的点数の低い役で多く和了することが多くの場合効率が良い．また，自分が和了することによって，相手の和了する機会を減らすという意味でも和了率の高さは重要である．

本研究では，平均和了点より和了率を重視したプレイヤ作成するため，一人麻雀の評価の際には和了したか否かのみで判断する．

3.8.2 平均化パーセプトロンによる一人麻雀プレイヤ

この節では一人麻雀プレイヤを作成方法について述べる．本研究では，平均化パーセプトロン [16] を用いた教師付き機械学習により一人麻雀プレイヤを作成した．パーセプトロン学習には，依存関係がある特徴量を大量に用いた学習を，比較的小さな計算コストで行うことができるという特長がある．

以下，$\mathbf{x}$ を局面の特徴ベクトル，$\mathbf{w}$ を重みベクトル，n をベクトルの次元とする．x_i は i 番目の特徴ベクトルの値，w_i は i 番目の重みベクトルの値とすると，ある与えられた局面のスコアは式（3.19）で計算される．

$$h\left(\mathbf{x}, \mathbf{w}\right) = \sum_{i=1}^{n} x_i w_i \tag{3.19}$$

14 枚の手牌から 1 枚切った 13 枚の手牌の特徴量を求め，それぞれについてスコアを求める．次に牌譜で実際に切られた手 t^* とスコアの最も高かった手 $\hat{t}$ を比較する．それらが異なる場合，t^* の特徴ベクトル $\mathbf{x}_{t^*}$ と $\hat{t}$ の特徴ベクトル $\mathbf{x}_{\hat{t}}$ を用いて，新しい重みベクトル $\mathbf{w}'$ を式（3.20）で計算する．

$$\mathbf{w}' = \mathbf{w} + \mathbf{x}_{t^*} - \mathbf{x}_{\hat{t}} \tag{3.20}$$

この更新を牌譜のすべての局面に対し繰り返し行い，毎局面の重みベクトルの平均値を最終的な重みとする．

特徴量としては北川らの論文 [72] を参考に設計した特徴量を用いた（表 3.5）．特徴ベクトルの次元は 37,514 となった．パーセプトロン学習における学習データ全体の走査回数は 3 回とした．

表 3.5. 一人麻雀プレイヤの特徴量

特徴量	次元数
通常手，七対子，国士無双の向聴数	$15+7+14=36$
4 色のうち n 色（n=1〜3）の牌を 0 にしたときの n と向聴数	$3\times16=48$
複数の色（1〜4）と各色の向聴数（0〜11 まで）の組み合わせ	$12+78+364+1365=1819$
色の中で最も多い色の数	15
色の中で最も多い色の数+字牌の数	15
2 から 8 牌の数	15
メンツと 1,9 を含まない連続した 2 牌のターツの合計とメンツ候補の組み合わせ	$6+12+72=90$
各字牌の枚数と役牌かどうかとドラの組み合わせ	$5\times3\times2=60$
数牌の数字-5 の絶対値と枚数とドラの組み合わせ	$5\times5\times2=50$
各色の 1〜9 の有無の組み合わせ	512
連続する n 種類の数牌の持っている枚数の組み合わせ（n=2〜6）	$100+500+1860+8634+23760=34854$

3.8.3 学習に用いる牌譜

教師データとしてはインターネット麻雀サイト天鳳 [69] の鳳凰卓の牌譜を用いた[1]．鳳凰卓でプレイできるのは全プレイヤの中でも上位 0.1% 程度であり，牌譜の質は高いと考えられる．以下，牌譜と示した場合はこの鳳凰卓の牌譜のことを指す．ただし，この牌譜には，四人麻雀に特有の局面と手が多く含まれており，このデータを一人麻雀の学習データとして直接用いるのは不適切である．そこで，各局において初めてリーチをしたプレイヤを確認して，そのプレイヤの指された手のみを教師データとする．また局面は配牌からそのプレイヤがリーチをかけるまでの局面とした．最終的な教師データの数は約 170 万局面となった．

3.8.4 一人麻雀の実力

学習が適切に行われているかを検証するため，得られた一人麻雀プレイヤの実力を評価した．他のプレイヤの影響をなくすため，テストはプレイヤ 1 人で行う．また運の要素はなるべく減らす必要があるため，複数の一人麻雀プレイヤの実力を同じ山を用いて比較した．テストセットとして異なる山を 100 個用意し，各プレイヤについて打った結果を評価した．プレイヤは「上級者（人間）」と「平均プレイヤ（人間）」を各 1 名と一人麻雀プレイヤ，そして先行研究 [64] で用いられた Plain UCT である．平均プレイヤは天鳳 [69] の中で成績が上位約 50% に位置するプレイヤであり，上級者は本論文の著者であり，麻雀サイト天鳳 [69] において成績が上位 0.1% に位置するプレイヤである．

結果を図 3.6 に示す．一人麻雀プレイヤは平均プレイヤ以上の和了率を達成している．先行研究が平均プレイヤにも大きく届かなかったことを考慮すると，一人麻雀ではあるものの特徴量の構築と学習が適切に行われたため，大きく性能が向上しているといえる．

[1] 2009 年 2 月 20 日から 2010 年 1 月 31 日までに行われた対局

表 3.6. 一人麻雀の和了率

プレイヤ	和了率 （%）	平均和了順目
上級者（人間）	17	13.8
一人麻雀プレイヤ	13	12.7
平均プレイヤ（人間）	12	12.8
Plain UCT	2	15.0

表 3.7. 1 万局の和了率と平均点数

プレイヤ	和了率 （%）	1 局あたりの平均点数
一人麻雀プレイヤ	20.4	733
小松らの研究 [66]	14.0	850

この一人麻雀プレイヤを小松ら [66] による一人麻雀プレイヤと比較する．ただし小松らは，ドラを使用し，結果を点数で評価しているため，本研究とは設定が異なっている．

山は異なるが 1 万局での和了率と 1 局あたりの平均点数の比較を行った．結果を図 3.7 に示す．1 局あたりの平均点数では小松らの研究に劣るものの和了率では上回っており，我々の研究における一人麻雀においては牌譜を用いた学習が成功していることを示している．

3.9　一人麻雀と四人麻雀の差異の解析

前節で，一人麻雀に関しては，牌譜を用いた機械学習によって人間の平均プレイヤに近い実力を持つプレイヤが実現できることを示した．本節では，一人麻雀と四人麻雀の差異を解析するため，四人麻雀の牌譜をもとにプレイした手と牌譜の手の違いを調査する．

前節で実現した一人麻雀プレイヤは「鳴く」ことができない．実際の牌譜の和了した局面について 80 万局[2]を調査したところ，鳴くことなく和了した割合は 52% であり，鳴いて和了した割合は 48% であった．この結果から鳴きができないことの影響は極めて大きいと考えられるものの，この節ではツモ時の局面についての一人麻雀と四人麻雀の差異を解析する．

具体的には以下のように行った．最初に実際の牌譜の局面に対して一人麻雀プレイヤが切りたい牌の上位 3 つを選択する．牌譜で実際に切られた牌が，この上位 3 つの中に入らなかった局面を調査し,「降り」や「回し打ち」といったタグを人手で付与し，それらの回数を集計する．この集計結果から一人麻雀と四人麻雀の差異を解析する．

まず一人麻雀プレイヤの四人麻雀における牌の選択精度を評価した．ここでプレイヤが指定した第 n 番目までの候補に牌譜での打牌が含まれる割合を Rank n とする．比較として上級者も評価を行った．

[2]2010 年 1 月 1 日から 2010 年 12 月 31 日までに行われた対局．一人麻雀のプレイヤの学習で用いた牌譜の使用が適切ではあるが，和了時の鳴きが含まれる割合を調べるにあたり異なる牌譜であっても値に影響はないと考える．

表 3.8. 牌譜との一致率

プレイヤ	Rank 1	Rank 2	Rank 3
上級者（人間）	0.62	0.85	0.93
一人麻雀プレイヤ	0.53	0.77	0.85

テストに用いる局面は，天鳳 [69] で公開されている鳳凰卓の牌譜からランダムに抽出した 27 局から 1,342 局面を選択した．一人麻雀であるため鳴くことができる局面や鳴いた後の局面は 1,342 局面の中に含まれていない．一人麻雀では相手の捨て牌や副露した牌などはないが，テスト中ではそれらの牌は残りの枚数を数えるのに使用した．上級者（人間）は四人麻雀を行うように手を選び，一人麻雀プレイヤは一人麻雀としてプレイした．結果を表 3.8 に示す．

上級者の Rank 1 の正解率は 62% 程度で一人麻雀プレイヤは 53% であった．テストセットは異なるものの 3 層ニューラルネットを用いた方法 [72] では Rank 1 の正解率として 56% の精度が報告されている．

次に一人麻雀プレイヤの Rank 3 までに正解が含まれなかった局面，つまり残り 15% の 193 局面を調査し手動でタグ付け・分類を行った．その結果を図 3.2 に示す．外した局面のうち約 4 割が降りるべき局面である．また全体としては「役牌」,「七対子」,「染め」,「タンヤオ」など役が絡んだ局面が多い．その他には一人麻雀プレイヤ自体の手が悪手を打った局面もある．

タグについて説明する．役に絡んだタグは一人麻雀プレイヤがその役を目指せなかった局面であり,「ドラ」は切る必要のないドラを切った局面である.「分類不可」はどれを切っても同じような牌が複数あり 3 つの候補のみでは選べない局面や牌譜自体が悪手を打った局面である．

この解析により，降りが一人麻雀を四人麻雀へと拡張するため一番大きな問題であることがわかり，降りを予測できれば，15%のうちの 4 割である 6% の牌譜との一致率の向上が予想される．

3.10 降りるべき局面の認識

前節での解析により，降りる局面を正しく認識できれば牌譜一致率を上げ，実力が向上することを示した．そこでこの節では降りる局面を認識するための局面のタグ付についてまず説明し，次にその提案手法について説明する．

3.10.1 教師データの作成

教師あり学習として降りる局面を認識するにはその教師データが必要になる，しかし牌譜に「降りる」とは明示的に記されてはいない．したがって人間の手でタグ付けをする．タグとして降りかそうでないかが必要になる．

[2]天鳳の開発者である角田氏に許可を頂いた．

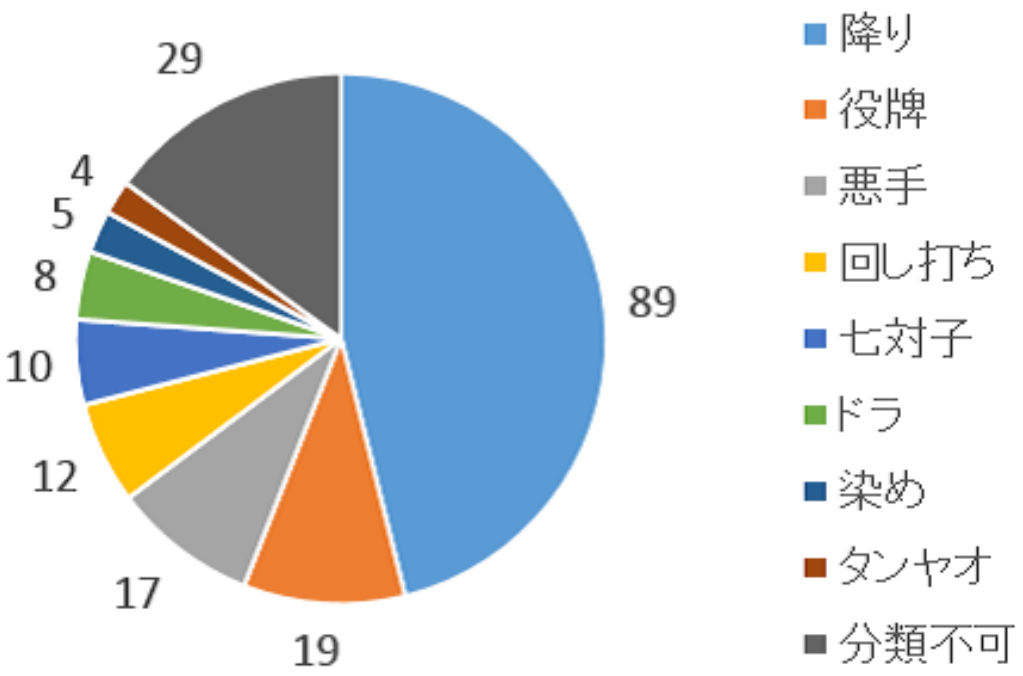

図 3.2. 外した局面の分類

表 3.9. タグ付一致数

A/B	降り	そうでない
降り	83	23
そうでない	25	922

具体例を図 3.3 に示す．図の局面はプレイヤ（画面下部）が 3 萬を切った局面である．いわゆるメンツの中抜きでありこの手の和了の可能性を諦めた手である．タグ付けにおいては一度降りるとその局では以降すべて降りたものとする．

この定義により牌譜から 509 局を通じ，5,716 局面に降りか否かのタグを著者（A）が手動でつけた．すなわち牌譜の選ばれた手を見て，降りかどうかのタグをつけることになる．

このタグ付けの妥当性を検証するため著者とは独立に研究の協力者（B）が，5,716 局面の一部である 1,053 局面にタグ付けを行った．B は麻雀サイト天鳳 [69] において成績が上位 10% に位置するプレイヤである．

両者のタグ付けの一致率を表 3.9 に示す．判断にはプレイヤの主観的な考え方が大きく作用するため，困難な問題である．しかしながら一致率は一定の水準に達している．偶然によらない一致率の指標であるカッパ係数 [44] は 0.75 であり，おおむね降りた局面とそうでない局面を当てられているといえる．一致しなかった 48 の局面を見ると「回し打ち」の局面が 47 あり，残りの 1 つは牌譜が明らかな悪手を指した手であった．

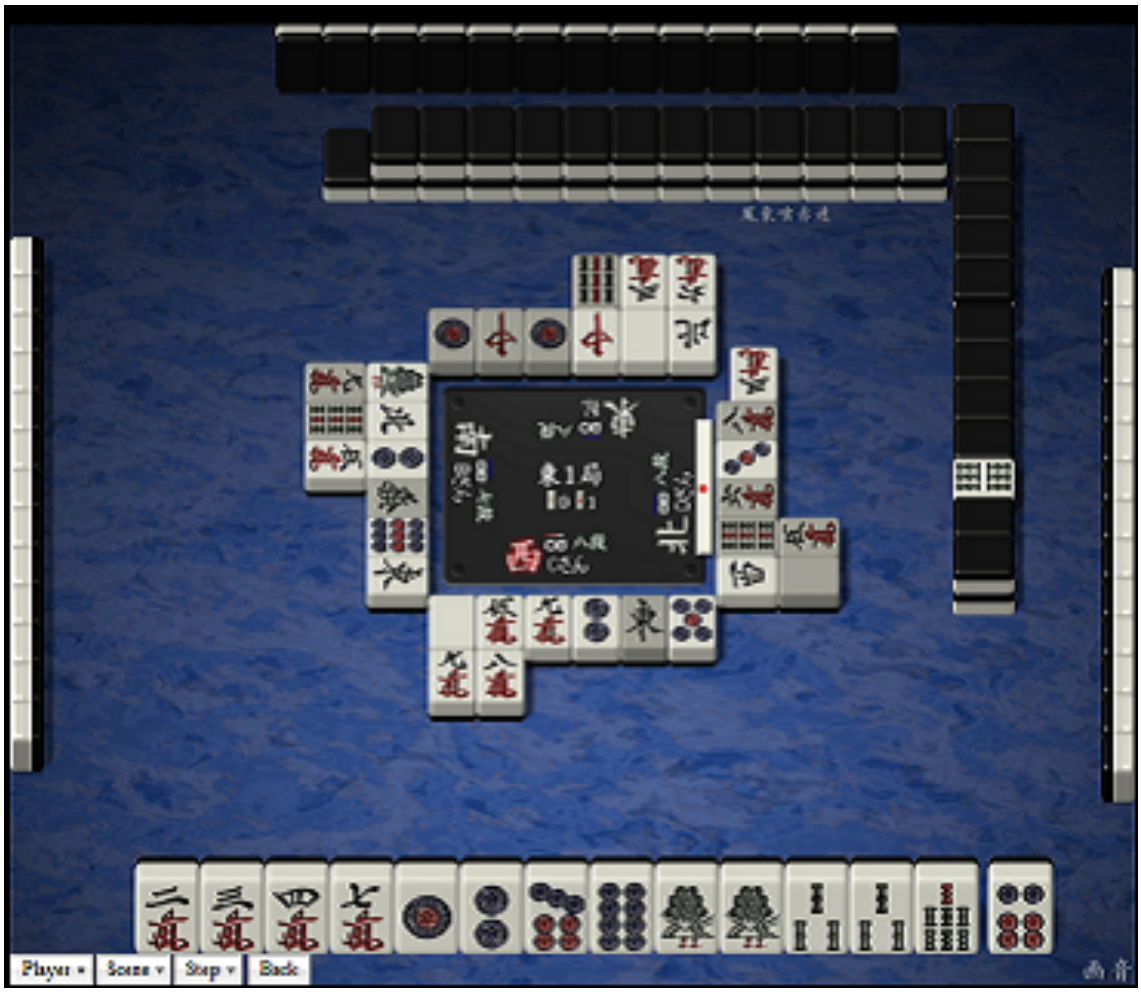

図 3.3. 降りた局面

回し打ちの具体的な例を図 3.4 に示す．他のプレイヤのことを気にしなければ最善手は 8 索であるが，右のプレイヤのリーチに対応して現物である 1 索を切ったというような局面である．

また最善手ではないと判断できても，それが降りなのか回し打ちなのか判断するのが困難な局面もあった．図 3.5 は二人の判断が分かれた局面である．ここで牌譜では 2 萬を切った．これをどう見るかであるが，メンツを崩しているという点で A は降りと判断した．しかし，見方を変えると 2 萬や 6 萬は右のプレイヤのリーチに現物であり，4 萬は筋で通りそうである．これらを切っている間に残りの形がメンツになることや危険な牌がリーチ者が切るかもしれないため，このプレイヤはまだ加点することを諦めていないだろうという点で B は回し打ちと判断した．

このように手が和了や流局時の聴牌を狙えるかどうか判断するのはそのプレイヤの主観的な考え方が大きく作用する．とくに七対子で和了できる可能性をどの程度見るかはプレイヤによってさら

表 3.10. 降り判断の特徴量

	項目	次元数
自分の状態	向聴数	1
	七対子に必要な牌の枚数	1
	リーチしているか	1
	親かどうか	1
	副露数	1
	副露により晒したドラの数	1
	持っているドラの数	1
相手の状態	リーチしているか	1
	副露数	1
	副露により見えたドラの数	1
	親かどうか	1
	切りたい牌のそのプレイヤに対しての安全度	4
場の状況	山の残りの枚数	1
	カンの数	1
	捨て牌にあるドラの数	1

に幅が大きく困難である．

また一度降りと判断した場合，定義により次からすべて降りと判断するため，降りと回し打ちで判断が食い違った場合，その次のツモ牌が安全牌であるとき片方は降り，片方は降りてないと判断される．このように一度判断を間違えるとさらに間違いが引き起こされ一致率が低下する．

3.10.2 降りるべき局面の認識

本節では，降りる局面の認識を機械学習によって行う手法を説明する．まず局面からどのような特徴量を取り出したのか説明し，認識精度を評価する．

本節の問題設定は 3.10.1 節での設定とは少し異なっている．3.10.1 節で行ったのは牌譜で何を選んだかを見てその手が降りかどうかを判断した．しかし実際には切る牌を選ぶ前に降りるかどうかを判断して，その判断に基づいた手を選択する必要がある．したがって 3.10.1 節で行った問題とは異なり，局面についての降りるかどうかを判断する分類問題とした．

降りかどうかを判断する局面の特徴量としては表 3.10 に示すものを取り出した．自分の状態と 3 人分の相手の状態と場の状況を考慮する必要があるため，特徴ベクトルの要素次元は 34（$7+3\times8+3$）である．

この研究では問題を単純にするため，降り判断は以下のようなモデルで行われていると考える．人間は降りるべき局面かどうかを決定したときに周りを気にしないときの最善手を選ぶ．そしてこの牌が周りのプレイヤに対して安全かどうかを考慮して，降りるかどうかを判断している．そのためこのモデルに必要な切りたい牌の周りのプレイヤに対しての安全度を考える．

安全度はヒューリスティクスにより決定した．安全度の高い順に以下のようになっている．

表 3.11. 降りる局面の分類結果

	降りる局面	そうでない局面
適合率	0.76	0.97
再現率	0.71	0.97
F 値	0.73	0.97

- 現物
- 1 枚以上切れている字牌または筋の 1，9 牌または壁
- 字牌または上記以外の筋
- 無筋

3.10.3　降りる局面認識の予備実験

3.10.2 節の特徴量の妥当性を検証するため，降りる局面認識の 2 クラス分類を行った．降り判断の学習に使用できる学習データは比較的少量である．そこで本研究では，マージン最大化により高い汎化性能を得られる，ガウシアンカーネルによる Support Vector Machine（SVM）を分類器として用いた．

評価は 5 分割交差検定で行った．SVM の実装として LIBSVM [14] を用い，グリッドサーチによってガウシアンカーネルで用いるパラメータのコスト（c）とガンマ（g）を最適化した．グリッドサーチの範囲はそれぞれ 2^{-10} から 2^{10} まで値を 2 倍刻みで行った．

学習データ中の降りる局面数は 531，そうでない局面は 5,185 であった．結果を表 3.11 に示す．表中の適合率とはシステムが出力した予測が実際に正解であった割合である．表中の再現率はテストデータの正解をどれだけ予測したできていたかという割合である．適合率と再現率の調和平均が F 値である．グリッドサーチの結果，$c = 2^8$，$g = 2^{-7}$ の時が最も降りる局面での F 値が高い．

この結果より比較的高精度で降りを予測できることが分かった．

3.11　一人麻雀プレイヤの鳴きの拡張

本節では一人麻雀プレイヤを鳴きができるように拡張する．またその鳴きの判断の正確さを牌譜をもとに確かめた．

3.11.1　平均化パーセプトロンによる鳴き局面の学習

鳴きについての学習も，3.8.1 節で行った平均化パーセプトロンを用いる．鳴いた後に切ることができる牌は，すでに完成しているメンツを鳴く行為（例えば 123 から 1 または 4 をチーして 1 を切

表 3.12. 一人麻雀プレイヤの鳴きに関して追加する特徴量

特徴量	次元数
副露数	5
リーチができる可能性があるか	2
各色の 1〜9 の各数牌の数を 0，1，2 以上で変換したパターン	19472
副露した中から選んだ n（1〜2）個の副露した種類の組み合わせ	$136 + 9316 = 9452$
手牌の同じ牌が 3 枚以上の数と 2 枚以上の数の組み合わせ	$5 \times 7 = 35$
手牌の同じ牌が 3 枚以上の数+ポンの数と手牌の 2 枚以上のものとの組み合わせさらにチーをしたか	$5 \times 7 + 1 = 36$
手牌と副露の中のタンヤオ牌の数と向聴数	$16 + 16 = 32$
連続する 3 つの数字の各色の有無とその数字に 1，9 が含むかとチーを含むかの組み合わせ	$512 \times 2 \times 2 = 2048$
チーを含む各色の 1〜9 の数字の有無	512
色の中で最も多い色の数+同じ色または字牌の副露 ×3	16
色の中で最も多い色の数+字牌の数+同じ色または字牌の副露 ×3	16
特定の色と字牌以外を 0 枚としたときの向聴数	16
2 つそろった役牌の数	8
3 つそろった役牌の数	6
ドラの数（0，1，2，3 以上）または 2〜8 牌のドラの数	$4 + 4 = 8$
副露数と向聴数の組み合わせ	$5 \times 15 = 75$
向聴数と役があるまたは条件付きの役または役がないかと順目の組み合わせ	$2 \times 3 \times 19 = 114$
役があるか条件付きの役かどうか役がない	3
1，9 牌を鳴いたかと 1，9 牌を加えた時に向聴数が何枚減るか（0，1，2 以上）	$2 \times 3 = 6$
手牌の 1，9 牌の枚数を 0 にしたときに向聴数が何枚増えるか（0，1，2 以上）	3
風牌 4 種類それぞれの副露を加えた手持ちの枚数（0，1，2，3 以上）の組み合わせ	$4 \times 4 \times 4 \times 4 = 256$
三元牌 3 種類それぞれの副露を加えた手持ちの枚数（0，1，2，3 以上）の組み合わせ	$4 \times 4 \times 4 = 64$
副露数と向聴数と自分の捨て牌を足して向聴数が減るか	$5 \times 4 \times 2 = 40$

る）を禁止するルールがあるため，ツモ局面と異なる．そのため鳴きとツモ局面の学習との相違点は，鳴ける局面の時に鳴かなかった手牌のスコアと鳴いて 1 枚切った手牌のスコアを比較するという点である．特徴量は 3.8.1 節に用いた特徴量に鳴きに関して以下の特徴量を追加する（表 3.12）．

特徴ベクトルの次元は 69,258 になった．パーセプトロン学習における学習データ全体の走査回数は 1 回とした．

3.11.2　鳴きの学習に用いる牌譜

3.8.1 節で使われたデータはリーチしたプレイヤの局面を教師データとした．しかし，これでは鳴いた局面が含まれないため，鳴きの学習用の教師データを 3.8.1 節で用いたものから鳴きに対応できるように変更する．各局において初めてリーチをしたプレイヤがリーチをかけるか，あるいは和了

表 3.13. 鳴きの正解率

牌譜の手	牌譜の数	完全一致した数	鳴きのみ正解	正解率
鳴く	6,230	3,755	1,435	84.2
鳴かない	22,164	19,392	N\A	90.7

したプレイヤが和了するまでの局面を教師データとした．最終的な教師データの数は約 300 万局面となった．またテストデータの数は鳴くことができる局面に限定し約 3 万局面となった．開発データとして 3,000 局面用意した．教師データの数を 30 分割し，ひとつ使用されるごとに開発データに対する正解率を観測した．

3.11.3　鳴きの正解率

鳴きの学習が適切に行われているかを検証するため，開発データに対する学習曲線とテストデータに対しての正解率を調査した．その結果を図 3.6 と表 3.13 に示す．

鳴くか鳴かないかの正解率が変動していないため，学習データの走査回数は 1 回でも十分だと考えられる．この鳴きの分類器は比較的，高精度で鳴きの局面に対して正解を出せている．データが異なるため単純な比較はできないが，北川らの研究 [72] では鳴いた局面での正解率は 21% である．鳴いた局面の正解率を大きく向上させた．鳴いた局面の我々の正解率が大きく上回った要因は学習した局面を全ての局面でなく攻めているであろう局面に限定したことにある．

以後この論文で “一人麻雀の手” とは鳴きまで含めた手のことを指す．

3.12　一人麻雀プレイヤの四人麻雀への適応

前節までの結果から降りる局面の認識と鳴きを行う機能ができたため，この節では一人麻雀プレイヤと組み合わせ，手の決定アルゴリズムについて述べる．

まず降り局面の認識と一人麻雀プレイヤの組み合わせについてゲームでの挙動を交えて説明する．相手の手番では捨て牌を見てロン和了できるのならば，すべて行う．図 3.7 は自分の番での挙動のフローチャートである．まず和了できるかどうか確認して，和了できるときはすべてツモ和了する．和了できない時は一人麻雀として牌を選択する．その牌をもとに 3.10.2 節で説明した特徴量から降りかどうかを判断する．降りでなければ，最初に選んだ牌を以下のルールで切る．降りであれば降りに必要な牌を切る．このアルゴリズムでは降りるかどうかが目的であるため，回し打ちは考慮しない．降りる牌の選択は，以下の基準で決定する．

- それぞれのプレイヤについて 3.10.2 節にある安全度の高い順位で牌を選ぶ．
- リーチしている人が複数人である場合，お互いに共通して安全度の高いものを選ぶ．
- リーチしたプレイヤがいない場合，鳴いた数の多いプレイヤに注目する．

図 3.8 は相手の番での挙動のフローチャートである．まず相手の牌で和了できる場合は和了する．次にその牌が鳴くことができて，なおかつ自分の番で一度も降りていない時に鳴きの判定を行う．この時に鳴く場合であれば，鳴く牌と切る牌を決定する．

図 3.4. 回し打ちした局面

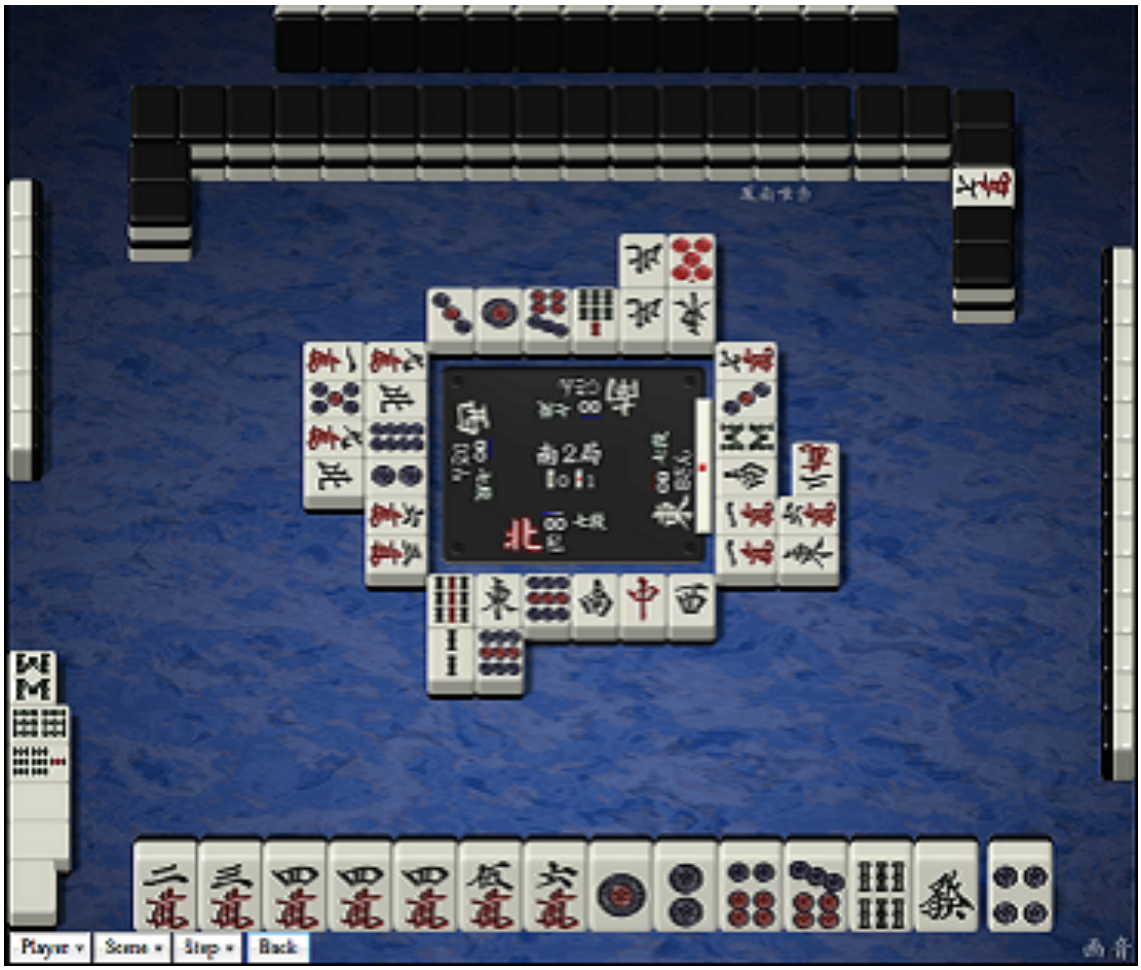

図 3.5. 降りか回し打ちか判断に困る局面

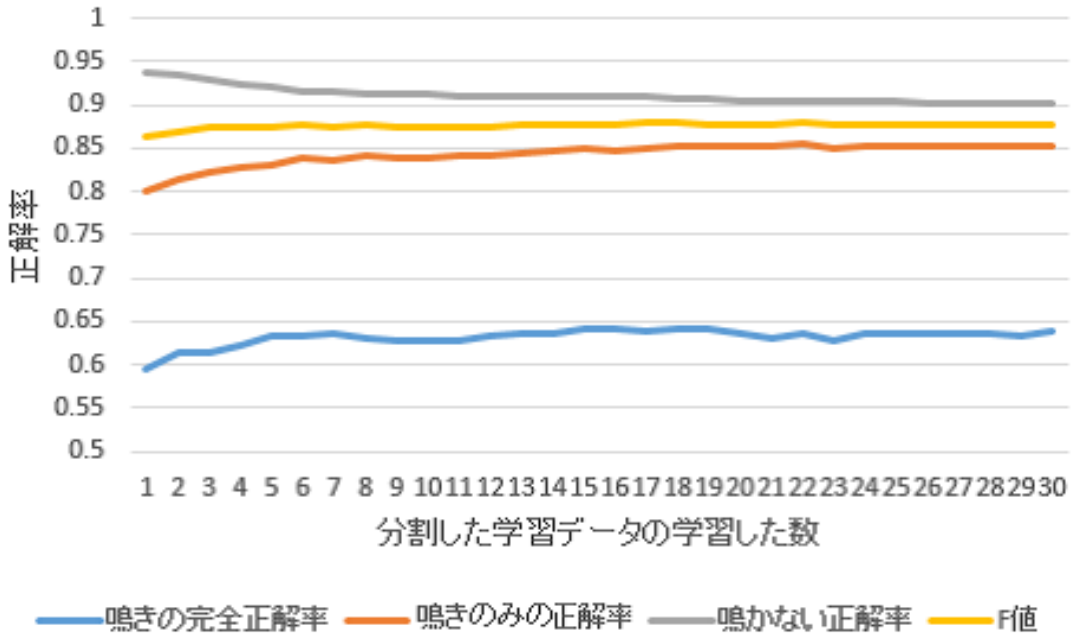

図 3.6. 学習曲線

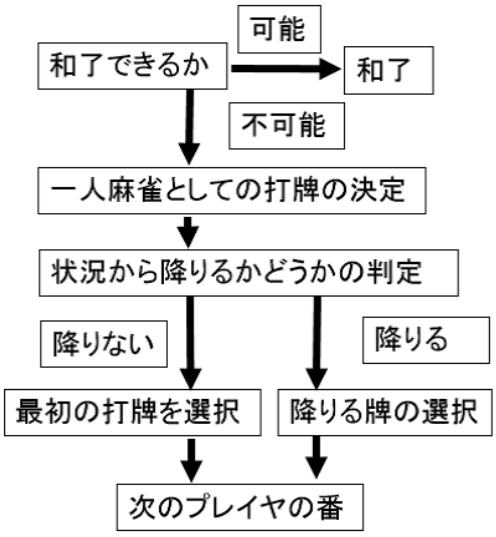

図 3.7. 自分の番でのフローチャート

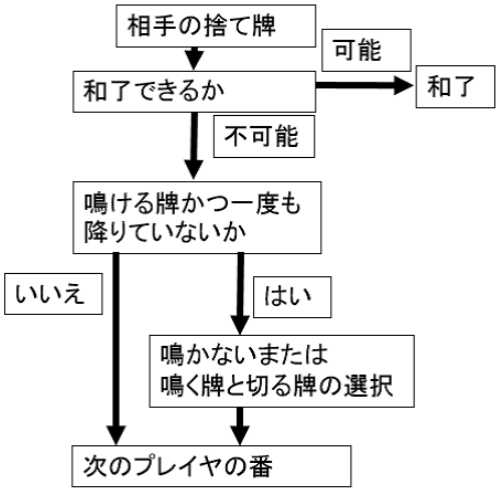

図 3.8. 相手の番でのフローチャート

第4章　相手のモデル化とモンテカルロ法による麻雀プログラム

前章では麻雀の多人数性に着目し，一人麻雀から考慮することでこれに対応した．この章では麻雀の複雑なゲーム性の要素である不完全情報ゲームということに対応する．具体的な方法は相手のモデル化である．モデル化とは相手の状態を「聴牌しているか」,「待ち牌は何か」,「点数は何点か」の三つの要素に分ける．そしてそれぞれを牌譜を用いて学習を行う．その結果をもとに四人麻雀を仮想的な一人麻雀に変換し，モンテカルロ法による手の決定を行う．

4.1　相手プレイヤの聴牌予測

この節では相手が聴牌しているかどうかを予測するモデルを学習する．麻雀のルール上，聴牌していなければ，ほかのプレイヤに点数に関して直接影響を与えることは不可能である．ゲームを人間が行う時には自分の聴牌，和了を目指し，相手も同じように聴牌，和了を目指すと考えるので，相手が聴牌をしているかどうかの判断は重要である．

4.1.1　聴牌予測の学習

聴牌予測には膨大な特徴量が必要であり，その調整を高速に行う必要がある．そこで聴牌予測には牌譜における聴牌かどうかの一致を目指した，2 値のロジスティック回帰モデルを用いる．

$\mathbf{x_p}$ を局面の相手プレイヤ p に対する特徴ベクトル，$\mathbf{w}$ を重みベクトルとすると，ある局面の聴牌率 $P(tenpai)$ は式 (4.1) で計算する．

$$P(tenpai) = \frac{1}{1 + \exp\left(\sum_{i=1}^{n} -x_{p,i} w_i\right)} \tag{4.1}$$

ここで n は特徴量の総数，$x_{p,i}$ は相手プレイヤ p に対する i 番目の特徴量のスカラー値，w_i は i 番目の重みベクトルのスカラー値である．

$\mathbf{X_N}$ を $\mathbf{x}$ のデータ集合，そのラベル値を $\mathbf{c_N}$，λ を正則化項とすると，目的関数 L_{tenpai} は式（4.2）で表される．学習はこの目的関数 L_{tenpai} を最小にする $\mathbf{w}$ を求めることである．

$$L_{tenpai}(\mathbf{w}) = -\sum_{i=1}^{N}(\mathbf{c_n} P(\mathbf{X_i}) + (1 - \mathbf{c_n})(1 - P(\mathbf{X_i})) + \frac{\lambda \mathbf{w^T w}}{N} \tag{4.2}$$

表 4.1. 聴牌予測の特徴量

特徴量	次元数
リーチを打っているか	1
副露数と捨て牌の数	$5 \times 19 = 95$
副露数とその順目	$4 \times 19 = 76$
副露数と手出しの数	$5 \times 19 = 95$
副露数と最後に手出しした牌の種類	$5 \times 37 = 185$
副露した種類と副露した時に切った牌の種類	$136 \times 37 = 5032$
ドラの種類を切ったか	34
赤ドラを切ったか	1
手出し牌とその次の手出し牌の組み合わせ	$37 \times 37 = 1369$

この重みベクトルの学習は FOBOS [20] を用いて学習を行う．学習率 η は Adagrad [19] を用いて決定する．

特徴量を 表 4.1 に示す．特徴ベクトルの次元は 6,888 になった．

4.1.2 学習に用いる牌譜

上記の聴牌予測には多くの教師データが必要になる．教師データとしてはインターネット麻雀サイト天鳳 [69] の鳳凰卓の牌譜を用いた[1]．

牌譜中のプレイヤの手番においてプレイヤの聴牌かどうかの 2 値データと，そのプレイヤ以外のプレイヤの視点から観測できる特徴量を生成し，教師データとする．局面数はおよそ 1.77×10^7 である．

4.1.3 聴牌予測の精度

学習が上手くできているかを調べるため，得られた分類器の聴牌予測の精度を調べた．評価は ROC 曲線下面積（AUC）を用いる．AUC は 2 群の分類問題における精度評価法の一つである．正例，負例を決める閾値を変え，横軸に false positive の割合，縦軸に true positive の割合を計算する．その時の曲線の面積が AUC である．理想は 1.0 でありランダムの分類器では 0.5 となる．

テストデータは牌譜から 1 局の中で 1 局面に限定し，100 局面を選択した．リーチしたプレイヤが聴牌しているかを予測するのは容易であるため，このテストデータにこのような状況は入っていない．学習率 (η) は 0.01，正則化項 (λ) は 1.0, 0.1, 0.01, 0.001, 0.0001, 0 の中から最も性能の良い値を選択する．結果は図 4.1 に示す．この開発データにはリーチした局面が含まれている．

次にテストデータに対しての結果を 表 4.2 に示す．上級者は麻雀サイト天鳳 [69] において鳳凰卓でプレイすることができるプレイヤのことを指す．これは全プレイヤの中でも 0.1% ほどであり本研

[1] 2009 年 2 月 20 日から 2013 年 12 月 31 日までに行われた対局

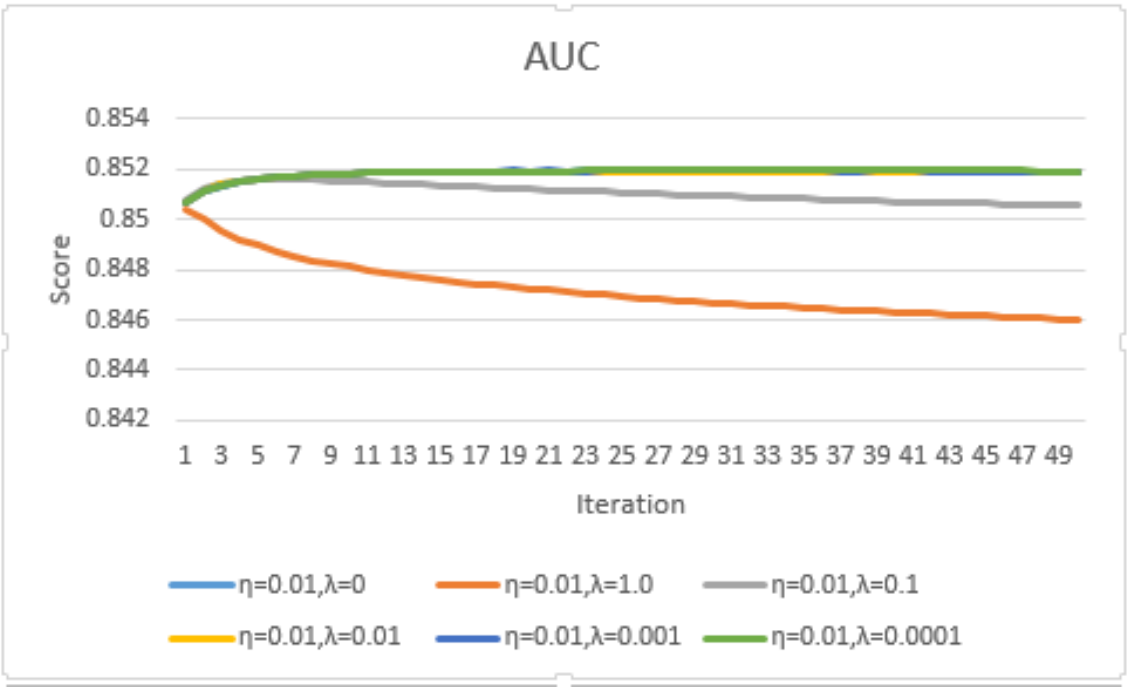

図 4.1. 聴牌予測の開発データに対する AUC

表 4.2. 聴牌に関する評価

プレイヤ	AUC
上級者	0.778
分類器	0.777

究では上級者とする．すなわち分類器は聴牌を判断する能力においては上級者に近い実力を示めしている．

4.2 待ち牌の予測

この節では相手が聴牌しているか時の相手の待ち牌を予測するモデルを学習する．麻雀のルール上，相手の待ち牌を自分が切ってしまった場合，相手の得点をすべて自分が払わなければならない．そのため相手の待ち牌を高精度で予測可能になることは重要である．

表 4.3. 待ち牌予測の特徴量

特徴量	次元数
牌の種類とその牌が何枚見えているか	$5 \times 34 = 170$
現物かどうか	34
n 回（n=0，1，2）手出しの間に通った牌	$3 \times 34 = 102$
ドラの種類	34
順目とその時に捨てた牌	$18 \times 37 = 666$
リーチ時に切った牌	37
副露の種類と副露時に切った牌	$136 * 37 = 5032$
二つの副露の種類	$136 * 136 = 18496$
捨て牌二つの種類と手出しかどうか	$37 \times 37 \times 2 \times 2 = 5476$
手出し牌とその次の手出し牌の組み合わせ	$37 \times 37 = 1369$

4.2.1 待ち牌の予測の学習

待ち牌は一般的には複数あるため，教師データは麻雀の牌の種類である 34 種類の待ち牌かどうかの 2 値分類の形になっている．それぞれの牌の種類について 2 値のロジスティック回帰モデルを用いて学習を行う．

目的関数 $L_{matipai}$ は式（4.3）で表される．

$$L_{matipai}(\mathbf{w}) = -\sum_{i=1}^{N}(\mathbf{c_n}P(\mathbf{X_i}) + (1-\mathbf{c_n})(1-P(\mathbf{X_i})) + \frac{\lambda|\mathbf{w}|}{N} \tag{4.3}$$

この重みベクトルの学習は前述の FOBOS [20] と Adagrad [19] を用いて学習を行う．

特徴量を表 4.3 に示す．特徴ベクトルの次元は 31,416 になった．

4.2.2 学習に用いる牌譜

上記の待ち牌の予測の学習には多くの教師データが必要になる．牌譜中のプレイヤの手番においてプレイヤの待ち牌とそのプレイヤ以外のプレイヤの視点から観測できる特徴量を生成し，教師データとする．局面数はおよそ 7.24×10^7 である．

4.2.3 待ち牌の予測の精度

待ち牌の予測の評価は麻雀のルールの特徴を反映している必要がある．麻雀のルール上，相手の待ち牌を一枚でも切った時点で終局するため，複数の待ち牌のうち一つだけが高精度で予測可能であっても意味がない．そこで評価としては次の手順を踏む．例を図 4.2 に示す．ある局面において特定のプレイヤに対し，待ち牌の確率が低いと予測した順に自分の牌を並べる．次にその順番通りに牌を切ったとして，初めて待ち牌と一致するまでの回数を数える．その数と和了しない牌の種類（理

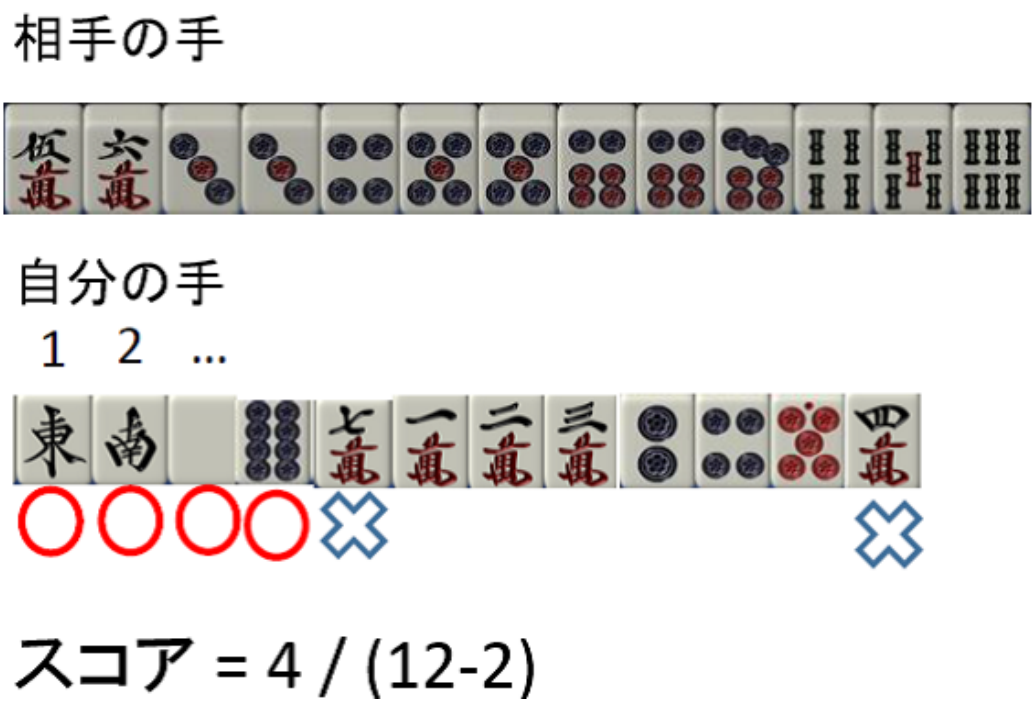

図 4.2. 待ちよみの評価例

表 4.4. 待ち牌予測に関する評価

プレイヤ	評価値
上級者	0.744
分類器	0.676
ランダム	0.502

想値）で割った値を用いる．複数の局面をテストする場合は一致までの回数と理想値の総和を割った値を用いる．テストデータは牌譜から 1 局の中で 1 局面に限定し，100 局面を選択した．なおテストでは局面に複数人が聴牌していても予測を行うプレイヤは 1 人に限定し，テストする被験者にはそのプレイヤの待ち牌が自分の手牌の中にあることを伝えている．

学習率は 0.01，正則化項は 1.0, 0.1, 0.01, 0 の中から最も性能の良い値を選択する．結果は図 4.3 に示す．テストデータに対する結果を 表 4.4 に示す．ランダムとは待ち牌の順番をランダムに並べ替えることである．分類器はランダムより性能が高く，上級者よりも低い結果になった．

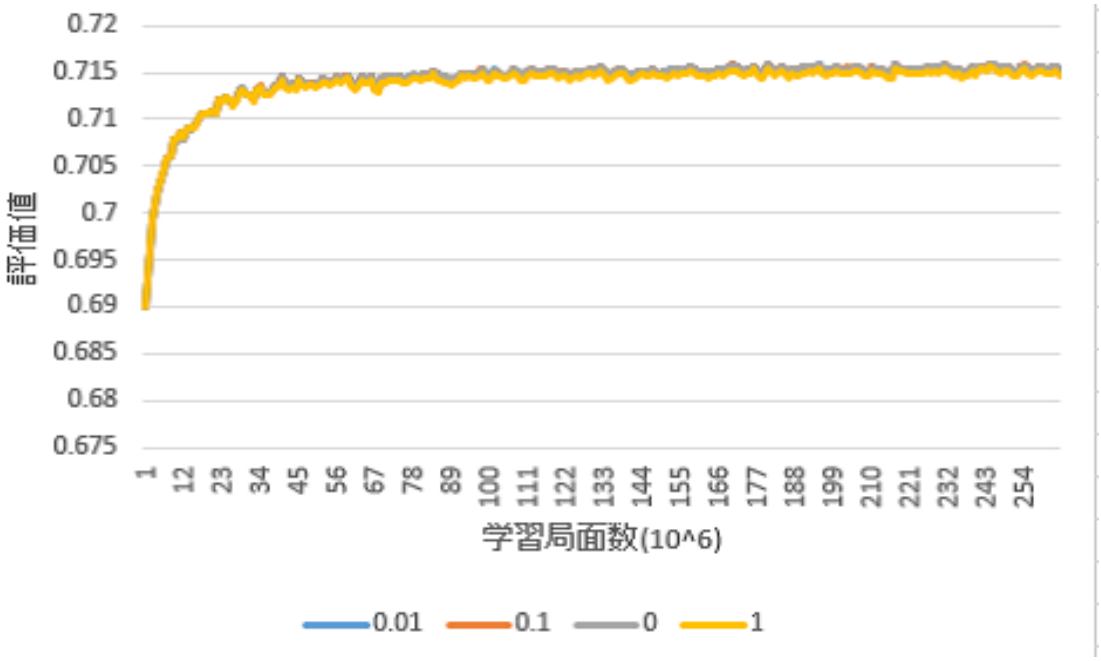

図 4.3. 待ち牌予測の開発データに対する評価値

4.3 得点の予測

この節では相手に対して待ち牌を切ってロンされた時の支払う得点を予測するモデルを学習する．この節はロンが主だが，ツモに関しても同様に学習を行う．

4.3.1 得点の予測の学習

麻雀のルール上，得点は翻数の 2 のべき乗に基本的には比例して増えていく．そのためゲームの点数をそのまま使用して学習を行った場合，同じ翻数の誤差であっても大きい翻数の方が得点の誤差が大きくなる．そこで教師データを作る際に得点に自然対数をとった．また点数は親が上がった時でも子の点数に変換を行った．教師データに対して重回帰モデルで学習を行う．

相手の予想得点 $Tokuten_i$ は式 (4.4) で計算する．

$$Tokuten_i = \sum_k x_{k,i} w_{k,i} \tag{4.4}$$

目的関数 $L_{tokuten}$ は式（4.5）で表される．

$$L_{tokuten}(\mathbf{w}) = \sum_{i=1}^{N} (Tokuten_i - \mathbf{c_i})^2 + \frac{\lambda \mathbf{w^T w}}{N} \tag{4.5}$$

表 4.5. 得点予測の特徴量

特徴量	次元数
親かどうかとリーチしているかどうか	$2 \times 2 = 4$
確定している役と見えているドラの枚数	$7 \times 8 = 56$
リーチかダマか副露と確定している役と見えているドラの枚数	$3 \times 7 \times 8 = 168$
副露の種類と確定している役と見えているドラの枚数	$136 \times 7 \times 8 = 7616$
二つの副露の種類	$136 \times 136 = 18496$
副露数と確定している役と見えているドラの枚数	$5 \times 7 \times 8 = 280$
リーチしているかどうか切った牌が筋になっているかタンヤオ牌かどうか	$3 \times 2 \times 2 = 12$
オタ風を鳴いた，さらに役牌を鳴いた，何も鳴いていない	3
切った牌がドラ，ドラの一つ隣，二つ隣，同じ色，無関係	5
タンヤオが可能な副露とドラがタンヤオと見えてるドラの枚数	$2 \times 2 \times 8 = 32$
ホンイツが可能な副露と確定している役とドラが染め色かどうか	$5 \times 7 \times 2 = 70$
チンイツが可能な副露と確定している役とドラが染め色かどうか	$5 \times 7 \times 2 = 70$
トイトイが可能な副露と確定している役とドラが染め色かどうか	$5 \times 7 \times 2 = 70$
三元牌が何種類鳴かれているか	3
風牌が何種類鳴かれているか	4

c_i は教師ありデータの得点である．この重みベクトルの学習は前述の FOBOS [20] と Adagrad [19] を用いて学習を行う．

重回帰モデルの計算は最小二乗法を用いて一意の解を求めるが可能であるが，この学習では学習の局面が多く計算機のメモリに乗らず，計算することが不可能である．そこで前述の FOBOS [20] と Adagrad [19] を用いて学習を行う．

特徴量を 表 4.5 に示す．特徴ベクトルの次元は 26,889 になった．

4.3.2 学習に用いる牌譜

上記の得点予測の学習には多くの教師データが必要になる．牌譜中のプレイヤの手番においてプレイヤの待ち牌を切られた時の得点と，そのプレイヤ以外のプレイヤの視点から観測できる特徴量を生成し，教師データとする．局面数はおよそ 5.92×10^7 である．

4.3.3 得点の予測の精度

評価は予測した得点と実際の得点との平均二乗誤差を用いる．テストデータは牌譜から 1 局の中で 1 局面に限定し，100 局面を選択した．なおテストでは被験者には何の牌を切り，どのプレイヤが上がるかを伝えている．

学習率は 0.01，正則化項 (λ) は 1.0, 0.1, 0.01, 0 の中から最も性能の良い値を選択する．結果は図 4.4 に示す．テストデータに対しての結果を 表 4.6 に示す．分類器は上級者を超える実力を得た．

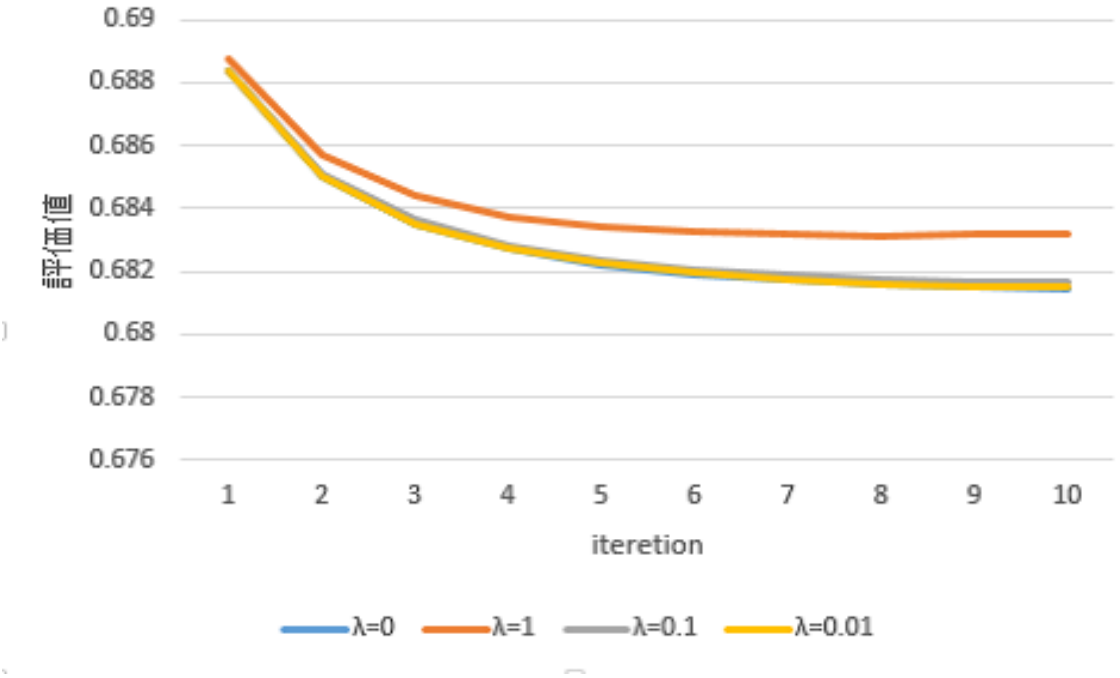

図 4.4. 得点予測の開発データに対する評価値

表 4.6. 得点予測に関する評価

プレイヤ	評価値
上級者	0.40
分類器	0.37

4.4 手の決定

前節までの結果から相手の抽象化したモデル化は上級者並みの精度で行えることを示した．この節ではこの抽象化したモデルを用いて自分の手の決定を行う．抽象化したモデルにより相手の得点を予測することが出来るが，一人麻雀の手の決定アルゴリズムにおいては，自分の手牌を良さを表す評価値はゲーム上の得点とは異なる．そこでモンテカルロ法を用いて自分の手牌をゲーム上の得点として扱うことにする．式（4.8）は手を選択するために用いる式であり，ある牌を Hai としたときのゲーム上の得点を計算する．$Sim(Hai)$ はこれから説明するモンテカルロ法によって求められるゲーム上の得点であり，二つ目の項はこの牌を切った時の期待損失点である．相手が親の時は期待損失点は 1.5 倍する．図 4.5 に示すように，シミュレーションは自分の牌だけでなく，後述の “仮想的降り” を行った時の得点（$Fold$）も計算する．$Sim(Hai)$ はシミュレーションの値と $Fold$ と比較して大きい値とする．最終的に各牌についての $Score$ を求め最大のものを実際のゲームで切る牌とする．“モンテカルロ法の手” はシミュレーションによって選ばれる手である．

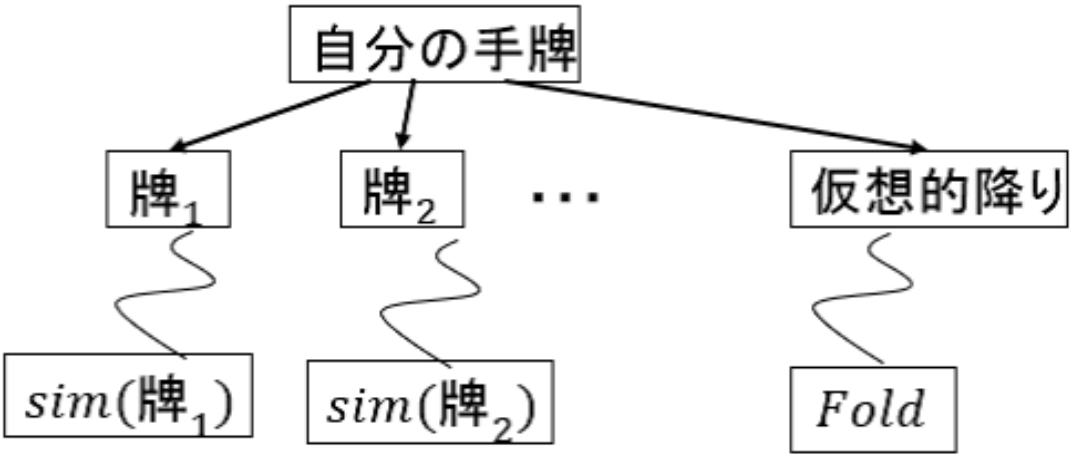

図 4.5. モンテカルロ法を用いた手の決定の概要

$$Score(Hai) = Max(sim(Hai), Fold) - \sum_{k=1}^{3} EV(k, Hai) \tag{4.6}$$

$$EV(k, Hai) = P(k = Tenpai) \times P(k, Hai = Agari) \times Tokuten(k, Hai) \tag{4.7}$$

(4.8)

麻雀におけるモンテカルロのシミュレーションには自分と相手の手番と山であるチャンスノードの時の挙動を決める必要がある．山は自分の見えている牌を数え，それ以外の牌をランダムに配置することで決定する．図 4.6 は自分と相手の手番のフローチャートである．自分の手番においては一人麻雀の手を選択する．相手の手番は抽象化したモデルが確率に基づいてリーチや和了また仮想的降りを行う．切られる牌については山からのツモ牌をツモ切りすることで決定する．このシミュレーションは一局終了まで行う．シミュレーション回数は全ての手に対して同じ回数だけ行う．シミュレーションは全ての手に対して山や相手の挙動を決める乱数は運の影響を少なくするため疑似乱数のシードを固定する．またモンテカロ木探索のようにゲーム木を深くすることは行わない．図 4.6 の降りは仮想的降りを指す．次の節から具体的に記述する．

4.4.1 自分の手番

自分の手牌のすべての牌に対して，その牌を切った局面からシミュレーションを行う．また鳴きが可能な局面では鳴いて牌を切った局面と鳴かない局面からシミュレーションを行う．

仮に現局面が降りるべき局面の時，その子ノードでは降りるための牌を切ることが出来るものの，シミュレーションでは和了しか目指さないため降りの戦略をとり続けた時の得点を概算できない．そ

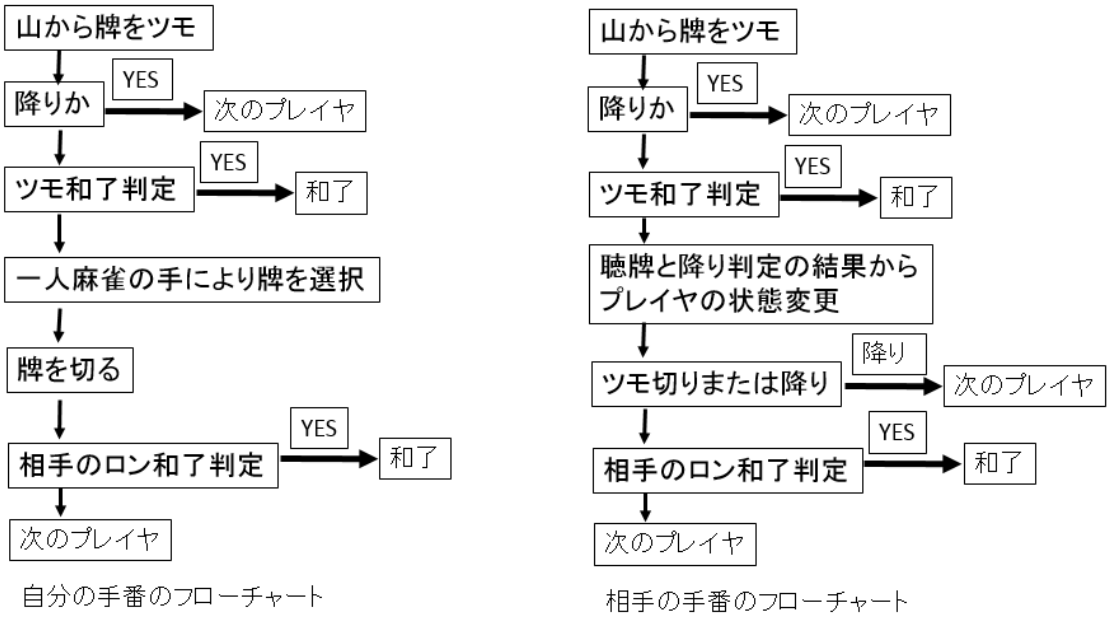

図 4.6. 各プレイヤのフローチャート

こで仮想的降りを導入し，現局面から一局終了まで降りの戦略を続けた時の得点を求める．仮想的降りは図 4.6 にあるように，山から牌をツモリ，山の数を減らすものの何も切らないで次のプレイヤの手番とする行動である．このシミュレーションでは放銃はしないが，ツモられた時の得点と流局時の得点を払うことになる．

4.4.2 相手の手番

相手の手番では相手の具体的な手牌は手を決定せずにモデル化した確率分布に基づいて行動する．シミュレーション中に相手が和了するには相手が聴牌であり，かつ切られた牌やツモ牌が和了牌である必要がある．切られた牌やツモ牌が和了牌であるかの判定を行うためには現局面での待ち牌予測した確率を用いる．シミュレーション中に待ち牌の確率分布は更新しない．

シミュレーション中の相手の聴牌かどうかを決定するには，相手のツモ局面において判定行う．聴牌していないプレイヤが聴牌する確率は一人麻雀の手を用いてあらかじめ調査した．牌譜を使用しないのは，牌譜には降りが含まれており，聴牌率が実際よりも低くなるためである．一人麻雀プレイヤに配牌とツモを与え，各順目において聴牌していない局面から聴牌になった局面を調べ，各順目の聴牌率を求めた．相手の聴牌の種類としては鳴きとリーチの二種類ある．リーチと鳴きの各順目の聴牌率を調べるときに，前者では相手の捨て牌がなく，後者は捨て牌があり，牌を鳴くことが出来る．それぞれ 10^6 回行って各順目の聴牌率を調べた．結果を 図 4.7 に示す．

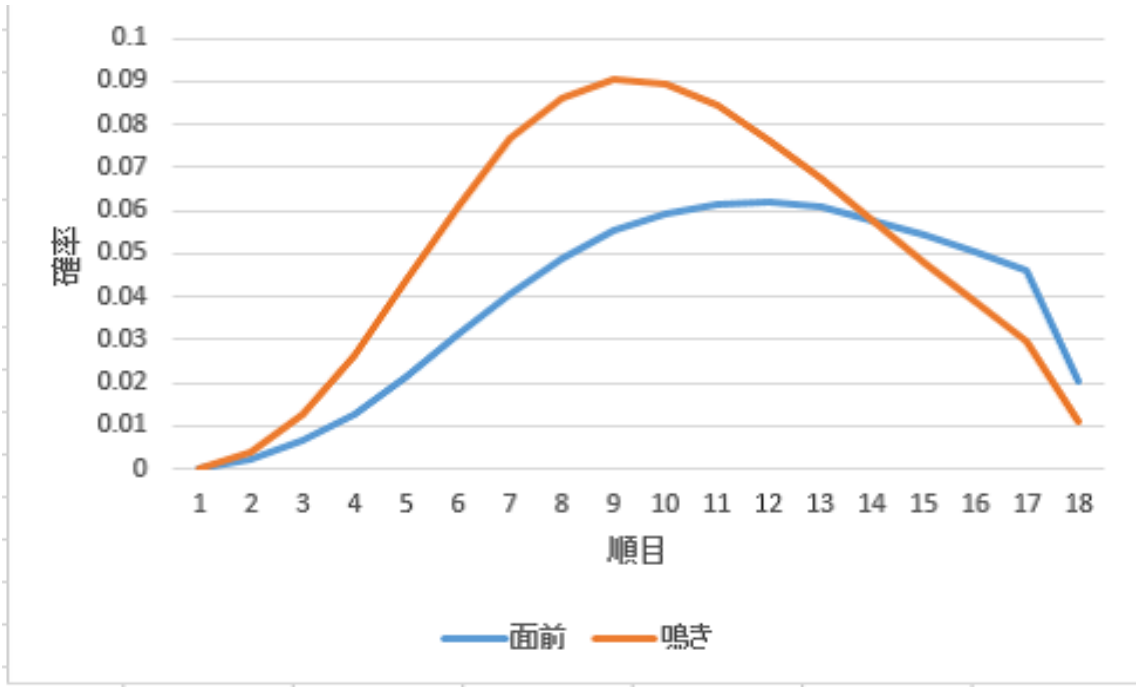

図 4.7. 各順目での初めて聴牌する確率

相手の聴牌しているかの初期値は，シミュレーション開始時に予測した聴牌率を用いて決定する．鳴きとリーチのうちどちらで聴牌するかはシミュレーション開始時に決定する．その方法は現局面で副露またはリーチしていないプレイヤに対して，副露率をもとに決定する．副露率は鳳凰卓の平均である 35%とした．

このシミュレーション中での相手の仮想的降りについての説明を行う．仮に降りがなければ，自分やほかの相手がリーチをしている局面などではリーチしたプレイヤが和了しやすくシミュレーションとしては適切でない．そこで自分の手番で前述した仮想的降りを用いる．自分の手番と相手の手番での仮想的降りとの差異は，自分の手番での仮想的降りは最初から最後まで一局を通して降りであるが，相手ノードでの仮想的降りはシミュレーションの途中からでも降りが可能なことである．

シミュレーション中に仮想的降りを行うかどうか判定は相手の手番の開始時に，場の状況からプレイヤが降りる確率を用いて判定する．一人麻雀の手と牌譜と一致しない局面は，降りた局面が多いことを利用して，降りる確率を牌譜と一人麻雀の手を用いて調べる．牌譜における自分が聴牌しているかという情報と相手の副露数とリーチしているプレイヤの数を調べ，場の状況として，その局面において一人麻雀プレイヤの選択する上位 3 つの中に牌譜で切られた手がなかった割合をその場の状況での降りる確率とする．例を図 4.8 に示す．自分が聴牌しておらず，リーチしたプレイヤが一人であるときは一致率が 60%である．一度仮想的降りの状態になるとその局はすべて降りを行う．仮想的降りの状態ではロン和了することはない．

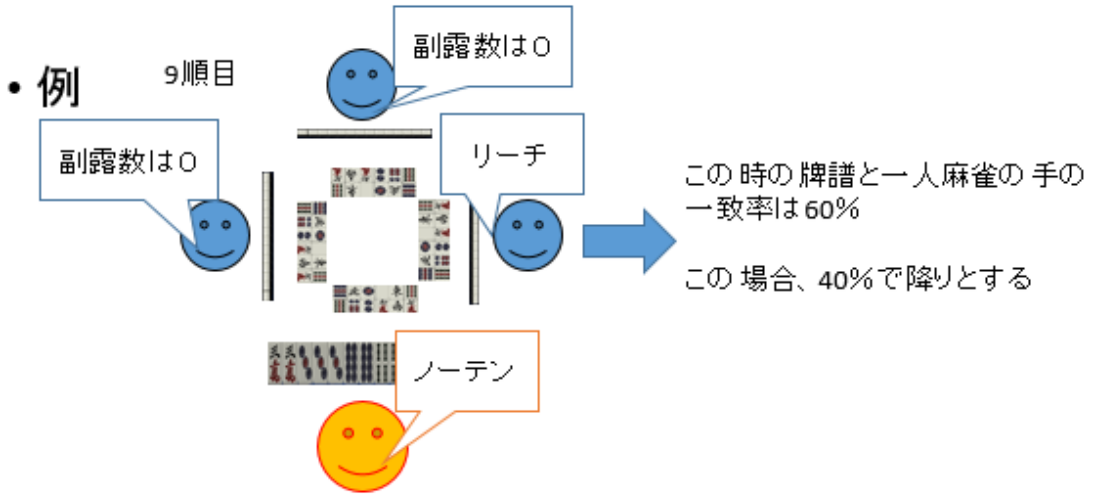

図 4.8. 相手のシミュレーション中の降りる確率

4.4.3 手の決定

序盤では自分や相手の和了までに時間がかかり手の評価が短い時間ではモンテカルロ法の手は精度が悪いことが予想される．そこで一人麻雀の手とモンテカルロ法の手を使い分ける方法を考える．

$$\begin{cases} \text{一人麻雀の手} & P(k = Tenpai) \leq \alpha \\ \text{モンテカルロ法の手} & otherwise \end{cases} \tag{4.9}$$

その方法は式 (4.9) で表される．序盤では一人麻雀の手を用いて，予測した相手の聴牌率が一人でも閾値を超えた時にはモンテカルロ法の手を採用する．この閾値は牌譜との一致率を用いて決定する．

牌譜の局面数は 10,000 局面，モンテカルロ法の手にかける時間は一手 1 秒とした．結果を 図 4.9 に示す．$Rank\ n$ は第 n 候補に牌譜での打牌が入っている割合とする．閾値が 0 の時はすべて一人麻雀の手であり，閾値が 1 の時はすべてモンテカルロ法の手であり，これらを閾値で分離することで牌譜一致率が上がる．実験の結果から閾値は 0.9 を用いて手を決定する．

4.5 評価設定

構築したプレイヤの実力を測定するため，インターネット上の麻雀対戦サイトである天鳳 [69] で対戦を行った．ルールは東風戦，赤あり，持ち時間は一手 5 秒に考慮時間が 10 秒である．鳴きについても同様の持ち時間があり，鳴ける局面では次のプレイヤは勝手に牌を引くことができない．天鳳 [69] では成績に応じて対戦できる卓が異なる．卓は 4 種類あり，上から鳳凰卓，特上卓，上級卓，一般卓がある．卓の種類は上の卓を選択可能な成績であっても 4.6 節では一般卓，4.7 節では上級卓のみに限定した．プログラムをサイト上で対戦させるための入出力インターフェースは自作した．評

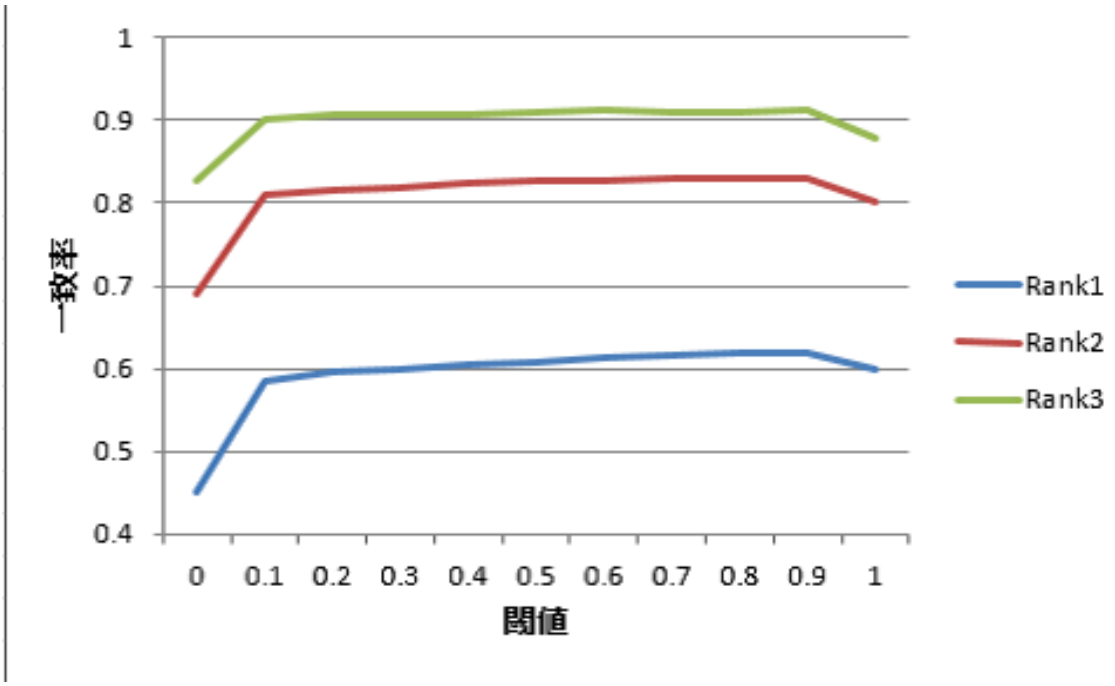

図 4.9. 各閾値における牌譜一致率

価としては平均順位（レーティング）を用いる．レーティング（R）とは平均順位と負の相関を持つ基準である．具体的には（4.10）で計算される．

$$R' = R + (50 - Rank \times 20 + \frac{AveR - R}{40}) \times 0.2 \tag{4.10}$$

$Rank$ は前回のゲームでの順位である．$AveR$ は卓の平均の R である．初期 R は 1500 であり，およそ平均順位が 0.1 下がるごとにレーティングは 100 点ほど上昇する．

レーティングを用いた評価方法としては安定レーティングと保証安定レーティング [59] の二つが提案されている．安定レーティングは今までの順位分布をとり続けた場合のレーティングである．しかし安定レーティングは短い試合数や成績の良い連続した試合を恣意的に抜き出した場合，本来の実力より高い値になる．また人間では実力が向上したのか，それとも恣意的に抜き出したのか，といった二つの事象の差異を見つけ出すのは困難である．したがって成績は恣意的に抜き出されたものとして比較するのが適当である．保証安定レーティングは連続する試合の結果を恣意的に切り出した時の安定レーティングを用いて，安定レーティングの現実的な範囲を保証する安定レーティングのことである．これは異なる試合数の恣意的に良かった結果を取り出しても比較可能な評価方法である．4.6 節ではレーティング，4.7 節では保証安定レーティングを用いている．

表 4.7. 順位分布

プレイヤ	1 位率	2 位率	3 位率	4 位率	平均順位	試合数	レーティング
ベースライン+降り+鳴き	0.236	0.321	0.234	0.209	2.41	402	1651
ベースライン+降り	0.237	0.240	0.259	0.264	2.54	834	1507
ベースライン	0.181	0.216	0.252	0.351	2.77	504	1262

表 4.8. 和了・放銃率

プレイヤ	和了率	放銃率
ベースライン+降り+鳴き	0.275	0.149
ベースライン+降り	0.181	0.144
ベースライン	0.188	0.190

4.6　一人麻雀プレイヤの四人麻雀への適応

4.6.1　対人戦での結果

3.8 節で作成した一人麻雀プレイヤである「ベースラインプレイヤ」と「ベースラインプレイヤ」に降りを追加したプレイヤとさらに鳴きを追加したプレイヤとの 3 つを比較した．結果は表 4.7 に示す．ウェルチの t 検定により，2 つのプレイヤの平均順位は有意水準 1% で差があるとわかった．

和了・放銃率の結果を表 4.8 に示す．和了率は 3.8.1 節で説明したとおり，上手いプレイヤほど高い傾向にある．放銃とは相手の和了牌を切り和了されることである．放銃すると相手の得点をすべて自分が払うことになるため放銃率は原則として低いほうが良い．1 局の中で得られる得点は和了・放銃率に影響し，1 局の繰り返しで最終的な順位が決まる．したがって和了・放銃率は順位分布と大きな関係がある [61].

「ベースライン+降りプレイヤ」は「ベースラインプレイヤ」と比較すると和了率がわずかに低下したが放銃率は大きく低下した.「ベースライン+降りプレイヤ」は和了が困難な場合に他人の和了を察知して手を崩して放銃しないように降りることを目的としていたため，和了率は低下するが放銃率がより低下することが期待されこの結果は期待通りであったといえる．

「ベースライン+降り+鳴きプレイヤ」は「ベースライン+降りプレイヤ」と比較すると放銃率がわずかに上昇したが和了率は大きく上昇した．鳴きを行うと自分の手牌が少なくなるため，放銃率が上昇することが予想されるが，それ以上に鳴きによって和了率が良くなり，これも結果は期待通りであった．

「ベースライン+降り+鳴きプレイヤ」のレーティングは 1651 点になった．これはレーティングが 1500 点の平均プレイヤと比較して強いといえる．具体的な役はあまり知らず，相手の点数状況なども一切，考慮しないにもかかわらず，これほどの実力を得たのは一人麻雀の必要性と降りと鳴きの重要性を示していると考えられる．

4.6.2 考察

「ベースライン+降り+鳴きプレイヤ」の挙動を観察したところ他人の和了を察知して降りた局面についてはおおむね正しく判断していた．その局面は他人の和了が近そうで自分の和了が遠く，残りの順目もない場合である．

問題は著者が降りるべきと判断した局面で「ベースライン+降り+鳴きプレイヤ」が降りないことが多いことである．それは自分の手の向聴数は確かに小さいが受け入れ枚数が少ない等，あまり手牌がよくない場合である．原因は自分の手牌の良さを向聴数とドラの数でしか判断していないためである．向聴数は手牌の良し悪しを判断する基準としては重要であるが手牌のすべてを判断することはできない．この原因は降りに関して学習した局面が少なく，手牌の情報を入れても予備実験の結果があまり良くなかったため，向聴数程度しか判断基準を入れられなかったためである．

さらに降りの精度を高くするには相手の和了牌を読む技術が必要になる．今の手法では牌の安全度を 4 段階でしか判別していないが，今後は相手の捨て牌などから，それぞれの牌の安全度を予測することが考えられる．例としては，相手の副露から役を読み，安全牌を理解する技術である．染め手といった役は比較的読みやすい．また牌の安全度がルールで書いたそれとは大きく異なる．リーチに対しても一般的な上級者は無筋の牌の中に順位があり，その中で降りるか否かを判断しているため，ルールをより複雑にする方法が考えられる．

一人麻雀としては，平均的人間プレイヤの和了率を上回っているものの，まだ悪手を選ぶこともある．例えば対子の価値である．対子は受け入れ枚数を 2 枚増やし，和了にも必ず必要であるため価値を判断するのは困難である．そのため対子を崩すという選択をあまりしない．その結果，対人戦での七対子の和了率が 3.3% と平均の 1.7% [61] の倍近い値になっている．まだ一人麻雀として上級者レベルにはなっていないため，特徴量を改良する必要がある．

鳴いた局面についてはおおむね正しかった．問題は教師データには和了した局面が多いため，自分の和了が困難な時に悪手となる鳴きを行うことである．教師データに和了をしていない局面を追加することで解決できる．

4.7 結果

4 章で構築したプレイヤと「ベースライン+降り+鳴きプレイヤ」をほかの麻雀プログラムであるまったり麻雀 [23] とインターネット上の麻雀対戦サイトである天鳳 [69] で対戦させた．ここでは 4 章で構築したプレイヤを “モンテカルロプレイヤ” と呼ぶ．

4.7.1 まったり麻雀との対戦における評価設定

実力を測定するため，まったり麻雀 [23] と対戦を行った．まったり麻雀 [23] は統計量をヒューリスティックに組み合わせて行動選択の評価関数を作成している．その実力は筆者の知る限り現存する麻雀プログラムの中で一番高い．

表 4.9. 対まったり麻雀での順位分布

	1 位率	2 位率	3 位率	4 位率	平均順位
まったり麻雀	0.248	0.247	0.250	0.255	2.51
モンテカルロプレイヤ	0.230	0.248	0.268	0.254	2.54
ベースライン+降り+鳴きプレイヤ	0.243	0.226	0.222	0.309	2.59

表 4.10. 対まったり麻雀での和了・放銃率

	和了率	放銃率
まったり麻雀	0.200	0.122
モンテカルロプレイヤ	0.201	0.121
ベースライン+降り+鳴きプレイヤ	0.228	0.178

ルールは東風戦，赤あり，ゲーム自体に制限時間はないが，一手 1 秒で行った．対戦相手の内訳はまったり麻雀が 3 人と筆者のプログラムが 1 人である．対戦方法としてまったり麻雀の対戦モードの中のデュプリケートモードを用いた．これは最初に山を作成するときに使用する乱数を選択できるモードで，これにより配牌やツモについての運の要素をなくすことができる．またオプションにより同じ乱数でまったり麻雀自体が打った成績と比較することができる．乱数は 0〜999 を用いて 1,000 試合を行った．評価は平均順位を用いる．

4.7.2 まったり麻雀との対戦における結果

モンテカルロプレイヤと 4.6 節で最も成績の良かった「ベースライン+降り+鳴きプレイヤ」とまったり麻雀 [23] の 3 つの比較を行った．結果を表 4.9 に示す．一番良い結果であるモンテカルロプレイヤでさえ，まったり麻雀に勝ち越すことやまったり麻雀自体が出した成績を超えることはできなかったが,「ベースライン+降り+鳴きプレイヤ」よりも 4 位率が大きく改善され，平均順位が良くなっており実力が向上したといえる．

和了・放銃率を 表 4.10 に示す．モンテカルロプレイヤはまったり麻雀と同じ成績を出すことができた．しかしながら平均順位で負けているのは今の順位や得点によって戦略を変更できない方法にあると考えられる．

4.7.3 天鳳での対戦における結果

モンテカルロプレイヤと「ベースライン+降り+鳴きプレイヤ」の 2 つの比較を行った．対戦結果を表 4.11 に示す．安定レーティングは大きく変わらないが，保障安定レーティングの向上が見られる．

和了・放銃率は 表 4.12 に示す．結果としてよくなったとは言えないが，モンテカルロプレイヤの和了率と放銃率はより人間の平均値に近い値を出した．

表 4.11. 順位分布

	1 位率	2 位率	3 位率	4 位率	平均順位	試合数	安定レーティング	保障安定レーティング
モンテカルロプレイヤ	0.231	0.269	0.268	0.233	2.50	2227	1681	1690
ベースライン+降り+鳴きプレイヤ	0.253	0.248	0.251	0.248	2.49	1441	1689	1610

表 4.12. 和了・放銃率

	和了率	放銃率
モンテカルロプレイヤ	0.237	0.127
ベースライン+降り+鳴きプレイヤ	0.256	0.148

4.7.4 考察

モンテカルロプレイヤの挙動を見る限り，自分の和了が遠く相手がリーチしている局においては降りになる牌を選択することが多い．また回し打ちの挙動も見られるなど人間に近い挙動を確認した．

まったり麻雀 [23] を相手にしたときでは成績が向上したのに対して，天鳳 [69] で対人戦を行ったときはあまり成績が向上しなかった．まったり麻雀 [23] の牌効率が上級卓のプレイヤより上手い．そのため対人戦ではまったり麻雀と対局するときに比べ相手が聴牌している割合が低くなる．モンテカルロプレイヤは相手の聴牌率を閾値にして手を決定しているため，聴牌率が低いと手はベースライン+降り+鳴きプレイヤもモンテカルロプレイヤも一人麻雀の手を返すだけである．そのため打牌アルゴリズムが同じ局面が増え成績が向上しなかったと考えられる．

問題として，モンテカルロ法を用いる閾値を相手の聴牌率のみで判定していたため，相手が副露して高い手を作っている局面において，一人麻雀の手を選択し放銃することがある．これを解決するためには一人麻雀の評価値と予測した危険度をもとに学習を行い，手を決定するモデルを構築する必要がある．

降りが上手くなったため，役の無い鳴きや牌効率の悪い手といった一人麻雀の手の粗さも目立つ．相手の聴牌率や降りのデータに一人麻雀の手を使用しているため，一人麻雀の手の改善は全体の改善につながる．また今は面前で聴牌したらすべてリーチやリーチ後の可能な牌はすべて暗槓といったヒューリスティックな部分も残っており，より精度の高い打牌をするためにリーチを打つか打たないかといった学習も行わなければならない．これらの解決のため一人麻雀の特徴量やモデルについても見直す必要がある．

4.8 おわりに

本研究では，不完全情報多人数ゲームの一つである麻雀において，多人数性を排除した一人麻雀から四人麻雀に適応する方法を提案し，評価を行った．提案された手法は二つあり，一つは一人麻雀から四人麻雀への機能の拡張，二つ目は四人麻雀を相手をモデル化した一人麻雀にする方法である．

3 章で麻雀の多人数を削除した一人麻雀を考え，それと四人麻雀の差を解析し，その差を埋めるように一人麻雀プレイヤを拡張する手法を提案した．作成したプレイヤを天鳳 [69] において 402 試合戦わせた結果，レーティングが 1651 点と平均的人間プレイヤよりも強いことが確認された．この結果から，一人麻雀から四人麻雀へ拡張していく方法は適切であったといえる．また麻雀に必要な技術は降りと鳴きであることも示すことが出来た．

4 章では相手の抽象化したモデル化を考え，牌譜をもとに学習を行い，その予測結果とモンテカルロ法を用いることでコンピュータ麻雀プレイヤの手を決定するアルゴリズムを提案した．各抽象化した要素の予測は人間と比較しても精度が高く，高速に行うことが可能であった．3 章で提案した方法は，牌譜に人手で降りかどうかのタグをつける必要があり，回し打ちが出来ない問題もあったがこれにより，二つの問題が改善した．実際の手を決定する局面においても今までの行われていた，降りるべき局面のタグを一切使うことなく降りと攻めのバランスを人間並みに行うことが可能になった．

結果として天鳳 [69] において 2,227 試合，戦わせた結果，安定レーティングが 1681 点という上級卓の平均並みの実力を得た．今までの先行研究では平均的人間を上回る実力のコンピュータプレイヤが報告されていないことを考慮すれば大きな進歩である．

4.9 現状の問題点

今のプレイヤは一局の報酬を最大にすることを目標としている．しかし最終的なゲームの目的は相手より点数を多く持つことであるため，この目標では目的を達成できない．一ゲームを個別の一局の集まりではなく，一局をゲーム全体においてどのように位置づけするかという大局観が必要になる．

今後の課題としては現在の得点状況からに応じて手を選択するモデルの構築である．今のモデルは現在の点数状況が入っておらず，どのような状況であっても同じ手を指す．当然，上級者は今の点数状況に応じて，何点の和了をすべきか考えており，またリーチなどにどれだけ攻めるか降りるかを判定している．これらの判断は最終的な順位に大きく影響するため実力向上に必須である．これらの判断にはシミュレーション中の報酬をゲームの得点では無く，順位に影響する何らかの値にする必要がある．例えば現在の点数状況からゲーム終了時の順位分布を予測が可能になれば，期待順位を計算しシミュレーションの報酬とすることが可能になる．

今のプレイヤは面前で聴牌するとすべてリーチを打つようなヒューリスティックな判断をしている．しかし最終局の一位で役がある手牌などはセオリーとしてリーチを打つ必要がない状況は多い．そして最終局のリーチは順位を決定する大きな要因でもある．得点状況からの適切な手の選択を実現するために一人麻雀の手にリーチを打つかどうかという判断を入れる必要がある．実現する方法としては一人麻雀の手の学習に使う牌譜を変更する必要がある．今はリーチを打った局面までを教師としていたが，それではリーチを打ったプレイヤしか学習できない．そこで面前で聴牌したプレイヤに限定することでリーチをしなかった局面を学習することが出来る．

鳴きに関しても降りと同様に成果を上げられた．次は点数状況に合わせた手作りである．点数状況によっては，鳴いて安い手を和了することは悪手になることがある．例えば最終局かつ最下位時に安い手で和了し最下位が確定してしまう場合である．また最終局かつ一位の時に安い手で和了す

るのが最善手だが高く遅い手を狙ってしまい，相手に逆転を許す場合もある．今のプレイヤはこのような悪手を指すことがよくある．

最善手を考慮するにあたって点数状況が絡む局面は当然最終局が多い．しかしながら最終局を有利な局面にするためにもその前から点数状況を考えなければならず，原理的には最初の局から点数状況を考慮した手が必要になる．

第5章　期待最終順位に基づく コンピュータ麻雀プレイヤの構築

5.1　はじめに

前章では麻雀の不完全情報ゲームという点に着目し，相手をモデル化することでこれに対応した．次に強化学習で用いる疑似的な報酬を設計する方法を述べる．また強化学習のベースラインとなるこの疑似的な報酬を用いて手を決定するコンピュータ麻雀プレイヤも構築する．

ゲームの中には「繰り返しゲーム」と呼ばれるゲームが存在する．これは同一の部分ゲームを繰り返し行うことで，最終的な勝者が決まるようなゲームである．具体例としてゲームの分野ではトーナメントポーカー，バックギャモンなどが繰り返しゲームであり，スポーツの分野でも野球やカーリングが例として挙げられる．このように，繰り返しゲームはボードゲームやスポーツなどにも幅広く存在し，エキスパートプレイヤは，部分ゲームの勝率を最大化するのではなく，長期的な戦略を持ってこれらのゲームをプレイしている．そのような長期的な戦略をコンピュータで実現することは，人工知能の研究においても有意義だと考えられる．

繰り返しゲームにおける戦略決定に関する研究として，Miltersen らはトーナメントポーカーにおいてプレイヤの取り得る行動を削減し，チップの状態数をまとめることで近似的な最適戦略を求めた [27]．Papahristou らはバックギャモンの自己対戦の結果からゲーム開始時の各点数状況における全体の勝率表を作成した [39]．これによりマッチプレイにおいて，部分ゲームの期待値を最大化するプレイヤに対して勝率を向上させた．

麻雀は 1 局ごとに役に応じた点数を獲得し，全部で 4 または 8 局行う．全ての局が終了した時点で最終的に最も多くの点を持っているプレイヤが麻雀の勝者である．この性質から麻雀は繰り返しゲームであるといえる．繰り返しゲームとして麻雀を対象にした研究は我々の知る限り存在しない．

本論文のテーマである報酬が疎という点に着目すると，麻雀の報酬というのは順位であるため全ての局が終了しないと報酬が手に入らない．これはほとんどの局面に対して報酬が得られない環境といえる．これは 2.2 節で定義した報酬が疎な環境といえる．同様に麻雀を 1 局単位としてみても，情報集合数が膨大ではあるものの点数を得られる状態は少なく報酬が疎な環境といえる．

麻雀を対象にした研究として，水上らは相手のモデル化とシミュレーションによる麻雀プレイヤを構築した [28]．モデル化では、相手の手牌を完全に予測しようとするのではなく，ゲームに大きく影響する部分のみを考慮している．具体的には (a) 聴牌である確率，(b) 特定の牌が相手への当たり牌になる確率，(c) 予想される相手の和了点数の 3 つを考え，これらの予測モデルを牌譜から学習している．学習された相手モデルを用いて現在の局面から 1 局終了までをシミュレーションすることで

表 5.1. 囚人のジレンマ利得表

	協力	裏切り
協力（C）	(3,3)	(0,5)
裏切り (D)	(5,0)	(1,1)

局単位での最適な手を決定している．その結果，局単位の収支を増加させることに成功し，それに伴い人間との対戦成績も向上した．しかしながらこのプレイヤが明らかに悪い手を選択する状況も存在する．具体例として 2 点挙げる．ひとつは，現在の順位と残りの局数に応じた押し引きができないことである．たとえば、現在プレイヤが 1 位であり，2 位以下を大きく引き離している場合であっても，放銃する危険を冒して和了を目指してしまう．もうひとつは，最終局において適切な点数で和了できないことである．すなわち．最終局において，和了しても最下位が確定するような得点の低い手を作ってしまう．この 2 つの問題点の原因は，最終順位を考慮せずに手を選択していることにある．

この問題を解決するひとつの方法はゲーム全体を通した最適戦略を計算することである．しかし，麻雀は情報集合の要素数が非常に多く，局ごとの最適戦略を計算することすら計算量的に困難である．そこで本章では，局終了時の各プレイヤの得点状況から最終的な順位を予測し，1 局において最終順位を考慮した手を選択するプレイヤを構築する手法を提案する．

また本章の本論文での全体的な位置付けを述べる．報酬が疎なゲームである麻雀を 1 ゲームの報酬をもとに強化学習することは困難である．そこで同じ報酬が疎なゲームではあるが全体に比べ小さいゲームである 1 局単位を強化学習させたい．そのためには 1 局単位の報酬である期待最終順位をもとに手を決定することで 1 ゲームが強くなることを示す．

本章は以下の構成になっている．初めに 5.2 章で関連研究を述べる．提案手法として，5.3 章で現在の点数状況からの期待最終順位予測，5.4 章で予測モデルとモンテカルロ法を用いた手の決定，5.5 章で提案手法の対戦結果について述べる．最後に 5.7 章で本章の結論について述べる．

5.2 関連研究

5.2.1 囚人のジレンマ

囚人のジレンマは繰り返しゲームとして知られ Merrill と Melvin が 1950 年に考案し，Tucker によって定式化が行われた [54]．囚人のジレンマを無限回繰り返すゲームに関しては，古くから最適戦略が知られていたが，誰も証明をつけなかったためフォーク定理と呼ばれてた．その後 Rubinstein が証明を行った [42]．

囚人のジレンマとは，2 人のプレイヤが独立に協力（C）または裏切り（D）のどちらかを選択し，2 人の選択の組合せによって各人の利得が決定するゲームである．表 5.1 に利得表の例を示す．両プレイヤが協力した場合は，両プレイヤともに 3 点を得る．両プレイヤが裏切った場合は，両プレイヤともに 1 点を得る．片方が協力し，片方が裏切った場合，裏切ったプレイヤが 5 点，協力したプレイヤが 0 点を得る．

5.2.1.1 有限回繰り返し囚人のジレンマ

この囚人のジレンマを 1 回だけ行う場合の最適な戦略を考える．最適な戦略を考えるうえでナッシュ均衡戦略 [34] と呼ばれる戦略を説明する．これは特定のプレイヤが今と違う戦略をとったとしても，そのプレイヤの利得が増えない戦略を指す．これはこの戦略はどのようなゲームにも混合戦略まで考えればナッシュ均衡は存在することが知られている．ここで混合戦略について説明する．これは戦略を確率で表現する戦略である．

1 回行う場合では裏切りがナッシュ均衡戦略ではあったが，このゲームを有限回繰り返す場合について考えてみる．この有限の回数を両プレイヤが既知であるとする．単純に解析する場合，すべての回数ごとに戦略を列挙し，それぞれの戦略に対応する利得表からナッシュ均衡戦略を求める方法である．しかしながらその場合，利得表が回数のべき乗で増大し，線形計画法で解くには限界がある．

ここでは後退帰納法を用いて，ナッシュ均衡戦略を求めることにする．N 回ゲームを行う時，N 回目は $N-1$ 回目の結果に影響を与えることが出来ない．したがって N 回目の最善の戦略は囚人のジレンマを 1 回だけ行うことに等しい．すなわち裏切りが選択される．これをもとに $N-1$ 回目の最善の戦略を考えると，N 回目の行動は裏切りで確定しているため，$N-1$ 回目の行動は N 回目においても影響はない．したがって囚人のジレンマを 1 回行うことに等しい．つまり裏切りが選択される．同様の議論を繰り返し，有限繰り返し囚人のジレンマのナッシュ均衡戦略は 1 回目から N 回目まですべて裏切ることである．

5.2.1.2 無限回繰り返し囚人のジレンマ

次に無限回繰り返し囚人のジレンマの最適な戦略について述べる．無限回での戦略を考慮するうえで困難なことは利得の計算である．どのような戦略であっても，囚人のジレンマを無限回行うことで，総利得は無限大になる．これでは，どのような戦略が有効なのか判断することが出来ない．そこで割引因子という考えを用いる．これは 1 ゲーム終了時に確率 p でもう一度ゲームを行い，$1-p$ の確率でゲームが終了するという考え方である．これにより利得が無限大に発散せず，戦略の比較が可能になる．

無限回繰り返し囚人のジレンマを解くにあたって天下り的ではあるが，トリガー戦術というもの考える．この戦術は二つのルールに基づいて手を決定する．

- 初回は協力をする．
- 2 回目以降は相手が裏切りがあった場合，裏切り続ける．

この戦略がナッシュ均衡戦略であることを示す．お互いがこのトリガー戦術を行った場合の利得を計算する．

$$3 + 3p + 3p^2 + \cdots = \frac{3}{1-p} \tag{5.1}$$

次にプレイヤの片方だけが最初から裏切ったとする．この場合の利得は

$$5 + p + p^2 + \cdots = 4 + \frac{1}{1-p} \tag{5.2}$$

すなわち式（5.1）が式（5.2）よりも大きな条件は $p > \frac{1}{2}$ である．p は無限に繰り返すことを想定しているため，十分に大きい．そのため裏切ることで利益を増やすことが出来ない．以上の議論からトリガー戦術はナッシュ均衡戦略であり，無限回繰り返し囚人のジレンマの最適な戦略は協力することである．

5.2.2 ポーカー

ポーカーを対象にした研究として Miltersen らはトーナメントポーカーにおいて近似の最適な戦略を求めた [27]．トーナメントポーカーとは相手のチップの数が 0 になるまでゲームを行うポーカーの一種である．そのため 1 ゲーム毎の期待値を最大化する「キャッシュゲーム」と異なり，トーナメントポーカーでは，現在のチップの数に応じた戦略が必要になる．この研究では，プレイヤの行動を削減するため Jam/Fold と呼ばれるプレイヤの行動をオールインするか降りるかどちらかしか行わないプレイヤを考え，チップの数を 50 刻みで抽象化することでチップの状態の削減も行った．この抽象化したゲームを解くことで，ゲーム全体の近似最適戦略を求めることに成功した．

5.2.3 バックギャモン

Papahristou らはバックギャモンのマッチプレイにおける全体の勝率表を作成した [39]．バックギャモンは勝利した条件によって 1 点または 2 点が与えられる．マッチプレイとはあらかじめ全体の勝利までの点数を設定し，1 ゲームを繰り返すことで先にその点数を満たしたプレイヤが最終的な勝者となる遊び方である．この研究では 1 ゲームを繰り返す強化学習によりバックギャモンプレイヤを構築した．

マッチプレイにおいて重要な概念は Match Winning Chance（mwc）である．これはマッチプレイの最終的な勝率を指す．これを最大になるような差し手がこのマッチプレイにおいても最善手であるといえる．この mwc をもとめるため，各開始局面の各プレイヤの持ち点に対して式（5.3）から勝率を求める．

$$\begin{aligned} mwc(\mathrm{A}, \mathrm{B}) = & S \times mwc(A-1, B) + \\ & D \times mwc(A-2, B) + \\ & S \times mwc(A, B-1) + \\ & D \times mwc(A, B-2) \end{aligned} \tag{5.3}$$

ここで A と B は各プレイヤの持ち点である．また $A = B$ の時は勝率は 0.5 とする．S, D はゲーム開始時にシングル勝ちとダブル勝ちが起こる確率である．その確率はモンテカルロ法を用いて決定する．ゲーム AI ではモンテカルロ法はランダム要素を混ぜてゲームを何度も行うことで今の局面の

勝率を推定する方法である．今回のケースではサイコロの出目がランダム要素になり，何度も自己対戦を行うことで，S, D を推定する．

得られた mwc を利用して，最終的な手を式（5.4）で決定する．

$$\begin{aligned} MWC =& WS \times mwc(A-1, B)+ \\ & WD \times mwc(A-2, B)+ \\ & LS \times mwc(A, B-1)+ \\ & LD \times mwc(A, B-2) \end{aligned} \tag{5.4}$$

結果的にこれによりマッチプレイにおいて，1 ゲームの期待値を最大化するプレイヤに対してマッチプレイの勝率を向上させた．

5.2.4 麻雀

とつげき東北は，現在の得点状況，和了率，放銃率等のパラメータからシミュレーションを用いて最終的な順位分布を予測する MJSIM0 を開発した [60]．このシミュレーションでは，プレイヤは現在の得点に依存しない確率に基づき，和了や放銃を行う．そのため予想される成績は，実際のものとはずれたものとなる．またシミュレーションに時間がかかるため既存の麻雀プログラムに利用するのは困難である．

麻雀ではポーカーと異なり手を効率的に抽象化する方法は提案されておらず，抽象化したゲームとして解くことは現在のところ不可能である．また，点数状況がバックギャモンに比べ膨大であるため，あらかじめ勝率表を持つことも不可能である．そこで本研究では現在の得点状況から最終順位の予測を行うモデルを牌譜から機械学習する．そしてその予測モデルを用いて最終順位を考慮したプレイヤを構築する．

5.3 順位予測

この章では現在の点数状況から最終順位を予測するモデルを学習する．この予測モデルを用いて手を決定するため，高い予測精度が求められる．

5.3.1 順位予測モデルの学習

麻雀の勝負において重要なのは最終得点ではなく最終順位である．最終順位を予測する方法としては，回帰で直接推定する方法と，各順位をとる確率を一旦求め，その確率を用いて期待順位を求める方法が考えらえる．今後の研究の展開として各順位の報酬が線形でない場合も想定しているため，

本研究では後者の方法をとる．具体的には，最終順位を予測するモデルとして多クラスロジスティック回帰モデルを使用することで 1 位から 4 位を予測する多クラス問題として捉える．出力としてソフトマックス関数を使用することで，1 位から 4 位のそれぞれの確率を出力し，現在の得点状況から期待される最終順位を推定する．ソフトマックス関数は次の式（5.5）で表される．

$$P(rank = r) = \frac{\exp\left(\mathbf{w_r}^{\mathrm{T}}\mathbf{x}\right)}{\sum_{i=1}^{4}\exp\left(\mathbf{w_i}^{\mathrm{T}}\mathbf{x}\right)}, \tag{5.5}$$

ここで $\mathbf{x}$ は現在の点数状況を表す特徴ベクトルである．$\mathbf{w_r}$ は順位 r の特徴量に対しての重みベクトルである．この式を利用して期待最終順位（Expected Final Rank，EFR）は次の式（5.6）で表される．

$$EFR(\mathbf{z}) = \sum_{r=1}^{4} r \times P(rank = r|\mathbf{z}) \tag{5.6}$$

ここで $\mathbf{z}$ は点数状況を表す．

学習に使用する特徴量を表 5.2 に示す．表中の自分の順位，相手の順位，残り局数，東風か半荘と点差に関して以下に述べる．前半部分はこの特徴量の次元を表し，点差とは特定の相手との点数を差を全てのプレイヤの持ち点の合計である 100,000 で割って正規化した値である．これにより，点差を 0 以上 1 以下の実数で表現でき，その値を特徴量の値とする．

このように相手との点差をそのまま特徴量として使用することも当然可能であるが，麻雀では和了点数が固定されているため，点差と期待順位に単純な線形相関はない．そこで意味のある点差を考慮する．意味のある点差とは逆転に必要な和了の点数を和了の種類によって点差を抽象化することである．ここでは和了の種類を出和了，ツモ和了，直撃，他者のツモ和了，流局の 5 種類とする．出和了とは特定の相手以外からのロン和了である．直撃とは特定の相手からのロン和了である．麻雀には子のロン和了でも 30 種類ほど和了点が存在するが，ここでは頻出の 10 種類ほどを用いている．例えば，相手と 9,700 点差と 9,900 点差と 10,100 点差は満貫を出和了しても逆転しない点差であるが，満貫をツモると 9,700 点差と 9,900 点差だけ逆転する．意味のある点差を考えることで 9,700 点差と 9,900 点差を同等に扱うことが出来る．

本場が 1 つ増えると出和了の場合は，300 点分追加で差が縮み，ツモの場合は，400 点分，追加で差が縮む．このように単純に一つの基準で意味のある点差を考えるとリーチ棒や本場に対応できない．しかしながら和了の種類を分けることでリーチ棒や本場の影響も考慮した点差を考えることが出来る．これにより，リーチ棒や本場といった要素を考慮した特徴量が生成可能になり，学習局面の少ないであろうリーチ棒や本場が多い局面であっても，単純にこれらを特徴量に加えるよりも精度の高い予測が可能になると考えられる．特徴ベクトルの次元は 8,161 になった．

予測モデルの学習は，以下の式（5.7）で表現される目的関数 L_{rank} を最小化することにより行った．

$$L_{rank}(\mathbf{w}) = -\sum_{i=1}^{N}\sum_{r=1}^{4} c_{i,r}P(\mathbf{X_i}) + \frac{\lambda|\mathbf{w}|^{\mathbf{2}}}{N}, \tag{5.7}$$

ここで N はトレーニングデータのサンプル数, $\mathbf{X_i}$ は i 番目のトレーニングデータ, c_i は 4 次元のベクトルで，各順位に対応する 2 値（1 または 0）のデータである．λ はトレーニングデータに過学習することを防ぐ正則化項である．ここでは正則化項 λ を 0.01 とする．

表 5.2. 順位予測の特徴量

特徴量	次元数
自分の順位，相手の順位，残り局数，東風か半荘と点差	$4 \times 4 \times 8 \times 2 = 256$
自分の順位，残り局数，残りの親の回数，サドンデスか，東風か半荘	$4 \times 8 \times 3 \times 2 \times 2 = 384$
自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），出和了の点差	$4 \times 4 \times 5 \times 11 = 880$
自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），ツモ和了の点差	$4 \times 4 \times 5 \times 11 = 880$
自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），直撃の点差	$4 \times 4 \times 5 \times 11 = 880$
自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），他者によるツモ和了の点差	$4 \times 4 \times 5 \times 12 = 960$
自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），流局の点差	$4 \times 4 \times 5 \times 5 = 400$
自分の順位，相手 2 人の順位，残り局数（0，1，2，3，4 以上），相手 2 人の出和了の点差	$4 \times 4 \times 4 \times 5 \times 11 = 3520$
バイアス項	1

t

表 5.3. 点差表

出和了	子	0, 1000, 1300, 2000, 2600, 3900, 5200, 7700, 8000, 12000, それ以上
	親	0, 1500, 2000, 2900, 3900, 5800, 7700, 11600, 12000, 18000, それ以上
ツモ和了	子対子	0, 1400, 1900, 2500, 3400, 5000, 6500, 9900, 10000, 15000, それ以上
	相手が親	0, 1600, 2200, 3000, 4000, 6000, 7800, 11900, 12000, 18000, それ以上
	自分が親	0, 2000, 2800, 4000, 5200, 8000, 10800, 15600, 16000, 24000, それ以上
直撃	子	0, 2000, 2600, 4000, 5200, 7800, 10400, 15400, 16000, 24000, それ以上
	親	0, 3000, 4000, 5800, 7800, 11600, 15400, 23200, 24000, 36000, それ以上
他者のツモ和了	子対子	0
	子対親	200, 300, 500, 600, 1000, 1300, 1900, 2000, 3000, それ以上
流局		0, 3000, 4000, それ以上

重みベクトルの学習に FOBOS [20] を使用する．また重みベクトルの更新は Adagrad [19] を使用する．学習率 η を 0.01 とする．

5.3.2 学習に用いる牌譜

上記の聴牌予測には多くの教師データが必要になる．教師データとしてはインターネット麻雀サイト天鳳 [69] の鳳凰卓の牌譜を用いた[1]．鳳凰卓でプレイできるのは全プレイヤの中でも上位 0.1% 程度であり牌譜の質は高いと考えられる．以下，牌譜と示した場合はこの鳳凰卓の牌譜のことを指す．

牌譜中の局開始時の点数状況から特徴量を抽出し最終順位を組み合わせて教師データとした．局面数はおよそ 2.5×10^7 である．

[1] 2009 年 2 月 20 日から 2013 年 12 月 31 日までに行われた対局

表 5.4. 順位予測に関する評価

プレイヤ	平均絶対誤差
ベースライン	0.809
予測モデル	0.763
-自分の順位，相手の順位，残り局数，東風か半荘と点差	0.764
-自分の順位，残り局数，残りの親の回数，サドンデスか，東風か半荘	0.764
-自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），和了の点差	0.765
-自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），ツモ和了の点差	0.764
-自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），直撃の点差	0.764
-自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），他者によるツモ和了の点差	0.764
-自分の順位，相手の順位，残り局数（0，1，2，3，4 以上），流局の点差	0.764
-自分の順位，相手 2 人の順位，残り局数（0，1，2，3，4 以上），相手 2 人の出和了の点差	0.764

5.3.3 聴牌予測の精度

学習が上手くできているかを調べるため，学習によって得られた順位予測モデルのの精度を調べた．評価は平均絶対誤差を用いる．テストデータは 95,856 局面を用いた．

テストデータに対しての結果を表 5.4 に示す．ベースラインは現在の順位を予想順位として返す．予測モデルはベースラインに比べ最終順位の予測性能が高い．結果の 3 行目以降は予測モデルから特定の特徴量を用いない場合の結果である．最も重要と思われる純粋な点差を除いたしても結果は大きく悪くならないことがわかる．どの特徴量も除くと結果は悪くなるため，すべての特徴量が有効であったといえる．

5.4 モンテカルロ法を用いた手の決定

この章では，プログラムが順位予測モデルを使ってどのように手を決定するかを述べる．この研究で用いるシミュレーションは，水上らが用いたシミュレーション [28] とほぼ同様であるが，相違点について以下について述べる．シミュレーションの報酬が先行研究ではゲーム上の得点であったが，今回はその得点から得られる次の局開始時の点数状況からの期待順位になるのが主な変更点である．シミュレーションによって決まる手のことを**モンテカルロの手**と呼ぶ

モンテカルロの手を用いることは序盤であっても可能であるが，和了まで時間がかかるため，精度の高い手を選択することは難しい．そのため序盤は一人麻雀の手を用いる．一人麻雀の手からモンテカルロの手に切り替えには以下の条件のいずれかを満たしたときである．

- 誰かがリーチをかけた時
- ツモが可能な牌の数が 16 枚以下
- $\frac{\sum_{p \in opponents} EL(p,Tile)}{fold\,value} \leqq 0.2$

3 つ目の条件は先行研究で使われた条件と同じである．$EL(p, Tile)$ は p に対して牌 $Tile$ を切った時の期待失点であり，$fold\,value$ は最初から降りた時の期待値である．

モンテカルロの手は次の式によって求められる．

$$\begin{aligned} & Score(Tile) \\ & = Sim(Tile) \times \prod_{p}(1 - LP(p, Tile)) + \sum_{p} LP(p, Tile) \\ & \times Max(EFR(\tilde{\mathbf{z}}), EFR(\mathbf{z})), \end{aligned} \tag{5.8}$$

ここで $Sim(Tile)$ はシミュレーションの結果であり，ここでは EFR で求められる期待最終順位である．$LP(p, Tile)$ は相手 p に対して牌 $Tile$ を切った時に放銃する確率，$HS(p, tile)$ は相手の予測点数である．$\tilde{\mathbf{z}}$ は現在の得点状況 $\mathbf{z}$ から放銃した時の点数状況である．すなわち $\tilde{\mathbf{z}}$ は手番プレイヤの点数から $HS(p, tile)$ を引きプレイヤ p に $HS(p, tile)$ 足した点数状況である．$EFR((\mathbf{z})$ は現在の期待最終順位であるから式の後半は振り込んだ後の点数状況が現在の期待最終順位よりも良くなることないという制約である．

この式はある牌を切った時の期待順位を表している．したがって $Score(Tile)$ は 1 以上 4 以下の値である．この計算をプレイヤの持っているすべての牌に対して計算し，最も小さい値すなわち期待順位が最も良い牌が切られる．

5.4.1 シミュレーションの中の挙動

$Sim(Tile)$ の中身について自分の手番での動作を図 5.1 に示す．先行研究と異なる点はシミュレーション中のリーフ（和了，降り，放銃）での報酬がゲーム上の値でなく，期待最終順位に変更される点である．$ODEV(One\text{-}Depth\,Expected\,Value)$ とは自分の手番から次の手番までの一手探索した結果である．この結果に基づいて降るかどうか判定する．当然この $ODEV$ の判定も期待最終順位に基づいて降りかどうかを決定する．

5.4.2 降り成功率

先行研究 [28] ではシミュレーション中に降りる場合，降りはどんな手牌であっても必ず成功するとしていた．そのため鳴きを多用し流局時の聴牌料を狙う挙動がみられた．実際には鳴いた後に放銃することが多く，鳴いた手は悪手となることが多い．悪手となる鳴きを防ぐため，今回はシミュレーション中に降りる場合には，降りの成功率を考慮する．具体的には現在の相手の待ち牌率を利用して近似の降り成功率を計算する．

降り成功率を求めるアルゴリズムを Alg 5.1 に示す．基本的に，最初にプログラムの手牌に流局までの手番の数だけ牌をツモると考える．プログラムから見えていない牌を数え，各牌のツモる確率を計算する．その確率に手番数を掛け合わせた牌を仮想的にツモ牌としてプログラムの手牌に加える．したがって手牌の数は整数でなく実数になる．次に相手の待ち牌率の予測結果に基づき安全な

表 5.5. 一致率

	Rank1	Rank2	Rank3
提案手法	0.619	0.827	0.908
降り成功率導入	0.621	0.829	0.909
水上 [28]	0.620	0.834	0.915

順から牌を切る．相手は確率に基づきツモやほかのプレイヤに放銃する．最終的にプログラムが放銃しない確率を降り成功率とする．

この降り成功率は，図 5.1 のリーフの降りの値を変更することで利用する．先行研究 [28] ではこの値はプログラムがツモの後は何も切らないという仮定を置いた時，すなわち相手のツモと流局時の失点の平均値を使用していた．今回は相手のツモ，流局，相手の放銃，自分の放銃それぞれの局面の期待最終順位をもとめ，確率を掛け合わせた値とする．

5.5 結果

提案手法によって得られた麻雀プログラムと牌譜との一致率を調べた．またインターネット上の麻雀対戦サイトである天鳳 [69] で対戦させた．比較となるプレイヤは水上 [28] と，このプログラムに降り成功率を導入した「降り成功率導入」と，さらに期待最終順位に基づく「提案手法」の 3 つである．実験で使用した CPU は Core i7 3632 QM でクロック数は 2.2GHz，8 コアを使用してシミュレーション部分を並列に計算した．提案手法のプログラムは一手 1.5 秒で行った．

5.5.1 牌譜との一致率

実力を測定するため牌譜との一致率を調べた．テストとして同じ牌譜中のツモ局面を 10,000 局面を使用した．結果を図 5.5 に示す．$Rank\,n$ は第 n 候補に牌譜での打牌が入っている割合とする．結果として牌譜一致率の向上は見られなかった．

5.5.2 天鳳での対戦における評価設定

実力を測定するため，インターネット上の麻雀対戦サイトである天鳳 [69] で対戦を行った．ルールは東風戦，赤あり，持ち時間は一手 3 秒に考慮時間が 5 秒である．鳴きについても同様の持ち時間があり，鳴ける局面では次のプレイヤは勝手に牌を引くことができない．天鳳 [69] では成績に応じて対戦できる卓が異なる．卓は 4 種類あり，上から鳳凰卓，特上卓，上卓，一般卓がある．卓の種類は上の卓を選択可能な成績であっても上級卓のみに限定した．プログラムをサイト上で対戦させるための入出力インターフェースは自作した．評価としては平均順位（レーティング）を用いる．レーティング（R）とは平均順位と負の相関を持つ基準である．具体的には（5.9）で計算される．

Algorithm 5.1 降り成功率

```
function 降り成功率
    turn                                                          ▷ 残り順目
    tile                                                          ▷ 自分から見える牌
    hand                                                          ▷ 自分の手牌
    winning_tile                                                  ▷ 待ち牌率
    sum = 0                                                       ▷ 牌の合計
    win = 1                                                       ▷ 特定牌の当たらない確率
    win_tsumo = 0                                                 ▷ 相手がツモ和了する確率
    discard_turn = 1                                              ▷ 各順で切る牌が当たらない確率
    foreach i ∈ all kinds of tiles do
        sum += tile_i
    end for
    foreach i ∈ all kinds of tiles do
        hand_i += (4 − tile_i)/sum × turn                         ▷ 仮想的に実数枚の牌を引く
        foreach j ∈ opponents do
            if Player_j.waiting = TRUE then
                win_i ×= 1 − winning_tile_ij
            end if
        end for
    end for
    foreach i ∈ all kinds of tiles do
        win_opponent += (4 − tile_i)/sum × (1 − win_i)
    end for
    tmp_turn = 0
    foreach i ∈ all kinds of tiles do                             ▷ win_i の高い順にソート
        while tmp_turn < turn do
            if tmp_turn + hand_i < tmp_turn + 1 then
                discard_turn_{int tmp_turn}
                ×= pow(win_i, hand_i)                             ▷ 実数枚の牌を切って当たらない確率
                tmp_turn += hand_i
            else
                discard_turn_{int tmp_turn}
                ×= pow(win_i, int(tmp_turn + 1) − tmp_turn)
                hand_i −= int(tmp_turn + 1) − tmp_turn
                tmp_turn = int(tmp_turn + 1)
            end if
        end while
        probability = 1                                           ▷ 降り成功率
    end for
    for i = 0 to turn − 1 do
        probability ×= discard_turn_i
        for j = 1 to sum(Player.waiting = TURE) do                ▷ 自分を除く聴牌しているプレイヤの数
            probability ×= (1−win_tsumo)×power(1.0−win_tsumo, sum(Player.waiting = TURE)−1)
        end for
    end for
    return probability
end function
```

表 5.6. 順位分布

	1 位率	2 位率	3 位率	4 位率	平均順位	試合数	安定レーティング	保証安定レーティング
提案手法	0.294	0.264	0.242	0.200	2.34 ±0.05	1508	1844	1752
降り成功率導入	0.277	0.273	0.251	0.198	2.37 ±0.03	4755	1821	1781
水上 [28]	0.241	0.281	0.248	0.230	2.46 ±0.04	2634	1718	1690

$$R' = R + (50 - Rank \times 20 + \frac{AveR - R}{40}) \times 0.2 \tag{5.9}$$

$Rank$ は前回のゲームでの順位である．$AveR$ は卓の平均の R である．初期 R は 1500 であり，およそ平均順位が 0.1 下がるごとにレーティングは 100 点ほど上昇する．

レーティングを用いた評価方法としては安定レーティングと保証安定レーティング [59] の二つが提案されている．安定レーティングは今までの順位分布をとり続けた場合のレーティングである．保証安定レーティングは連続する試合の結果を恣意的に切り出した時の安定レーティングを用いて，安定レーティングの現実的な範囲を保証する安定レーティングのことである．これは異なる試合数の恣意的に良かった結果を取り出しても比較可能な評価方法である．

5.5.3 天鳳での対戦における結果

提案手法と降り成功率導入と先行研究である水上らの手法 [28] の 3 つの比較を行った．対戦結果を図 5.6 に示す．Bootstrap Resampling [36] を用いて平均順位の信頼区間（p-value ≤ 0.05）を求めた．信頼区間はパーセンタイル法によって決定した．ブートストラップ回数は $10,000$ 回とした．

水上 [28] と降り成功率導入の平均順位の差はウェルチの t 検定（p-value $= 9.3 \times 10^{-5}$）により統計的に有意な差が見られた．降り成功率導入と提案手法の平均順位の差はウェルチの t 検定（p-value $= 0.26$）により統計的に有意な差が見られなかった．

降り成功率導入と提案手法の保証安定レーティングについては降り成功率導入のほうが高かった．この値は実力が安定しない人間同士が成績を比較するときに有用であり性質上，試合数が多いほうが高い値を出しやすい．実際，降り成功率導入の保証安定レーティングを求めるときに切り出した試合数は 4,300 ほど使っており，提案手法の 1,500 試合程度の良かった部分を切り出してもあまり成績は伸びない．

和了・放銃・副露時放銃率と局収支は表 5.7 に示す．これらは 1 局のプレイヤの強さを測定するためによく用いられている．先行研究に比べ降り成功率を導入することで，無駄な鳴きを減らし副露時放銃率を下げることに成功した．提案手法と降り成功率導入では和了率，放銃率とも少し改善した．これはこのプログラム自体が上級卓平均に比べ強く終盤を点数状況的で優位に立つ局面が多いため，オーラス等で無理して攻める機会が減り，降りていればよい局面が増えたため放銃率の改善したと思われる．またオーラス 4 位などは攻めて和了するケースが増え，和了率が改善したと思われる．

局収支とは全局で得られた得点を全局数で割った値である．基本的にはこの値が高いほど強いプレイヤとされる．降り成功率導入はこの値を最大化する目的で手を選択していた．提案手法は局収支

表 5.7. 和了・放銃率と局収支

	和了率	放銃率	副露時放銃率	局収支
提案手法	0.253	0.113	0.114	582
降り成功率導入	0.247	0.116	0.114	619
水上 [28]	0.245	0.131	0.127	412

表 5.8. 終盤に関するデータ

	トップ死守率	トップまくり率	ラス回避率
提案手法	0.807	0.104	0.237
降り成功率導入	0.781	0.106	0.227
水上 [28]	0.750	0.086	0.241

が降り成功率導入より低い．しかしながら平均順位は降り成功率導入よりも向上しているため，この指標からこのプログラムが局収支で無く最終順位を考慮したプレイヤであることがわかる．

点数状況判断の影響が最も出やすい終盤や順位に関してのデータを図 5.8 に示す．表中のトップ死守率はオーラスでトップであるときに，最終結果がトップであった割合である．トップまくり率はオーラスにトップでない時に，最終結果がトップになった割合，ラス回避率はオーラスで 4 位の時に，最終結果が 4 位以外である割合である．提案手法は降り成功率導入に比べ，トップ死守率，ラス回避率の成績が向上した．この指標から，提案手法が点数状況の判断が向上したといえる．

5.5.4 考察

図 5.2 に得点を考慮した手の例を示す．局面はオーラス，点数状況はプログラムが 35,700 点で 1 位であり，2 位の親が 35,400 点，あとの二人とは点数的には離れている．今，4 位からリーチがかかり，同順にプログラムが聴牌したところである．

この局面で大事なことはリーチを打って順位を悪化させないことと振り込む危険のある牌を切らないことである．人間であれば，このリーチに振り込みさえしなければトップであり，リーチを打つことで順位が悪くなるため，リーチ者に対して現物かつ聴牌も取れる 6 ピンを切ってダマにする．これにより次の順目でツモの可能性が残り，危険牌を引いた時には降りに回ることもできる．点数状況を考慮に入れない先行研究のプログラムであれば，6 ピンを切ってリーチという選択であったが，これは点数状況的に悪手である．提案手法は 3 ピンを切って降りる選択をした．これは 6 ピンを切ってリーチするより良い手である．最終的にはこの局は流局して，1 位を守った．

現在のプログラムが最善手である 6 ピン切ってダマが出来ない理由は，このプログラムがリーチが打てる時にはすべて打つというルールを用いているためである．得点状況を考えてリーチを打つ判断は強くなるうえで必須である．

提案手法の挙動を見る限り，現在の順位が 1 位の時は引き，4 位の時は押す傾向が確認された．相手の得点予測を平均値で推定しているため，相手との点差が平均値の 2 倍離れている場合は，放銃し

ても逆転されることはないとプログラムは誤って考えてしまう．実際には裏ドラが乗れば逆転することが頻出する．そこで振り込んだ場合の期待順位は現在の期待順位を上回ることはないとしている．そのため局を進めるために安い手に対してあえて放銃するという判断ができない．これを解決するためには相手の得点予測を平均ではなく分布で予測する必要がある．これによって逆転の起きる確率を考慮することが可能になり，意味のある得点差を考慮した特徴量もより効果的に働くと期待される．

5.6 シミュレーションの改良

この章では前章からの改良点を説明し，その成果を自己対戦によって調べる．

5.6.1 全体像と改良点

ここではアルゴリズムの全体像について説明する．基本的には第 5 章と同じである．ここで差異も含め，全体像を記述する．

5.6.1.1 序盤と中終盤のアルゴリズムの選択

提案手法では一局を通して，同じアルゴリズムを使うのではなく序盤と中終盤にわけ，それぞれに異なるアルゴリズムを用いて手を決定している．序盤では一人麻雀の手を用いて，手を決定する．中終盤ではモンテカルロ法を用いたシミュレーションを行い，手を決定する．

序盤と中終盤の切り替える条件の変更を行う．以下の条件のいずれかを満たしたときアルゴリズムが変更され，一度でも中終盤に切り替わるとその局以降はすべて中終盤のアルゴリズムで手の決定を行う．

- 誰かがリーチをかけた時
- リーチを宣言できる時
- ツモが可能な牌の数が 16 枚以下
- 一人麻雀の手を選択した時の放銃する確率が 2%を超えた時
- 現局面の期待最終順位とその放銃による期待最終順位の絶対値の差が 0.01 を超えた時

5.6.2 報酬の変更

相手のモデル化では相手の打点の平均点を回帰で予測するモデルを構築した．シミュレーションの報酬が点数である場合には問題は起きない．しかしながら順位を報酬とする場合には問題が起きる．期待最終順位は点数が相手よりも点数が上回っているかどうかの影響が非常に大きい．

そこで予測する対象を平均点ではなく各点数が起きる確率とする．モデルは多クラスロジスティック回帰モデルを使用する．麻雀の性質上，頻度の少ない和了点数の学習は適切に行われないと考え，最も点数の近い和了点数と同じ扱いを行った．具体的には 60，70 符は翻数を繰り上げ 30 符，80，90 符は翻数を繰り上げ 40 符，100，100 符は翻数を繰り上げ 50 符とした．これにより得点分布を当てる問題を 17 クラス分類問題とした．特徴量の変更を行わない．正解率は 37.9% となった．シミュレーションの報酬として使用する時は各点数が起きる確率にその時の点数状況からの期待最終順位を掛け合わせた値を用いる．

5.6.3 リーチダマの追加

以前のアルゴリズムではリーチを打てる局面ではダマにするシミュレーションは行っていなかった．当然，ダマが最善手になることも多いため，このシミュレーションでは適切な手を選択できない．そこでリーチを打てる時にはダマも探索する　ノードを追加した．

5.6.3.1 ベタオリと差し込み専用ノードを追加

シミュレーションでは全ての牌に対して探索を行う．そしてそのシミュレーションでは，プレイヤは特定の牌を切った後の手牌から和了に向かう．しかしながらそれではベタオリや差し込みといった和了を目指さない戦略を適切に評価することは出来ない．そこでベタオリと差し込み専用のノードを追加する．ベタオリのノードの牌は現局面における期待最終順位の悪化しない牌である．差し込みノードの牌は現局面の放銃率が高い牌である．このノード以下のシミュレーションは通常と異なり，プレイヤはルートで選択された条件を満たす牌を切り続ける．

またモンテカルロの手の評価値は以下の式で計算する．

$$\begin{aligned} &Score(Tile) \\ &= Sim(Tile) \times \prod_{p}(1 - LP(p, Tile)) + \sum_{p \in winpattern} LP(p, Tile) \\ &\times DP(p, Tile) \times EFR(\mathbf{z}), \end{aligned} \tag{5.10}$$

DP は得点分布である．以前と異なる点は以前の式ではダブロンを考慮した式になっていないがここでは考慮する．さらに失点も線形回帰で予測した点数でなく各点数が起こる確率を求め，その時の期待最終順位とする．これによって現局面の特定の牌を切ったときの評価がより正確に行えることになる．

5.6.3.2 UCB 値の導入

以前のシミュレーションでは全ての牌が均等に選ばれるようなアルゴリズムであった．当然ながら価値の少なそうな牌も高そうな牌も同じ回数だけ選択されるため，一番良い牌を選択する方法と

表 5.9. 順位分布

	1 位率	2 位率	3 位率	4 位率	平均順位	試合数
打点分布	0.252	0.251	0.249	0.246	2.49 ±0.01	81092
ダマ	0.254	0.253	0.247	0.243	2.48 ±0.01	72247
ベタオリと差し込み	0.253	0.249	0.256	0.241	2.49 ±0.01	40863
UCB	0.248	0.251	0.252	0.247	2.50 ±0.01	48881
山読み	0.252	0.253	0.247	0.246	2.49 ±0.01	39035

して効率的とは言えない．そこで UCB を用いて次にシミュレーションする牌を選択する．式は以下で表現される．

$$UCB = (rank(a_t) - 1)/3 - \sqrt{\frac{2\log(t)}{n(a_t)}}, \tag{5.11}$$

$rank(a_t)$ は時刻 t における牌 a を選択した場合の期待最終順位である．また期待最終順位は $[1, 4]$ であるから正規化してある．さらに期待最終順位は低いほど価値が高いので次に選択される牌は UCB が最も低い牌である．

5.6.3.3 山読み

シミュレーションを行うときには山を生成しプレイヤが次に自模る牌を決定する．現状では見えていない牌を数え上げ，矛盾がないように山を生成している．しかしながら局の終盤では相手の手牌に残っていそう牌とそうでない牌が存在し，そのから逆算して山を生成する方法はその精度が高くないと考えられる．そこで相手のモデル化に使用した特徴量と牌譜を用いて，山に特定の牌が何枚残っているかを予測するモデルを構築した．この予測モデルの結果を用いてシミュレーションごとに山を生成した．

5.6.4 最終的な結果

上記の改良の自己対戦での結果を表 5.9 に示す．Bootstrap Resampling [36] を用いて平均順位の信頼区間（p-value ≤ 0.05）を求めた．UCB 導入を除き，ほかの改良は信頼区間を見ると統計的有意な差をつけて古い AI よりも良い結果を出している．

これらの全ての改良を導入したアルゴリズムを天鳳の特上卓で対戦させた．結果を表 5.10 に示す．この結果から現状のアルゴリズムでは人間トッププレイヤに匹敵する実力をつけたことがわかる．

表 5.10. 天鳳での順位分布

	1 位率	2 位率	3 位率	4 位率	平均順位	試合数
全ての改良	0.297	0.263	0.234	0.206	2.35 ±0.05	2300
人間トッププレイヤ	0.287	0.269	0.266	0.178	2.34 ±0.04	3385

5.7 おわりに

本研究では現在の点数状況から最終順位を予測するモデルを構築し，シミュレーションの報酬として予測順位を使用することで，局収支ではなく最終的な順位を考慮したコンピュータ麻雀プレイヤの手を決定するアルゴリズムを提案した．予測順位自体の精度も高く，既存の研究よりも高速に行うことが可能になった．実際の手を決定する局面においても 1 位なら引き，4 位なら押し気味の手を選択することが可能になり，人間のような長期的な視点による手の選択が可能になった．結果として天鳳 [69] において 1,508 試合を戦わせた結果，安定レーティングが 1844 点となり，上級卓の平均を超え，その上の特上卓でプレイ可能な実力にまでなった．さらにシミュレーションの改良を加えることで人間トッププレイヤに匹敵する実力をつけた．

今後の課題のひとつは点数状況に応じた 1 局の序盤の手作るモデルの構築である．今のプログラムは序盤は一人麻雀の手を選択するだけであり，どのような点数状況であっても同じ手を指す．そのためオーラスで 4 位という状況にあっても，和了したとしても順位の変わらない手を選択してしまう．当然，上級者は今の点数状況に応じて，何点の手を作るべきかを考慮しており，その判断は最終的な順位に大きく影響するため実力向上に必須である．

また本章の初めに述べたように期待最終順位が 1 局単位の報酬として有用であることを示した．次の章ではこの 1 局の点数と期待最終順位を用いた強化学習について説明する．

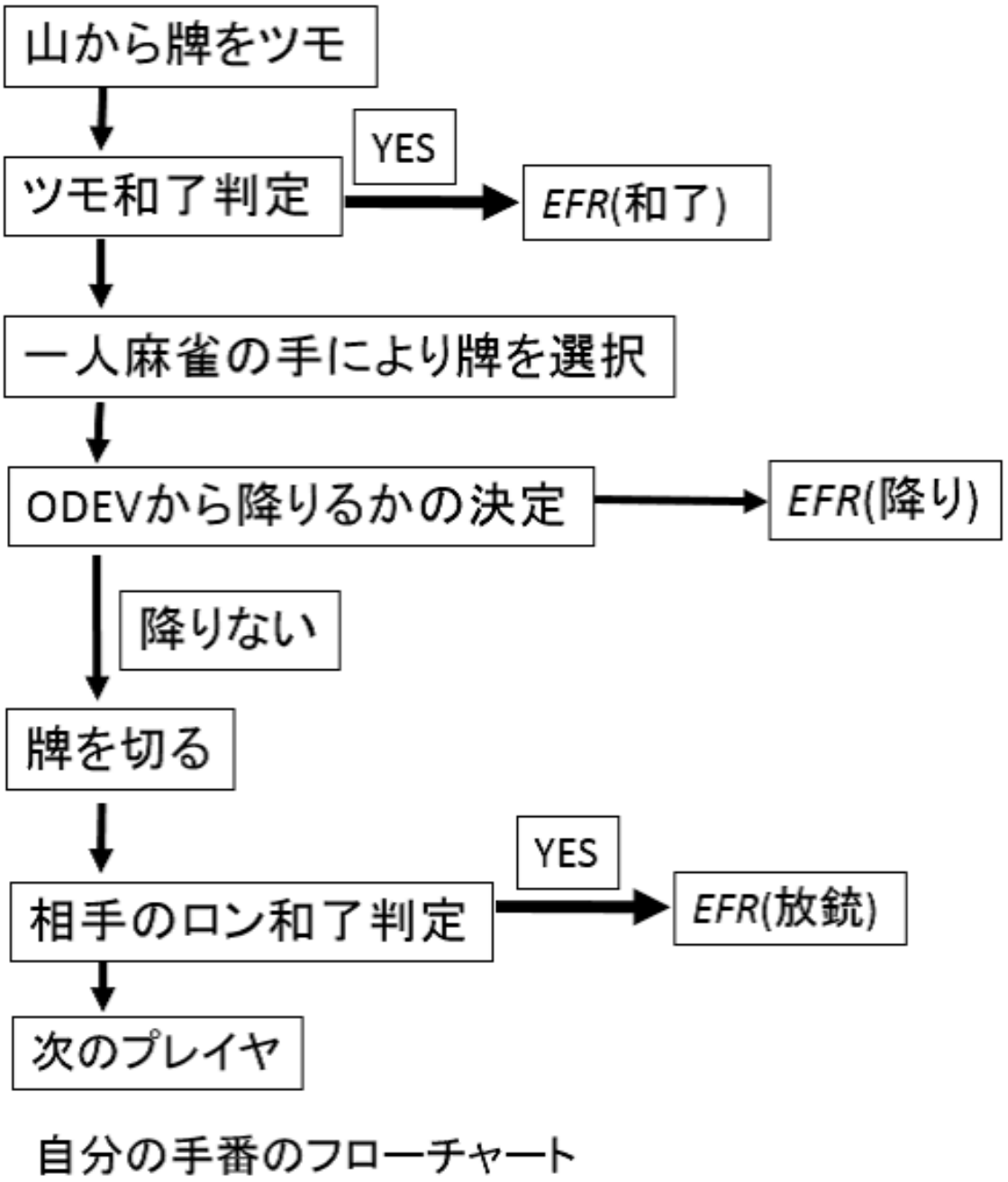
山から牌をツモ
ツモ和了判定
YES
EFR(和了)
一人麻雀の手により牌を選択
ODEVから降りるかの決定
EFR(降り)
降りない
牌を切る
相手のロン和了判定
YES
EFR(放銃)
次のプレイヤ
自分の手番のフローチャート

図 5.1. フローチャート

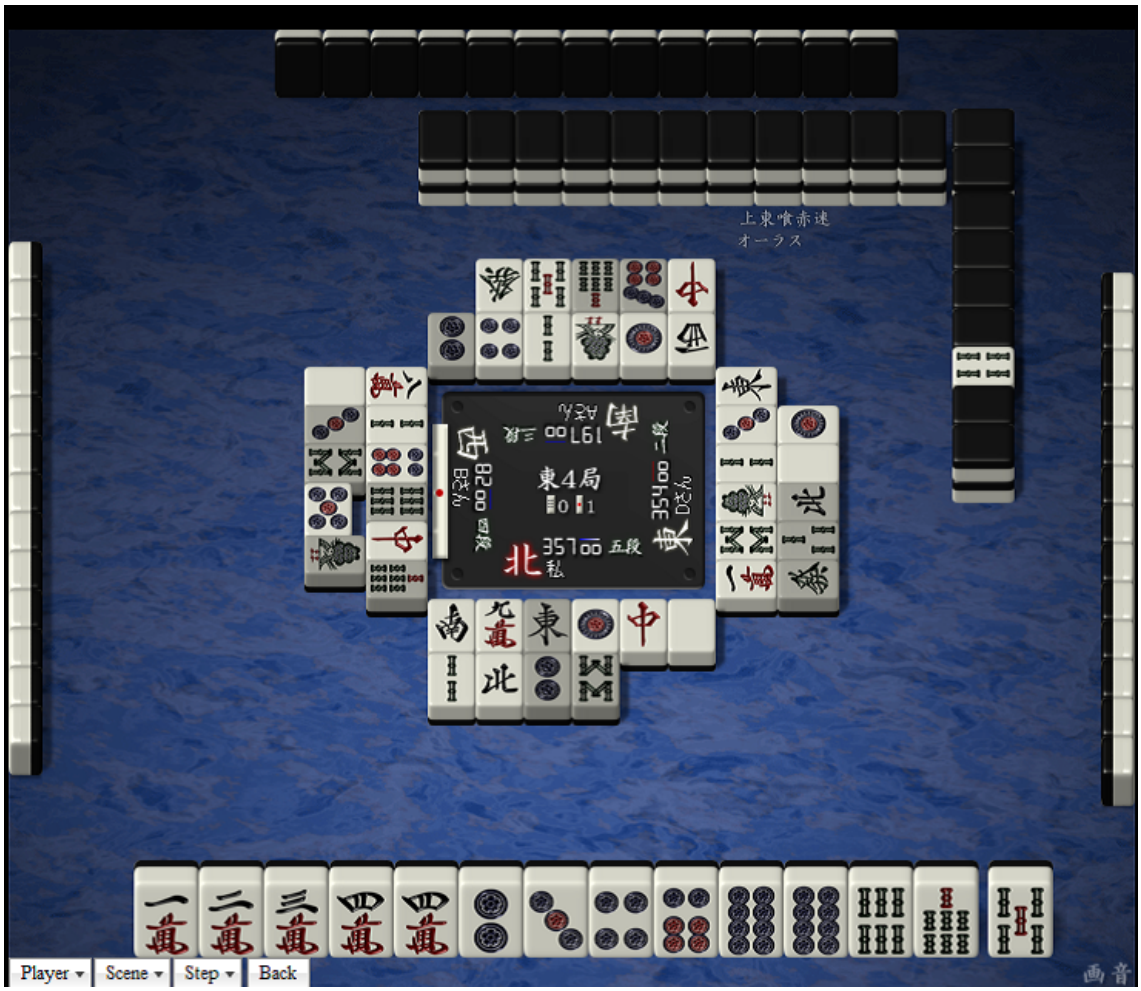

図 5.2. 得点を考慮した例

第6章　強化学習を用いた効率的な和了を行う麻雀プレイヤ

6.1　はじめに

前章では麻雀における疑似的な報酬を設計し，その有用性を確認した．この章では，これまでに構築したモデルを利用して本研究の目的である報酬が疎なゲームにおける強化学習を行う．

囲碁や将棋といった完全情報ゲームにおいてゲーム AI は自己対戦による強化学習によって指し手や局面に関する精確な評価関数の学習に成功している [46]．一方，不完全情報ゲームである麻雀では人間の牌譜を用いて学習を行うことで AI は上位プレイヤ並みの実力を獲得している [67].

本研究の対象となる麻雀は「不完全情報」や「多人数性」といった，AI の開発を難しくする性質を持っているが,「繰り返しゲーム」の性質もまた麻雀を複雑にしている要因である．麻雀は 1 局ごとに役に応じた点数を獲得し，全部で 4 または 8 局行う．全ての局が終了した時点で最も多くの点を持っているプレイヤが麻雀の勝者とみなされる．このようなゲーム性から麻雀は報酬が疎なゲームであることを 5 章で述べた．

5 章で述べたように繰り返しゲームとしての麻雀を対象にした研究として，筆者ら [67] は期待最終順位に基づいたプログラムを構築した．期待最終順位とは現在の局面から予想される最終的な順位のことである．この研究では期待最終順位を出力するモデルを牌譜を教師情報として学習を行った．そしてその出力である期待最終順位をシミュレーションの報酬とすることで点数状況を考慮したプログラムの構築に成功した．その結果，局単位の収支は減少する一方，トップ死守率などの順位にかかわる指標が向上し，それに伴い人間との対戦成績も向上した．このことから報酬が疎である麻雀の 1 ゲームに対する適切な手を選択することは，報酬は疎であるがそれよりは小さいサブゲームである 1 局の最適な手を選択することに置き換えることができる．

この研究では実力を判断する基準は人間の牌譜との一致率や対戦成績が用いられている．牌譜との一致率で比較するとそのプログラムの選択する手は従来のプログラムよりも低下している．しかし対戦成績で比較すると実力は向上している．そのため現在のプログラムは一部の局面において人間よりも良い手を選択していると考えられる．そこで本論文ではプログラム同士の対戦の牌譜を生成し，その牌譜を基に機械学習を行うことで元のプログラムの更なる実力の向上を試みた．

確かに筆者ら [67] の研究により点数状況を考慮した手を選択することが一部の局面において可能になった．しかしこのプログラムが明らかに点数状況を考慮していない悪手を選択する問題も存在する．それは特に 1 局の序盤において効率の良い和了を目指していないことである．麻雀において一般に安い手は和了しやすく，高い手は和了しにくい．そのため人間プレイヤは役の難易度と和了し

た時の点数状況のバランスを考慮して手を選択している．例えば最終局において 4 位の時，人間プレイヤは相手の点数を逆転する手を作ろうとする．また 1 位の時，高得点の和了の価値は低くいため時間をかけて高得点を目指す必要はない．そのため得点の高低よりも和了そのものを目指す．筆者ら [67] の研究において構築されたプログラムは局終盤の自分の和了した時の点数がほぼ確定した状態において攻めるべきか降りるべきかということは理解したものの，序盤において何点の手を作るべきかという長期的な戦略は苦手である．この原因は 1 局の序盤で用いるモデルでは現在の手牌が将来的に何点になるかを考慮していないモデルだからである．そこで本章では自己対戦による棋譜から教師あり学習を行い，現在の手牌から和了できる点数を予測するモデルを構築し，それを用いて効率の良い和了を行う麻雀プログラムを構築する手法を提案する．そのためには報酬が疎である 1 局から効率的に報酬である得点を得ることが必要になる．そして本章での手法は報酬が疎である麻雀をこの得点から期待最終順位を用いることで強化学習により元のプログラムの更なる実力の向上を試みた．

この章は以下の構成になっている．初めに 6.2 章で関連研究，6.3 章で本研究のベースラインとなる一人麻雀政策について述べる．提案手法として，6.4 章で自己対戦の棋譜生成と教師あり学習の方法，6.5 章で提案手法の対戦結果について述べる．

6.2 関連研究

6.2.1 オセロの関連研究

当時最強の AI であった **Logistello** [11] について説明する．オセロは角を取れば有利ということはは初心者でも理解できるが，その他の形は上級者ではないと判別が困難である．そこで Logistello はあらかじめオセロの重要な特徴量を決めて，その重みを学習するという方法を取る．またオセロは局面の進行度でもその特徴の価値が大きく変わるため，Logistello は局面の進行度をディスクの数で 13 種類に分類し，同じ特徴であっても進行度で異なる重みを持たせた．

特徴量の更新は自己対戦を行い局面を生成し，その局面の最終的なディスクの差を学習する．まず自己対戦として他のプログラムである **KITTY** と以前の Logistello との対局を 20,000 局行い．さらに Logistello 同士で 60,000 局行った．

こうして得られた局面に対して以下の式に基づいて特徴量の学習を行った．目的関数 E は以下の式で表される．

$$E(\mathbf{w}) = \frac{1}{N}\sum_{k=1}^{N}(r(p_k) - \sum_{i=1}^{n} w_i f_i(p_k))^2, \tag{6.1}$$

$\mathbf{w}$ が重みベクトル，p_k が局面，$r(p_k)$ が実際のゲームの最終局面のディスクの差，$\mathbf{f(p_k)}$ がその局面の特徴ベクトルである．したがってこの計算では実際のディスクの差と今の局面から推定されるディスクの差の二乗誤差を表している．これを次の式で更新する．

$$\mathbf{w_{t+1}} - \mathbf{w_t} = \alpha \nabla_{\mathbf{w}} E_{\mathbf{w_t}}, \tag{6.2}$$

Logistello と人間のトッププレイヤとの対戦は 1997 年に行われた．Logistello は結果として世界チャンピオンである村上健氏に 6 戦全勝した [12].

6.2.2 バックギャモンの関連研究

バックギャモンは強化学習を用いて初めて成功したゲームといわれている．その AI である **TD-gammon** [53] について説明する．TD-gammon は現在の局面をベクトル表現に落とし込んで入力させ，中間層の計算を経て最終結果（SW, DW, SL, DL）が起きる確率を推定する．

バックギャモンの知識を排除した特徴量を生成し，ニューラルネットワークの入力としている．具体的には各ポイントに対して 4 ユニットを割り当てる．3 つはバイナリで表現され，条件がそれぞれ石の数が 1，2 以上,3 である．4 つめのユニットは石が 4 以上の場合 $0.5 \times (n-3)$ の値とする．バーにおかれた石の数は $n/2$ で，ゴールした石の数は $n/15$ で表現される．最後に自分の手番をバイナリで表現する．これが黒と白の二つの色に応じてあるためユニットは全部で $(4 \times 24 + 3) \times 2 = 198$ 次元である．

TD(λ) [51] は自己対戦によって得られた局面の期待勝率と最終結果が等しくなるように評価関数を修正する方法である．更新式は以下のようになっている．

$$\mathbf{w_{t+1}} - \mathbf{w_t} = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_{\mathbf{w}} P_k, \tag{6.3}$$

$\mathbf{w_t}$ は重みベクトル，ステップ t における勝率を表したのが P_t であり λ は適格度トレースである．$\lambda = 1$ の場合，現在の局面からの最終的な勝率を当てるような更新がなされ，反対に $\lambda = 0$ の場合は現在の局面から次の局面の勝率を当てるような更新が行われる．λ を適切な値にすることで報酬を前の局面に減衰させながら報酬を伝搬させていく．実際には $\lambda = 0.7$ が用いられた．自己対戦では現在の局面の合法手の中で推移した局面の三層ニューラルネットワークで計算された最も評価値の高い手を選択する．結果として TD-gammon は世界チャンピオンに非常に近い実力となった．

6.2.3 テキサスホールデムの関連研究

不完全情報ゲームのポーカーの一種であるテキサスホールデムでは，Heinrich ら [21] は **Neural Fictitious Self-Play (NFSP)** を用いることで **CounterFactual Regret minimization (CFR)** [22] で構築されたプレイヤに迫る実力を得たと報告している．NFSP が導出されるまでの経緯を説明する．**Ficitious play** [9] は自己対戦からの学習方法である．これはプレイヤが対戦相手の最適な戦略を取ることで平均するとナッシュ均衡戦略 [34] に収束するというものである．ナッシュ均衡戦略とは自分一人が戦略を変えたとしても利益を増やすことが出来ない戦略のことを指す．

Ficitious play は標準形ゲームに対して定義されており，展開型ゲームには効果がなかった．**Full-Width Extensive-Form Fictitious Play (XFP)** [21] が考案され，fictitious players の戦略の更新が線形時間で可能になった．さらに計算時間のかかる最適戦略の計算と平均戦略の更新を強化学習と教師あり学習にそれぞれ置き換えた **Ficitious Self-play (FSP)** が提案された．

NFSP は FSP をニューラルネットを用いたという点で進化している．NFSP は強化学習用と教師あり学習の二つのデータセット M_{RL}, M_{SL} を取り使う．それぞれ行動価値関数 F_Q の学習と過去のプレイヤの行動 F_S を学習するためである．近似の最適戦略として $\beta = \epsilon\text{-}greedy(F_Q)$，エージェントの平均戦略を $\pi = F_S$ として学習中のエージェントの行動は $\sigma = (1-\eta)\pi + \eta\beta$ で決定される．η は実数であり，**Anticipatory parameter** と呼ばれる．結果として NFSP は CFR のプレイヤにはやや劣る実力であった．

6.2.4 囲碁の関連研究

囲碁において AlphaGo では指し手を確率分布でもつモデルをより賢くかつ勝率に変換する方法として強化学習が用いられた [45]．まず以前の研究 [48] のように教師あり学習によってロールアウトポリシー P_π とポリシーネットワーク P_σ を学習する．これらは局面が与えられたときに各合法手を打つ確率分布を返す．学習局面は 3000 万局の局面を KGS という囲碁サーバーから取得した．ロールアウトポリシー P_π の学習は高速で計算する必要があるため線形モデルで行われ，ポリシーネットワーク P_σ は 13 層の畳み込みネットワークで行われる．以前の研究での次の一手を当てる予測精度は 44.4% [15] であったが．ポリシーネットワークの正解率は 57%になった．

教師あり学習によって得られたポリシーネットワーク P_σ を強化学習によって RL ポリシーネットワーク P_ρ を構築する．ネットワークの構成は両方とも全く同じである．まず初期値として $\rho = \sigma$ とする．自己対戦では現在のポリシーネットワーク P_ρ とランダムに選択された過去のポリシーネットワーク P_{ρ^-} と対戦する．対戦結果に応じて以下の式によってポリシーの更新が行われる．

$$\Delta\rho \propto \frac{\delta log P_\rho(a_t|s_t)}{\delta\rho} z_t, \tag{6.4}$$

ここで t はステップ時間，a が指し手，s が局面，z_t は勝った場合 1 負けた場合-1 とする．500 試合行うたびにその時のポリシーネットワーク P_ρ を過去のポリシーネットワークに追加する．この P_ρ の実力を測るために **Pachi** [3] と対戦させた．Pachi は一手に 10 万回のシミュレーションを行い，P_ρ は探索は行わない．結果として P_ρ の勝率は 85% となった．一方，強化学習前の P_σ は 11% となった．このことから強化学習による方法は有効であったとわかる．

次のフェーズでは強化学習によって得られた P_ρ を用いて局面の勝率を予測するモデルの $V_\theta(s)$ を構築する．方法は自己対戦を行い，その中で一つの局面を取り出して，最終的な結果を予測するような学習を行う．学習に用いる局面の作り方を説明する．学習に用いる局面のステップ数をランダムに決める．ステップ数のひとつ前まではロールアウトポリシー P_π によって手を決める．次に完全にランダムな手を決め，その手を着手した局面が学習に用いる局面である．最終結果はその局面から P_ρ を用いてゲームを進めることで得る．AlphaGo は 3000 万の局面を生成し学習を行った．結果として AlphaGo は世界チャンピオンに勝利するまでになった．

6.2.5 一人麻雀の関連研究

小松ら [66] や海津ら [65] はモンテカルロ法を用いて　鳴きは行えない一人麻雀の手作りを行うプレイヤを構築した．この手法はランダムに手を選択するのではなく，和了できる組み合わせになるまでランダムに牌を足し続け，各牌の和了点の寄与を調べることで，和了に必要な牌を求めた．海津らは早和了に必要なパラメータを導入し，早和了と高得点のバランスをとった．しかしながらどの局面において早和了または高得点を目指すべきかということについては触れられていない．

栗田ら [71] は有向非巡回グラフを用いて一人麻雀の探索アルゴリズムを提案した．手牌と打牌回数とゲームの進行フェーズに基づいて有向非巡回グラフの接点をまとめた．これにより終端ノードでの報酬を設定すれば，麻雀の高い専門知識を使用することなく，3,4 シャンテンの手牌であれば効率的な一人麻雀プレイヤの構築に成功した．

6.3 一人麻雀政策

本章では本研究で重要な役割を果たす一人麻雀政策 [68] について述べる．以前の研究では一人麻雀の手 [68] と呼んでいたが，手という表現は麻雀において複数の意味を持つため本研究では一人麻雀政策と呼ぶ．一人麻雀政策は人間の牌譜から教師あり学習によって得られたモデルにより決定する．そしてこの政策は単純に和了に向かう行動だけを選択する．

なぜ一人麻雀政策が重要な役割を果たすか説明する．麻雀の 1 局を考慮してもランダムな手を選択し続けても報酬が疎であるため和了し報酬を得ることは困難である [64]．従って報酬を得るためには効率的な探索を行う必要がある．本研究の Atari では探索ボーナスを定義したが，これを利用しても 1 局の開始状態が毎回異なるため，探索ボーナスが減らず効率的な探索は困難である．そこでより和了し報酬を得ることを目的とした一人麻雀政策が必要になる．これにより報酬が疎である麻雀の効率的な探索を行うことが可能になる．

麻雀では和了に向かう目的だけでなく放銃を避ける目的など異なる目的を意図してプレイヤは牌を選択する．そのためすべての牌譜を学習に用いることは一人麻雀政策の学習には適切ではない．そこで一人麻雀政策の学習ではすべての牌譜の中から和了に向かう手を含んだ牌譜を抽出する．しかし和了に向かう手は人間同士でも判断基準が異なるため，選択した意図を正確に読み取るのは困難である．そのため客観的に判断可能な誰もリーチを宣言していない状態で和了したプレイヤまたは最初にリーチを宣言したプレイヤの牌譜を用いる．人間の牌譜はインターネット麻雀サイト天鳳 [69] の鳳凰卓の牌譜である[1]．

学習では牌譜中で実際に選択された牌と現在の重みベクトルから選択される牌の評価を近づけるように重みベクトルの調整を行う．具体的な学習方法は与えられた手牌から一つ牌を切ったときの手牌を想定し，その手牌から抽出される特徴量と重みベクトルの内積によって評価値を計算する．これをすべての牌について行い一番評価値の高い牌と実際の牌譜で切られた牌が異なる場合に重みベクトルを更新する．これを繰り返すことで重みベクトルは牌譜と同じ手を選択するように調整され

[1] 2009 年 2 月 20 日から 2015 年 12 月 31 日までに行われた対局

る．牌譜自体は和了に向かう手であるため，そこから得られる一人麻雀政策は和了に向かう手である．教師あり学習の方法として平均化パーセプトロン [16] を用いた．

筆者らの研究 [68] ではこの特徴量では役の表現力が乏しく，当時のプログラムは実際の対局において役を完成できない鳴きを頻繁に行った．そのためこの特徴量は，本研究では役を作るための特徴量として適さないと判断した．そこで本研究では特徴量の改善を行った．詳細は表 6.1，表 6.2 に示す．多くの特徴量はいくつかの要素の組み合わせで構成されている．すなわち組み合わせの全ての要素が満たされるときにベクトルの値が 1 となるような特徴量である．特徴量の数は合計で 6,661,309 である．

改善した特徴量を用いた実験の結果を表 6.3 に示す．表中のツモ局面とは打牌を選択する局面であり，鳴く局面とは牌譜中のプレイヤが鳴いた局面であり，鳴かない局面とは牌譜中のプレイヤが鳴ける局面で鳴かなかった局面を指す．その鳴く局面において一人麻雀政策と牌譜の晒した牌が一致し，さらにそこから切った牌も一致した局面数が完全一致数である．晒した牌は一致したものの切った牌が異なる場合は鳴きのみ正解数としてカウントされる．鳴いたことが重要であるため鳴きに関する正解率は完全一致数と鳴きのみ正解数の合計を鳴き局面数で割った値とする．

テストデータが異なるため単純な比較はできないが，以前の結果 [68] では鳴く局面での正解率は 84.2% であり，鳴かないときの正解率は 90.7% であった．このことから特徴量を改善することで一致率が向上し，役が完成しなくなる鳴きは減少したと考えられる．本研究ではこの一人麻雀政策を用いて実験を行う．

一人麻雀政策はある手牌が与えられたときに牌譜中のプレイヤが最も選択するであろう牌を切る．そのため打牌の基準となる評価値は牌の選択されやすさであり，現状の手牌が何翻で和了できるかは全く考慮されていない．そのため一人麻雀政策に従うと平均的な局面においては悪手であることは少ないが，オーラスといった得点状況に応じて最善手が変わるケースにおいて悪手となりうる．次の章はこの原因を解消する手法について述べる．

6.4 自己対戦の棋譜を用いた教師あり学習による役作り

前章で述べたように一人麻雀政策は現在の手牌の将来的な報酬を理解していない．この章では自己対戦の棋譜を用いた教師あり学習を用いてこの問題に取り組む．

図 6.1 は提案手法の全体像である．前章の人間の牌譜から一人麻雀政策を作成し，この一人麻雀政策を利用して AI 同士の牌譜を生成，この AI 同士の牌譜から翻数予測モデルを構築する．すなわち牌譜の生成には一人麻雀政策に従う AI が 4 体の一人麻雀でない四人の麻雀を行う．この翻数予測モデルに以前の研究で用いた期待最終順位と和了できないペナルティやゲーム木の探索を組み合わせることで、提案手法となる序盤アルゴリズムを構築する．対戦実験では以前の研究の麻雀プログラムと新しい麻雀プログラムとの対局を行う．新しい麻雀プログラムと以前の研究の麻雀プログラムの差は序盤アルゴリズムが提案手法の序盤アルゴリズムに置き変わったことである．すなわち中終盤は共通のアルゴリズムを用いた序盤だけ戦略の異なる麻雀プログラム同士の対局である．

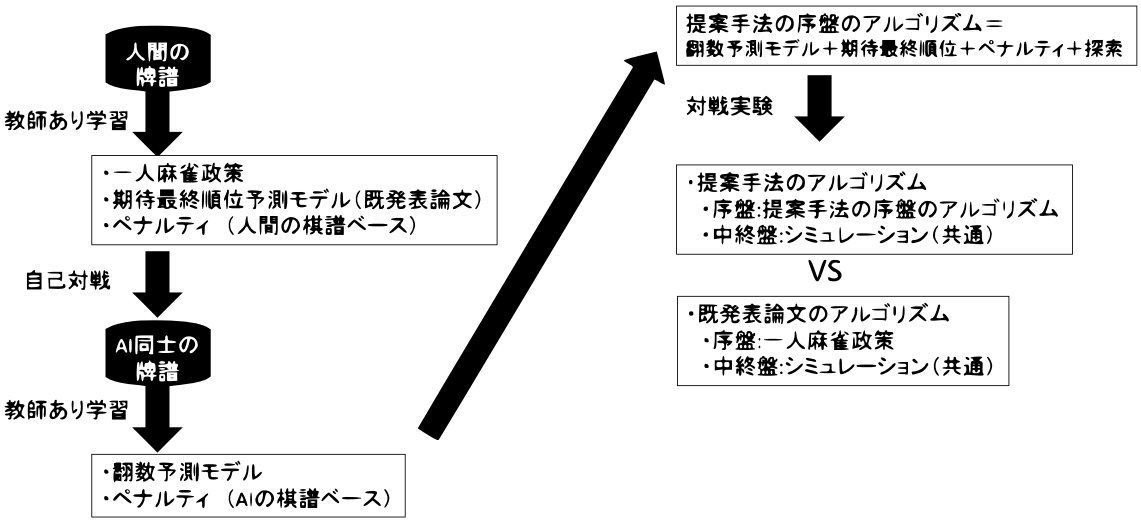

図 6.1. 提案手法の全体像

6.4.1 自己対戦の棋譜を用いた教師あり学習の方法

本研究の目的は現在の手牌から局終了時に得られる報酬を予測することである．そのための予測モデルを自己対戦の棋譜を用いた教師あり学習を用いて構築する．このモデルを翻数予測モデルと呼ぶ．翻数予測モデルの出力は特定の翻数を和了する確率とする．すなわち翻数予測モデルの出力された値は一人麻雀政策の評価値ではなく実際の麻雀における報酬を表現する値である．この値を用いることで本研究では現在の一人麻雀政策の将来的な報酬を理解していないという問題に取り組む．

具体的な手法は麻雀の特徴を考慮する．麻雀ではランダムな手を選択し続けても和了し報酬を得ることは困難である [64]．すなわち特徴量の重みを 0 もしくはランダムに初期化して強化学習を行う方法は報酬による学習が行われないため，うまくいかないと考えられる．しかしながら前章で述べたように一人麻雀政策はある程度は強いため，本研究ではこれを活用した手法を提案する．

本研究では AlphaGo [45] の評価値ネットワークの学習に用いられた手法を参考にする．AlphaGo の評価値ネットワークの出力は現在の局面の勝率である．これを実現するためには局面と最終結果のペアが必要になる．そこで AlphaGo は人間の棋譜から学習した政策ネットワークをベースに二段階の強化学習を行い評価値ネットワークに必要な学習局面を生成した．一段階目では現在の政策ネットワークと過去の政策ネットワークを対戦させることで政策ネットワークを改良した．二段階目は改良した政策ネットワークを用いて評価値ネットワークに必要な棋譜を生成する．牌譜の生成アルゴリズムは AlphaGo に対局中に一手だけ完全なランダムな手を打たせ，その後を終局まで改良した政策ネットワークを用いた手を打たせることで実現した．これらの対局を大量に行うことで評価値ネットワークに必要なランダムな手を打った局面とその最終結果を生成した．学習の手法を参考するという点において AlphaGo における一段階目の生成物は政策ネットワークであり，前章の結果を踏まえると特定の局面から終局まで行う役割を一人麻雀政策は果たせると考える．本研究は二段階

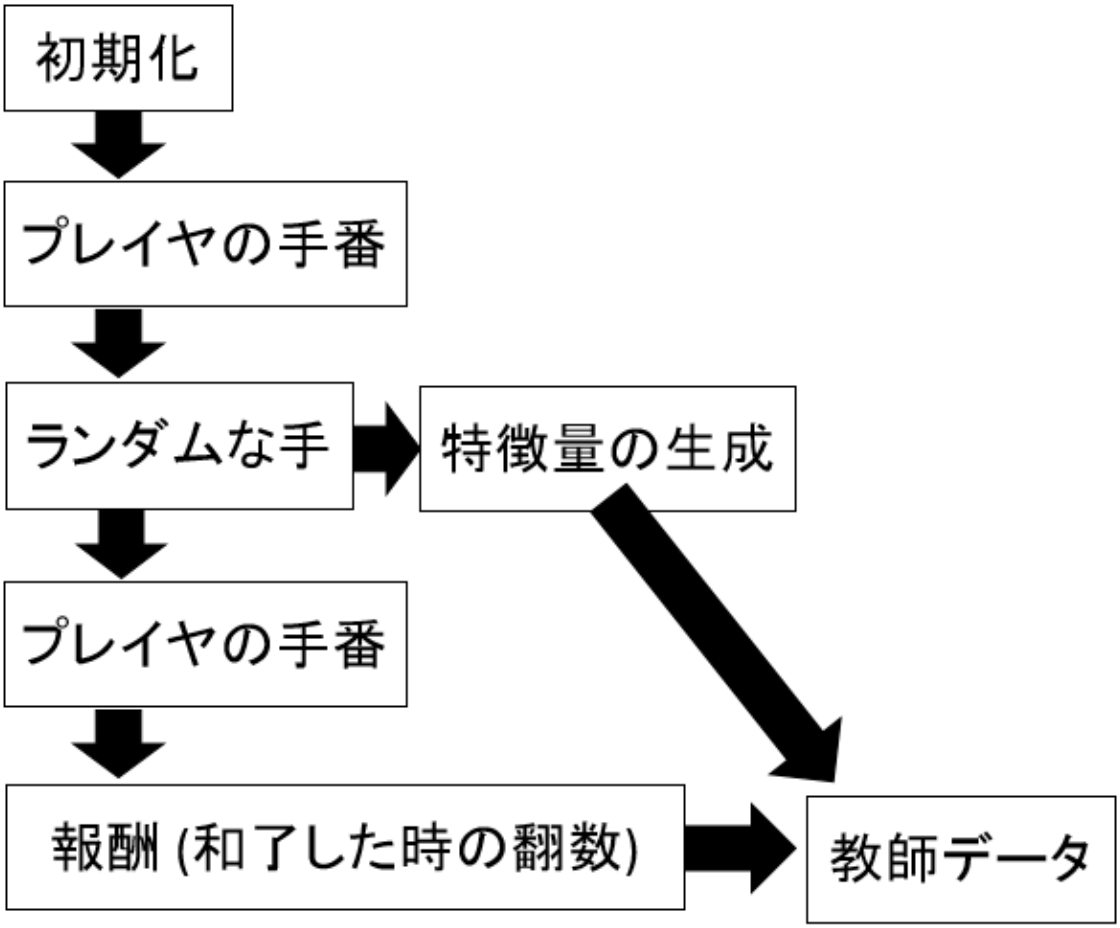

図 6.2. 牌譜生成のフローチャート

目の評価値ネットワークの構築法を参考にする

AlphaGo の評価値ネットワークの構成に成功した理由は，多様な局面を生成した点と考えられる．同じような局面ばかり生成して評価値ネットワークの学習を行った場合，未知の局面に対する適切な評価を評価値ネットワークが出力することは困難である．一方で完全にランダムな手を何度も用いると現実に起こりにくい局面を生成してしまい学習の効率が悪い．適切にランダムな手を混ぜることで学習に適切な未知の局面を生成することが可能になったと考えられる．

麻雀においても多様な局面をサンプリングする方法が必要になる．しかし囲碁と麻雀ではゲーム性が大きく異なるため，単純に置き換えることはできない．以下，本研究の局面の生成法について述べる．

図 6.2 は牌譜生成のフローチャートである．初期化の対象はプレイヤの配牌，山，ドラ，自風，場風，教師データとして使用する巡目でありそれぞれランダムに決定される．自風は各風が 1/4 の確率

で選択される．場風は本研究では東風戦しか行わないものの，一般的なルールでも成り立たせるために東南戦の西入まで考慮する．実際には西入することは少ないため場風が西のデータは東と南に比べ少なくても問題ない．これを考慮して本研究では場風は東と南は 4/9，西は 1/9 の確率で選択する．

初めにプレイヤの手番について述べる．AlphaGo の場合，二人のプレイヤのうちどちらか一人だけがランダムな手を一回打つ．四人で行う麻雀において二人以上が完全にランダムな手を打つ場合，得られる牌譜に悪手が増え学習局面としては適切でない．また誰も完全にランダムな手を打たない場合，局面の多様性がないため学習に適切な局面数が増加しない．そのため本研究では特定のプレイヤ一人が一局において一回だけランダムな手を打ち，その局面を学習局面とする．すなわち生成されるすべての学習局面はランダムな手を打ったプレイヤからの視点での情報のみ使用し，残り三人のプレイヤの手牌の情報は学習局面に反映されない．また報酬もランダムな手を打ったプレイヤからの視点であり，ほかのプレイヤの和了は報酬を 0 として扱う．以下，その特定のプレイヤを自分プレイヤ，それ以外の残り三人を相手プレイヤと呼び，各プレイヤの手番での行動を説明する．

まず自分プレイヤの手番の挙動について説明する．上記で説明したようにランダムな手を選択し続けても和了することは困難であるため，そのような方法で生成された牌譜は学習にはあまり役に立たない．そこで自分の手番においてプレイヤは基本的に一人麻雀政策に従う．報酬が疎なゲームにおいてはその局面を適切に評価するためには多様な局面を探索する必要がある．そこで例外的に一度だけ，自分プレイヤは一局の間に一度だけランダムな手を選択し，その直後の局面を学習局面に使用する．そしてそれ以降はその局面の精度の高い報酬を得るため一人麻雀政策に従う．

ランダムな手とは合法手の中から一人麻雀政策の評価値とは関係なく完全にランダムに選択された手である．ツモ局面であれば，各牌を切る（5 と赤 5 は同一とする）手と加カンと暗カンが合法手にあたる．鳴ける局面におけるランダムな手は少し複雑であり鳴き方と切る牌の二つを考慮する必要がある．ポンやカンの場合は鳴くか鳴かないの二種類で済むが，チーの場合，牌の晒し方が最大三種類ある．また鳴いた後に切ることができる牌は，すでに完成しているメンツを鳴く行為（例えば 123 から 1 または 4 をチーして 1 を切る）を禁止するルールが天鳳では採用されているため，すべての牌が選択可能とは限らない．そのため鳴ける局面における合法手は鳴き方と直後に切る牌を組み合わせによって決定する．

次に相手プレイヤの手番の挙動について説明する．得られた牌譜を使用して学習するため相手のプレイヤの挙動は牌譜の質に影響するため重要な要素である．当然，牌譜の質が実際のゲームと乖離していれば，その牌譜で学習したプレイヤは強くないことが予想される．また学習には報酬の獲得が容易であることも重要である．麻雀は相手がいない一人麻雀でも報酬が疎であるため相手のプレイヤの行動によっては報酬が容易に獲得できず学習が有効に行われない可能性がある．すなわち牌譜の質を高さと報酬が獲得という両立できない二つの要素がある．

そこで本研究では相手プレイヤを二種類用意した．一つ目の相手プレイヤはツモ切りを続けるプレイヤである．このプレイヤはどのような局面においてもツモ切りを行い，鳴きや和了は一切しない．これにより相手に邪魔をされることなる和了できるため報酬が密であることが期待される．

二つ目のプレイヤは一人麻雀政策に従うプレイヤである．ツモ切りプレイヤが三人いる状況においてプレイヤが和了することは実際の麻雀と比較して容易である．現実の麻雀では他のプレイヤが和了するため，一人麻雀のように 18 回のツモで和了すればよいのではなく，局が終了するまでのツ

モはそれよりも少なくなる．この状況を実現するため，相手プレイヤの挙動を一人麻雀政策にすることで牌譜が生成される環境を実際の麻雀の状況に近づけた．

ここでは報酬の設計について説明する．麻雀の点数は翻と符によって決まる．単純に翻と符をペアにして学習を行うとクラス分類数が増大し，学習が有効に働かない．そこで点数において符の影響は小さいため，これを無視する．すなわち報酬は 0 (和了できない), 1, 2, 3, 4 翻以上とした．跳満以上は，狙ったとしても簡単に和了できないため本研究では 4 翻以上は同じカテゴリとして分類する．

麻雀では手牌が和了に近づくにつれて必要な牌の種類が少ないためツモによって手牌が更新されることが少なくなる．そのため終局までの手牌をすべて学習に用いると，同じような局面をモデルが学習してしまう．そのため学習局面は 1 局に対して 1 局面までとして 1 億局面を用意した．リーチを打った後の局面は合法手が一つしかないため教師局面には使用しない．また天和や地和で和了した時の局面は自分が一手も指していないため使用しない．

上記の方法によって現在の手牌から将来の報酬を予測に必要な牌譜生成を行う．しかしながら実際の麻雀と比較し考慮できていない部分も存在する．

- 一人麻雀政策はリーチするかどうかは判断できないため，リーチが宣言できる局面においてリーチは全て宣言するとした．
- 相手プレイヤが降りるといった戦略をとらないので，切られにくい牌を待つ戦略などが不当に高く評価される可能性がある．
- 一手しかランダムな手を混ぜていないため，無理やり特定の役を狙うことは考慮に入れていない．
- 和了することが重要であるため和了が可能な時はすべて和了する．

6.4.2 翻数予測モデルの学習

ここでは生成した教師データを用いて翻数予測モデルを学習する方法について述べる．予測モデルの出力形式として回帰と分類が考えられる．回帰モデルを用いると出力結果は有理数であり，その値は実際の麻雀のゲームに存在せず扱いづらい．そこで翻数予測モデルは分類モデルを利用する．

分類モデルを利用するものの翻数の数字自体は大小関係が成り立ち無関係ではない．つまり一翻を和了するモデルの学習に二翻で和了した学習局面を正例とするか負例とするかは自明でない．そこで本研究では二種類の方法を用意した．一つ目はちょうど特定の翻数を和了できるかどうか学習するモデルである．つまり一翻を和了するモデルの学習に二翻で和了した学習局面を負例として扱う．予測する結果は牌譜生成の報酬をもとに 0 (和了できない)，1, 2, 3, 4 翻以上の 5 種類とする．現在の手牌から予想される翻数を予測するということは多クラスロジスティック回帰モデルを使用することで 5 クラスの多クラス分類問題として捉えることができる．

出力としてソフトマックス関数を使用することにより各翻の和了できる確率として出力することができる．ソフトマックス関数は次の式で表現される．

$$P_{mc}(\mathbf{x}, h) = \frac{\exp\left(\mathbf{w}_h^{\mathrm{T}}\mathbf{x}\right)}{\sum_{i=0}^{4}\exp\left(\mathbf{w}_i^{\mathrm{T}}\mathbf{x}\right)} \tag{6.5}$$

ここで $\mathbf{x}$ は現在の手牌を表す特徴ベクトル，h は翻数である．$\mathbf{w}_h$ は各翻数 h の特徴量に対しての重みベクトルである．

目的関数 L_{mc} は次の式で表現した．

$$L_{mc}(\mathbf{w}) = -\sum_{i=1}^{N}\sum_{h=0}^{4} c_{i,h}\log(P_{mc}(\mathbf{X}_i, h)) + \frac{\lambda|\mathbf{w}|^{\mathbf{2}}}{N}$$

ここで N は学習データの事例数, $\mathbf{X}_i$ は i 番目の学習事例, $c_{i.h}$ は学習事例の結果と各翻数に対応する 2 値（1 または 0）のラベルである．λ は学習データに過学習することを防ぐ正則化の係数である．本研究では λ を 0.01 とした．

二つ目は特定の翻数以上を和了できるかどうか学習するモデルである．つまり一翻を和了するモデルの学習に二翻で和了した学習局面を正例として扱う．手牌から予想される翻数をロジスティック回帰モデルを 4 つ構築することで表現する．4 つのモデルはそれぞれ 1 翻以上, 2 翻以上, 3 翻以上, 4 翻以上を和了できるかどうかを予測する．

$$P_{bc}(\mathbf{x}, h) = \frac{1}{1+\exp\left(\mathbf{w}_h^{\mathrm{T}}\mathbf{x}\right)} \tag{6.6}$$

目的関数 L_{bc} は次の式で表現した．

$$L_{bc}(\mathbf{w}) = -\sum_{i=1}^{N} c_{i,h}\log(P_{bc}(\mathbf{X_i})) + (1-c_{i,h})\log((1-P_{bc}(\mathbf{X_i}))) + \frac{\lambda|\mathbf{w}|^{\mathbf{2}}}{N}$$

これら二つの重みベクトルの学習は確率的勾配降下法の一種である FOBOS [20] を用いて学習を行う．学習率は Adagrad [19] を用いて決定する．$\mathbf{x}$ は一人麻雀政策の学習に使用した特徴量と同じ特徴量を使用する．以後，式（6.5）を用いたプレイヤを **Multi Class Player (MCP)** と呼び，式（6.6）を用いたプレイヤを **Binary Class Player (BCP)** と呼ぶ．

6.4.3 提案手法の結果

翻数予測モデルが実際に有効に活用できるかを調べるために評価を行う．この評価では学習したモデルを使用して配牌とツモから特定の翻数 (1, 2, 3, 4) で和了できたかを調べる．特定の翻数もしくはそれ以上の翻数で和了した場合を成功とする．評価基準は成功率，すなわち成功した数と試行回数の商である．評価時の相手は牌譜生成時の相手と同じである．すなわち牌譜生成時の相手がツモ切りである場合，評価の相手もツモ切りである．同様に牌譜生成時の相手が一人麻雀政策である場合，評価の相手も一人麻雀政策である．テスト時，BCP は式（6.6）をそのまま用い，MCP では特定の翻数以上で和了できる確率の合計を用いる．BCP も MCP も打牌の選択は，ある牌を切った手牌の特定の翻数で和了できる確率を求め，最も大きな確率を与えた牌を選択する．

各評価実験ごとに配牌やツモによって結果が変わらないようにするため同じ山を使用する．各翻数ごとに一万局を評価する．牌譜生成時と同じ条件にするため，和了可能な場合は特定の翻数を満たしているかどうかに関わらず，すべて和了する．

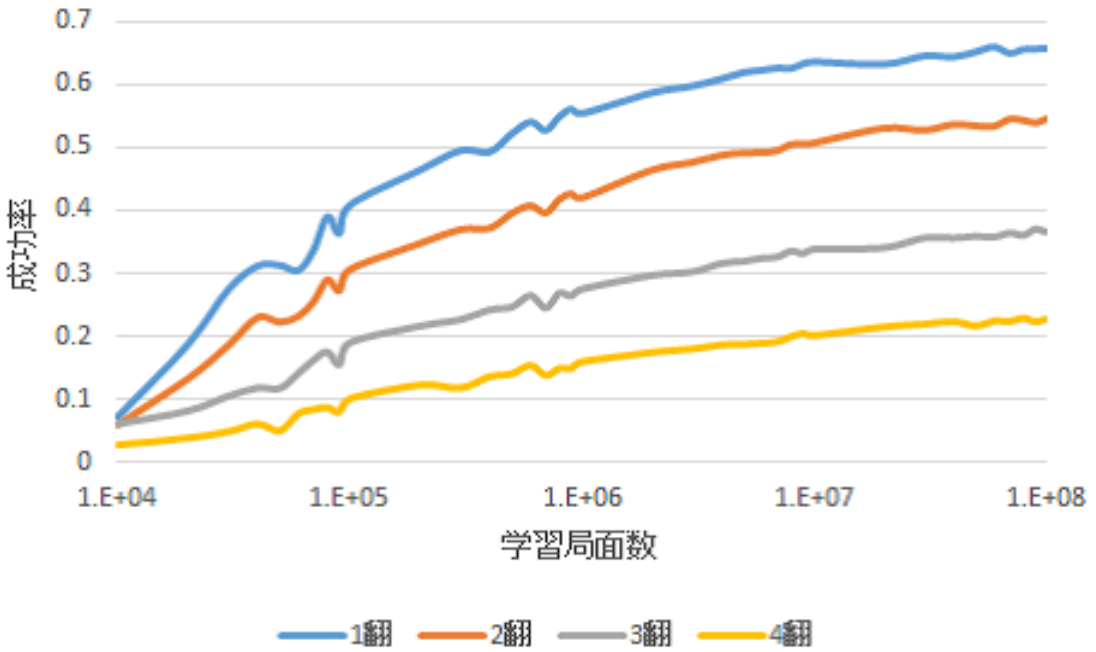

図 6.3. MCPvs ツモ切りにおける局面数と各翻数の成功率

提案手法が有効に行われているかを調べるため学習局面の数とその時点での成功率を調べる．結果を図 6.3, 図 6.4, 図 6.5 に示す．牌譜生成時の相手がツモ切りである場合，翻数予測モデル名に “vs ツモ切り” を付け，相手が一人麻雀政策の場合は何も付けない．学習局面の数は対数軸である．どの学習方法においても，基本的には局面を増やすほど成功率が高くなっているため学習が上手く行われているといえる．

1 億局面を学習した時の評価の結果を表 6.4 に示す．ベースラインとして予測モデルの代わりに一人麻雀政策を使用した結果も示す．MCPvs ツモ切りの場合，ベースラインと比較してすべての翻数において成功率が向上している．一人麻雀政策の学習は実際の牌譜であるため相手の行動はツモ切りではない．そのためその牌譜を基に学習した一人麻雀政策よりも MCPvs ツモ切りは相手の行動に応じて適切に打牌を選択しているといえる．

MCP や BCP の場合，低い翻数の時には一人麻雀政策に負けているものの，4 翻時には成功率が向上している．その原因は一人麻雀政策の学習に用いる牌譜の性質が影響していると考えられる．一人麻雀政策の学習に用いる牌譜は誰もリーチを宣言していない状態で和了したプレイヤまたは最初にリーチを宣言したプレイヤの牌譜を用いる．そのため牌譜の中のプレイヤは早くて安い手を和了する行動が多く含まれていることになる．この牌譜の影響から一人麻雀政策は高い手を和了することが苦手であると考えられる．

現状の一人麻雀政策の具体的な問題点はゲーム終盤の逆転に必要な点数を作る技術がなかったことである．そのため高い点数を和了する技術が向上したことは一定の成功を収めたといえる．

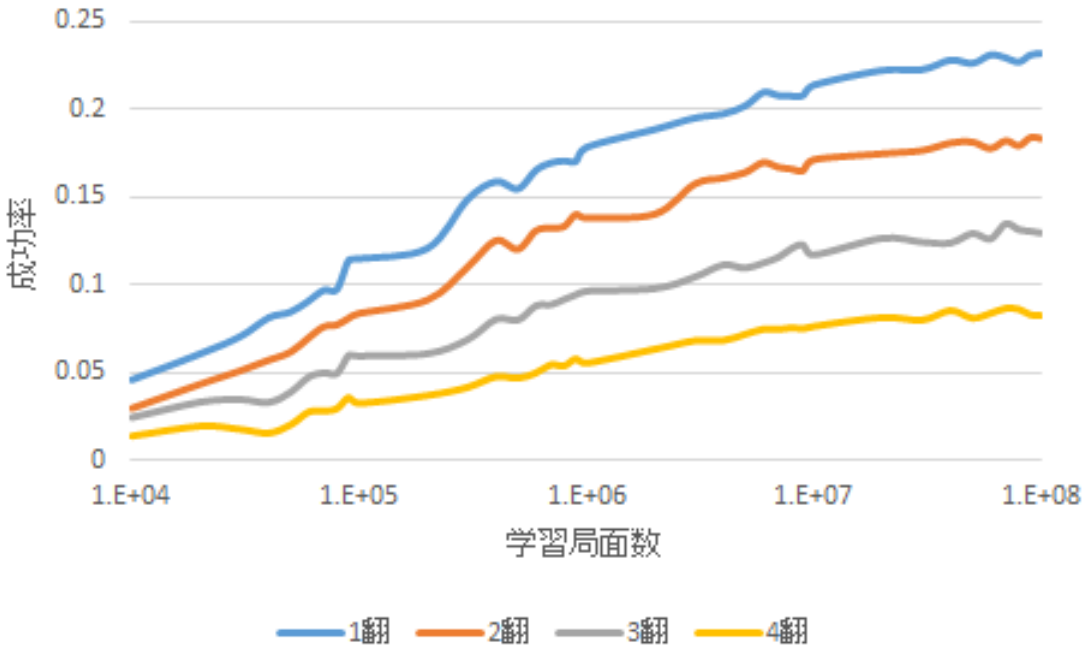

図 6.4. MCP における局面数と各翻数の成功率

6.4.4 最終的な順位を考慮した和了を行う麻雀プログラム

得られた翻数予測モデルを使用して最終的な順位を考慮した和了を行う麻雀プログラムを構築する．これを実現するための評価値は各翻数を和了できる確率に和了した時の期待最終順位（**Expected Final Rank, EFR**） [67] の総和に基づく．和了した際の点数の評価は重要ではあるものの，麻雀においては和了率は 2 割程度であり，8 割近くの場合和了できない．和了できなかった場合の行動をどのように評価するべきかという問題は相手の手牌や戦略に依存するため，自明な解決方法は存在しない．

この問題を解決する単純な方法は，点数移動が行われなかったと仮定して，その時の期待最終順位を評価とする方法が考えられる．しかし自分が和了できないときには他のプレイヤが和了しているため，期待最終順位は悪化するケースが多い．さらにこの方法では相手がツモ和了や自分が相手に放銃する可能性を無視しており，現実の麻雀を反映しているとはいえない．

そこで本研究では点数移動が行われなかったとするのではなく，現実的に起こりうるすべての点数移動を考慮し，その時の期待最終順位を和了できない時のペナルティとして扱う．具体的には，麻雀の相手のツモ和了や放銃など，1 局で起こりうるすべての局終了時の点数状況を考慮し，その時の期待最終順位とその状況が起きる確率との積和をペナルティとする．このペナルティの計算には和了できない時の各点数状況の確率を推定する必要がある．本研究では二種類の牌譜からこれらの確率を求めた．一つ目は人間の牌譜と二つ目は翻数予測モデルの構築に使用した牌譜である．

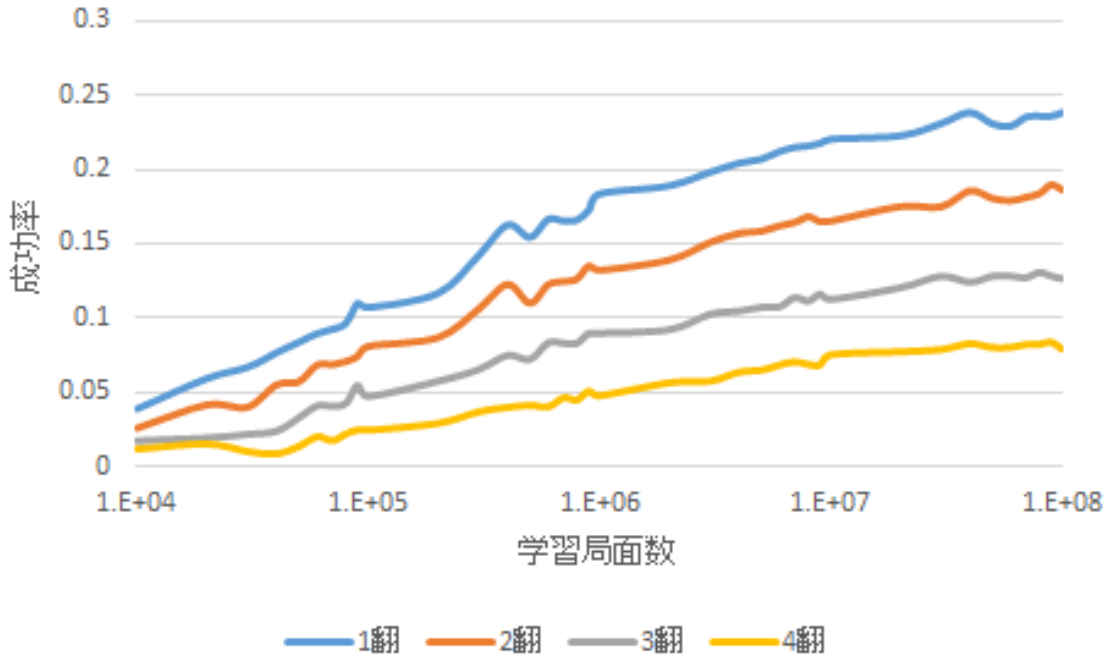

図 6.5. BCP における局面数と各翻数の成功率

このペナルティを考慮した和了を行うためのスコアの計算式は以下のようになる．

$$
\begin{aligned}
&Score(\mathbf{x}) \\
&= \sum_{i=1}^{4} \sum_{j \in J} P(i,j|\mathbf{x}) \times EFR(\tilde{\mathbf{z}}) \\
&+ (1 - P(1|\mathbf{x})) \times Penalty(z)
\end{aligned}
$$

ここで J はロン和了とツモ和了の集合，i は和了した翻数である．$\tilde{\mathbf{z}}$ は i 翻で j の種類で和了した時の全プレイヤの点数とする．$P(i,j|\mathbf{x})$ は手牌 $\mathbf{x}$ が与えられた時の i 翻で j の種類で和了する確率であり，翻数予測モデルを基に算出される．各プレイヤに対するロン和了とツモ和了が起きる確率は簡単のため同じとする．すなわちロン和了が起こる確率の合計はツモ和了の 3 倍とした．関数 EFR はこの引数の時の点数状況における期待最終順位を返す．符は頻出頻度の高い 30 符とする．第二項は和了できない確率に前述したペナルティ項を示している．z は現在の点数状況であるため，$Penalty(z)$ は固定値でなく局ごとに異なる値を持つ．

MCP では $P(i|\mathbf{x}) = P_{mc}(i|\mathbf{x})$ とし，BCP は，$P(i|\mathbf{x})$ を以下のように置き換える．

$$
P(i|\mathbf{x}) = \begin{cases} 1 - P_{bc}(i|\mathbf{x}) & \text{if } i = 0 \\ P_{bc}(i|\mathbf{x}) - P_{bc}(i+1|\mathbf{x}) & \text{otherwise} \end{cases}
$$

ただし $P(5|\mathbf{x}) = 0$ とする．

ここからは以前の筆者らの研究 [67] に用いた麻雀プログラムとこの翻数予測モデルを組み合わせた提案手法の麻雀プログラムについて説明する．以前の筆者らの研究のプレイヤは序盤と中終盤において手を決定するアルゴリズムが異なる．序盤は一局開始時から以下の条件のいずれかを満たすまでと定義する．

- プレイヤがリーチ可能な手牌であるとき
- 相手プレイヤがリーチを宣言しているとき
- 一人麻雀政策に従った時の放銃率が 2
- 一人麻雀政策に従った時の期待最終順位が 0.01 悪化するとき
- ツモが可能な牌の数が 16 枚以下

現状において問題となるのは序盤による手作りであるため，序盤のアルゴリズムのみを今回の手法に置き換える．

6.4.5 一人麻雀の探索

前節で述べたように，翻数予測モデルと期待最終順位を組み合わせることで将来の報酬に基づいた手の選択が可能である．ここでは現在の手牌の報酬をより精度高く評価するため疑似的な麻雀のゲーム木を探索する．

本来の麻雀のゲーム木を探索するには自分プレイヤのほかに 3 人の相手プレイヤの手牌や行動を考慮する必要がある．このような設定ではゲームのノードが膨大になり，現実的な時間内での有効な探索は行えない．そこで疑似的な麻雀のゲーム木では自分プレイヤのツモと打牌のみ（鳴きは行わない）を考慮する．ゲーム木のノードには二種類のノードが存在する．本研究では一つ目をツモ局面の**ツモノード**と二つ目を打牌後の**打牌ノード**と呼ぶ．二つのノードの評価値は以下の式で表す．

$$V_{tsumo}(\mathbf{X}, depth) =$$

$$\begin{cases} EFR(\mathbf{z}) & \text{if } \mathbf{X} = win \\ max[V_{move}(\mathbf{X} - Tile, depth), Tile \in \mathbf{X}] & \text{otherwise} \end{cases}$$

$$V_{move}(\mathbf{X}, depth) =$$

$$\begin{cases} Score(\mathbf{X}) & \text{if } depth\text{=Max depth} \\ \sum_{Tile \in Tiles} P(Tile) \times V_{tsumo}(\mathbf{X} + Tile, depth + 1) & \text{otherwise} \end{cases}$$

ここで V_{tsumo}, V_{move} はツモノードと打牌ノードの評価値，$\mathbf{X}$ は手牌，$Tiles$ は 34 種類の牌，$P(Tile)$ は $Tile$ をツモる確率である．$P(Tile)$ は見えていない牌を数え上げることで求める．$depth, Max\ depth$ は探索を制御するパラメータであり，それぞれ現在の深さ，打ち切りまでの深さを表す．ツモノードであるルートノードの $depth = 0$ とする．

ツモノードの評価値は和了した場合その時の期待最終順位とし，それ以外は手牌の中で最も高い打牌ノードの評価値となる．打牌ノードの評価値は終端ノードであれば前章の最終的な順位を考慮した和了を行う式により算出され，それ以外はある牌をツモったツモノードの評価値とそのツモ確率の積和とする．

上記のゲーム木探索を行うと現実的な計算時間では $Max\ depth = 2$ が限界であった．そこでより深い探索を可能にするため枝刈りを行う．枝刈りは手牌を切る局面とツモ番の二つの場面においてそれぞれ異なる基準を設ける．手牌を切る局面では全ての手牌について評価するのではなく，翻数予測モデルの上位 3 つまでを探索する．ツモ番ではシャンテン数を減らす牌と孤立牌の関連牌のみを探索する．関連牌とは数字の前後二つの牌のことを指す．これ以外の牌はツモ切りが起こるとして，その場合のノードの評価値を 0 とした．そして得られた積和の値を正規化するために評価値をツモ切りが起きなかった確率で割りツモノードの評価値とした．これらの枝刈りにより現実的な計算時間に深さ 3 まで評価することが可能になった．

6.5 対戦実験と結果

本章では提案手法によって得られた麻雀プログラムの強さを対戦によって評価する．以前の麻雀プログラムとの相違点は序盤の戦略である．以前の麻雀プログラムは一人麻雀政策に対して，本研究での序盤は翻数予測モデル（MCP や BCP）を用いる．

6.5.1 自己対戦における設定

自己対戦では翻数予測モデルを用いた麻雀プログラム 1 体と参考文献 [67] の麻雀プログラム 3 体で対局を行う．一ゲームは東風戦で行われる．シミュレーションにかける時間は中終盤では 1 手 1.5 秒とする．また序盤の一手毎の時間は以前の麻雀プログラムは 1 秒未満である．また本研究の序盤の麻雀プログラムの時間も 2 秒で計算できるであろう探索の深さとした．

6.5.2 自己対戦における結果

麻雀プログラムについて説明する．Penalty（自己対戦）とは自己対戦により生成された牌譜を使ってペナルティを計算した麻雀プログラムである．また Penalty（人間）とは天鳳の牌譜から特定のプレイヤが和了できなかった時の条件におけるペナルティを計算したものである．探索を行う際の深さは 3 であり，その他の麻雀プログラムは深さが 1 である．

結果を表 6.5 に示す．BCP と比較して人間の牌譜を用いたペナルティや探索を加えることで実力が向上した．しかしながらいずれの麻雀プログラムも以前の麻雀プログラムと比較して大きく負け越している．特に自己対戦からのペナルティを用いた麻雀プログラムは大きく実力が下がった．これは自己対戦の麻雀と現実の麻雀を比較すると特定の事象の確率が乖離しているためである．すなわち局面生成時に相手がツモ切りや一人麻雀政策しか選択しないのは相手プレイヤの戦術としては単

純である．これらの戦術では降りなどを行わないため，相手の和了率が上昇しそれにより流局率が下がるといった現実の麻雀と大きな隔たりが生じる．このため自己対戦におけるペナルティが適切に機能しなかったと考えられる．

和了・放銃は表 6.6 に示す．これらは 1 局のプレイヤの強さを測定するためによく用いられている．相手をツモ切りから一人麻雀政策と強くすることで得られる麻雀プログラムの和了率も向上している．しかしながら対戦相手の和了率と比較して大きな差をつけられている．このことから翻数予測モデルを組み込むことで単純な牌効率が悪化しておりその結果，平均順位も大きく悪化したのではないかと考えられる．

6.5.3 考察

牌効率が悪化しているとわかる典型的な例は図 6.6 の手牌である．この手牌の BCP による評価値を表 6.7 にまとめた．この手牌から 3 翻以上の高い手を目指す場合は，人間なら索子の混一色にすることを考える．BCP においても同様に考え，萬子や筒子を切る手を高く，字牌を切る手は価値が低いと評価している．反対に 1 翻などの安い手を和了しようとすると，人間では 78p を残し白を刻子にすることを考える.[2]．その点は 7p や白を切ったときの評価が低いことから BCP も理解している．切るべき牌を考えた時に北は 5m に比べ両面になることがないためこの手牌では 1 翻を和了するためには一番価値が低いと人間は考える．しかしながら BCP は北よりも 5m を切る方が価値が高いと考えている．このケースではターツ候補が揃っており 5m を切ってしまったために和了できないケースが少なくそこが上手く学習できないことが原因であろう．

翻数予測モデルの予測と実際の局面との比較を表 6.8 にまとめた．図 6.6 の手牌から全員が一人麻雀政策に従った時の特定の翻数以上を実際に和了できた確率を求めた．確率を近似的に求めるため各候補手に対して一万回試行を行った．各施行ごとに相手の手牌や山をランダムに変更した．結果である表 6.8 と表 6.7 を比較すると BCP の値は小さい．局面生成時にランダムな手を混ぜたことにより，和了しにくい手牌として学習されたことが考えられる．

実際の対局の点数状況を考慮した打牌の評価を表 6.8 にまとめた．現状の一人麻雀政策の問題点は点数状況に応じた手を選択できないことであった．そこでオーラスで満貫以上を和了できなければ最下位という状況で逆転する手を作れるかを調べた．オーラス最下位時において図 6.6 の手牌では一人麻雀政策では評価値の最も高い北を選択するが，提案手法では期待最終順位の最も低い 7p を選択する．表 6.8 を参照すると，7p が 4 翻以上を和了する確率が高いため，高い点数を和了するという目的としては提案手法が効果的に機能している．

もう一つの例は図 6.7 の手牌である．この例は翻数予測モデルが役を認識できていない例である．状況はこの手牌において北が切られ，鳴くかどうかを考慮する局面である．一巡目でシャンテン数が 1 であるため手牌自体はよく，パスをすることで和了できる確率が高いのも納得できる．ポンをすることに関して北は役牌ではないので，鳴いても役がないことは明らかである．しかしながら翻数予測モデルでは 1 翻以上で和了できる確率が 0 になっていない．このように牌効率が悪くなった要因は役がない鳴きが頻出していることも考えられる．

[2]p はピンズ,m はマンズである

図 6.6. 問題のある手牌

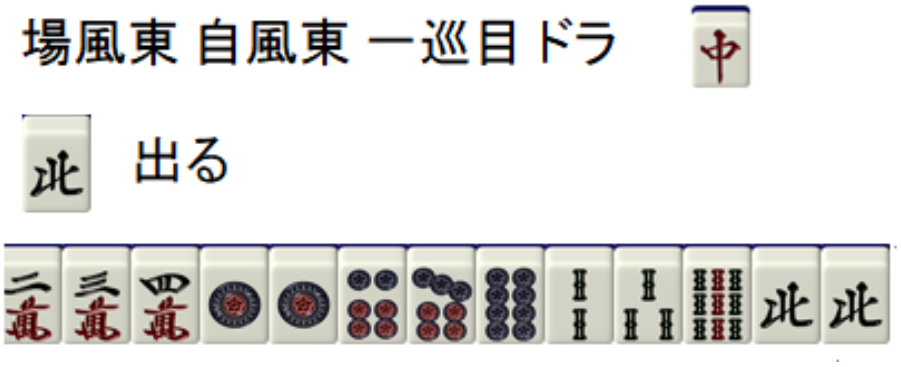

図 6.7. 鳴き局面における手牌

6.5.4 まとめ

本章では報酬が疎である麻雀に対して，これまでの章の結果を利用し強化学習を行った．前章の結果から期待最終順位を用いることで得点を疑似的な報酬として扱うことに成功している．また 1 局単位で見ても得点を得ることも疎なゲームではあるが，一人麻雀政策を用いることでランダムに探索を行うよりも早く確実に報酬を獲得することが可能になってた．この結果から報酬が疎なゲームにおける報酬を得ることの困難さについては克服したといえる．しかしながら得られた疑似的な報酬をもとに有効な学習を行う点はうまくいかなかった．原因は局面の探索を有効に活用しなかったことと考えられる．麻雀は局面遷移が既知であるため，学習局面の生成や実際のプレイにおいて Atari と異なりゲーム木の探索を行うことが比較的容易である．不完全情報ゲームであるため相手の手牌などの推察は困難であるが，本研究では序盤のみのを対象にしているため自分の手牌だけに限定した探索だけでも有効である．そのため本章では学習局面の生成に探索を行わなかったため，報酬が疎なゲームにおいて報酬を得ることに成功しても結果的には元の AI の実力を超えることがなかったと考えられる．

表 6.1. 一人麻雀プレイヤの特徴量

特徴量	次元数
通常手，七対子，国士無双の向聴数	$15+7+14=36$
副露数	5
向聴数，副露数	$15\times5=75$
リーチが可能か	2
向聴数，副露数，min(受け入れ枚数,20)	$15\times5\times21=1,575$
副露した種類	136
役牌の刻子の数	5
向聴数の悪化しない頭の数，役牌の対子の数	$6\times6=36$
役牌の刻子があるか，向聴数の悪化しない頭の数，役牌の対子の数， 浮いた役牌の数，min(向聴数, 4)，副露数	$2\times6\times6\times16\times5\times5=28,800$
色の中で最も多い色の数+染め役は不可能，副露数, 混一色または清一色	$(14+1)\times5\times2=150$
min(ドラ+赤ドラの数,3)	4
役がある，ない，片上がり	3
min(向聴数, 3)，役がある，ない，片上がり，巡目	$4\times3\times18=216$
min(向聴数, 3)，副露数，振聴か，min(役のある待ち牌の数,7)	$4\times5\times2\times8=320$
両面を優先した時の両面+面子の数，向聴数	$7\times15=105$
面子+ターツ+ターツ候補，向聴数	$11\times15=165$
両面を優先した時の両面+面子の数，面子+ターツ+ターツ候補，向聴数	$7\times11\times15=1155$
min(全帯幺九の向聴数, 4)，max(全帯幺九の枚数-6)，全帯幺九のメンツまたはターツ候補， min(受け入れ枚数/4,4)，min(全帯幺九の向聴数-[向聴数, 3)	$5\times8\times8\times5\times4=6,400$
min(全帯幺九の向聴数, 4)，max(全帯幺九の枚数-6)，2378 の暗刻があるか，副露数， 両面を優先した時の面子の数，両面を優先した時の両面の数，愚形の数， 面子の減らない 19 字牌の頭の数，面子の減らない 2378 字牌の頭の数	$5\times2\times5\times5\times4\times8\times2\times2=32,000$
ドラの種類 (19, 28, 37, 46, 5, 役牌, オタ風)，ドラの数， 見えているドラの数，現在の巡目/2，赤ドラの数	$7\times4\times4\times8\times4=6,400$
min(全帯幺九の向聴数, 4)，max(全帯幺九の枚数-6)，全帯幺九のメンツまたはターツ候補， min(受け入れ枚数/4,4)，min(全帯幺九の向聴数-[向聴数, 3)	$5\times8\times8\times5\times4=6,400$
両面を優先した時の両面+面子の数，愚形の数，両面対子の数，愚形対子の数，浮き牌があるか， 暗刻があるか，頭の数， min(向聴数, 3)，完全一, 二向聴, そうでない，リーチが可能か，巡目/3	$8\times8\times4\times4\times2\times2\times4\times4\times3\times2\times7$ $=2,752,512$
面子を優先した時の両面+面子の数，愚形の数，両面対子の数，愚形対子の数， 浮き牌があるか，暗刻があるか，頭の数， min(向聴数, 3)，リーチが可能か，巡目/3	$8\times8\times4\times4\times2\times2\times4\times4\times2\times7$ $=917,504$
min(色の中で最も多い色の数の向聴数,4)，両面を優先した時の面子の数， 両面対子+愚形対子の数，副露数	$5\times5\times8\times8\times5=8,000$
min(19 字牌抜いた時の向聴数,4)，両面を優先した時の面子の数， 両面を優先した時の面子+愚形の数，タンヤオのドラの数，副露数，巡目/3， タンヤオの向聴数=向聴数か，min(19 字牌の受け入れ枚数,2) ，max(タンヤオ牌-11,0)	$5\times5\times8\times4\times5\times6\times2\times3\times3=432,000$
両面を優先した時の両面+面子の数，愚形の数，七対子の向聴数， min(向聴数, 3)，完全一, 二向聴, そうでない，リーチが可能か， 浮き牌の種類 (19,28,34567, 字牌)，その浮き牌の枚数	$8\times8\times8\times4\times3\times2\times4\times4$ $=196,608$
両面を優先した時の両面+面子の数，愚形の数，二度受けの両面の数，二度受けの愚形の数， min(向聴数, 3)，完全一, 二向聴, そうでない，リーチが可能か	$8\times8\times4\times4\times4\times3\times2=24,576$
min(向聴数, 4)，七対子の向聴数，向聴数の悪化しない頭の数， 両面を優先した時の両面+面子の数，リーチが可能か，完全一, 二向聴, そうでない，	$5\times8\times8\times8\times2\times3=15,360$
min(向聴数, 3)，役牌の刻子があるか，役牌の対子があるか，両面を優先した時の面子の数， 両面+愚形，向聴数の悪化しない頭の数，min(副露数,3)， 役がある，ない，片上がり	$4\times2\times2\times5\times8\times4\times4\times4\times3$ $=122,880$
両面を優先した時の両面+面子の数，愚形の数，浮き牌の最も外側の種類 (19,23,3456, 字牌)， 頭と頭の組み合わせ，頭の数，完全一, 二向聴, そうでない	$8\times8\times5\times16\times4\times3=61,440$
min(向聴数, 4)，七対子の向聴数，向聴数の悪化しない暗刻の数，副露数， チーがあるか，完全一, 二向聴, そうでない，	$5\times8\times5\times5\times2\times3=6,000$

表 6.2. 一人麻雀プレイヤの特徴量

各字牌に対して見えている数，持っている数，ドラか，max(色の中で最も多い色の数-6,0), 両面を優先した時の両面+面子の数，max(巡目,8)，字風，場風，東南西北+三元牌	$5\times5\times2\times8\times8\times8\times4\times35\times5$ $=1,536,000$
各数牌に対して数牌の種類 (19,28,37,46,5)，持っている枚数，ドラとの近さ (0,1,2,3, 違う色)	$5\times5\times5=125$
連続する n 種類の数牌の持っている枚数の組み合わせ（n=2〜6），リーチが可能	$(100+500+1860+8634+23760)\times2=69,708$
各色の 1 から 9 の組み合わせ．各数字は最高で 2	$19,472$
暗刻の数，対子の数，刻子にならない対子の数	$5\times7\times2=70$
刻子の数，対子の数，刻子にならない対子の数+対々和ができない	$5\times7\times2+1=71$
min(タンヤオの向聴数, 4)，min(タンヤオの向聴数-向聴数, 3), max(タンヤオの枚数-9, 0)，副露数，max(タンヤオの頭,3)+タンヤオができない	$5\times4\times5\times4+1=401$
ドラの数，タンヤオのドラ，min(タンヤオの向聴数, 4), min(タンヤオの向聴数-向聴数, 3)，タンヤオができるか	$4\times4\times5\times4\times2=640$
タンヤオができるか，全帯幺九ができるか，min(19 字牌の受け入れ枚数,3), あリーチができるか，副露数	$2\times2\times4\times2\times5=160$
三色に最も近い枚数，向聴数，副露数	$10\times14\times5=700$
各三色の可能性について, 123,789 か，各数字を持っているか，min(向聴数, 3), 三色に近づく受け入れがあるか，リーチができるか	$2\times512\times4\times2\times2=16,384$
各一通の可能性について, 各数字を持っているか，min(向聴数, 4)，副露数, 両面+面子+愚形 ≧5，両面+面子 ≧4，完全一, 二向聴, そうでない	$512\times5\times5\times2\times2\times3=153,600$
一通に最も近い枚数, 面子，両面，愚形-頭の数，頭があるか, 副露数，一通に近づく受け入れがあるか	$10\times5\times8\times8\times2\times5\times2=64,000$
各風牌の枚数, 最高 3 枚	$4\times4\times4\times4=256$
各三元牌の枚数, 最高 3 枚	$4\times4\times4=64$
各和了牌について, 枚数，翻数，巡目/3	$4\times9\times7=252$
各和了牌について, 牌の種類 (19,28,37,46,5, ダブ東南, 役牌, オタ風)，枚数, 翻数，ツモとロンでの翻の差，リーチか，ドラ待ちか，筋待ちか，フリテンか	$8\times4\times8\times3\times2\times2\times2\times2=12,288$
min(役ありの和了牌数,9)，min(役なしの和了牌数,5)，副露数，七対子または国士無双か	$10\times6\times5\times2=600$
min(一向聴時の受け入れ,31)，副露数，完全一, 二向聴, そうでない	$32\times5\times3=480$
4 色の選んだ 3 色の受け入れ枚数 (最大 20) までの組み合わせ	$20+231+1771=480$

表 6.3. 人間の牌譜との一致率

局面の種類	牌譜の数	完全一致数	鳴きのみ正解数	正解率
ツモ局面	1,140,576	859,088	N/A	75.3
鳴く局面	68,397	46,945	11,731	85.8
鳴かない局面	252,666	240,450	N/A	95.1

表 6.4. 成功率

モデル	相手	1 翻	2 翻	3 翻	4 翻
一人麻雀政策	ツモ切り	0.6411	0.5235	0.3169	0.1708
MCPvs ツモ切り	ツモ切り	0.6574	0.545	0.3663	0.2274
一人麻雀政策	一人麻雀政策	0.2446	0.1979	0.1278	0.0734
MCP	一人麻雀政策	0.2318	0.1833	0.1296	0.0825
BCP	一人麻雀政策	0.2382	0.1863	0.1267	0.0793

表 6.5. 順位分布

	1 位率	2 位率	3 位率	4 位率	平均順位	試合数
MCPvs ツモ切り	0.185	0.251	0.282	0.280	2.65 ±0.01	30505
MCP	0.180	0.248	0.292	0.276	2.67 ±0.01	62742
BCP	0.194	0.253	0.283	0.270	2.62 ±0.01	44550
BCP+Penalty (自己対戦)	0.121	0.179	0.324	0.375	2.95 ±0.05	1163
BCP+Penalty (人間)	0.227	0.244	0.262	0.264	2.56 ±0.04	3589
BCP+Penalty (人間)+探索 (depth=3)	0.240	0.244	0.250	0.265	2.54 ±0.02	14804

表 6.6. 和了・放銃率

	和了率	放銃率	相手の平均和了率	相手の平均放銃率
MCPvs ツモ切り	0.193	0.118	0.217	0.119
MCP	0.194	0.114	0.215	0.120
BCP	0.201	0.115	0.214	0.119
BCP+Penalty (自己対戦)	0.093	0.114	0.235	0.110
BCP+Penalty (人間)	0.202	0.123	0.226	0.123
BCP+Penalty (人間)+探索 (depth=3)	0.195	0.128	0.220	0.123

表 6.7. BCP による図 6.6 の評価値

牌	1 翻以上	2 翻以上	3 翻以上	4 翻以上
5m	0.353	0.338	0.221	0.111
7p	0.294	0.303	0.240	0.136
北	0.328	0.285	0.161	0.079
白	0.173	0.168	0.149	0.063

表 6.8. 実際の対局による図 6.6 の評価値

牌	1 翻以上	2 翻以上	3 翻以上	4 翻以上
5m	0.401	0.373	0.192	0.117
7p	0.325	0.323	0.298	0.203
北	0.405	0.388	0.241	0.107
白	0.203	0.201	0.164	0.080

表 6.9. オーラス最下位時の図 5.2 の BCP と一人麻雀政策の評価値

牌	BCP の評価値	一人麻雀政策の評価値
5m	3.889	-12447
7p	3.797	-12587
北	3.93	-12326
白	3.94	-12595

表 6.10. 1 翻以上の評価値

行動	1 翻以上
パス	0.821
ポン かつ 9s	0.161

第7章　おわりに

7.1　本研究のまとめ

本研究では，報酬が疎なゲームに対して強化学習を用いたプレイヤを構築した．報酬が疎なゲームを局面遷移が未知と不完全情報ゲームという点で分け，それぞれに対して AI を構築した．手法としてはそれぞれのゲームに対して疑似的な報酬を設計し，それを利用した学習を行うことである．

局面遷移が未知の環境である Atari では探索 UCB ボーナスを定義した．これは局面の訪問回数とトレーニングの進捗度の組み合わせることで，その局面を訪問することで得られる情報の価値として解釈することが可能である．これにりエージェントは “Private Eye” と “Solaris” において比較した手法の中で最も高いスコアをとるようになった．

局面遷移が未知の環境ではゲーム木の探索が困難であるためほかの研究 [10, 40] でも示すように現在の局面の表現方法が重要である．そのため探索 UCB ボーナスが効果的に働いた．さらにほかの研究とは異なる活用戦略の学習方法がは短期的な負の報酬の影響を受けず，長期的な大きな正の報酬をとる戦略の獲得につながった．ただこの手法によって活用戦略の理論的な最適性の保証を与えられるかは不明である．しかしながら局面から未知さを数値化し疑似的な報酬とする手法であるため Atari のなかの複数のゲームに対して優れた結果を出すなど一般的に適応可能な手法である．

不完全情報ゲームである麻雀ではゲームの途中経過である点数状況から最終的な報酬を予測するモデルを構築し，その出力を疑似的な報酬とした．これを基に一人麻雀の手を使い局面を生成し，手牌から各翻数を和了する確率を求め，期待最終順位と組み合わせることで最終的な順位を 1 局の序盤から考慮するプレイヤの構築に成功した．結果として得られた AI は牌譜を用いて相手の抽象化や期待最終順位を出力するモデルの構築に成功し，シミュレーションを組み合わせることで人間のトッププレイヤに並ぶ実力になった．また自己対戦による牌譜から局の報酬を予測するモデルによって高い翻数を和了するモデルを構築することに成功した．しかしながらそれを組み込んだ AI 自体は元の AI に勝ち越すことは出来なかった．

麻雀に対して行われた手法は人間の知識や牌譜を用いるため，実力が局所的な解になっていることを否定できない．また麻雀に関する専門知識を使っているため，ほかのゲームに対して直接的に適応できることは困難である．しかしながら背後にある技術は他のゲームの問題を解決するのに役立つかもしれない．例えば，対象となるゲームが繰り返しゲームであるならばサブゲームの開始地点と最終結果を紐付け学習することは有用であると期待できる．また対象となるゲームが局面遷移の既知であるゲームである場合，本研究の結果から得られる知見がある．それは局面遷移の既知であるゲームである場合は探索を行うことが重要であるということである．AlphaZero [46] は自己対戦時に探索を行い手を決定する．そのため学習が進むほど，深く探索した結果が現在の局面に反映さ

れる．これにより AlphaZero は成功したと考えられる．

完全情報ゲームに限らず栗田ら [71] の研究で示されている通り麻雀においても探索というのは有効である．本研究でも，打牌決定時に探索を加えることで実力の向上が示唆されている．しかしながら局面生成時の手の決定に一人麻雀政策を用いるのみ一手のみの探索でゲーム木の深い探索は行っていない．そのため局面の探索せずに学習した翻数予測モデルは一人麻雀政策と比較して出力の形式は変更したが，出力自体は未来の局面からの情報を現在の局面に反映させる学習にはなっていない．そのため得られた翻数予測モデルは一人麻雀政策と比較して強くならなかったのであろう．

7.2 今後の課題

Atari においては報酬が疎なゲームにおいてはある程度成功したので報酬が密なゲームにおいても有効に働くかを検証を行う必要がある．またハイスコアを記録して使用する方法は学習速度を高める意味で効果的であったと考えられるのでこれの改良を考えたい．例えばハイスコアだけでなく上位いくつかを記憶しておくとか，その中でも途中から履歴を使用しないとかすることである程度は無駄な局面を探索することを防ぎつつ有効な局面を探索することが可能になると考えられる．

麻雀に関しては現状では高い翻数を和了する技術は向上したが反対に低い翻数を和了する技術は低下している．麻雀においては和了率が実力に影響するためこれが自己対戦で勝ち越せない原因と考える．局面の生成の方法は適当な巡目でランダムな手を指す，これでは鳴きの局面がツモ局面に比べ少ない．また鳴いてランダムに切るため，鳴いたのが悪かったのか鳴いて切った牌が悪いのか理解するの時間がかかり，学習が上手くいかなかった可能性がある．そこでシャンテン数が下がる牌が切られた場合にはその局面をサンプルする局面とし，鳴くかどうかだけをランダムにして局面生成することでうまくいくと考える．現状では序盤の戦略だけではあるがこの先には全体を最適化を行いたい．

参考文献

[1] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. 47th Annual IEEE Symposium on*, pp. 459–468, 2006.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems 30*, pp. 5048–5058. 2017.

[3] Petr Baudiš and Jean-loup Gailly. Pachi: State of the art open source go program. In *Advances in Computer Games*, pp. 24–38. Springer, 2011.

[4] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, Vol. 47, pp. 253–279, 2013.

[5] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems 29*, pp. 1471–1479, 2016.

[6] David Blackwell, et al. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, Vol. 6, No. 1, pp. 1–8, 1956.

[7] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold ' em poker is solved. *Science*, Vol. 347, No. 6218, pp. 145–149, 2015.

[8] Michael Bowling, Nicholas Abou Risk, Nolan Bard, Darse Billings, Neil Burch, Joshua Davidson, John Hawkin, Robert Holte, Michael Johanson, Morgan Kan, et al. A demonstration of the polaris poker system. In *Proceedings of The 8th International Conference on Autonomous Agents and Multi-agent Systems-Volume 2*, pp. 1391–1392, 2009.

[9] George W Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, Vol. 13, No. 1, pp. 374–376, 1951.

[10] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[11] Michael Buro. Logistello: A strong learning othello program. In *19th Annual Conference Gesellschaft für Klassifikation eV*, pp. 1–3. Citeseer, 1995.

[12] Michael Buro. Experiments with multi-probcut and a new high-quality evaluation function for othello. *Games in AI Research*, pp. 77–96, 1997.

[13] Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. Deep blue. *Artificial intelligence*, Vol. 134, No. 1, pp. 57–83, 2002.

[14] Chih-Chung Chang and Chih-Jen Lin. Libsvm. `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`, 2001.

[15] Christopher Clark and Amos Storkey. Training deep convolutional neural networks to play go. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1766–1774, 2015.

[16] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 1–8. Association for Computational Linguistics, 2002.

[17] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *In Proceedings of the 5th International Conference on Computer and Games*, pp. 72–83. Springer, 2006.

[18] Aaron Davidson. Opponent modeling and search in poker: Learning and acting in a hostile and un-certain environment. Master's thesis, University of Alberta, 2002.

[19] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, Vol. 12, pp. 2121–2159, 2011.

[20] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *The Journal of Machine Learning Research*, Vol. 10, pp. 2899–2934, 2009.

[21] Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 805–813, 2015.

[22] Johannes Heinrich and David Silver. Smooth uct search in computer poker. In *Proceedings of the 24th International Joint Conference on Artifical Intelligence*, 2015.

[23] kmo2. まったり麻雀. `http://homepage2.nifty.com/kmo2/`, 2015.

[24] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Proceedings of 17th European Conference on Machine Learning*, pp. 282–293, 2006.

[25] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, Vol. 6, No. 1, pp. 4–22, 1985.

[26] Carol Luckhart and Keki B Irani. An algorithmic solution of n-person games. In *AAAI*, Vol. 86, pp. 158–162, 1986.

[27] Peter Bro Miltersen and Troels Bjerre Sørensen. A near-optimal strategy for a heads-up no-limit Texas hold'em poker tournament. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pp. 191–198, 2007.

[28] Naoki Mizukami and Yoshimasa Tsuruoka. Building a computer mahjong player based on monte carlo simulation and opponent models. In *Computational Intelligence and Games*, 2015.

[29] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*. 2013.

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, Vol. 518, No. 7540, pp. 529–533, 2015.

[32] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. In *ICML Deep Learning Workshop*, 2015.

[33] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning*, pp. 807–814, 2010.

[34] John Nash. Non-cooperative games. *Annals of mathematics*, pp. 286–295, 1951.

[35] John F Nash, et al. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, Vol. 36, No. 1, pp. 48–49, 1950.

[36] Eric W Noreen. Computer-intensive methods for testing hypotheses: an introduction. *Computer*, 1989.

[37] Vinyals Oriol, Babuschkin Igor, Chung Junyoung, Mathieu Michael, Jaderberg Max, Czarnecki Wojtek, Dudzik Andrew, Huang Aja, Georgiev Petko, Powell Richard, Ewalds Timo, Horgan Dan, Kroiss Manuel, Danihelka Ivo, Agapiou John, Oh Junhyuk, Dalibard Valentin, Choi David, Sifre Laurent, Sulsky Yury, Vezhnevets Sasha, Cai James, Molloy amd Trevor, Budden

David, Paine Tom, Gulcehre Caglar, Wang Ziyu, Pfaff Tobias, Pohlen Toby, Yogatama Dani, Cohen Julia, McKinney Katrina, Smith Oliver, Schaul Tom, Lillicrap Timothy, Apps Chris, Kavukcuoglu Koray, Hassabis Demis, and Silver David. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/, 2019.

[38] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances In Neural Information Processing Systems 29*, pp. 4026–4034, 2016.

[39] Nikolaos Papahristou and Ioannis Refanidis. Opening statistics and match play for backgammon games. In *Artificial Intelligence: Methods and Applications*, pp. 569–582, 2014.

[40] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 2778–2787, 2017.

[41] Jonathan Rubin and Ian Watson. Computer poker: A review. *Artificial Intelligence*, Vol. 175, No. 5, pp. 958–987, 2011.

[42] Ariel Rubinstein. Equilibrium in super games with the overtaking criterion. *Journal of Economic Theory*, Vol. 21, No. 1, pp. 1–9, 1979.

[43] Maarten PD Schadd and Mark HM Winands. Best reply search for multiplayer games. *Computational Intelligence and AI in Games, IEEE Transactions on*, Vol. 3, No. 1, pp. 57–66, 2011.

[44] Statistical Concepts Series. Measurement of observer agreement. *Radiology*, Vol. 228, pp. 303–308, 2003.

[45] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, Vol. 529, No. 7587, pp. 484–489, 2016.

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

[47] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Learning by playing - solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.

[48] David Stern, Ralf Herbrich, and Thore Graepel. Bayesian pattern ranking for move prediction in the game of go. In *Proceedings of the 23rd international conference on Machine learning*, pp. 873–880. ACM, 2006.

[49] Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, Vol. 74, No. 8, pp. 1309–1331, 2008.

[50] Nathan R Sturtevant and Richard E Korf. On pruning techniques for multi-player games. *AAAI/IAAI*, Vol. 49, pp. 201–207, 2000.

[51] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, Vol. 3, No. 1, pp. 9–44, 1988.

[52] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. *arXiv preprint arXiv:1611.04717*, 2016.

[53] Gerald Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, Vol. 6, No. 2, pp. 215–219, 1994.

[54] Albert William Tucker. *Contributions to the Theory of Games*, Vol. 4. Princeton University Press, 1959.

[55] AAJ Van Der Kleij. Monte carlo tree search and opponent modeling through player clustering in no-limit texas hold ' em poker. Master's thesis, University of Groningen, 2010.

[56] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pp. 2094–2100, 2016.

[57] Dwango Media Village. `https://dmv.nico/ja/articles/mahjong_ai_naga/`, 2019.

[58] Wang Ziyu, Freitas Nando de, and Lanctot Marc. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 1995–2003, 2016.

[59] とつげき東北. 保障安定レーティング. `http://totutohoku.b23.coreserver.jp/hp/SLtotu14.htm`, 2001.

[60] とつげき東北. Mjsim0. `http://totutohoku.b23.coreserver.jp/hp/MJSIM0.htm`, 2003.

[61] 福地誠とつげき東北. おしえて!科学する麻雀, 2009.

[62] 我妻敦, 原田将旗, 森田一, 古宮嘉那子, 小谷善行. Svr を用いた麻雀における捨て牌の危険度の推定. 情報処理学会研究報告. GI,[ゲーム情報学], Vol. 2014, No. 12, pp. 1–3, 2014.

[63] 古居敬大. 相手の抽象化による多人数ポーカーの戦略の決定. Master's thesis, 東京大学, 2013.

[64] 三木理斗. 多人数不完全情報ゲームにおける最適行動決定に関する研究. Master's thesis, 東京大学, 2010.

[65] 海津純平, 成澤和志, 篠原歩. 一人麻雀における打ち方を考慮した評価指標に関する研究. In *Proceedings of the 20th Game Programming Workshop*, pp. 172–178, 2015.

[66] 小松智希, 成澤和志, 篠原歩. 役を構成するゲームに対する効率的な行動決定アルゴリズムの提案. 情報処理学会研究報告. GI,[ゲーム情報学], Vol. 2012, No. 8, pp. 1–8, 2012.

[67] 水上直紀, 鶴岡慶雅. 期待最終順位の推定に基づくコンピュータ麻雀プレイヤの構築. In *Proceedings of the 20th Game Programming Workshop*, pp. 179–186, 2015.

[68] 水上直紀, 中張遼太郎, 浦晃, 三輪誠, 鶴岡慶雅, 近山隆. 多人数性を分割した教師付き学習による四人麻雀プログラムの実現. 情報処理学会論文誌, Vol. 55, No. 11, pp. 1–11, 2014.

[69] 天鳳. `http://tenhou.net/`, 2015.

[70] 田中哲朗, 金子知適. 4 大規模クラスタシステムでの実行: Gps 将棋の試み (¡ミニ特集¿ コンピュータ将棋の不遜な挑戦). 情報処理, Vol. 51, No. 8, pp. 1008–1015, 2010.

[71] 栗田萌, 保木邦仁. 有向非巡回グラフで表現された 1 人麻雀の探索アルゴリズム. In *Proceedings of the 22th Game Programming Workshop*, pp. 42–49, 2017.

[72] 北川竜平. 麻雀の牌譜からの打ち手評価関数の学習. In *Proceedings of 12th Game Programming Workshop*, 2007.

本研究に関する発表文献

査読付き論文誌

1. 水上 直紀, 鈴木 潤, 亀甲 博貴, 鶴岡 慶雅. 報酬が疎な環境に適した深層強化学習法, 情報処理学会論文誌, 2019 （印刷中）

2. 水上 直紀, 鶴岡 慶雅．自動対戦棋譜の教師あり学習による翻数予測に基づく麻雀プログラム, 情報処理学会論文誌 (投稿中)

査読付き国際会議論文

1. Naoki Mizukami, Jun Suzuki, Hirotaka Kameko, and Yoshimasa Tsuruoka. Exploration Bonuses Based on Upper Confidence Bounds for Sparse Reward Games, Fifteenth International Conference on Advances in Computer Games (ACG 2017) Exploration Bonuses Based on Upper Confidence Bounds for Sparse Reward Games. In: Winands M., van den Herik H., Kosters W. (eds) Advances in Computer Games. ACG 2017. Lecture Notes in Computer Science, vol 10664. pp. 165-175, Springer, Cham. 2017

査読付き会議論文 (Extended Abstract による)

1. 水上 直紀, 鶴岡 慶雅．強化学習を用いた効率的な和了を行う麻雀プレイヤ, 第 21 回ゲームプログラミングワークショップ（GPW2016）, pp.181-88 Nov. 2016.

2. 水上 直紀, 鶴岡 慶雅．期待最終順位の推定に基づくコンピュータ麻雀プレイヤの構築, 第 20 回ゲームプログラミングワークショップ（GPW2015）, pp.179-186 Nov. 2015.

メディア・雑誌出演

1. TOKO FM, ピートの不思議なガレージ, 2018
2. ITmedia NEWS, http://www.itmedia.co.jp/news/articles/1802/23/news023.html, 2018
3. NHK, 人間ってナンだ？超 AI 入門, 2017
4. 日経 BP 社, 日経トレンディ2017 年 11 月号, 2017

謝辞

本論文を書くにあたって多くの方々にお世話になりました．鶴岡慶雅准教授には勉強会などで研究についてのさまざまな指摘や参考となる論文紹介，論文の推敲，発表の方法などアドバイスをいただきました．

同期の江里口瑛子氏と亀甲博貴氏と橋本和真氏は研究のことはもちろんのこと授業である輪講や就職活動といった大学生活の情報を共有できたおかげで大学生活が快適に送ることができたこと思っています．特に亀甲博貴氏と後輩の河村圭悟氏との議論というよりは手助けによって研究の問題点が解消したことが多数ありました．ここには書ききれませんが鶴岡研究室の多くの皆様の支えにより学生生活を送ることが出来たことを感謝します．

明治大学の横山大作准教授には自己対戦に必要な大量の計算資源を数えきれない日数を提供していただきました．この計算資源がなければ今の研究は成り立っていません．

この研究を始めるきっかけになった，角田真吾氏には牌譜の公開や天鳳での対人戦を許可してくださり感謝しています．また実際に天鳳で対局してくださった皆様にも感謝しています．この御協力があってこの研究は完成したと思います．

最後にこのような研究の機会を与えてくださった家族に感謝します．