修　　士　　論　　文

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

Research on decision making in multi-player games with imperfect information

指導教員　　近山 隆 教授

東京大学大学院
工学系研究科
電気系工学専攻

氏 名　　37-086521 三木 理斗

提 出 日　　平成 22 年 2 月 9 日

概 要

多人数不完全情報ゲームでは，囲碁や将棋などの二人完全情報ゲームで有効であるとされている従来手法をそのまま適用することが困難であることが多い．本研究では，多人数不完全情報ゲームである麻雀の打牌を決定するための手法として，UCT 探索を利用した打ち手の探索と，大量の補助問題を利用した評価関数の学習を行った．UCT 探索の応用では麻雀に適応できるようにアルゴリズムを変更し，パラメータの調整によってほとんど知識を利用することなく上級者との打牌一致率を 46 %まで上げることができた．補助問題を用いた評価関数の学習では，牌譜に含まれる情報を補助分類問題として抽出し，有効な特徴空間を構築することで学習の改善を目指した．約 60000 局面のサンプルを用いて学習した結果，補助問題を用いた場合は用いなかった場合よりも上級者との打牌一致率が最大で約 9 %上昇した．

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景と目的

多人数不完全情報ゲームでは，囲碁や将棋などの二人完全情報ゲームで有効であるとされている従来手法をそのまま適用することが困難であることが多い．また，一般に多人数不完全情報ゲームの多くは研究が浅く，その面でも未解決の問題が多く存在する．困難である点として例えば以下のようなものが挙げられる．

- 多人数ゲーム
  - 状況によってそれぞれのプレイヤの目的が異なる場合がある (単純な零和ゲームではない)
- 不完全情報ゲーム
  - 局面に不完全情報や確率的要素があるため，探索の分岐数が膨大である
- 研究の浅いゲーム
  - 評価関数の特徴要素の構築や重み付けが難しい

本研究では，多人数不完全情報ゲームである麻雀の打牌を決定するための手法について，大きくわけて二つの提案を行う．

一つは，モンテカルロシミュレーションをベースにした UCT (UCB applied to Trees) [11] を応用して，麻雀の局面の探索を行う手法である．これは上記のうち，多人数不完全情報ゲームとしての問題を，探索の面から解決するための手法である．多人数ゲームや不完全情報ゲームでは，戦略モデルの複雑さや不確定要素による探索空間の肥大化などの問題があり，二人完全情報ゲームで一般的に用いられる Minimax 探索の適用が難しい．また，探索を制御するために必要な評価関数などを設計するには，ゲームに関する深い知識が必要になる．これらの問題を解決するために，本研究では囲碁などで注目を集めている UCT を応用して麻雀の局面を探索する手法を提案する．UCT は戦略モデルや不確定要素の存在に依存しないため，多人数不完全情報ゲームにおいても有効であると考えられる．また，シミュレーションをベースにした手法を用いることで，麻雀のように研究が浅く知識の利用が難しいゲームにおいても有効な探索を行うことができると考えられる．

もう一つは牌譜に含まれる情報から大量の補助分類問題を作成し，それを用いて特徴空間を拡張した上で，改めて牌譜から局面の評価関数を学習する手法である．これは上記のうち，研究の浅いゲームとして問題を，評価関数の面から解決するための手法である．比較学習など，ゲームで広く用

いられている機械学習の手法では，局面の特徴要素に対して重み付けをすることで評価関数の調整を行う．精度の高い重み付けを行うためには，棋譜に含まれる情報をなるべく有効に活用する必要がある．例えば麻雀の終局には単純な勝敗だけではなくあがり役や点数，待ち形など，豊富な情報が含まれている．これらの情報を効率良く抽出することで，より良い重み付けをすることができると考えられる．また別の問題として，局面の性質を表す特徴要素は人手で抽出するには限界がある．例えば「この局面からあがれる可能性はどれくらいか」ということを人手で記述することは難しい．このような抽出の難しい特徴を，既存の特徴の重み付け・組み合わせによって獲得することができれば，より優れた評価関数を設計することができると考えられる．そこで本研究では，牌譜に含まれる情報をラベルとする大量の補助分類問題を作成し，それを利用した評価関数の学習を提案する．補助分類問題を利用することで，より効率的な牌譜からの情報抽出を行うともに，補助分類問題で得られたモデルや予測結果から抽出した特徴空間を新たな特徴として追加することで，人手での抽出が難しいような特徴空間を構築することができると考えられる．

1.2 本研究の位置づけ

本研究が対象としている多人数不完全情報ゲームの麻雀は，知名度の割に研究が浅く，高度な評価関数や探索アルゴリズムが開発されていないゲームである．

そのようなゲームに対して，

- 『なるべく高度な知識を使わずに良い手を求める』
- 『知識をデータから自動的に獲得する』

という 2 つの側面からアプローチし，研究の浅いゲームに対して発展の足がかりを作ることが本研究の主旨である．

1.3 本論文の構成

以降，本論文の構成は次のようになっている．

まず第 2 章で，UCT 探索を利用した麻雀打ち手の探索について，関連研究，提案手法，実験およびその評価を述べる．

続いて第 3 章で，大量の補助問題を利用した麻雀評価関数の学習について，関連研究，提案手法，実験およびその評価を述べる．

最後に第 4 章で，各研究のまとめと本研究の総合的な結論，今後の課題について述べる．

第2章　UCT探索による打ち手の探索

2.1　背景

多人数ゲームや不完全情報ゲームでは，二人完全情報ゲームで一般的に用いられる Minimax 探索の適用が難しい．まず多人数ゲームは，二人ゲームと違って各プレイヤの戦略モデルが単純でないという問題がある．二人ゲームではそれぞれのプレイヤは各々の獲得報酬を最大化し，それは相手の獲得報酬を最小化することと等価である．しかし，そのような前提は多人数ゲームでは必ずしも成り立たない．例えば，報酬が最大となる手が複数存在する場合に，相手プレイヤがどのようにして手を決定するかは単純には決まらない．したがって，多人数ゲームにおいて適切な探索を行うためには，何らかの方法で相手プレイヤの戦略モデルを仮定しなければならない．また，麻雀の鳴きのように不規則に手番が入れ替わるようなゲームにも Minimax 探索は適用しづらい．次に不完全情報ゲームでは，確定できない情報が存在するために探索空間が非常に大きくなるという問題がある．バックギャモンなどの不確定要素のあるゲームに対して Minimax 探索を応用した例はあるが [3][7]，それに加えて相手の手札などが不明である不完全情報ゲームでは，隠されている情報を何らかの方法で推定するか，可能性のある状態を全て探索対象にしなければならないため困難である．さらに Minimax 探索を利用する場合の難しい課題として，探索打ち切りの局面を評価するための静的評価関数が必要となる点も挙げられる．

これらの問題を解決する方法の一つとして，囲碁などで有効であるとして注目を集めている UCT (UCB applied to Trees) [11] を用いることが考えられる．UCT とはモンテカルロシミュレーションをベースにした木探索手法の一つで，ランダム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より多くの探索を行う手法である．実際に UCT を多人数ゲームや不完全情報ゲームに応用した研究の例が存在し [15][13]，両方の性質を併せ持った多人数不完全情報ゲームに対してもその有効性が期待できると考えられる．

本稿では麻雀における手の探索を目的として実験を行った．麻雀は多人数ゲームであるため相手プレイヤの戦略が複雑になるが，相手プレイヤの手の選択は UCT アルゴリズムをそのまま利用することである程度多様な戦略モデルに対応した混合戦略を得ることができると考えられる．また麻雀は不完全情報ゲームであるため考えうる全ての場合について探索を行うことが困難であるが，ランダムに生成した局面のシミュレーションを繰り返すことによってそれを補うことができると考えられる．さらに麻雀は静的評価関数を作成することが難しいが，UCT では評価値は終局の結果をそのまま用いることができるため評価関数は必要ない．このように UCT を用いることで，高度な戦略モデルや評価関数を作成することなく，様々なケースに対応した手を選択することができると考えられる．

2.2 関連研究

本節では関連研究として，最初に 2.2.1 で Minimax 探索の拡張手法について紹介する．次に 2.2.2 で UCT について説明し，続いて 2.2.3 で UCT を多人数ゲームに適用した研究，2.2.4 で UCT を不完全情報ゲームに適用した研究について紹介する．

2.2.1 Minimax 探索の拡張

ここでは Minimax 探索を不確定ゲームに拡張した手法である*-Minimax 探索と，多人数ゲームに拡張した手法である Max^n 探索について紹介する．

2.2.1.1 *-Minimax 探索

ノード間の遷移が確率的であるような木で Minimax 探索を行う手法として，Ballard によって提案された*-Minimax 探索がある [3][7]．Minimax 木での Min ノードと Max ノードに対して，子ノードへの遷移が確率的であるようなノードを Chance ノードと呼ぶ．Ballard は Chance ノードを含むようなゲーム木で Alpha-Beta 探索のような枝刈り探索手法を実現するアルゴリズムを提案した．近年になってバックギャモンのプレイヤにそのアルゴリズムを適用する研究が行われ [8]，枝刈りを行わない場合に対して最大で 95%の探索時間削減を実現したと報告されている．

2.2.1.2 Max^n 探索

Minimax 探索を多人数ゲームに応用した手法として Luckhardt らによる Max^n アルゴリズムがある [12][14]．図 2.2 に 3 プレイヤゲームの Max^n 探索木の一例を示す．Max^n 探索は各ノードの手番のプレイヤが自身の評価値が最大となる手を選ぶという前提で最善手の探索を行う．Luckhardt らはさらに枝刈りによって多人数ゲームでも効率的な探索が可能であることを示した．

2.2.2 UCT

UCT (UCB applied to Trees) は，多腕バンディット問題における効率的な試行方策である UCB1 を木探索に応用したもので，2006 年に Kocsis らによって提案された [2][11].

UCT 探索のアルゴリズムについて簡単に説明する．UCT は図 2.3 に示すように 4 つのステージからなる．

- 子ノードの選択

  まず未探索の子ノードがある場合にはそれを優先的に選択する．子ノードが全て探索済みの場合，UCT では (2.1) 式で求められる UCB (Upper Confidence Bounds) 値が最大となる子ノー

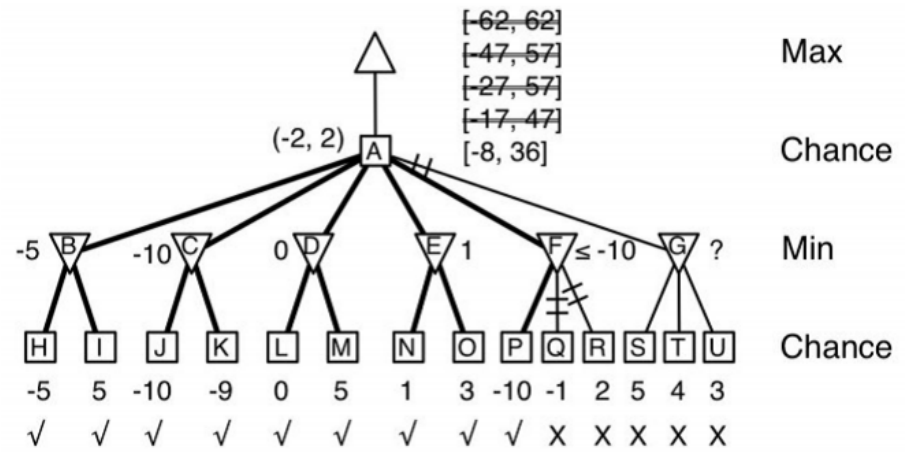

図 2.1: *-Minimax tree search. [Hauk, 2004]

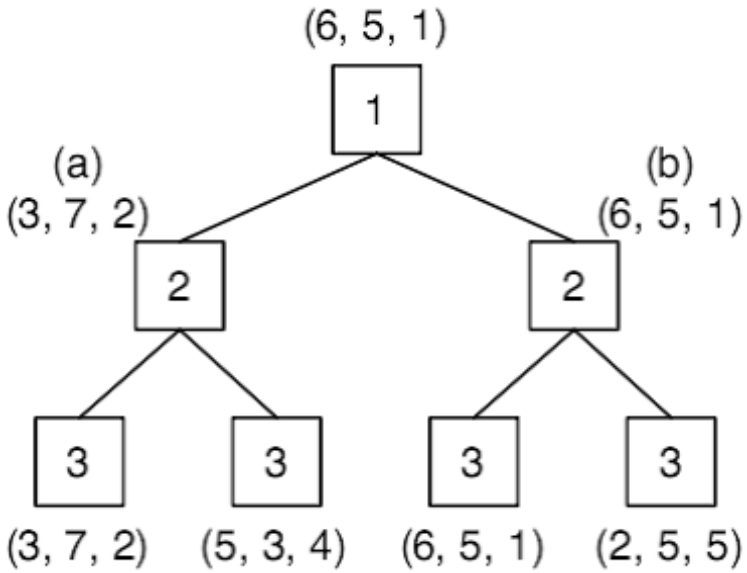

図 2.2: A sample max^n Tree. [Sturtevant, 2003]

ドを探索する．

$$\overline{X}_i + C\sqrt{\frac{\ln T}{T_i}} \tag{2.1}$$

$\overline{X}_i$ は子ノード i の平均報酬，T_i は子ノード i の探索回数，T は親ノードの探索回数，C はバランスパラメータである．つまり，基本的には第 1 項の平均報酬の高い手が探索されるが，探索回数が少なくてまだ高い報酬を得る可能性のある手も第 2 項が大きくなるため探索されやすいというアルゴリズムである．

- 子ノードの生成

  子ノードが存在しない場合には合法手を生成して，それに伴って遷移する局面を生成し，新たな子ノードとして追加する．

- プレイアウト

  未探索ノードに到達した場合には終局までゲームのシミュレーションを行う．シミュレーションはランダムに行うことも可能だが，ヒューリスティックや評価関数を用いてシミュレーションの質を向上させることもできる [17]．このシミュレーションのことをプレイアウトと呼ぶ．

- 更新

  プレイアウトで得られたゲームの結果そのものを報酬として探索経路上のノードを更新する．

以上の操作を終了条件 (時間や探索回数など) を満たすまで繰り返し，最終的に平均報酬の最も高い子ノードを次の手として選択する．

UCT はシミュレーションを利用したアルゴリズムであるため，精度の高い静的評価関数を使う必要がないのも特徴である．

2.2.3 多人数ゲームにおける UCT

Sturtevant は 3 プレイヤ以上の多人数ゲームにおける UCT の性能解析の研究を行った [15]．図 2.2 の (b) のノードに注目すると，このノードの手番であるプレイヤ 2 にとって二つの子ノードの評価値は等しい．この例では単純に最も左の手を選択しているが，このような局面で相手プレイヤがどちらの手を選択するかを判断するためには相手プレイヤの戦略モデルが必要であり，適切な戦略モデルを作成することは一般に難しい．

同様のケースに対して UCT を適用すると，(b) のノードではいずれの子ノードからも同じ報酬が返ってくるため UCB 値による手の選択は探索回数によって決定される．結果的に両方の子ノードは同回数の探索が行われ，親の (b) のノードの評価値は二つの子ノードの平均となる．このようにして，多人数ゲームに UCT を用いるとシミュレーションに基づいた探索が行われるため，相手プレイヤについての混合戦略を得ることができる．

Chinese Checkers (ダイヤモンドゲーム) での実験では，UCT は評価関数を用いた Max^n 探索を上回る性能を発揮したと報告されている．

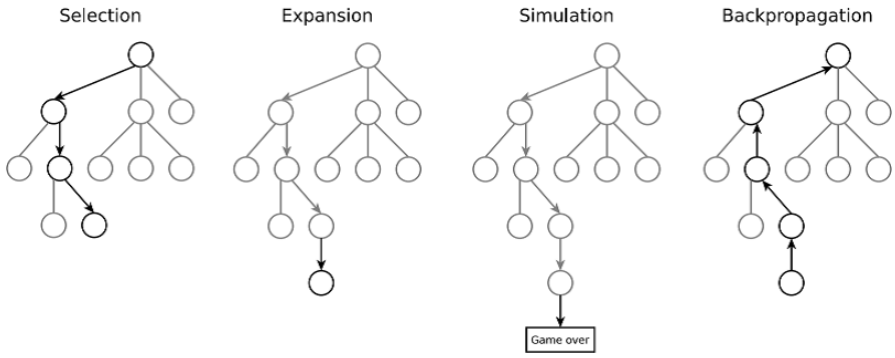

図 2.3: UCT 探索アルゴリズム [Schäfer, 2008]

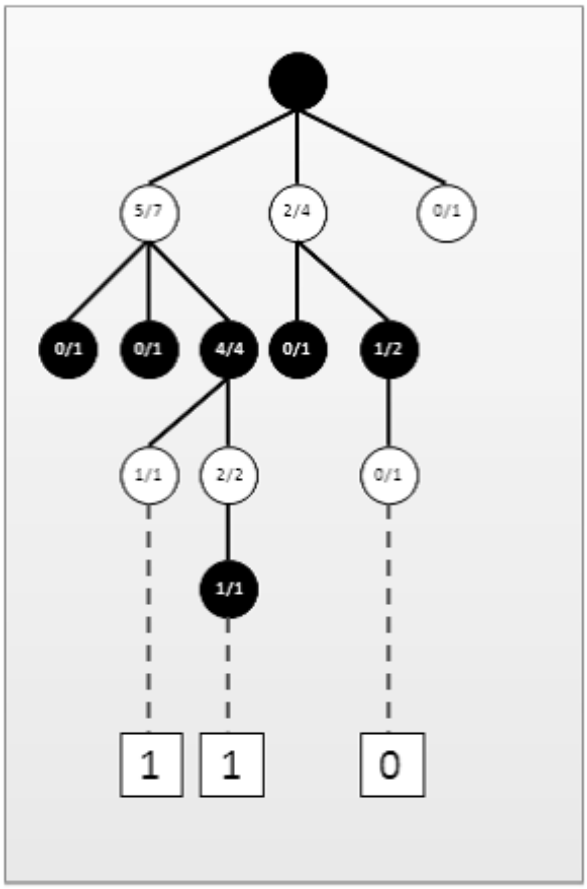

図 2.4: UCT 探索木．ノードの数値は (獲得報酬 / 探索回数)

2.2.4 不完全情報ゲームにおける UCT

Schäfer は不完全情報ゲームである Skat における UCT の性能解析の研究を行った [13]．Skat は 3 人のプレイヤが手札を 1 枚ずつ出して，最も強いカードを出したプレイヤが得点を取っていくトランプゲームである．3 人の中でソロイストと呼ばれるプレイヤが 1 人存在し，ソロイストは最初に宣言した得点を取ることを目指し，他のプレイヤはそれを阻止することを目的とする．特別な場合を除いて各プレイヤの手札は伏せられているので，Skat は不完全情報ゲームである．

不完全情報ゲームに対して UCT を適用するために，Schäfer は UCT の繰り返しの度に相手の手札をランダムに生成して局面を仮定するという操作を行った．仮定とシミュレーションを何度も繰り返すことで，それぞれの手を打った場合の報酬の期待値を得ることができると考えられる．また相手の手札の生成を完全にランダムにするのではなく，これまでのプレーから持っているスートを限定することでより良い性能を出したとも報告されている．実験では Skat のコンピュータプレイヤとして一般的な DDSS (Double Dummy Skat Solver) という手法に対して，互角の性能を出すことができたと報告されている．

2.3 提案手法

この節では本研究の具体的な手法について述べる．

Minimax 探索や学習を用いる方法はいずれも評価関数や牌譜など，ゲームに関する知識を必要とするものである．そのため麻雀のように研究が進んでおらず，体系だったコンピュータプレイヤの知識が集積していないゲームにおいては適用が難しい．

本研究では多人数不完全情報ゲームである麻雀に対して UCT 探索を適用した．多人数不完全情報ゲームに対して UCT を用いれば，シミュレーションによって複雑な戦略や可能性のある膨大な局面の探索を補うことができるため，多様なケースに対応した手の探索を行うことができると考えられる．また UCT は Minimax 探索と違って不規則な手番の入れ替わりにも対応しやすいため，鳴きがある麻雀にも適していると考えられる．

手法の全体的な流れは以下のようになる．

1. 局面中の不完全情報を仮定する
2. UCT アルゴリズムのループを 1 回行う
3. 1，2 を繰り返す
4. 最も平均報酬の高い手を選択する

不完全情報ゲームで探索を行うため，最初に局面の仮定を行う．見えていない牌を数え上げ，それをランダムに振り分けて相手の手牌とツモ山，王牌を生成する．この局面の仮定操作は UCT アルゴリズムの 1 回のループの前に毎回行う．局面を仮定したらルートノードから探索を開始する．ここで，麻雀において手を決定する 2 種類の局面のうち，手牌が全部で 14 枚あってどれか 1 枚を捨てる (あるいは暗カンや加カンをする) 局面を 14 牌ノード，手牌が全部で 13 枚あって相手の捨て牌で鳴くか

鳴かないかを選択する局面を 13 牌ノードと呼ぶことにする．それを踏まえて麻雀における UCT 探索のアルゴリズムを Algorithm 1 に示す.

麻雀のゲームの特性上，一般的な UCT とは異なっている個所について説明する.

- 手の生成

  不完全情報局面の仮定によって相手の手牌やツモ山が毎回変わるため，すでに子ノードが展開されているノードでも訪問するたびに手の生成を行う必要がある．そこで新しい手が存在すれば，それに対応する新しいノードを生成して追加する．また，ルートノードでの他プレイヤの鳴きの手は最終的な選択候補に入らないため生成しない.

- 手の選択

  手を選択するときにも手牌やツモ山が変わっているために，すでに展開されたノードの中には打てない手が含まれているので，今回生成した手の中から選ぶようにする必要がある．14 牌ノードにおける手の選択は，その手番のプレイヤが完全に決定権を持つため，UCB 値の最大のノードを選択する．一方で 13 牌ノードにおける手の選択は，手番でないプレイヤによる鳴きが入るかどうかで決定権が変わるため，決定権の高いプレイヤの順 (ロン，明カンまたはポン，チーの順) に UCB 値が鳴かない場合より高いかどうかを調べ，高い場合はその手を選択する.

- プレイアウトと報酬

  プレイアウトはいずれかのプレイヤがあがるか流局になるまでゲームを進め，4 人のプレイヤそれぞれの報酬を観測する.

Algorithm 1 麻雀における UCT 探索アルゴリズム

```
while 終了条件を満たしていない do
  局面を仮定する
  i ← 0
  node[i] ← ルートノード
  while node[i] の探索回数 > 0 do
    if node[i] が終局 then
      break
    end if
    手を生成
    if 新しい手がある then
      新しい子ノードを生成
    end if
    if 生成された手の中に未探索の子ノードがある then
      node[i + 1] ← 未探索の子ノード
    else
      if node[i] が 14 牌ノード then
        node[i + 1] ← 生成された手の中で UCB 値が最大の子ノード
      else if node[i] が 13 牌ノード then
        node[i + 1] ← 生成された手の中で鳴きの決定権と UCB 値によって選択した子ノード
      end if
    end if
    i ← i + 1
  end while
  プレイアウトを行って報酬を観測する
  while i ≥ 0 do
    node[i] の報酬と探索回数を更新
    i ← i − 1
  end while
end while
move ← 平均報酬が最大の子ノードへの手
```

2.4 実験

本節では，実際に麻雀において UCT 探索を行うプレイヤを実装して行った実験の方法と結果について述べる．

2.4.1 実験環境

実験は CPU Core2 Duo 3GHz ，メモリ 2GB のマシン環境で行った．実装は C++言語を用いた．

2.4.2 牌譜との一致率

UCT 探索によって選ばれた手と上級者の牌譜の手との一致率を評価する実験を行った．牌譜データとして，システマティック麻雀研究所[1]で公開されている東風荘の牌譜からランダムに抽出した 1000 局面を用いた．実験方法として以下の 3 種類のパラメータを変化させたときの一致率の変化を測定した．

- 報酬
- UCB 値のバランスパラメータ C
- UCT アルゴリズムのループ回数

2.4.2.1 報酬

プレイアウトの終局で観測する報酬の種類を以下の 4 つに設定して 14 牌ノード局面での一致率を測定した．

- 得点収支
  あがりの点数による損益をそのまま報酬とした．バランスパラメータ C は得点のスケールに合わせて 1000 に設定した．
- 順位点
  順位点とは，清算後の持ち点から原点（東風荘では 30000 点）を引いて零和化し，1000 で割って順位ごとの賞与点を加えた最終点数である．この点数が試合の結果として各プレイヤの成績に反映される．バランスパラメータ C は順位点のスケールに合わせて 10 に設定した．
- 得点収支を正規化したもの
  (2.2) 式のようなシグモイド関数を用いて正規化を行った．

$$f(x) = \frac{1}{1 + e^{-kx}} \tag{2.2}$$

  バランスパラメータ C は $\sqrt{2}$ に設定した．k は 0.0003 に設定した．

[1] http://www.interq.or.jp/snake/totugeki/

- 順位点を正規化したもの
  バランスパラメータ C は $\sqrt{2}$ に設定した．シグモイド関数の k は 0.1 に設定した．
- あがりの二値
  あがったプレイヤの報酬を 1，振り込んだプレイヤもしくはツモられた場合の他 3 人の報酬を 0，収支のなかったプレイヤの報酬を 0.5 とした．バランスパラメータ C は 1 に設定した．

結果を表 2.1，表 2.2 に示す．UCT のループ回数は表 2.1 では 100000 回，表 2.2 では 10000 回に設定して実験を行った．結果として，いずれも得点収支を報酬として用いたときが最大の一致率を示した．

2.4.2.2 UCB 値のバランスパラメータ C

(2.1) 式のパラメータ C を 100，1000，10000 の 3 種類に設定して 14 牌ノード局面での一致率を測定した．報酬は得点収支，UCT のループ回数は 10000 回に設定した．結果を表 2.3 に示す．$C = 1000$ のときが一致率最大となった．これは得点収支のスケールとほぼ一致しているためと考えられる．

2.4.2.3 アルゴリズムのループ回数

UCT アルゴリズムのループ回数を 100 回から 500000 回まで変化させて 14 牌ノード局面での一致率を測定した．また，ループ回数を 100 回から 50000 回まで変化させて 13 牌ノード局面での一致率を測定した．報酬は得点収支を用いた．(2.1) 式のバランスパラメータ C は 1000 に設定した．

結果を図 2.5，図 2.6 に示す．凡例の Random は完全にランダムに手を選択する場合の一致率，SVM は線形カーネルの SVM-rank[10][9] で学習した分類器によって得られた最善手との一致率である．SVM の学習にはシステマティック麻雀研究所で公開されている東風荘の牌譜からランダムに抽出した 75 試合分の牌譜を使用し，表 2.4，表 2.5 に示す約 60000 の特徴要素を用いて行った．

14 牌ノード局面においては，ループ回数が 100000 回のときに UCT の一致率は最高値である 46.1%を記録し，SVM を上回る結果となった．この結果は北川らの研究の結果や SVM の学習に木カーネルを用いた場合には若干及ばないものの [21][18]，知識を必要としない手法も有効であるということを示していると言える．

一方で 13 牌ノード局面においては，ループ回数が 10000 回以上になると一致率が大きく低下するという現象が見られた．13 牌ノード局面はそのうち 9 割ほどが牌譜では鳴かないという偏りがあるため，図 2.7, 2.8 に牌譜で鳴いた局面のみの場合と鳴かなかった局面のみの場合を示した．これによると，ループ回数が増えるにつれて鳴く手が選択されやすくなっていることがわかる．ランダムプレイアウトではほとんどあがることができないため (10000 回のプレイアウトをテストしたところ，あがりによって終局したのはわずか 38 回であった)，有効な報酬の大半は流局のノーテン罰符によるものとなる．したがって，あがることよりもテンパイすることを目指すような探索になってしまい，シャンテン数を減らせる鳴きの手が選択されやすくなっていると考えられる．プレイアウトに評価関数などを用いて終局であがれる割合を増やせば，この傾向は変わっていくと考えられる．

表 2.1: 報酬による一致率変化 (100000 ループ)

報酬	得点収支	順位点	正規化得点	正規化順位点
一致率 [%]	50.0	34.6	32.7	26.6

表 2.2: 報酬による一致率変化 (10000 ループ)

報酬	得点収支	あがり二値
一致率 [%]	39.7	29.8

表 2.3: UCB 値の C による一致率変化 (10000 ループ)

C	100	1000	10000
一致率 [%]	24.3	39.7	33.6

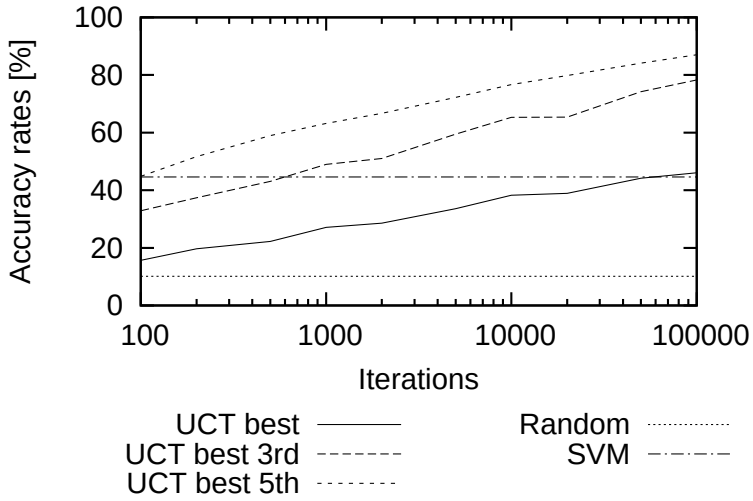

図 2.5: 14 牌ノードでの一致率

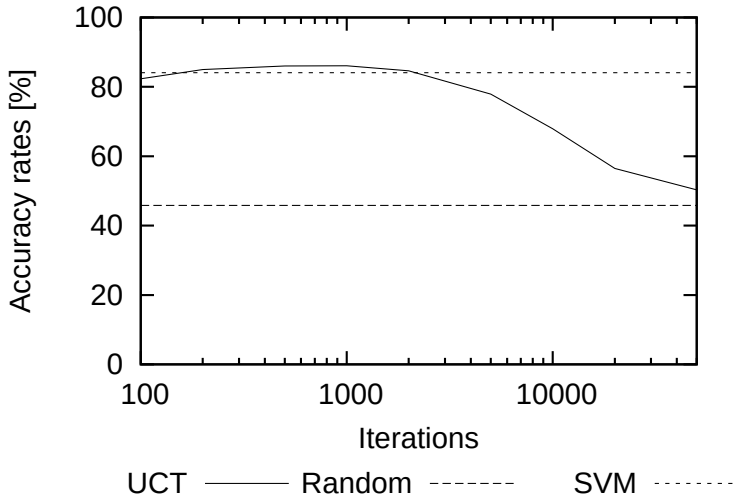

図 2.6: 13 牌ノードでの一致率

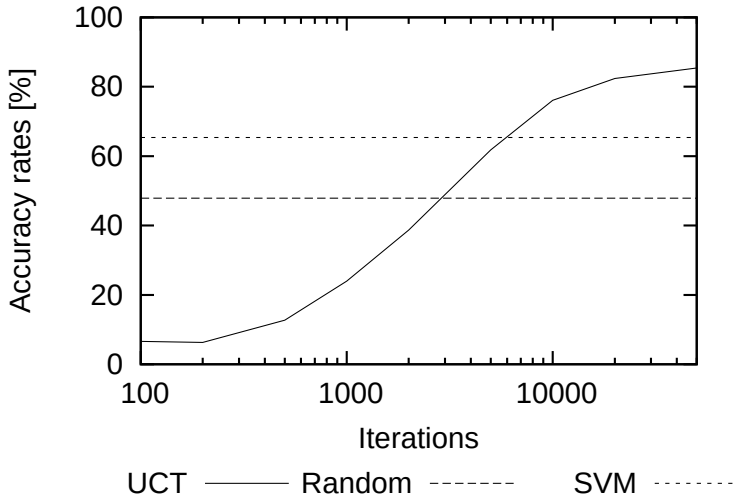

図 2.7: 牌譜で鳴いた局面の一致率

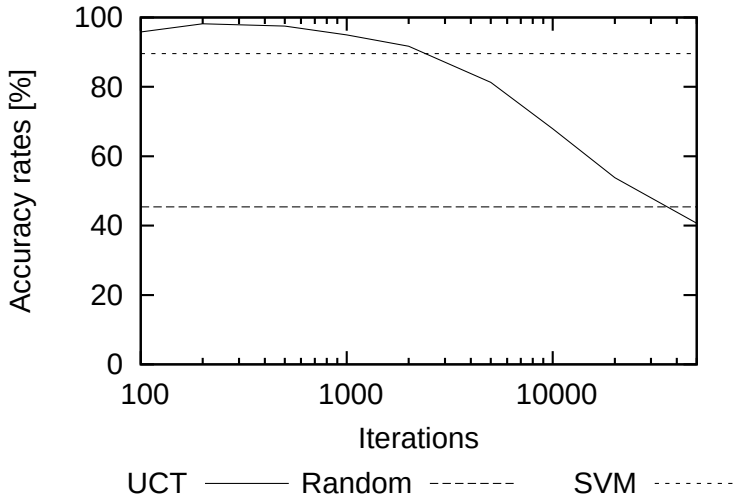

図 2.8: 牌譜で鳴かなかった局面での一致率

表 2.4: 線形 SVM の特徴要素 1

手牌	各牌の所持数
	面子の所持数
	含まれる面子の総数
	面子候補の所持数
	含まれる対子の総数
	含まれる両面の総数
	含まれるカンチャンの総数
	含まれるペンチャンの総数
	フーロ数
	面前かどうか
	ホンイツにするための不要牌の数
	チンイツにするための不要牌の数
自分の状態	シャンテン数
	ドラの所持数
	ドラそばの所持数
	親かどうか
	上がり牌の残り枚数
	確定している役数

表 2.5: 線形 SVM の特徴要素 2

相手の鳴き牌	フーロ牌
	確定役数
	見えているのが一色かどうか
	フーロ数
場	供託棒の点数
	自分以外のリーチ者の数
	親がリーチしているか
	局番
	本場
	オーラスかどうか
	自分の順位
	トップとの点差
	一つ上位との点差
	一つ下位との点差
	ラスとの点差
相手の捨て牌	リーチ順目
	一発があるか
	現物
	スジ
牌の組み合わせ	1 牌の所持 boolean
	2 牌の組み合わせ
	3 牌の組み合わせ

2.4.3 コンピュータプレイヤとの対戦

グリーディプレイヤと SVM プレイヤを相手に 2 対 2 の対戦を行った．グリーディプレイヤは全ての合法手の中で打った後のシャンテン数が最も小さい手の集合を生成し，その中からランダムに手を選択するプレイヤである．対戦は連荘なしの東風戦で行い，ゲームの不確定性による成績のばらつきを抑えるために 1 試合中の 4 局では全く同じ配牌とツモ山になるようにした．席の配置は同種のプレイヤが常に対面の状態で行った．UCT のシミュレーションは 1 手あたり 5 秒に設定し，報酬は得点収支を用いた．(2.1) 式のバランスパラメータ C は 1000 に設定した．

対戦の結果を表 2.6,2.7 に示す．グリーディプレイヤとの対戦では UCT プレイヤが和了率，平均収支ともに若干ながら上回っている．同一の配牌，ツモ山に対して，UCT プレイヤの方がより良い手を選択できている可能性があると考えられる．SVM プレイヤとの対戦でもほぼ互角の性能を発揮しており，UCT を用いることで知識なしでもそれなりに良い手を求めることができていると言える．

表 2.6: グリーディ対 UCT (62 試合 248 局)

プレイヤ	Greedy A	UCT A	Greedy B	UCT B
和了率 [%]	11.69	18.55	14.92	16.13
平均収支	761	787	716	783

表 2.7: SVM 対 UCT (100 試合 400 局)

プレイヤ	SVM A	UCT A	SVM B	UCT B
和了率 [%]	19.25	17.75	16.75	19.25
平均収支	71.25	85.75	29	84

第3章　大量の補助問題を用いた評価関数の学習

3.1　背景

将棋などで広く用いられている比較学習では，棋譜で選択された局面を選択されなかった局面と比較して，棋譜の局面の評価値が最大となるように特徴要素に重み付けすることで評価関数を調整する．調整の際には棋譜に含まれる情報をなるべく有効に活用することが望ましい．しかし，特徴要素などを人手で設計するだけでは，それらを十分に活用することは難しい．

例えば，棋譜の局面比較のみでは区別できないようなデータがあった場合には，それらに関係する特徴要素には適切な重み付けができない可能性がある．このような場合に，棋譜の別の情報を用いて補助問題を作成し，それによってデータを区別することができれば，より良い重み付けができると考えられる．

また，別の問題として，局面の性質を表す特徴要素は人手で抽出するには限界がある．例えば「この局面からあがれる可能性はどれくらいか」ということを人手で記述することは難しい．このような抽出の難しい特徴を，補助分類問題の予測結果という形で既存の特徴の重み付け・組み合わせによって獲得することができれば，より優れた評価関数を設計することができると考えられる．

そこで本研究では，牌譜に含まれる情報をラベルとする大量の補助分類問題を作成し，それを利用した評価関数の学習を提案する．例えば麻雀の終局には単純な勝敗ではなくあがり役や点数，待ち形など，豊富な情報が含まれている．人間の打ち手は手を選択するときにこれらの情報を用いていると考えられるため，学習においてもその情報を利用することができれば，より優れた学習が可能であると考えられる．補助分類問題を利用することでより効率的な牌譜からの情報抽出を行うともに，補助分類問題を解くことで得られるモデルや予測結果から抽出した特徴空間を新たな特徴として追加することで，人手での抽出が難しいような特徴空間を構築することができると考えられる．

3.2　関連研究

3.2.1　比較学習

比較学習とは，1989 年に Tesauro によって提案された評価関数の学習手法の一つである [16]．それまでの教師あり学習は局面の具体的な点数付けを必要としていたため，非常にコストが高い上，柔軟な調整が困難であった．

比較学習では，個々の局面に具体的な数値を与えることはせず，局面 A は局面 B よりも良い (評価値が高い)，というような相対的な順序の情報のみを与える．このようにして与えた順序情報を満たすように評価関数のパラメータを調整する．

比較学習は具体的な数値を与える必要がないため訓練局面のラベル付けがずっと容易である上，局面の相対的な関係を考慮することは人間のプレイヤが実際に行っていることと直感的にも合致していると言える．

近年では比較学習を利用した将棋のコンピュータプレイヤである Bonanza が非常に高い性能を発揮し，注目されている学習手法の一つである．

3.2.2 Bonanza

Bonanza は，保木によって開発された将棋のコンピュータゲームプレイヤである [19]．2006 年度の世界コンピュータ将棋選手権で優勝しており，現在でも世界トップクラスの将棋プログラムのひとつである．

Bonanza は比較学習を用いて評価関数の学習を行っているが，棋譜の局面を直接比較するのではなく，棋譜の局面から探索した先の局面を比較するように損失関数を設計している．Bonanza で用いられた損失関数を (3.1) 式に示す．

$$l(s,\vec{\theta}) = \sum_{m=1}^{M-1} T[\xi(s'_m,\vec{\theta}) - \xi(s'_{m=0},\vec{\theta})] \tag{3.1}$$

$\vec{\theta}$: パラメータベクトル

s'_m : 局面 s を合法手 m で進めた子局面

M : 局面 s での合法手の数

$m=0$: 棋譜で指された手

$\xi(s'_m,\vec{\theta})$: $Minimax$ 探索の評価値

$T(x)$ は，損失関数の形状を決定する関数である．棋譜の指し手の評価値よりも十分大きい場合には 1，十分小さい場合には 0 になるような判別関数として設計するのが好ましいと言える．実際に用いられたのはシグモイド関数と呼ばれる，階段関数の傾きをやや緩やかにした (3.2) 式のようなものである．k は傾きを持つ領域を調整する正の実数パラメータである．

$$T(x) = \frac{1}{1+e^{-kx}} \tag{3.2}$$

3.2.3 麻雀評価関数の学習

北川らは比較学習の手法に基づいて牌譜の打ち手と評価関数による最善手の一致度の誤差を最小化することを目指し，3 層ニューラルネットワークを用いて打ち手の評価関数を学習する研究を行っ

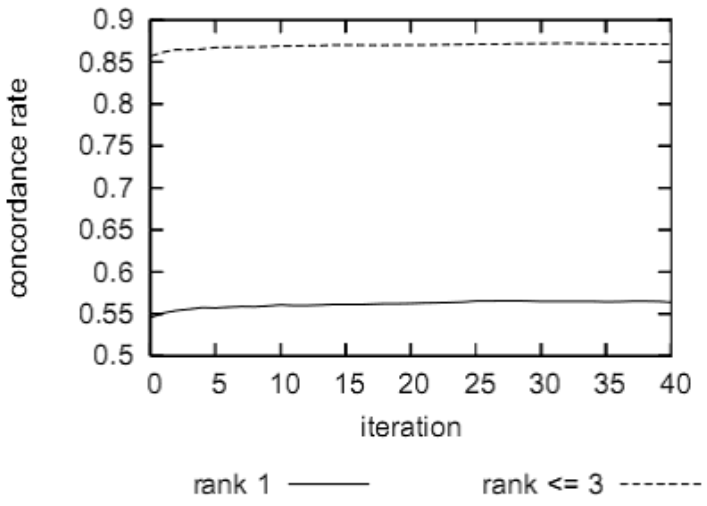

図 3.1: ツモ局面での一致率 [Kitagawa et al., 2007]

た [21]．ニューラルネットワークの入力層には人手で作成した 1532 の boolean の特徴要素 (表 3.1) を使用し，6079 試合分の牌譜からツモ局面については 575906 局面を学習データとして用い，学習の繰り返し数を 40 回まで増やしながら実験を行っている．結果は図 3.1 のようになり，最大で 56 %の一致率に到達した．

また，我々は木カーネルを用いた SVM による麻雀の打ち手の順位付けの学習を行い，打ち手の順位関係を分類する分類器を作成することで最善手を予測する研究を行った [18]．この研究で我々は図 3.2 の麻雀の手牌を図 3.3 のように木構造で表現し，木カーネルと呼ばれるカーネル関数を用いた非線形 SVM によって打ち手の順序関係を学習した．結果として，手牌以外の要素を特徴に用いなかったにもかかわらず，牌譜との一致率は 53%に達した．

図 3.2: 手牌の例 [Miki et al., 2008]

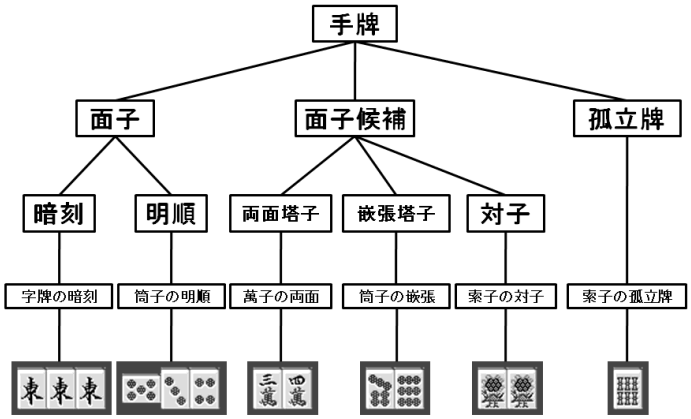

図 3.3: 手牌の木構造の抽出 [Miki et al., 2008]

表 3.1: 麻雀の特徴要素 [Kitagawa et al., 2007]

自分の状態	面前の持ち牌
	面前の持ち牌 2 枚の組み合わせ
	面前の持ち牌 3 枚の組み合わせ
	鳴いた牌の構成と状態
	面子数
	両面搭子数
	カンチャンとペンチャンの和
	対子数
	テンパイしているかどうか
	ドラの枚数
	面前であるかどうか
	親であるかどうか
	リーチしているかどうか
	自分が捨てたことのある牌
他プレイヤの状態	鳴いた牌の構成と状態
	鳴いた回数
	鳴いた牌のドラの枚数
	親であるかどうか
	リーチしているかどうか
	そのプレイヤに対する完全安牌
	筋や壁などで安全度が高い牌
	自分との点差
場の状態	オーラスかどうか
	見えていない牌の残り枚数

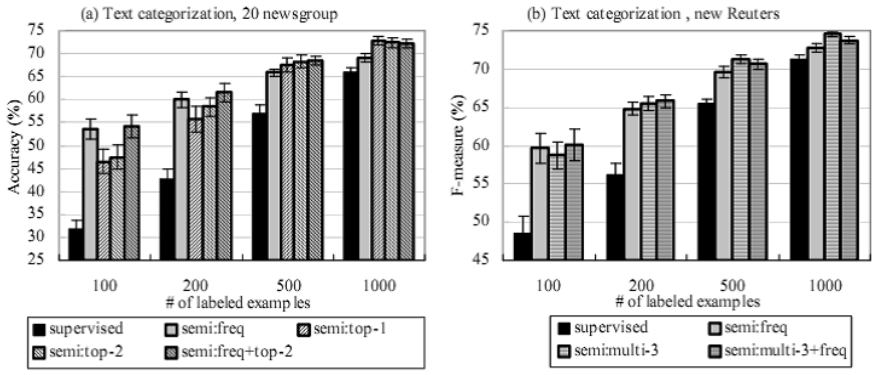

図 3.4: Text categorization performance results. [Ando et al., 2005]

3.2.4 SVD-ASO

SVD-ASO (Singular Value Decomposition-based Alternating Structure Optimization) は，Ando らが提案した半教師あり学習手法で，自然言語処理の分野でその有効性が示されている [1]．自然言語処理などでは訓練データのラベル付けに非常にコストがかかるため，ラベル付けされていないデータを併用する半教師あり学習が重要な研究課題となっている．

SVD-ASO の手法の概要を以下に示す．

1. ラベル付けされていない大量のデータから大量の補助問題を作成する
2. 補助問題を学習して得られたモデルに対して特異値分解 (SVD) を行って，有効な特徴空間を抽出する
3. 抽出された特徴を既存の特徴空間に追加して，ラベル付きデータを用いて主問題の学習を行う

補助問題には次のような条件が要求される．

1. ラベル付けが容易である (自動化できる) こと
2. 主問題と関連性が深い問題であること

特に条件 2 が重要であり，主問題の部分問題になっているなど，主問題と関わりの深い補助問題を作成することが必要であるとされている．これは特徴空間を SVD によって重み付け・抽出して加えるため，補助問題の解決に有効な特徴空間は主問題の解決にも有効に働くという前提が必要なためである．

Ando らは，主問題である『文書カテゴリ分類』と『単語の品詞タグ付け』に対して，次のような補助問題を提案・作成した．

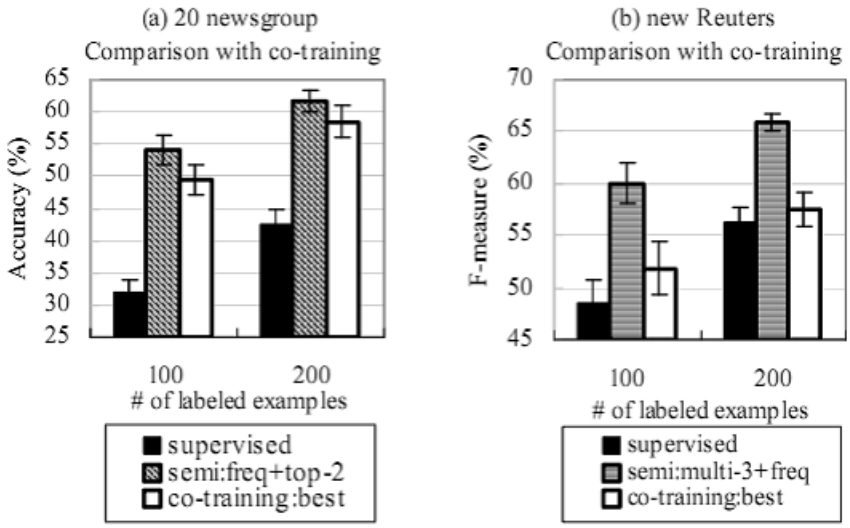

図 3.5: Comparison with co-training. [Ando et al., 2005]

- 教師なしのもの

  頻出単語の推定 文書カテゴリ分類のための補助問題で，いくつかの頻出単語を隠し，それを周囲の単語の出現頻度から推定する問題．隠す頻出単語の種類だけ補助分類問題が作成される．

  単語列の推定 単語の品詞タグ付けのための補助問題で，ある区間の単語列を，周囲の語から推定する問題．推定する単語列の数だけ補助分類問題が作成される．

- 一部教師ありのもの

  分類器の予測の予測 まず一部の種類の単語のみを用いて，教師あり学習で分類器を学習する．そして，その分類器が『何を分類するものであるか』を別の種類の単語を用いて補助分類問題として解く．

  分類器の上位 k 個の予測 上記とほぼ同様で，教師あり学習で作成した分類器が分類するクラスを，信頼度の高い順に上位 k 個予測する補助分類問題として解く．

  分類器の予測信頼範囲の予測 上記とほぼ同様で，教師あり学習で作成した分類器が，特定のクラスを分類する信頼度が一定以上あるかどうかを補助分類問題として解く．

実験結果の一例を図 3.4，図 3.5 に示す．補助分類問題を用いてラベルなしデータを利用した学習を行った場合，ラベルありデータのみの教師あり学習や，他の半教師あり学習手法である co-training[4] よりも高い性能を発揮している．また，性能の差は訓練データ数が少ないときほど大きくなっているのが特徴的である．

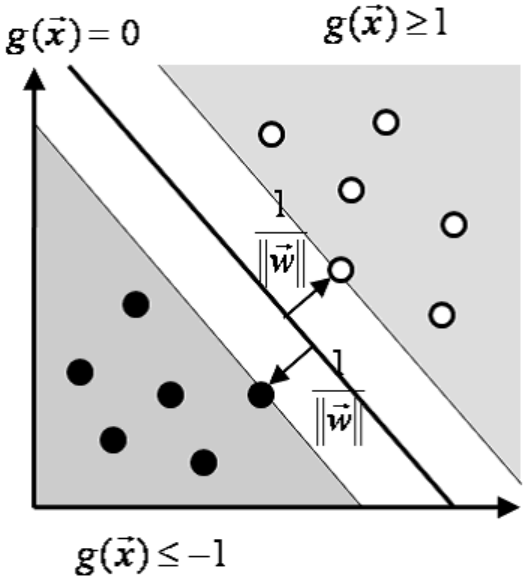

図 3.6: Support Vector Machine

3.2.5 SVM

本研究では補助分類問題および主問題の学習に線形 SVM を用いた.

SVM[5] は，1992 年に Vapnik らによって提案された 2 クラスの分類器の 1 つである．SVM は，超平面とそれに最も近い例との距離 (マージン) を最大化するように，訓練データを線形分離する (分離) 超平面を求める.

分離超平面の法線ベクトルを $\mathbf{w}$ とし，識別関数を $g(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ とすると，図 3.6 のようになる．y_i を $\mathbf{x}_i$ が正例なら 1，負例なら-1 をとる数とすると，$y_i(\mathbf{w} \cdot \mathbf{x} + b) \geq 1$ の条件の下でマージン $2/\|\mathbf{w}\|$ 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，マージン最大化は $\sum_{i=1}^{n} \alpha_i y_i = 0$ および $\forall_i \alpha_i \geq 0$ の条件下で (3.3) 式を最大化する問題に置き換えられる.

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{3.3}$$

3.3 提案手法

関連研究で挙げたような比較学習でコンピュータプレイヤの評価関数を調整する手法では,『上級者の手か，そうでないか』ということを正確に学習するために，適切な重み付けがされるような特徴要素をうまく設計する必要がある．しかし，特徴要素の設計や適切な重み付けは一般には難しい問題である．一方で，棋譜などの訓練データには多くの情報が含まれていると考えられ，そのような情報をうまく抽出して学習に利用することができれば，より優れた学習を行うことができると考えられる．

本研究では，牌譜に含まれる情報を補助分類問題として抽出・利用することによって，麻雀の局面評価関数の学習を改善する手法を提案する．

手法の流れは次のようになる．

1. 牌譜から大量の補助分類問題を作成し，解く
2. 補助分類問題の分類結果を用いて特徴空間を拡張する
3. 牌譜の打牌を用いて評価関数を学習する

以下では個々のポイントについて詳しく説明する．

3.3.1 補助分類問題

麻雀には比較的わかりやすい補助問題が大量に存在する．補助問題の例を表 3.2 に示した．

補助問題はさらに作ることが可能である．例えば，表 3.2 の問題を，(局面もしくは上がりまでの) 手数やシャンテン数ごとに学習することがあげられる．また，特定のカテゴリ (牌の種類や待ちの形) に属する次の打牌の選択問題などは，多く作成でき，有効な特徴の発見に有用であろう．また，点数の情報を教師とした学習などの回帰問題も存在する．

3.3.2 特徴要素

補助問題を表現するために，ある程度は人手で特徴を設計することが必要である．自分の牌・敵の捨て牌・山から取り出した情報など既存の有効な特徴を利用する．

3.3.3 特徴空間の拡張

補助問題を利用して特徴空間を拡張する方法として，本研究では次の二つを提案する．

- モデルを利用した拡張
- 予測結果を利用した拡張

以下ではそれぞれの手法について説明する．

3.3.3.1 モデルを用いた特徴空間の拡張

ここではモデルを用いて特徴空間を拡張する方法について説明する．これは SVD-ASO[1] で用いられた手法と同等である．

補助問題の中には，強いプレイヤの打牌の学習に使われる情報の部分情報を利用するものがある．これらの関連した部分問題からのモデルを利用して，主成分分析 (Principal Component Analysis) などを利用し**モデルの重みから抽出した有用な特徴空間**を，既存の特徴空間に加える．これは，例えば，手牌だけを見せて『何を切る』という問題の学習を行った場合には，学習したモデルにおいて大きく重み付けがされている特徴セットは，実戦局面の打牌の学習においても有効に利用されると考えられるためである．

各補助問題のモデルを並べた行列 $\mathbf{M}$ に対する主成分分析の変換行列 (結合係数行列) を $\mathbf{P}$ とすると，拡張された特徴空間 $\mathbf{f^{ext}}$ は，既存の特徴空間 $\mathbf{f^{org}}$ と, それを $\mathbf{P}$ で変換したベクトル $\mathbf{Pf^{org}}$ を並べたものになる．

$$PCA(\mathbf{M}) = \mathbf{PM} \tag{3.4}$$

$$\mathbf{f^{ext}} \leftarrow \{\mathbf{f^{org}},\ \mathbf{Pf^{org}}\} \tag{3.5}$$

SVD-ASO では行列 $\mathbf{P}$ の計算を特異値分解 (Singular Value Decomposition) によって行っており，特異値分解による右特異ベクトルの行列が主成分分析の結合係数行列と一致することを利用している．

$$SVD(\mathbf{M}) = \mathbf{U\Sigma P}^T \tag{3.6}$$

$$\mathbf{f^{ext}} \leftarrow \{\mathbf{f^{org}},\ \mathbf{Pf^{org}}\} \tag{3.7}$$

3.3.3.2 予測結果を用いた特徴空間の拡張

ここではモデルによる予測結果を用いて特徴空間を拡張する方法について説明する．

補助問題の中には，強いプレイヤの打牌の学習には直接的には含まれていない情報を扱うものがある．これらのモデルによる**予測結果**を追加の特徴要素として既存の特徴空間に加える．これは，例えば,『どの役であがれそうか』という情報は，強いプレイヤの打牌での学習では利用されておらず，この予測は学習の特徴として有効に働くと考えられるためである．

拡張された特徴空間 $\mathbf{f^{ext}}$ は，既存の特徴空間 $\mathbf{f^{org}}$ と，各補助問題の予測結果のベクトル $\mathbf{c}(\mathbf{f^{org}})$ を並べたものになる．

$$\mathbf{f^{ext}} \leftarrow \{\mathbf{f^{org}},\ \mathbf{c}(\mathbf{f^{org}})\} \tag{3.8}$$

3.3.3.3 線形分類問題における特徴空間の変換

前述の二つの方法によって特徴空間の拡張を行うと，新たな特徴空間 $\mathbf{f^{all}}$ は次のようになる．

$$\mathbf{f^{all}} \leftarrow \{\mathbf{f^{org}},\ \mathbf{Pf^{org}},\ \mathbf{c}(\mathbf{f^{org}})\} \tag{3.9}$$

補助問題のモデルを並べた行列を $\mathbf{M}$ とし，補助問題を次のような線形分類問題であるとすれば，

$$\mathbf{c}(\mathbf{f^{org}}) = \mathbf{M}\mathbf{f^{org}} \tag{3.10}$$

これらの特徴を用いて学習したモデル $\mathbf{w^{all}}$ を用いた主問題の予測結果の計算は次のようになる．

$$\mathbf{w^{all}}^T \cdot \mathbf{f^{all}} = \mathbf{w^{all}}^T \cdot \begin{pmatrix} \mathbf{f^{org}} \\ \mathbf{P}\mathbf{f^{org}} \\ \mathbf{M}\mathbf{f^{org}} \end{pmatrix} = \mathbf{w^{all}}^T \cdot \begin{pmatrix} \mathbf{I} \\ \mathbf{P} \\ \mathbf{M} \end{pmatrix} \cdot \mathbf{f^{org}} \tag{3.11}$$

$$= \mathbf{u}^T \cdot \mathbf{f^{org}} \tag{3.12}$$

ここで簡単のためにバイアス項を 0 としたが，バイアス項があっても同様に計算できる．補助問題と主問題がともに線形部類問題である場合には，両者の重みを合わせたベクトル $\mathbf{u}$ を事前に計算することができるので，補助問題が増えても主問題は**元の線形分類器と同様の計算量で予測可能**である．

今回は上記のように補助問題と主問題がともに線形分類問題であるとしたが，その場合には補助問題がない場合と比べて主問題のモデルの表現力は増加しない．したがって，学習結果は補助問題がない場合と同じ解に収束する可能性があると考えられる．実際には (3.13) 式のように学習の目的関数で重みに対する正則化をかけるため，同じ解に収束することは少ないと考えられるが (付録参照)，表現力が本質的には増加しないことには変わりはない．

$$\text{目的関数} = L(\mathbf{X}, \mathbf{w}) + C\frac{\|\mathbf{w}\|^2}{2} \tag{3.13}$$

L : 損失関数

$\mathbf{X}$: 棋譜の入力データ

$\mathbf{w}$: 重みベクトル

C : 正則化の強さを決める定数

しかし，訓練データが十分に得られない場合は主問題のみではうまく学習できないという現象が起きることが考えられる．例えば，二つの特徴のどちらが有効であるかを主問題のみで区別することができないような場合に，二つのデータを分類できる補助問題があり，さらにその補助問題によって抽出された特徴空間が主問題に対して有効であるならば，主問題のみでは区別できないデータを分類できるようになると考えられる．現実的には十分に学習できるほどの訓練データを利用できることはほとんどないと考えられるため，たとえ線形分類問題であっても，補助問題がない場合よりも学習結果がよくなることがあると考えられる．

3.3.4 評価関数の学習

主問題として，牌譜の局面における打牌間の比較学習を行う．今回は分類器として SVM を使用し，(牌譜の手による局面の特徴 – その他の手による局面の特徴) を正例，(その他の手による局面の特徴 – 牌譜の手による局面の特徴) を負例として学習を行った．

表 3.2: 補助分類問題の例

問題のカテゴリ	問題の数
(敵・自分の) あがり	4
(敵・自分の) 面前でのあがり	4
(敵・自分の) 鳴いてのあがり	4
(敵・自分の) 順位の上がるあがり	4
(敵・自分の) 特定の点数以上でのあがり	4 * ??
(敵・自分の) 特定の役でのあがり	4 * ??
振り込み	1
順位の下がる振り込み	1
(それぞれへの) 振り込み	3

3.4 実験

3.4.1 補助分類問題

主にあがりに関して，次のような 22 種類の補助分類問題を実装した．いずれも局面の特徴として記述するのは困難な要素であると考えられる．

1. 自分があがるかどうか
2. 自分が面前であがるかどうか
3. 自分が鳴いてあがるかどうか
4. 自分が順位の上がる点であがるかどうか
5. 自分が 3900 点以上であがるかどうか
6. 自分が 7700 点以上であがるかどうか
7. 自分が 11600 点以上であがるかどうか
8. 自分が特定の役であがるかどうか (10 種類)
9. 自分が特定のドラ枚数であがるかどうか (5 種類)

3.4.2 特徴要素

表 3.3 に示すような特徴要素を用いた．それぞれの要素について詳細を以下に示す．

- 牌割について
  - それぞれの牌の所持数
- 面子について
  - 手牌に含まれる暗刻の種類ごとの数
  - 手牌に含まれる暗順の種類ごとの数
  - 鳴き牌面子の種類ごとの数
  - 上記の合計数
- 面子候補について
  - 手牌に含まれるトイツの種類ごとの数
  - 手牌に含まれるリャンメン塔子の種類ごとの数

  - 手牌に含まれるカンチャン塔子の種類ごとの数
  - 手牌に含まれるペンチャン塔子の種類ごとの数
  - 手牌に含まれるトイツの総数
  - 手牌に含まれるリャンメン塔子の総数
  - 手牌に含まれるカンチャン塔子の総数
  - 手牌に含まれるペンチャン塔子の総数
- 自分の状態について
  - シャンテン数
  - ドラの所持数
  - ドラそばの所持数
  - テンパイ時の残り当たり牌の数
- 手牌と河の組み合わせについて
  - 河の 1 枚 (席と種類) に対する，手牌の種類ごとの所持数
- 状況別牌危険度について
  - 順位 (4 パターン)
  - 危険度 (親リーチなど危険な攻撃がある，一人リーチ者がいる)
  - 手の進み具合 (5 枚以上待ちテンパイ，4 枚以下待ちテンパイ，ノーテン)
  - 以上の $4 \times 2 \times 3$ パターンそれぞれについて，現物，スジ，オタ風，役牌，無スジ，現物 (ドラ)，スジ (ドラ)，オタ風 (ドラ)，役牌 (ドラ)，無スジ (ドラ) の所持数

表 3.3: 元となる特徴要素

カテゴリ	要素数
牌割	38
面子	239
面子候補	133
自分の状態	6
手牌の 1 枚と河の 1 枚の組み合わせ	5776
状況別牌危険度	240

これらの特徴は比較学習に用いるため，比較する局面間で差の出ないものは学習されないので加えても意味がない．ただ，比較学習には無意味であっても補助問題の学習には有効なものが存在する可能性はあると考えられるが (例えば，現在の持ち点や，親であるかなど)，今回は簡単化のためにそれらの特徴は用いなかった．

3.4.3 訓練データ

学習に用いる訓練データは次のようにして作成した．

- システマティック麻雀研究所[1] で公開されているとつげき東北氏の牌譜の中からランダムに最大 200 試合を取り出す．
- その中に含まれるリーチしていない打牌の局面 (打ち手問わず) を取り出す．

この方法で 200 試合の場合に 60370 局面が抽出された．

3.4.4 実験環境

実験には補助問題の作成・学習を並列実行するため，広域分散クラスタ環境である InTrigger [2]を利用した．実際に利用したのは kyushu，hiro，mirai の 3 クラスタで，マシン環境は CPU Xeon E5410 2.33GHz 4Core Dual，メモリ 16GB である．ただし，200 試合分の牌譜の学習のみはメモリ消費量が大きいため，別に CPU Opteron 2216 2.4GHz 2Core Dual，メモリ 32GB の環境で行った．

実装には C++言語を用いた．SVM の実装としては LIBLINEAR [3] を使用した．学習法は L2 損失・L2 正則化・SVM(双対) を用いた．各種パラメータはデフォルト値を用いた．行列の特異値分解の実装としては SVDLIBC [4] を使用した．

3.4.5 牌譜との一致率・不一致度

学習した評価関数によって選ばれた最善手と上級者の牌譜の手との一致率および不一致度を評価する実験を行った．不一致度とは，牌譜の手の予測順位が平均して 1 位からどのくらい離れているかを示す．例えば，牌譜の手が 1 位なら不一致度は 0，2 位なら不一致度は 1 となり，それの平均を算出する．テスト用の牌譜データとしては，3.4.3 と同様の方法で別途に 50 試合分の局面を抽出した．

以下，3.4.5.1 では 22 個の補助問題を用いた場合，3.4.5.2 ではそれぞれの補助問題を残りツモ枚数ごとに別々に作成した場合，3.4.5.3 では分類性能の高い補助問題のみを選別した場合についての実験結果を示す．

[1] http://www.interq.or.jp/snake/totugeki/
[2] http://www.intrigger.jp/wiki/index.php/InTrigger
[3] http://www.csie.ntu.edu.tw/~cjlin/liblinear/
[4] http://tedlab.mit.edu/~dr/SVDLIBC/

3.4.5.1 22 個の補助問題を用いた場合

3.4.1 に示した補助分類問題を用いて特徴空間の拡張を行った場合の実験結果は図 3.7，図 3.8 のようになった．

最終的に一致率はモデルのみを用いた拡張で最大の 52.3%を記録した．しかし，モデルのみを用いた場合は学習結果が不安定で，一致率が大きく変動する様子が見られる．

一方で予測結果のみを用いた場合は，一致率は 46.9%とやや低いものの，非常に安定して学習が進行していると言える．また，不一致度も補助問題を用いていない場合よりも良くなっている．

モデルと予測結果の両方を用いた場合は，学習の安定性もやや低く，最終的な一致率も補助問題を用いていない場合と同程度にとどまった．また，不一致度に関しても補助問題を用いていない場合よりも悪くなってしまった．

3.4.5.2 大量の補助問題

補助問題は比較的容易に増やすことができる．3.4.1 に示した補助分類問題を，残りツモ数ごとに別々に作成した．これによって補助問題の合計数は 396 になった．これで同様に学習を行い，一致率と不一致度を計測した結果は図 3.9，図 3.10 のようになった．

全体的に一致率，不一致度ともに悪化が見られる．特に学習局面数が多い時の一致率の低下が顕著である．この原因としては，補助問題によっては訓練データが十分でなくてあまり学習できていないということや，過学習を起こしていることが考えられる．

3.4.5.3 補助問題の絞り込み

そもそも補助分類問題がある程度学習できていなければ，それを利用しても意味がない．そこで，得られた補助分類問題の分類器の性能を測定し，ある程度高い精度が出たものだけを絞り込んで使用するようにした．具体的には，訓練データに対して 4-fold cross validation を行い，22 個の補助問題から F 値が 0.4 以上のものを 12 個選び出した．また，残りツモ数ごとに作成した 396 個の補助問題からも，同様に 4-fold cross validation を行い，こちらは F 値が 0.1 以上のものを 24 個選び出した．これで同様に学習を行い，一致率と不一致度を計測した結果は図 3.11，図 3.12，図 3.13，図 3.14 のようになった．

予測結果のみを用いた場合では全体的に一致率が上昇し，最終的に 50.6%に達した．さらに不一致度も安定して補助問題なしよりも良い傾向を示した．

モデルを用いた場合では不安定さがやや抑えられたものの，最終的な一致率も低下し，補助問題を用いていない場合とほぼ同じとなった．

モデルと予測結果の両方を用いた場合については，一致率に関しては訓練データが少ないところでやや改善が見られたものの，最終的には補助問題なしと同じところに落ち込んだ．また，不一致度は一貫して補助問題なしよりも悪い傾向を示した．

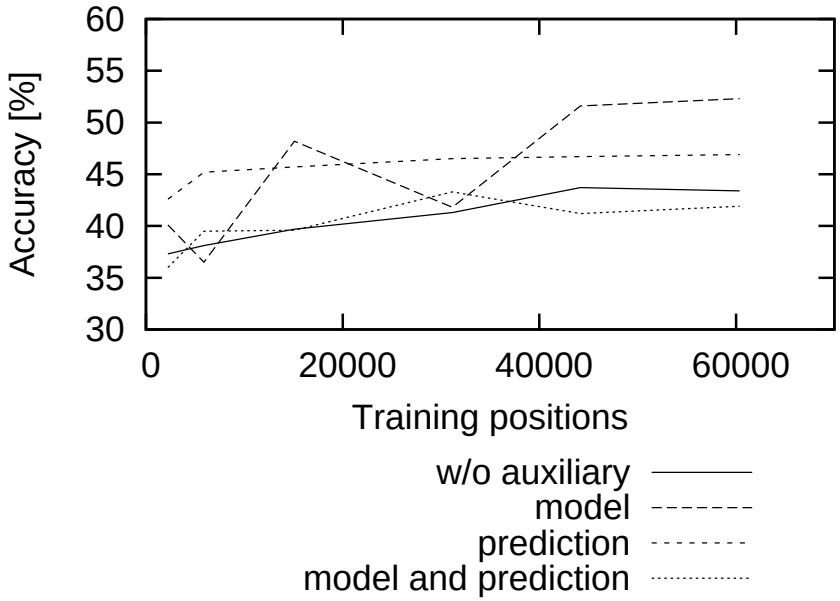

図 3.7: 一致率 (22 個の補助問題を用いた場合)

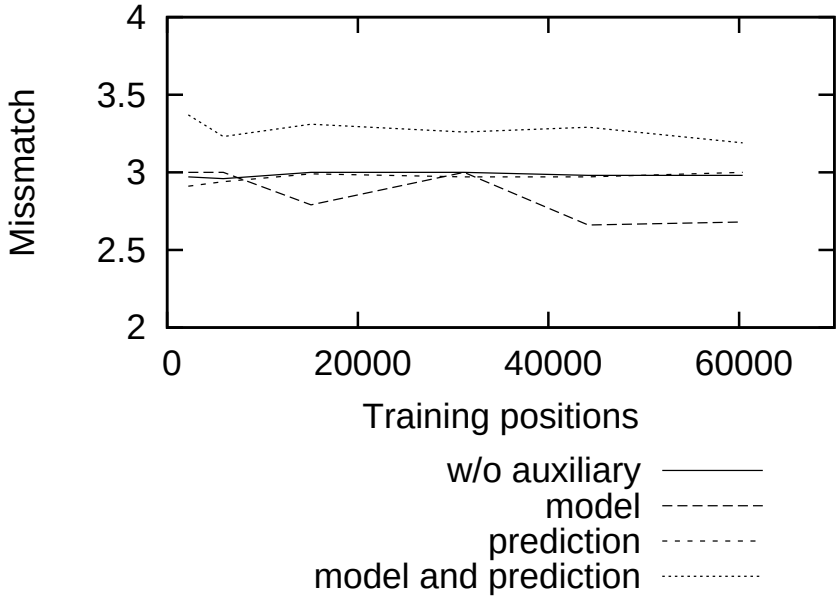

図 3.8: 不一致度 (22 個の補助問題を用いた場合)

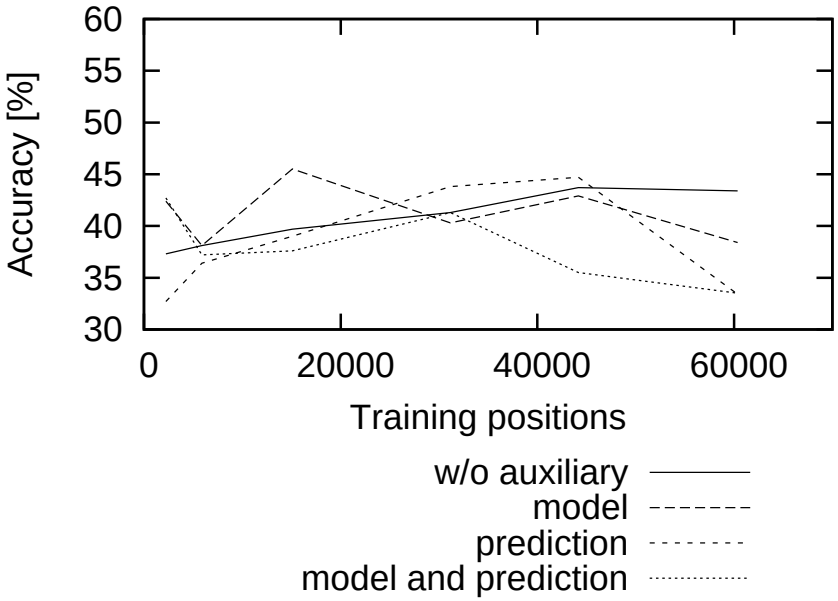

図 3.9: 一致率 (残りツモ数ごとに 396 個の補助問題を作成)

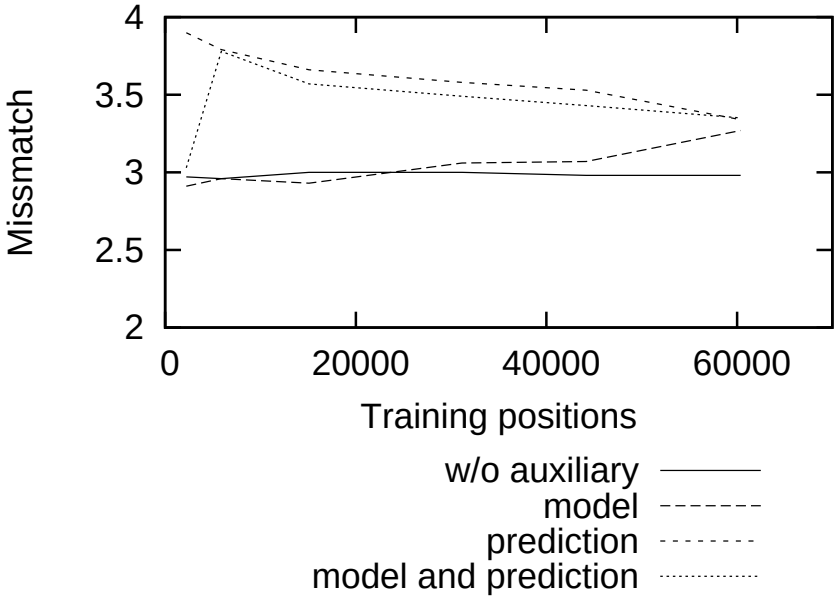

図 3.10: 不一致度 (残りツモ数ごとに 396 個の補助問題を作成)

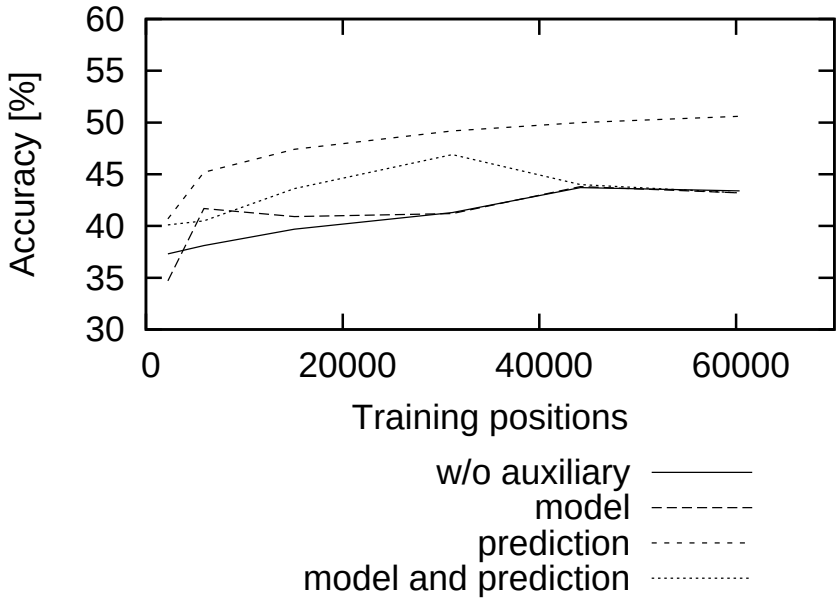

図 3.11: 一致率 (22 個から F 値 0.4 で絞り込み)

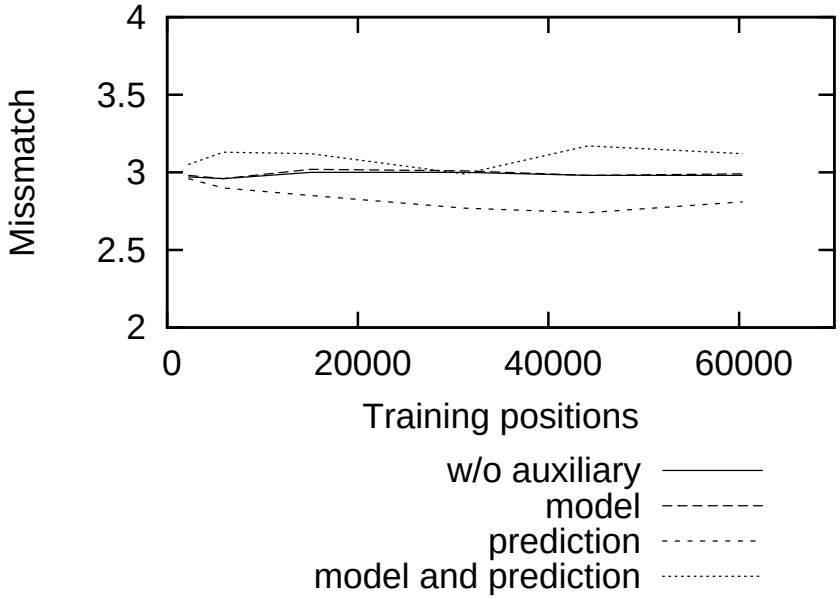

図 3.12: 不一致度 (22 個から F 値 0.4 で絞り込み)

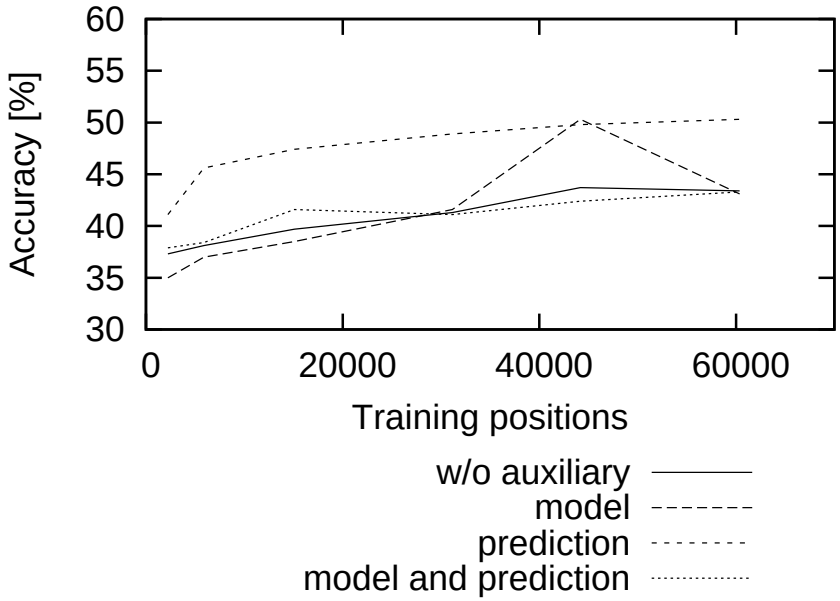

図 3.13: 一致率 (396 個から F 値 0.1 で絞り込み)

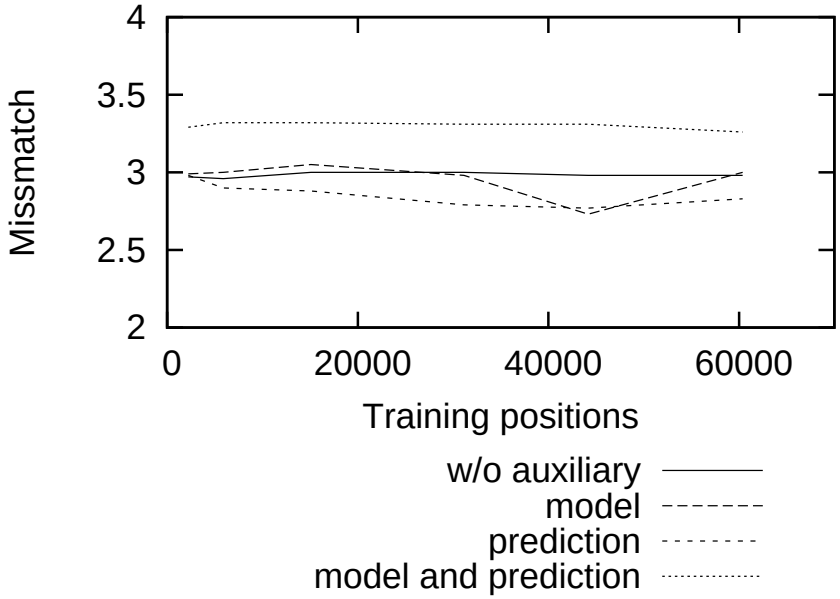

図 3.14: 不一致度 (396 個から F 値 0.1 で絞り込み)

3.4.6 対戦

学習した評価関数のみで手を選択するプレイヤを作成し，評価対戦を行った．対戦は鳴きなし，連荘なしの東風戦で行い，ゲームの不確定性による成績のばらつきを抑えるために 1 試合中の 4 局では全く同じ配牌とツモ山になるようにした．対戦は一度に 2 種類のプレイヤを用い，片方を 1 人ともう片方を 3 人の形で対戦させた．

対戦相手は次の 2 パターンを行った．

- 予測結果を用いた拡張 (22 個から F0.4 で絞り込んだ 12 個) 対 補助問題なし
- モデルのみを用いた拡張 (22 個の補助問題) 対 補助問題なし

結果は表 3.4，表 3.5 のようになった．予測結果を用いた拡張では補助問題なしに対する優位性は見られなかったが，モデルのみを用いた拡張では和了率・平均収支とも 1 人だけ高くなっており，良い手を選択していると考えられる．

表 3.4: 予測結果を用いた拡張 (22 個から F0.4 で絞り込んだ 12 個) 対 補助問題なし (400 試合)

プレイヤ	prediction	w/o A	w/o B	w/o C
和了率 [%]	17.88	19.44	18.81	16.69
平均収支	37	-46	166	-141

表 3.5: モデルのみを用いた拡張 (22 個の補助問題) 対 補助問題なし (400 試合)

プレイヤ	model	w/o A	w/o B	w/o C
和了率 [%]	20.13	18.31	17.44	17.75
平均収支	185	21	-107	-84

第4章　結論

4.1　まとめ

本研究では，多人数不完全情報ゲームである麻雀の打牌を決定するための手法として，UCT 探索を利用した打ち手の探索と，大量の補助問題を利用した評価関数の学習を提案した．

4.1.1　UCT による打ち手の探索について

UCT 探索を利用した探索では，局面の見えていない部分の情報をランダムに生成し，UCB1 アルゴリズムにしたがって木を探索し，末端ではランダムなプレイアウトを行うことで，不完全情報や不確定要素，多人数戦に対しても有効な局面の探索アルゴリズムを実装した．

アルゴリズムのパラメータとして，報酬，UCB 値のバランスパラメータ C の最適値を求める推定実験を行った．その結果，報酬は得点収支をそのまま用い，C は 1000 に設定したときに探索結果の牌譜との一致率が最も良くなり，この条件下で UCT アルゴリズムを 100000 回ループさせて探索したところ，14 牌ノード局面 (切りの局面) での一致率は 46 %に達した．一方，13 牌ノード局面 (鳴きの局面) ではループ回数が増えると一致率は低下した．

コンピュータプレイヤと対戦させたところ，グリーディプレイヤに対しては和了率，平均収支ともに上回り，線形 SVM プレイヤに対してもほぼ互角の性能を出すことができた．

以上の実験により，本研究では次のような知見が得られた．

- 多人数不完全情報ゲームに対する UCT の有効性
- 適切な報酬の選択の必要性
- 適切な UCB 値のパラメータ C の選択の必要性

まず多人数不完全情報ゲームに対する UCT の有効性について説明する．今回実装した探索アルゴリズムは比較的ナイーブなもので，局面の不完全情報の推定やプレイアウトを完全にランダムに行っており，特に知識を用いた改良を行っていない．それにも関わらず，牌譜との一致率を 50%まで引き上げることができたのは，報酬に基づいたランダムシミュレーションによる探索自体が，麻雀に対してそれなりに有効であるということを示していると言える．知識を必要としないという点は，他の研究の浅いゲームや他の分野への応用が効くという意味で非常に重要である．

次に適切な報酬の選択の必要性について説明する．知識をほとんど用いない UCT 探索において，終局の報酬というのは非常に貴重な情報源である．今回は得点収支をそのまま用いるのが最適とい

う結果になったが，麻雀にはその他にも報酬として使える指標が多く存在すると考えられ，それらをうまく利用して報酬を構成することで，よりよい探索を行うことができるようになると考えられる．

最後に適切な UCB 値のパラメータ C の選択の必要性について説明する．今回の実験で得られた最適な C の値は 1000 であった．(2.1) 式にある二つの項のうち，第 1 項は獲得報酬に基づいた量で，第 2 項は探索回数に基づいた量である．囲碁のように獲得報酬に [0,1] の値を使う場合は，C は $\sqrt{2}$ 程度が良いとされている．一方で今回は獲得報酬に得点収支を用いたが，これはおおよそ 1000 のオーダーの値である．この結果，麻雀の UCT においても，(2.1) 式の二つの項のスケールの比が囲碁と同じようなところで最適になることがわかった．

4.1.2 補助問題を利用した評価関数の学習について

補助問題を利用した評価関数の学習では，牌譜に含まれる情報を抽出するために補助分類問題を作成し，それを学習して得られたモデルから主成分分析で取り出した特徴空間と，補助分類問題の予測結果で構成される特徴空間を追加して学習を行った．

結果として，一致率はモデルによる拡張のみを用いた場合に最高で 52.3%を記録したが，学習結果は不安定であった．一方で予測結果による拡張のみを用いると，補助問題を用いていない場合よりも高い性能を発揮しながら，学習も非常に安定して進行した．

また，利用する補助分類問題を分類性能によって絞り込んだところ，予測結果のみを用いた拡張はさらに性能が向上し，一致率は 50.6%に達した．

補助分類問題を残りツモ数ごとに作成することで数を増やした実験については，あまり効果が得られなかった．

本研究で得られた一致率は先行研究 [21][18] よりも若干低い．この原因としては，先行研究が，

- 特徴要素が異なる
- 非線形な学習をしている
- 打ち手のレーティングが高い訓練データのみを使用している

ということが影響していると考えられる．

以上の実験により，本研究では次のような知見が得られた．

- 予測結果を用いた特徴空間の拡張の有効性
- 線形補助分類問題による拡張の有効性

まず予測結果を用いた特徴空間の拡張の有効性について説明する．SVD-ASO で提案されたようなモデルを用いた拡張に加えて，本研究では補助分類問題の予測結果をそのまま特徴として加える手法を提案した．実験結果として，予測結果による拡張ではモデルによる拡張よりも学習が非常に安定して進行するということがわかった．『人手で抽出することが難しい特徴』を補助分類問題の予測結果として取り出すことは，牌譜からの情報抽出の方法として，とても直感的でわかりやすく，かつ有効な手法になり得ると考えられる．

次に線形補助分類問題による拡張の有効性について説明する．3.3.3.3 で説明したように，主問題と補助問題が線形である限り，特徴空間の拡張を行っても本質的にモデルの表現力は増加しない．しかし，実験の結果，予測結果を用いた拡張では補助問題を用いていない場合よりも安定して高い性能を示している．これにより，線形分類問題のみによる特徴空間の拡張でも，現実的な訓練データの範囲ではよりよい学習ができる可能性があることがわかった．

4.1.3 総合して

冒頭に挙げた，多人数不完全情報ゲームの麻雀における困難な点について再掲する．

- 多人数ゲーム
  - 状況によってそれぞれのプレイヤの目的が異なる場合がある (単純な零和ゲームではない)
- 不完全情報ゲーム
  - 局面に不完全情報や確率的要素があるため，探索の分岐数が膨大である
- 研究の浅いゲーム
  - 評価関数の特徴要素の構築や重み付けが難しい

本研究では，これらの問題のうち，2 章において多人数不完全情報ゲームとしての問題を解決する手法を提案した．また，3 章において研究の浅いゲームの問題である評価関数の構築と重み付けを解決する手法を提案した．

ここで麻雀固有の問題として，状況によって大きく手の方向性を変える必要があり，評価関数を特徴要素の線形和で精度良く表すことが難しいという点が残っている．一般的に，攻めるべきところではリスクを無視して収益が最も見込める手を選び，降りるべきところでは徹底的に降りる，という打ち方が優れていると言われている．一つの評価関数のみに基づいて手を決定していると，リスクリターンが微妙なときに中途半端な手を選択してしまうことが多いと考えられる．

本研究で作成したコンピュータプレイヤをインターネットの麻雀対戦サーバである東風荘[1]で対戦させてみたところ，いずれもレーティングが初期の 1500 から，1000〜1100 程度にまで低下してしまった．フリテンなどが存在する麻雀では，一手の悪手が致命的になることが多いため，50%程度の一致率では，まだ人間と満足に対戦できるレベルには至っていないと考えられる．

また，本研究では探索と評価関数という二つの側面から良い手を求めることを試みた．探索と評価関数はコンピュータゲームプレイヤの性能を決定する重要な要素である．完全な探索もしくは完全な評価関数が利用できれば，どちらかだけで事足りるが，現実的にはどちらも困難である．そのため，探索の打ち切り局面で評価関数を利用するなど，両者を併用することでお互いを補い合うことが有効であるとされている．しかし本研究では問題を細分化するため，探索のみ，もしくは評価関数のみに絞った実験と評価を行った．それに対し，探索と評価関数の併用として，本研究で提案した 2

[1] http://mj.giganet.net/

つの手法を組み合わせることもできると考えられる．例えば，UCT における不完全情報の推定やプレイアウトの改善に，補助問題を用いて学習した評価関数を利用することなどが考えられる．

4.2 今後の課題

今後の課題としては次のようなことがあげられる．

- UCT 探索による打ち手の探索について

  不完全情報の推定 今回は不完全情報を仮定するときに相手の手牌をランダムに振り分けており，終盤でも相手の手があがりから程遠いという不自然な状態になってしまっている．そこで相手の手牌を確率的に推定するなどして，より現実的な局面を生成して探索の質を高めることが考えられる．

  プレイアウトの改善 プレイアウトの部分にパターンなどの知識に基づいた着手や評価関数を組み入れてシミュレーションの質を向上させることが考えられる [17]．その際には，今回用いた SVM による評価関数を利用することも考えられる．

  探索効率の向上 探索効率を向上させる方法として，囲碁で用いられている UCT の改良手法を応用することも考えられる．例えば異なる順序の着手で到達した等しい状態の局面は同一のノードとして更新を行う RAVE (Rapid Action Value Estimation) と呼ばれる手法などを応用することが考えられる [6]．さらに探索時間の有限性を利用した枝刈りによって探索効率を改善するなど [20]，様々な手法の適用が考えられる．

- 補助問題を利用した評価関数の学習について

  補助問題の改善 今回実装した補助問題はあがりに関するものばかりで，数もあまり多く作成できなかった．線形分類問題では計算量が変わらないという利点もあるため，より多くの有効な補助問題を活用することでさらに学習を改善できると考えられる．

  非線形学習 今回は学習全体のフレームワークの検証的な意味合いから，扱いやすい線形分類学習を利用した．しかし，補助問題の利用によって実質的に表現力を増やすためには，非線形な学習を取り入れることが必要であると考えられる．例えば，補助問題は線形分類のままで，主問題のみを非線形問題にすることなどが考えられる．

  他のゲームへの応用 補助問題を利用して特徴空間を拡張するというフレームワークは，麻雀に限らず他のゲーム・分野にも応用が効くと考えられる．例えば，人手での特徴要素の抽出が難しい囲碁などのゲームに対して，本手法は有効である可能性があると考えられる．

- 他の問題として

  より実戦的なプレイヤの設計 先行研究を含めて，作成されたプレイヤが実戦では強くないことを考えると，麻雀では一つの評価関数にしたがって手を決定するのは難しいと予想される．学習器に非線形なものを使用するという方法も考えられるが，例えば，降りるべきかどうかを判断する分類器や相手の待ちを予測する分類器などを利用することが考えられる．それらの出力によって使用する評価関数を切り替えるなど，有効とされている打ち方に近くなるようにプレイヤそのものを設計していくことが必要だと考えられる．

付録

線形分類問題における正則化による補助問題の効果

元の特徴空間を 2 次元であるとし，その重みベクトルを $\mathbf{w}^{\mathbf{org}} = (w_0\ w_1)^T$ とする．補助問題による拡張は 1 次元であるとし，その重みを w_a とする．補助問題における元の特徴の重みベクトルを $\mathbf{a} = (a_0\ a_1)^T$ とする．

補助問題がない場合，元の特徴のみでは

$$\|\mathbf{w}^{\mathbf{org}}\| = \sqrt{w_0^2 + w_1^2} \tag{1}$$

に対して正則化がかかるので，原点を中心とする円状に正則化の力が働き，なるべく原点に近いところで収束させようとする．

補助問題がある場合には，

$$\|\mathbf{w}^{\mathbf{all}}\| = \sqrt{w_0^2 + w_1^2 + w_a^2} \tag{2}$$

に対して正則化がかかるので，原点を中心とした球面状に正則化の力が働く．

ここで，線形分類の場合には次のような式変形が可能である．

$$\begin{aligned} \mathbf{w}^{\mathbf{all}T} \cdot \mathbf{f}^{\mathbf{all}} &= \mathbf{w}^{\mathbf{all}T} \cdot \begin{pmatrix} \mathbf{f}^{\mathbf{org}} \\ \mathbf{a}\mathbf{f}^{\mathbf{org}} \end{pmatrix} = \begin{pmatrix} w_0 & w_1 & w_a \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ a_0 & a_1 \end{pmatrix} \cdot \mathbf{f}^{\mathbf{org}} && (3) \\ &= \mathbf{u}^T \cdot \mathbf{f}^{\mathbf{org}} && (4) \end{aligned}$$

$$\mathbf{u} = \begin{pmatrix} w_0 + w_a a_0 \\ w_1 + w_a a_1 \end{pmatrix} \tag{5}$$

(5) 式より，結合された新しい重みベクトルである $\mathbf{u}$ に対する正則化は，$(w_a a_0,\ w_a a_1)$ が中心となることがわかる．つまり補助問題を加えると正則化にバイアスがかかり，原点から偏ったところ (補助問題で最適だったところ) に収束させようとする力が働くことになる．正則化の強さは，$\mathbf{w}^{\mathbf{all}}$ の正則化空間を 3 次元空間上で $z = |w_a|$ の面で切った切断面に相当するので，補助問題の重みが大きいほど正則化が強くかかる．

参考文献

[1] R.K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, Vol. 6, pp. 1817–1853, 2005.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, Vol. 47, No. 2-3, pp. 235–256, 2002.

[3] BW Ballard. *-Minimax search procedure for trees containing chance nodes. *Artificial Intelligence*, 1983.

[4] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. p. 100, 1998.

[5] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, Vol. 20, No. 3, pp. 273–297, 1995.

[6] S. Gelly and D. Silver. Combining online and offline knowledge in UCT. *Proceedings of the 24th international conference on Machine learning*, pp. 273–280, 2007.

[7] T. Hauk. Search in trees with chance nodes. Master's thesis, University of Alberta, 2004.

[8] T. Hauk, M. Buro, and J. Schäfer. Minimax performance in backgammon. Vol. 3846, pp. 51–66, 2004.

[9] T. Joachims. SVM-LIGHT: an implementation of Support Vector Machines (SVMs) in C. *http://svmlight.joachims.org/*.

[10] T. Joachims. Optimizing search engines using clickthrough data. *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), ACM*, 2002.

[11] L. Kocsis and C. Szepesvari. Bandit based monte-carlo planning. *European Conference on Machine Learning*, pp. 282–293, 2006.

[12] C. Luckhardt and K. Irani. An algorithmic solution of n-person games. *AAAI*, Vol. 1, pp. 158–162, 1986.

[13] J. Schäfer, M. Buro, and K. Hartmann. The UCT Algorithm Applied to Games with Imperfect Information. *Diploma thesis. Otto-von-Guericke-Universität Magdeburg*, 2008.

[14] N. Sturtevant. Last-Branch and Speculative Pruning Algorithms for Maxˆ n. *IJCAI*, Vol. 669–678, , 2003.

[15] N.R. Sturtevant. An Analysis of UCT in Multi-player Games. *Proceedings of the 6th international conference on Computers and Games*, pp. 37–49, 2008.

[16] G. Tesauro. Connectionist learning of expert preferences by comparison training. 1989.

[17] Y. Wang and S. Gelly. Modifications of UCT and sequence-like simulations for Monte-Carlo Go. *IEEE Symposium on Computational Intelligence and Games, 2007. CIG 2007*, pp. 175–182, 2007.

[18] 三木理斗, 三輪誠, 近山隆. 木カーネルを用いた麻雀打ち手の順位学習. *Proceedings of 13th Game Programming Workshop*, pp. 60–66, 2008.

[19] 保木邦仁. 局面評価の学習を目指した探索結果の最適制御. *Proceedings of 11th Game Programming Workshop*, pp. 78–83, 2006.

[20] 北川竜平. 投入計算量の有限性に基づく UCT 探索の枝刈り. Master's thesis, 東京大学, 2009.

[21] 北川竜平, 三輪誠, 近山隆. 麻雀の牌譜からの打ち手評価関数の学習. *Proceedings of 12th Game Programming Workshop*, 2007.

発表文献

[1] 三木理斗, 三輪誠, 近山隆. 木カーネルを用いた SVM による麻雀打ち手の順位学習. 第 13 回ゲーム・プログラミング ワークショップ, pp. 60–66, 2008.

[2] 三木理斗, 三輪誠, 近山隆. UCT 探索による不完全情報下の行動決定. 第 14 回ゲーム・プログラミング ワークショップ, pp. 43–50, 2009.

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.

近山隆教授には，研究についてのご指摘や発表における様々なアドバイスをいただきました.

元近山研究室博士の三輪誠先輩には，研究の方針や手法について様々なアドバイスをいただいたり，研究の参考となる文献を数多く紹介していただきました.

他の近山・田浦研究室の皆様にも様々な助言をいただき，またとても刺激になる議論をすることができたりと，本当にお世話になりました.

また，本研究の一部は文部科学省科学研究費補助金特定領域研究「情報爆発に対応する高度にスケーラブルなソフトウェア構成基盤」の助成を得て行われました.

この場をお借りして厚くお礼申し上げます.

平成 22 年 2 月 9 日