学位論文

ゲーム知識を表現する語彙の棋譜データからの自動獲得

(Automatic acquisition of vocabularies for representing game knowledge from game records)

概要

コンピュータゲームプレイヤは人工知能の分野においてルールの決まった世界におけるアルゴリズム検証の場として長年にわたり研究されてきた．この長年の研究において，コンピュータゲームプレイヤを強くするという研究はその目的の明確さから広く行われてきた．このため多くのゲームにおいて強いプレイヤが作成されている．このような強いプレイヤの作成には一般化された効率的な探索手法とその対象とするゲーム特有の知識が必要とされる．この知識は評価関数や探索手法の効率化，知識データベースなど様々な場面において利用されるものである．

このようにゲーム特有の知識はコンピュータゲームプレイヤを強くするにあたり必要不可欠なものである．その知識の獲得には過去のゲームや探索結果などの過去の経験を入力データとして機械学習などの手法を用いて獲得する手法が広く研究されている．このような入力データは獲得対象となる知識を得るのに適した表現をしている必要がある．ここで問題となるのが，この知識を得るのに適した表現を実現する，その知識の含まれたデータやその知識そのものを上手く表現するための語彙の獲得である．この語彙としては現在は多くのコンピュータゲームプレイヤにおいては人手で選択したものが用いられている．この選択を行う際にはこの語彙にはその過不足や選択の難しさの問題があり，この語彙が不十分であると知識を得るのに必要な入力データそのものには含まれているであろう情報が欠落してしまう，もしくは重要な情報を抽出することが難しく知識が得られないという結果になる．またあるゲームで用いられた手法を別のゲームに適用したい場合にも語彙を前提とした手法であることが多く，その語彙の取得方法が不明瞭なため容易には適用できないという問題もあり，ゲーム研究全体の進展における大きな障害の1つとなっている．最後にゲームにおいて学習を行う際には少ないデータで良い結果が得られている例は少なく，大量の学習例・語彙を扱うことが一般的に行われている．これはゲームにおいては棋譜や自己対戦などから多くの学習例を獲得できるためである．しかし，一方で大量の学習例を用いた学習はそのコストが高く難しいため，多くの有用な機械学習手法の適用が困難になってしまっている．

本論文ではこのような対象ゲームや対象問題への語彙の不足・語彙の取得方法の欠如・大量例からの学習に関する問題に対処するために，ゲームの知識を表現できる適切な語彙を棋譜データから獲得する手法について提案する．本手法により語彙の取得方法を提案した上で語彙の不足を解消することができると共に適切な語彙に縮約することで大量の学習例を問題に必要な情報のみであらわされた縮約された学習例に変換することが可能となる．提案手法の前提として，一般に語彙を無から取得することは不可能であるので，獲得

する語彙を表現する対象ゲームについての基本的な語彙は与えられているものとする．この語彙は対象ゲームのルールなどから獲得可能なものである．この既存の語彙で表現された棋譜データをもとにその共起する語彙を組み合わせることで不足している語彙の獲得を行う．この棋譜データにおける共起として主なものには指し手列に代表される時系列的な共起関係と同局面に現れる特徴に代表される空間的な共起関係があげられる．本論文ではこの棋譜データからの語彙の獲得手法として，この2つの共起関係それぞれついて提案し，評価を行った．1つは指し手列に起こる共起関係について注目し語彙を時系列的に組み合わせて新たな語彙を獲得するという手法を，指し手列の解析に基づく将棋における指し手の分類精密化手法において提案する．もう1つは局面に同時に現れる語彙から新たな語彙を発見するという手法を，コンピュータゲームプレイヤにおける評価要素の自動生成において提案する．

指し手列の解析に基づく将棋における指し手の分類精密化ではコンピュータ将棋プレイヤ「激指」において利用されている指し手をその性質ごとに分けたカテゴリを既存の語彙として指し手列において連続して起こる指し手列を基に指し手のカテゴリを拡張する．実現確率打ち切り探索とは深さの代わりに探索するルート局面からのその局面へ至る確率 (実現確率) を閾値として探索を行う手法である．実現確率はあるカテゴリの手がその局面で選択される遷移確率を用いて計算される．カテゴリは指し手の性質に基づいて分けたものであり，その遷移確率は棋譜データから計算している．このカテゴリは局面と指し手から決定される．他の多くのゲームと同様に将棋においても重要な指し手列が存在していると言われている．この重要な指し手列には手筋などと呼ばれるように連続した指し手において特に特徴的なものが存在し，プレイヤに取り入れられている．実現確率を計算するカテゴリにもいくつかの決まった手筋は人手で入力されているが，その統一的な扱いや定式化はほとんどなされていない．このようなことから多くの棋譜を基に指し手の履歴としてカテゴリの履歴を抽出し，それをもとにこれまでの実現確率探索におけるカテゴリの遷移確率を指し手の履歴情報を考慮したカテゴリの遷移確率として拡張する．評価としてプロやインターネット将棋サーバの強いプレイヤの対局結果の棋譜 47,321 局から抽出したこの遷移確率を激指に実装し，1手5秒・220手で引き分けのルールで，この拡張を行った遷移確率を用いたコンピュータ将棋プレイヤとこの拡張を行わず従来どおりカテゴリの遷移確率を用いたプレイヤで 150 試合対戦を行った．また次の一手問題集への解答についても評価を行った．結果としては 83 勝 67 敗と元のプレイヤに勝ち越すことができた．またこの拡張を行ったプレイヤは元のプレイヤよりも次の一手問題集においてほぼ一手深く読んだプレイヤと同等数の問題を正解することができた．

コンピュータゲームプレイヤにおける評価要素の自動生成では既存の語彙として局面を区別して表現できる基本的な語彙を既存の語彙として棋譜に現

れる局面に同時に現れる語彙を抽出し，選択することで局面を評価する評価関数における評価要素を作成する．評価関数はコンピュータゲームプレイヤにおいて探索結果に直接影響するコンピュータゲームプレイヤにおいて対象ゲームの知識を表現する主要部分である．このような評価関数における評価要素はこれまで多くのゲーム，特に将棋・囲碁などの比較的複雑なゲームにおいては人手で選択するのが一般的である．この人手での選択には対象とするゲームに関する深い知識が必要であり，そのゲームについて知らなければ作ることすらできない上に，知っていたとしてもコンピュータゲームプレイヤ作成者に大きな負担のかかる部分である．本手法では多くのゲームに応用可能な対局の勝ち・負けや詰み問題の詰み・不詰などの2クラスの分類問題を対象として既存の語彙のうち共起したものを組み合わせたのち選択することで評価要素を作成する．既存の語彙は2値であらわしたものを用い，局面に共起したものを論理積で表現し，そのうち頻度と条件付き相互情報量を用いて選択したものを評価要素とする．この生成された評価要素にこれまでに提案されている手法を用いて重みづけを行うことで評価関数の作成が可能である．また，ゲームでは既存の語彙が多い場合や学習対象とする棋譜が多い場合が多く，既存の多くの機械学習における特徴生成手法の適用が困難であった．本手法では問題を分割して処理する方法を同時に提案しており，このような多くの手法が適用困難な問題に対しても適用可能である上に並列化によって高速に処理を行うことも可能となっている．評価としては200,000局分のOthelloの局面を用い，Othelloの位置と色の192個の既存の語彙から評価要素の作成を行った．これにより分割処理によって大きな問題を扱うことができるとともに並列化により高速に実行できることを確認できた．また，生成した評価要素を用いたNaive Bayesian分類器は他の分類器による単純な特徴を用いた分類よりも良い精度で分類できることを示し，有用な評価要素が得られていることを確認した．

本論文ではこのような既存の語彙を共起関係に基づいて拡張する2つの手法の提案とその評価を通して，コンピュータゲームプレイヤにおいて棋譜の中で同時に起こる共起関係を基に既存の語彙を拡張することで新たな語彙の獲得が可能となることを示した．またその語彙を用いることで既存の語彙のみを利用した知識よりも有用な知識が得られることを示した．この拡張はこれまで多く人手でなされてきたことの代替となるものであり人手での抽出の負担を軽減することができ，また人手の偏った知識に依らず一般的な指標をお用いて拡張することができる．また特定のゲームに依存しない方法で語彙の獲得ができるため，多くのプレイヤに適用できる．さらにそれと共に大規模データについても語彙を獲得できる手法を提案することでこれまで対象とすることが難しかったゲームについても適用できる．

対象ゲーム・対象問題に適した語彙なしには対象問題に対する知識を表現することは難しい．このため対象ゲームへの語彙の獲得なしにはコンピュー

タゲームプレイヤの研究における強いコンピュータゲームプレイヤの一般的な手法での作成という課題の解決はできないと考えられる．本論文ではそのような課題の解決策の１つとして，既存の語彙を拡張することで問題対象の知識を記述する語彙をゲームの知識に依らない方法で獲得する手法を提案し，探索の効率化・評価関数について有用な語彙を獲得できることを示した．

謝辞

本研究を進め，本論文をまとめるにあたり，多くの方々にお世話になりました．

特に，近山隆先生には指導教官として，研究内容から論文作成，研究発表など多くの場面において数多くの助言，ご指導をいただきました．また，横山大作先生には研究生活から研究内容，その進め方について日頃から相談に乗っていただきました．

石塚満先生，伊庭斉志先生，喜連川優先生，柴田直先生，豊田正史先生には本論文の審査をしていただき，貴重な助言をいただきました．

また同近山・田浦研究室の皆様にも公私にわたり多くの助言を頂きました．

ここに，心より感謝の意を表します．

目 次

図目次

表目次

第1章　はじめに

1.1　コンピュータゲームプレイヤ

コンピュータゲームプレイヤは1952年にSamuelがコンピュータチェッカープレイヤを作成 [85] して以来，人工知能の分野においてアルゴリズム検証のよいテストベッドとして長年にわたり研究されてきた．このような研究の中でコンピュータゲームプレイヤを強くする研究は他のプレイヤや人間のチャンピオンに勝つという目的の明確さから特に広くなされている．現在，探索手法と評価関数，そのゲームに対する知識を利用した効率化により，いくつかのゲームについて解が得られ，またそれ以外のゲームにおいても多くの強いプレイヤが作成されている．例えば，連珠 (盤面サイズの決まった五目並べ)・Awari・Connect4・チェッカーなどのゲームや5x5の囲碁や6x6のオセロなどのゲームの部分問題が解かれている．また，バックギャモンやオセロ，チェスにおいては人間の世界チャンピオンに勝つという輝かしい成績を残している．それらのゲームより難しいとされる将棋においてはアマチュア5〜6段程度の棋力を持ったプレイヤが作成されている．さらに難しい囲碁においても囲碁を対象として作成されたプレイヤは9路盤においてアマチュア5〜6段，19路盤においてはまだ弱いとはいえアマチュア1級〜初段程度の棋力を持っているとされている．

このコンピュータゲームプレイヤは一般化された探索アルゴリズムとその対象とするゲーム特有の知識を上手く利用することで強くなっている．

1.2　ゲームにおける知識獲得

コンピュータゲームプレイヤにおいてその対象とするゲームに関する知識は局面を評価する評価関数において必要とされる．またそのような知識は評価関数のみならず，その探索において良い手を早く探し必要のない手を探索せずに済むようにするための探索の制御や効率化，探索を事前に長時間行っておくことで得られるような実際のゲーム中では得るのが難しい手順や多くのゲームから有用であると判断された手順などを表現した定石[1]においても利用されるものであり，探索手法と共にコンピュータゲームプレイヤの振舞いやその強さを決定づける最も重要なものである．

[1]囲碁では定石，将棋では定跡であるが本論文では定石で統一する．

この知識については人手で作られたものが使われていることが多いが，自動的に獲得する手法も提案されており，この対象ゲームに関する知識の獲得には過去の経験が利用される．ここで利用されるゲームにおける過去の経験としてはプロや強いコンピュータゲームプレイヤなどの強いプレイヤの行ったゲームの指し手の履歴を記録した棋譜[2]や自己対戦や他との対戦の結果，(長時間かけた) コンピュータゲームプレイヤの探索結果などが挙げられる．このような知識獲得では，過去の経験を学習データとした機械学習手法が用いられる．この学習データは生のデータであり獲得する対象となる知識について考慮されていない．対象とする知識によっては必ずしもすべての局面を区別して表現する必要はないため，このようなデータには対象とする知識の獲得には不要な情報が含まれている．また対象問題を得るために有用な情報がわかりやすく表現されていない．このような問題に対処するためにはデータを対象問題に適した表現にする必要がある．この対象問題に適した表現の基準としては学習器・学習対象・学習モデルに適しているかどうかという基準やデータの相関など学習の代替となる基準が用いられる．この基準に見合ったデータにするために，現在このような対象問題に対し知識獲得を行う際には，局面や手の履歴などの学習データを人手で用意した対象問題の語彙を用いて表現することで，学習に利用可能かつ有用なデータに変形する．このゲームに関する語彙とは学習においては特徴 (属性・素性) とよばれるものである．またこの現在使われている人手で用意した語彙とはその対象問題について人間の持っている知識を元にしたものである．これは例えば将棋における評価関数でよく用いられるものとしては持ち駒の数や王の周りの駒などであり，その獲得したい複雑な知識を表現するのに必要な比較的簡単な知識である．この語彙は対象とするゲームそれぞれに対して特有のものであり，獲得対象となる知識に応じて取捨選択されたものであることが望ましい．

対象ゲームの知識獲得に対して人間の持っているゲームの語彙を用いた機械学習は多くの有用な結果も得られていることから有用である．しかし，このような方法での知識の獲得には大きく次の 3 つの問題がある．

- 対象ゲーム・対象問題に対する語彙の不適切
- 語彙の取得方法の不明瞭
- 大量データの利用

対象ゲーム・対象問題に対して語彙が不適切な問題とは対象とするゲームや問題に関する知識を表現できない語彙や表現はできるものの複雑な表現を行わなければ表現できないような語彙が存在する問題である．この問題には次のようなものがあげられる．

[2]ゲームの指し手の履歴を記録したものは麻雀では牌譜と呼ばれるなど呼称は異なるが本論文では棋譜に統一する．

語彙の不十分 これは学習における生のデータには含まれている情報であり，知識を取得する際に必要となるにもかかわらず，その必要性がわからないもしくはその存在を忘れていたり知らなかったりするためにデータからその情報が失われてしまう場合である．

人手の作業による知識表現の限界 これは人間にはわかるものであり表現が非常に困難な知識や表現は単純かもしれないが曖昧であったり明らかでなかったりする暗黙知のような言語化されない知識が存在するために結果としてその知識を表現することができなくなってしまうことにより起こる語彙の不足である．

語彙の表現力の過不足 これは上の2つにも関係するが語彙の抽象性・具体性に関する問題である．上位の概念にあるような抽象化された語彙は多くのものが含まれるが，その一方で細かい表現は難しい．一方ですべての局面を区別できるような表現をすることは比較的容易であるが，語彙が多くなるとともに，対象問題に対して一般的な知識を得ることが難しくなる．このようなことから抽象性と具体性のトレードオフ点を探る必要があるが，人間が知識を入れる際にはその好みや知識の量によって偏ってしまい，それによってこのような語彙の抽象度のレベルを上手く調整できない．また，人間の対象問題に対する知識の欠如がある場合もある．これは対象問題に対する語彙が必要かどうかわからないため，無いよりはあるほうが良いという比較的安易な考えによって語彙を必要以上に増やしてしまうような場合である．このようなことがあると，細かく表現されるが一般性に欠ける知識や一般性はあるが多くの違いを区別できないような知識が (混在して) 得られるという結果になることが考えられる．また，このような語彙の追加によって，冗長な語彙が含まれてしまうという問題も起こる．このような場合にはデータ・知識の表現方法が複数あり，この表現の自由度のために知識の表現の選択候補が多くなり，知識の表現が難しくなってしまう．

データに対する複雑な処理の必要 これはデータには明確に表れていないが，複雑な処理をすることでその値が明らかになるものである．例えば，先読みの結果などはこのようなものに含まれる．このようなものは重要なものも存在する可能性があるが，語彙の利用のためにそのゲームのプレイ中に実際に得る場合にはその処理時間も考慮して評価する必要がある．

このように語彙の不適切による問題の多くは，人間の知識に依存することで知識に偏りが存在することが原因となっている．

また語彙の取得方法の不明瞭というのはその語彙の取得方法が他のゲームへすぐに適用できるような明らかなものでないということである．これは，現在使われている語彙はその対象ゲームについて人間の持っている知識を元

にしたものであるということが原因である．ゲームにおける知識獲得手法の多くは機械学習という一般的なモデル化の手法を用いているものの，その結果は対象ゲームに対する語彙の選択の結果によって変わってしまう．語彙を前提とした方法でゲーム特有の知識を自動的に獲得する手法は多く提案されているが，対象ゲームによらず一般的な手法で語彙を含めた知識を獲得する，もしくは，その語彙を様々なゲーム・問題を対象として一般化した手法で獲得する，という手法は今のところ提案されていない．このような語彙の選択方法が明らかでない限り同等の結果をすぐに得ることは他のゲームに対しては難しく，対象ゲーム特有の手法となってしまっている．このように知識の獲得はその語彙のために他のゲームに容易に適用できるものではなく，他のゲームにはすぐには利用できない一般的でない手法となってしまっていることは対象ゲームの異なる研究間における大きな壁の１つとなっており，ゲーム研究の進展の大きな妨げとなっている．

最後に大量データの利用というのはゲームの学習に用いる学習データの量についての問題である．データが多い場合はそれを利用できるのであれば利用した方が良いが，機械学習における有用なアルゴリズムには大規模データを対象とした学習が計算量的に困難なものが多い．その一方で棋譜データは日々増え続けている上に自己対戦や探索など計算により非常に多くのデータを得ることができる．このようにゲームでは他の学習問題と違い，その学習データ数を自由に増やすことが可能である．その学習データ数をどのくらいにするべきであるかということは明らかにされていないもののゲームの学習において少ない学習データによって良い結果を得られることは少なく，大規模・高次元のものを扱うことができるものが学習手法として求められる．

こういった知識獲得における問題への対処として対戦結果など過去の経験のデータからの語彙の獲得が有効である．

過去の経験には対象問題に関する情報が含まれており十分な量の過去の経験を利用してその情報を元にデータから冗長な語彙を減らし，不足した語彙をうまく表現・抽出できれば語彙の問題は大幅に解消される．

またこのようにして語彙が一般化された方法で獲得可能となれば他のゲームへの応用もできるようになる．ゲームによらない一般化した手法を用いた(一定の強さをもった) コンピュータゲームプレイヤの作成は，対象ゲームの異なる研究間における大きな壁を取り除くための１つの方法として考えられ，コンピュータゲームプレイヤの研究における大きな課題の１つである．このような目的のために AAAI GGP (General Game Playing) Competition[3]などゲームを記述する言語を規定した上での複数ゲームに対する知識を必要としないプレイヤの作成コンテストなども開かれている．このような中でゲーム特有の語彙を獲得する手法は，一般化した手法でのコンピュータプレイヤの作成のために欠かせない現在最も必要とされているものの１つである．

[3] http://games.stanford.edu/

最後に大規模データの扱いについても，語彙の表現力の過不足について説明したとおり，用いている語彙が必要かどうかわからないため追加され，不必要な情報や重複した情報が増えてしまっているため，データが冗長となりデータの規模がその持っている情報以上に大きくなってしまっていることが考えられる．対象問題に適した語彙をうまく表現することができれば，この不要な情報・重複した情報を削減することで大規模なデータを縮約することが可能となる．また，このような表現が可能となれば，複雑なモデルを前提とした学習器を用いずとも，線形モデルなどの計算量の少ないアルゴリズムが有効に働くことも考えられる．

1.3 過去の経験からの表現語彙の獲得

過去の経験のデータには対象問題についての情報が含まれている．このようなデータを十分に用いてその情報を上手く表現できるような語彙を獲得できれば，表現語彙としては有用なものが得られると考えられる．このようなデータを用いた表現語彙の獲得にはデータからの特徴生成を利用できる．

この特徴生成は機械学習の分野において学習の前処理として研究されており，データから冗長な表現や無駄な表現を省き，データを上手く表現する特徴を生成する手法である．このような特徴は表現語彙として望ましいものである．この特徴生成手法は特徴の次元を減らすことで計算コスト・空間コストを減らしつつ，表現を対象問題について分かりやすいものにすることで学習の精度を上げる次元縮約手法 (dimensionality reduction) として多くの提案がなされている．

このような次元縮約手法では大規模なデータに対しては，比較的計算量の少ない不要な特徴のみを削除する (必要な特徴の部分集合を見つける) 特徴選択が用いられる．特徴選択では既存の特徴から選択をするのみであり，得られる知識は既存の特徴に依存したものとなる．このためこの特徴選択は新たな特徴を獲得する手法であるとはいえない上に既存の特徴に依存するためその表現の自由度は下がる．また，既存の特徴に対象問題を上手く説明できる特徴が含まれていない限り，大幅な精度の向上は難しい．またデータマイニングにおいて提案されている頻度を用いた関係ルールに基づく相関のある特徴の組合せの抽出も用いられる．しかし，このような関係ルールに基づく組合せは，特徴間の依存を対象としてその特徴の関係のみに注目して抽出しているため，必ずしも対象問題に対して特徴を抽出しているわけではない．

このような手法と異なり，扱える次元・データ数は特徴選択ほど大きくはないため，大規模なデータに対しては必ずしも適用できないものの，既存の特徴を組合せて新たな特徴を作成するような手法も提案されている．この特徴を組み合わせる手法としては特徴抽出や特徴構築が提案されている．この手法は複数の特徴を組合せて対象問題に関する情報を上手く縮約して表現で

きる特徴を見つける．このため対象問題に関する情報をすべて含んだような特徴集合が利用されており，十分な量のデータがあれば，対象問題を上手く表現する新たな特徴を発見することができるであろう．そのためこのような特徴生成手法を大規模なデータに対して適応できればこれまでの少ない次元・データ数で発見した特徴よりも対象とする問題に適した特徴の獲得が可能となると考えられる．

1.4 本論文の提案

本論文では，対象問題に適した語彙の対象ゲームに依らない方法での取得を目的として，棋譜データ内の共起関係に着目した大規模な棋譜データからの対象問題に適した語彙の獲得手法を提案する．

本研究により対象問題に適した語彙の獲得ができれば，対象ゲーム・対象問題への語彙の不足，語彙の取得方法の欠如，大規模データからの学習に関する問題に対処できる．これは

- 一般的な方法で語彙を取得できる
- 一定の定まった基準で問題に適した語彙を選択できる
- 問題に適した縮約された語彙のみで対象問題をあらわすことで大規模データを縮約して表現することができる

ためである．また，この提案によりこれまで解決できなかったコンピュータゲームプレイヤの一般的な手法での作成という課題における大きな問題の 1 つである，対象ゲームによらない一般的な語彙の獲得についてもその全ての問題ではないものの解決される．

語彙を表現する要素もないような無からの語彙の獲得は不可能である．そのため本研究では，その前提として，データからの表現語彙の獲得のためにはデータ自体も何らかの語彙で表現されており，棋譜データをすべて区別して表現できるようなゲームの基本語彙については既に与えられているものとする．このような基本語彙はゲームルールにおいて抽出できるものであり，また同時に対象とするゲームを作成する際に必要なものであり，対象ゲームに関する深い知識がなければ得られないようなものではない．本研究ではこのような既存の語彙で表現された棋譜データを用いて共起関係を基にした語彙の組合せによる語彙の拡張を行う．棋譜データに現れる共起としては指し手列 (もしくはそれに伴う局面列) に代表される時系列における共起と同局面内における出現に代表される空間的な共起が主な共起関係として挙げられる．この 2 つの共起関係についてそれぞれ新たな語彙を獲得する手法について提案を行う．1 つは指し手列に注目することで語彙を時系列的に組合せて新たな語彙を獲得するという手法について，指し手列の解析に基づく将棋におけ

る指し手の分類精密化において提案する．もう１つは同時に現れる語彙から新たな語彙を発見するという手法であり，コンピュータゲームプレイヤにおける評価要素の自動生成において提案する．

1.5 本論文の貢献

本論文の貢献は次のようなものがある．

データからの語彙の獲得 既存の語彙を組合せることで得た新たな語彙を利用することで，既存の語彙のみを利用した結果よりも有用な結果を得られることを示した．これより既存の語彙を組合せることによる新たな語彙の獲得の有効性を示した．

語彙の拡張による語彙作成の負担の軽減 既存の語彙を組合わせて一般的な方法で共起関係を元に語彙を抽出する方法を提案した．これにより手筋のようなこれまで人手で共起関係から選択・抽出していた語彙を作る負担を軽減することができ，また一定の指標を用いて選択することで偏りのない選択をすることができる．

ゲームに関する知識を必要としない語彙の獲得 本提案手法はゲームに関する既存の語彙は必要とされるものの，特にその局面内の共起においてはルールなどから容易に取得できるものである．このように本論文では対象とするゲームによらずに語彙を獲得できる一般的な手法を示した．

大規模データの利用による語彙の詳細化 これまでの多くの機械学習アルゴリズムでは扱うことが難しかった大規模データを有効に利用することで既存の語彙をより詳細な語彙に分けることが可能であることを示した．

大規模データを対象とした特徴生成手法の提案 これまで特徴生成手法では適用が困難であったような問題に適用できる特徴生成手法を提案した．同時に大量データとその特徴を用いた容易な機械学習アルゴリズムにより既存の特徴を用いた複雑な機械学習アルゴリズムよりも精度のよい学習ができることを示し，その有効性を示した．

コンピュータゲームプレイヤの一般的な方法での作成への貢献 コンピュータゲームプレイヤにおいて対象とするゲームに関する語彙を一般的な手法で取得する方法を提案し，有用な語彙の獲得が可能となった．これは一般的な機械学習による評価関数の重みづけ手法と組合せることで評価関数をゲームによらない方法で作成できるということである．そして一般的な探索手法と組合せることで，コンピュータゲームプレイヤの一般的な方法での作成が可能だと考えられる．このように一般的な強いプレイヤの作成にはまだまだ多くの課題が残されてはいるものの，本

提案により一般的なコンピュータゲームプレイヤ作成というコンピュータゲームプレイヤの大きな課題について１つの可能性を示した．

1.6 本論文の構成

本論文では大規模棋譜データからの対象問題に適した語彙の獲得手法について提案する．本論文の流れとしてはまずその関連技術として，コンピュータゲームプレイヤにおける知識獲得と機械学習における特徴生成手法について述べる．その後，本論文で提案するコンピュータゲームプレイヤにおける共起関係に基づいた語彙の獲得手法についてその詳細と評価を行った結果について述べる．そして最後に本論文をまとめる．

本論文の具体的な構成は次のようになる．

第２章では本研究で対象としたコンピュータゲームプレイヤと評価関数作成において機械学習を用いた手法について紹介する．ここではコンピュータゲームプレイヤにおける研究を基にして作成されている現在の一般的なコンピュータゲームプレイヤの中で用いられている探索技術，知識の利用方法について俯瞰する．そしてその研究の中で提案されてきた知識獲得について詳細に説明する．

第３章では機械学習における対象問題を上手く記述するための特徴を得る手法であり，既存の語彙獲得手法である特徴生成について述べる．本論文では特徴生成を特徴抽出，特徴選択，特徴構築に大別し，それぞれについて提案されている多くの手法のうち代表的なものを，その前提やモデル，コストなどについて説明する．

第４章で語彙を時系列的に組合せて新たな語彙を獲得するという手法である指し手列の解析に基づく将棋における指し手の分類精密化について提案し，評価を行った結果について述べる．これにより語彙を時系列的に組合せて新たな語彙を獲得する手法が有効であることを示す．

第５章で同時に現れる語彙から新たな語彙を発見するコンピュータゲームプレイヤにおける評価要素の自動生成について提案し，行った評価について示す．この研究により局面に同時に現れる語彙を組合せることの重要性について示す．

最後に第６章で本研究をまとめる．既存の語彙から語彙を生成することの重要性について述べ，本研究のゲーム研究における位置づけとさらなる展開について述べる．

第2章　コンピュータゲームプレイヤにおける知識獲得

本章ではコンピュータゲームプレイヤにおける知識獲得について述べる．コンピュータゲームプレイヤは主にゲーム木探索アルゴリズムと対象ゲームの知識を基にした評価関数から構成される．さらに，強いコンピュータゲームプレイヤを作るためには対象ゲームの知識を利用した様々な方法での効率化が必要とされる．

ゲーム木探索アルゴリズムはゲームの先の局面を展開することで先の変化した局面での判断をすること可能にする手法である．このような探索を行うのは先を読むことで得られる利点があるためである．この利点は次のように説明できる．多くのゲームにおいてはゲームの局面は指し手によって変化する．このような変化はルールには従っているものの一般的な関数形で簡単に表現することが難しいような変化である．このような変化を考慮して局面を評価するのは難しく，変化した先の局面を判断の対象とすることでより容易により正しい判断ができるようになると考えられる．

ゲーム木探索において変化した先の局面の良しあしを決めるのが評価関数である．この評価関数はその局面 (とそれまでの履歴) だけを利用してその局面の有利さの度合を出力するものであり，その有利さの度合は局面同士の順位づけを行うのに用いられる．評価関数として正確なものを実現するためにはゲームの知識が必要とされる．

このゲームの知識は評価関数だけではなく，定石やゲーム木探索の効率化にも用いられる．定石はゲームによく現れる局面において，単純な解析では見つけられないけれども深く探索すれば見つけられる，もしくは過去の蓄積から明らかとなっている指し手の知識を，あらかじめ保存しておき利用することでコンピュータゲームプレイヤを改善する．またゲーム木探索において探索を効率的にするためには，良い手を早く見つけることで探索する必要がないとわかる手や明らかに悪い手を枝刈りする必要があり，そのためにもゲームの知識は用いられる．

このように多くの場面においてコンピュータゲームプレイヤを作成する際には対象ゲームの知識が必要とされる．このような対象ゲームの知識としては人間が経験から獲得したものが多く利用されている．それと共にこれまで行われてきた過去のゲームや探索の結果などから対象ゲームの知識を獲得するような対象ゲームからの知識獲得に関する研究もなされている．このよう

な研究としてはそのゲームに対する語彙が与えられた状態での知識獲得が多くなされている.

本章では, まずコンピュータゲームプレイヤについて紹介したのち, コンピュータゲームプレイヤにおける対象ゲームからの知識獲得手法について説明する.

2.1 コンピュータゲームプレイヤの概要

コンピュータゲームプレイヤはゲームの局面を入力として受け取り, 次の手を出力するプログラムである. ここで扱うゲームは次のような条件を満たした2プレイヤ零和有限確定完全情報ゲームと呼ばれる.

2プレイヤ 2プレイヤとは敵と味方の2グループに分かれていることである. 麻雀などは4プレイヤとなる. 1グループには何人いても構わない.

零和 零和とはプレイヤ両者の利得の和が零になり勝ちと負けもしくは引き分けで終わることである. ギャンブルなど他から利得の和に変動を受けるような, 両者勝ち, 両者負けなどが起こりうるものは零話ではない.

有限 有限とはいつかはゲームが終了することである.

確定 確定的なゲームとは偶然の要素がないゲームのことである. さいころを振るようなゲームは確定ではない.

完全情報 完全情報とはゲームの局面を見れば現在の状況がすべて分かり隠れている情報がないことである. たとえば隠れた手札や場に伏せたカードがあるようなトランプゲームは完全情報ではない.

このような条件を満たしたゲームはオセロ・チェス・将棋・囲碁[1] など数多く存在する.

このようなゲームにおいてコンピュータゲームプレイヤはゲーム中に現れた局面から次の可能な指し手を繰り返し展開し, 展開した先の局面を評価することで次の手を決定する. この次の指し手を展開する手法をゲーム木探索, 次の手の中から最も良い手を選ぶための基準となる現在の局面の良さを表わす値 (評価値) を得る関数を (静的) 評価関数という. 例として図 2.1 に○×ゲームにおける探索深さ2でのゲーム木とその探索を示した. 本節ではこのゲーム木探索, 評価関数についてそれぞれ 2.1.1 項, 2.1.2 項で説明する.

[1] 日本ルールを用いた囲碁は実際には有限ではないが有限のものとして扱っている. (中国ルールを用いた囲碁は有限なゲームである.)

2.1.1 ゲーム木探索

コンピュータゲームプレイヤはゲーム木探索を行うことで次の手を決定する．ゲーム木とは図 2.1 に示したような現在の局面をルートノード・次の指し手を枝・その指し手によって新たにつくられる局面をその枝につながったノードとすることで，ルートノードから再帰的に展開することで出来るグラフのことである．このゲーム木を用いて最善の手を選ぶゲーム木探索アルゴリズムには Minimax 探索・共謀数探索 [72]・UCT アルゴリズム [58] など様々なものがある．囲碁においては UCT (Upper Confidence bounds applied to Trees) を用いたプレイヤ，それ以外のゲームでは Minimax 探索に枝刈りを追加した Alpha-beta 探索を用いたプレイヤが主流である．本論文では多くのゲームについて長く研究されている Minimax 探索を基にした探索を中心に説明を行う．UCT 探索においてはランダムシミュレーションの結果によって評価値を更新しながら用いる，つまりランダムシミュレーションを評価関数とすることが一般的になっており，ここで説明する静的評価関数が評価関数として用いられることはない．しかし評価関数を評価値の初期値として用いる [42] など知識の利用はなされており，知識の利用は必要とされる．

先の局面を読む探索アルゴリズムは 2.1.2 項に示す評価関数により次の手の局面同士の比較が正確にできるのであれば必要ない．しかし，実際にはそのような正確な比較は難しく，先を読むことで単純な関数形では表現が難しい将来の局面の情報を用いた評価が可能となると共に，安定したもしくはわかりやすい局面での評価が可能となる．また Minimax 探索を始めとしたゲーム木探索では評価関数を絶対の基準としてその評価基準に基づいて最善な手を選ぶ．このため評価関数の基準が Minimax 探索の結果を決定づけ，強いプレイヤを作るには評価関数の基準をうまく設定する必要がある．

2.1.1.1 Minimax 探索

Minimax 探索とは 2 プレイヤ両者が最善の手を選ぶとしたときに最も自分に有利な状態になる次の手を選ぶ手法である．実際に最善の手を評価するには展開した木が終局でなければ正確な判定はできず，それは多くのゲームにおいては不可能なので評価関数において最善の手を選ぶ．図 2.1 に示したのが Minimax 探索を行った後のゲーム木の状態である．Minimax 探索ではまず現在の局面からある一定の深さまで手を展開する．次にその展開した葉ノード (leaf node) について評価関数を用いて評価を行う．葉ノードの親ノードが自分の手番であれば自分に有利な手つまり評価値の高い手を，相手の手番であれば相手に有利な手つまり評価値の低い手を選び，その評価値をその親ノードの評価値とする．この評価値の親への伝搬を繰り返し，ルートノードまで評価した際，ルートノードと同じ評価値を持ったルートノードの子ノードへの手が最善手となる．このルートノードと同じ評価値を持った葉ノードまで

の最善手のノード群を PV ノード (Principal Variation nodes) , 特にその葉ノードを PV リーフという.

2.1.1.2 探索の効率化

Minimax 探索を基にしたゲーム木探索ではその探索するノード数が深さに応じて指数的に増大するためその探索ノード数の増大をいかに抑えるかが問題となる. ここではその探索ノード数の増大を防ぐ手法について紹介する. この探索コストを減らすために特によく用いられている手法として, 探索する必要のないノードを発見し探索しないようにする枝刈り・探索する必要のないノードを発見するために良いノードを早めに発見する指し手の順位づけ・一度探索したノードと同じノードを探索しないようにする置換表・通常の探索よりもより高速な方法で探索することで早めに結果を見つける終盤探索・事前の知識を蓄えて探索をせずに結果を見つけるデータベースの利用について紹介する.

Alpha-beta 探索 (後向き枝刈り手法)

探索ノード数の増大を抑える最も基本的な手法としてよく用いられるのが, Minimax 探索と同じ探索木を効率的に探索する Alpha-beta 探索である. Alpha-beta 探索では探索中のノードの結果を利用した枝刈りにより効率的に探索を行う. Alpha-beta 探索では深さ優先で探索を行い, 評価値の上限・下限を表わす探索窓をそれまで探索中に評価したノードの値によって更新・伝搬する. この探索窓から外れたノードは探索する必要がないため, このようなノードを探索せずに効率的に正確な探索をすることができる.

枝刈りの例として図 2.2 に示した Minimax 探索における探索木について説明する. 図 2.2 について図の左から深さ優先で探索を行い, R まで探索が終了したとする. N の値は O と S の小さい方となるので 30 以下であることがわかる. この時 B の値も決定しているので, A の値は (40, $+\infty$) の範囲の探索窓の中にあることはわかっており, S 以下の値は A に影響を与えないことがわかり, S 以下のノードは探索しなくてもよいことがわかる. Alpha-beta 探索はこのような枝刈りを行うことで効率的な探索を行う.

Alpha-beta 探索では最善手列 (PV ノード) を優先して探索することができればその探索結果を利用して探索を効率化できる. 特に Alpha-beta 探索で最善手列を最初に探索することができれば理論的に最小の木の展開のみで探索が終わることがわかっており Minimax 探索と同等の結果を得られる手法としては最善の手法となる. この際同じ時間で Minimax 探索のほぼ倍の深さまで探索できる.

図 2.3 に示したのが図 2.2 に示した Minimax 探索における探索木に対応する Alpha-Beta 探索でもっとも効率よく探索された場合の探索木 (critical tree) である. この必ず探索しなければならない critical tree はコンピュータ

ゲームプレイヤにおいて重要なものである．ここではよく用いられる Knuth と Moore が考察した critical tree のノードについての 3 つの分類について紹介する．ノード内に書かれた番号 1, 2, 3 はそれぞれ次に説明する type1, type2, type3 ノードをあらわす．

type1 ルートノードは type1 である．type1 の子ノードのうち最初に探索されるノードは type1，その他の子ノードは type2 である．type1 の子ノードはすべて探索される．このノードは PV ノードである．

type2 type2 の子ノードは type3 である．type2 の子ノードは 1 つだけ探索すればよい．

type3 type3 の子ノードは type2 である．type3 の子ノードはすべて探索される．

この Minimax 探索と同等の結果を最小の探索木で得られる Alpha-beta 探索は理論的には最良の探索手法であるが，実際には最善手列を最初に探索することは困難なため Alpha-beta 探索より経験的に効率のよい手法が提案されている．このような手法には探索窓の幅を 1 した null window search を利用した PVS (Principal Variation Search) や MTD(f) (Memory-enhanced Test Driver) [81] などの手法がある．このような Alpha-beta 探索などの Minimax 探索と探索結果が変わらない[2]枝刈り手法は後向き枝刈りと呼ばれる．

指し手の順位付け (move ordering)

Alpha-beta 探索では最善手をどれだけ早く探索するかが重要であり，そこで用いられるのが指し手の順位付け (move ordering) である．この順位付けの精度は探索の効率を左右する重要な要素となる．このための手法には指し手の評価関数の利用のほかに，ヒストリーヒューリスティクス [86] に代表される反復深化法 (iterative deepening) による浅い探索の最善手の情報やキラー応手 (killer heuristics) と呼ばれる兄弟ノードでの最善手の利用などが挙げられる．

置換表 (transposition table)

ゲーム木は実際には探索木中に同一局面が出現するためグラフとなる．この同一局面における同一深さ[3]の探索は同じ結果となるため無駄になる．この無駄な探索を抑えるため置換表 (transposition table) と呼ばれる局面のハッシュと探索結果を保存する表を用いる．

前向き枝刈り

[2]局面表の利用などによって変わることもある．
[3]前の探索より深い探索の結果はより正確であると考えられるためそれを利用することもある

多くの探索ノード数を減らすために，あるヒューリスティクスに基づいてそのノードの深い探索での評価値を予測したり，そのノード以下の探索の必要性の有無を評価することで，そのノード以下を枝刈りできるかどうかの判断をし，枝刈りを行う手法も多く存在する．ここで紹介する枝刈りは Alpha-beta 探索の枝刈りとは違って Minimax 探索と探索結果が変わる可能性があり前向き枝刈りと呼ばれる．

この前向き枝刈りには元の探索結果から変わらないように安全な枝刈りを行うような手法と経験的な知識を多く使い，安全でないような枝刈りをする手法が提案されている．

安全な枝刈りを行う手法としては次のようなものがあげられる．浅い手の探索結果を用いてその手以下の評価値の上限を予測して枝刈りを行う ProbCut [14]，葉ノードに近いところで評価関数を呼び出すことで評価値の上限を予測して枝刈りを行う Futility pruning [48]，一手パスして浅い探索を行った結果を評価値の下限とする null-move pruning [91] などが挙げられる．

安全でないような枝刈りを行う手法では同じ深さでは探索結果も変わってしまうが，深くまで読むことができるようにするためで精度の高い探索を行うことができるようになる．このような手法には将棋において探索の深いところで駒損する手を読まない・深い手ほど読まないなどの詳細な知識を組み合わせた枝刈りや定石を用いて数手分をスキップして探索する手法や実現確率打ち切り探索 [99]，などが挙げられる．

終盤探索

チェッカー [87] や LOA (Lines of Action) [105]，将棋，囲碁などにおいてはその勝ち，負けのみを高速に探索する手法も多く利用されている (利用箇所は終盤のみとは限らない)．このような探索手法で特に将棋において広く用いられているのが詰将棋を対象として多くの研究がなされ，有用な提案がなされている AND/OR 木探索手法である．このような AND/OR 探索手法で有用なものは証明数探索と呼ばれるものである．証明数探索では，その勝ちを証明するために証明する必要があるノード数である証明数とその負けを証明するために証明する必要があるノード数である反証数を用いて，その証明数・反証数の少ない方から探索を行うことで高速に勝ち・負けを証明することを目的とした探索を行う．このような探索には PDS (Proof-number Disproof-number Search), df-pn (depth-first proof-number search) [77] などが提案されている．

データベースの利用

人間が長年の経験を培って得てきた次の手を読むことなしに選択できるような知識やコンピュータゲームプレイヤが長い時間をかけて得た最善手などはコンピュータゲームプレイヤが探索中に発見することが難しい．そのためこのような知識はコンピュータゲームプレイヤが保存しておいて利用するこ

とが望ましい．しかしその一方でそのような知識を上手く切り分けてすべて保存するということは難しい．

このため保存した局面が現れやすく，その利用価値の高い初盤・終盤においてはこのような探索中に発見が難しい知識を保存したデータベースがよく用いられる．

初盤データベースは多くのゲームにおいてゲーム初期配置は決まっているため，Samuel のチェッカー [85] で用いられて以来よく用いられる．初期局面が決まっていれば初期局面からの指し手列とその重要度を比較的少ない量で保存しておくことが可能である．また初期局面は 1 ゲーム中に必ず起こる局面であるためそのデータベースにおける知識はその一部を必ず使うことができ，保存しておくメリットが大きい．また，評価関数の作成においてゲームの進行していない初盤の評価は難しいことが多く，そのような局面を人間の知識や深い探索の知識で代替することでそのような信頼度の低い評価関数を利用せずに済むようになる．さらに，探索では常に同じ手を返すため，常に同じプレイをするプレイヤとなってしまうため，違った指し手を打たせる必要がある．このような問題に対応するために初盤データベースが用いられる．

またチェスやチェッカー [87]，中国象棋 [30] などのような終盤に近づくにつれ駒が少なくなるようなゲームにおいては駒数が少ないときの結果は時間をかければ先に探索し，結果を得ることができる．このようなデータについては初盤データベースほどは使われにくく，その保存しておく量も多くなるものの，その結果は勝ち・負け・引き分けといった勝負を決定づける正確な情報でありそのデータを見つければ勝負が決まるため保存しておく価値がある．

その他の改良

実装における改良は局面の差分更新や盤面更新や指し手生成処理の効率化を目的とした bitboard を用いたなどがある．

また今回は主な対象としておらず，探索ノード数を削減するための手法ではないものの，複数のプロセッサを使って探索を行う並列探索に関する研究も多くなされている．この研究はゲーム木探索の研究の大きな 1 つのテーマとして提案され，YBWC (Young Brothers Wait Concept) [33]，APHID (Asynchronous Parallel Hierarchical Iterative Deepening) [13]，TDSAB (Transposition-table Driven Scheduling Alpha-Beta) [57] などが提案されている．

2.1.2 評価関数

コンピュータゲームプレイヤにおいて評価関数とは一般には静的評価関数のことを示す[4]．本論文においても評価関数はこの静的評価関数の意味で用い

[4]評価関数は広義では「局面を『評価』する『関数』」である．この意味ではゲーム木探索アルゴリズムしいてはコンピュータゲームプレイヤも 1 つの評価関数であると見做すことも可能であるが，このような意味で用いられることは少ない．

る．静的とは先読みをせず，現在の局面とそれまでに至った指し手という探索の必要のない既知の情報のみを使うということである．このようにする理由は評価する局面から展開されるであろう局面を利用して評価を行うと，探索アルゴリズムとの整合性が取りづらくなるためである．例えば，その展開の予想が容易な局面については深い局面に対する評価値を返し，そうでない局面には浅い局面に対する評価値を返すという評価関数は兄弟局面について同じ深さで評価しているにもかかわらずその評価値の信頼度がバラバラであって利用しづらいということになる．またその探索アルゴリズムの展開と評価関数での展開が一致しない場合には 2 つの指標による探索木の展開を行うことになる．

評価関数の主な用途はゲーム木探索における指し手の順位付け (move ordering) や葉ノードの評価である．そのために評価関数はある指標に基づき入力局面の良し悪しを判断し，この判断結果として評価値を出力する．この評価値は一般にはスカラー値で表し，評価値を用いることで局面を一意に順位づけすることが可能となる．評価値はコンピュータゲームプレイヤの振舞いや強さを決める直接的な要因となるため，精度が高いことが要求されるが，その一方でその評価に時間がかかるとゲーム木探索で探索できる木が小さくなってしまう．このように評価関数においてはスピードと精度がトレードオフの関係にあり，その利用目的に応じて最適な均衡点を探る必要がある．

評価関数では局面を評価するために局面をコンピュータが評価できるような表現に変える必要がある．この表現として評価関数では局面の情報をあらわす評価要素を用いる．この評価要素には様々なものがあり，たとえば駒の数や局面の一部の形などのようなものが用いられる．評価関数は重み付け和などによってこの評価要素を基に作成される．

評価関数の例として図 2.1 に示した○×ゲームの評価関数 E を紹介する．

$$E(p) = (\text{手番が線を引ける数}) - (\text{相手番が線を引ける数}) \qquad (2.1)$$

ここで p は局面である．この評価関数では「線を引ける数」が評価要素であり，手番によって+1, -1 の重みがつけられた線形和として表現されている．この関数は局面を入力としてその手番の優位さを評価しているため評価関数であるということができる．○×ゲームは単純なゲームであり，このように比較的単純な評価関数を用いてもよい精度で探索することができるが，他の複雑なゲームではそれが難しく，評価要素の選択，評価関数の構成など多くの工夫が必要とされる．この評価関数の現在の作成方法については 2.2.1 において説明する．

2.1.2.1 評価関数の信頼度の向上

評価関数はその絶対的な精度を向上させることはもちろん有用であるが，それ以外にもその評価関数が評価しやすいような局面を評価できるように調

整することで，評価関数の信頼度を向上させることが可能となる．このような評価関数が評価しやすい局面を実現する方法として，ここでは探索中にその深さの調整によって評価しやすい局面まで展開する手法と評価関数をその使う場面に応じて複数作成する手法についてそれぞれ説明する．

探索による調整

探索中に評価関数による評価値の信頼度を向上させるための手法は評価値が正確になることで探索の精度を向上させることのできる有用な手法であるので多く利用されている．

これを行う理由としては静的な評価が難しいような局面に対して急な探索打ち切りによって評価関数を利用してしまい間違った評価値を得てしまう水平線効果 (holizon effect) という問題があるためである．この問題の例としては将棋において角交換中の手などでは交換の先手が角を 2 枚持っているような状況が存在しその先手のほうが有利に評価されてしまうことや回避が容易な王手をされている場合にその王手によって不利に評価されてしまうことが挙げられる．また手番によっても最後に手を打ったほうが若干有利に見えてしまう odd-even effect も存在する．

これらを回避するための手法としては必然手・強制手や取り合いの手を進めることで静かな局面を用いて評価する静止探索 (quiescence search) や Deep Blue が用いた同じ兄弟ノード間において評価値の際立って高い手を深く探索する singular extensions [4] に代表される selective extensions が挙げられる．

複数の評価関数の利用

利用する場面によって評価関数を切り替えることも多くなされている．この中で最も多く利用されているものとしては進行度があげられる．進行度とはゲームの進行具合を表現したもので，簡単なものでは手数があげられる．この進行度を用いることで初盤，終盤などでの評価関数の切り替えや変化が可能となり，ゲーム全体の局面を評価できる 1 つの評価関数を作成するという困難な問題をよりわかりやすい問題に変えることができる．

2.2 ゲームにおける知識獲得

ゲームにおいて知識は評価関数において利用され，その探索結果を決定づける手法である．また 2.1.1.2 節に示した通り定石やゲーム木探索の効率化など多くの場面において有用である．

このためこの知識を獲得する手法は機械学習を利用した手法を中心に多く研究されてきた．このような知識を獲得する際，ゲームでは次のことに注意を払う必要がある．

知識の利用対象 学習結果による間違った知識を探索で補正するような利用方法であればその精度が致命的な問題となることは少ないが，その学習結果が探索の結果に直に反映するような利用をした場合にはその精度が探索の精度を下げるような致命的なものとなり，それにより深く探索できるよりはそれを使わずに浅く正確に探索したほうがよいということもある．このように得られた知識の利用方法に注意を払う必要がある．

精度と速度 実際にその学習結果は探索中に用いられるため，その学習結果を利用するために必要な計算コストが問題となる．一般にモデルを複雑にし上手く学習できれば精度は向上するが，その計算コストによってはその結果を利用せずに代替となる探索を行ったほうが良いこともある．このように精度のみを上げればよいというわけではなく，精度と速度のトレードオフ点を探る必要がある．

また，長い時間をかけた探索や自己対戦，他のコンピュータゲームプレイヤとの対戦などは計算機上で人間への負担をかけずに実行可能である．このようにゲームでは計算によって計算機のみで学習対象の作成が可能である．これは機械学習が用いられる他の対象とゲームが異なる点である．このため他の学習対象に比べ，容易に学習データを大量に取得することができる．学習においては大量データを用いることが必ずしも良いとは限らないが，これまで多くなされてきたゲーム研究において少ないデータ数で良い結果が得られた研究は少なく，ゲームを特徴づける語彙はそれほど少ないとも考えにくい．

このようなことからゲームにおける知識獲得では

- 学習結果を高速に利用できるモデルを仮定し，
- 大量データを対象にした学習手法を用いた

ものが多い．

本節ではこの知識獲得手法について，その知識の利用目的ごとに，評価関数の獲得，データベースの構築，探索の効率化における知識獲得について説明する．

2.2.1 評価関数の獲得手法

一般的には評価関数は次のように作成される．

1. 局面の評価に必要な評価要素を選ぶ
2. 評価要素を基に評価関数を作成する

局面の評価に必要な評価要素の選択は将棋や囲碁など比較的難しいゲームにおいては人手で行われる．この評価要素はゲームのあらゆる状況を明確に区別でき，重み付けを行いやすい形で表現できることが理想である．しかし，

実際にこのような評価要素を見つけることは難しく，対象とするゲームごとに深い人間の知識を基に有用と考えられる評価要素を選択する．このような選択においては見落としや知識の不足により，評価関数が局面の正しい順位づけを行うことができないことも多く発生する．そのため人間がある知識の不足に気づいた場合はアドホックに評価要素を追加することで評価関数の精度を上げるということが頻繁に行われることとなり，精度の高い評価要素の選択は人間への大きな負担となっているのが現状である．

評価関数の作成については評価要素の組み合わせ方を決め，その重みを調整する．線形和の評価関数については人手での重み調整を行うことが頻繁に行われているが，一方で線形和・ニューラルネットワークなどを用いて評価要素を組み合わせ，統計的な手法や機械学習を用いて学習し，重み調整を行う手法も多く提案されており非常に良い成果を出している．

また，ゲームの知識をあまり必要としない単純な評価要素を使い，それから自動的に評価要素を作成する研究も行われている．

2.2.1.1 評価関数の重み調整

評価関数における評価要素が与えられているという前提で，その評価関数の重みを自動的に調整する方法としては機械学習を用いた手法は非常に多く提案されている．

このような手法には主に TD 学習や進化的な手法などの強化学習，教師あり学習をもとにした手法が挙げられる．強化学習では自己対戦や対局サーバ上のプレイヤとの対戦を通して評価関数の重みを調整する．また教師あり学習では棋譜や人間の知識による局面の評価が与えられた状況でその棋譜の指し手や結果，局面の評価と一致するように評価関数の重みを調整する．ほとんどのゲームにおいて教師あり学習で学習したプレイヤは強化学習を用いて学習したプレイヤよりも強くなっている．しかし，その一方で教師ありの手法はその教師データの精度に影響されるものが多く，強いプレイヤのいないゲームや棋譜などの学習データの得にくいようなゲームにおいては強化学習の手法が用いられる．

またこの両方に属した方法を用いたプレイヤも存在する．Utgoff は強化学習における親子ノード学習と教師あり学習の指し手の一致における兄弟ノード学習の両方を組み合わせて良い結果を得る手法 [101] も提案しており，どちらの手法よりもよい結果が得られたと報告している．

本節では強化学習，教師あり学習それぞれについて説明する．

強化学習

強化学習では自己対戦や他とのプレイヤとの対戦を通じて評価関数の重みを調整する．強化学習は教師データが得られないような状況でも利用できるため，棋譜などの知識のたまっていないゲームに適用できる．

このような学習を行った初めてのプログラムは Samuel のチェッカー [85] であり，探索先のノードの評価値にルートノードの評価値が近づくように駒価値の調整を行っている．この学習は明確な報酬がなく TD 学習とは異なるものの，これ以降 20 年近く学習によるプレイヤの作成は行われておらず，革新的な手法である．

強化学習を用いた手法には終局結果からの報酬の伝搬により評価値を補正する TD 学習を用いたものと複数のプレイヤを作成しその中でよいものを進化的な方法で選別・発展させる進化的手法を用いたものが挙げられる．

TD 学習を用いた調整

TD (temporal difference) 学習を基にした手法を用いた重み調整は多く行われている．TD 学習とは未来の状態とその報酬から現在の状態の補正を行う手法である．具体的にゲームでは終局における勝ち・負けの報酬を終局に至るまでの局面に伝搬することで終局に至るまでの局面が終局の評価値に近づいて評価されるように補正を行う．この TD 学習を用いたプレイヤとしては TD-Gammon [94]，KnightCap [7], Herakles [97], rlgo [88] など多く提案されている．

TD-Gammon [94, 41] は Tesauro によって作られたバックギャモンのプレイヤであり，TD 学習を用いて作られた最初の強いプレイヤである．このプレイヤの研究を発端として TD 学習を用いたゲームプレイヤの研究は盛んに行われてきた．TD-Gammon では TD(λ) という手法を用いて学習を行っている．TD(λ) とは未来の状態からの報酬の伝搬を $\lambda(0 \leq \lambda \leq 1)$ 倍に減衰させて伝搬する手法である．TD-Gammon 3.0 で用いられている評価関数は 300 個近くの入力要素，160 個の中間層を，7 個の出力層を持った 3 層パーセプトロンを約 150 万の自己対戦を行って調整することにより得られたものであり，人間のチャンピオンに匹敵するかそれ以上の強さのプレイヤを得ることができていると報告されている．

KnightCap [7] は Baxter らによって作られたチェスのプレイヤであり，駒価値の線形和の調整を TDLeaf(λ) という学習手法で行っている．TDLeaf(λ) とはゲーム木の評価値を決定する PV リーフ (PV ノードの葉ノード) を現在の状態として TD(λ) を行う手法である．TDLeaf(λ) は探索を行った結果を用いて学習を行うため TD(λ) に比べてそのコストが高く，多くの試合をこなして学習することは難しいが，多くのゲームをこなした TD(λ) よりも少ないゲーム数で学習した TDLeaf(λ) のほうが良い結果が得られている．Baxter らはインターネットチェスサーバ上で 3 日間のプレイで 308 ゲームを行い学習を行ったところ，レーティングが 1650 から 2150 まで向上し，人間のマスターレベルにまで達したと報告している．

Herakles [98] は Tournavitis によって作られたオセロのプレイヤであり複数の思考エンジンを持ったプレイヤである．そのうちの Charly という思考エンジンは MOUSE(μ) (MOnte-carlo learning Using heuriStic Error reduc-

tion) [97] という手法を用いてオセロの評価関数を学習している．MOUSE(μ) とは TD(λ) 学習にある期間学習期間を設けてその間よく出てくるような信頼できる変更を重視して更新を行うように拡張した手法である．Charly では 150 万近くのパラメータを持った線形の評価関数を MOUSE(μ) を用いて調整することで Generic Game Server [17] において最強のコンピュータオセロプレイヤより 100 程度レーティングが低いもののレーティング 2,600 以上の強さを得ることができている．

rlgo [88] は Silver らによって作られた 9 路盤の囲碁のプレイヤである．rlgo では絶対位置，相対位置それぞれに対し，図 2.4 に示したような 1×1 マス，1×2 マス，...，5×5 マスのそれぞれの形にに現れる石の組み合わせを (一部ハッシュ化したものを) 盤面の特徴として，それぞれの特徴の重みを TD(0) ($\lambda = 0$ の TD(λ)) を用いて学習している．150 万近くの特徴を用いて違った振る舞いをする 50 近くのプレイヤのいる Computer Go Server[5] で学習を行ったところ-170 のランダムプレイヤから+1860 の最も強いプレイヤまでいる中で+1210 程度の Elo rating が得られている．この結果は知識ベースのプレイヤが+700〜+850 程度であることからよい結果であると Silver らは報告している．

TD 学習では自己対戦によりプレイヤを強化できるため棋譜の無いゲームに適用可能であり良い成果を挙げているものも多い．しかし，その一方で局所最適に陥りやすく，棋譜を使ったプレイヤほど強いプレイヤを作成できていないのが現状である．KnightCap においても自己対戦よりサーバ上の多種類のプレイヤとの対戦を行ったほうが良い結果となったことが報告されている [7].

進化的手法を用いた調整

TD 学習における局所最適の問題の影響を受けにくい手法として進化的手法を用いたプレイヤも少ないながら提案されている．このようなプレイヤとしては進化的アルゴリズムを用いた Fogel による○×ゲーム・Checker (Blondie24) [35]・チェス (Blondie25) [36]，Chellapilla らによる Checker [22] などがある．また群知能アルゴリズムの 1 つである Particle Swarm Optimization (PSO) を用いた Messerschmidt の○×ゲーム [73] や Franken のチェッカー [37] などもある．

このような手法では○×ゲームやチェッカーなどの簡単なゲームを対象としていることからもわかるとおり，その計算コストが大きくなることが問題となり大きなゲームには適用しづらい．このような中で Fogel らはチェスという比較的大きな対象ゲームについて複数のプレイヤを用いて重みを調整することでレーティング 2,650 程度の強いプレイヤを得られており，多プレイヤを用いることにより良い結果を得ることができている．これより難しいと

[5] http://cgos.boardspace.net/9x9.html

されているゲームに対して同様の結果が得られるかは今のところ不明であるが，この結果からも過去の棋譜を利用しない (もしくは利用できないゲームにおいての) 評価関数の作成方法としては現在もっとも有用な選択肢の 1 つであると考えられる．

教師あり学習を用いた調整

教師あり学習を用いたプレイヤは棋譜を基に機械学習を行うことで重み調整を行っている．このような学習では

指し手の順序関係を教師とした学習 Preference learning [40] と呼ばれる手法もしくはそれに似た目的の手法で，指し手の順序関係を教師とした学習は指し手の順序が評価関数での評価と一致するように学習を行う．

棋譜の終局結果を対象とした学習 終局結果を対象とした学習では対戦結果を予測できるように評価関数を学習させる．その多くは結果との一致を目指し最小 2 乗法やロジスティック回帰を行う．

の 2 つが主になされてきた．

このような学習においてはプロなどの強いプレイヤの棋譜があることが望ましい．そしてそのような強いプレイヤの棋譜がある場合には以降に示す Logistello や Bonanza などのように非常に良い結果を残している．指し手との一致を目指した手法は，評価関数の学習という目的においては間接的な訓練信号しか得られないものの初盤・中盤などにおいても教師例が得られるため終局結果のみを用いた学習よりも良い成果を挙げている．

指し手の順序関係を教師とした学習

指し手の順序関係の学習ではその順序関係との一致を目指し学習する．

指し手の順序関係を教師とした学習は古いものでは 1985 年に Marsland [70] が考察を行っており，推薦された手を良い手とするのは効果があるものの全ての手よりも良くするようなことをするのは逆効果であると結論付けている．

Tesauro [95] は実際の局面で指した手によって変化した局面とほかの候補手によってできる局面を用いた兄弟ノード間の比較学習を行うことで王の安全度をプロの棋譜のデータベースから学習し，その安全度の重みを上昇させている．Tesauro らは当時世界ランキング・トップのチェスプレイヤーであった Gary Kasparov との 1997 年の 2 度目の対戦のときにこの値を用い，この値が Kasparov に勝ち越す要因になったと報告している．

保木の作成したコンピュータ将棋プレイヤ Bonanza [51] では棋譜の指し手と評価関数の順位が一致するように最適制御理論を用いて線形の評価関数における重みを調整している．Bonanza ではプロの棋譜 30,000 局と将棋クラブ 24[6] の棋譜 30,000 局を用い，2 手先からの静止探索を行った後の PV リーフ

[6] http://www.shogidojo.com/

を指し手の先の局面とし，それらの棋譜の指し手と評価値の順位が一致するように約3カ月かけて学習を行っている．結果としては2006年の世界コンピュータ将棋選手権で優勝しており，有用な評価関数が作成できているといえる．

Gomboc ら [43] はあらかじめ人間の評価を基にラベルづけされた棋譜との評価の一致を順位相関係数を用いて評価し，その一致度を高めるように学習する手法を提案している．Gomboc らはこの手法をチェスプログラム Crafty に適用し，多くは敗退したもののパラメータによっては元のプレイヤと同等以上の強さを得ることもできており，有望な結果が得られたと報告している．しかし，実際に他のゲームに適用することは強い人間のプレイヤの評価が必要であることから多くの場合については困難である．

終局結果を対象とした学習

終局結果を対象とした学習では現在の局面の評価関数の評価値が終局の結果に近づくように学習を行う．

最強のコンピュータオセロプレイヤの1つである Logistello [16] ではそのゲームの進行度に合わせて評価関数を13段階に分け，オセロの終局の最終石差との一致を目的として2.2.1.2節において得られた評価要素を最小2乗法で線形評価関数の重み調整を行っている．同様の学習は Herakles [98] における Caesar というエンジンにおいてもなされており，約67万ゲームを学習局面として Generic Game Server [17] においてレーティング2,700以上の最強のコンピュータオセロプレイヤと同程度のプレイヤを作成することができていると報告している．

竹内ら [92] は局面の評価値とその局面からの勝率の一致を目指した評価関数の学習と評価値と局面の評価値の関係を可視化することでの特徴の過不足の発見の手助けとすることを提案している．竹内らはこの手法をオセロプレイヤ zebra，チェスプレイヤ Crafty，将棋プレイヤ GPS Shogi に適用し，最小2乗法やロジスティック回帰を用いて勝率との一致を目的として補正をかけたところ，全てのプレイヤについて以前のプレイヤと同等かそれ以上の強さのプレイヤを作成できている．またこのような可視化のヒントを用いて手調整したプレイヤは他のプレイヤよりも強くなっており，可視化の利点が得られているとともに自動調整の難しさがわかる結果になっている．

2.2.1.2 評価要素の自動生成

評価要素を自動的に生成する方法は1990年代から少ないながら行われている．これらの手法では単純な評価要素を組み合わせた評価要素の良し悪しを棋譜を基にして評価することで評価要素を作成する．このような手法には ELF [102], GLE M [16], ZENITH [31] やそれを改良した金子らの方法 [56] などがある．

ELF [102] は Utgoff らの提案したチェッカーに適用された 2 値の単純な特徴の積で表された評価要素を追加しつつ学習を行う手法である．ラベルの付けられた新たな訓練局面についてこれまで選んだ評価要素の重みを調整した後にその誤差との差分が最も小さくなるような (誤差を最も表現するような) 新たな評価要素を追加する，ということを繰り返すことにより評価要素を逐次生成する．追加した評価要素の重みは 0 から始め，一定期間重みが 0 からあまり変化しないような評価要素は削除される．

Buro の提案した GLEM [16] では 2 値の単純な特徴について，訓練局面セットに頻出する組合せをアプリオリ法 [2] に似た手法で選択しそのすべてを評価要素とする．Buro はこの手法をコンピュータオセロプレイヤ Logistello に適用しているが，角の 3×3 のパターンのようにパターンと呼ばれる組合せる特徴を限定する範囲を使って組合せ爆発を避けている．パターンの例を図 2.5 に示した．このパターンはオセロに特有のもので一般的なパターンの作成方法については提案されていない．この Logistello は現在は更新されていないものの 2000 年代前半において最も強いコンピュータオセロプレイヤである．またこの GLEM を用いたプレイヤである Herakles の Caesar というエンジンは 2.2.1.1 節の教師あり学習にも書いたとおり，約 67 万ゲームを学習局面として Generic Game Server [17] においてレーティング 2,700 以上の最強のコンピュータオセロプレイヤと同程度のプレイヤを作成することができていると報告している．

ZENITH [31] やそれを改良した金子らの方法 [56] ではゲームのルールと目的から次のように評価要素を作成する．まず，ホーン節で記述したゲームのルールと目的 [49] から新たなホーン節の生成を行い，そのホーン節から単純な特徴の論理積に変換することで評価要素候補を作成する．この候補から訓練局面について頻度がある範囲以内にあるものを選び，それから訓練局面のラベルを最もよくあらわす特徴を変数増加法を用いて選択し，評価要素を作成している．訓練局面が多い場合には訓練局面をいくつかのサブセットに分けそれぞれについて選択を行っている．結果としてはパターンを利用しないため Buro の手法よりもその速度は遅いものの，その最小 2 乗誤差において精度の高い評価関数が得られたことを金子らは報告している．

このほかにも評価要素の表現を決める F という言語を提案している Duminy の方法 [28] もあるが，まだその手法を適用したプレイヤについては公表されていない．

2.2.2 データベースの構築

局面とその指し手を保存した知識を蓄えたデータベースはその知識が実際に使われやすく，保存しておく価値があるような初盤・終盤において用いられる．

初盤データベース

初盤データベースを構築する手法として最も簡単なものはプロの棋譜や決まった定石を入力しておくことで初盤データベースを作成する方法である．またこのデータベースを自動的に獲得もしくは拡張する手法も多く提案されている．この方法の多くは事前探索や自己対戦や他との対戦の結果を利用して，その勝敗や利用された回数・探索の評価値などをデータベースに保存しておき利用するものである [18, 54, 15, 25]．

終盤データベース

事前計算が可能な終盤の結果については事前に計算しておき終盤データベースに追加する．このような終盤解析に関する研究は多くなされており，後退解析 (Retrograde analysis) [96] の手法が広く用いられている．Schaeffer らはチェッカー [87] を Chinook を使って解いており，その時には約 3.9×10^{13} 局面を保持した 10 ピースまでの終局データベースを利用したと報告している．

2.2.3 探索の効率化における知識獲得

探索の効率化に関する知識としてもっとも広く用いられているのが指し手の順位付けである．指し手の順位付けには評価関数そのものを用いるものもあるが評価関数は一般的には計算コストのかかるものであるため，それより単純なものを用いることが多い．また囲碁においてよく用いられている UCT 探索では終局までのランダムゲームの結果を利用して探索を行うことが一般的になされている．このような探索においても実際のゲームでめったに指されないような手を探索するよりも良い指し手を多く読んでランダムゲームの結果の精度を上げたほうが良い結果が得られることがわかっており，指し手の順位づけが重要になっている．

Minimax を基にした探索においては指し手の順位付けとしては人間の知識によるものが利用されている．例えばチェスにおいては「王手」や「駒をとる手」,「動きづらい駒を動かす手」などを，将棋においては「直前に動いた駒をとる手」や「直前の手で狙われた駒が逃げる手」などを他の手より先に探索すると探索の効率が上がることが知られている．このようなゲームにおいて指し手の順位付けでヒューリスティックな方法以上に成功を収めているものはほとんど提案されていない．

Kocsis らはチェスにおいてニューラルネットワークを用いた学習結果とヒストリーヒューリスティクス [86] を組み合わせることで指し手の順位づけの精度を向上させている [60, 59]．Kocsis らはこの手法を LOA (Lines of Action) プレイヤ MIA に適用し元のプレイヤに若干ではあるが勝ち越したことを報告している．また，チェスプレイヤ Crafty に適用することで，多くの指し手の順位づけ手法よりも良い結果が得られ，Crafty の 7 段読みと同程度の精度で指し手の順位づけができていると報告している．

また囲碁においても知識をパターンにしたものが用いられ，そのパターンを基に指し手の順位づけがなされている．このパターンとは「盤の隅に打つのは良くない」,「相手の2つの連が繋がるのを阻止するのは良い」,「自分の眼を埋め尽くすな」，あるいは「アタリにはノビよ」 の様なヒューリスティックなものである [82]．また，囲碁においてはこのパターンを自動的に抽出する，もしくは，そのパターンの順位付けを学習する手法が特に多く提案されており広く用いられている．パターンの自動抽出の手法は強いプレイヤの棋譜から頻出するパターンを抽出し評価する手法が多く用いられている．またパターンの順位づけを学習する手法として最も単純なものとしては出現したパターンについてその打たれた割合を計算する手法が提案されている．このような割合だけではパターン間の関係を見ていないため同時に出現するパターン間の優劣を基に学習する手法として打たれたパターンを他のパターンへの勝利としてパターンの Elo レーティングを計算する手法が提案されている [82]．

Rémi はこのパターンのレーティングを用いた探索をコンピュータ囲碁プレイヤ CrazyStone に適用し，9 路盤，19 路盤両方について多くのコンピュータ囲碁プレイヤ大会で優勝するなど有用な結果を挙げている．

また指し手の順位づけ以外にも探索延長・枝刈りにおいて知識獲得をする研究もなされている．このような研究としては Björnsson らが指し手から必要な探索ノード数を最急降下法により学習し，探索延長に利用する手法を提案している [11]．Björnsson らはコンピュータチェスプログラム Crafty の探索制御パラメータの調整を行っており，元のプログラムとの強さの違いはそれほど見られないものの 100 ゲームの対戦から学習をかけたプレイヤが最も良い結果を出していることを報告している．

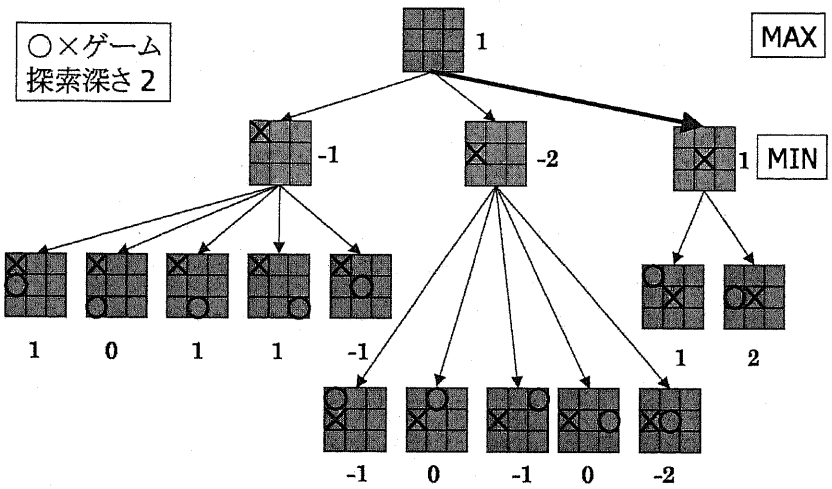

図 2.1: ○×ゲームにおけるゲーム木

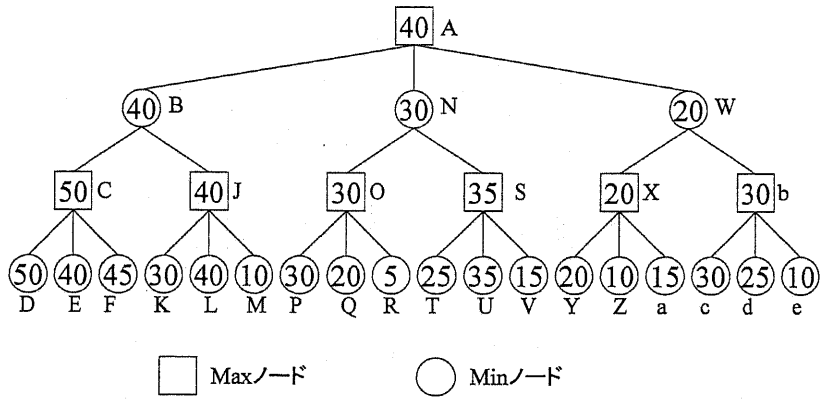

図 2.2: minimax tree

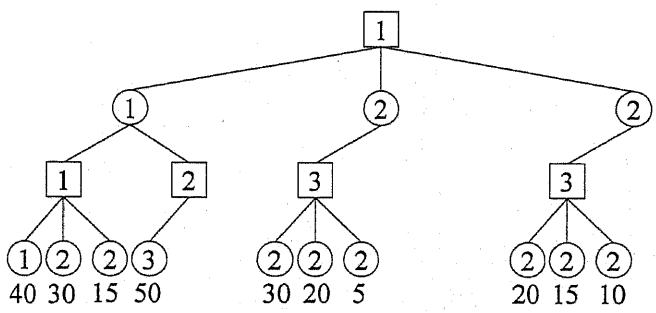

図 2.3: critical tree

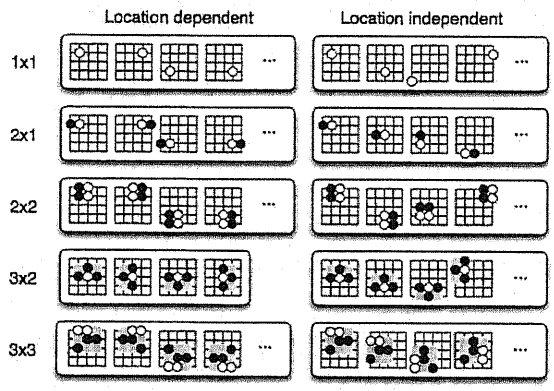

図 2.4: rlgo における特徴

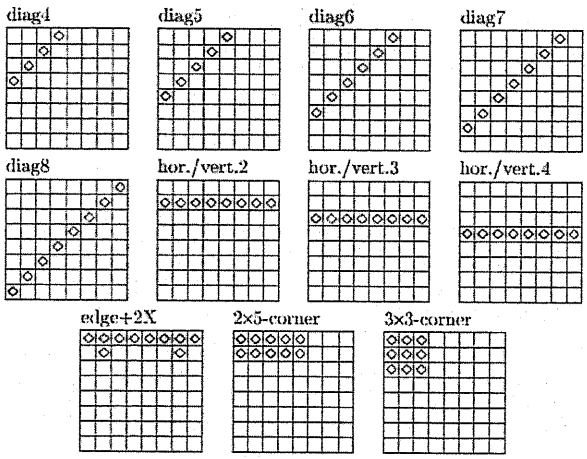

図 2.5: logistello におけるパターン

第3章　機械学習における特徴生成手法

この章では機械学習における特徴生成手法について紹介する．

機械学習とは与えられた入力データを元にそのデータもしくはそのデータに付加されたラベルを上手く説明するモデルを構築する手法である．この機械学習で得られたモデルを用いることで，新たなデータについての評価が可能となり，それに基づいて新たなデータに対する判断を行うことができる．機械学習はこの入力データの形式とラベル，モデルとその基準，判断結果 (出力) の形式によって様々なものが存在する．

このような機械学習を行う際に問題となるのが，入力として与えられるデータそのものの表現である．機械学習においてほとんどの場合入力データは学習手法に適した形で表現されていない．このようなデータの問題には大きく分けて次のような3つの問題がある．

データの異常 これはデータのラベル付けにおける矛盾やデータがない欠損値の存在によって学習が難しくなる問題である．この矛盾にはデータに関する特徴不足に起因する必然的な矛盾である場合，ラベルづけの間違いによる場合などがある．欠損値が起こる場合としては，値が決して得られない場合，回答拒否・不明の選択などにより欠損した場合，計測におけるデータ異常 (これが異常であるかどうかを判定することは困難であることもある) や計測値の限界を超えたことによる場合などがある．

データ形式の非統一 これはデータ形式 (連続値・離散値・記号データ) の非統一，正規化の必要などによる問題である．機械学習アルゴリズムが扱えない形式のデータが入力データに存在する場合はそのアルゴリズムを適用することができない．また距離などの指標を使って特徴同士を組み合わせるようなアルゴリズムではその絶対的な値が影響を及ぼすため，正規化を行っていないことで大きい値を持った特徴が重視されてしまう結果になるといった問題がある．

データを表現する特徴の問題 これは機械学習に適用できる形であっても表現しているその特徴 (属性，変量) によって学習が困難になり学習の精度が下がってしまう問題である．これは主にデータを冗長でなくかつ十分に表現できる特徴が得られないことによって起こる問題である．このよ

うなものには学習に必要な情報はデータには含まれているものの，その特徴をいくつか組み合わせなければ明らかにならないような情報になっているという場合がある．もちろん，特徴が足りずにそもそもデータに学習に必要な情報が十分に含まれていない場合もあり，そのような場合には可能であればデータを取り直すことが望ましい．ただ，このような場合においてもその含まれている情報すらもうまく抽出できないことがある．また，逆に情報が冗長に表現されている場合も，同じ情報の発見や最適な表現の選択など表現の自由度が増えることによる問題が起こる．

このようなデータの問題に対処するためにはデータを問題に適した形に変換する必要がある．このようなデータを整形する前処理はデータマイニングでも用いられており，データクレンジングと呼ばれる．本章で扱う問題はこのうち最後の特徴に起因する問題である．この他の問題についてもそれに対処するデータの変換手法やデータの形式に見合った機械学習手法の選択について多くの提案がなされているが本論文の対象外であるのでここでは紹介しない．この特徴に起因する問題を解決する方法として，このようなデータを表現している特徴を組み合わせ・選択することでデータを学習しやすい形にする特徴生成手法が用いられる．

この特徴生成手法は特徴の多いデータに対してその次元を下げることを目的として行われることが多く次元縮約手法もしくは次元削減手法とも呼ばれる．このような次元を削減する理由としては高次元での機械学習を行う際におこる次の 2 つの問題があるためである．

- 計算・空間コストが大きいこと
- 汎化誤差が向上しなくなること

まず，計算・空間コストが大きい問題に関しては次のようなことがあげられる．現在提案されている機械学習アルゴリズムの多くは計算量が大きく，次元やデータ量の指数のオーダのコストがかかる手法が多い．そのため高次元のデータを直接扱うことがその計算量の点から現在の環境では困難である．また全てのデータを一度に処理しなければならないようなアルゴリズムにおいては現在の環境ではそのデータをメモリ上に展開することすら困難であり，その実行自体が不可能である．

また高次元のデータを扱った学習にでは汎化誤差が向上しなくなるという問題は次元の呪いやヒューズの現象 [53] として知られている．これは多数の特徴を用いた学習においては

- 次元が増えるにつれデータ間の距離が全て離れてしまい，似たような位置に属してしまう球面集中現象 [115] などでよく説明されるようにその相関の高い情報が増えてしまう

- 次元に対して十分な訓練データが得られない

ことから学習が難しくなるためである．

また特徴を同等に扱うことに関しても問題があることが分かっている．これは醜いアヒルの子の定理 [115] において説明されている．醜いアヒルの子の定理とは，各特徴量を全て同等に扱うと，醜いアヒルの子を含む n 匹のアヒルがいた時に，その醜いアヒルの子と普通のアヒルの子は任意の 2 匹のアヒルと同じくらい似ているという結果になるというものである．この定理は特徴を同等に扱うことをせずに何らかの重要度を持って利用しなければならないということが説明されている．このことは全ての特徴を同等に扱うことの危険性を示唆しており，特徴生成手法によって多くの特徴から重要な特徴を選別する必要があることを示している．

このように高次元の問題，特徴を同等に扱うことの問題など特徴が多く十分な組み合わせ，選別がなされていない場合には多くの問題があり，特徴生成が必要である．この特徴生成手法は大きく次の 3 つに大別される．

特徴抽出 (属性構築, feature extraction) 特徴で表現される高次元の特徴空間を元の空間とは異なる低次元の特徴空間に射影することで次元削減を行う手法

特徴選択 (属性選択, feature selection) 特徴から重要な特徴を選ぶもしくは不必要な特徴を省くことで次元削減を行う手法

特徴構築 (Feature construction) 特徴を組合せて多くの特徴を作ったのちに特徴選択により重要な特徴を選択する手法

実際には特徴抽出と特徴選択の 2 つに分けて論じられることが多い．また特徴抽出の意味で特徴生成の用語が用いられることもある．さらに，特徴構築はそのハイブリッドな手法として，特徴抽出か特徴選択のいずれかにおいて議論されることが多い．そのような場合においては特徴構築は新しい特徴を作るという意味でその目的は特徴抽出に近い．本論文では次元削減を主な目的としない，次元が元の特徴よりも増加することのある手法として特徴構築を分けて紹介する．

図 3.1, 図 3.2 に特徴抽出，特徴構築と特徴選択についての簡単な例を挙げた．ここで示した例は体重と身長によって肥満かそうでないかという例をプロットしたものだと思ってもらいたい．このような場合，特徴抽出，特徴構築では図 3.1 のように 2 つの特徴を組み合わせて新しい軸を作成する．この組み合わせ方の違いが特徴抽出・特徴構築の違いである．一方で特徴選択では図 3.2 のようにどちらか片方の特徴を選択してそれを軸とする．このような単純な例でもわかるとおり，特徴抽出や特徴構築はできるだけその対象問題に関する情報を失わないような軸を構成するため，軸を選んでその他の軸の情報は捨ててしまう特徴選択に比べて精度が高い．一方で特徴選択ではその

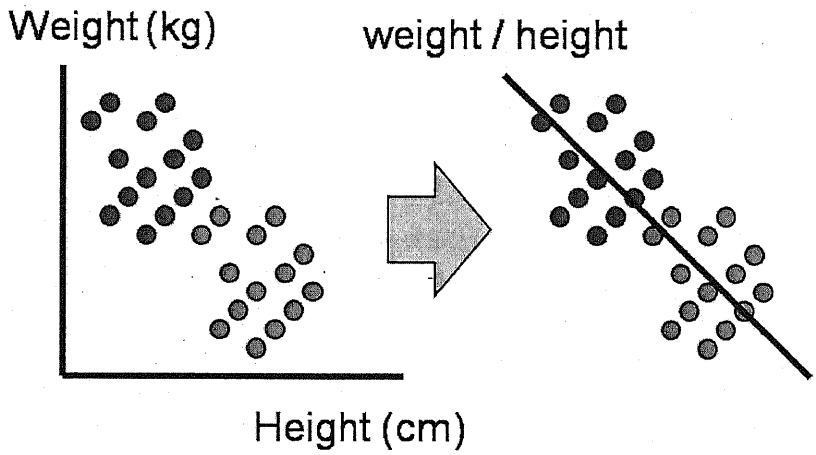

図 3.1: 特徴抽出，特徴構築の例

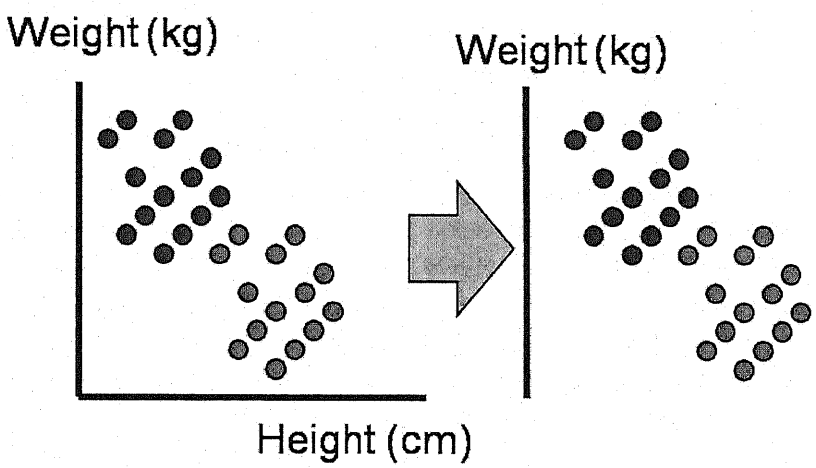

図 3.2: 特徴選択の例

体重と身長の2つを比較しどちらが重要か判断するだけでよいが，特徴抽出や特徴構築はその2つをデータに合うようにどのように変換するかを計算する必要がある．また，その特徴生成が終わった空間においても，特徴選択の後の空間は体重のみを評価すればよいが，特徴抽出や特徴構築では体重と身長両方を測って軸に合わせる計算を行う必要がある．このため特徴抽出，特徴構築の方が特徴選択よりもコストが高い．

このように一般に特徴選択，特徴構築，特徴抽出の順に精度が高いが，その分計算コストも大きい．ノーフリーランチ定理 [27] にも示されているとおり，どのような問題にも有効に働く手法というものは存在せず，これらの手法はそのコスト・精度・前提とするモデルによって，対象とする問題によってどの手法が有効であるかは異なる．対象とする問題領域によってはその問題領域に適したモデルなどの事前知識を用いてより良い手法が提案されているものもあるが，ここでは汎用に適用できる手法として提案されたもしくは利用されている手法について説明する．本章ではこの3つの手法それぞれについてその手法の例を示すとともに，それぞれの手法における利点，問題点について説明する．

3.1 特徴抽出

特徴抽出 (Feature Extraction) [20, 103] とは高次元の特徴空間にあるデータを元の空間とは異なる低次元の特徴空間に射影することで次元縮約をしようとする手法である．この低次元の空間はそのデータの (もしくはデータのラベルに関する) 情報をできるだけ失わずにデータを表現できるようなものが求められる．特徴抽出の目的は設定した基準においてこの空間の射影に必要なパラメータを求めることである．特徴抽出では，特徴空間を張る軸として特徴を，特徴空間上の点の集合としてデータを扱う．そして次元縮約するためにはこの点の集合を別の特徴空間に射影することを考える．この射影を行うパラメータを求めるには全てのデータを扱う必要があるため計算コスト・必要とするメモリ量ともに多くなる．このため大規模な問題についてこの手法を当てはめることは困難である．またデータを処理する際にもこの射影を行ってから処理する必要がありランニングコストも高くなる．このようにコストは高いものの全く別の特徴空間に射影するため，その指標・学習データについて最適なものを求められることが多い．このため得られた結果がそのままクラスタリングや分類器としても利用できる手法も多く存在する．

特徴抽出の手法には大きく

1. ラベルのついていないデータを扱う教師なしの手法
2. あらかじめ分類するためのラベルがついたデータを扱う教師ありの手法

がある．教師なしの手法ではラベルがないためにそのデータ自体を上手く表現できるものを選択する．一方で教師ありの手法ではデータそれ自体の持っている情報は重視せずラベルを上手く表現できるものを選択する．

また特徴抽出手法はその視点によって次のような手法がある．

1. 線形射影による手法
2. 非線形射影による手法

線形射影による手法では元の特徴空間と同じ空間において違った軸をとることでデータを変換する．非線形射影による手法では元の特徴空間とは違った空間に射影することでデータを変換する．

また，特徴抽出の空間コストを減らし巨大なデータ・逐次追加されるデータに対応するためにオンライン学習やインクリメンタル学習と呼ばれる追加学習の手法も提案されている．

3.1.1 教師なし特徴抽出

教師なしの特徴抽出は教師ありの手法では扱えないラベルのないデータやラベルを付けることが難しい問題に適用される．教師なしの手法ではラベルがないため表現力の高いもの，特徴的なものを選択することが目的となり，この表現力・特徴的といった基準の違いや問題に対する視点の違いが手法の違いとなる．

線形射影による特徴抽出

教師なし特徴抽出の手法として最もよく用いられるのが主成分分析 (PCA, Principal Component Analysis) [89] である．また最近，主成分分析では特徴抽出が難しい問題に対して適用できる独立成分分析 (ICA, Independent Component Analysis) [55] も利用されるようになってきた．これらの手法ではラベルがないためその低次元の特徴空間を表現するそれぞれの特徴を表現力の高いものにすることを目的として変換行列を求める．

主成分分析 [89] は射影後の特徴空間で表現したデータが互いに無相関になる変換行列を選ぶ手法である．この方法ではデータの分散が大きくなるようにデータは変換される．また変換された成分の分散の大きさの全体への割合を寄与率といい，この寄与率の和 (累積寄与率) が一定以上になるまで選択することでそのデータに関する情報損失を抑えながら次元削減をすることができる．

独立成分分析 [55] は得られた M 個の観測値から N の信号元を推定する blind source separation 問題を解く問題として提案された．このような blind source separation 問題の例としては多くの人の話している中から注目している人の声だけを聞き取ることのできるカクテルパーティ効果がよく挙げられ

る．この方法では，それぞれが独立と仮定した信号元を仮定して，$M \geq N$ として互いに独立となる成分を選択すること信号元の予測値とする．このために独立成分分析では射影後の特徴空間で表現したデータが互いに独立になる変換行列を選ぶ．この独立性の評価基準には様々な基準が用いられるが，結果として主成分分析では分散がうまく分解される正規直交した軸がとられるのに対し，独立成分分析では必ずしもそのようなことはなく多くは非正規性を最大化するようにデータは変換される．

この教師なし特徴抽出手法には主成分分析，独立成分分析の他にも因子分析 (factor analysis)，射影追跡法 (projection pursuit) [38]，多次元尺度構成法 (MDS, Multi-Dimensional Scaling) [24]，局所性保存射影法 (LPP, Locality preserving projections) [47] など数多く提案されている．

因子分析 (Factor Analysis) は雑音のある状況で観測要素がある要素 (共通因子) の合成量であると仮定し，個々の共通因子を得る手法である．この方法は，因子数を推定する基準，因子の見やすさ (単純性) を求める基準などの設定に多くの基準が提案されており，実際に利用するには因子分析に関する十分な経験や対象問題に対する十分な知識が必要とされる．

射影追跡法 [38] はデータの一番興味深い ("interesting" な) 射影方向を見つけることで次元縮約する手法である．データの興味のある方向としては通常はその正規分布から最も掛け離れた方向が選択される．これは完全にランダムなものは正規分布になるためである．この手法における非正規性の尺度と独立成分分析で用いられる独立性の評価基準には近いものが多い．

多次元尺度構成法 [24] はいくつかの対象間の（非）類似度を基にその類似度と最もよく一致するように多次元空間に対象を配置する手法である．この方法では対象間の関係を保存することが重視される．

局所性保存射影法 [47] は空間全体の関係を保存することを目的とせずに，類似度行列を基に元の空間で局所的なものがより局所的に配置されるような射影を求める手法である．

また特異値分解 (SVD, Singular Value Decomposition) を用いた PCA と同等の手法が自然言語処理では LSA (Latent Semantic Analysis) [64] と呼ばれるように分野ごとの特徴抽出手法も存在するが，ここでは紹介しない．

非線形射影による特徴抽出

これまでに挙げた特徴抽出手法は線形射影する手法であるが，それだけでは表現が難しいデータも存在する．このような問題に対しての特徴抽出手法として非線形空間に射影して次元縮約する手法が近年多く提案されている．このような手法には自己組織化マップ (SOM, Self Organizing Map) [63], GTM (Generative Topographic Mapping) [10], Deep belief networks [50], kernel PCA [9], Isomap (Isometric feature mapping) [93], LLE (Locally Linear Embedding) [84], Laplacian eigenmaps [8], Hessian eigenmaps [26], などがあげられる．

自己組織化マップ [63] は競合学習及び近傍学習により入力特徴をある特徴マップに射影する階層型ニューラルネットワークを構成する手法である．GTM [10] は自己組織化マップの代替として提案された手法であり，EM (Expectation Maximization) アルゴリズムを用いてパラメータ調整された制約付きガウス混合分布を用いて射影を行う手法である．

Deep belief networks [50] とは多層のニューラルネットワークを構成する手法である．

kernel PCA [9] はカーネル法を用いて特徴空間を高次元の特徴空間に射影し，その高次元の空間でPCAを行う手法である．

Isomap [93] はサンプルデータと近傍のデータを接続したグラフを基にそのグラフ上の距離に基づいて多次元尺度構成法を行うことで，局所的な構成を保存しつつ射影を行う手法である．

LLE [84] は元のデータ空間における距離関係が射影後においても保たれるような射影を見つける手法である．Laplacian eigenmaps [8], Hessian eigenmaps [26] も LLE と同様，元のデータの近傍の関係を保ちつつ射影を行う手法である．

インクリメンタルな特徴抽出

これまで特徴抽出手法は特にその射影行列を目的とした手法についてはその計算量の問題から大規模なデータを一度に扱うことは難しい．そのため追加学習を行う手法も近年提案されている．このような手法にはPCAを追加学習可能にしたIPCA (incremental PCA) [44, 45, 5] や CCIPCA (Candid Covariance-Free Incremental Principal Component Analysis) [104], kernel PCA を追加学習可能にした incremental kernel PCA [23] などがある．

3.1.2 教師あり特徴抽出

教師あり特徴抽出の手法ではあらかじめ分類するためのラベルが与えられているためそのラベル付けされたデータがそれぞれの基準において最も上手く分類できる特徴空間に射影できるような変換行列を求める．

教師ありの手法はそのラベルをあらかじめ与えられていることから十分なデータがあればそのラベルによって与えられた基準において精度の高い特徴空間に射影することができる．一方データ数が少ない場合にはこの訓練データの影響を受けることにより教師なしの手法よりも悪い結果となることもある [71].

線形射影による特徴抽出

教師あり特徴抽出の手法としては線形判別分析 (LDA, Linear Discriminant Analysis, 以降，判別分析) [108] がよく用いられる．また判別分析よりも効率的な手法として近年 MMC (Maximum Margin Criterion) [65] や OC

(Orthogonal Centroid algorithm) [79, 52] が提案されている．これらのアルゴリズムの違いはラベルづけされた上手く分類ができるようになると評価するための目的関数の違いである．

判別分析では各クラス間の距離をクラス内分散で割ったものが最大となるように目的関数を設定する．この基準が最大になるにはクラス同士が離れ，同クラスが近づくときとなり，ラベルに関して分類しやすい空間となると考えられる．ただし判別分析では特異値分解や一般化特異値分解，逆行列演算などが必要であり計算コストが高い．

MMC では各クラス間の距離からクラス内分散を引いたものが最大となるように目的関数を設定する．これも判別分析と同様クラス同士が離れ，同クラスが近づくとき最大となり，判別分析とほぼ同じ基準で特徴抽出が可能となる．判別分析との違いは差分を使っていることで逆行列演算が不要となり，判別分析に比べて高速に特徴抽出が可能となることである．

OC では各クラス間の距離が最大となるように目的関数を設定する．この基準はクラス同士が離れていれば最大となるためクラス自身が密になるようなものは求めていない．このため基準が若干緩いものとなるが，OC では判別分析に必要な特異値分解や逆行列演算を必要とせず，高速な QR 分解を用いた解法が可能であるため判別分析よりも高速に特徴抽出が可能となる．

このように線形射影による特徴抽出ではそのラベルづけされたクラスが上手く分類できるように空間に射影することを目的とする．

非線形射影による特徴抽出

教師なし特徴抽出ほど多くはないが教師あり特徴抽出についても非線形空間に射影して選択する手法が提案されている．これには判別分析をカーネル法で拡張した Kernel Fisher Discriminant Analysis (KFD) [75] や Generalized Discriminant Analysis (GDA) [6] などがあり，よい成果を示している．

インクリメンタルな特徴抽出

教師あり特徴抽出についても追加学習を行う手法についても近年提案されており，判別分析を元にした IDR/QR (Incremental Dimension Reduction algorithm via QR decomposition) [109] や ILDA (Incremental Linear Discriminant Analysis) [78], MMC を元にした IMMC (Incremental Maximum Margin Criterion) [108], OC を元にした IOC (Incremental Orthogonal Centroid) [107] などが提案されている．

3.2 特徴選択

特徴選択 (Feature Selection) [1] とは与えられた特徴の中から重要な特徴を選択する手法である．特徴選択では特徴の一部に着目してその善し悪しを

決めて選択を行うものがほとんどである．このため 3.1 節の特徴抽出のようにすべてのデータを用いた計算を行う必要がない．またデータを処理する際にも元の特徴で表現されたデータをそのまま扱うことができる．このため計算コスト・必要とするメモリ量ともに (特にフィルタメソッドでは) 特徴抽出に比べて非常に少なく大規模な問題も比較的容易に扱うことができる．しかしその一方で特徴選択では特徴から選ぶだけなので選ばれた特徴は元の特徴に依存し，選択した特徴においてもそれぞれ独立ではなく互いに相関のある特徴が存在する．また選択の順番によって結果が異なる手法もあり特徴抽出に比べて精度が低い．

この特徴選択手法には大きく分けて次の 2 つの手法が存在する [61].

フィルタメソッド 特徴の評価に機械学習の代わりに適当な代替基準を用いる手法

ラッパーメソッド 選ばれる候補となる特徴を用いた機械学習を行い，その結果をもとに特徴を選択する手法

フィルタメソッドでは直接機械学習を行う必要がないため，一般に低コストかつ高速に特徴選択を行うことができる．ラッパーメソッドでは機械学習を行うためコストが高いものの，その学習手法に適した特徴を選ぶことができるため精度の高い選択が可能となることが多い．

ここでは，選択基準として代表的なカイ 2 乗統計量, 情報量についての用語を説明したのち，それぞれのメソッドについて紹介する．

3.2.1 用語説明

カイ 2 乗統計量

カイ 2 乗統計量とは特徴とラベルとの間に相関があるか否かを調べる統計的な尺度である．特徴 t について訓練データにおけるカイ 2 乗統計量 $\chi^2(t)$ は次のように計算される．クラス c_j について k_{11} を「t と c_j が同時に現れる数」, k_{10} を「c_j のみ現れる数」, k_{01} を「t のみ現れる数」, k_{00} を「t と c_j のどちらも現れない数」とすると特徴 t, クラス c_j についてのカイ 2 乗統計量は次のように表される．

$$\begin{aligned}\chi^2(t,c_j) &= n(k_{11}k_{00}-k_{10}k_{01})^2 \\ &\times (k_{11}+k_{10})^{-1}\times(k_{01}+k_{00})^{-1} \\ &\times (k_{11}+k_{01})^{-1}\times(k_{10}+k_{00})^{-1} \qquad (3.1)\end{aligned}$$

これより特徴 t についてのカイ 2 乗統計量 $\chi^2(t)$ は

$$\chi^2(t)=\sum_{j=1}^{c} P_r(c_j)\chi^2(t,c_j) \qquad (3.2)$$

と求められる．ここで $P_r(c_j)$ は訓練データにおける c_j の事前確率である．カイ 2 乗統計量は特徴 t について観測値から推定した場合の理論値と実際の値のずれを表す．カイ 2 乗統計量が小さいほど独立性が高くカイ 2 乗統計量が大きいほど独立性が低い．ここで選別したい特徴はラベルを推定できる特徴であるとすると，ラベルとの独立性が低い特徴つまりカイ 2 乗統計量が大きい特徴を選択すればよい．

情報量

ある確率変数 X が与えられた時，X のエントロピー $\hat{H}(X)$ は次のように定義される．

$$\hat{H}(X) = -\sum_{x \in X} p_x \log p_x \tag{3.3}$$

このエントロピー $\hat{H}(X)$ は確率変数 X の乱雑さを示す．

また，確率変数 Y が観測されたのちの確率変数 X のエントロピー $\hat{H}(X|Y)$ は次のように定義される．

$$\hat{H}(X|Y) = -\sum_{y \in Y} p_y \sum_{x \in X} p_{x|y} \log p_{x|y} \tag{3.4}$$

この $\hat{H}(X|Y)$ は Y を観測したあとにのこる確率変数 X の乱雑さをあらわす．

この Y を観測したことによるエントロピーの減少量を情報利得 (IG, Information Gain) と呼び，次のように定義される．

$$\begin{aligned} IG &= \hat{H}(X) - \hat{H}(X|Y) \\ &= \hat{H}(Y) - \hat{H}(Y|X) \\ &= \hat{H}(X) + \hat{H}(Y) - \hat{H}(X,Y) \\ &= \hat{I}(Y;X) = \hat{I}(X;Y) \end{aligned} \tag{3.5}$$

この情報利得は相互情報量とも呼ばれる量である．相互情報量 $\hat{I}(Y;X)$ は X が持つ Y に関する情報量ということができる．ここで選別したい特徴はラベルを推定できる特徴であるとすると，ラベルと特徴の相互情報量が大きい特徴を選択すればよいことになる．ここで $\hat{H}(X,Y)$ は結合エントロピーと呼ばれ，次のように計算できる．

$$\hat{H}(X,Y) = -\sum_{x \in X} \sum_{y \in Y} p_{x,y} \log p_{x,y} \tag{3.6}$$

また条件付き相互情報量 $\hat{I}(Y;X|Z)$ は次のように定義できる．

$$\begin{aligned} \hat{I}(Y;X|Z) &= \hat{H}(Y|X) - \hat{H}(Y|X,Z) \\ &= \hat{H}(Y,Z) - \hat{H}(Z) \\ &\quad -\hat{H}(Y,X,Z) + \hat{H}(X,Z) \end{aligned} \tag{3.7}$$

この条件付き相互情報量とは Z を知った上での X の持つ Y に関する情報量ということができる．

3.2.2 フィルタメソッド

特徴選択におけるフィルタメソッドとは特徴の重要性を評価する基準を設定し，それに基づいて選択を行う手法である．フィルタメソッドではラベル付けされたデータをもとにある評価基準に基づいて特徴の重要度を決め，探索により選択を行う．

評価基準の指標としては，距離尺度，情報尺度，依存尺度，整合性尺度が用いられる．

距離尺度 2つのクラスに含まれる訓練例の距離ができるだけ遠くに，同じクラスに含まれる訓練例の距離ができるだけ近くになるものを良いとする指標である．

情報尺度 情報量基準に基づきクラスに関する情報利得が多いものを良いとする指標である．

依存尺度 クラスとの独立性が低いもの，相関の大きいものを良いとする指標である．

整合性尺度 全ての属性値が同じであればクラスも同じになるようにする指標である．

また探索手法としては完全性を保証する探索を行うもの，ヒューリスティクス (逐次選択) による探索を行うもの，ランダムに探索を行うするものがある．

完全性を保証する探索 この探索手法は評価基準において最適な結果を得る方法である．特徴セットの部分集合は特徴セットよりよくなることはないという単調性をもった指標においてはビーム探索法や分枝限定法などを用いることで全探索をせずとも完全性を保証できる．ただし，単調性をもった指標は少なく，それを使った探索の精度もそれほどよくはないというのが現状である．

ヒューリスティクスによる探索 この探索手法は完全性をあきらめ逐次選択する手法である．このため得られる解が必ずしも評価基準に最適になるとは限らないが，完全性を保証する方法に比べ計算コストが低いことが多い．この手法には特徴を1つずつ追加しながら評価を行う forward stepwise selection, 全体の集合から1つずつ削除しながら評価を行う backward stepwise elimination, 両方向から選択を行う bi-directional hill-climing methods などがある．

ランダム探索 この探索手法は完全性を保証する探索コストを減らしつつ，局所最適に陥る問題を緩和するために用いられる方法である．このランダム探索は整合性尺度を基にした手法のみ提案されている．遺伝的ア

ルゴリズムや焼きなまし法のようにヒューリスティクスによる探索に乱雑さを追加する手法とラスベガスアルゴリズムを用いた選択のように完全にランダムに行う手法がある．

フィルタメソッドには非常に多くの手法が提案されている [1]．ここでは一般に同時に扱う特徴の数が大きいものほど計算コストのかかる手法となることから扱う特徴数に着目し，特徴を個別に評価する手法，2 つの特徴 (の相関) を用いて評価する手法，それ以上の多くの特徴を同時に評価する手法それぞれについてその代表的な手法を紹介する．

特徴を個別に評価する手法

特徴を個別に評価する手法としてはそれぞれの尺度に基づいた個別の特徴の評価をもとに逐次選択を行う手法がある．これには整合性尺度以外の尺度が用いられ，情報尺度である情報利得の最大となる特徴を選ぶ手法や依存尺度であるカイ 2 乗統計量の最大となる特徴を選ぶ手法 [29]，距離尺度であるクラス間距離の最大となる特徴を選ぶ OCFS (Orthogonal Centroid Feature Selection) [106] などがある．

このような特徴それぞれを個別に評価する手法としてよく用いられている手法として Relief がある．Relief は距離尺度を用いてヒューリスティクスによる探索を行う手法である．Relief ではデータセットのサンプリングを行い，その選択されたサンプルを基に特徴の重要度を更新する．選択されたサンプルと最も近い同じクラスのサンプルをニアヒット，最も近い異なるクラスのサンプルをニアミスとして，それぞれの特徴についてその重要度からニアヒットとの距離を引いてニアミスとの距離を足すことで重要度を更新する．ニアミスとの距離が大きいほど，ニアヒットとの距離が小さいほど重要度は大きくなるため，クラスとの関連性の高い属性ということができる．

このような特徴を個別に評価する手法は非常に高速であるためよく用いられるが，特徴間の比較をすることがないため重複した情報を持った特徴を取り除くことができないという大きな問題がある．

2 つの特徴 (の相関) を用いて評価する手法

2 つの特徴を用いて評価する手法には，2 つの特徴とラベルから得られる情報尺度とヒューリスティクスによる探索を用いる手法がある．このような手法は特徴同士の比較を行うことが可能であることから相関のある特徴を取り除くことが可能である．これには FCBF (Fast Correlation-Based Filter) [110], CMIM (Conditional Mutual Information Maximization) [34], mRMR (minimum Redundancy Maximum Relevance) [80], DISR (Double Input Symmetrical Relevance) [74] などがある．

FCBF は前処理としてラベルとの相互情報量 (情報利得) の低い特徴を削除したのち，ラベルよりも別の特徴と関係が深い特徴を削除する．より具体

的には，ある 2 つの特徴 X1, X2 について X2 の方がラベルに関する相互情報量が大きく，X1 がラベルとの相互情報量よりも X2 との相互情報量が大きい場合に X1 を削除する．これにより FCBF は特徴を選択する．

CMIM は条件付き相互情報量最大化の名前のとおり，これまで選んだどの特徴に対してもそれを選んだ上でのラベルとの相互情報量 (条件付き相互情報量) が大きくなるように選択する．具体的にはこれまで選んだ特徴がある上でのラベルとの条件付き相互情報量の最小値が最大となるものを選択する．

mRMR は他の特徴との関係が少ない冗長でない特徴であり，ラベルに関係の深い特徴を優先して選ぶことを目的とした手法である．このために mRMR ではラベルとの相互情報量が大きくなるように他の特徴との相互情報量の平均が小さくなるように選択する．この 2 つの軸を同時に扱うために mRMR ではラベルとの相互情報量から他の特徴との相互情報量の平均を引いたものが最大になるものを選択する．

DISR は相互情報量 $\hat{I}(Y;X)$ を結合エントロピー $\hat{H}(X,Y)$ で割ったものを symmetric relevance として定義し，これまで選んだ特徴とある特徴を同時に選んだ時のラベルとの symmetric relevance について全てのこれまで選んだ特徴についての和が最大となるものを選択する．

このように 2 つの特徴を利用する手法では，その 2 つの特徴を比較することでいずれの手法についても冗長な (相関の高い) 特徴を除去することを目指しながら，ラベルをできるだけ正確に説明できる特徴を選択する．

多くの特徴を同時に評価する手法

多くの特徴を同時に評価する手法はこれまで選んだ特徴と現在の特徴もしくは特徴全体を評価することで複数の特徴によって表現される冗長性を取り除くことが可能となる．多くの特徴を同時に評価する手法では整合性尺度を用いた手法として代表的なものに Focus がある．Focus は完全性を保証する探索を行う手法であり，空の特徴集合からサイズを増やしていき，そのサイズのすべての部分集合について整合性のチェックを行いながら，その整合性が得られるまで探索を行う．また，他の指標と比べてコストは高いものの精度の高い結果を出している手法としてカーネル法を利用した高次元での依存尺度 HSIC (Hilbert-Schmidt Independence Criterion) に基づき backward stepwise elimination を用いて特徴選択する手法 [90] が提案されている．この手法はカーネルによっては前に選んだ特徴によらず特徴を個別に選択する手法となる．

このような多くの特徴を同時に利用する手法はコストが高いがその分精度の高い選択が可能となる．

3.2.3 ラッパーメソッド

ラッパーメソッドでは機械学習の結果を用いて特徴を選択する．このラッパーメソッドはフィルタメソッドにおける 3.2.2 節で紹介した評価基準の代わりに機械学習の学習結果の精度を用いる手法である．ラッパーメソッドはフィルタメソッドに比べ精度の高い特徴選択が可能であるが，計算時間が膨大になるため大規模なデータに用いるは一般に困難である．また過適合 [67] についての報告もなされており，それを上手く避けつつ精度をあげるのは難しい．ラッパーメソッドではその計算時間のため実行が非現実的になることが多いため，フィルタメソッドに比べて研究は少ない．

ラッパーメソッドでは 3.2.2 節で紹介した forward stepwise selection, backward stepwise elimination がよく用いられる [62]．このうち backward stepwise elimination はまずすべての特徴について機械学習をしなければならないため計算量の観点から難しく，少ない特徴で機械学習を行えばよい forward stepwise selection のほうがよく用いられる．

また 3.2.2 節で紹介したランダム探索を基にした手法も用いられる．このような手法の中で代表的なものに LVW (Las Vegas algorithm for Wrapper feature selection) [66] がある．LVW はランダムに選択した特徴をその度，機械学習で精度を評価し，得られた最もよい (精度が高く，特徴の数が少ない) 特徴集合を出力する手法である．

3.3 特徴構築

特徴構築 (Feature Construction) とは特徴の一部を用いてそれらを組合せることで新たな特徴を作成し，その中から特徴選択を行うことで与えられた特徴よりも対象問題に適した形で表現する手法である．このように特徴選択を用いることから特徴構築は特徴選択の派生として説明されることも多い．

特徴構築の手法には進化的プログラミングを用いるもの [111] や Kernel PCA など非線形特徴抽出を行った後に特徴選択を行うもの [19]，探索を行うもの [69] などの手法がある．このような手法で汎用的な手法を提案しているものは少ないが，特定の分野に対象を絞ってその知識を用いて効率的に特徴構築を行う手法は多く提案されている．

ここでは特徴の組み合わせと選択により特徴構築を行う汎用的な手法として Markovitch らによって提案されている FICUS (Feature Incremental ConstrUction System) [69] について説明する．FICUS では FSS (Feature Space Specification) という問題を記述する言語を提供し，それによって記述された問題領域について特徴構築を行う．FSS では基本的な特徴，特徴構築を行うための関数とその領域・範囲・制約について記述を行うことができる．特徴構築は特徴抽出・概念学習・特徴選択の繰り返しからなる．特徴抽出では FSS において記述された特徴とその関数を用いてビーム探索法を用い高次

の特徴を生成する．概念学習ではこの特徴を用いて学習 (決定木) を行うことで学習器を生成する．最後に特徴選択では学習結果を元に重要な特徴の選択を行う．このようにして特徴構築することで Markovitch らは UCI Machine Learning Repository[1] の多くの問題について元の特徴を用いた学習よりも高い精度での分類が可能となったと報告している．この結果は特徴構築の有用性を示す上で重要な結果であるといえるが，FSS を用いた記述は容易ではなく対象問題への知識が必要である．また評価に用いられたデータについても○×ゲームを対象としたもの (特徴が 100 程度のもの) が最大であり特徴構築にかかるコストは非常に大きい．

特徴構築手法では特徴のいくつかを取り出して組合せを行うものがほとんどである．このため一部例外はあるものの特徴抽出のように一度に多くの特徴を扱う必要がない．また選択の手法は特徴選択と同じであるが，特徴を組み合わせることで，互いに相関が少なく多くの情報を持った特徴を作成することができるため，特徴選択よりも情報の損失が少なく次元削減を行うことができる．

[1] http://www.ics.uci.edu/~mlearn/MLRepository.html

第4章　指し手列の解析に基づく将棋における指し手の分類精密化

本章においては既存の語彙を時系列的に拡張する手法について提案する．既存の語彙としてコンピュータ将棋プレイヤ「激指」における指し手のカテゴリを用い，棋譜内で連続した指し手の系列を利用することでカテゴリを拡張する．

将棋においてこれまでの指し手は他の多くのゲームと同様に次の指し手を効率的に決定するのに重要な要素の一つである．このため多くのプログラムには手筋などとして取り入れられているがその統一的な扱いは今のところなされていない．このような指し手列に関する語彙の不足を解消し，この指し手の履歴を統一的に扱う方法として，本研究では指し手の履歴としてカテゴリの履歴を抽出した．そしてその履歴の存在する確率をもとに実現確率探索におけるカテゴリの遷移確率を指し手の履歴の情報を含んだカテゴリの遷移確率として拡張することを試みた．評価としてプロやインターネット将棋サーバの強いプレイヤの対局結果の棋譜 47,321 局についてカテゴリの抽出を行った．抽出したカテゴリの履歴を用いて遷移確率を拡張したコンピュータ将棋プレイヤと用いない元のプレイヤで対戦を行い 150 戦中 83 勝 67 敗と勝ち越すことができた．また問題集についても元のプレイヤよりも多くの問題を正解することが可能となった．

4.1　はじめに

コンピュータ将棋においてよい手を深く読むことで探索結果が改善されることは経験的に知られている．そのようなよい手を深く読む手法として鶴岡らの実現確率打ち切り探索 [120] が広く利用され，研究されている．実現確率打ち切りを用いた手法についてはその実現確率を計算する遷移確率の計算にはその局面と指し手の評価が用いられており，必然手・手筋などの用語でいわれるようなその局面に至った手順は一部には組み入れられているもののその統一的な扱い・定式化はほとんどなされていない．

その一方で指し手の履歴に基づいた研究は広くなされている．将棋に関する研究では大槻らによる n-gram 統計を用いた「必然手」の抽出に関する研

究 [116] や History Analyser を用いたカテゴリのより詳細な分類を行った研究 [117] がある．指し手の履歴の意図に沿った手は有望な手である可能性が高く，このような指し手の履歴を用いることは効率的な探索を行うのに有効であると考えられる．

このことから本研究では多くの棋譜を元に指し手の履歴としてカテゴリの履歴を抽出し，それをもとにこれまでの実現確率探索におけるカテゴリの遷移確率を指し手の履歴の情報を含んだカテゴリの遷移確率として統一的に拡張することを試みた．評価としてはプロやインターネット将棋サーバの強いプレイヤの対局結果の棋譜 47,321 局についてカテゴリの履歴を抽出し，コンピュータ将棋プレイヤに実装した．1 手 5 秒・220 手で引き分けのルールで，抽出したカテゴリの履歴に基づく拡張を行った遷移確率を用いたコンピュータ将棋プレイヤとこの拡張を行わず従来どおりカテゴリの遷移確率を用いたプレイヤで 150 試合対戦を行った．また次の一手問題集への解答についても評価を行った．結果としては 83 勝 67 敗と元のプレイヤに勝ち越すことができた．またこの拡張を行ったプレイヤは元のプレイヤよりも次の一手問題集において多くの問題を正解することができた．

本章の構成を以下に示す．4.2 節で関連研究を紹介する．次に 4.3 節で指し手の履歴の抽出に基づくカテゴリの拡張について提案する．そして 4.4 節でその評価について報告する．最後に 4.5 節でまとめと今後の課題を述べる．

4.2 関連研究

本節では本研究で用いた探索アルゴリズムである実現確率打ち切り探索について述べ，次に指し手の履歴を抽出・利用した研究について紹介する．

4.2.1 実現確率打ち切り探索

実現確率打ち切り探索 [120] とは鶴岡らが提案した深さの代わりに局面の実現確率を閾値として深さ優先探索をおこなう探索手法である．局面の実現確率とはルート局面からその局面が実現する確率であり

$$(\text{局面の実現確率}) = (\text{直前の局面の実現確率}) \times (\text{遷移確率}) \tag{4.1}$$

で計算される．ここで遷移確率とはある局面がその可能な指し手によって遷移しうる局面のうちのある 1 つの局面に変化する確率である．またルート局面は既に実現されているため実現確率は 1 である．この遷移確率をすべての手について求めることは困難であるので実現確率打ち切り探索では指し手の性質を分けたカテゴリを用いる．将棋についてこのカテゴリの実際の遷移確率を直接知ることは不可能であるため遷移確率は棋譜を用いて計算される．

$$(\text{遷移確率}) = \frac{(\text{実際にカテゴリの手が指された数})}{(\text{カテゴリの手が可能である局面数})} \tag{4.2}$$

で求められる．激指ではプロ棋士の棋譜は他のプレイヤの棋譜に比べて信頼度が高いことから羽生善治実戦集に含まれている約 600 局のプロ棋士の実戦棋譜を用いている [119]．実際には複数のカテゴリに含まれる手も多く存在するが激指ではそのような手についてはそれぞれの確率の最大値をその手の遷移確率としている．

実現確率打ち切り探索では探索手法そのものは Alpha-beta 探索アルゴリズムと同じであり，この局面の実現確率を閾値とする点が大きな違いであるが，この閾値を用いて探索を行うことでより「ありそうな」局面を深く読み「なさそうな」局面をあまり読まないという探索を行うことができる．

4.2.2 指し手の履歴についての研究

将棋には必然手・手筋などの用語で言われるような一連のまとまりとして扱うことのできる指し手が存在する．そのような手は多くのコンピュータゲームプレイヤにおいて特別な手として扱われている [121, 113]．またその局面に至った指し手はプレイヤの意図を反映したものである場合もありその指し手の意図にあった指し手を優先して指すことは探索を効率化するのに有効な手段であると考えられる．

ここでは将棋において指し手の履歴を用いた研究として 2 つの研究を紹介する．

指し手の履歴を用いた研究の 1 つとして大槻による n-gram 統計を用いた「必然手」の抽出に関する研究 [116] がある．この抽出した必然手を用いた探索ではその必然手以外の手の探索ノード数を一段減らすことで探索を効率化している．この探索ではコンピュータ将棋の進歩 2 [114] の問題 48 題中最善手・手順が変わらなかったものが 46 題あり，その中で 32 題において探索ノード数を削減できている．ただその探索ノード数は元の探索ノード数の 99.5%から 100.5%に 37 問がおさまっており，全体でも 99.76%までしか変化していない．大槻はこれは探索中の「必然手」の出現頻度が少ないためであると考察している．

またもう一つの指し手の履歴を用いた研究として竹歳らによる History Analyser を用いたカテゴリのより詳細な分類が挙げられる．History Analyser とは指し手の履歴から指し手の連続性を考慮した分類を行うものである．この分類の実現に向けての具体的な手法についてはあまり述べられていない．この分類により「連打の歩」や「歩のつき捨てからの歩のたらし」などのような指し手の分類が可能となっていると報告されている．また，盤面の進行状況による分類とこの分類をおこなったプログラムとそれらを行わないプログラムの対戦を行っている．対戦の詳細としてはルールとして 1 手 10 秒・300 手引き分けで 100 対戦を行っている．結果として 61 勝 38 敗 1 分けと詳細な分類を行ったプログラムのほうがおよそ 6 割の勝率で勝ち越したことが報告されている [117]．

またここでは詳しくは述べないが将棋以外の指し手の履歴に関する研究もある．これには例えば囲碁に関する研究として中村による過去の棋譜から n-gram 統計を用いて定型手順の獲得を行う研究 [118] や梶山らによる過去の棋譜をもとにその局面に至る着手の系列から次の着手の候補を生成する研究 [112] などが挙げられる．

4.3 履歴に基づく実現確率の拡張

実現確率打ち切りを用いた手法では遷移確率の計算には指し手を種類ごとに分けたカテゴリが用いられる．カテゴリにはその局面と着手が元になっているものが多く，必然手・手筋などの用語でいわれるようなその局面に至った経緯は一部には組み入れているもののその統一的な扱い・定式化はほとんどなされていない．

本手法では多くの棋譜から指し手の履歴をカテゴリの履歴として抽出しそれをもとにこれまでの実現確率探索におけるカテゴリの実現確率を拡張する．これによりこれまでの指し手の履歴を考慮した探索が可能となり，指し手の履歴の流れに沿った有望なノードを先に深く探索することができるようになると考えられる．本手法では次のように遷移確率を再計算することにした．

1. 棋譜からカテゴリに分類された指し手のカテゴリの履歴を抽出する．
2. n-gram 統計をもとに指し手の履歴による遷移確率 $P_{history}$ を推定する．
3. 履歴による遷移確率 $P_{history}$ と現在局面の特徴とその着手をもとにしたカテゴリの遷移確率 $P_{category}$ をもとに遷移確率 P を再計算する．

本節では以降この 3 つについてそれぞれ説明する．

4.3.1 指し手の履歴の抽出

将棋の手の種類は指し手の位置・駒の種類を合わせると非常に多く，抽出する棋譜に対してスパースであると考えられる．そのため本手法では実現確率探索におけるカテゴリを用いて指し手の履歴をカテゴリの履歴として抽出する．カテゴリを用いて履歴を抽出することでこのデータスパースネスの問題が緩和されることが期待できる．

4.3.2 n-gram 統計を用いた指し手の履歴による遷移確率

本手法では指し手の履歴に基づく遷移確率を推定するために n-gram モデル [68] を用いる．今回の指し手の履歴における n-gram とは n 個の連続した指し手のことである．例えば「歩で角，桂頭を攻める手」→「自玉の周辺に駒を

埋める手」という2つの連続した指し手は2-gram (bi-gram) である．n-gram モデルでは手の発生する確率がその直前の手に依存するという仮定があるものの，その扱いの容易さや有用性から n-gram モデルは自然言語を中心として多くの分野で用いられている．

本手法ではこの n-gram についてその次の手があるカテゴリに含まれている確率を求めることにより指し手の履歴による遷移確率とした．実現確率探索のカテゴリの遷移確率と同様この遷移確率を知ることは不可能であるため我々の手法では過去の棋譜を用いてこの遷移確率を求める．この棋譜の選択によりその確率は異なると考えられるが実現確率探索と同様，強いプレイヤの棋譜を用いることで有用な確率が得られることが期待できる．棋譜中に現れる n-gram の数を N，そのうち次の手がそのカテゴリ i に含まれている数 N_i とするとこの遷移確率 P_i は次のように計算できる．

$$P_i = \frac{N_i}{N} \tag{4.3}$$

我々の手法ではカテゴリを用いるためデータスパースネスの問題を緩和することはできるが解決することはできない．そのため本手法ではこのデータスパースネスの問題に対して次のような対策を行うことにした．このためにカテゴリが事前分布として同様の頻度で現れると仮定した古典的な手法の1つである Jeffreys Perks [68] を用いた．式 (4.3) について Jeffreys Perks を用いると確率 P_i は

$$P_i = \frac{N_i + 1/2}{N + C/2} \tag{4.4}$$

として計算することができる．ここで C はカテゴリの数である．これにより滅多に現れない連続した指し手の確率が大きくなることを避けることができる．

4.3.3 指し手の履歴を考慮した遷移確率

履歴を考慮した遷移確率を直接求めることは困難であるため，本手法では履歴による遷移確率 $P_{history}$ と現在のカテゴリの遷移確率 $P_{category}$ をもとに指し手の履歴を考慮した遷移確率 P を計算する．この $P_{history}$ と $P_{category}$ の相関を得ることは困難であるため，ここでは簡単のために2つの確率が互いに独立であるとの仮定をおく．これにより P は次のように計算することができる．

$$\begin{aligned} P &= 1 - (1 - P_{history})(1 - P_{category}) \\ &= P_{cateory} + P_{history}(1 - P_{category}) \qquad (4.5) \\ &\geq P_{cateory} \qquad (4.6) \end{aligned}$$

この式 (4.6) より遷移確率 P は全てにおいて $P_{category}$ 以上となることがわかる．つまりすべての局面において元の実現確率より履歴を含めた実現確率の

ほうが大きいということになる．これはこれまでに履歴による遷移確率を考慮していなかったためであり当然の結果である．

この増大した確率をそのままこれまでの探索に適用するには探索の閾値が同じでも多くの局面を探索することになり時間の調整が困難になる．我々が用いた将棋プログラム「激指」では遷移確率を扱う際に確率そのものではなくその対数を取ったLogPという値を用いている．激指では$P_{category}$が0.5のときにLogPの値を200としているが，指し手の履歴を考慮した遷移確率を用いる際には探索量を同じにするために$P_{category}$が$0.5 \cdot P_{history}$が$1/C$のときにLogPが200になるようした．これにより同じ探索の閾値を用いたときの探索ノード数がほぼ同じになることが期待される．

図4.1に[99]における疑似コードに履歴を追加したものを示した．n-gramの情報はツリーで表現されており，過去の指し手から現在の指し手までを辿ることで履歴による遷移確率$P_{history}$を取得できる．元のプログラムとの違いとしては遷移確率の計算の違いであり，13行目で盤面のこれまでの履歴からn-gramのデータベースを辿り，最後のn-gramツリーの葉ノードの親ノードを取得する．そして14〜18行目において指し手の履歴を考慮した遷移確率Pを計算し，19行目で遷移確率の高い方からソートしている（このため11行目では元の疑似コードとは違いソートされた手を返さない．）．ここのconstとは定数であり，$P_{category}$が$0.5 \cdot P_{history}$が$1/C$のときにLogPが200になるようにするための値である．そして以降は現在のカテゴリの遷移確率$P_{category}$の代わりに指し手の履歴を考慮した遷移確率Pを用いている．

4.4 評価

評価としては指し手の履歴を抽出しそれから得られた履歴を考慮した遷移確率を用いて探索を行った．本節ではこの評価についてその評価方法，得られた指し手の履歴，履歴を考慮した探索における対戦の結果・問題集の結果について示す．

4.4.1 評価方法

評価に用いる棋譜としてはプロの棋譜・将棋倶楽部24[1]のレーティング2,200以上の棋譜あわせて47,321局を用いた．この局面の中には総じて3,696,309手が含まれている．

将棋プログラムとしては東京大学近山・田浦研究室の有志の学生が開発した将棋プログラム「激指」を用いた．Intel Xeon 2.40GHz dual・メモリ2GB，AMD Opteron Processor 248 dual・メモリ2GBの2つの環境で評価を行った．言語としてはC++言語を用いて実装した．

[1] http://www.shogidojo.com/

```
1  int search(Board* board,
2             double realization_probability,
3             double min_realization_probability,
4             int alpha, int beta, Move& best_move)
5  {
6    // Cutoff test
7    if (realization_probability < min_realization_probability)
8      return leaf(board);
9    // Move generation
10   Move move[MAX_NUMBER_OF_MOVES];
11   int number_of_moves = generate_moves(board, move);
12   int best = alpha;
13   double *history_probability = n_gram_database->get(board->move_history());
14   for (int i = 0; i < number_of_moves; i++) {
15     move[i].probablility =
16       constant * (1.0 - (1.0 - move[i].category_probability)
17                * (1 -  history_probability(move[i]->category)));
18   }
19   sort(&move[0], &move[MAX_NUMBER_OF_MOVES]);
20   for (int i = 0; i < number_of_moves; i++) {
21     board->move(move[i]);
22     int value;
23     Move dummy;
24     if (move[i].probability < 0.5) {
25       // Preliminary search (Null window)
26       value = -search(board,
27                       realization_probability * move[i].probability,
28                       min_realization_probability,
29                       -(best+1), -best, dummy);
30       if (value > best) {
31         // Re-search
32         value = -search(board,
33                         realization_probability * 0.5,
34                         min_realization_probability,
35                         -beta, -best, dummy);
36       } else {
37         // Normal search
38         value = -search(board,
39                         realization_probability * move[i].probability,
40                         min_realization_probability,
41                         -beta, -best, dummy);
42       }
43       board->reverse();
44       if (value > best) {
45         best = value;
46         best_move = move[i];
47         if (best >= beta)
48           return best;
49       }
50     }
51   }
52   return best;
53 }
```

図 4.1: 履歴を考慮した実現確率探索アルゴリズム

カテゴリとしては激指に含まれている 60 個のカテゴリを用いた．激指ではそれぞれのカテゴリにおいてその位置などによりその実現確率を変えている．そのためカテゴリはもっと多くとることもできるが今回はそのようなものは考えないことにした．このカテゴリには当たりに関する手 (大駒への当たり，ききを通すあたりなど)，王手に関する手 (王手を防ぐ，駒得しながら王手など)，守る手 (囲いに関する手，自玉の周りに駒を埋める手など)，相手の手を防ぐ手 (事前に逃げる手，敵の打ちたいところに打つ手など)，取る手・成る手，探索に由来する手 (キラー応手など)，それ以外の手 (定跡手，ハッシュに格納された手) などが含まれている．

4.4.2 得られた指し手の履歴について

4.4.1 節における棋譜から抽出して得られた n-gram を表 4.1 に示す．表 4.1 には 2-gram, 3-gram, 4-gram についてエントロピーの上位 5 つずつ示している．ここでエントロピーとは n-gram の次に現れる手の偏りを示した値でありエントロピーが小さいほど次の手が偏っていることをあらわす．エントロピー E は式 (4.4) で定義した P_i を用いて

$$E = -\sum_i P_i \log P_i \tag{4.7}$$

で計算される．次の手をよく限定することが期待されるエントロピーが高いものについては棋譜の中に 1 万手以上存在し，データ過小の問題を緩和するのにこの手法は効果があると考えられる．表 4.1 によると抽出された指し手の多くは「カテゴリ外の手」となっている．これらの手は激指における特定のカテゴリに含まれていない手であり，このような手を意味をもった手として作成することは重要な課題の一つである．激指のカテゴリに含まれている手については 2-gram の結果より激指が持っている手筋が上位に現れており妥当な結果であると考えられる．また自玉周りを埋める手 (自玉近くに駒を埋める手) が一手置きに現れているものも多く見られる．これは序盤・中盤において囲いを作る手が抽出されているのではないかと考えられる．

4.4.3 履歴を考慮した探索について

履歴を考慮した探索の有効性を示すために指し手の履歴を考慮した遷移確率を用いた激指ともとのカテゴリの遷移確率を用いた激指とで対戦を行った．指し手の履歴を考慮した遷移確率については激指の持っている進行度をもとに前半・後半に分け 2-gram を用いることで得られた．4.4.1 節の Xeon を用いて定跡を抜けた 75 局面について先手後手を入れ替えて，1 手 5 秒・220 手で引き分けのルールで，150 対戦行ったところ 83 勝 67 敗という結果になった．この結果では統計的にはどちらが優位と判断することはできないものの，

表 4.1: 抽出された指し手の履歴 (n-gram の n=2, 3, 4 の場合についてそれぞれエントロピー上位 5 位まで)

n	指し手	エントロピー
2	手筋の 3 手目 取る手・成る手	2.548
2	自玉周りを埋める手 手筋の 3 手目	2.557
2	手筋の 3 手目 敵の長距離ききをブロックする手	2.583
2	自玉周りを埋める手 歩で角，桂頭を攻める手	2.633
2	カテゴリ外の手 カテゴリ外の手	2.637
3	歩で角，桂頭を攻める手 自玉周りを埋める手 カテゴリ外の手	2.491
3	カテゴリ外の手 カテゴリ外の手 カテゴリ外の手	2.545
3	自玉周りを埋める手 カテゴリ外の手 カテゴリ外の手	2.548
3	自玉周りを埋める手 手筋の 3 手目 敵の長距離ききをブロックする手	2.550
3	カテゴリ外の手 自玉周りを埋める手 カテゴリ外の手	2.562
4	自玉周りを埋める手 歩で角，桂頭を攻める手 自玉周りを埋める手 カテゴリ外の手	2.536
4	カテゴリ外の手 カテゴリ外の手 カテゴリ外の手 カテゴリ外の手	2.562
4	自玉周りを埋める手 自玉周りを埋める手 自玉周りを埋める手 自玉周りを埋める手	2.563
4	自玉周りを埋める手 カテゴリ外の手 カテゴリ外の手 カテゴリ外の手	2.565
4	カテゴリ外の手 自玉周りを埋める手 自玉周りを埋める手 自玉周りを埋める手	2.567

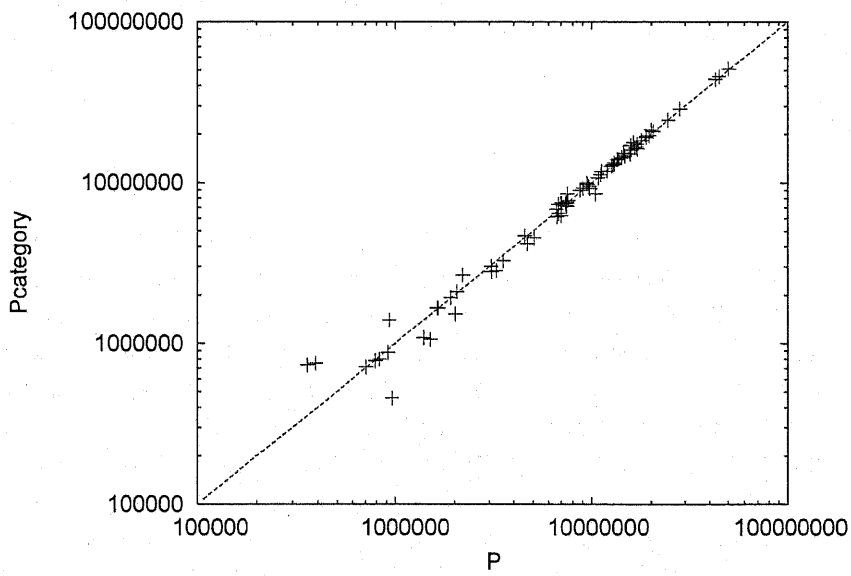

図 4.2: 対戦ごとの履歴を考慮した場合 (P) とそうでない場合 ($P_{category}$) のリーフノード数

指し手の履歴を考慮した遷移確率を用いた激指のほうが若干ではあるが勝ち越している．

次に図 4.2 に探索したリーフノード数を対戦ごとに載せた．図 4.2 においてノードが多いものについてみられるように時間制限が同じであってもリーフノード数は少なくなっている．これは履歴を考慮した遷移確率の計算にかかるコストのためであると考えられる．このコストのため元の激指に比べて (同じ局面について) 同じ時間で探索できるリーフノード数は 1%から 3%程度少なく平均して 1%程度リーフノード数が少なくなっている．このコストは現在の実装では LogP から確率 $P_{category}$ を計算しそれから P それを用いた LogP を求めるということを行っているためであり，本手法を用いる際にはこのコストを減らすことは可能であると考えられる．

また 4.4.1 節の Opteron を用いてコンピュータ将棋の進歩 2 [114] の次の一手問題 48 題を解く実験を行った．解答は激指におけるレベル 12 (LogP=1200, 深さ 12 段に相当) で行った．結果を表 4.2 に示した．総じて正答数は 4 問増えており，不正解だったものを正解できたものが 8 問，正解だったものが不正解になったものが 4 問あった．また最善手が変わらなかったものは 15 問あった．この結果より多くの問題について探索結果が大きく変わっており履歴を考慮した遷移確率により探索に非常に大きな影響があったことが分かる．次の一手問題のみでその探索の効率化の性能を測ることはできないが，この問題集についてはレベル 13 (LogP1300) での正答数が 30 問であることを考慮すると指し手の履歴を考慮することで少ないリーフノード数で効率的に探索できていることが分かる．

4.5 おわりに

本章では指し手の履歴を実現確率探索におけるカテゴリを用いて抽出することにより，指し手の履歴を考慮した遷移確率を求める手法について提案した．結果としては 4.4 に示した通り，83 勝 67 敗と若干であるが元のプレイヤに対して履歴を考慮したプレイヤが勝ち越すという結果になった．またコンピュータ将棋の進歩 2 [114] の問題についても元のプレイヤが正解した問題数よりも多くの問題を正解することができた．この結果よりカテゴリを指し手列という時系列的な共起関係を用いて拡張する手法は有効であることがわかった．

今後の課題としては今回用いたカテゴリの整理が挙げられる．今回の方法を用いることで手の履歴による遷移確率 $P_{history}$ と局面による遷移確率 $P_{category}$ を分けて考えることが可能となった．この $P_{history}$ と $P_{category}$ を一度に扱うにはそのスパースネスの問題や局面と手の履歴両方から実現確率を計算する困難さの問題がある．現在のカテゴリには「駒得しながら王手」や「歩のあたり」など直前の手によらないものもあれば「直前に動いた駒を取る手」など

直前の手によっているものもあり，$P_{history}$ と $P_{category}$ を独立したものとして扱うという前提は今回のカテゴリについては誤っていると考えられる．しかし一方でこの誤った仮定を用いても手の履歴による遷移確率を用いることで勝ち越しており，カテゴリを直前の手によらないもので整理することにより，より正確な実現確率の計算が可能になるのではないかと考えられる．

表 4.2: コンピュータ将棋の進歩 2 の問題 48 題の結果 (レベルは実現確率における閾値であり同等の深さに相当)

	正答数	時間 [s]	探索ノード数	リーフノード数
P (レベル 12)	29	107	25,578,562	16,492,634
$P_{category}$ (レベル 12)	25	107	25,617,309	16,537,701
$P_{category}$ (レベル 13)	30	228	54,234,789	34,673,338

第5章　コンピュータゲームプレイヤにおける評価要素の自動生成

本章においては既存の語彙を同時に起こる共起関係をもとに拡張する手法について提案する．既存の語彙として単純な特徴を使い，そのうち同じ局面内に共起するものを抽出し，選択することで局面を評価する評価関数における評価要素を作成する．

評価関数はコンピュータゲームプレイヤにおいてその対象ゲームの知識を表現する部分であり，振る舞いを決める重要な要素の一つである．評価関数はゲームにおける局面の特徴を表す評価要素の重み付き線形和で表現するのが一般的である．この評価要素の選択にはその対象とするゲームに関する深い知識が必要である．

本章ではゲームの履歴に基づいた高速でスケーラブルな評価要素の自動生成手法について提案する．本手法では多くのゲームに応用可能な2クラスの分類問題を対象とし，評価要素を単純な特徴の組み合わせで表現し，頻度と条件付き相互情報量の2つの指標を用いて選択する．

本手法の評価としては200,000のOthelloの局面を用いOthelloの位置と色の192個の単純な特徴から評価要素の作成を行った．これにより分割処理によって大きな問題を扱うことができ，並列化により高速に実行できることを確認できた．また，生成した評価要素を用いたNaïve Bayesian分類器は他の分類器による単純な特徴を用いた分類よりも良い精度で分類できることを示し，有用な評価要素が得られていることを確認した．

5.1　はじめに

評価関数は探索手法とともにコンピュータゲームプレイヤの振る舞いを決める重要な要素の一つである．評価関数はゲームの局面を評価し，その局面の有利・不利の度合いを表す評価値を出力する．この評価値は探索中に出現する局面同士の比較が可能なものでなければならない．この評価関数は局面の特徴を表す評価要素の重み付き線形和で表し，評価値としてスカラー値を出力するのが一般的である．

このような評価関数は対象ゲームに関する知識を表現した最も重要な部分であり，現在，多くのゲーム，特に将棋・囲碁などの比較的複雑なゲームにおいては，評価関数 (特にその評価要素) は人手で作成するのが一般的である．人手の生成では適切な評価要素を選択し，その評価要素の重み付けを行う．このためには重要な評価要素の選択に対象とするゲームに関する深い知識が必要であり，また重み付けの調整についても多くの時間が必要である．このようにして出来た評価関数についても実際にはそれで完成ではなく，実際に知っているにもかかわらず気付かずに入れ忘れた，もしくは，表現が思いつかず入れられなかった評価要素や重複して考えてしまったために重みづけにおいて重視しすぎた，もしくは軽視しすぎたような重みなどのためにおかしな振舞いをしてしまい，それについて気づく限り調整を行っていくということが頻繁になされている．このように人手での作成手法はアドホックなものであり，多くの知識・時間を必要とする上に終わりがないというのが現状である．

この人手のコストを削減しつつ，高精度に局面を評価する評価関数を作成する手法として評価関数の自動生成がある．現在，多くのゲームについてインターネットに存在する対局サーバでの対局結果やプロの棋譜データなど多くの対戦結果を得ることが容易になっている．また，そのような結果が得にくいゲームにおいても計算機の計算能力の向上・計算資源の増大により多くの自己対戦を短期間で行うことができるようになってきている．これらから得られる情報を用いて，バックギャモン [94] や Othello [16] などの比較的簡単なゲームにおいては評価関数の自動生成に関する研究が盛んになされており，人手で作った評価関数よりも有用な評価関数が作成されその結果人間よりも強いプレイヤが多く作成されているなど有用な結果が得られている．このような有用な結果が得られている一方で，人間のチャンピオンやプロにまだ届いていないような現在のところコンピュータゲームプレイヤには難しく数多くの研究の対象とされている将棋や囲碁などの難しいゲームへの応用はなされていない．また，どのゲームにも適用できるような決定的な手法も見つかっていない．これは計算コストが高かったり，また特定のゲームのための知識表現を他のゲームに適用することが困難であったりするためである．

ここではゲームの対戦結果に基づくゲームに対する深い知識を必要としない評価要素の自動生成手法を提案する．本手法は対局の勝ち・負けや詰み問題の詰み・不詰などの 2 クラス問題を扱い，多くの種類のゲームに広く適用可能である．本手法では 2 値の単純な特徴を基にその共起したものを論理積で表現し，頻度と条件付き相互情報量を用いて選択したものを評価要素とする．この生成された評価要素にこれまでに提案されている手法を用いて重みづけを行うことで評価関数の作成が可能である．本手法では問題を分割して処理する方法を提案しており，これによりこれまで多くの手法が適用困難であった大きな問題を扱うことを可能にすると共に並列化による高速化も可能

にしている.

本手法の評価としては200,000のOthelloの局面を用い，Othelloの位置と色の192個の単純な特徴から評価要素の生成を行った．生成した評価要素を用いたNaïve Bayesian分類器は他の分類器による単純な特徴を用いた分類よりも良い精度で分類でき，有用な評価要素が得られていることを確認した．また分割処理によって大きな問題を扱うことができ，並列化により高速に実行できることも確認できた．

本章の構成は次のようになっている．5.2節で関連研究を紹介する．次に5.3節でコンピュータゲームプレイヤにおける評価要素の自動生成について提案する．そして5.4節で評価方法について5.5節で評価結果について報告する．最後に5.6節でまとめと今後の課題を述べる．

5.2 関連研究

評価関数の生成に関する研究は1950年代から行われている [39]．多くの手法では評価要素は人手で行い，評価要素の重み付けのみを自動化している．このような手法にはニューラルネットワークや強化学習 [39]，順序相関を用いた手法 [43]，最適制御理論をもとにした手法 [51] がある．

評価関数の自動生成については1990年代から比較的簡単なゲームについて研究されている．これらの手法は過去のゲームの対戦履歴を用いてその入力局面を表現する単純な要素を元に評価関数を作成する．評価関数の自動生成は評価関数の作成の方法から直接的な手法と階層的な手法の2つに分けることができる．

直接的な手法では多層ニューラルネットワークなどを用いて単純な要素からなる評価関数を作成する．この方法では評価関数を評価要素の非線形結合で表す．この方法は表現能力が高く，より正確な評価関数を実現できる可能性がある．しかしその一方で計算コストが高く，生成された評価関数の分析が困難である．このような研究には強化学習を用いたTD-Gammonや進化的アルゴリズムを用いたFogelによる○×ゲーム・Checker (Blondie24) [35]・チェス [36]，Kumarらによる Checker [22]，Particle Swarm Optimization (PSO) を用いたMesserschmidtの○×ゲーム [73] やFrankenのcheckers [37] などがある．

階層的な手法では人手で評価関数を作る時と同様，単純な要素から有利・不利に関係の深い要素を生成・選択し，次にその評価要素を最適に組み合わせるという，2階層に分けた評価関数生成を行う．この方法では多くは評価関数を評価要素の線形和で表す．この方法では直接的な手法ほど表現能力は高くないが，生成・実行コストが少ない．また，個々の評価要素とその重みを直に見ることができるため，解析や人手での高速化などが可能である．このような研究にはELF [102],GLEM [16], ZENITHやそれを改良した金子らの方

法 [56], Dummy の方法 [28] などがある.

これら評価要素の自動生成手法は多くのコンピュータゲームプレイヤに適用され成功を収めている [3]. しかし, その計算コストや知識表現の制限のため簡単なゲームについてしか適用されていない. 直接的な手法と階層的な手法のどちらが優れているということは一概には言えないが複雑なゲームへの応用という点においては生成コストの少ない後者の方法が優れていると考えられる.

5.3 提案手法

本手法では 2 クラスにラベルづけされた訓練局面を元に評価要素を自動生成する. この訓練局面は前述したとおり容易に取得できる. このような 2 クラスの問題には勝ち・負けや詰み・不詰などがあり, 多くのゲームに適用できる.

本手法の処理の流れを図 5.1 に示した. 局面を表現する 2 値の単純な特徴 (以降, 特徴要素) を用意し, それらの論理積 (以降, パターン) から選択を行うことで評価要素を作成する. 生成した評価要素にこれまで提案されてきた重み付け手法を適用すれば自動的な評価関数の生成が可能である.

本手法では計算コストを抑えつつ重要なパターンを選択するため頻出・飽和と条件付き相互情報量の基準による選択を行う. まず頻出飽和パターンの選択を行う. これは全てのパターンを評価する必要をなくし計算コストを減らすとともに, 過学習の危険を回避するためである. この頻出飽和パターンの選択には頻出飽和パターン選択アルゴリズムである LCM (Linear time Closed set Miner) [100] を用いる. 次にその頻出飽和パターンから条件付き相互情報量を基に選択する. 条件付き相互情報量はラベルをよく表す重要な特徴を選択し, 従属性の高い (似たような) 特徴を削減するために用いる. この選択には特徴選択アルゴリズム CMIM (Conditional Mutual Informaiton Maximization) [34] を用いる.

本手法ではまた LCM, CMIM それぞれのアルゴリズムについて問題を分割して処理する手法についても提案する. この提案によりそれぞれがこれまで扱うことのできなかった大きな問題に対処することができる. またこの分割した処理を並列に実行することにより, 評価要素の選択を高速に実行できる.

本手法の一例として図 5.2 に特徴要素からの評価要素の生成を示す. A, B, C, D の 4 つの特徴要素から特徴要素と空パターンを含む 16 ($=2^4$) のパターンが生成される (図 5.2 には空パターンは示されていない.) . これらのパターンから訓練局面における頻度と条件付き相互情報量を元に評価要素を選択する.

本節では本手法で対象とする評価関数の表現について説明したのちに, 特徴要素の抽出と特徴要素を元にした評価要素の生成について説明する.

5.3.1 評価関数の表現

評価関数は評価関数はゲームの局面を評価し，その局面の有利・不利の度合いを表す評価値を出力する．この評価値は探索中に出現する局面同士の比較が可能なものでなければならない．このように正確なものが求められる一方で評価関数は探索中に用いられるものであり，コストによっては探索を少なくする必要があるためその速さも同時に要求される．このように評価関数には高速かつ正確であることが要求され，それぞれはトレードオフの関係にあり良いトレードオフ点を見つける必要がある．

高速な評価関数であるためにほとんどの評価関数 F は次のような特徴要素の重み付き線形和で表現される．

$$F(p) = \Sigma_i w_i \cdot f_i + b \tag{5.1}$$

ここで，p は入力局面，f_i は (真偽値の値を持つ) 評価要素，w_i, b は定数である．

正確な評価関数の実現には線形和よりも表現能力の高い表現が好ましく，より効率的な表現ができる可能性がある．しかし，実際にどのような表現が適しているのかは明らかではないために表現によってはコストのみが増えるということになりかねない．また，実行にかかる計算コストが変化するため実際に利用する際には探索を用いて探索を含めた精度を評価する必要があり，それを考慮した評価を評価関数ごとに行うのはコストが高く，多くのゲームでは困難である．また過学習の危険も考えられる．このように表現能力の高い表現を上手く選び用いることには非常に大きな労力が必要となる．また，線形[1]の関数を用いることには評価要素を個別に確認でき，w の値がその評価要素の重要度を直に表しているため解析・チューニングが容易であるという利点がある．このようなことから単純な線形和を用いることが多く，本研究においてもこのような形の評価関数を想定して評価要素の生成を行う．

5.3.2 特徴要素・パターン・評価要素

評価関数に局面を入力するには局面の何らかの表現方法が必要である．この方法として我々は人手で選択した単純な 2 値の要素を用いる．この要素はゲームに現れる全ての局面を区別して表現できる必要がある．

本手法では局面の情報を表し，意味の上でそれ以上分割できない真偽値の要素をこの単純な 2 値の要素 (特徴要素) として選択する．この特徴要素は最小の評価要素である．特徴要素としては例えば○×ゲームについて「左上に○がある」というものが挙げられる．この特徴要素はそれを用いることで正確に全ての局面を区別できるように選択する必要がある．このような特徴の

[1] この線形というのは特徴要素に関して線形という意味ではなく評価要素に関して線形という意味である

発見はルールの記述もしくはルールに見合ったプレイヤの作成の過程において可能であると考えられ，対象ゲームに関する深い知識を必要とするようなものではない．[2]

本手法は対象を真偽値としているが，真偽値でない離散値については2値化の手法や離散値それぞれを特徴とした真偽値であらわすことで利用可能である．また連続値についても2値化やその範囲を区切る離散化 [32] の手法を用い離散値にすることで同様に利用できる．

本手法では特徴要素を論理積で組み合わせることで新たな特徴を作成する．この特徴要素の論理積であらわされる組み合わせをパターンと呼ぶ．ある局面においてパターンに含まれる全ての特徴要素が真となるときにそのパターンは真となり，それ以外の場合は偽となる．パターンが真である場合とはそのパターンがその局面に現れているという場合であり，その逆も成り立つ．評価要素はパターンのうち評価に有用なものを選択して用いる．本手法では論理和や否定・四則演算など別の組み合わせは扱わない．このため，評価要素の表現力は下がるが，評価要素の生成コストを下げられる．ただし，論理和は評価要素の線形和によってある程度表現でき，また特徴要素の否定は特徴要素に含めることで利用できる．

5.3.3 頻出パターンの抽出

本手法ではまず訓練局面によく現れるパターンのみを抽出する．これは特徴要素の数を n とするとパターンの数は 2^n となり莫大な数となり，これらを全て評価要素として扱うこと・これらを全て評価して選択することはできないためである．またこの抽出により訓練局面にめったに出現しない過学習の原因となるパターンは取り除かれる．ここでこの頻出パターンの抽出により重要であるが滅多に現れないパターンが見逃されている可能性はあるが，そもそもそのようなパターンの重要性は訓練局面のみでは評価できない．

頻出パターンは次のように定義される．

Definition 1 頻出パターン
N 個の訓練局面の中に α 個以上出現するパターン

この α は最小サポートと呼ばれる．ある k 個の特徴要素からなるパターンの出現数は，そのパターンと特徴要素の論理積であらわされる $k+1$ 個のパターンの出現数以上となる．このため特徴要素を加えながら (もしくは減らしながら) 調べることで効率的に頻出パターンを抽出できる．

[2]特徴要素としてより高度な要素をあらかじめ人手で与えることも可能である．これは，例えば○×ゲームにおいてその結果を決定づける「○がそろっている」「×がそろっている」というものである．定石などの決まった知識や勝敗を分けるような決定的なルールの存在するゲームにおいては，そのようなものが評価要素として選ばれる可能性が高い．また，そのようなものを特徴要素としてあらかじめ与えることで，評価関数の表現能力が向上するとともに評価要素の生成のコストを削減できる．

パターンにはあるパターンが出現したら必ず同時に出現し，それ以外の場合には出現しないようなパターンも存在する．このような完全従属なパターンは訓練局面について同じ情報を持ったパターンであるため，CMIM による選択において１つを除き残りはすべて削減される上にその削減にコストがかかり無駄なパターンであるといえる．この無駄なパターンは飽和パターンのみの抽出により事前に効率的に削減できる．ここで飽和パターンとは次のように定義できる．

Definition 2 飽和パターン
出現する局面集合が等しいパターン集合の極大元 (ただし，極大を与える半順序は特徴要素の集合であるパターンの集合包含関係)

図 5.3 に訓練局面から頻出パターン・頻出飽和パターンを抽出した例を示す．特徴要素には 1 から 9 があり，左端の列は 6 つの訓練局面をその値が真である特徴要素で表現している．最小サポートは 3 であり，3 つ以上の訓練局面に現れるパターンが頻出となる．頻出パターンにおいて同じ訓練局面に現れるパターン集合のうち，他のいずれのパターンにも含まれない極大のパターンが頻出飽和パターンとして選択される．図 5.3 では同一の訓練局面に現れている頻出パターンを中央の列の各行に示している．その中の極大のパターンが右端の列の同じ行の頻出飽和パターンとして選択される．この表における行の関係はわかりやすさのために恣意的に表記したものであり，実際にこの順序で見つけられるとは限らない．

頻出飽和パターンを抽出する方法としてはその特徴要素の積を出現する要素の組み合わせと見なすことでデータマイニングにおけるバスケット分析などで用いられる頻出飽和アイテム集合抽出手法が有用である．頻出飽和アイテム集合抽出手法にはアプリオリ法 [2] や Fp-growth [46] を始めとして非常に多くの手法が提案されているが，我々はこの手法として LCM (Linear time Closed set Miner) [100] を用いた．これは LCM が IEEE ICDM'04 (IEEE International Conference on Data Mining 2004) 併設のワークショップ FIMI'04 (Workshop on Frequent Itemset Mining Implementations) におけるコンテストにおいて優勝しており，頻出飽和パターン数え上げにおいて最も高速なアルゴリズムの１つであるためである．また他の手法に比べて，少ないメモリで数え上げが可能であることも同時に示されているため今回のような大規模なデータからパターンを抽出する際には有用であることも LCM を用いる理由である．本節では以降この LCM アルゴリズムとその拡張について説明する．

LCM

LCM は Prefix 保存飽和拡張を用いた深さ優先探索とデータベース縮約により，高速にかつ解保存用メモリを必要とせずに重複無く頻出飽和パターンを抽出する手法である．LCM の時間計算量は頻出飽和パターンの数を n と

すると $O(n)$ となる．図 5.4 に LCM の探索木を示す．ここでは LCM の特徴である Prefix 保存拡張，データベース縮約について説明する．

まず Prefix 保存拡張を用いた数え上げの順序決定について説明する．LCM の探索木において，全てのノードは頻出パターンに対応しており，ルートノードは空パターンである．ノードの子ノードはそのパターンにある一定の順序で特徴要素を加えることで作成される．このため子ノードは親ノードのパターンを含むことになる．この親ノードのパターンから子ノードのパターンに一定の順序でパターンを拡張する手法を Prefix 保存拡張と呼ぶ．この順序を保存していることで最後に追加したパターンさえ覚えておけばよく，これまで探索した情報全てを保持し続ける必要がなくなりメモリを節約して効率的に探索できる．またパターンは一度しか出現することがないことが保証されるため重複なく抽出できる．このパターンの拡張により LCM は高速に数え上げることができる．

次にデータベース縮約について説明する．LCM の探索木においてそれぞれのノードはその対応しているパターンを扱うのに十分な情報を持ったデータベースを保持している．このデータベースを出現表と呼び，この出現表は縮約後に子ノードに伝搬される．この出現表に含まれなければならない情報は現在のノードのパターンを含むデータとその頻度である．このデータはこれから追加してテストしなければならない特徴要素のみで表現されていればよい．親のパターンにパターンを追加することで子のパターンが作成されるため，親のパターンの含まれるデータは子のパターンの含まれるデータを含んでおり，子のパターンの含まれるデータは親のパターンの含まれるデータより小さくなる．このようなことから探索木の深いノードほど出現表は小さくすることができる．LCM は深さ優先で探索を行うことでルートノードから探索しているノードまでのノードの出現表のみを保持すればよく LCM は低い空間コストで探索できる．

LCM ではこのように記憶しておかなければならない情報を削減することでメモリを節約しながら，探索順序を一定にすることで重複無く頻出飽和パターンを抽出する．

LCM に対する拡張

我々はこの LCM に次の 2 つの拡張を行った．

1. 情報利得による選択
2. LCM の探索木の分割による並列化

頻出飽和パターンは CMIM (5.3.4 節参照) での選択を行うには多すぎる場合が多く，そのような場合には高速な特徴選択手法である情報利得による選択が有用である．この情報利得は出現表への訓練局面のラベルの挿入により容易に計算できる．このラベルの挿入はラベルに矛盾がないという前提では

出現表のエントリ分のみ増えるだけで済み，コストも非常に少なく拡張できる．ここでパターンの情報利得とはラベルとパターンの相互情報量 (5.3.4 節参照) である．このような情報利得の高いものから選択を行うことで，そのパターンだけでラベルとの関係の高いような特徴のみを選択することができる．ただし，この選択はパターン同士の関係を無視して選択する手法であるので，必ずしも CMIM で選択されないものを破棄するとは限らない．この手法は CMIM での選択のコストを減らすためにのみ行うものであり，この基準で選択された上位のパターンが必ず評価要素になるなどというものではない．

LCM においてはその Prefix 保存拡張により同じパターンが複数回現れることはなく，またそのノードのパターンは親ノードのみに依存しているので，兄弟ノード間に依存性がない．このため LCM はその探索木の浅いノードを分割することで小さな問題に分割でき，その並列化は表 5.1 に示したように単純なワークキューを用いた手法で行うことができる．前提として全てのデータは同じデータを持っているものとする．ワークキューとは最も単純な並列化手法の１つであり，サーバがタスクを追加するキューを用意し，クライアントがそのキューからタスクを取り出し処理を行うことで並列化を実現する手法である．この並列化において全てのノードは頻出パターンを抽出する訓練局面を保持している必要がある．訓練局面が多くなるにつれて出現表とその送受信のコストは大きくなる．そのためパターンを送信し，クライアントは局面とパターンを用いて１から局面表を構築する．この局面表の構築は並列化することによって必ず起こるオーバヘッドとなる．またその方法によってオーバヘッドの影響が異なるような，並列化によって起こるオーバヘッドには次のようなものが考えられる．まずサーバがワークキューにクライアント数分のタスクをいれるまで暇なクライアントが存在する．またタスクの分割によっては多くの並列化手法と同様クライアントのタスクの不平等が起こり，適切な粒度になるまでサーバがタスクを分割しなければ終了時近くに暇なクライアントが発生する．このオーバヘッドはより複雑な仕組みをもった並列化手法によって小さくできるが今回の並列化では処理ができることを目標としているのでこのような単純な手法を用いた．これらの分割した探索において重複して頻出パターンを数えることはないので，結果を統合する処理は必要ない．

図 5.5 に並列 LCM のサーバの疑似コード，図 5.6 に並列 LCM のクライアントの疑似コードを示した．ここでは簡単のため，サーバはクライアント数を知っていること，ファイルは事前に共有していることを前提としている．また，通信やロックについては記述していないが，ワークキューへのアクセスは適宜，通信やロックがなされているものとする．サーバでは 13 行目でツリーを１段展開したのち，十分な粒度になるまで適宜展開する．サーバは深さ優先で探索を行っていないため効率が悪いが，多くの問題では１段目で十分な量のタスクができ，深く探索されることはない．タスクが十分な量できるま

表 5.1: 並列 LCM

サーバ

step 1,	頻出する特徴要素を抽出し，Prefix 保存拡張のための特徴要素の追加の順番を決める
step 2,	特徴要素が少ない，もしくはクライアントのタスクの不平等が起こる場合には特徴要素の一部に対して探索を行う
step 3,	探索が必要な頻出パターンをワークキューに追加する

クライアント

step 1,	頻出する特徴要素を抽出し，Prefix 保存拡張のための特徴要素の追加の順番を決める
step 2,	ワークキューからタスクを受け取る
step 3,	ルートノードからタスクに含まれるパターンまで探索を行うことで出現表を構築する
step 4,	タスクに含まれるパターンを含む頻出飽和パターンを探索，出力する
step 5,	ワークキューが空なら終了，空でなければ step 1 に戻る

でクライアントが待つかどうかは自由であるが，ここではクライアントは十分の粒度のタスクができるのを待つこととし，サーバが 19 行目に行ったところでクライアントにタスクができたことが通信されることとする．クライアントはワークキューからタスク (パターン) を受け取り，出現表を計算したのち，それ以降を LCM を用いて探索し，自分のローカルディスクに出力する．

5.3.4 条件付相互情報量による評価要素の選択

頻出飽和パターンの選択は全てのパターンを評価する困難を回避するために行ったものであり，選択されたパターンが対象問題に対して有用であるかどうかについては評価していない．そのため頻出飽和パターンから有用な特徴のみを選択する必要がある．このような特徴選択には機械学習で用いられている特徴抽出や特徴選択の手法が有用である．頻出飽和パターンは，パターン全体に比べれば少ないとはいえ非常に多く，その特徴全体を元に選択する PCA などの特徴抽出の手法やラッパーアルゴリズムのような直接学習を行うことによって特徴選択を行うようなアルゴリズム [55] を用いることは難しい．このため用いることのできる手法は特徴選択において学習以外の代替の評価基準を用いて行うフィルタアルゴリズムとなる．このようなコストの低いフィルタアルゴリズムとしてまず考えられるのがカイ 2 乗値や情報利得 [29] などパターンそれぞれを個別に扱う手法であるが，これは高速であり有用ではあるもののその選択した要素の冗長性を回避できない．このため我々は情報量に基づき冗長性を回避しつつ重要な要素を選択できる CMIM (Conditional Mutual Informaiton Maximization) [34] を用いる．本節では以降，CMIM について説明したのち，その特徴を分割して選択することで空間コストを一定にしつつ特徴選択を行う手法として分割統治 CMIM を提案し，さらにその分割統治 CMIM を並列化する手法である並列 CMIM について説明する．

CMIM

CMIM はラベルに関する情報を最も大きく得られる冗長性の低い特徴の部分集合の選択を目的とした手法である．このために CMIM はこれまで選んだ特徴とラベルがある時にそれに追加することで増える情報量ができるだけ大きい特徴を次の特徴として選ぶ．具体的に CMIM は次のようにこれまで選択されたそれぞれのパターンとの条件付き相互情報量の最小値が最大になるパターンを選択する．

$$
\begin{aligned}
v(1) &= \arg\max_n \hat{I}(Y;X_n) \qquad (5.2)\\
v(k+1) &= \arg\max_n \left\{ \min_{l\leq k} \hat{I}(Y;X_n|X_{v(l)}) \right\}\\
&\qquad (1\leq k < K)
\end{aligned}
$$

ここでYはラベル，Xはパターン，Kは選択されるパターンの数である．また$\hat{I}(Y;X_n)$は相互情報量，$\hat{I}(Y;X_n|X_{v(l)})$は条件付き相互情報量である．相互情報量$\hat{I}(Y;X)$とはXが持つYに関する情報量ということができ，次のように計算できる．

$$\hat{I}(Y;X) = \hat{H}(Y) + \hat{H}(X) - \hat{H}(Y,X) \tag{5.3}$$

ここで$\hat{H}(X)$はエントロピーと呼ばれ確率変数Xの乱雑さを示す．エントロピーは

$$\hat{H}\ (\ X_1,\ldots,X_n) = -\sum_{x_1,\ldots,x_n\in\{0,1\}^n} p_{x_1,\ldots,x_n}\log p_{x_1,\ldots,x_n} \tag{5.4}$$

と計算される．ここでnはパターン数，x_iはi番目のパターンの値，$p_{x_1,\ldots,x_n}$は訓練局面についてその全体に対する$X_1,\ldots,X_n$が$x_1,\ldots,x_n$である割合である．条件付き相互情報量$\hat{I}(Y;X|Z)$とはZを知った上でのXの持つYに関する情報量ということができ，次のように計算できる．

$$\begin{aligned}\hat{I}(Y;X|Z) &= \hat{H}(Y|X) - \hat{H}(Y|X,Z)\\ &= \hat{H}(Y,Z) - \hat{H}(Z)\\ &\quad -\hat{H}(Y,X,Z) + \hat{H}(X,Z)\end{aligned} \tag{5.5}$$

CMIMではこのようにこれまで得られたパターンとの冗長性を評価しつつ，ラベルとの相関をみる条件付き相互情報量を利用することで依存の少ないパターンを選択できる．CMIMでは扱う全てのパターンを同時に評価する必要がないため，比較的高速に選択できる．Mをパターンの数・Nを訓練局面の数・Kを選択するパターンの数とすると CMIMの時間計算量は$O(KMN)$，空間計算量は$O(MN)$となる．

分割統治 CMIM

ここでは CMIMのパターンを分割して選択を行う手法として分割統治 CMIMを提案する．

CMIMにおいては相互情報量と条件付き相互情報量の計算が計算のほとんどを占めるため高速な処理を行うためには主記憶での処理を行うことが望ましい．しかし頻出飽和パターンが非常に多い場合にはこれは不可能となる．この空間コストの問題を解決するために分割統治に似た方法を用いてパターンの選択をおこない，それぞれの分割された部分問題について処理することで主記憶上での計算が可能となる．

分割統治 CMIMではパターンをいくつかに分割し，それぞれについて選択を行うことでそれぞれの選択を主記憶で処理できるようにする．分割を行う方法としては訓練データを分割する方法とパターンの集合を分割する方法が

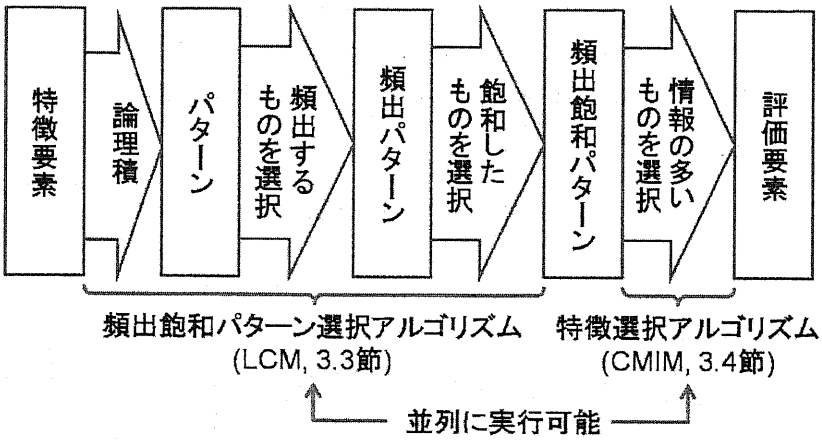

図 5.1: 特徴要素からの評価要素生成の流れ

表 5.2: 分割統治 CMIM

step 1,	パターンをランダムに S 個ずつの集合に分ける
step 2,	それぞれのパターンについて step3-4 を繰り返す
setp 3,	訓練データをパターンを用いて表現する
step 4,	CMIM を用いて一定値 (打ち切り条件付き相互情報量と呼ぶ) 以上の 条件付き相互情報量をもつパターンを選択する
step 5,	選択したパターンを統合する.
step 6,	選択されたパターンがある基準に達しない場合には 1 に戻り同様の操作を繰り返す

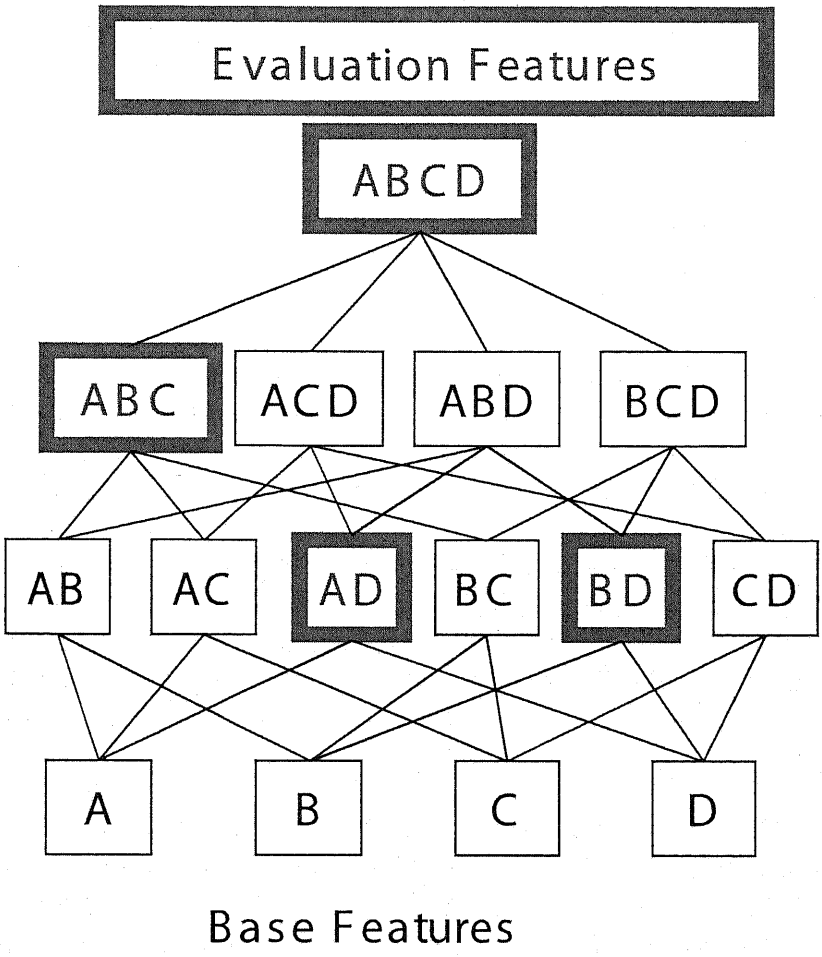

図 5.2: 特徴要素・パターン・評価要素

訓練局面	頻出パターン	頻出飽和パターン
{1, 2, 5, 6, 7, 9},	{1} {1, 7} {1, 9} {1, 7, 9}	→{1, 7, 9}
{2, 3, 4, 5}, {2},	{2, 7} {2, 9} {2, 7, 9}	→{2, 7, 9}
{1, 2, 7, 8, 9},	{7} {9} {7, 9}	→{7, 9}
{1, 7, 9}, {2, 7, 9}	{2}	→{2}

図 5.3: 訓練局面・頻出パターン・頻出飽和パターン (最小サポート=3).

考えられる．訓練データを分割した手法については [76] において以前提案しているが，パターン数に対して学習データをより少なくするということになってしまい，選択が困難になってしまっていることが考えられるため，パターンを分割する手法の方が望ましい．分割統治 CMIM の詳細について表 5.2 に示した．step 4 の基準については繰り返しの数 (今回の実験では 5 回とした) や step 4 で選択されなかったパターンの数があげられる．それぞれの分割されたパターンの集合については処理を独立に行うことができ，低い空間コストで選択できる．このそれぞれの選択の空間コストは $O(SN)$ であり分割以前のパターンの数によらない．この分割によって得られたパターンは全てから一度に選択したパターンと同じものが選択されるとは限らないが，CMIM の観点からはほぼ同じ情報を持つ．

CMIM による選択とこの分割統治 CMIM を用いた選択の相違を示すために NIPS2003 における特徴選択チャレンジ[3]の問題の一つである Dorothea を用いて事前実験を行った．Dorothea は 139,351 の特徴で表された 800 の訓練例からなる．我々はまず最小サポートを 16 (2%) として頻出飽和パターン 146,380 パターンを選択した．次に打ち切り条件付き相互情報量を 0.001, 0.0001 としてこの頻出飽和パターンから選択された特徴の CMIM によって選択された特徴との相違を図 5.7 に示した．coverage は CMIM で選択された特徴のうち分割統治 CMIM でも選択した特徴の割合を示しており，分割統治 CMIM を行って選択した特徴は CMIM で選択された特徴のほとんどを選択できていることがわかる．

並列 CMIM

分割統治 CMIM において表 5.2 の step 2, 3, 4 におけるそれぞれの選択は依存関係がないため，それぞれの選択を個別に実行することで並列化できる．この並列化は LCM と同様ワークキューを用いて実現できる．この分割統治 CMIM を並列化したものを並列 CMIM と呼ぶ．並列 CMIM の詳細については表 5.3 に示した．この並列化についても並列 LCM と同様並列化によって起こるオーバヘッドが存在する．並列 LCM に比べサーバがワークキューを作成する時間は無視できるほど小さい．しかし，並列 LCM と同様集合の分割によってクライアントのタスクの不平等が起こり，表 5.3 における step 3 において同期をとる必要がありこの同期を取る際に暇なプロセッサ存在することになる．このオーバヘッドは分割統治 CMIM と結果が変わる可能性があるものの表 5.3 において step 3 において全ての集合が終わるのを待たずに，処理の終わった集合のみをまとめて分割し，暇なプロセッサで処理をすることで小さくできる．

並列 LCM と同様，図 5.8 に並列 CMIM のサーバの疑似コード，図 5.9 に並列 CMIM のクライアントの疑似コードを示した．ここでは簡単のため LCM

[3] http://www.nipsfsc.ecs.soton.ac.uk/

表 5.3: 並列 CMIM

サーバ

step 1,	パターンをランダムに S 個ずつの集合に分ける
step 2,	それぞれの集合をワークキューに追加する
step 3,	全ての集合についてクライアントが終了したらクライアントから受け取ったパターンをまとめ，そのパターンがある基準に達しない場合には 1 に戻る

クライアント

step 1,	ワークキューからタスクを受け取る
step 2,	タスクに含まれるパターンを用いて訓練データを表現する
step 3,	CMIM を用いて一定値 (打ち切り条件付き相互情報量と呼ぶ) 以上の条件付き相互情報量をもつパターンを選択する
step 4,	ワークキューが空なら終了，空でなければ step 1 に戻る

の結果は統合されており，サーバは結果が集まるまで待つものとした．また，LCM と同様，訓練データは共有しており，通信やロックは適宜なされているものとする．CMIM と LCM において最も大きく異なるところは 9 行目の繰り返しであり，本実装では 5 回繰り返すこととした．また 7 行目と 22 行目に示した通りに事前にパターンをシャッフルすることでパターンの分割がランダムになるようにした．

5.4 評価設定

評価として Othello の勝ち・負け問題に本手法を適用し，Othello の評価関数生成のための評価要素の自動生成を行った．Othello とは Reversi とも呼ばれる日本発のよく知られているボードゲームである．Othello は 8×8 の白黒の面を持ったディスクを使って 2 プレイヤで行うゲームである．プレイヤは白黒それぞれの色に別れ，プレイヤは空のマスに他のプレイヤの色のディスクを自分の色のディスクで挟むように置き，その挟んだディスクを裏返す．そのような挟める場所がない場合はパスをする．両プレイヤがディスクを置けないときにゲームは終わり，自分の色のディスクが多いほうが勝ちとなる．本節では評価に用いた Naïve Bayesian 分類器について説明した後に，特徴要素，データセット，評価環境について説明する．

5.4.1 Naïve Bayesian 分類器

選択した評価要素を評価するために，我々は高速な分類アルゴリズムの 1 つである Naïve Bayesian 分類器 [27] を用いた．

Naïve Bayesian 分類器とは特徴が互いにラベルについて条件付独立な確率変数であるという仮定を置いて評価対象の確率を計算することで分類を行う．条件付独立という仮定が成り立たない場合でも精度の高い分類をできることが経験的に知られている．2 つのクラスについてそれぞれ 0, 1 のラベルが付けられている際の N 個の評価要素 x_i で表された評価対象 $\mathbf{x}$ を Naïve Bayesian classifier で分類する基準 $f(\mathbf{x})$ を式 (5.6) に示す．

$$f(\mathbf{x}) = \sum_{i=1}^{N} \left\{ \log \frac{\hat{P}_{1,1}(x_i)\hat{P}_{0,0}(x_i)}{\hat{P}_{0,1}(x_i)\hat{P}_{1,0}(x_i)} \right\} x_i + b \tag{5.6}$$

ここで $\hat{P}_{\alpha,\beta}(x)$ とは評価要素 x が α (局面に出現する時 1, そうでなければ 0), ラベルが β であるものが訓練局面に含まれる割合である．b は定数であり分類超平面における $f(\mathbf{x})$ が 0 となるように求められる．この基準により $\mathbf{x}$ は $f(\mathbf{x})$ が正であればクラス 1, また負であればクラス 0 と分類される．

5.4.2 評価設定

特徴要素

今回はそれぞれのマスの状態を特徴要素とした．それぞれのマスは「黒がある」「白がある」「空である」の3種類192個の特徴要素で表現した．特徴要素の排他的な条件や局面の対称性などのような，特徴要素以外のゲームの知識は用いなかった．

データセット

評価要素を抽出する局面として200,000局面を用いた．また分類精度を試すための局面として別の962,439局面を用いた．これらの局面は全て60ディスクが載っており，空のマスが4つ残っている局面であり，勝ち・負けのラベルが付いている．このラベルは黒のプレイヤのゲームの結果であり，その局面から全探索を行うことで求めた．これらの局面はGeneric Game Server [17]から得られたものである．

評価環境

評価環境としてはそれぞれのノードがIntel Xeon 2.4GHz dual, 2GB RAMで構成されている50ノードのPCクラスタを用いた．評価にはPythonとC++を用いた．

5.5 評価結果

本節では得られた評価要素とそれを用いた分類結果について示し，考察を行う．

5.5.1 評価要素

まず5.4.2節に示した200,000局面から頻出飽和パターンを並列LCMを用いて選択した．図5.10に最小サポートを変化させた際に得られた頻出飽和パターンの数を示した．頻出飽和パターンは最小サポートが小さくなるにつれ急激に多くなっている．

次に並列CMIMを用いて評価要素を頻出飽和パターンから選択した．図5.11に打ち切り条件付き相互情報量を変化させた場合の評価要素の数を示した．最小サポート4,000の場合，頻出飽和パターンは非常に多くなるため情報利得による事前選択を行った．情報利得が0.7以上のもの (パターンが現れる訓練局面に勝ちもしくは負けの偏りが80%を超えるもの) を選択した．選択に用いた頻出飽和パターンの数は最小サポート6,000については282,615,853, 4,000については172,022,168であった．分割統治CMIMの選択は5回繰り

返した．初めの 4 回は 2,000 パターン，最後の選択では 10,000 パターンの集合に分割した．

並列 LCM において負荷の不平等が起こり暇なプロセッサがでないようにするために LCM の処理が終わったプロセッサは CMIM を実行するようにした．選択には最小サポートが 6,000, 打ち切り条件付き相互情報量 0.001 の時, 49 ノード (98 プロセッサ) を使って約 2 日かかった．この選択は 1 台のプロセッサでは 1 か月では終わらなかった．空間コストについては，LCM での選択では最大で 140MB，CMIM での選択では最大で 250MB で済み，これにより十分に少ない空間コストで選択を実行でき，より大きなデータを扱うことも可能である．

5.5.2 分類結果

図 5.12 に得られた分類器の正解率を示した．評価要素としては図 5.11 に示したものを用いた．それぞれのテストには 5.4.2 節に示した 962,439 局面を用い, 5 分割交差検定を行った．得られた分類器で最も良かったものは正解率 77.2%，適合率 0.773，再現率 0.778，F1 値 0.780 であった．勝ちとラベルづけされた局面が勝ちと予測された数を a・勝ちと予測された数を b・勝ちとラベルづけされた局面を c とすると適合率は a / b・再現率は a / c・F1 値は 2 × (適合率) × (再現率) / ((適合率) + (再現率)) である．

また表 5.4 に特徴要素を用いて同様のデータについて 4 つの分類器で分類を行った結果を示した．この結果より生成した評価要素を用いた Naïve Bayesian 分類器は他の特徴要素を用いた分類器よりも良い結果を示していることがわかる．ここで Naïve Bayesian 分類器と判別分析 [27] については結果が一意に得られるものであるが, 3 層ニューラルネットワークは学習を止める必要があるため, ここには最も結果が良かったものを示した．3 層ニューラルネットワークの学習には RPROP [83] を用いた．これらのことから今回得られた評価要素は特徴要素を直接用いるより分類問題をよりうまく表していることがわかる．また Support Vector Machine (SVM) についても評価を行ったが, 直接解法を用いたもの (LIBSVM [21] を用いて評価を行った) については 1 週間かけても終わらなかった．ここに示したのは確率的最急降下法 (SGD, Stochastic Gradient Descent) [12] を用いた結果である．

5.6 おわりに

本章ではゲームの対戦履歴を元にして評価要素を自動的に生成する効率的で問題の大きさに対してスケーラブルな手法を提案した．この手法は単純な特徴要素の集合を組み合わせて頻度と条件付き相互情報量を用いて選択を行うことで評価要素を作成する．我々は本手法を Othello の勝ち負け問題に適

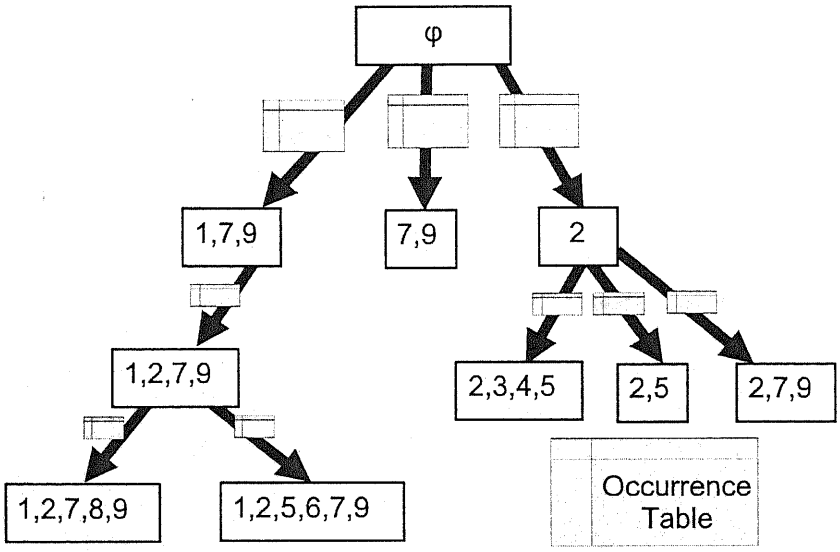

図 5.4: LCM の探索木.

表 5.4: 特徴要素を用いた場合の分類器の正解率

classifier	accuracy
Naïve Bayesian 分類器	73.9%
3 層ニューラルネットワーク	~75.2%
判別分析	73.4%
SVM (Linear, SGD)	74.5%

```
1  //input_file:頻出要素を数え上げるデータ
2  //work_queue:ワークキュー
3  OccurrenceTable table;
4  PrefixList list;
5  int server(File input){
6    //出現表の作成
7    table = OccurrenceTable(input);
8    //prefix 保存拡張のための順番の計算
9    list = table.calc_prefix_list();
10   //空集合のパターン
11   Pattern pattern;
12   //ワークキューにタスクを追加
13   append_work(pattern, 0, table);
14   //タスクが十分になるまで分割する
15   while(!is_sufficient(work_queue.size(), client_size)){
16     task = work_queue.get();
17     split_task(task);
18   }
19   notify_clients();
20   //ワークキューが空になるまで待つ
21   wait();
22 }
23
24 void append_work(Pattern pattern, int index, OccurrentTable current_table){
25   for(int i = index;i < list.size();i++){
26     // list[i] から同時に起こる特徴を 1 つ以上追加する
27     pattern->append_next(list, i);
28     // サーバで出現した頻出飽和パターン
29     if(is_closed(pattern, current_table)){
30       output(pattern);
31     }
32     work_queue.append(new task(pattern));
33     // 最後に追加したパターンを削除する
34     pattern->remove_last();
35   }
36 }
37
38 void split_task(task){
39   Pattern current_pattern = task.get_pattern();
40   // 最後に追加したパターン
41   int start = pattern->last_index();
42   // 出現表を更新する
43   OccurrenceTable current_table = table.update(current_pattern);
44   append_work(current_pattern, start, current_table);
45 }
46
```

図 5.5: 並列 LCM のサーバの疑似コード

```
1  //input_file:頻出要素を数え上げるデータ
2  //work_queue:ワークキュー
3  OccurrenceTable table;
4  PrefixList list;
5  int client(File input){
6    //出現表の作成
7    table = construct_occurrence_table(input);
8    //prefix 保存拡張のための順番の計算
9    list = table.calc_prefix_list();
10   //空集合のパターン
11   Pattern pattern;
12   while(work_queue.size() != 0){
13     //ワークキューからタスクをもらい LCM を実行
14     task = work_queue.get();
15     run_lcm(task->pattern());
16   }
17 }
18
19 void run_lcm(task){
20   Pattern current_pattern = task.get_pattern();
21   // 最後に追加したパターン
22   int start = pattern->last_index();
23   // 出現表を更新する
24   OccurrenceTable current_table = table.update(current_pattern);
25   LCM(current_pattern, start, current_table);
26 }
27
```

図 5.6: 並列 LCM のクライアントの疑似コード

用し評価要素を生成した．192 個の特徴要素から数千の評価要素を生成し，その評価要素を使うことで分類精度 77.2%の Naïve Bayesian 分類器を作成できた．この結果は特徴要素を使った 4 つの分類器よりも良い結果が得られた．他の特徴抽出アルゴリズムでは扱えないような大きな問題も，提案した問題を分割するアルゴリズムを用いることで適度な空間コストで扱うことができ，その分割した問題を並列に処理することにより高速に処理することもできた．また分類精度を向上できる重要な要素の発見もできた．この評価要素は人手で評価関数を作成する開発者へのヒントとしても利用可能であろう．このような結果から特徴要素という単純な語彙を局面における共起関係に基づいて拡張することの有用性がわかった．

今後の課題としては提案した特徴選択手法の改善がある．近似解法や GLEM の *pattern* [16] などが有用であると考えられる．GLEM の *pattern* を Othello 以外のゲームにあてはめる一般的な手法はなく，特徴要素を自動的に分類する手法を構築する必要がある．また将棋や囲碁などより難しいと言われる他のゲームへの適用も課題の一つである．我々はこれまでに並列化を行わないで訓練例の分割を行う手法において将棋の詰み問題に適応できること [76] を示している．今回の並列化により，より大きな問題に適用でき，Othello の終盤のような比較的簡単な問題に限らず，多くの難しいと言われているゲームにも適用できる．

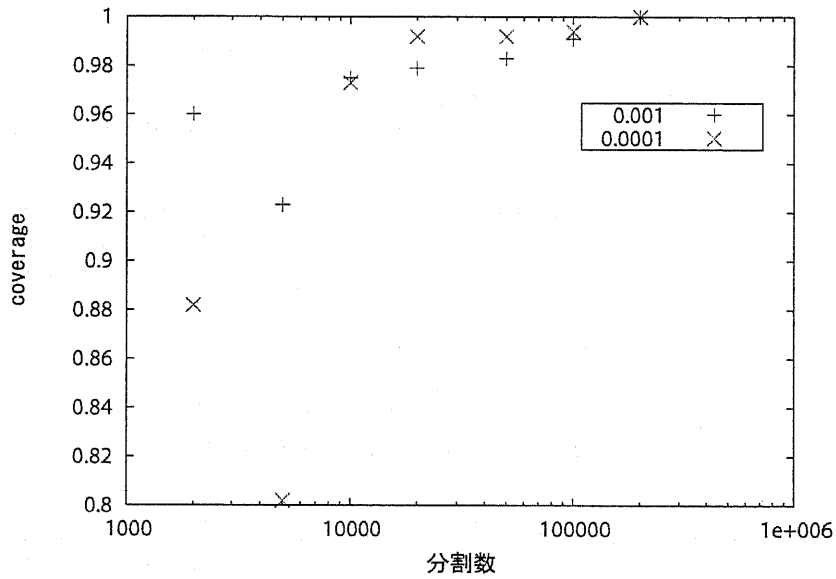

図 5.7: Dorothea における CMIM で選択された特徴と分割統治 CMIM で選択された特徴の相違.

```
1  //pattern_file:選択前の頻出飽和パターン
2  //work_queue:ワークキュー
3  int server(File pattern_file){
4    //最初のパターン
5    Pattern pattern(pattern_file);
6    //パターンをランダムに分割
7    pattern->randomize();
8    //パターンが十分な基準を満たしていれば終了
9    while(!is_sufficient(pattern)){
10     int size = pattern->split(s);
11     //ワークキューに分割したパターン集合を追加
12     for(int i = 0;i < size;i++){
13       work_queue.append(pattern->subset(i));
14     }
15     //クライアントに開始を通知
16     notify_clients();
17     pattern->clear();
18     //クライアントの結果をランダムに統合
19     while(work_queue.size() != 0){
20       wait_client();
21       Pattern client_pattern = receive_result();
22       pattern->merge_random(client_pattern);
23     }
24   }
25 }
26
```

図 5.8: 並列 CMIM のサーバの疑似コード

```
1  //input:特徴要素で表現されたパターンを評価するラベル付き訓練データ
2  //work_queue:ワークキュー
3  //cutoff_mutual_info:打ち切り条件付き相互情報量
4  int client(File input){
5    Data data(input);
6    while(work_queue != 0){
7        //ワークキューからタスクをもらい CMIM を実行
8      task = work_queue.get();
9      Pattern pattern = task->pattern();
10     Data data = data->translate(pattern);
11     Pattern result = CMIM(data, cutoff_mutual_info);
12     send_result(result);
13   }
14 }
15
```

図 5.9: 並列 CMIM のクライアントの疑似コード

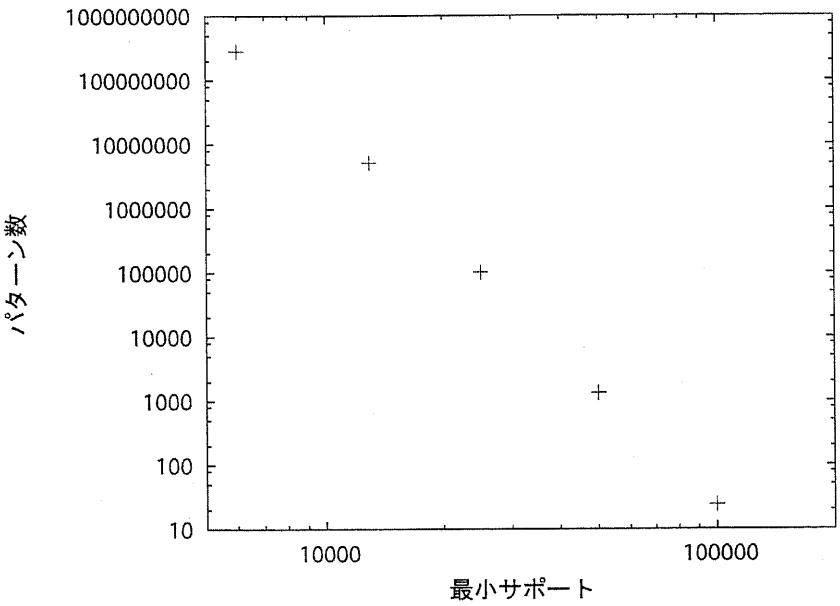

図 5.10: 様々な最小サポートにおける 200,000 局面から選択した頻出飽和パターンの数

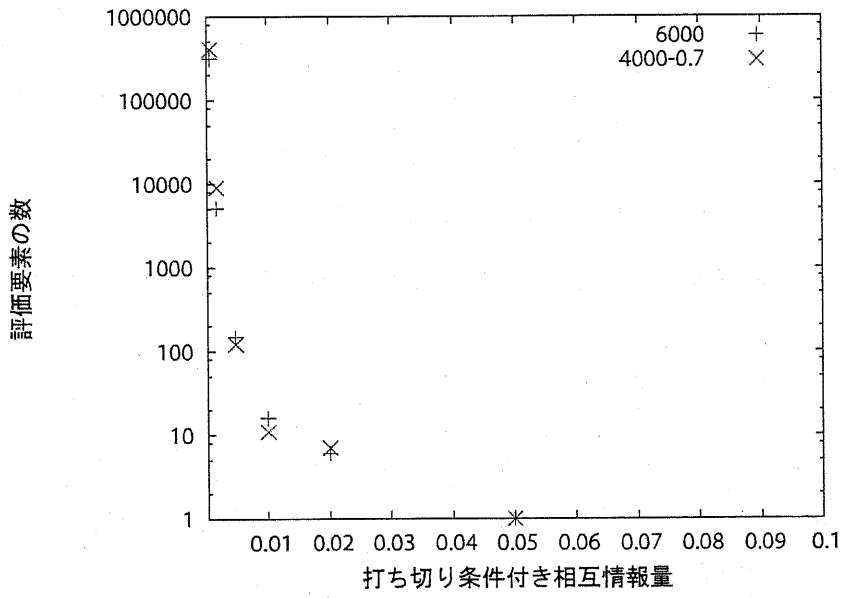

図 5.11: CMIM における打ち切り条件付き相互情報量を変化させた場合の特徴要素の数

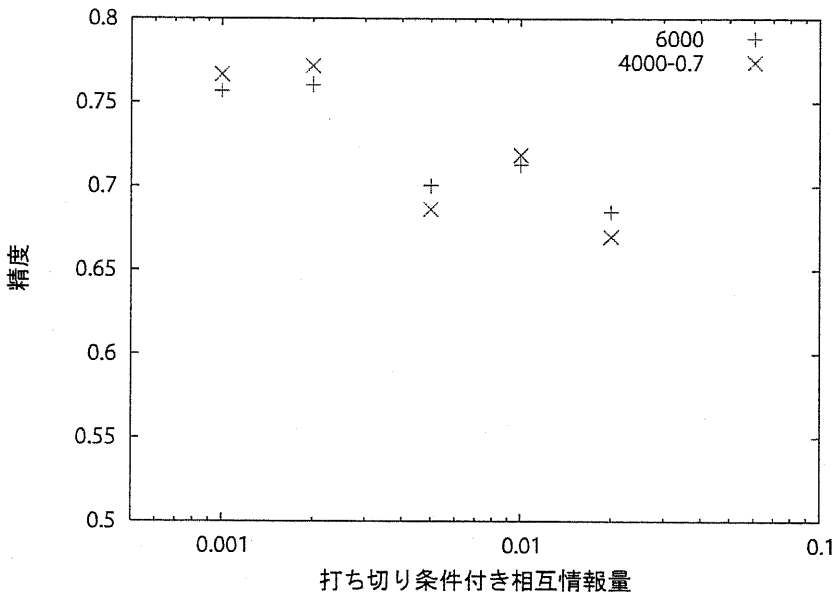

図 5.12: 選択した評価要素を用いた際の Naïve Bayesian 分類器の正解率

第6章　おわりに

6.1　本論文のまとめ

コンピュータゲームプレイヤを作成する際，一般的な探索手法は多く提案され利用されているが，対象とするゲームに関する知識を記述する語彙の獲得には一般的な手法はない．このため一般的な機械学習などの手法で知識を獲得する手法が多く提案されているにもかかわらず，語彙があるという前提があるために他のゲームに容易にすることが困難となっており，対象ゲームの異なる研究における大きな壁となっている．また，コンピュータゲームプレイヤを一般的な手法で作成することはコンピュータゲームプレイヤの研究における１つの大きな課題であり，そのような点においても一般的な手法での語彙の獲得は必要なものである．さらに現在利用されている語彙はそのほとんどが人間に知識を基に選択されたものである．そのためこのような語彙は対象に関する知識の偏りなどにより，その過不足があることは明らかである．このような過不足の問題を解決もしくは緩和するためにも，一定の基準での語彙の獲得は必要なものである．

本論文ではこのような一般的な手法での語彙の獲得を目指して，ゲーム知識の表現語彙の棋譜データからの自動獲得を目的として単純な語彙から棋譜データにおける共起をもとに語彙を拡張する手法について提案を行った．この共起関係として本論文では次の２つを対象に提案，評価を行った．

- 局面 (指し手) 間に時系列的に起こる共起
- 局面内に同時に起こる共起

時系列的に共起する既存の語彙から新たな語彙を獲得する手法を第４章の指し手列の解析に基づく将棋における指し手の分類精密化において提案した．この手法ではコンピュータ将棋プログラム「激指」に実装されている指し手をその機能ごとに分けたカテゴリを基本語彙とし，カテゴリの時系列的な並びを n-gram を用いて抽出した．その抽出したカテゴリの時系列的な並びをカテゴリの履歴として，その履歴をもとに実現確率探索におけるカテゴリの遷移確率をカテゴリの履歴の情報を含んだカテゴリの遷移確率として拡張することを試みた．評価として得られた結果としては，抽出したカテゴリの履歴を用いて遷移確率を拡張したコンピュータ将棋プレイヤと用いない元のプ

レイヤで対戦を行い150戦中83勝67敗と勝ち越すことができた．また問題集についても元のプレイヤよりも多くの問題を正解することが可能となった．

また，局面に同時に現れる語彙から新たな語彙を発見する手法を第5章のコンピュータゲームプレイヤにおける評価要素の自動生成において提案した．この手法ではゲームの棋譜に基づいた高速でスケーラブルな評価要素の自動生成手法を提案した．この手法では勝ち・負けや詰み・不詰など多くのゲームに応用可能な2クラスの分類問題を対象問題とした評価要素を2値の単純な特徴の論理積で表現し，頻度とクラスとの条件付き相互情報量の2つの指標を用いて選択する手法について提案した．評価としてOthelloの位置と色の192個の単純な特徴から評価要素の作成を行うことで得られた結果としては分割処理によって大きな問題を扱うことができ，並列化により高速に実行できることを確認できた．また，生成した評価要素を用いたNaïve Bayesian分類器は他の分類器による単純な特徴を用いた分類よりも良い精度で分類できることを示すことができた．

このような結果から，本論文で提案した共起に基づく知識獲得を用いる利点は大きく次のようなものがあげられる．

棋譜データからの語彙獲得 コンピュータゲームプレイヤにおいて過去の経験として代表的なゲームの記録を用いて既存の語彙を拡張することできた．またその語彙を利用した結果，既存の語彙を使って得られた知識よりもより良い知識が得られており，その語彙の有用性を示すことができた．

一般的な指標による語彙の拡張 今回得られた知識のいくらかは人間が既存の語彙を組み合わせることにより人手で取捨選択されているものである．そのため本手法を用いることで人手でのコストを削減することができる．またさらに人手での語彙の拡張を行うことで人手での選択において起こりやすい知識の偏りの問題に対し，本手法ではそのような問題が起こりづらい統一的な手法で語彙の拡張を行うことができる．

多くのゲームへの利用 本手法は基本的な語彙は必要であるものの，それ以外の対象ゲームへの知識を必要としないで語彙を獲得できる手法であり，広く多くのゲームに適用できる．また，多くの棋譜・多くの特徴要素でも扱えるような手法を提案したことでこれまでそのコストのため生成が難しいといわれていたゲームにも適用できると考えられる．

強いコンピュータゲームプレイヤの作成には対象ゲームへの知識を上手く表現する必要がある．本論文で目的とした知識を表現するための語彙の獲得はこのような知識表現を得るためには必要なものである．そして一般的な強いコンピュータゲームプレイヤを作成する手法を提案するには，一般的な方法でこのような語彙を獲得する必要がある．本研究では大規模の棋譜・パターン数を扱いながら対象への知識を必要とせずにそのような語彙を発見するこ

とができており，このような一般的な手法で語彙の獲得ができたことは，一般的な手法でのコンピュータゲームプレイヤの作成へ貢献することができたということができる．

6.2 今後の展望

6.2.1 一般的な問題に対する語彙の獲得

本論文では一般的な方法での語彙の取得方法について，棋譜データにおける共起関係を元に語彙の獲得を行った．今回は指し手の履歴の一般化，2値の分類問題を対象としてその語彙の獲得を行った．

本研究の更なる課題としてはより一般的な問題に対する語彙の獲得があげられる．知識の利用が提案されている問題対象によっては，これらの手法のみではカバーできないものが多く存在する．このような問題対象として特に多いものには連続値を扱う回帰の問題，順位関係の学習に関する問題，強いプレイヤの棋譜が容易に得られないゲームの問題，があげられる．これらの全ての問題対象についてそれを一括して扱うことのできるゲーム一般を扱う語彙の獲得は非常に難しい問題であり現実的ではない．

このような問題に対する現実的な解の１つとしては，それぞれの知識を獲得する対象となる問題を抽象化・一般化し，それぞれについて一般化した手法での解決策を求めることであろう．このような問題の一般化により，その一般化の過程で様々な対象ゲームについて共通した知識，またその共通した利用法を見つけることができればコンピュータゲームプレイヤの研究への大きな貢献となると考えられる．また，それとは逆に獲得している知識の違い，利用法の違いなど一般化が難しいものがあるような場合には，その違いを調べることでゲームの一般化できる部分・そうでない部分の切り分けやチェスライクなゲームなどといったゲームのカテゴリ分けを一般化する切り口となると考えられる．そのようなことができればこちらについてもコンピュータゲームプレイヤの研究への大きな貢献となる．このように知識獲得対象の問題の一般化はコンピュータゲームプレイヤ研究の多くの課題を含んでいると共に，これからも多くの研究者が労力をかけて研究するに値する課題である．このような一般化された問題に対する語彙の獲得ができれば，多くのゲームにおいて別々に提案されていた手法をまとめることができる．またこのようなことが出来ることで初めて，これまで多くの研究が目的としてきた一般化された手法での知識の獲得が可能となる．

6.2.2 一般的な手法でのコンピュータゲームプレイヤの作成

本論文で提案した一般的な方法での語彙の取得は，知識獲得手法についての１つの提案である．そしてこの知識獲得手法はコンピュータプレイヤの一

般的な方法での作成という課題における１つの問題である．

一般的な手法でのコンピュータゲームプレイヤについてはその対象ゲームや強さによっては現在の技術によって可能であるが，この現在の一般的な手法で作成されたコンピュータゲームプレイヤの強さは満足できるものではないということについてはほとんどのコンピュータゲームプレイヤ研究者が知識を利用してより強いプレイヤを目指していることからも明らかである．今のところこの一般的な手法でのコンピュータゲームプレイヤという課題の定義について決まったものはない．ただし，多くのゲームについてはこれまで人間の知識を元にした強いプレイヤが存在しており，このようなプレイヤをリファレンスプレイヤ (強さを測るプレイヤ) としてその手法の有用性を測ることは可能であると考えられる．

このような一定の強さを持ったプレイヤの一般的な方法での作成には，現在のコンピュータゲームプレイヤ作成において用いられている手法について，それらの手法の組み合わせを考慮しつつ，それらの手法の一般化を図ることが，現在までのコンピュータゲームプレイヤの研究を利用できる点からも最良の方法であると考えられる．このような手法としては探索手法，知識獲得手法，知識利用手法が挙げられる．そのプレイヤを実際にプログラムとして実現する際にも多く問題は存在する．

このように一般的な方法でのコンピュータゲームプレイヤの作成にはまだまだ多くの課題が残っているが，本研究において対象とした知識獲得手法はその一般化が一番遅れている部分であり，この手法を一般化する今回の研究をより発展させていくことがこの大きな課題への解決の近道であると考えている．

参考文献

[1] Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 4, pp. 491–502, 2005. Senior Member-Huan Liu and Student Member-Lei Yu.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining Association Rules between Sets of Items in Large Databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pp. 207–216, Washington, D.C., May 1993.

[3] Ingo Althöfer. Computer-aided game inventing. October 2003.

[4] T. Anantharaman, M.S. Campbell, and F. Hsu. Singular Extensions: Adding Selectivity to Brute-Force Searching. *Artificial Intelligence*, Vol. 43, pp. 99–109, 1990.

[5] Matej Artač, Matjaž Jogan, and Aleš Leonardis. Incremental pca for on-line visual learning and recognition, 2002.

[6] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, Vol. 12, No. 10, pp. 2385–2404, 2000.

[7] Jonathan Baxter, Andrew Trigdell, and Lex Weaver. Knightcap: a chess program that learns by combining TD(λ) with game-tree search. In *Proc. 15th International Conf. on Machine Learning*, pp. 28–36. Morgan Kaufmann, San Francisco, CA, 1998.

[8] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation, 2002.

[9] Klaus-Robert Müller Bernhard Schölkopf, Alexander Smola. Nonlinear component analysis as a kernel eigenvalue problem, 1998.

[10] Christopher M. Bishop, Markus Svensen, and Christopher K. I. Williams. Gtm: The generative topographic mapping. *Neural Computation*, Vol. 10, No. 1, pp. 215–234, 1998.

[11] Yngvi Björnsson and T. Anthony Marsland. Learning extension parameters in game-tree search. *Inf. Sci.*, Vol. 154, No. 3-4, pp. 95–118, 2003.

[12] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems*, Vol. 20. MIT Press, Cambridge, MA, 2008. to appear.

[13] Mark G. Brockington and Jonathan Schaeffer. APHID: Asynchronous parallel game-tree search. *Journal of Parallel and Distributed Computing*, Vol. 60, No. 2, pp. 247–273, 2000.

[14] Michael Buro. ProbCut: An effective selective extension of the alpha-beta algorithm. *ICCA Journal*, Vol. 18, No. 2, pp. 71–76, 1995.

[15] Michael Buro. Toward opening book learning. In H. Iida, J. Schaeffer, J. W. H. M. Uiterwijk, and Y. Saito, editors, *Proceedings of the IJCAI-97 Workshop on Using Games as an Experimental Testbed for AI Research*, Nagoya, Japan, 1997.

[16] Michael Buro. From Simple Features to Sophisticated Evaluation Functions. In H. J. van den Herik and H. Iida, editors, *Proceedings of the First International Conference on Computers and Games (CG-98)*, Vol. 1558, pp. 126–145, Tsukuba, Japan, 1998. Springer-Verlag.

[17] Michael Buro and Igor Durdanovic. An overview of neci's generic game server, 2001.

[18] Murray Campbell. Knowledge discovery in deep blue. *Communications of the ACM*, Vol. 42, No. 11, pp. 65–67, 1999.

[19] Bin Cao, Dou Shen, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Feature selection in kernel space. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, Corvallis, Oregon, 2007. AAAI Press.

[20] M. Á. Carreira-Perpiñán. *Continuous latent variable models for dimensionality reduction and sequential data reconstruction.* PhD thesis, Dept. of Computer Science, University of Sheffield, UK, 2001.

[21] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[22] Kumar Chellapilla and David B. Fogel. Evolving an Expert Checkers Playing Program Without Using Human Expertise. *IEEE Transactions on Evolutionary Computation*, Vol. 5, No. 4, pp. 422–428, 2001.

[23] Tat-Jun Chin and David Suter. Incremental kernel principal component analysis. *IEEE Transactions on Image Processing*, Vol. 16, No. 6, pp. 1662–1674, June 2007.

[24] Jan de Leeuw. Applications of convex analysis to multidimensional scaling. In J.R. Barra, F. Brodeau, G. Romier, and B. Van Cutsem, editors, *Recent Developments in Statistics*, pp. 133–146, Amsterdam, 1977. North Holland Publishing Company.

[25] Chrilly Donninger and Ulf Lorenz. Innovative opening-book handling. In *ACG*, pp. 1–10, 2006.

[26] D. Donoho and C. Grimes. Hessian eigenmaps: locally linear embedding techniques for high dimensional data. *Proc. of National Academy of Sciences*, Vol. 100, No. 10, pp. 5591–5596, 2003.

[27] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd ed.)*. Wiley Interscience, 2000.

[28] W. H. Duminy and Andries Petrus Engelbrecht. Composing linear evaluation functions from observable features. *South African Computer Journal*, Vol. 35, pp. 48–58, 2005.

[29] Susana Eyheramendy and David Madigan. A novel feature selection score for text categorization. In *Proceedings of the International Workshop on Feature Selection for Data Mining: Interfacing Machine Learning and Statistics (in conjunction with 2005 SIAM International Conference on Data Mining)*, Newport Beach, CA., April 2005.

[30] Haw-ren Fang. The nature of retrograde analysis for chinese chess. Technical Report CS-TR-4645, UMIACS-TR-2004-86, UM Computer Science Department, jan 2006.

[31] Tom Elliott Fawcett. *Feature Discovery for Problem Solving Systems*. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, MA, 1993.

[32] U. M. Fayyad and BI Keki. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *IJCAI-93*, pp. 1022–1027, 1993.

[33] Rainer Feldmann, Peter Mysliwietz, and Burkhard Monien. Game tree search on a massively parallel system. In H. J. van den Herik, I. S. Herschberg, and J. W. H. M. Uiterwijk, editors, *Advances in Computer Chess7*, pp. 203–218, Masstricht, The Netherlands, 1994. University of Limburg.

[34] François Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research (JMLR)*, pp. 1531–1555, May 2004.

[35] David B. Fogel. *Blondie24: Playing at the Edge of AI.* Morgan Kaufmann, September 2001.

[36] David B. Fogel, Timothy J. Hays, Sarah L. Hahn, and James Quon. A self-learning evolutionary chess program. In *Proceedings of the IEEE*, Vol. 92, pp. 1947–1954, 2004.

[37] N. Franken and Ap Engelbrecht. Comparing pso structures to learn the game of checkers from zero knowledge. In Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, and Tom Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pp. 234–241, Canberra, December 2003. IEEE Press.

[38] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, Vol. 23, No. 9, pp. 881–890, 1974.

[39] Johannes Fürnkranz. Machine Learning in Games: A Survey. In J. Fürnkranz and M. Kubat, editors, *Machines that Learn to Play Games*, chapter 2, pp. 11–59. Nova Science Publishers, Huntington, NY, 2001.

[40] Johannes Fürnkranz and Eyke Hüllermeier. Preference learning. *KI*, Vol. 19, No. 1, pp. 60–, 2005.

[41] Tesauro G. Programming backgammon using self-teaching neural nets. *Artificial Intelligence*, Vol. 134, pp. 181–199, January 2002.

[42] Sylvain Gelly and David Silver. Combining online and offline knowledge in uct. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pp. 273–280, New York, NY, USA, 2007. ACM.

[43] D. Gomboc, T. A. Marsland, and M. Buro. Evaluation fuction tuning via ordinal correlation. In Heinz van den Herik, Iida, editor, *Advances in Computer Games*, pp. 1–18. Kluwer, 2003.

[44] Peter M. Hall, David Marshall, and Ralph R. Martin. Incremental eigenanalysis for classification. 1998.

[45] Peter M. Hall, David Marshall, and Ralph R. Martin. Merging and splitting eigenspace models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 9, No. 9, pp. 1042–1049, September 2000.

[46] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In Weidong Chen, Jeffrey Naughton, and Philip A. Bernstein, editors, *2000 ACM SIGMOD Intl. Conference on Management of Data*, pp. 1–12. ACM Press, May 2000.

[47] X. He and P. Niyogi. Locality preserving projections, October 2002.

[48] E.A. Heinz. Extended futility pruning. *ICCA Journal*, Vol. 21, No. 2, pp. 75–83, 1998.

[49] S. Hettich, C.L. Blake, and C.J. Merz. UCI Repository of machine learning databases, 1998.

[50] G. E. Hinton and R. Salakhutdinov. Non-linear dimensionality reduction using neural networks. *Science*, Vol. 313, pp. 504–507, jul 2006.

[51] Kunihito Hoki. Optimal control of minmax search results to learn positional evaluation. In *The 11th Game Programming Workshop (GPW 2006)*, pp. 78–83, 2006. (In. Japanese).

[52] P. Howland and H. Park. Generalizing discriminant analysis using the generalized singular value decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 8, pp. 995–1006, August 2004.

[53] Gordon F. Hughes. On the mean accuracy of statistical pattern recognition. *IEEE Transaction on Information Theory*, Vol. 14, No. 1, pp. 55–63, January 1968.

[54] R.M. Hyatt. Book Learning - A Methodology to Tune an Opening Book Automatically. *ICCA Journal*, Vol. 22, No. 1, pp. 3–12, March 1999.

[55] Aapo Hyvärinen. A survey on independent component analysis. *Neural Computing Surveys*, Vol. 2, pp. 94–124, 1999.

[56] T. Kaneko, K. Yamaguchi, and S. Kawai. Automated Identification of Patterns in Evaluation Functions. In H. J. van den Herik, H. Iida, and E. A. Heinz, editors, *Advances in Computer Games 10*, pp. 279–298. Kluwer Academic Publishers, 2004.

[57] Akihiro Kishimoto. Transposition table driven scheduling for two-player games. Ms thesis, University of Alberta, Edmonton, Canada, 2002.

[58] Levante Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European Conference on Machine Learning (ECML)*, pp. 282–293, 2006.

[59] Levente Kocsis, Jos W. H. M. Uiterwijk, Eric O. Postma, and H. Jaap van den Herik. The neural movemap heuristic in chess. In *Computers and Games*, pp. 154–170, 2002.

[60] Levente Kocsis, Jos W. H. M. Uiterwijk, and H. Jaap van den Herik. Move ordering using neural networks. In *IEA/AIE '01: Proceedings of the 14th International conference on Industrial and engineering applications of artificial intelligence and expert systems*, pp. 45–50, London, UK, 2001. Springer-Verlag.

[61] Ron Kohavi and George H. John. Wrappers for feature subset selection. 1997.

[62] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, Vol. 97, No. 1-2, pp. 273–324, 1997.

[63] Teuvo Kohonen. *Self-Organizing Maps*, Vol. 30 of *Springer Series in Information Sciences*. Springer, third extended edition edition, 2001.

[64] Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse Processes*, Vol. 25, pp. 259–284, 1998.

[65] Haifeng Li, Tao Jiang, and Keshu Zhang. Efficient and robust feature extraction by maximum margin criterion. 2004.

[66] H. Liu and R. Setiono. Feature selection and classification - a probabilistic wrapper approach.

[67] J. Loughrey and P. Cunningham. Overfitting in wrapper-based feature subset selection: the harder you try the worse it gets. In *the Twenty-fourth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, 2004.

[68] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing.* The MIT Press, Cambridge, Massachusetts., 1999.

[69] Shaul Markovitch and Danny Rosenstein. Feature generation using general constructor functions. *Machine Learning*, Vol. 49, pp. 59–98, 2002.

[70] T. A. Marsland. Evaluation function factors. *Journal of the International Computer Chess Association*, Vol. 8, No. 2, pp. 47–57, 1985.

[71] Aleix M. Martínez and Avinash C. Kak. Pca versus lda. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 2, pp. 228–233, February 2001.

[72] D.A. McAllester. Conspiracy Numbers for Min-Max search. *Artificial Intelligence*, Vol. 35, pp. 287–310, 1988.

[73] L. Messerschmidt and Andries Petrus Engelbrecht. Learning to play games using a pso-based competitive learning approach. *IEEE Trans. Evolutionary Computation*, Vol. 8, No. 3, pp. 280–288, 2004.

[74] Patrick Emmanuel Meyer and Gianluca Bontempi. On the use of variable complementarity for feature selection in cancer classification. In Franz Rothlauf, Jürgen Branke, Stefano Cagnoni, Ernesto Costa, Carlos Cotta, Rolf Drechsler, Evelyne Lutton, Penousal Machado, Jason H. Moore, Juan Romero, George D. Smith, Giovanni Squillero, and Hideyuki Takagi, editors, *EvoWorkshops*, Vol. 3907 of *Lecture Notes in Computer Science*, pp. 91–102. Springer, 2006.

[75] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Muller. Fisher discriminant analysis with kernels, 1999.

[76] Makoto Miwa, Daisaku Yokoyama, and Takashi Chikayama. Automatic construction of static evaluation functions for computer game players. In Ljup*breve*co Todorovski, Nada Lavrač, and Klaus P. Jantke, editors, *Proceedings of the 9th International Conference on Discovery Science*, pp. 332–336. Springer, October 2006.

[77] A. Nagai and H. Imai. Proof for the Equivalence Between Some Best-First Algorithms and Depth-First Algorithms for AND/OR Trees. *IEICE Trans. Inform. & Systems*, Vol. E85-D, No. 10, pp. 1645–1653, 2002.

[78] Shaoning Pang, Seiichi Ozawa, , and Nikola Kasabov. Incremental linear discriminant analysis for classification of data streams. *IEEE Trans. on Systems, Man, and Cybernetics - Part B*, Vol. 35, No. 5, pp. 905–914, October 2005.

[79] Haesun Park, Moongu Jeon, , and J.Ben Rosen. Lower dimensional representation of text data based on centroids and least squares. *BIT*, Vol. 43, No. 2, pp. 1–22, February 2003.

[80] Hanchuan Peng, Fuhui Long, and Chris H. Q. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 27, No. 8, pp. 1226–1238, 2005.

[81] Aske Plaat. *Research Re: search & Re-search.* PhD thesis, Rotterdam, Netherlands, 1996.

[82] Coulom R. Computing elo ratings of move patterns in the game of go. In *Computer Games Workshop*, 2007.

[83] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. of the IEEE Intl. Conf. on Neural Networks*, pp. 586–591, San Francisco, CA, 1993.

[84] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, Vol. 290, No. 5500, pp. 2323–2326, December 2000.

[85] Arthur L. Samuel. Some Studies in Machine Learning Using the Game of Checkers. II — Recent Progress. *IBM Journal of Research and Development*, Vol. 11, No. 6, pp. 601–617, 1967.

[86] J. Schaeffer. The History Heuristic and Alpha-Beta Search Enhancements in Practice. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 11, pp. 1203–1212, 1989.

[87] Jonathan Schaeffer, Neil Burch, Yngvi Bjornsson, Akihiro Kishimoto, Martin Muller, Robert Lake, Paul Lu, and Steve Sutphen. Checkers is solved. *Science*, pp. 1144079+, jul 2007.

[88] David Silver, Richard S. Sutton, and Martin Müller:. Reinforcement learning of local shape in the game of go. In *20th International Joint Conference on Artificial Intelligence*, pp. 1053–1058, 2007.

[89] Danijel Skocaj, Martina Uray, Ales Leonardis, and Horst Bischof. Why to combine reconstructive and discriminative information for incremental subspace learning. In *Proceedings of the Computer Vision Winter Workshop 2006*. Czech Society for Cybernetics and Informatics (Czech Pattern Recognition Society group), Ondrej Chum and Vojtech Franc, February 2006.

[90] Le Song, Alex Smola, Arthur Gretton, Karsten M. Borgwardt, and Justin Bedo. Supervised feature selection via dependence estimation. In Zoubin Ghahramani, editor, *ICML*, Vol. 227 of *ACM International Conference Proceeding Series*, pp. 823–830. ACM, 2007.

[91] Omid David Tabibi and Nathan S. Netanyahu. Verified null-move pruning. Technical Report CAR-TR-980, CS-TR-4406, UMIACS-TR-2002-39, Center for Automation Research, University of Maryland, 2002. Published in ICGA Journal, Vol. 25, No. 3, pp. 153–161.

[92] Shogo Takeuchi, Tomoyuki Kaneko, Kazunori Yamaguchi, and Satoru Kawai. Visualization and adjustment of evaluation functions based on evaluation values and win probability. In *AAAI*, pp. 858–863, 2007.

[93] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, Vol. 290, No. 5500, pp. 2319–2323, December 2000.

[94] Gerald Tesauro. TD-Gammon, A Self-teaching Backgammon Program, Achieves Master-Level Play. In *AAAI Press Technical Report FS93-02*, pp. 19–23, 1993.

[95] Gerald Tesauro. Comparison training of chess evaluation functions. pp. 117–130, 2001.

[96] Ken Thomson. Retrograde analysis of certain endgames. *ICCA Journal*, Vol. 9, No. 3, pp. 131–139, 1986.

[97] Konstantinos Tournavitis. Mouse(μ): A self-teaching algorithm that achieved master-strength at othello. In *Computers and Games*, pp. 11–28, 2002.

[98] Kostas Tournavitis. Herakles. available at `http://www.herakles.tournavitis.de/`.

[99] Yoshimasa Tsuruoka, Daisaku Yokoyama, and Takashi Chikayama. Game-tree Search Algorithm based on Realization Probability. *ICGA Journal*, Vol. 25, No. 3, pp. 145–152, 2002.

[100] T. Uno, M. Kiyomi, and H. Arimura. LCM ver 3.: Collaboration of Array, Bitmap and Prefix Tree for Frequent Itemset Mining. In *Open Source Data Mining Workshop on Frequent Pattern Mining Implementations*, Chicago, IL, August 2005.

[101] Paul E. Utgoff and Jeffery Clouse. Two kinds of training information for evaluation function learning. In *Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-91)*, pp. 596–600, Anaheim, CA, 1991. AAAI Press.

[102] Paul E. Utgoff and Doina Precup. Constructive Function Approximation. In H. Liu and H. Motoda, editors, *Feature Extraction, Construction and Selection: A Data Mining Perspective*, Vol. 453 of *The Kluwer International Series in Engineering and Computer Science*, chapter 14. Kluwer Academic Publishers, 1998.

[103] L. J. P. van der Maaten, E. O. Postma, and H. J. van den Herik. Dimensionality reduction: A comparative review. 2007.

[104] Juyang Weng, Yilu Zhang, and Wey-Shiuan Hwang. Candid covariance-free incremental principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 8, pp. 1034–1040, August 2003.

[105] M.H.M. Winands. *Informed Search in Complex Games*. PhD thesis, Universiteit Maastricht, Maastricht, The Netherlands., 2004.

[106] Jun Yan, Ning Liu, Benyu Zhang, Shuicheng Yan, Zheng Chen, Qiansheng Cheng, Weiguo Fan, and Wei-Ying Ma. OCFS: Optimal orthogonal centroid feature selection for text categorization. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 122–129, Salvador, Brazil, August 2005. ACM Press.

[107] Jun Yan, Benyu Zhang, Ning Liu, Shuicheng Yan, Qiansheng Cheng, Weiguo Fan, Qiang Yang, Wensi Xi, and Zheng Chen. Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 18, No. 3, pp. 320–333, March 2006.

[108] Jun Yan, Benyu Zhang, Shuicheng Yan, Qiang Yang, Hua Li, Zheng Chen, Wensi Xi, Weiguo Fan, Wei ying Ma, and Qiansheng Cheng. IMMC: Incremental maximum margin criterion. 2004.

[109] Jieping Ye, Qi Li, Hui Xiong, and Ravi Janardan. Idr/qr: An incremental dimension reduction algorithm via qr decomposition. 2004.

[110] Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of ICML*, pp. 856–863, 2003.

[111] Y Zhang and P I Rockett. Domain-independent approaches to optimise feature extraction for multi- classification using multi-objective genetic programming. Technical report, VIE, 2007.

[112] 梶山貴司, 中村貞吾. 囲碁の着手符号列に対する確率文法モデルの作成. ゲームプログラミング ワークショップ '99, pp. 161–168, October 1999.

[113] 山下宏. YSS-『コンピュータ将棋の進歩2』以降の改良点. 松原仁（編）, アマトップクラスに迫る コンピュータ将棋の進歩 5, pp. 1–32. 共立出版, December 2005.

[114] 松原仁（編）. コンピュータ将棋の進歩 2. 共立出版, May 1998.

[115] 石井健一郎, 上田修功, 前田英作, 村瀬洋. わかりやすいパターン認識. オーム社, 1998.

[116] 大槻知史. n-gram 統計からの「必然手」の抽出. 第 10 回 ゲームプログラミング ワークショップ 2005, pp. 89–96, November 2005.

[117] 竹歳正史, 橋本剛, 梶原羊一郎, 長嶋淳, 飯田弘之. コンピュータ将棋における実現確率探索の研究. 第 7 回 ゲームプログラミング ワークショップ 2002, pp. 87–92, November 2002.

[118] 中村貞吾. n-gram 統計を用いた棋譜データベースからの定型手順の獲得. ゲームプログラミング ワークショップ '97, pp. 96–105, October 1997.

[119] 鶴岡慶雅. 将棋プログラム「激指」. 松原仁（編）, アマ 4 段を超えるコンピュータ将棋の進歩 4, pp. 1–17. 共立出版, July 2003.

[120] 鶴岡慶雅, 横山大作, 丸山孝志, 近山隆. 局面の実現確率に基づくゲーム木探索アルゴリズム. 第 6 回 ゲームプログラミング ワークショップ 2001, pp. 17–24, October 2001.

[121] 有岡雅章. 将棋プログラム KFEnd における探索. 松原仁（編）, アマ 4 段を超える コンピュータ将棋の進歩 4, pp. 18–40. 共立出版, July 2003.

本論文に関連する発表文献

査読付論文

[1] 三輪誠, 横山大作, 近山隆. コンピュータゲームプレイヤにおける評価要素の自動生成に関する研究. 情報処理学会論文誌, Vol.48, No.11, November 2007.

査読付会議論文

[1] 三輪誠, 横山大作, 近山隆. 駒位置と効き関係に注目した詰み評価関数の自動生成. 第 10 回ゲーム・プログラミング ワークショップ, November 2005.

[2] Makoto Miwa, Daisaku Yokoyama, and Takashi Chikayama. Automatic construction of static evaluation functions for computer game players. In Proceedings of the 9th International Conference on Discovery Science, pp. 332–336. Springer, October 2006.

[3] 三輪誠, 横山大作, 近山隆. 指し手の履歴の抽出に基づくカテゴリの拡張. 第 11 回ゲーム・プログラミング ワークショップ, November 2006.

[4] Makoto Miwa, Daisaku Yokoyama, and Takashi Chikayama. Automatic generation of evaluation features for computer game players. In IEEE Symposium on Computational Intelligence and Games (CIG '07), April 2007.