駒位置と効き関係に注目した詰み評価関数の自動生成

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いて精度の高い評価関数の作成は、探索アルゴリズムの選択とともに重要な要素のひとつである。精度の高い評価関数の作成のためにはその特徴を良く表す評価要素の選択が不可欠である。本研究では詰将棋を対象として駒位置と効き関係の単純な特徴要素から評価要素を自動生成することを目的として研究を行った。評価としては、生成した評価要素について単層パーセプトロンを用いて学習させることを行った。結果として、数千〜数十万個程度の評価要素を生成し、76%程度の正解率で詰み・不詰を判別できる評価関数を作成することができた。

Automatic construction of estimation of mates based on positions and effects of pieces

Makoto MIWA,[†] Daisaku YOKOYAMA[†]
and Takashi CHIKAYAMA[†]

Constructing evaluation functions with high accuracy is one of the critical factors like the selection of the fast search algorithm in computer game players. The selection of the useful evaluation features is essential to construct evaluation functions with high accuracy. In this paper, we generate evaluation features for evaluation functions in tsume-shogi problem automatically. We made the features from simple features, positions and effects of pieces. As an evaluation, we constructed evaluation functions by using the single-layer perceptron and the generated evaluation features. As experimental results, evaluation function using generated everal hundred thousand evaluation features was able to classify positions about 76% of accuracy.

1. はじめに

コンピュータゲームプレイヤにおいて静的評価関数 (以降、評価関数) はその振る舞いを決める重要な要素のひとつである。評価関数とは局面の有利・不利の判定を行うためのものであり、多くのゲームプレイヤプログラムではその評価する対象の特徴を表す評価要素の重み付き線形和で表されている。現在、将棋などの比較的難しいゲームにおいては、重み付けなどについて部分的に自動化されているものはあるものの、そのほとんどは人手で作られている。評価関数を人手で作る際にはそのゲームに関する深い知識が必要とされ、重要な評価要素の見落としの回避、足りない要素の発見のために多くの時間を費やす必要がある。選択した要素に重み付けを行うことについても同様に多くの時間を費やす必要がある。また、人手で作る方法はアドホックな方法であり、最適に近い評価関数が得られるとは限らない。

このような評価関数作成の問題を解決する一つのアプローチとして評価関数の自動生成がある。評価関数の自動生成は、オセロなどの比較的簡単な問題においては多く研究がなされており有用な結果が得られている。その一方で、それより難しいといわれるチェス・将棋・囲碁などのゲームについては研究が少なく決定的な手法も見つかっていない。

本稿では将棋の部分問題である詰将棋について、ある局面を与えた時、その局面が詰むか詰まないかの程度を返すような評価関数の評価要素の自動生成を目的として研究を行った。詰将棋を対象としたのは、詰み・不詰が明確であり、手法の評価対象として適切であると考えたためである。本研究では次の手順で詰将棋の評価関数を作成した。まず、局面の駒位置と効き関係を真偽値の特徴要素として抽出した。次に、その特徴要素の論理積を取ったもののうち、訓練局面の中に頻出であり、また詰み・不詰の選別に有用なものを選別し、評価要素とした。最後にパーセプトロンを用いて重み付けをすることで、評価関数を作成した。結果と

† 東京大学大学院新領域創成科学研究科
Graduate School of Frontier Sciences, University of Tokyo

しては、約 76%の正解率で詰み・不詰を判定できる評価関数を作成することができた。

本論文では以降、2 章で関連研究を紹介し、3 章で本研究の手法、4 章で実験結果について説明し、5 章でまとめと今後の課題を述べる。

2. 関 連 研 究

本章では関連研究として 2.1 で評価関数の自動生成に関する研究について紹介し、 2.2 で本研究で用いた頻出パターン抽出手法について述べる。

2.1 評価関数の自動生成

評価関数の自動生成は、コンピュータゲームプレイヤの自動生成につながることもあり、人工知能の大きな目標の 1 つである。しかし、その困難さから評価関数の自動生成に関する研究は非常に少ない。機械学習を用いた評価関数の生成に関する研究は、評価要素を人手で作成し、その重み付けを自動化するというものがほとんどである[12]。

評価関数の自動生成は単純な評価要素 (以降、特徴要素) を元に評価関数を構築する。その研究は評価関数の作成の方法から主に 2 つに分けることができる。

1 つは評価要素の生成を先に行い、抽出した評価要素に重み付けを行うことで評価関数を表現する方法である。この方法では、評価関数は評価要素の線形和で表される。このような研究には、ZENITH[7]・ELF[21]・GLEM[3]・金子らの手法[14),15)] などが挙げられる。

もう 1 つは特徴要素を元に多層ニューラルネットワークなどを用いて直接的に評価関数を表現する方法である。この方法では、評価関数は特徴要素の非線形結合で表される。このような研究には、強化学習を用いた TD-Gammon[18] や進化的アルゴリズムを用いた Fogel による Tic Toc Toe[8]、Kumar らによる Checker[6]、The Distributed Chess Project[13] などが挙げられる。

どちらの方法が良いと一概に言うことはできないが、前者の方法を用いる利点としては後者の方法に比べて

- 少ない資源で高次の評価要素を作成できること
- 評価関数の分析が容易であること
- 評価関数の高速化が容易であること

などが挙げられる。一方で後者の方法の利点としては表現能力が高いことが挙げられ、より正確な評価関数を実現できる可能性がある。

2.1.1 GLEM

GLEM[3),4)] とは、Generalized Linear Evaluation Model の略であり、M. Buro が提案した汎用の評価関数自動生成手法である。GLEM は彼が開発した 1997 年に人間のチャンピオンを倒した最強のコンピュータオセロプレイヤの 1 つである Logistello[2] に実際に用いられている。この手法は現在最も強いコンピュータオセロプレイヤの 1 つであるといわれている Herakles[19] など多くのコンピュータオセロプレイヤで使われ、非常に良い結果を残している。

GLEM は次のように評価関数を生成する。まず、その評価関数のもととなる特徴要素 ($atomic\ features$) を抽出する。これは、人手で行われ、候補としては他の特徴要素の組み合わせによって表現できない真偽値で表現された要素が挙げられる。次にアプリオリ法[1] に似た方法で、特徴要素の頻出な組み合わせを抽出し評価要素 ($configuration$) とする。最後に最小二乗法で重み付けを行う。

GLEM はその特徴として、人手で抽出した $pattern$ を用いることで、ある特定の形にあった特徴要素の組み合わせのみを対象としている。これにより頻出要素の生成を容易にし、GLEM はその高速な評価要素へのアクセスが可能となる。

Buro は 1,700 万の訓練局面、51 の $pattern$ を用いて、GLEM の手法に基づき、4 手ずつの 13 段階に分けて Logistello の評価関数を作成した。実験環境は CPU Athlon XP 1800+ (1,666MHz) を用いている。結果として約 150 万の評価要素が得られ、評価要素の重み付けに 6 時間かかり、1 秒に 140 万局面を評価できる高速な評価関数が得られている。

2.2 頻出パターン抽出手法

頻出パターンの抽出アルゴリズムは、POS, web-log, 化合物, ゲノムなど多くの応用があり、データマイニングの分野においてここ数年で飛躍的に進歩している。以前はアプリオリ法[1] が主流であったが、現在良い結果を出しているアルゴリズムの多くは、2000 年に提案された FP-growth[11] アルゴリズムにおける FP-tree というデータ構造を基にしている。頻出パターンとは、トランザクションがあるアイテム集合の部分集合として表現される場合に、その n 個のトランザクションの内に α 個 (最小サポートと呼ばれる) 以上出現する集合のことをいう。頻出パターンは、あるトランザクションの部分集合であるので、トランザクションの含まれるアイテム集合の部分集合として表現される。

このような頻出パターンを抽出するアルゴリズムで特に高速であるといわれているものには、FP-growth*[9] や LCM (Linear time Closed itemset Miner)[16),17)] などが挙げられる。

このように高速に抽出することはできるものの、特にその最小サポートを小さくしたとき、このような

頻出パターンは非常に大量になる。それを軽減することのできる方法の 1 つとして頻出飽和パターンのみを抽出する手法が提案されている。頻出飽和パターンとは、そのパターンが含まれるトランザクションの集合が等しいパターンについてその極大元のみを抽出する手法である。このような手法として FPclose[9] や LCM が提案されている。FPclose は FP-growth 同様に FP-tree を構築し、FP-tree から抽出したパターンを CFI (Closed Frequent Itemset) -tree という構造に挿入し、頻出飽和パターンを高速に抽出するアルゴリズムであり、IEEE ICDM'03 併設のプログラミングコンテスト FIMI'03 において優勝している。また、LCM は prefix 保存飽和拡張という手法を用いることで、解保存用メモリを持たずに重複なく頻出パターンを抽出することを可能にしており、IEEE ICDM'04 併設のプログラミングコンテスト FIMI'04 において優勝している。

3. 局面からの評価要素の抽出

本研究は詰み・不詰の訓練局面から目的とする詰み・不詰の程度を返す評価関数の評価要素を生成することを目的としている。

本研究において、この詰将棋を対象としたのは、

- 人手で抽出した特徴についてそれなりの高い精度で学習できており[22] 学習対象となりうる問題であることがわかっていること
- 詰将棋が詰み・不詰という明確な値を持った問題であり目的としている評価関数の評価要素の評価が比較的容易であること

の理由が挙げられ、本手法の対象として適切であると考えたためである。また、評価が容易であることのほかにも、精度の高い評価関数を作成することができれば、詰探索の制御などへの応用が可能であるという知見も[22] において得られており、そのような評価関数の作成を目的とすることは価値があると考えられる。

また、本研究では詰将棋を対象としているが、本手法は、勝ち・負けを基にした方法を用いることで実際の本将棋にも適用可能であると考えらえる。ただし、現在の環境で本将棋を学習の対象とするのは難しく、今のところ本将棋は本手法を評価するには適切な対象ではないと考えられる。実際、本将棋について現在の人手で作成した評価関数と同等の精度の評価関数を機械学習によって作るということができている研究を見つけることはできなかった。

本章では、以降、本手法において評価要素を生成する手法について説明する。本手法は次の 3 つの過程からなる。

(1) 局面の駒位置と効き関係を特徴要素として抽出する。
(2) 頻出飽和パターン (特徴要素を組み合わせ) を抽出する。
(3) 頻出飽和パターンから詰み・不詰の判定に有用なものを選択することで評価要素を生成する。

以降この 3 つについて説明を行う。

(1一, 香, 後手)
(2一, 桂, 後手)
...
(8九, 桂, 先手)
(9九, 香, 先手)
(1一, 後手)→(1二, 空)
(1一, 後手)→(1三, 後手)
...
(9九, 先手)→(9七, 先手)
(9九, 先手)→(9八, 空)

図 1 初期盤面の特徴要素

(歩1 , 歩, 先手)
(歩2 , 歩, 先手)
(飛1 , 飛, 後手)

図 2 持ち駒の特徴要素

3.1 局面の駒位置と効き関係の抽出

詰将棋の評価要素を作成するためには、その評価関数への入力を表現するために、詰将棋の局面の状態を表現する集合が必要である。それぞれが局面の特徴の一端を表す単純な要素 (特徴要素) のセットとして、本研究では局面の駒位置と効き関係について抽出を行い、詰将棋に関する情報とした。局面の効き関係を用いたのは、3.2 に示すように今回は論理積のみを対象としていることから駒位置によってのみでは表現できない効き関係が存在するためである。

駒位置は (位置, 駒の種類, 手番) の形で、また効き関係については (位置, 手番) → (位置, 手番) の形でそれぞれ表現した。図 1 に初期盤面の特徴要素を示す。持ち駒については図 2 のように駒位置の位置に盤外の駒用のマスを設けた。また、効き関係については効き対象の手番に空マスをいれることで、空マスについての情報も考慮に入れることにした。

これにより局面は 37,336☆の真偽値の要素によって表現することができ、この 37,336 の要素を特徴要素とした。

3.2 頻出飽和パターンの抽出

本研究では 3.1 で説明した特徴要素の単純な論理積で評価要素を表現する。

評価要素を得る方法には、ZENITH[7] などのように詰将棋のルールを与えてそれを展開して得る方法と、GLEM[3] のように特徴要素を組み合わせることで得る方法が考えられる。前者については、抽象度の高い情報を与えることが可能であり、また特徴要素がそのルールを表現できるものでなくてはならず特徴要素に関して見落としがないという利点があるが、そのルールの記述は困難である。後者については、その特徴要素の選択には多くの可能性があり、特徴要素の選択を誤ると詰将棋の評価関数に必要な要素を全ては表現できないという問題もあるが、その特徴要素の抽出は比較的容易である。本研究では後者のような特徴要素を組み合わせる方法で評価要素を抽出することにした。

ここで、上記の 4 万程度の特徴要素の全ての論理積を取ると、組み合わせ爆発が起こってしまう。例えば、要素 3 つの全ての論理積を記憶するのに必要な記憶容量は単純に考えると 200TByte 以上☆☆になるため、通常のコンピュータの容量では全ての論理積を試せるのは要素数 2 までである。この制限では表現力の高い評価要素が得られなくなってしまう。このため本研究では、この組み合わせ爆発を防ぐために大量の棋譜に頻出する論理積のみを抽出した。めったに出現しないが詰みであることを確定できるような要素があればそのような要素を見逃していることも考えられるがそのような要素を特定することは困難である。また、棋譜にめったに出現しないような特徴については学習において証拠が少ないことになり、重み付けにおいて過学習の原因となりやすいため出現頻度で要素の数を削減することはいずれにせよ必要である。

この論理積の抽出は、その特徴要素が真になっているラベルのセットで局面を表現することで、頻出パターンの抽出問題と考えることができる。頻出パターンの抽出にはデータマイニングの分野で注目されている 2.2 に示した頻出飽和パターン抽出手法を用いた。頻出飽和パターンは頻出パターンに比べて数が少なく、抽出されるパターンにおける冗長性を減らすことができる。3.3 に示す指標や多くの学習アルゴリズムには、評価要素それぞれが独立であるという仮定があり、この多くの従属なパターンの削除はその仮定を満たさない評価要素の削減につながる。このため、学習をより効果的に行うことができると考えられる。本研究では頻出飽和パターン抽出手法の一つである LCM を用いた。

3.3 評価要素の選択

頻出飽和パターンは非常に多い。そのため実際の評価関数に用いるにはその学習が困難である上に、評価にかかる時間も多くなってしまうことが考えられる。そこで、この大量のパターンから重要であるパターンのみを選別することを考えた。この選別の基準としては、ベイズ学習に基づいた確率の推定値[20]、カイ 2 乗テスト、相互情報量[5] などが考えられる。本研究ではこのうち比較的計算コストの小さい確率の推定値とカイ 2 乗テストを用いることにした。

- ベイズ学習に基づいた確率の推定値
  確率の推定値とは局面にあるパターンが出現した時のその局面における詰みの確率の推定値ことである。証拠の数が多ければ、あるパターンが出現した時の詰みの確率の推定値 p は、そのパターンが含まれる訓練局面数 n、そのうち詰みである局面数 k について、

$$p = \frac{k}{n} \tag{1}$$

  として、最尤推定できる。しかし、実際には 2.2 に示した頻出パターンの最小サポートはそれほど大きくない。このため本研究ではベイズ学習の手法を用いて確率の推定値を次のように求める。まず、パターンの確率を確率変数 $\tilde{\theta}$ として、パターンの確率を求める問題をこの $\tilde{\theta}$ の期待値を求める問題と考えることにする。ここで、訓練局面について、対象とするパターンを含む局面が n 局面あり、そのうちの k 局面が詰みであったとする。このような事象を A とすると、ベイズの定理に基づき、事象 A が起こったという条件のもとでの $\tilde{\theta}$ の事後確率密度は、

$$\begin{aligned} P(\theta|A) &= \frac{P(\theta)P(A|\theta)}{P(A)} \\ &= \frac{P(\theta)P(A|\theta)}{\int_0^1 P(\theta)P(A|\theta)d\theta} \end{aligned} \tag{2}$$

  で与えられる。ここで、事象 A をベルヌーイ試行と考えることができるので、その確率は 2 項分布により次のように与えられる。

$$P(A|\theta) = {}_nC_k\theta^k(1-\theta)^{n-k} \tag{3}$$

☆ $9 \times 9 \times 14 \times 2$ (盤上の駒) $+38 \times 2$ (持ち駒) $+ 9 \times 9 \times 2 \times 9 \times 8 \times 3$ (効き)

☆☆ ${}_{37336}C_3 \times \ln 37336 \times 3$[bit]

これを代入して、

$$P(\theta|A) = \frac{P(\theta)_n C_k \theta^k (1-\theta)^{n-k}}{\int_0^1 P(\theta)_n C_k \theta^k (1-\theta)^{n-k} d\theta} = \frac{P(\theta)\theta^k (1-\theta)^{n-k}}{\int_0^1 P(\theta)\theta^k (1-\theta)^{n-k} d\theta} \quad (4)$$

が得られる。ここで、事前確率密度 P(θ) をどうするかという問題がある。ここでは、対象とするパターンに関して、事前知識が全くないものと考えて、事前確率密度は一様分布とする。これは、

$$P(\theta) = 1 \quad (5)$$

とするということである。これにより、

$$P(\theta|A) = \frac{\theta^k (1-\theta)^{n-k}}{\int_0^1 \theta^k (1-\theta)^{n-k} d\theta} = \frac{\theta^{(k+1)-1}(1-\theta)^{(n+2)-(k+1)-1}}{\int_0^1 \theta^{(k+1)-1}(1-\theta)^{(n+2)-(k+1)-1} d\theta} \quad (6)$$

となる。この確率分布は、ベータ分布であり、期待値は次のようになる。

$$E(\theta) = \frac{k+1}{n+2} \quad (7)$$

この結果から、あるパターンが出現した時の詰みの確率の推定値 p は、そのパターンが含まれる訓練局面数 n、そのうち詰みである局面数 k について、

$$p = E(\theta) = \frac{k+1}{n+2} \quad (8)$$

で表される。この結果を見ればわかるとおり、計算コストは非常に小さいものである。

この値について、詰みまたは不詰の確率が高いもの、すなわち確率の大きいもの・小さいものを選択すれば、有用な情報として用いることができると考えられる。

- カイ 2 乗テスト

  カイ 2 乗テストとは、クラスのラベルとパターンの間に相関があるか否かを調べる統計的な尺度である。カイ 2 乗テストは、独立性の指標となるカイ 2 乗統計量を用いる。n 局面の訓練局面において、クラス i、パターン t について、k_{i0} を「クラス i に属するもののうち t を含まない局面数」、k_{i1} を「クラス i に属するもののうち t を含む局面数」とし、クラスを 0,1 とすると、カイ 2 乗統計量 χ^2 は、次のように表される。

$$\chi^2 = n(k_{11}k_{00} - k_{10}k_{01})^2 \times (k_{11}+k_{10})^{-1} \times (k_{01}+k_{00})^{-1} \times (k_{11}+k_{01})^{-1} \times (k_{10}+k_{00})^{-1} \quad (9)$$

  カイ 2 乗統計量は、パターン 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。

  ここで選別したいものはラベルを推定できるものであり、ラベルとの独立性が低いもの、つまりカイ 2 乗統計量が大きいものを選択すれば、有用な情報として用いることができると考えられる。

本研究ではこのようにして選択したパターンを論理積で表現したものを評価要素とした。ただし、このようにして選択した評価要素のみでは、全ての局面を網羅することはできないため特徴要素をパターンに残すことで対処した。

3.4 評価要素の重み付けによる評価関数の生成

評価要素の重み付けは、その評価要素の数が比較的多いため、学習が容易な単層パーセプトロンを用いて学習を行った。

4. 評　　価

4.1 実験方法

実験は Intel Xeon 3.06GHz dual・メモリ 2GB(以降 Xeon)、AMD Opteron 840(1.4GHz)×4・メモリ 4GB(以降 Opteron) の 2 台のマシン上で行った。実装には C++言語を用いた。

評価対象としては、訓練局面 80,000 局面・テスト局面 9,768 局面を用いた。これらの局面は、将棋倶楽部 24 のレーティング 2,200 以上の棋譜 9,144 局について

(1) それぞれについて終局までの最後の 10 局面を取り出す。
(2) 取り出した局面それぞれについて浅く探索し、探索中に出てきたノードをランダムに 1 つずつ取り出す。

ことで抽出を行ったものである。抽出した局面数が得られるであろう局面数よりも少ないのは、読めない棋譜や同じ局面を排除したためである。また、ラベル付けに関しては、PDS で 5,000,000 ノード探索するうちに詰みであると判定された局面を詰み、それ以外を不詰としてラベル付けした。

4.2 実験結果

抽出した頻出飽和パターンの数を図 3 に、4.1 に示した Xeon において抽出にかかった時間を図 4 に示した。結果として最小サポートが 2%のときに 38,173,197 の頻出飽和パターンが得られている。頻出パターンを全て抽出した場合には、319,528,422 ものパターンが得られており、飽和パターンのみを抽出することで頻出

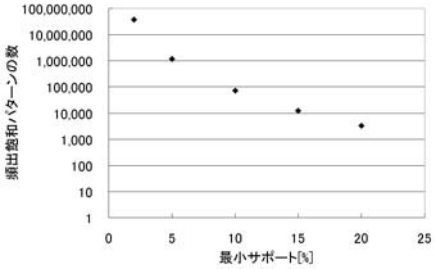

図 3 最小サポートに対する頻出飽和パターンの数

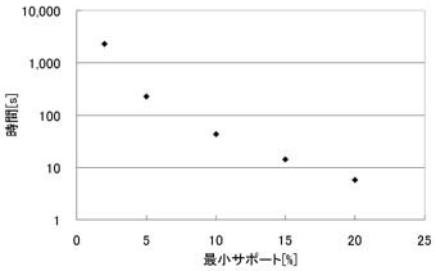

図 4 最小サポートに対する頻出飽和パターン抽出にかかった時間

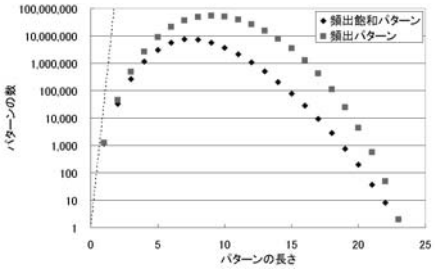

図 5 頻出飽和パターンの長さごとの分布

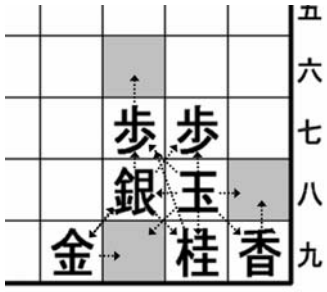

図 6 長い頻出飽和パターンの例

パターンの 88%程度を削減できており、飽和パターンの抽出はパターンの削減に大いに貢献していることがわかる。この最小サポート 2%の頻出飽和パターンの長さごとの分布を図 5 に示す。頻出パターンについてもその長さごとの分布を示した。点線は全パターンを抽出した場合の数を示しており、ほとんどのパターンが頻出でないことがわかる。図からわかるとおり 23 の特徴要素の積で表されるような長いパターンを得ることができている。この長いパターンの例を図 6 に示した。矢印は効きを、灰色のマスは駒がないことが効きによりわかったマスである。

次にこの最小サポート 2%の頻出飽和パターンについて、評価要素の選択を行った。3.3 に示したベイズ学習に基づいた確率の推定値、カイ 2 乗テストそれぞれについて選択を行った結果を図 7・8 に示した。確率については、詰みの確率が 0.5 以下の確率のものは不詰の確率について、それ以外のものは詰みの確率について、それぞれの確率の下限を超えるものを抽出するようにした。カイ 2 乗テストについては、カイ 2 乗分布表の値を外れたものが多かったため、カイ 2 乗統計量そのものの値について、その値よりも大きいものを選択するようにした。選択には 4.1 に示した Opteron において約 14 時間かかった。

最後にこのようにして抽出した評価要素を特徴量として、単層パーセプトロンで学習させた。学習係数 0.001・学習回数 1,000 回として学習させた結果を図 9・10 に示す。図 9・図 10 に示したように最も良いもので 76.62%の正解率が得られた。いずれについても、評価要素が多いものについて正解率が下がっているが、これは学習係数・学習回数を固定して比較しているために過学習が起きていることが考えられる。また、この結果は人手で[22] に示された評価要素を選別

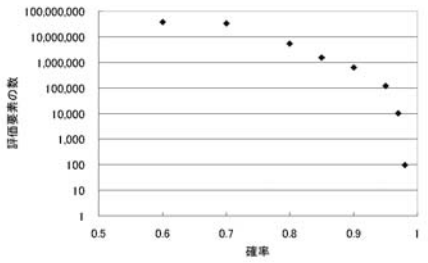

図 7 確率の推定値に基づいた選択

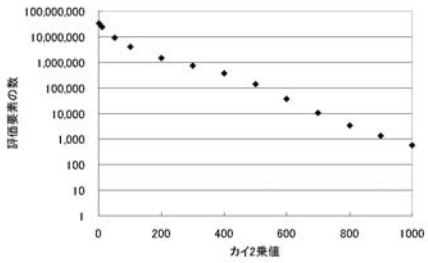

図 8 カイ 2 乗テストに基づいた選択

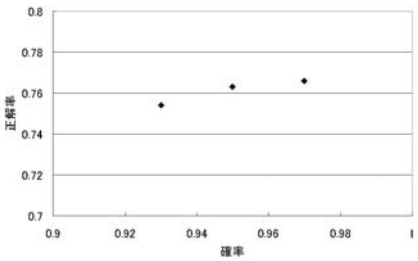

図 9 確率の推定値に基づいた評価要素を用いた学習

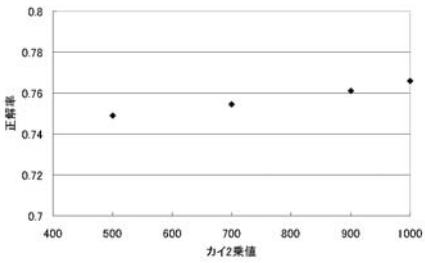

図 10 カイ 2 乗テストに基づいた評価要素を用いた学習

し、重み付けした評価関数よりも 5%程度悪い。これは、抽出した特徴要素が学習局面数についてスパースであること、頻出する組み合わせの下限の設定が大きすぎたことが原因であると考えられる。

5. おわりに

本研究では、3 章で示したように、局面の駒位置と効き関係を表す単純な真偽値の特徴要素を元に詰むか詰まないかの程度を返す評価関数を作成した。具体的には、特徴要素について訓練局面から頻出飽和パターンを抽出し、用意した確率またはカイ 2 乗の指標において有用なものを選択し、それを評価要素とし、単層パーセプトロンで学習することで評価関数を構築した。結果として 4 章で示したように 4,000 万弱の頻出飽和パターン、数千から数十万の評価要素が得られ、最も良いもので 76%程度の正解率が得られた。

今後の課題としては、1 つは特徴要素の組み合わせの選択をより洗練させることが挙げられる。これには、選択の精度を上げることと計算量を減らすことの 2 つの対処が必要である。選択の精度を上げるには、重要な要素の選択に現在機械学習で提案されている他の特徴選択手法[10] を用いることが考えらえる。これらの手法には今回の方法に比べると計算コストがかかるものの、精度の高い選択ができているものが存在する。また、計算量を減らすにはビーム探索法や遺伝的アルゴリズムなどの近似アルゴリズムを用いることが考えられる。特徴要素を減らして最小サポートを小さくするなどの方法もあるが、この対処では特徴要素の数が詰将棋よりも多くなってしまうような問題に応用できず、根本的な解決にはなっていない。また、表現能力を上げるために和や否定を用いることも考えられるため、このような計算量を減らす手法は必要である。

また、もう 1 つの課題として、実際の評価関数として用いるための高速化が挙げられる。このためには、GLEM における *pattern* が有用であると考えられる。*pattern* を自動生成する方法は今のところ提案されていないが、特徴要素のクラスタリングなどを行うことで可能ではないかと考えている。この *pattern* を用いることで失われる組み合わせも存在するが、特徴要素の組み合わせの計算量を削減できることも考えられる。また、最小サポートをより小さくすることも可能となり、今回見逃した有用な要素を発見できる可能性がある。

参 考 文 献

1) Agrawal, R., Imielinski, T. and Swami, A. N.: Mining Association Rules between Sets of Items in Large Databases, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data* (Buneman, P. and Jajodia, S.(eds.)), Washington, D.C., pp. 207–216 (1993).
2) Buro, M.: LOGISTELLO. available at http://www.cs.ualberta.ca/ mburo/log.html.
3) Buro, M.: From Simple Features to Sophisticated Evaluation Functions, *Proceedings of the First International Conference on Computers and Games (CG-98)* (van den Herik, H. J. and Iida, H.(eds.)), Vol. 1558, Tsukuba, Japan, Springer-Verlag, pp. 126–145 (1998).
4) Buro, M.: Improving Heuristic Mini-Max Search by Supervised Learning, *Artificial Intelligence*, Vol. 134, No. 1-2, pp. 85–99 (2002). Special Issue on Games, Computers and Artificial Intelligence.
5) Chakrabarti, S.: *Mining the Web: Discovering Knowledge from Hypertext Data*, Morgan Kaufmann Pub (2002).
6) Chellapilla, K. and Fogel, D. B.: Evolving an Expert Checkers Playing Program Without Using Human Expertise, *IEEE Transactions on Evolutionary Computation*, Vol. 5, No. 4, pp. 422–428 (2001).
7) Fawcett, T. E.: *Feature Discovery for Problem Solving Systems*, PhD Thesis, Department of Computer Science, University of Massachusetts, Amherst, MA (1993).

8) Fogel, D. B.: Using Evolutionary Programming to Construct Neural Networks that are capable of playing Tic-Tac-Toe, *Proceedings of the IEEE International Conference on Neural Networks (ICNN-93)*, San Francisco, pp. 875–879 (1993).
9) Grahne, G. and Zhu, J.: Efficiently using prefix-trees in mining frequent itemsets, *In Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations* (2003).
10) IsabelleGuyon, A.E.: An Introduction to Variable and Feature Selection, *JMLR Vol.3*, pp. 1157–1182 (2003).
11) J.Han, J.Pei, Y.Y.: Mining Frequent Patterns without Candidate Generation, *SIGMOD Conference 2000*, pp. 1–12 (2000).
12) rnkranz, J. F.: Machine Learning in Games: A Survey, *Machines that Learn to Play Games* (Fürnkranz, J. and Kubat, M.(eds.)), Nova Science Publishers, Huntington, NY, chapter2, pp. 11–59 (2001).
13) Seliger, R.: THE DISTRIBUTED CHESS PROJECT. http://neural-chess.netfirms.com/.
14) T. Kaneko, K. Y. and Kawai, S.: Automatic Feature Construction and Optimization for General Game Player, *The 6th Game Programming Workshop*, pp. 25–32 (2001).
15) T. Kaneko, K. Y. and Kawai, S.: Automated Identification of Patterns in Evaluation Functions, *Advances in Computer Games 10* (van den Herik, H. J., Iida, H. and Heinz, E. A.(eds.)), Kluwer Academic Publishers, pp. 279–298 (2004).
16) T. Uno, M. Kiyomi, H. A.: LCM ver 3.: Collaboration of Array, Bitmap and Prefix Tree for Frequent Itemset Mining, Chicago, IL (2005).
17) Takiake Uno, Masashi Kiyomi, H. A.: LCM ver.2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets, *International Conference on Data Mining, Frequent Itemset Mining Implementations 2004* (2004).
18) Tesauro, G.: TD-Gammon, A Self-teaching Backgammon Program, Achieves Master-Level Play, pp. 19–23 (1993).
19) Tournavitis, K.: Herakles. available at http://www.herakles.tournavitis.de/.
20) Tsuruoka, Y. and Chikayama, T.: Estimating Reliability of Rules in Decision Lists using Bayesian Learning, *Journal of Natural Language Processing*, Vol. 9, No. 3 (2002). in Japanese.
21) Utgoff, P. E. and Precup, D.: Constructive Function Approximation, *Feature Extraction, Construction and Selection: A Data Mining Perspective* (Liu, H. and Motoda, H.(eds.)), The Kluwer International Series in Engineering and Computer Science, Vol. 453, Kluwer Academic Publishers, chapter 14 (1998).
22) 三輪誠, 横山大作, 近山隆: SVM による将棋の詰みの予測とその応用, 第 9 回ゲーム・プログラミング ワークショップ, pp. 143–150 (2004).