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

三輪 誠[†] 横山大作[†] 近山 隆[†]

正確な評価関数の生成はコンピュータゲームプレイヤにおいてその振舞いを決める重要な要素の 1 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．本稿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合せで表現し，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た．また分割処理によって大きな問題を扱え，並列化により高速に実行できることも確認できた．

Automatic Generation of Evaluation Features for Computer Game Players

Makoto Miwa,[†] Daisaku Yokoyama[†] and Takashi Chikayama[†]

Accuracy of evaluation functions is one of the critical factors in computer game players. Evaluation functions are usually constructed manually as a weighted linear combination of evaluation features that characterize game positions. Selecting evaluation features and tuning their weights require deep knowledge of the game. In this paper, we propose a new fast and scalable method to automatically generate game position features based on game records to be used in evaluation functions. Our method treats two-class problems which are common in many types of games. Evaluation features are built as conjunctions of the simplest features representing positions. We select these features based on two measures: frequency and conditional mutual information. To evaluate the proposed method, we applied it to 200,000 Othello positions. The proposed selection method is found to be effective, because the Naive Bayesian classifier using automatically generated features is more accurate than other classifiers using simple features. We also show that this generation method can treat large problems by dividing them into small problems and can be parallelized easily.

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舞いを決める重要な要素の 1 つである．評価関数はゲームの局面を評価し，有利・不利の度合いを表す評価値を出力する．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．

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．

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対局サーバ上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[21] や Othello[2] 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

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

研究がさかんであり，有用な結果が得られている．一方で，これらの手法の複雑なゲームへの応用はなされておらず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．

本稿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．また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付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ともに並列化による高速化も可能にしている．

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[14]．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[14]，順序相関を用いた手法[5]，最適制御理論をもとにした手法[15] 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基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 らによる○×ゲーム・Checker（Blondie24）[11]・チェス[12]，Chellapilla らによる Checker[4]，Particle Swarm Optimization（PSO）を用いた Messerschmidt の○×ゲーム[18] や Franken らの checkers[13] などがある．

階層的な手法では人手で評価関数を作るときと同様，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じか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[23]，GLEM[2]，ZENITH やそれを改良した Kaneko らの方法[17]，Dummy らの方法[7] 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[1]．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い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．

3. 提案手法

本手法では 2 クラスにラベル付けされた訓練局面を基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 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．これはすべ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）[22] 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）[10] 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

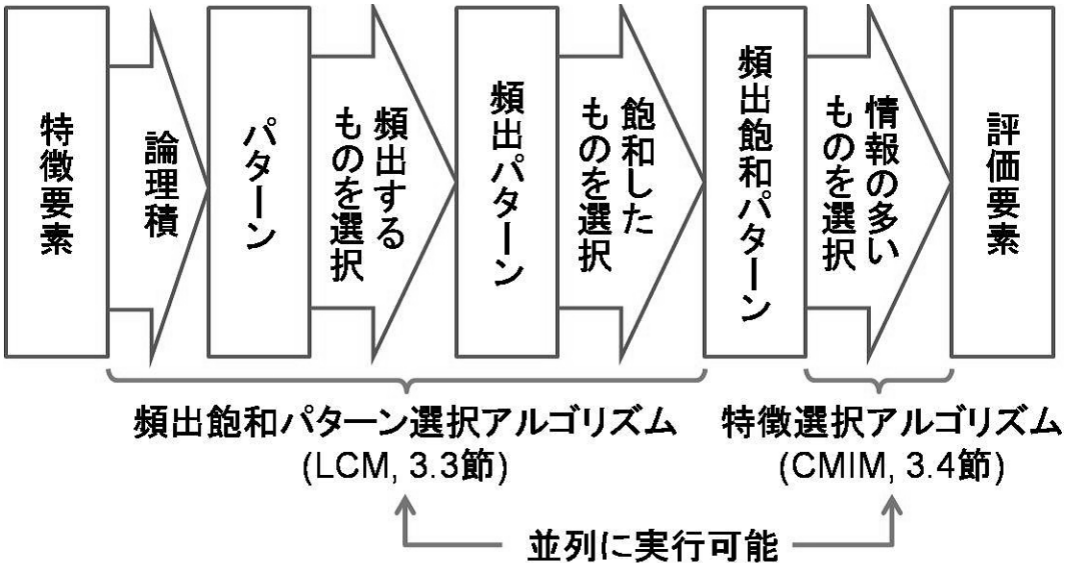

図 1 特徴要素からの評価要素生成の流れ
Fig. 1 Flow to generate evaluation features from base features.

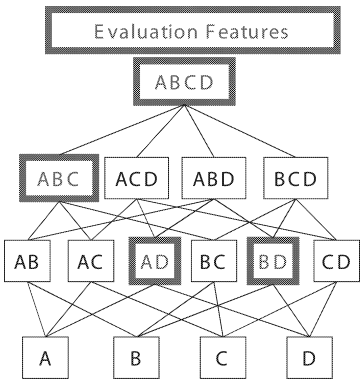

図 2 特徴要素・パターン・評価要素
Fig. 2 Base features, patterns and evaluation features.

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 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．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基に評価要素を選択する．

本章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基にした評価要素の生成について説明する．

3.1 評価関数の表現

評価関数には高速かつ正確であることが要求される．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．

$$F(p) = \Sigma_i w_i \cdot f_i + b \qquad (1)$$

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．しかし，生成・実行にかかる計算コストや過学習の危険があるため，単純な線形和を用いることが多い．

また，このような線形☆の関数を用いることで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じかに表しているため解析・チューニングが容易である．

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すべての局面を表現できる必要がある．

局面の情報を表し，意味のうえ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．特徴要素としてはたとえば○×ゲームについ

☆ 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．

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 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）
Fig. 3 Training positions, frequent patterns and frequent closed patterns when the minimum support is 3.

て「左上に○がある」というものがあげられる．この特徴要素はそれを用いることで正確にすべての局面を区別できるように選択する必要がある☆．

本手法は対象を真偽値としているが，連続値についてもその範囲を区切って複数の真偽値で表す離散化[9]の手法を用いることで利用できる．

本手法では特徴要素の論理積で表される組合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すべての特徴要素が現れるときにそのパターンは真となる．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合せは扱わない．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．

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，これらをすべて評価要素として扱うこと・これらをすべ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めった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．

Definition1 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表される $k+1$ 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．

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する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そのほとんどは削減されるにもかかわらず，その削減にコストがかかる．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．

Definition2 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 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 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．

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合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．我々はこの手法として LCM（Linear time Closed set Miner）[22] を用いた．LCM は IEEE ICDM'04 併設のプログラミングコンテスト FIMI'04 において優勝しており，頻出飽和パターン数え上げにおいて最も高

☆ 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，たとえば○×ゲームにおいて「○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．

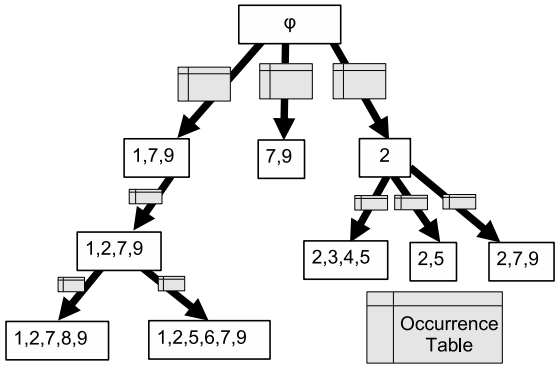

図 4 LCM の探索木

Fig. 4 A search tree in LCM. A node has a pattern, and children are made by appending base features to the pattern. The node's *occurrence table* contains the information about positions related to its pattern and is delivered to its children after database reduction.

速なアルゴリズムの 1 つである．

図 4 に LCM の探索木を示す．すべてのノードは頻出パターンを持っており，ルートノードは空パターンを持っている．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．この拡張により LCM は高速に数え上げることができる．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．それぞれのノードはその持っ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．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．

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（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（3.4 節参照）である．

LCM においては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 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．この並列化においてすべ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．ま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．

表 1 並列 LCM

Table 1 Parallelized 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,	特徴要素の追加の順番と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,	ワークキューからタスクを受け取る
step 2,	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 3,	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 4,	ワークキューが空なら終了，空でなければ step 1 に戻る

3.4 条件付き相互情報量による評価要素の選択

頻出飽和パターンは対象問題に対して有用であるかどうかは分からない．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，その特徴全体を基に選択する PCA や直接学習を行うことによって選択を行うようなアルゴリズム[16] を用いることは難しい．カイ 2 乗値や情報利得[8] などパターンそれぞれを個別に扱う手法は高速であり有用ではあるが，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）[10] を用いる．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．

$$v(1) = \arg\max_n \hat{I}(Y; X_n) \quad (2)$$
$$v(k+1) = \arg\max_n \left\{ \min_{l \le k} \hat{I}(Y; X_n | X_{v(l)}) \right\}$$
$$(1 \le k < K)$$

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．

$$\hat{I}(Y; X) = \hat{H}(Y) + \hat{H}(X) - \hat{H}(Y, X) \quad (3)$$

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

$$\hat{H}(X_1, \ldots, X_n) = -\sum_{x_1, \ldots, x_n \in \{0,1\}^n} p_{x_1, \ldots, x_n} \log p_{x_1, \ldots, x_n} \quad (4)$$

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 を知ったうえでの 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．

$$\begin{aligned} \hat{I}(Y; X|Z) &= \hat{H}(Y|X) - \hat{H}(Y|X, Z) \\ &= \hat{H}(Y, Z) - \hat{H}(Z) \\ &\quad -\hat{H}(Y, X, Z) + \hat{H}(X, Z) \end{aligned} \quad (5)$$

この選択により依存の少ないパターンを選択できる．CMIM では扱うすべ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．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行う．分割統治 CMIM の詳細について**表 2** に示した．step 3 の基準については繰返しの数（今回の実験では 5 回とした）や step 2 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すべてから 1 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☆の問題の 1 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** 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分かる．

表 2 の step 2 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

表 2 分割統治 CMIM
Table 2 Divide-and-conquer-like CMIM selection.

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,	それぞれの集合から一定値（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持つパターンを選択する
step 3,	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

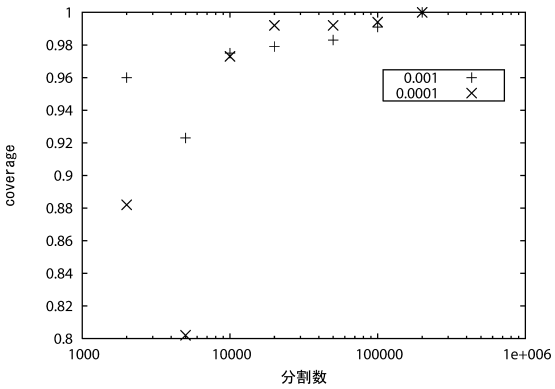

図 5 Dorothea における CMIM で選択された特徴と分割統治 CMIM で選択された特徴の相違
Fig. 5 Difference between selected features using CMIM and ones using the divide-and-conquer-like CMIM selection in Dorothea. The coverage shows how selected features using the divide-and-conquer-like CMIM selection covers selected features using CMIM.

☆ http://www.nipsfsc.ecs.soton.ac.uk/

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．

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．本章では評価に用いた 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．

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[6] 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き独立な確率変数であるという仮定をおいて評価対象の確率を計算することで分類を行う．条件付き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 (6) に示す．

$$f(\mathbf{x})=\sum_{i=1}^{N}\left\{\log\frac{\hat{P}_{1,1}(x_i)\hat{P}_{0,0}(x_i)}{\hat{P}_{0,1}(x_i)\hat{P}_{1,0}(x_i)}\right\}x_i+b \quad (6)$$

ここで $\hat{P}_{\alpha,\beta}(x)$ とは評価要素 x が α（局面に出現するとき 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．

4.2 評価設定

4.2.1 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，これ以外のゲームの知識は用いなかった．

4.2.2 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．これらの局面はすべて 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[3] から得られたものである．

4.2.3 評価環境

評価環境としてはそれぞれのノードが Intel Xeon 2.4 GHz dual，2 GB 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. 実験結果

本章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．

5.1 評価要素

まず 4.2.2 項に示した 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 6** 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 7** 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

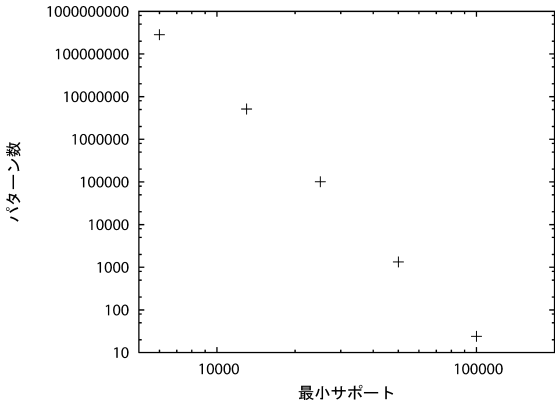

図 6 様々な最小サポートにおける 200,000 局面から選択した頻出飽和パターンの数

Fig. 6 Number of frequent closed patterns along the minimum support out of a total of 200,000 positions.

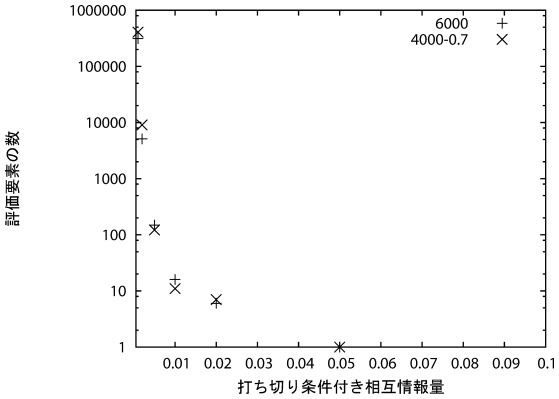

図 7 CMIM における打ち切り条件付き相互情報量を変化させた場合の特徴要素の数

Fig. 7 Number of selected evaluation features along the cut-off conditional mutual information using a CMIM selection (6,000: minimum support=6,000, selected from 282,615,853 patterns, 4,000-0.7: minimum support=4,000, information gain ≤ 0.7, selected from 172,022,168 patterns).

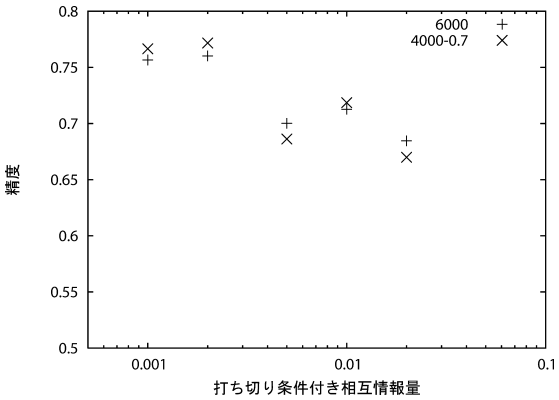

図 8 選択した評価要素を用いた際の Naïve Bayesian 分類器の正解率

Fig. 8 Accuracy of Naïve Bayesian classifiers using selected features.

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 のとき，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 140 MB，CMIM での選択では最大で 250 MB 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．

5.2 分類結果

図 8 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 7 に示したものを用いた．それぞれのテストには 4.2.2 項に示した 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．勝ちとラベル付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・勝ちとラベル付けされた局面を 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．

表 3 特徴要素を用いた場合の分類器の正解率
Table 3 Accuracies of classifiers using base features.

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%

また**表 3** に特徴要素を用いて同様のデータについて 3 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分かる．ここで Naïve Bayesian 分類器と判別分析[6] 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[20] 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分かる．また Support Vector Machine（SVM）についても実験を行ったが，SVM には有効な並列化手法は今のところ提案されておらず，逐次では 1 週間かけても終わらなかった．これより今回の手法では既存の手法が扱えないような問題にも対応できていることが分かる．

6. まとめと今後の課題

本稿ではゲームの対戦履歴を基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 3 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させることが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．

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* [2] 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いわれる他のゲームへの適用も課題の 1 つである．我々はこれまでに並列化を行わないで将棋の詰み問題に適応できること[19] 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．

参 考 文 献

1) Althöfer, I.: Computer-Aided Game Inventing (2003). http://citeseer.ist.psu.edu/674109.html
2) Buro, M.: From Simple Features to Sophisticated Evaluation Functions, *Proc. 1st International Conference on Computers and Games* (*CG-98*), Tsukuba, Japan, van den Herik, H.J. and Iida, H. (Eds.), Vol.1558, pp.126–145, Springer-Verlag (1998).
3) Buro, M. and Durdanovic, I.: An Overview of NECI's Generic Game Server (2001). http://citeseer.ifi.unizh.ch/600506.html
4) Chellapilla, K. and Fogel, D.B.: Evolving an Expert Checkers Playing Program Without Using Human Expertise, *IEEE Trans. Evolutionary Computation*, Vol.5, No.4, pp.422–428 (2001).
5) Gomboc, D., Marsland, T.A. and Buro, M.: Evaluation Fuction Tuning Via Ordinal Correlation, *Advances in Computer Games*, van den Herik, Iida, H. (Eds.), pp.1–18, Kluwer (2003).
6) Duda, R.O., Hart, P.E. and Stork, D.G.: *Pattern Classification, 2nd ed.*, Wiley Interscience (2000).
7) Duminy, W.H. and Engelbrecht, A.P.: Composing linear evaluation functions from observable features., *South African Computer Journal*, Vol.35, pp.48–58 (2005).
8) Eyheramendy, S. and Madigan, D.: A Novel Feature Selection Score for Text Categorization, *Proc. 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. (2005).
9) Fayyad, U.M. and Keki, B.: Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning, *IJCAI-93*, pp.1022–1027 (1993).
10) Fleuret, F.: Fast Binary Feature Selection with Conditional Mutual Information, *JMLR*, Vol.5, pp.1531–1555 (2004).
11) Fogel, D.B.: *Blondie24: Playing at the Edge of AI*, Morgan Kaufmann (2001).
12) Fogel, D.B., Hays, T.J., Hahn, S.L. and Quon, J.: A Self-Learning Evolutionary Chess Program, *Proc. IEEE*, Vol.92, pp.1947–1954 (2004).
13) Franken, N. and Engelbrecht, A.: Comparing PSO structures to learn the game of checkers from zero knowledge, *Proc. 2003 Congress on Evolutionary Computation CEC2003*, Canberra, Sarker, R., Reynolds, R., Abbass, H., Tan, K.C., McKay, B., Essam, D. and Gedeon, T. (Eds.), pp.234–241, IEEE Press (2003).
14) Fürnkranz, J.: Machine Learning in Games: A Survey, *Machines that Learn to Play Games*, Fürnkranz, J. and Kubat, M. (Eds.), chapter 2, pp.11–59, Nova Science Publishers, Huntington, NY (2001).
15) Hoki, K.: Optimal control of minmax search results to learn positional evaluation, *The 11th Game Programming Workshop* (*GPW 2006*), pp.78–83 (2006). (In Japanese).
16) Hyvärinen, A.: A survey on Independent Component Analysis, *Neural Computing Surveys*, Vol.2, pp.94–124 (1999).
17) Kaneko, T., Yamaguchi, K. 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.), pp.279–298, Kluwer Academic Publishers (2004).
18) Messerschmidt, L. and Engelbrecht, A.P.: Learning to Play Games Using a PSO-Based Competitive Learning Approach., *IEEE Trans. Evolutionary Computation*, Vol.8, No.3, pp.280–288 (2004).
19) Miwa, M., Yokoyama, D. and Chikayama, T.: Automatic Construction of Static Evaluation Functions for Computer Game Players, *Proc. 9th International Conference on Discovery Science*, Todorovski, L., Lavrač, N. and Jantke, K.P. (Eds.), pp.332–336, Springer (2006).
20) Riedmiller, M. and Braun, H.: A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP algorithm, *Proc. IEEE Intl. Conf. on Neural Networks*, San Francisco, CA,

pp.586–591 (1993).
21) Tesauro, G.: TD-Gammon, A Self-teaching Backgammon Program, Achieves Master-Level Play, AAAI Press Technical Report FS93-02, pp.19–23 (1993).
22) Uno, T., Kiyomi, M. and Arimura, H.: LCM ver.3: Collaboration of Array, Bitmap and Prefix Tree for Frequent Itemset Mining, *Open Source Data Mining Workshop on Frequent Pattern Mining Implementations*, Chicago, IL (2005).
23) Utgoff, P.E. and Precup, D.: Constructive Function Approximation, *Feature Extraction, Construction and Selection: A Data Mining Perspective*, Liu, H. and Motoda, H. (Eds.), chapter 14, The Kluwer International Series in Engineering and Computer Science, Vol.453, Kluwer Academic Publishers (1998).

(平成 19 年 1 月 18 日受付)
(平成 19 年 9 月 3 日採録)

三輪　誠（学生会員）

1980 年生．2005 年 3 月東京大学大学院新領域創成科学研究科基盤情報学専攻修士課程修了．2005 年 4 月より東京大学大学院新領域創成科学研究科基盤情報学専攻博士課程在学中．コンピュータゲームプレイヤ，機械学習，データマイニングに興味を持つ．

横山　大作（正会員）

1974 年生．2000 年東京大学大学院工学研究科情報工学専攻修士課程修了．2002 年同博士課程中退．同年より東京大学大学院新領域創成科学研究科助手．2006 年同大学より博士号取得．博士（科学）．並列計算量理論，並列プログラミングライブラリ，およびゲームプログラミングに関する研究に従事．ソフトウェア科学会，IEEE-CS 各会員．

近山　隆（正会員）

1953 年生．1982 年 3 月東京大学大学院工学系研究科情報工学専門課程博士課程修了．同年 4 月富士通株式会社入社．同年 6 月（財）新世代コンピュータ技術開発機構に出向．1995 年 4 月東京大学大学院工学系研究科助教授，同教授を経て，1999 年 4 月より東京大学大学院新領域創成科学研究科教授．日本ソフトウェア科学会，人工知能学会，ACM，Association for Logic Programming 会員プログラム言語と処理系，並列分散処理，機械学習に興味を持つ．