情報処理学会論文誌 Vol. 52 No. 12 3828–3838 (Dec. 2011)

推薦論文

ゲーム構成要素を組み合わせた特徴の最適化

矢野友貴[†1,*1] 三輪誠[†2]
横山大作[†3] 近山隆[†1]

近年の計算資源の充足にともない，基礎的な特徴の組合せを用いた識別モデルが広く用いられるようになった．組合せ特徴は対象とする問題の知識によらず簡単に設計可能であるが，組合せ爆発を起こすために特徴数が膨大となり，高次の組合せ特徴を扱うことは現在の計算機では困難となっている．本稿では，組合せ特徴が利用される分野の 1 つであるコンピュータゲームプレイヤを題材に，基礎特徴間の関連性に注目して有効な組合せ特徴の絞り込みを行うことにより，従来よりも高次の組合せ特徴を活用する手法を提案する．将棋を例に既存手法と比較した結果，機械的に組合せを抽出した提案手法によって，既存評価関数と同程度の精度を維持しつつ，総特徴数を大きく削減することに成功した．

Optimizing Conjunctive Features of Game Components

Yuki Yano,[†1,*1] Makoto Miwa,[†2]
Daisaku Yokoyama[†3] and Takashi Chikayama[†1]

Since massive computing resources have been recently available with the progress of information processing, constructing discriminative models with features based on all possible combinations of primitive features has become worth considering. Constructing all possible combinations is easy, and does not require any deep knowledge of the target problem. However, combinatorial explosion results in a huge number of features, which is difficult to be handled efficiently even with massive computational resources. In this paper, we propose a new method to pick out effective features from high dimensional conjunctive features. Our method succeeds in greatly reducing the number of features without the reduction of accuracy in the game of shogi.

1. はじめに

与えられた問題に対して，コンピュータになんらかの判断を行わせる場合，その問題を表現するモデルを構築する必要がある．良いモデルを構築するためには，注目すべき問題の特徴を適切に選ぶことが重要となる．しかし，この選別は問題に対する深い知識が要求されるため困難である．問題に対する知識によらず精度の良いモデルを構築することは，人工知能の分野における大きな課題の 1 つである．このようなモデル構成の問題に対し，近年，問題に関する基礎的な特徴を組み合わせることでモデルを構築する手法が提案されている．このような単純なモデルが利用されるようになった背景には，計算資源の充足や機械学習手法の発展にともない，従来では扱うことが難しかった膨大な特徴量を扱えるようになったことがあげられる[1),2)]．

組合せ特徴は，問題の基礎的な特徴の組合せをすべて数え上げるだけで構成することが可能なため，対象とする問題に関する深い知識なしでも簡単にモデルの設計ができるという利点がある．組合せ特徴はその汎用性から幅広い分野で利用されており，高い精度が得られることが示されている[3),4)]．一方で組合せ特徴には，意味のない組合せも多く生成してしまう冗長性のために，組合せ爆発を起こすという問題点がある．組合せ爆発は，必要とするメモリ空間の増大，計算速度の悪化を招き，特に基礎的な特徴の総数が多い問題では，これらの理由から高次の組合せ特徴の利用が困難となっている．

組合せ特徴が利用される分野の 1 つとして，コンピュータゲームプレイヤがある．コンピュータゲームプレイヤでは，局面の優劣を判断する評価関数の構成に駒などのゲーム構成要素の組合せ特徴が有効であるため，得られた特徴の組合せの意味が比較的評価しやすい問題といえる．他方，コンピュータゲームプレイヤは特に時間に関する制約の厳しい分野であり，高次組合せ特徴の適用は非常に限定的な範囲にとどまっている．

†1 東京大学大学院工学系研究科
Graduate School of Engineering, The University of Tokyo
†2 マンチェスター大学コンピュータ科学科
School of Computer Science, The University of Manchester
†3 東京大学生産技術研究所
Institute of Industrial Science, The University of Tokyo
*1 現在，株式会社インターネットイニシアティブ
Presently with Internet Initiative Japan Inc.
本稿の内容は 2010 年 11 月のゲームプログラミングワークショップにて報告され，同プログラム委員長により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

本研究では，ゲームとして将棋を例に，前述の組合せ特徴の冗長性の緩和により組合せ爆発の抑制を実現する手法の提案を行う．将棋では，駒の位置関係が組合せ特徴として用いられるが，既存手法では最大でも一部の 3 駒間までの利用にとどまっている[5]．提案手法では，ゲーム構成要素間の関連度に注目することで冗長性を緩和し，さらにハッシュ関数により特徴の次元数の圧縮を行うことで組合せ爆発の抑制を図る．実験では，将棋における既存の組合せ特徴との比較を行い，その結果，精度を維持しつつ，総特徴数を半分以下に抑えることに成功した．

本稿では以降，2 章で関連研究について述べたのち，3 章で提案手法を，4 章で実験と結果について述べ，最後に 5 章でまとめと今後の課題について述べる．

2. 関 連 研 究

2.1 評価関数と組合せ特徴

評価関数とは，ゲーム中の各局面について有利不利の度合いを数値として計算する関数である．評価関数は多くの場合，局面 s から抽出される特徴ベクトル$\boldsymbol{\phi}(s)$ と各特徴の重要度を表す重みベクトル$\boldsymbol{w}$を用いて式 (1) のように構成される．

$$f(s) = \boldsymbol{w} \cdot \boldsymbol{\phi}(s) \tag{1}$$

近年では，局面の評価要素としてゲーム構成要素の組合せ特徴が広く用いられており，本稿で対象とする将棋においても組合せ特徴の利用に関する研究が行われている．

将棋の評価関数に駒による組合せ特徴を初めて利用した事例として，金子らの研究があげられる[4]．金子らは，2 駒の位置関係を用いることによって既存の評価要素の多くを表現可能であることを示しており，兄弟モデルによる学習を行うことで棋譜に近い指し手を生成することに成功している．一方で，金子らは大駒の長距離利きなど 2 駒間だけでは表現できない評価要素についても言及しており，将来的に 3 駒を用いることで精度の向上が期待されるが，提案が行われた時点では 3 駒の組合せ特徴をすべて用いることは組合せ爆発のために難しいと述べている．

組合せ特徴を利用した将棋プログラムの代表例として保木によって開発された「ボナンザ」がある[5]．ボナンザは第 16 回世界コンピュータ将棋選手権で優勝を収めるなど，現在のトップレベルの将棋プログラムの 1 つとなっている．ボナンザでは，玉を含む 3 駒の組合せ特徴を評価要素として利用しており，総特徴数は約 1 億個と膨大な数となっている．保木は，このような膨大なパラメータを調整するために，将棋の特性を活かした学習手法の提案を行っている[6]．この手法はボナンザメソッドとして知られており，多くの将棋プログラムで利用されている．

2.2 ハッシュ関数と次元圧縮

配置パターン，文字列，部分木といった特徴を独立した次元として組み合わせると，次元数が非常に多くなり，すべてをそのままメモリ上に展開することは現実的ではない．このような空間コストの問題に対して，ハッシュ関数を用いて次元圧縮を行う手法が提案されている．

Silver らは囲碁において，碁石の配置パターンのハッシュ値を利用した評価関数の学習を行っている[7]．Silver らの研究では，5×5 までのパターンに対して，4×3 以上のパターンを Zobrist Hash[8] でハッシュ値に変換することで膨大なパターンの利用を可能にしている．9 路盤において強化学習を用いて評価関数を調整し，CGOS で対戦をした結果，Elo rating で+1,210 [⋆1]を得ることに成功したと述べている．

ハッシュ関数の用いた次元圧縮に関する他の研究としては，Ganchev らによる Random Feature Mixing[9]，Shi らによるハッシュカーネル[10] などがあげられる．

2.3 有効パターンの選別

組合せ特徴では可能な組合せをすべて展開するため，特徴が非常に冗長となる．このような冗長性の問題に対し，特徴選択[11] を行うことで計算の高速化を図る研究が行われている．

特徴の重みを利用して特徴の絞り込みを行った研究としては，Kudo らによる多項式カーネルの高速化があげられる[3]．Kudo らは，正例，負例での各特徴の出現頻度から重みの推定を行い，PrefixSpan[12] を利用して有効な組合せ特徴を選別する手法の提案を行っている．Kudo らは，単純な多項式カーネルとの比較実験の結果，精度を維持しつつ，最大で 300 倍の高速化に成功したと述べている．

統計情報を利用して特徴の絞り込みを行った研究としては，Suzuki らによる木カーネルの枝刈りがあげられる[13]．Suzuki らは，訓練データから各部分木のカイ二乗値を計算し，カイ二乗値が一定値以上の部分木のみを展開することで，木カーネルの計算の最適化を行っている．実験の結果，従来では過学習を起こすような巨大な部分木においても，提案手法では安定した精度が得られることが示されている．

Kudo らの手法[3] および Suzuki らの手法[13] と提案手法は有効なパターンを選別するという点は同様であるが，その選別基準の算出方法が大きく異なる．2 つの関連研究では，特徴選択の基準として訓練データ中の可能な組合せ特徴を展開することで厳密な特徴選択を

⋆1 実験時ではランダムプレイヤが −170，最も強いプレイヤが +1,860.

行っているのに対し，提案手法では 3 章で述べるように，組合せ特徴のサブパターンのみに注目し，選別基準を近似値として算出することで事前計算のコストの削減を図っている．

3. 提 案 手 法

既存のコンピュータゲームプレイヤの評価関数では，単純にすべての組合せを展開した組合せ特徴を利用している．そのため，計算機の制約から利用可能な組合せ数が大きく制限される．将棋における駒情報を用いた組合せ特徴の場合，ほとんどのコンピュータゲームプレイヤは 2 要素間までしか用いておらず，3 要素以上の利用も将棋プログラム「ボナンザ」のように限られた範囲とせざるをえない．より精度の高い評価関数を作成するためには，局面のより詳細な特徴を活用する必要がある．そのため，従来よりも高次の組合せ特徴を活用できれば，精度向上が得られる可能性は高いといえる．

本研究では，棋譜から得られる情報をもとに組合せ特徴の絞り込みを行うことで，組合せ特徴の選別を行う手法を提案する．具体的には，**図 1** のように盤面を駒や升で構成された重み付きグラフであると見なし，各エッジの重みに応じて組合せ特徴を選択する．なお，グラフの各ノードには駒や升の状態に応じたラベルを付与する．盤面グラフでは，組合せ特徴はグラフ上のパスとして表現される．

盤面をグラフと考えた場合，組合せ特徴のすべてを利用することは盤面をゲーム構成要素を全対全でつないだ完全グラフと考え，すべてのパスを展開することに相当する．しかし，グラフのエッジの数を $|E|$，パスの最大サイズを d とすると，グラフ中のパス，つまり組合せ特徴の総数は $\mathcal{O}\left(|E|^{d-1}\right)$ と巨大なものになる[⋆1]ため，すべてのパスを展開することは困難である．これは，従来の組合せ特徴が要素間のつながりを同等に扱うために，無駄な組合せ特徴を生成していることが原因であると考えられる．そこで，提案手法では各エッジ e_k に結合度 $con(e_k)$ を定義し，それに基づいてパスの枝刈りを行う．ここで，結合度とは各ノードのつながりが局面評価においてどれほど重要であるかを表す指標であり，$[0.0, 1.0]$ の範囲の値をとる．具体的には，式 (2) のようにパス（$p = \{e_1, e_2, \cdots, e_n\}$）が経由したエッジの結合度の積をパスの重要度 $imp(p)$ とし，**図 2** のように重要度があらかじめ設定した閾値を下回ったパスを枝刈りすることで組合せ特徴の選別を行う．なお，図 2 中の threshold は重要度の閾値を，max length は組合せ数の最大値を表す．結合度は棋譜から学習することで導出する．

$$imp(p) = con(e_1) \times con(e_2) \times \cdots \times con(e_n) \tag{2}$$

高次の組合せ特徴を用いる場合，最終的に出現する組合せ数がメモリに収まりきらない可能性がある．また，上記の枝刈りにともないパスの長さが不均一になるため，各組合せ特徴にどのようにインデックスを振るかが問題となる．これらの問題に対し，提案手法で

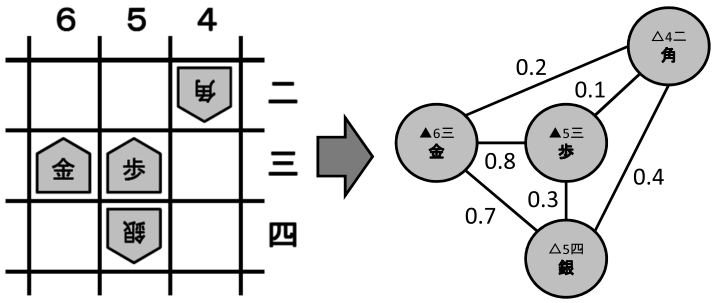

図 1　盤面のグラフ表現
Fig. 1　Representation of the board as a graph.

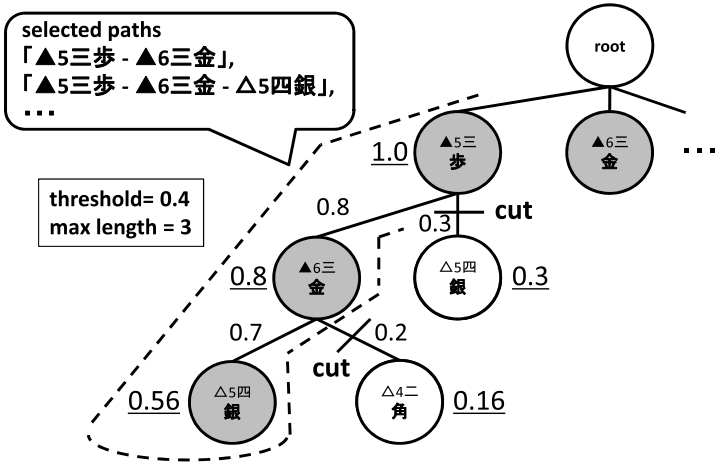

図 2　パスの展開と枝刈り
Fig. 2　Expanding and pruning paths.

⋆1 実際には重複分があるが，将棋では $|E| = 40$ に対し $d = 3$ 程度と小さいため $\mathcal{O}\left(|E|^{d-1}\right)$ とした．

は Zobrist Hash[8] を用いたインデックス付けを行う．具体的には，盤面グラフ上の各ノードに対して乱数を振り，式 (3) のようにパス p 上の各要素 n の乱数 $r(n)$ の XOR を計算することでハッシュ値を求める．ハッシュを用いることにより，あらかじめ特徴の総数をハッシュのサイズに決定することができ，また式 (3) を用いることによって各パスのインデックスを容易に計算可能となる．

$$h(p) = r(n_1) \oplus r(n_2) \oplus \cdots \oplus r(n_{|p|}) \tag{3}$$

評価関数の重みベクトルを $\boldsymbol{w}$，局面 s での盤面のグラフを $G(s)$，パス p 上のノード数を $|p|$，パス p の重要度を $imp(p)$ としたとき，提案手法での評価関数 $f(\boldsymbol{w}, s)$ は式 (4) のようになる．

$$f(\boldsymbol{w}, s) = \boldsymbol{w} \cdot \boldsymbol{\phi}(s) = \sum_i w_i \phi_i(s) \tag{4}$$

$$\text{where} \quad \phi_i(s) = \sum_{\substack{p \in G(s), 1 < |p| \le d, h(p) = i \\ imp(p) \ge \text{threshold}}} \lambda^{|p|-1}$$

ここで，d は最大組合せ数，λ は高次組合せ特徴の影響を制御する減衰定数である．式 (4) において，パスの長さ $|p|$ の範囲を $1 < |p| \le d$ と制限しているのは，多数の要素を同時に考えることは計算コストのうえで不可能なためである．評価関数計算の擬似コードを**図 3** に示す．図 3 中の get_sorted_edge_list(*graph*, *node*) は *graph* 中の *node* を始点とするエッジの集合を結合度の大きい順に返す関数であり，*imp* は現在のパスの重要度を，con(*edge*) は *edge* の結合度を表している．なお，本稿の実装では重複するパスは以下のように扱う．

- パス中のノードの組合せおよび順番が完全に一致するパス（左右対称のパス）はカウントしない．
- パスは異なるがハッシュ値が同じパス（ノードの順番は異なるが組合せが同じパス）は重複してカウントする．

ハッシュ値が同じパスを重複してカウントすることとなるが，これは本稿では複数の重複するパスで重要度の高い組合せ特徴はより重要な特徴であると仮定しているためである．

```
// initialize variables
eval ← 0
graph ← current_board_graph
hash ← 0
imp ← 1

calc_eval(eval, graph, path, hash, imp)
    // check path length
    if len(path) ≥ max_length
        return
    endif

    // check all edges from the last node
    sorted_edge_list ← get_sorted_edge_list(graph, tail(path))
    for all edge ∈ sorted_edge_list

        // get next_node and check next_node has been already checked
        next_node ← next_node(edge)
        if is_checked(next_node)
            continue
        end if

        // update importance of path and compare with the threshold
        next_imp ← imp × con(edge)
        if next_imp < threshold
            break
        end if

        // check next_node, update path and hash
        check(next_node)
        push_back(path, next_node)
        hash ← hash ⊕ rand(next_node)

        // update evaluate value and call calc_eval recursively
        if len(path) > 1
            eval ← eval + λ^(len(path)−1) × weight(hash)
        end if
        calc_eval(eval, graph, path, hash, new_imp)

        // restore path and hash, uncheck next_node
        hash ← hash ⊕ rand(next_node)
        pop_back(path)
        uncheck(next_node)
    end for
end
```

図 3　提案手法の評価関数計算の擬似コード
Fig. 3　Pseudocode of the evaluation function.

4. 実験と結果

4.1 実 験 方 法

提案手法の有効性を評価するために，将棋プログラム「激指[14]」を用いて実験を行った．激指は，第 20 回世界コンピュータ将棋選手権において優勝を収めるなど，トップレベルの

強さを誇る将棋プログラムである．本実験では，第 20 回世界コンピュータ将棋選手権でのバージョンを利用した．評価関数として単純な駒割[⋆1]に提案手法や比較手法の特徴を追加したものを用意し，それぞれに対して学習を行うことで性能評価を行った．なお，実験では駒割を固定の値として扱い，パラメータの調整は追加部分に限定した．また，組合せ特徴の要素は盤面上の駒に限定し，持ち駒は用いないこととした．

実験では，4.5 節で述べるように提案手法と比較を行う既存手法としてボナンザで利用されている kkp-kpp [⋆2]を激指上に実装したものを用いた．kkp-kpp を実装した激指を既存手法と見なすことの根拠は以下のとおりである．

- 既存の将棋プログラムの評価関数の中で，最も多くの組合せ特徴を利用しており，かつ高い実績を持つ評価関数がボナンザの kpp-kpp であること．
- 激指は将棋のベースプログラムとして用いているのみであり，評価関数自体は既存手法である kkp-kpp と同等であること．

学習手法としては激指の方法[15] を用いた．激指では，ボナンザメソッド[6] および Averaged Perceptron [16] をベースにした学習手法が用いられている．ボナンザメソッドは，棋譜の手筋に沿うような評価関数を得る学習手法で，具体的には各局面において棋譜の手が他の手に比べて相対的に高く評価されるようパラメータの調整を行う．Averaged Perceptron はオンライン学習の一種で，学習途中の各ステップでのパラメータの総和を保持し，それを総ステップ数で割った値を最終的なパラメータとする手法である．Averaged Perceptron ではパラメータの平滑化を行うことで，ノイズに強い学習を実現している．ステップ t での具体的なパラメータ $\boldsymbol{w}^t$ の更新は式 (5) のとおりである．

$$\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t + \frac{1}{|M|} \sum_{m_j \in M} y\left(\boldsymbol{\phi}(s'_1) - \boldsymbol{\phi}(s'_j)\right) \tag{5}$$

ここで，y は局面の手番に応じたラベル[⋆3]，s'_j は j 番目の合法手 m_j の後からの最善応手手順後の局面（m_1 は棋譜の手），M は式 (6) を満たす合法手 m_j の集合である．

$$y\left(\boldsymbol{w}\cdot\boldsymbol{\phi}(s'_1) - \boldsymbol{w}\cdot\boldsymbol{\phi}(s'_j)\right) < margin \tag{6}$$

式 (6) は合法手 m_j に比べて棋譜の手 m_1 を相対的に悪く評価したことを表す．なお，$margin$ は棋譜の手を相対的にどれだけ良く評価すべきかを決める閾値で，終盤になるほど大きな値が設定される．

実験では学習用に 30,000 棋譜，テスト用に 250 棋譜のプロの対局の棋譜を用意した．また学習時には，速度向上のために各局面ではすべての合法手を比較対象とするのではなく，棋譜の手以外はランダムに 16 手のみ選び学習を行った．

4.2 評価基準

実験では評価基準として以下の 3 つを用いた．

(1)　テスト用の 250 棋譜に対する一致率（%）
(2)　テスト用の 250 棋譜に対する不一致度
(3)　ノード数制限による対戦結果

ここで，一致率は式 (7)，不一致度は式 (8) のように定義した．

$$\text{一致率 (\%)} = \frac{\text{探索した手と棋譜の手が一致した局面数}}{\text{棋譜中の全局面数}} \times 100 \tag{7}$$

$$\text{不一致度} = \frac{1}{N}\sum_{i=1}^{N}\sum_{m_j \in M_i} T(\boldsymbol{w}\cdot\boldsymbol{\phi}(s'_1) - \boldsymbol{w}\cdot\boldsymbol{\phi}(s'_j)) \tag{8}$$

式 (8) の N は棋譜中の全局面数，M_i は局面 i での棋譜の手以外の合法手の集合，$T(x)$ は式 (9) で定義されるシグモイド関数である．

$$T(x) = \frac{1}{1+\exp(3x/100)} \tag{9}$$

一致率が大きいほど精度が良いことを表すのに対し，不一致度は小さいほど精度が良いことを表す．

対戦は双方はじめの 30 手を固定とし，その後は 1 手あたりの探索ノード数を最大 50 万ノードに制限して最善手の探索を行わせた．また，対戦で用いる初期局面は，学習で用いなかった棋譜のうち，30 手目において激指の評価関数での先手後手の優劣の差が 50 以下[⋆4]となっており，それぞれ 30 手目での駒の配置が異なる 250 局面を利用した．評価のための対戦は各初期局面を先手後手入れ替えて計 500 局行い，250 手目以降で自分が不利である場合[⋆5]には投了するように設定した．なお，対戦結果を示す際，有意水準 5%の 2 項検定で有意な差のついた結果[⋆6]は太字で表記し，さらに有意に勝ち越した結果には下線をつけている．

⋆1 駒に対する価値，たとえば盤面上に歩が 1 枚あれば 100 点．
⋆2 2 玉と 1 駒（king king piece，kkp）および 1 玉と 2 駒の関係（king piece piece，kpp）．
⋆3 先手なら +1，後手なら −1．
⋆4 評価値が ±50 以内，この数値は歩 1 枚の点数である 100 点から決定した．
⋆5 先手ならば局面の評価値が負であった場合，後手ならば評価値が正であった場合．
⋆6 具体的には，500 戦中 273 勝以上もしくは 227 勝以下．

4.3　結合度の学習

3 章で述べた各エッジの結合度を導出するために，各エッジの結合度の導出およびそれに対する予備評価を行った．具体的には，グラフのエッジを特徴とする線形分類器を考え，それをロジスティック回帰で学習することで各エッジの結合度を導出した．具体的な特徴は 2 駒の位置関係，計 2,570,778 特徴量とした．学習で用いる訓練データは学習用の 30,000 棋譜から以下の手順によって作成した．

(1)　棋譜中の各局面について，棋譜の手とそれ以外の合法手の直後の局面のペア $\{s_1, s_j\}$ をランダムに 1 つずつ抽出する．

(2)　各ペアについて，それぞれ局面の特徴ベクトル（= エッジ）の差 $\boldsymbol{x} = \boldsymbol{\phi}(s_1) - \boldsymbol{\phi}(s_j)$ を計算し，訓練データとする．

(3)　各訓練データのラベル y について，元の局面が先手なら $y = +1$（正例），後手なら $y = -1$（負例）とする．

(1) において，各局面から抽出する手のペアを 1 つに絞っているのは，訓練データが大きくなりすぎることを防ぐためである．30,000 棋譜から得られたデータファイルのサイズは 7.5 GB となった．

ロジスティック回帰による学習には LIBLINEAR [17] を用い，具体的には式 (10) の最適化を行った．

$$\min_{\boldsymbol{w}_e} \quad \frac{1}{2}\|\boldsymbol{w}_e\|_2^2 + C \sum_{(\boldsymbol{x}_i, y_i) \in S} \log\left(1 + e^{-y_i \boldsymbol{w}_e \cdot \boldsymbol{x}_i}\right) \tag{10}$$

S は訓練データの集合，C は 2 つの項の影響度を制御するパラメータである．本実験では $C = 1$ とした．結合度の学習には Intel Xeon X5680（3.33 GHz），メモリ 96 GB のマシンを用いて 62 分の時間を要し，全体の 55%の特徴に重みがついた．得られた重みは式 (11) を用いて結合度に変換した．なお，$con(e_k)$ と w_{e_k} はそれぞれエッジ e_k に対応する結合度，重みを表す[⋆1]．式 (11) より，結合度は $0.0 < con(e_k) < 1.0$ の範囲の値となる．

$$con(e_k) = \frac{1}{1 + \exp(-w_{e_k})} \tag{11}$$

ロジスティック回帰を用いて式 (11) のように結合度を計算した場合，結合度は以下のように 2 通りのとらえ方ができる．

⋆1 w_{e_k} は式 (10) 中の $\boldsymbol{w}_e$ の k 番目の要素でもある．

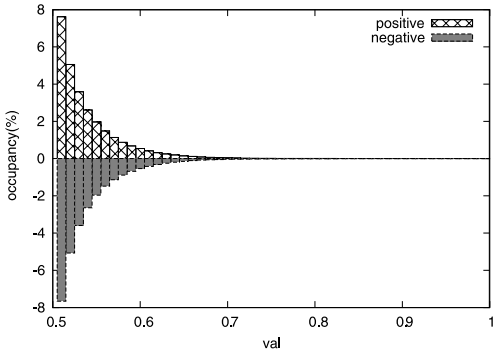

図 4　結合度の分布
Fig. 4　Distribution of defrees of connections.

- 正例（先手）側から見た結合度 $con_{pos}(e_k) = con(e_k)$
- 負例（後手）側から見た結合度 $con_{neg}(e_k) = 1 - con(e_k)$

このため，本実験では各パスの重要度を正例側，負例側の 2 つの結合度を用いて別々に計算し，値の大きい方をそのパスの重要度として採用することとした．得られた結合度について，$con(e_k) = 0.5$ (i.e. $w_{e_k} = 0$) となるエッジを除外した場合の結合度の分布は**図 4** のようになった．

得られた結合度を用いて，ボナンザのパターンでのカバー率の評価を行った．実験では，ボナンザのパターンをその重みの絶対値が大きい順にソートし，各パターンの重要度が閾値を超えるかを上から順に調べた．閾値を 0.4 とした場合のカバー率の推移を**図 5** に示す．図 5 の x 軸は何位までパターンをチェックしたか，y 軸はチェックしたパターン全体でのカバー率を表している．グラフを見ると，カバー率の推移が右肩下がりになっていることが分かる．これは，結合度を用いることによって重要度の高い上位のパターンを重点的に展開し，重要度の低い下位のパターンを枝刈りできていることを示す結果であり，提案手法によって有効パターンをうまく抽出できているといえる．他方，ボナンザのパターンのカバー率の最大値は 25% 程度にとどまっているが，これは提案手法において 2 駒間の結合度から 3 駒間のパターンの重要度を近似していることが影響していると考えられる．

次に，得られた結合度を用いて具体的なパターンの抽出を行った．具体的には，テスト用の棋譜を含む 500 棋譜中の 30 手から 70 手までの局面から 4 駒間のパターンを抽出し，重要度の高い上位 1,000 パターンを選択した．得られたパターンのうち，重要度が最も高かっ

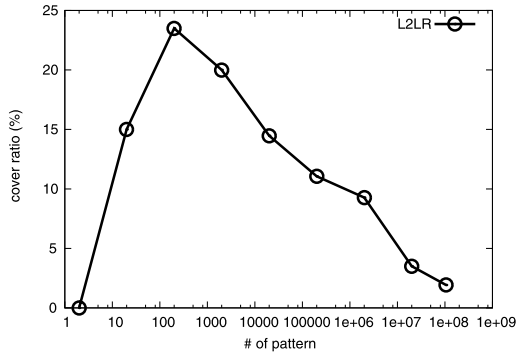

図 5　ボナンザのパターンのカバー率
Fig. 5　Pattern coverage against Bonanza.

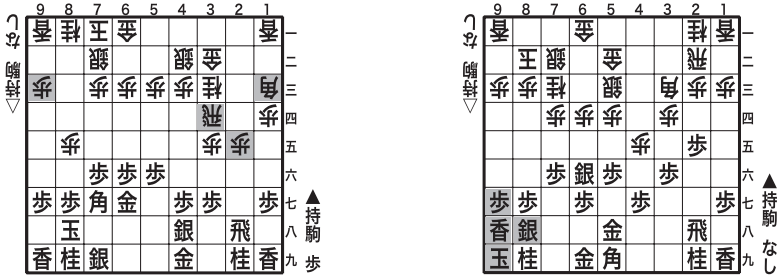

図 6　具体的なパターン（左：重要度が最大，右：評価値が最大）
Fig. 6　Pattern examples (left: estimated to be the most important, right: with the highest score).

たパターンおよび評価値が最も高かったパターンを**図 6** に示す．図 6 を見ると，重要度が最も高いパターンは△ 1 三角と△ 3 四飛のように部分的に意味のありそうなパターンはとれているものの，全体では意味のあるパターンとはなっていない．これは，ボナンザとのカバー率で述べたように，提案手法が 2 駒の結合度から 3 駒以上の重要度の近似を行っているため，重要度の高いパターンが必ずしも意味のあるパターンになるとは限らないためである．高次の組合せにおいても整合性のとれる結合度の導入は今後の課題の 1 つといえる．一方，評価値が最も高いパターンは穴熊囲いの形となっており，意味のあるパターンとなっていることが分かる．

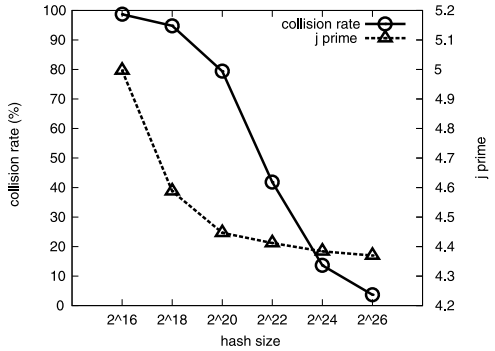

図 7　衝突率と精度の関係
Fig. 7　Correlation of collision rates and accuracy.

4.4　ハッシュサイズと精度

特徴のインデックス付けで利用するハッシュのサイズが精度に与える影響と将棋におけるハッシュ利用の有効性を評価するために，ハッシュサイズと精度の関係について実験を行った．衝突率（$coll\_rate$）の計算は，局面評価で出現する特徴の種類の総数（$\#feature$）とそれに対応するハッシュの種類の総数（$\#hash$）を用いて式 (12) のように行った．

$$coll\_rate(\%) = \frac{\#feature - \#hash}{\#feature} \times 100 \tag{12}$$

実験では，あらかじめ激指によって最善応手手順を求めた 500 棋譜を用意し，ハッシュサイズを 2^{16}，2^{18}，2^{20}，2^{22}，2^{24}，2^{26} とした 6 種類の提案手法の学習を行い，衝突率と不一致度の関係，各ハッシュサイズでの強さ比較を行った．なお，提案手法の減衰定数は 0.8，閾値は予備実験により 0.4．組合せ特徴の最大要素数は 4 とした．また，精度の基準には 4.2 節で述べた不一致度を用いた．

衝突率と精度の関係は**図 7** のようになった．なお，図 7 の縦軸である j_prime は不一致度を表す．図 7 を見ると，ハッシュサイズが 2^{20} 以下の範囲では精度が大きく悪化しているのに対し，2^{20} 以上では精度を維持できていることが分かる．また，2^{20} での衝突率より，本実験の条件では衝突率が 80%近くあっても精度をある程度維持できていることが読み取れる．なお，4.5 節で述べる既存手法 kkp-kpp，atkk においても同様に衝突率 80%が精度維持の境目となる結果が得られている（詳細は付録 A.1 を参照されたい）．

異なるハッシュサイズで学習を行った提案手法間の相互対戦の結果を**表 1** に示す．相互

表 1　各ハッシュサイズでの相互対戦結果
Table 1　Won-lost records between different hash sizes.

自分 \ 相手	2^{16}	2^{18}	2^{20}	2^{22}	2^{24}	2^{26}
2^{16}		**36.0**	**28.2**	**27.6**	**25.2**	**22.0**
2^{18}	**64.0**		**42.6**	**44.2**	**38.6**	**41.8**
2^{20}	**71.8**	**57.4**		46.6	45.6	49.2
2^{22}	**72.4**	**55.8**	53.4		47.4	49.6
2^{24}	**74.8**	**61.4**	54.4	52.6		49.2
2^{26}	**78.0**	**58.2**	50.8	50.4	50.8	

対戦は 4.2 節で述べた条件によって行った．表 1 を見ると，評価関数のみの精度の観点からの強さにおいても衝突率と精度と同様に 2^{20} が精度維持の境目となっていることが分かり，ハッシュサイズが 2^{20} 以上では多少の差はあるものの優位な差はみられない．この結果から，衝突率がある閾値を下回っていれば，評価関数のみの精度の観点からの強さに関しても悪化を抑制することができるといえる．

4.5　既存手法との比較

提案手法の有効性を評価するために，既存手法との比較を行った．実験は次の 4 手法を用いて行った．このうち，atkk は既存のヒューリスティクスによって特徴選択を行った提案手法に相当し，減衰定数を用いて低次の組合せ特徴の影響力を相対的に大きく，高次の組合せ特徴の影響度を相対的に小さくしている点が all-two，kkp-kkp と異なる．

- 2 駒の組合せ特徴（all-two）
- 玉を含む 3 駒の組合せ特徴（kkp-kpp）
- 2 駒の組合せ特徴 + 玉を含む 3 駒の組合せ特徴（atkk）
- 提案手法による 4 駒の組合せ特徴（proposed）

本節の実験では既存手法についても減衰定数を導入し，式 (4) のように，2 要素間の組合せ特徴には λ の減衰を，3 要素間の組合せ特徴には λ^2 の減衰を与えるものとした．また，3 要素間の組合せ特徴を利用する kkp-kpp，atkk については proposed と同様にハッシュを利用した．なお，減衰定数は予備実験によって最適値の推定を行い，all-two と kkp-kpp は 1.0，atkk と proposed は 0.8 とした．また，ハッシュサイズは 2^{24} とした．

各手法での学習結果を**表 2** に示す．表中の学習時間および速度は Intel Xeon X5560 (2.8 GHz)，メモリ 24 GB の環境において，学習時は 8 スレッドによる並列計算で，速度はシングルスレッドで実行した結果を表す．また，有効特徴数は非零の特徴数を表す．表 2 を見ると，2 要素の組合せ特徴のみを用いる all-two に比べ，3 要素以上の組合せ特徴も併用して用いる atkk，proposed の 2 つの手法が一致率，不一致度ともに上回っていることが分かり，高次組合せ特徴を利用することの有効性がうかがえる．一方で，3 要素の組合せ特徴のみを用いた kkp-kpp は他の手法に比べて悪い結果となっている．この原因としては，3 要素というスパースな特徴のみを用いて局面評価を行うため，学習時に出現しなかった特徴が含まれる局面評価がうまくいっていない可能性が考えられる．atkk，proposed の 2 つに注目すると，精度面では同程度となっているものの，最終的な特徴数は proposed が atkk の半分以下にとどまっており，提案手法によってより少ない特徴で既存手法と同程度の精度が得られていることが分かる．これは，提案手法での組合せ特徴の選別がうまく働いていることを示す結果である．他方，速度では提案手法は既存手法に比べ劣る結果となっている．これは，本稿での提案手法の実装では重複する組合せ特徴をまだ完全には排除できていないため，グラフの構築，パス展開に大きな計算コストがかかるためである．グラフ関連の処理の改善による速度の向上は今後の課題の 1 つといえる．

表 2　各手法での学習結果
Table 2　Learning results of different methods.

手法	学習時間	速度（$\times 10^5$ nps $^{\star 1}$）	有効特徴数	一致率（%）	不一致度
all-two	7 h 54 m	3.00	1,569,828	40.2	4.51
kkp-kpp	14 h 55 m	1.82	15,607,513	39.1	4.70
atkk	19 h 21 m	1.47	15,642,643	40.8	4.39
proposed	18 h 54 m	1.15	6,194,174	40.8	4.38

表 3　各手法での相互対戦結果
Table 3　Won-lost records between players with different methods.

自分 \ 相手	all-two	kkp-kpp	atkk	proposed	順位
all-two		**59.6**	**44.6**	50.4	3
kkp-kpp	**40.4**		**34.4**	**36.2**	4
atkk	**55.4**	**65.6**		51.8	1
proposed	49.6	**63.8**	48.2		2

次に，各種法での相互対戦の結果を**表 3** に示す．ここで，対戦結果の順位は有意に勝ち越した数と負け越した数の差を基準に決めており，差が同じ手法については直接対決の勝率をもとに順位付けを行った．表 3 を見ると，提案手法である proposed が他の手法に対し

$\star$1 nodes per sec，単位秒あたりの探索ノード数

て有意に勝ち越している atkk と互角の結果を残していることが分かり，既存手法に比べ評価関数のみの精度の観点からも同程度以上の強さを得ていると考えられる．有効特徴数で proposed が勝っている点を考慮すると，本節の実験結果から，機械的に組合せ特徴の選別を行った提案手法によって，既存手法の組合せ特徴と互角以上の成果をあげることに成功したといえる．

5. おわりに

本稿では，ゲーム構成要素間の関連度を用いて組合せ特徴の選別する手法の提案を行った．実験では，将棋を例に単純な組合せ特徴を用いた従来手法との比較実験を行った．実験の結果，結合度による特徴の選別およびハッシュ関数による次元圧縮を組み合わせた提案手法により従来では用いることが困難であった 4 駒間の組合せ特徴の利用を実現し，また，従来手法に比べ精度を維持しつつ，総特徴数を半分以下に削減することに成功した．本稿の実験では，対象を将棋に絞って評価を行っているが，提案手法はゲームに限らず一般的な組合せ特徴にも適用可能である．本実験の結果は要素間のつながりによる組合せ特徴の生成手法の有効性を示す結果でもあり，今後の高次組合せ特徴の利用の発展が期待される．

今後の課題としては，実行速度の向上，複数要素間での重要度の近似精度の向上があげられる．速度の向上に対しては，Yoshinaga らの手法[18] のように頻出パターンを事前計算する方法が考えられる．一方，近似精度の向上では，近似によって求めた重要度と実際に学習される重みとの順位付けの差を縮めるように補正をすることで，より良い精度を得られる可能性が高いといえる．

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

参考文献

1) Gao, J., Andrew, G., Johnson, M. and Toutanova, K.: A Comparative Study of Parameter Estimation Methods for Statistical Natural Language Processing, *Proc. 45th Annual Meeting of the Association of Computational Linguistics*, Prague, Czech Republic, Association for Computational Linguistics, pp.824–831 (2007).
2) Yuan, G.-X., Chang, K.-W., Hsieh, C.-J. and Lin, C.-J.: A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification, *Journal of Machine Learning Research*, Vol.11, pp.3183–3234 (2010).
3) Kudo, T. and Matsumoto, Y.: Fast Methods for Kernel-based Text Analysis, *ACL'03: Association for Computational Linguistics*, Association for Computational Linguistics, pp.24–31 (2003).
4) 金子知適，田中哲朗，山口和紀，川合　慧：駒の関係を利用した将棋の評価関数，第 8 回ゲームプログラミングワークショップ，pp.14–21 (2003).
5) 保木邦仁：Bonanza – The Computer Shogi Program，ジオシティーズ (online), available from ⟨http://www.geocities.jp/bonanza_shogi⟩ (accessed 2010-12-10).
6) 保木邦仁：局面評価の学習を目指した探索結果の最適制御，第 11 回ゲームプログラミングワークショップ，pp.78–83 (2006).
7) Silver, D., Sutton, R. and Müller, M.: Reinforcement Learning of Local Shape in The Game of Go, *IJCAI'07: Proc. 20th International Joint Conference on Artifical Intelligence*, pp.1053–1058, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2007).
8) Zobrist, A.: A New Hashing Method with Application for Game Playing, Technical Report 88, Univ. of Wisconsin (1970).
9) Ganchev, K. and Dredze, M.: Small Statistical Models by Random Feature Mixing, *Workshop on Mobile Language Processing*, p.19 (2008).
10) Shi, Q., Petterson, J., Dror, G., Langford, J., Smola, A. and Vishwanathan, S.: Hash Kernels for Structured Data, *Journal of Machine Learning Research*, Vol.10, pp.2615–2637 (2009).
11) Guyon, I. and Elisseeff, A.: An Introduction to Variable and Feature Selection, *Journal of Machine Learning Research*, Vol.3, pp.1157–1182 (2003).
12) Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U. and Hsu, M.: PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth, *Proc. 17th International Conference on Data Engineering*, pp.215–224, IEEE Computer Society (2001).
13) Suzuki, J. and Isozaki, H.: Sequence and Tree Kernels with Statistical Feature Mining, *Advances in Neural Information Processing Systems*, pp.1321–1328, MIT Press (2005).
14) 近山・田浦研究室：将棋プログラム「激指」のページ，近山・田浦研究室（オンライン），入手先⟨http://www.logos.ic.i.u-tokyo.ac.jp/~gekisashi⟩（参照 2010-05-30）.
15) 鶴岡慶雅：選手権優勝記—激指の技術的改良の解説，情報処理，Vol.51, No.8, pp.1001–1007 (2010).
16) Collins, M.: Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms, *EMNLP'02: Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp.1–8 (2002).
17) Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R. and Lin, C.-J.: LIBLINEAR: A Library for Large Linear Classification, *Journal of Machine Learning Research*,

Vol.9, pp.1871–1874 (2008).
18) Yoshinaga, N. and Kitsuregawa, M.: Polynomial to Linear: Efficient Classification with Conjunctive Features, *EMNLP'09: Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp.1542–1551 (2009).

付　　録

A.1　既存手法におけるハッシュサイズと精度

4.5 節で述べた kkp-kpp，atkk の 2 つの手法について，ハッシュを用いることによる精度への影響を評価するために，4.4 節での衝突率と不一致度に関する実験を行った．具体的には，4.4 節で提案手法に行ったものと同様の実験方法を用い，kkp-kpp，atkk におけるハッシュサイズ以外の条件には 4.5 節と同様のものを用いた．

kpp-kpp，atkk における衝突率と精度の関係はそれぞれ**図 8**，**図 9** のようになった．図 8，図 9 より，kkp-kpp，atkk においても提案手法と同様にハッシュサイズが 2^{20} 以上では精度を維持できており，また衝突率 80%が精度維持の境目となっていることが読み取れる．

(平成 23 年 4 月 20 日受付)
(平成 23 年 9 月 12 日採録)

推　薦　文

この論文は，ゲームプログラミングの重要なテーマである評価関数の機械学習において，特徴の高次の組合せと絞り込みを提案し，効果を示したものである．近年，評価関数の学習について多くの研究がなされてきたが，効果的な特徴をどのように用意するかは未解決の問題であり，この成果は重要である．よって，論文誌の推薦論文としてふさわしいものである．

（ゲームプログラミングワークショッププログラム委員長　鶴岡慶雅，金子知適）

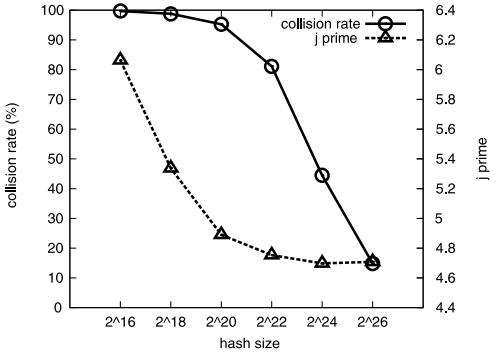

図 8　kkp-kpp における衝突率と精度の関係
Fig. 8　Correlation of collision rates and accuracy in kkp-kpp.

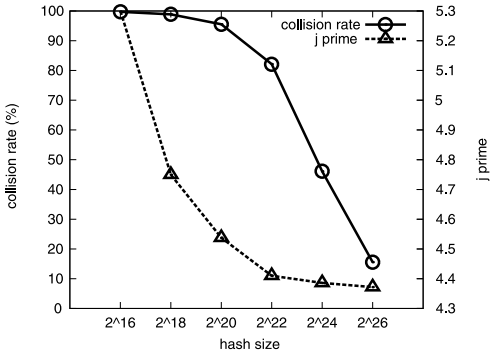

図 9　atkk における衝突率と精度の関係
Fig. 9　Correlation of collision rates and accuracy in atkk.

矢野　友貴（正会員）

2009 年東京大学工学部電子情報工学科卒業．2011 年同大学大学院工学系研究科電気系工学専攻修士課程修了．同年株式会社インターネットイニシアティブ入社．現在ソフトウェア開発に従事．

三輪　　誠（正会員）

1980 年生．2008 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博士課程修了．博士（科学）．同年 4 月より同大学大学院情報理工学系研究科特任研究員を経て，2011 年 4 月より英国マンチェスター大学コンピュータ科学科リサーチアソシエイト．言語処理学会，人工知能学会各会員．自然言語処理，機械学習，コンピュータゲームプレイヤに興味を持つ．

横山　大作（正会員）

2006 年東京大学より博士号取得．博士（科学）．2002 年より同大学大学院新領域創成科学研究科助手等を経て，2009 年より同大学生産技術研究所助教，現在に至る．並列プログラミング環境，ゲームプログラミングに関する研究に従事．

近山　　隆（正会員）

1977 年東京大学工学部卒業，1982 年同大学大学院工学系研究科博士課程修了・工学博士．同年より（財）新世代コンピュータ技術開発機構において第五世代コンピュータプロジェクトの研究開発に従事，1995 年東京大学大学院工学系研究科助教授，1996 年同教授，以降学内の異動を経て 2008 年 4 月より工学系研究科電気系工学専攻教授．