

# 駒の関係を利用した将棋の評価関数

## Evaluation Functions Based on Pairs of Pieces

金子 知適<sup>\*†</sup>      田中 哲朗<sup>‡</sup>      山口 和紀<sup>‡</sup>      川合 慧<sup>†</sup>  
T. Kaneko      T. Tanaka      K. Yamaguchi      S. Kawai

東京大学総合文化研究科<sup>†</sup> 東京大学情報基盤センター<sup>‡</sup>  
Graduate School of Arts and Sciences, The University of Tokyo<sup>†</sup>  
Information Technology Center, The University of Tokyo<sup>‡</sup>

### 概要

将棋の評価関数として、駒(所有者,種類,位置を含む)の2項関係を評価するモデルと,棋譜を使った自動的な調整方法を提案する.将棋の評価関数では駒の損得を基本とした上で,駒の働きや玉の危険度など様々な評価項目が用いられている.しかしそのような駒の損得以外の評価項目は設計が難しく,また自動的な値の調整も困難であった.本稿では,駒の関係という単純な評価項目を提案し,既存の評価項目の多くが駒の関係により表現可能であることを示す.さらに,棋譜の指手に注目した判別分析を行なうことで,好形を評価するように重みを自動的に調整したところ,実験で良い結果を得られた.

### Abstract

New evaluation functions for Shogi-game, based on pairwise relations among pieces, are proposed. Existing evaluation functions treat a lot of specific aspects, as well as material balance, of the game. Most of such aspects can, we show, be handled by the proposed method. We also proposed the automatic adjustments of the weights of pairwise relations, by means of discriminant analysis of many records of the game. Our experiments showed that reasonably accurate evaluation functions for preferable configuration of pieces can successfully be constructed.

<sup>\*</sup>kaneko@graco.c.u-tokyo.ac.jp

### 1 はじめに

強いゲームプログラムを作るためには良い評価関数が必要であり,そのためには評価項目の設計と各項目の重みづけが必要である.評価項目とは局面の中で何に注目するかであり,将棋の場合,駒の損得や玉の危険度などが知られている [8].

評価関数は複数の評価項目を考慮して総合的に有利・不利を判定する.一般的に用いられる線形結合の場合は,各評価項目の重みの総和が評価値となる.各評価項目の重みは事前に決めておく必要がある.

複雑な評価関数において,プログラマが最適な値を探したり各評価項目に矛盾なく重みを設定することは難しい.そのため,できるだけ評価項目を単純にし,重みの自動調整を可能とすることが望ましい.しかし,将棋の様々な評価項目について,調整を自動的に行なって成功した例は今までなかった.

本稿では,駒の関係という単純な評価項目を提案し,既存の評価項目の多くが駒の関係により表現可能であることを示す.また駒の関係は機械的に扱いやすいため,適切な教師例を用意すれば自動的な重みの調整が可能である.そこで,実際に好形悪形を学習させたところ十分な結果を得ることができた.

### 2 関連研究

現在までの将棋プログラムの評価関数として最も基本的な評価項目は駒の損得である.駒の重みは,“激指”やYSSのものは公開されている [8, 11].TD法 [4] を用いて自動的な調整を試みた研究もある [2].

一方、将棋では駒の損得以外の評価項目が重要である。チェスと比べて駒の損得の重要性は低く [15]、玉の危険度の評価をしないとレーティング換算で 200 点以上弱くなると報告されている [9]。駒の動きとして、相対位置を計算し相手玉に近づくほど高く評価する方法は多くのプログラムで採用されている [11, 12, 9]。玉の危険度については、玉の周辺の利きや玉の自由度を測る方法が用いられている [12, 7]。囲いについては、落とし穴方式と呼ばれる表に基づく評価方法 [11] や一般的な規則による評価 [9] などがある。これらの点数の決め方は将棋プログラマの経験に基づいて決められていて、自動的に値を調整した例は著者の知る限りはない。また、本稿で取り上げる好形悪形の評価は、評価関数に認識させることは困難であると言われており、KFFend では悪形チェックと呼ばれる特別なアルゴリズムを導入している。しかし、それには副作用もあるため、可能であれば評価関数に認識させる方が望ましいとされている [6]。

なお、探索量が増えるほど評価関数は単純ですむと言われており [9]、実際に大駒を取られる可能性は以前より重視されていない [12]。ハードウェアの進歩や探索の深さの制御の進歩 [5] によりこの傾向はある程度続くと思われる。しかし、駒の損得のみで強いプログラムを作れるとは考えにくいいため、評価項目の設計は将来に渡って重要な課題である。

将棋以外のゲームではオセロ [3] やバックギャモン [4] などで、重みを自動的に調整した関数が手で設計した評価関数の性能を上回っている。特にオセロ [3] では約 150 万という多数の評価項目を用いて正確な評価関数を実現した。本研究で提案する手法では約 500 万で、今までの将棋で使われている手法よりも多く、正確な評価関数となることが期待できる。

一般に、重みを自動的に調整するには、訓練例を用意する必要がある。オセロでは終局時の石の数の差を勝敗の差の度合と考えると、終盤の評価関数を訓練した後に中盤/序盤の評価関数を探索結果をもとに順に訓練する方法が取られた [3]。一方、将棋では接戦かどうかの数值化は容易ではない。本稿の実験では訓練例を用意する問題を避けるため、目標とする概念を絞り好形の判定のみ学習させた。

終盤では駒得より寄せを重視するなど、ゲームの進行に伴って各評価項目の重要性は変化するため、評価関数を使い分けることは多くのプログラムで行われてきた [7, 11, 12]。進行度を、敵陣へ近づいた駒を

もとに数值化し、終盤に重視する項目では進行度を重みにかけることでなめらかな変化を実現する手法もある [9]。オセロなど手数がほぼ固定のゲームは、手数ごとに評価関数を用意し、あらかじめ、前後の進行度の評価関数と平均しておくことで重みをならすことが行なわれている [3]。本稿の実験の範囲では特に進行度は考慮しなかったが、実際に強いプログラムを作る上では重要な概念である。

## 3 駒の関係に基づく評価関数

### 3.1 評価項目

まず本研究で提案する評価関数の評価項目を説明する。提案する評価関数では、駒の関係に重みを割り当てる。駒の関係とは、盤上の二駒の  $p, q$  に対する両者の関係  $R_{p,q}$  で、各駒について位置、種類、所有者を考慮して、 $R_{p,q} = ((\text{位置}_p, \text{種類}_p, \text{所有者}_p), (\text{位置}_q, \text{種類}_q, \text{所有者}_q))$  とする (同じ駒の場合 ( $p = q$ ) も含む)。例えば、先手 73 銀、後手 82 飛の二駒の関係  $R$  は、 $R = ((73, \text{銀}, \text{先手}), (82, \text{飛}, \text{後手}))$  と表す。局面  $s$  の評価値は、局面上に成り立っている関係に対する重み  $value(R_{p,q})$  の総和とする：

$$\text{評価値}(s) = \sum_{R_{p,q} \in s} value(R_{p,q}). \quad (1)$$

このモデルは現在よく使われている基本的な手法を包含している。駒の価値のみの評価関数は提案する方式の特殊な場合で、 $p = q$  の時に  $value(R_{p,q})$  を駒の価値とし他は 0 とした場合に相当する。敵玉への迫り方を考慮した評価関数も同様に、 $value(R_{p,q})$  が  $p$  も  $q$  も玉でない場合ならば 0 である場合に相当する。飛車は敵陣にいる方が価値が高い [7] といった一つの駒と位置のみの評価も、同じ駒の関係 ( $R_{p,p}$ ) を調整することで組み込むことが可能である。

一方、大駒などの遠くまで届く利きは、さえぎる駒が関係するため直接は表現できない。<sup>1</sup> 従って、飛車の利きが玉の側に届いているかなどは、別に評価項目を作った方が良い可能性がある。また、囲いについても直接認識するわけではなく、2 つの駒に分解して重みをつける。しかし本稿の実験の範囲では個別に取り扱う必要は認められなかった。

<sup>1</sup>短い利きについては位置だけで決まるため表現されている。

表 1: 1 局面あたりの評価にかかる時間 (clocks)

評価なし	駒の損得	駒の関係
98.2	106.3	1078.4

### 3.2 実行コスト

評価に要する時間は、現在の局面で成り立つ関係を調べ、その重みを表から求める時間である。平手戦の場合 40 枚の駒があり、1 局面では  $(\sum_{i=1}^{40} i) = 840$  の関係が存在する。新規に計算する場合は全ての駒の関係を調べるため 840 回の表引きが必要となる。探索においては差分計算が可能であり、その場合は指手で変化した関係のみを調べるため、表をひく回数は 80 回 (単純な移動の場合)、約 160 回 (駒を取る場合) となる。単純な駒の損得の計算よりはコストが大きい、条件判断がないため、プログラマがリストアップした形を調べるような複雑な評価関数よりは速いと期待できる。

実際に、著者らが開発に参加している将棋ライブラリ [14] で速度を測定して結果を表 1 にまとめた。所要時間は終盤の局面から 3 手の全探索で約 300 万局面を評価した時の平均である。「評価なし」は、指手生成と局面変更のみを行なった場合で、利きやハッシュコードの管理などは行っていない。「駒の損得」と「駒の関係」はそれぞれの評価値を計算した場合である。測定には AthlonMP 1.7GHz の PC を用いた。駒の関係の評価を行なうと駒の損得の場合と比べて約 10 倍の時間がかかるが、それでもまだ 1 秒間に約 160 万局面を評価できる速度であり、代償に見合う価値があると考えている。

駒の種類は所有者と成不成を併せて 28 種類で、駒の位置は盤上と駒台で 82 種類であるため、重みを 1 バイトで表現して単純な表を作ると大きさは  $(28 \cdot 82)^2 =$  約 5 メガバイト強となり現実的な範囲である。なお、 $R_{p,q} = R_{q,p}$  であり、また同じ場所には一つの駒しかないという制約があるため、可能な関係の種類は 2,578,852 である。先手と後手を対称と考えればさらに半分になるが、今回の実験では区別した。

## 4 好形の学習

続いて、前節で提案した評価関数について、各関係の重みを自動的に調整する方法を提案する。ここ

では好形悪形の評価を題材とする。

好形を評価する評価関数は、駒の損得の無い指手が多くある局面で形の良さを細かく判断するものである。定跡から外れてもそれなりに指すなどの用途を想定している。

重みを自動調整するためには、目標とする概念と学習アルゴリズムにあわせた訓練例を用意する必要がある。本稿では、判別分析 [13] を採用し、「局面 a は局面 b より良い (もしくは悪い)」という優越関係を棋譜から作って訓練例として使用した。以下それぞれ詳しく説明する。

### 4.1 好形の定義と訓練例の作成

訓練例を準備するためには、好形とは何かを決める必要がある。今回は、人間が多く指した手是好形であるとして棋譜を用いた。但し、駒の損得の無い指手のみを対象とした。これは、駒得をする指手は形が悪くても指す可能性があり、また駒損の手を指す際は詰みや将来の駒得が絡むなど読みの入った手であり形が良いから指しているとは考えにくいのである。実験を単純にするために、勝敗やプレイヤーの強さなどは考慮していない。

続いて、棋譜の指手から判別分析に必要な訓練例を作成する。以下に述べる 2 つが自然なモデルとして考えられるため、本稿では 2 種類の実験を行なった。

#### 4.1.1 親子モデル

親子モデルでは、棋譜の指手の前後を比較した時に「指した後は指す前よりも手番のプレイヤーにとって良くなる」という仮定から優越関係を作る。有効な手を指している限りは妥当な仮定と思われるが、指さなかった手はモデルに入らないため悪い形を認識できない可能性もある。

具体的には、手を指す前の局面  $t$  と指した後の局面  $s$  を比較して、 $t$  が先手番の場合、

$$\text{評価値}(s) - \text{評価値}(t) > 0$$

という訓練例とし、後手番の場合は

$$\text{評価値}(s) - \text{評価値}(t) < 0$$

という訓練例とする。(評価値は先手が有利の時に正、後手が有利の時に負とする。)

#### 4.1.2 兄弟モデル

兄弟モデルでは、「棋譜で指された手はその局面の他の候補手よりも良い」という仮定から優越関係を作る。指さなかった手を明示的にモデルに入れることで悪い形も認識できると期待できるが、指さなかった手は全て悪手という仮定は強すぎると考えられることから、誤った訓練例を作ってしまう可能性もある。具体的には、ある局面 ( $t$ ) で、棋譜の指手の後の局面 ( $s_0$ ) と他の候補手の後の局面 ( $s_1, \dots, s_n$ ) があったとする。  $t$  が先手番の場合、

$$\text{評価値}(s_0) - \text{評価値}(s_i) > 0$$

という訓練例を各  $i(0 < i \leq n)$  について作る。後手の場合は、親子モデルと同様に不等号を逆にする。

#### 4.2 判別分析

最後に判別分析 [16] を用いて評価関数の重みを決める。判別分析とは統計の手法で 2 群の判別に用いられるが、ここでは訓練例として与えられた不等式をなるべく満たすような重みを求める手法として使用する。実際の計算では線形回帰に変形して重みを求めた。すなわち、 $X$  を訓練例を表す行列、 $w$  を重みのベクトル、 $y$  を教師値のベクトルとして、

$$X^t X w = X^t y \quad (2)$$

を解いて  $w$  を求める。この時、教師値  $y$  は、正例 (左辺が正の訓練例) の場合 1、負例 (左辺が負の訓練例) の場合  $-1$  とした。行列  $X$  は、各訓練例の左辺と対応する行ベクトルを並べたもので、 $k$  番目の行ベクトル  $x[k]$  は、重み  $w$  との内積を取ると訓練例  $k$  の左辺となるものである。提案する評価関数の場合は、 $x_i[k] = r_i[k_0] - r_i[k_1]$  となる。但し、各関係  $R_{p,q}$  に番号  $i$  を割りあて、その重み  $value(R_{p,q})$  を  $w_i$  と表す。変数  $r_i[k]$  は、各関係  $i$  が局面  $k$  で成立するかどうかを 1 と 0 で表す。評価値は式 (1) の形なので、

$$\begin{aligned} w \cdot x[k] &= \sum_i w_i x_i[k] = \sum_i w_i (r_i[k_0] - r_i[k_1]) \\ &= \text{評価値}(k_0) - \text{評価値}(k_1) \end{aligned}$$

となる。なお、式 (2) を解く際には、直接解法は使えず、反復解法を用いる必要がある。これは  $X^t X$  が、次数が重みの数 ( $> 200$  万) の正方行列となり現在の PC のメモリに載せられないためである。本稿の実験

表 2: 駒の価値

歩	香	桂	銀	角	飛
100	400	400	550	800	950
馬	龍	金・他の成駒			
1150	1300	600			

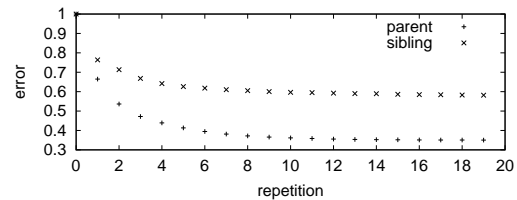


図 1: 収束の状況

では共役勾配法 [1] を用いた。一般に  $Aw = b$  を解いて  $w$  を求める際に、ベクトル  $p$  を与えられて、積  $q = Ap$  を計算できれば反復解法が使える。今回の場合  $A = X^t X$  であり以下のように変形できる。

$$\begin{aligned} q_i &= \sum_j (a_{ij} p_j) = \sum_j (x[i] \cdot x[j] p_j) \\ &= \sum_j \sum_k (x_i[k] x_j[k]) p_j \\ &= \sum_k \sum_j (x_i[k] x_j[k] p_j) \end{aligned}$$

この変形を用いると、行列  $A$  を保持していなくても各訓練例  $k$  を巡回しながら  $q_i$  に  $\sum_j (x_i[k] x_j[k] p_j)$  を足すことで  $q = Ap$  を計算できる。判別分析にかかる時間は、この積の計算がほとんどである。各訓練例  $k$  において、 $x_i[k]$  が 0 の場合はそれが関係する積の計算を省略できるため、計算時間は  $x_i[k]$  の 0 でない個数の二乗に比例する。 $x_i[k]$  は訓練例で比較する局面二つでそれぞれ成り立つ駒の関係の差分であるため、似た局面同士の場合 0 であるものが多い。兄弟モデルでは、親子モデルのほぼ 2 倍の差分があり、計算時間は 4 倍近くなる。

## 5 実験結果

親子モデルと兄弟モデルに基づいて 2 つの評価関数を作り、得られた評価関数の評価を行なった。

訓練例としては、将棋倶楽部 24 [10] の棋譜集から駒の損得のない局面を抽出して用いた。駒の損得の有無は、取り合い探索を行なった時に評価値が変わらないことで判定した。取り合い探索で用いる駒の価値としては“激指”で用いられている値 [8] を参考に表 2 の値 (盤上の駒も持ち駒にも同じ点) を用いた。

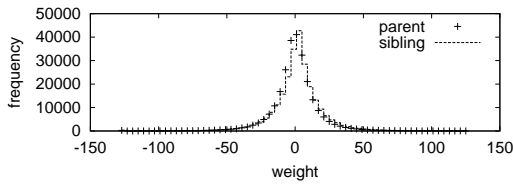


図 2: 重みの分布 (ヒストグラム)

表 3: 重みの上位 3 つ (上:無制限, 下:出現数 200 以上)

種類	駒 1	駒 2	出現数	重み
親子	12 馬	79 龍	23	2.56
	22 歩	25 成銀	56	2.27
	19 玉	78 香	34	2.19
兄弟	79 成銀	98 成香	30	2.66
	62 馬	64 馬	24	2.61
	36 龍	63 成桂	25	2.32
親子 ( $\geq 200$ )	26 飛	27 歩	4011	1.01
	79 銀	88 銀	39650	0.90
	76 飛	77 歩	1533	0.84
兄弟 ( $\geq 200$ )	25 玉	26 歩	419	1.05
	24 香	25 歩	765	1.04
	38 馬	39 金	378	1.03

棋譜の数は親子モデルの場合 20 万棋譜, 兄弟モデルの場合 4 万棋譜を用いた. 訓練例の数はそれぞれ, 約 1500 万と約 9 億となった. 学習の安定性を考慮して, 訓練例の中で 20 局面以上に現われた関係のみを用いた. 親子モデルの場合, 全部で約 258 万種類の関係の中で約 106 万のみが対象となった. 訓練例が異なるため本来は別に測定するべきだが, 実験の都合で兄弟モデルでも同じ関係に重みをつけた.

図 1 に収束の様子を示す. 反復毎に少数のテスト例での回帰の誤差をプロットしている. 親子モデル (parent) の方が誤差が小さく, 兄弟モデル (sibling) の方が難しい制約が多いことを示唆している. 親子モデルは 36 回反復させ, 兄弟モデルは実験時間の都合で 20 回で打切った. それぞれのモデルの上位 3 つの重みを調べたところ表 3 の上半分のように不自然な関係が表れた. いずれも出現回数が少ないため出現回数 200 回以上の条件で調べ直したところ, 表の下半分のように利きをつける関係などが表れた. 出現数の閾値には調整の余地が残っていると言える.

最後に重みが  $[-127, 127]$  の範囲でほぼ山型となるようにそれぞれ 128 倍, 160 倍にスケールして離散化した. そのヒストグラムを図 2 に掲載する. 見易さのために 4 つ刻みでプロットした. 離散化後に 0 でない重みがついた関係はそれぞれ約 97 万と約 99

表 4: 指手の前後での評価値の上昇

手数	対象 (%)	上昇率 (%)	
		親子	兄弟
0-20	98.4	99.9	93.2
20-40	90.7	98.3	87.2
40-60	65.8	96.1	82.4
60-80	49.8	96.9	81.1
80-100	46.3	97.2	80.2
100-	46.2	96.4	79.0
全体	67.5	97.8	85.6

表 5: 一手読みによる指手の順位づけ

手数	候補手の数		順位	
	合法手	候補手	親子	兄弟
0-20	35.0	30.5	12.2	1.7
20-40	44.6	27.2	8.3	2.3
40-60	68.9	17.9	5.7	2.0
60-80	109.3	12.9	4.1	1.8
80-100	133.2	10.3	3.3	1.7
100-	161.3	9.1	2.9	1.9
全体	75.7	21.1	7.3	1.9

万, 範囲を越えた重みはどちらも約 500 であった.

## 5.1 指手による評価値の上昇の確認

テスト用のデータとして, 訓練例とは別の 1 万試合について, 棋譜の指手の中で指手の前後で駒の損得が変わらない局面を訓練例と同様に選び, 約 76 万局面を用意した. これらに対して指手の前後で評価値が指した手番からみて上昇したかどうかを調べたところ, 表 4 の結果となった. 表で「対象」とは対象となった指手の割合を示し, その率の低下は中盤から終盤へと進むにつれて駒の損得がある指手が増えるという将棋の性質を示している.

親子モデルでは, 全体で 97.8% とほぼモデル通りの学習できたと言える. 一方, 兄弟モデルでもかなりの確率で指した後の評価値が上昇しているが, 親子間の比較はモデルに入っていないため, 親子モデルには及ばない. さらに手数毎に細かく見ると, 成績は手数が進む毎に両モデルとも成績が悪くなり, 兄弟モデルの方がより低下が大きい.

## 5.2 一手読みの評価

同じテストデータに対して, 棋譜の指手が他の候補手よりも高く評価されるかどうかを調べた. その

表 6: 矢倉定跡の評価

定跡	親子		兄弟	
	順位	最善手	順位	最善手
76 歩	24	48 王	0	
84 歩	8	62 王	1	34 歩
68 銀	5	48 王	2	26 歩
34 歩	8	62 王	0	
77 銀	2	77 角	1	66 歩
62 銀	9	62 王	0	
26 歩	11	38 飛	1	66 歩
42 銀	4	52 王	5	85 歩
48 銀	7	48 王	1	25 歩
32 金	12	52 王	1	54 歩
78 金	15	39 金	1	56 歩
54 歩	16	71 金	0	
56 歩	12	39 金	3	25 歩
41 王	6	71 金	0	
69 王	6	39 金	0	
52 金	16	71 金	0	
58 金	13	39 金	0	
74 歩	3	31 王	0	
36 歩	3	98 香	1	25 歩
33 銀	11	12 香	2	64 歩
79 角	15	98 香	0	
31 角	9	31 王	1	64 歩
66 歩	8	46 角	7	35 歩
44 歩	12	73 銀	2	64 角

結果を表 5 に掲載する。まず合法手を全て作成し、その中で駒の損得がない手を候補手とした。指手が進むにつれ合法手の中の候補手の割合は減ってゆくが、全体としては 4 分の 1 程度 (21.1/75.7) であった。それらの候補手を、指した後の局面の評価値順に並べ、棋譜の指手の順位を調べた。兄弟モデルでは平均約 2 番目でほぼモデル通り学習できたと言える。もし駒の損得のみの評価関数を使うと候補手 21 手は同じ値で、その中からランダムに選ぶとすると平均順位は 10 番目程度になるため、大幅な向上である。親子モデルでも棋譜の指手は平均約 7 番目とそれなりに高く評価されているが、親子モデルでは兄弟の比較はモデルに入っていないため兄弟モデルには及ばない。親子モデルでは手数にほぼ関係なく候補手の数の約 3 分の 1 程度の順位であるが、兄弟モデルによる順位は候補手の数の影響が小さい。

### 5.3 定跡の評価

別の実験として、序盤の定跡の手をどの程度高く評価できるかどうかを矢倉を題材に調べた。局面で



図 3: 悪形の例題 (1)

定跡の指手が何番目に高く評価されたかと、定跡と一致しなかった場合に最善と評価された手を表 6 にまとめた。兄弟モデルでは比較的良く定跡に一致し、また一致しなかった場合も無難な手を選択しているので実戦で効果があると期待できる。一方、親子モデルの方は不安な指手が多く序盤には向かないと思われる。

### 5.4 悪形の回避

続いて、学習した評価関数が、一般的に悪形と言われている避けるべき形を回避できるかどうかを実験した。悪形の例題として、Web で公開されている「将棋プログラム KFEnd」の一部の「悪形チェック」[6] の中で取り上げられているものを用いた。文献 [6] では 30 種類の項目が挙げられているが、評価関数で評価するために、一般的な規則ではなく、具体的な局面があるものを取り上げた。

表 7 に学習した評価関数による評価結果を載せる。数値は指手をさす前後の評価値の増減であり、大きい方が先手有利という評価である。

例題 1(図 3)で、36 飛あるいは 46 飛は「自歩先の飛」であり攻める態勢を作り難しくしてしまう悪形、また 25 歩は「自らの飛筋を止める歩を打つ」悪形と解説されている。16 歩、46 歩は比較のために著者が加えた候補手である。親子モデルでは、46 飛をもっとも高く評価しているが、普通の手にも同程度の点がついている。一方、兄弟モデルのほうは、最高点は 16 歩であり妥当な評価と思われる。

例題 2(図 4)は、77 角成と角を交換しに来られた図で、77 同金や同桂は壁形の悪形であり、77 同銀が普通と解説されている。この問題については、

表 7: 各モデルの評価関数による悪形の例題の評価

親子モデル						
	例題 1 指手 評価	例題 2 指手 評価	例題 3 指手 評価	例題 3' 指手 評価	例題 4 指手 評価	
悪手	36 飛 9 46 飛 61 27 飛 21 25 歩 30	77 桂 5 77 金 -8	18 飛 99 ( 28 銀) 52	18 飛 88 ( 28 銀) 115	39 王 -41 26 歩 -3	
普通の手	16 歩 55 46 歩 41	77 銀 -22	26 歩 7	26 歩 121	56 歩 19 46 歩 26	
兄弟モデル						
	例題 1 指手 評価	例題 2 指手 評価	例題 3 指手 評価	例題 3' 指手 評価	例題 4 指手 評価	
悪手	36 飛 -15 46 飛 -13 27 飛 -17 25 歩 -41	77 桂 -25 77 金 -42	18 飛 3 ( 28 銀) -21	18 飛 -38 ( 28 銀) -38	39 王 -25 26 歩 -40	
普通の手	16 歩 38 46 歩 -5	77 銀 9	26 歩 -9	26 歩 59	56 歩 2 46 歩 9	

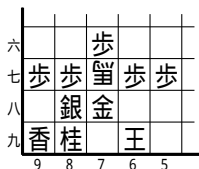


図 4: 悪形の例題 (2)

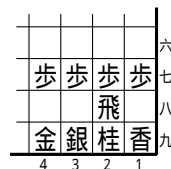


図 5: 悪形の例題 (3)

表 8: 例題 2 の指手に関する重みの抜粋

駒 1	駒 2	親子	兄弟
88 銀	78 金	8	7
77 銀	78 金	3	7
77 金	88 銀	9	-7
77 金	89 桂	23	2
78 金	88 銀	8	7

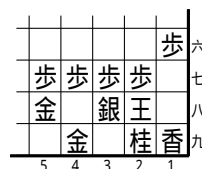


図 6: 悪形の例題 (4)

兄弟モデルは 77 同銀に良い評価をし、また悪形になる手をはっきり悪く評価できたが、親子モデルは逆の結果となった。原因を調べるために、評価値に影響する関係の重みを抜き出して表 8 に掲載した。親子モデルと兄弟モデルを比較するとほぼ似たような重みがついているが、77 金に対する重みが親子モデルでは高すぎるのが問題のようだ。

例題 3(図 5) では、18 飛は自陣の端に移動する悪形で、さらに 28 銀とするとさらに悪形である。

28 銀は 18 飛の後で指す手のため、表では括弧をつけて区別した。この例題では悪形を作る指手が、著者が考えた普通の手である 26 歩と比べて良い点になってしまっている。そこで、初期局面の盤面全

体を使って同じ候補手を試したところ (例題 3')、26 歩がもっとも良い手と評価された。従って例題 3 は局面が狭すぎたと解釈できる。

例題 4(図 6) で、39 玉は囲いを崩す悪形と解説されている。また 26 歩は、原文にはこの図との対応はないが、「自玉の頭の歩突き」に相当する悪形と判断し、著者が加えた。46 歩 56 歩は普通の手例として比較のために著者が加えた。どちらのモデルもまともな判断をしている。

総合して、どちらのモデルも悪手を避けるためには有効であり、特に兄弟モデルでは全ての例題で悪形を避けることができた。兄弟モデルの評価関数は、実験全体を通して良い成績であったため、兄弟モデルに基づいた学習は効果的であったと結論できる。

## 6 おわりに

本稿では駒の関係を利用した局面の評価方法を提案し、既存の評価項目の多くが駒の関係により表現可能であることを示した。また、好形の判定を題材に、そのような評価関数の重みを自動的に調整する方法として、棋譜の判別分析を提案した。実験の結果、提案手法を用いて重みを自動調整した評価関数が、定跡や人間の指手をかなりの程度模倣し、また悪形を避けることに有効であることが確認できた。棋譜から訓練例を作る手法として、指手の前後の局面の比較をもとに学習する親子モデルと指手と他の候補手を比較して学習する兄弟モデルの二つの手法を比較したところ、後者の方が良い成績であった。

今後、強いプログラムを作るには、詰む確率や玉の危険度を評価し、さらに、それらがどの程度の駒損と釣り合うかを調整することが重要である。適切な教師例を用意できれば、駒の関係でそれらを表せると考えている。また評価関数以外にも、駒の関係という着眼点は探索の深さの制御(実現確率 [5]) などでも有効と思われる。将来、精度を上げるためには、駒のペアの代わりにトリプレットに基づく評価を行い、「龍の利きを遮る金底の歩」などの概念が表すことも考えられる。しかし表の要素数は  $(28 \cdot 82)^3$  となり 12 ギガバイトを越え、また訓練に必要なデータや時間が大幅に増えるため現時点では難しい。

## 参考文献

- [1] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994. (<http://netlib2.cs.utk.edu/linalg/html.templates/Templates.html>).
- [2] D. F. Beal and M. C. Smith. First results from using temporal difference learning in shogi. In *Proceedings of the First International Conference on Computers and Games*, pp. 113–125, Tsukuba, Japan, Nov. 1998. Springer-Verlag.
- [3] M. Buro. Improving heuristic mini-max search by supervised learning. *Artificial Intelligence*, 134(1–2):85–99, Jan. 2002.
- [4] G. Tesauro. Programming backgammon using self-teaching neural nets. *Artificial Intelligence*, 134(1–2):181–199, Jan. 2002.
- [5] Y. Tsuruoka, D. Yokoyama, and T. Chikayama. Game-tree search algorithm based on realization probability. *ICGA Journal*, 25(3):145–153, 2002.
- [6] 有岡. 悪形チェック (Inside KFEnd). Web, 2000. ([http://plaza9.mbn.or.jp/%7ekfend/inside\\_kfend/bad\\_shape.html](http://plaza9.mbn.or.jp/%7ekfend/inside_kfend/bad_shape.html)).
- [7] 柿木. 将棋プログラム K3.0 の思考アルゴリズム. 松原 (編), コンピュータ将棋の進歩, 第 1 章, pp. 1–22. 共立出版, 1996.
- [8] 鶴岡. 将棋. 情報処理, 44(9):900–904, 2003. 特集 ゲーム情報学.
- [9] 鶴岡. 将棋プログラム「激指」. 松原 (編), アマ 4 段を超えるコンピュータ将棋の進歩 4, 第 1 章, pp. 1–17. 共立出版, 2003.
- [10] 久米. 将棋倶楽部 24 万局集. ナイタイ出版, 2002.
- [11] 山下. YSS—そのデータ構造, およびアルゴリズムについて. 松原 (編), コンピュータ将棋の進歩 2, 第 6 章, pp. 112–142. 共立出版, 1998.
- [12] 金沢. 金沢将棋のアルゴリズム. 松原 (編), コンピュータ将棋の進歩 3, 第 2 章, pp. 15–26. 共立出版, 2000.
- [13] 奥野, 久米, 芳賀, 吉沢. 改訂版 多変量解析法. 日科技連, 1981.
- [14] 田中, 副田, 金子. 組合わせて利用可能な高速将棋ライブラリの作成. 第 8 回ゲームプログラミングワークショップ, Nov. 2003.
- [15] 棚瀬. IS 将棋のアルゴリズム. 松原 (編), コンピュータ将棋の進歩 3, 第 1 章, pp. 1–14. 共立出版, 2000.
- [16] 田中, 垂水, 脇本. パソコン統計解析ハンドブック. 共立出版, 1984.