情報処理学会論文誌 Vol.53 No.11 2544–2551 (Nov. 2012)

入玉指向の将棋プログラムの作成

滝瀬 竜司[1,a)] 田中 哲朗[2,b)]

受付日 2012年2月20日, 採録日 2012年9月10日

概要：現在の将棋プログラムの多くは入玉が絡む局面の扱いを不得意としているが，トッププロに勝つためには，入玉が絡む局面を正しく扱い「自玉が入玉できそうなときは入玉を目指す」,「敵玉の入玉を正しく阻止する」将棋プログラムの作成が不可欠だと考えられる．本稿ではオープンソースの将棋プログラム Bonanza の評価関数を変更することにより入玉指向の将棋プログラムを作成する試みを行った．その結果，元の Bonanza の評価関数に「入玉ステップ数」という特徴を加えて学習した評価関数を用いることにより，勝率を落とさずに入玉率を大幅に上げることができた．

キーワード：将棋，入玉，評価関数，機械学習，Bonanza

Development of Entering-king Oriented Shogi Programs

Ryuji Takise[1,a)] Tetsuro Tanaka[2,b)]

Received: February 20, 2012, Accepted: September 10, 2012

Abstract: Most of recent shogi programs are thought not to treat entering-king positions correctly. The goal of this paper is to make a shogi program that treats entering-king positions correctly. We have conducted a couple of experiments to create an entering-king oriented shogi program by modifying the evaluation function of an open source shogi program "Bonanza". We made an evaluation function which has a new feature "entering-king step" besides original features, and adjusted the weight parameter of this feature based on comparison of moves that appeared in a database of game records. As a result, we succeeded to raise the entering-king rate without decreasing the original winning rate.

Keywords: Shogi, entering-king, evaluation function, machine learning, Bonanza

1. はじめに

近年のコンピュータ将棋の進歩は目覚ましく，この数年の間に行われた，公開の場でのトップアマや女流棋士 [1]，引退棋士相手の対局では，大幅に勝ち越している．そのため，公式の場でトッププロを破る日も遠くないといわれている．その一方で，対コンピュータ将棋戦略を研究しているアマチュア強豪の多くが，コンピュータ将棋の苦手な展開がいくつかあることを指摘している [2]．そこで指摘されている展開の 1 つが入玉絡みの展開である．

自分の玉を敵陣（相手側の 3 段以内）に移動することを入玉という．入玉して成駒で周りを固めると，敵から詰まされにくくなるので有利になる．両方の玉が入玉すると，両者ともに相手玉を詰ますことができなくなり，勝負が終わらなくなるので，駒の点数を計算して，それによって勝敗および持将棋（引き分け）を判定する．

本稿では，「入玉勝ち」を「勝ちが決まった時点（相手の投了，勝ち宣言）で勝ったプレイヤの玉の位置が敵陣にある」ことと定義する．この定義は，機械的に判定が可能であり，人間の感覚と異なるケースはそれほど多くないと考えられる．

人間の将棋でも「入玉勝ち」の頻度はそれほど多くない．本稿の実験で使った，プロの棋譜と将棋倶楽部 24 の棋譜集 [3], [4] 合わせて約 52 万局中に入玉勝ちの棋譜は約

[1] 東京大学総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo, Meguro, Tokyo 153–8902, Japan
[2] 東京大学情報基盤センター
Information Technology Center, The University of Tokyo, Bunkyo, Tokyo 113–0032, Japan
a) takise@tanaka.ecc.u-tokyo.ac.jp
b) ktanaka@ecc.u-tokyo.ac.jp

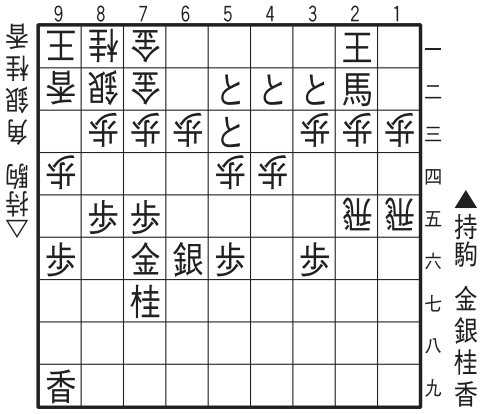

図 1 先手勝勢だが Bonanza が後手良しと判断する局面（文献 [5] より）

Fig. 1 Black has more chance to win, but Bonanza thinks that white is better.

1.5%含まれていた．一方，コンピュータ将棋どうしの自動対局が行われている floodgate[*1]の 2010 年度の対局約 10 万局中では，入玉勝ちは約 2.6%なので，コンピュータ将棋の方が頻度が大きいといえる．

将棋は相手玉を詰ますことが目的のゲームであり，通常は自玉の周りに駒を集めて囲いを作り，また敵玉を攻撃するために駒得を目指す．しかし，入玉絡みの局面では，自玉の周りの駒が邪魔になることもあるし，片方が入玉し，他方が入玉できない状況では，入玉していない側の駒得がどれだけ大きくても，不利になっていることが多い．両方が入玉して点数勝負になると，小駒（金銀桂香歩，および，それらの成駒）の価値が同じになってしまい，通常の将棋とは違うゲームのように見えることもある．

コンピュータ将棋の特徴を決めるのは評価関数と探索だが，コンピュータ将棋が入玉絡みの局面を苦手としている原因としては，入玉絡みの局面の評価関数が適切でないことがあると指摘されている [5]．**図 1** は，文献 [5] で紹介されている先手勝勢の局面だが Bonanza [6] は後手良しと判断する．

以前のコンピュータ将棋では，人間の経験則によって決定した駒の評価値をベースに，玉の安定度や大駒の mobility などに関連する特徴を加え，人手で重みをつけた評価関数を用いるのが一般的だったが，2006 年に浅い探索を組み合わせた兄弟ノードとの比較により評価関数の重みパラメータを学習させた Bonanza が第 16 回世界コンピュータ将棋選手権で優勝したのをきっかけに，多くの上位プログラムが評価関数の重みパラメータを学習により求める手法を用いるようになった．この方法により，教師とする上質の棋譜を十分入手すれば，多数の重みパラメータを持つ複雑な評価関数も扱えるようになった．

Bonanza の最新版（Bonanza version 6）では，評価関数の特徴として玉を含むすべての 3 駒の関係（駒の種類と座標）を用いているが，これらの特徴に対応する重みパラメータを入れるファイル中の重みパラメータ数は約 9,300 万個になっている[*2]．

これだけの大規模な評価関数を用いながら，Bonanza が入玉を扱うのが苦手な理由としては，以下の理由が考えられる．

(1) 学習する棋譜の中で入玉棋譜が不足している．
(2) 3 駒の関係を特徴とするだけでは，入玉絡みの局面を正しく扱えない．

Bonanza の最新版では，重みパラメータの学習のためのプログラムは公開されているが，学習に用いた棋譜は公開されていないので (1) は直接は確かめることはできない．ただし，Bonanza の最初のバージョンに関する文献 [6] では，学習に用いた棋譜のうち，アマチュアの棋譜に関しては，どちらかの玉が相手陣の手前（先手玉の場合は 4 段目，後手玉の場合は 6 段目）まで到達したものだけを用いているという記述があるので，同じ方法を用いているとしたら，入玉棋譜の大部分が学習に使われたことになり，この仮説は棄却される．ただし，

- 利用可能な棋譜の選択を行わずに，学習に用いる棋譜数を増やす必要があった，
- 利用可能な棋譜の選択を行うことにより棋力が低下したので，棋譜の選択をやめた，

という可能性は存在する．

(2) に関しては，

- 入玉できるかできないかにより，盤面の評価値を大きく変える必要がある，
- 盤面の配置の微妙な違い（たとえば，盤上の駒 1 つと持駒との交換）により，入玉できるかできないかが決まることがある，
- この違いを 3 駒の関係の特徴に関する重みパラメータだけで表現しようとすると，入玉に関係しない局面の評価が適切に行われない可能性がある，

などを考えると関係している可能性が高い．ただし，どのような特徴を加えるのが良いのかは容易には決定できない．

本稿では，まず (1) の可能性を検証するために，Bonanza version 6 の特徴をそのまま用いて，教師を入玉勝ちの棋譜の勝った側の着手だけに限って重みパラメータを学習させる試みを行った．

また，入玉絡みの局面の評価に有効であると考えられる特徴として，「入玉ステップ数」という特徴を提案し，入玉勝ちの棋譜を用いてその特徴の重みパラメータだけを学習する試みを行った．

「入玉ステップ数」は局面からただちに得られる特徴ではなく，後述するように，局面に対してドメインを限った軽い探索を行った結果得られる特徴である．2 人ゲームに

[*1] http://wdoor.c.u-tokyo.ac.jp/shogi/floodgate.html

[*2] 独立な重みパラメータ数は対称性の関係でこれより小さい．

おいて，通常探索とドメインを限った軽い探索を組み合わせることは，チェスなどにおける静止探索，将棋における詰み探索など，多く用いられているが，評価関数に軽い探索の結果得られる特徴を用いて，機械学習により重みを調整するという手法は本研究以前には一般的には用いられていなかった．

以下，2 章では関連研究に関して述べ，3 章では提案する「入玉ステップ数」という特徴の詳細を述べる．4 章では，重みパラメータの学習および，得られた重みパラメータを用いた評価関数を用いた将棋プログラムを用いた対戦実験を行い，5 章では結論と今後の課題を提示する．

2. 関連研究

本研究ではオープンソースの将棋プログラム Bonanza [6] をベースに実験を行った．以下の理由から Bonanza を用いた．

- オープンソースの将棋プログラムの中でもトップクラスの棋力がある．
- ソースが小さく手を入れやすい[*3]．
- 評価関数に用いられている特徴が単純．

Bonanza が評価関数に用いる駒割以外の特徴は，基本的には玉を含む 3 駒の関係（駒の種類，位置）のみである[*4]．テーブルとしては，以下の 2 つに分かれている．

kkp 両者の玉と，それ以外の駒 1 つの関係（駒の種類，位置）

kpp 片側の玉と，相手玉以外の駒 2 つの関係（駒の種類，位置）

これらの特徴と重みパラメータとの線形和を評価関数として用いている．左右を入れ替えた関係，手番を入れ替え盤面を上下に入れ替えた関係は同じものと見なし，持駒は個数に応じて別の位置にあると見なしている．

Bonanza では浅い探索を組み合わせた兄弟ノードとの比較により評価関数の重みパラメータを調整する手法を用いていて [6]，その部分のソースも公開されている．今回の実験でも，学習ルーチンに手を入れて学習対象の局面，調整する特徴は変更したが，重みパラメータの調整手法はそのまま用いた．Bonanza の思考アルゴリズムはチェスで広く用いられている全幅探索に基づく．全幅探索において，枝刈りの手法を用いることで探索範囲を削減している．

Bonanza では，残り深さの少ないとき評価値の改善可能性が低い手を枝刈りする futility pruning，パスをすると形勢が悪化する性質を利用する null move pruning などを用いている [7]．これらの枝刈り手法に関してもそのまま用いた．

軽い探索の結果を特徴に用いて機械学習と組み合わせる方法は，本研究以前に一般的に用いられていないが，簡単な探索の結果を特徴に用いる方法としては，Amazons というゲームにおいて，軽い探索をして特定のマス目への駒の歩数を得て評価関数の特徴に使用するという先行例がある [8]．

入玉指向の将棋プログラムを作るという試みは，単に強いだけでなく個性を持ったゲームプレイヤを作る試みであると見なすこともできる．この視点から見た先行研究はいくつかあり，なかでも文献 [9] は Bonanza を用いて，プロ棋士の棋譜を用いて重みパラメータを調整し，棋風を模倣させようというもので，本稿と関連が深い．

3. 提案する特徴

入玉指向のプログラムを作成するために，最初に考えられるのは，玉が敵陣にあるときのみボーナスを与える評価関数を使う方法だろう．

しかしこの方法は，探索木の末端で入玉済みの局面が現れるような，入玉直前の局面でしか有効ではない．一般に入玉を指向するか，囲いを強化するかなどの選択は自玉が自陣にいるうちから決定しなくてはならないが，かなりの深さを読まないと探索木内にこのような局面が出てこない．

一方入玉前の状態，すなわち自玉が 4，5，6 段目にあるときも距離に応じたボーナスを与えれば，浅い探索木でも入玉を指向する手を選択する確率が上がることが期待される．しかし，一般に中段（4，5，6 段）の玉は危険であり，入玉できそうにない場合にとどまるのは賢明とはいえない．

同様の議論は詰みに関しても適用できる．敵玉を詰めると勝ちになるが，詰みにいたる手数は実戦でも数十手になることもあるし，途中では大幅な駒損をすることもある．通常の探索では読み切れないし，詰手順の途中の評価関数の値を高くしようとすると，詰まないのに詰ませにいって駒損して負けてしまう危険性が増す．そのため，多くの将棋プログラムは通常探索とは別に詰将棋ルーチンを使って，詰みの有無を探索している．

入玉に関しても同様の専用ルーチンで探索をさせることが考えられるが，以下の点が詰将棋と異なっているため，専用ルーチンを用いても探索空間がそれほど減らない可能性がある．

- 入玉を目指す手として，玉を移動する手，道を空ける手，敵の大駒の利きを止める手など多種類があり，詰将棋における王手と比べて多数の手を探索する必要がある．
- 入玉を阻止する手としても，敵玉を詰ませる手，逃げ道をふさぐ手など多数の手がある．
- 詰みの発見はゲームの勝敗を決定するが，入玉はそこ

[*3] 著者の 1 人はオープンソース将棋プログラムである GPS 将棋の作者の 1 人でもあり，今回も GPS 将棋をベースに実験することも検討したが，教育的効果を考えて大学院生でもソースの全容を把握可能な Bonanza を選択した．

[*4] Bonanza version 6 では，いわゆる丸山スペシャル（第 20 回世界コンピュータ将棋選手権で稲庭将棋が用いて二次予選に進出して注目された）への対策のための評価関数の補正が入っている．

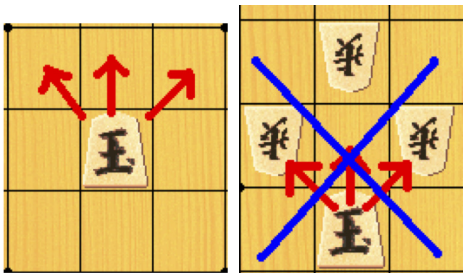

図 2 入玉ステップ数計算のための玉の動き
Fig. 2 King moves for calculating “entering-king step”.

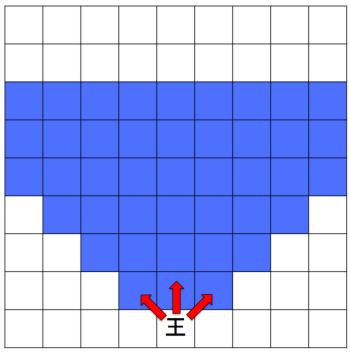

図 3 入玉可能性の探索範囲（最悪の場合）
Fig. 3 Largest search area for judging whether the king can archive an entering-king or not.

で勝負がつくわけではない．

これらの理由により，入玉可能性を探索により完全に求めるのは困難と思われる．そこで，以下のような探索によって入玉可能性の高い局面を検出する特徴を求め，その重みパラメータを学習によって決定した評価関数を用いる方法を提案する．

- 図 2 左のように入玉側の指し手は玉が前，または斜め前に移動する手のみ．
- 図 2 右のように移動の際に相手の駒はとらない（移動できるのは空きマスのみ）．また，敵の利きのある地点には移動しない．
- 敵側は何も指さない．

実際に入玉する場合では，入玉を阻止する側が道をふさぐ手を指し，入玉側の指し手も前進だけでなく，駒をとったり，玉以外の駒を動かしたりして道をあけることもある．そのため，上記のルールでは完全ではないが，入玉可能かを調べるのは最大で 42 マスを調べればよく，効率が良いため，コストの問題を考えて，今回はこのルールを適用した．最大で 42 マスというのは図 3 のように 5 九に王がいる場合である．ルールどおり指し手は王が前，または斜め前に移動する手のみを生成するので，王が前進するごとに調べるマスは多くなり，5 九に王がいる場合は図 3 の青い箇所を調べることで入玉可能性を擬似的に考えることができる．

以下では「入玉ステップ数」としてこのルールで入玉可能なときはこの手数[*5]，このルールでは入玉不可のときは ∞ と定義する．この定義に従って，片側の玉の「入玉ステップ数」を求める関数 king_step の擬似コードを図 4 に示す．visited をすべて false で初期化したうえで，自玉のあるマスを引数として呼び出すことを想定している．擬似コードの「左上」などは玉の手番から見た座標となる．容易に理解できるように，現在の定義では「入玉可能」と判断されたときは，玉が何段目にあるかだけでこの「入玉ステップ数」が決まる．この「入玉ステップ数」は探索中に静的評価関数が呼ばれるたびに，先手後手両方に関して計算する．

```
king_step(square){
  if(square が盤外) return ∞;
  if(visited[square]) return ∞;
  visited[square]=true;
  if(square に自玉以外の駒があるか敵の利きがある) return ∞;
  if(square が敵陣内) return 0;
  return min(king_step(square の左上),
             king_step(square の上),
             king_step(square の右上))+1;
}
```

図 4 入玉ステップ数を求める関数
Fig. 4 Pseudo code for the “entering-king step” function.

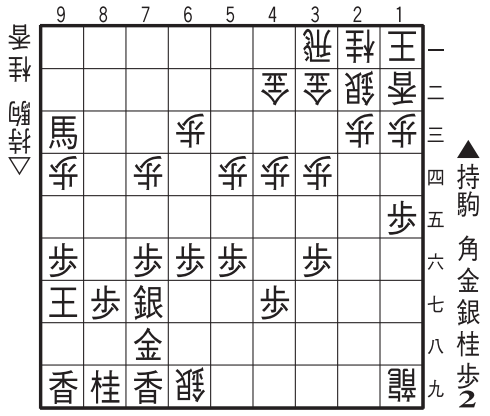

図 5 先手の「入玉ステップ数」が 4 手の局面
Fig. 5 Position in which “entering-king step” of black is four.

図 5 の局面の「入玉ステップ数」はこの定義に従うと，先手が 4 手，後手が ∞ 手となる．実際にはこの局面では後手が飛車を回る，桂馬を打つ，金や香車をとってから打つなどして，入玉を防ぐ手があるので，入玉の可能性は自明ではないが，単に玉が 7 段目にいるという特徴よりは，より入玉しやすさを示す特徴になっている．

4. 実験

4.1 特徴追加なしの入玉勝ち棋譜を用いた学習

まず，Bonanza で使われている特徴の追加をせずに，入玉勝ち棋譜だけを用いて，重みパラメータを学習させる実験を行った．学習にはプロの棋譜と将棋倶楽部 24 の棋譜

[*5] すでに入玉済みのときは 0.

表 1 対戦結果
Table 1 Match results.

プログラム	先手番			後手番			勝率	入玉勝率（%）
	勝ち (入玉)	負け	引き分け	勝ち (入玉)	負け	引き分け		
BonanzaStd	256(13)	235	9	235(7)	256	9	0.5	2
LearnEking-3	117(11)	133	0	87(14)	160	3	0.411	5
LearnEking-5	91(14)	158	1	115(13)	134	1	0.414	5.4
LearnEking-12	87(11)	160	3	79(15)	171	0	0.335	5.2
LearnRandom-12	122(5)	121	7	117(4)	127	6	0.478	2.6

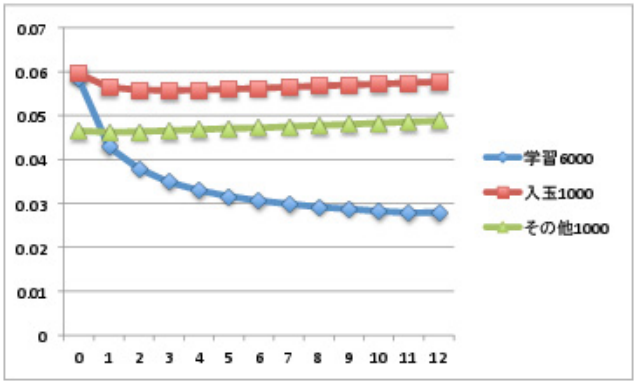

図 6 反復回数ごとの target 値の変化
Fig. 6 Target values during learning.

集の棋譜計 525,119 棋譜の中から入玉勝ち 7,822 棋譜中の 6,000 局を用いた．学習の際には，入玉勝ちした側の指し手のみを学習対象とするように Bonanza の learn コマンドを改造したものを用いた．重みパラメータの初期値は Bonanza 付属の重みパラメータ（fv.bin）を使い，1 回の反復ごとの重みパラメータの更新回数（step）は 32 回とした．学習時に探索する深さは標準値（SEARCH_DEPTH=2）を用いた．

重みパラメータの数に比して，学習に用いた 6,000 局というのはかなり少ない．しかも，入玉勝ちした側の指し手のみを学習対象としたため，通常の学習の 3,000 局分の局面しか学習に用いていないということになる．そのため，過学習の可能性があるので，学習中の重みパラメータに対して，入玉して勝ったうちで学習に用いなかった 1,000 棋譜と入玉していない 1,000 棋譜に関する target 値（文献 [6] の学習の目的関数 J' を指し手の数で正規化したもの）を観察することにした．この値が小さいほど，棋譜の手と，その時点の重みパラメータで求めた評価値に従って浅い探索で求めた最善手との一致度が高いことを示している．反復回数 12 までの結果を**図 6** に示す．

非入玉棋譜の目的関数は反復ごとに上昇しているが，学習に用いなかった入玉棋譜に対しても反復回数 3 回が最小となりそれ以降は上昇している．学習に用いた棋譜に対する目的関数が減少しているので，過学習に陥っている可能性が推測され，反復回数 3 回程度の重みパラメータを用いることが有効であると予想される．

4.2 対戦実験（特徴追加なし）

提案する手法の有効性を確認するために，対戦実験を行う．実験に用いる将棋プログラムは以下のとおりである．

BonaznaStd 標準の Bonanza.

LearnEking-3 特徴は増やさずに入玉勝ち棋譜 6,000 局で勝った側の指し手のみを用いて重みパラメータを学習．反復回数 3 回後の重みパラメータを使用．

LearnEking-5 LearnEking-3 の反復回数を 5 にしたもの．

LearnEking-12 LearnEking-3 の反復回数を 12 にしたもの．

LearnRandom LearnEking-12 の入玉勝ち棋譜 6,000 局の代わりに 525,119 局中から乱数で抽出した 6,000 局で勝った側の指し手のみを学習させたもの．

オリジナルの Bonanza との差が重要なので，対戦相手はすべて標準の Bonanza（BonanzaStd）とする．その他の実験条件は以下のとおりである．

- 相手時間を使用した先読み（ponder）は用いない．
- 思考ノード数を 250,000（用いた計算機では 1 秒程度の思考時間に相当）に制限．
- 評価値が手番のプレイヤにとって，−5,000 を下回ったときに投了．
- 総手数が 2,048 手になったら引き分けとする．
- 先手，後手それぞれ 250 局の計 500 局対戦．

対戦実験の結果を**表 1** に示す．BonanzaStd どうしの対戦も 500 局だが，先手後手ともに同じプログラムなので，勝敗の数は 2 倍多くなっている．定跡手は疑似乱数を用いて選ばれるため，500 局の対戦内で同一の棋譜が複数生成される可能性は低い．実際に同じ棋譜が生成されなかったことは確かめた．

特徴を増やさずに入玉勝ち棋譜で学習させたプログラム LearnEking-{3,5,12} はともに標準を上回る入玉勝率を達成した．フィッシャーの正確確率検定により，LearnEking-{3,5,12} の入玉勝率が BonanzaStd の入玉勝率を有意に上回った．

勝率（千日手および引き分けは 0.5 勝に換算）は，LearnEking-{3,5,12} ともに 5 割を下回った．BonanzaStd より勝率が下回っていることはカイ二乗検定により確認できた．乱数で抽出した 6,000 局で反復回数 12 回学習させ

た LearnRandom-12 の勝率は 5 割を下回るものの，カイ二乗検定では有意性が確かめられない程度の落ち込みしか見せていないことから，Bonanza に特徴を追加せずに，入玉勝ちの棋譜だけを用いた学習を行うと，入玉が絡まない局面の評価が正確に行えない重みパラメータが得られる可能性が高いことが分かった．

4.3 「入玉ステップ数」の特徴を加えた学習

「入玉ステップ数」の特徴を加えた評価関数に関する学習は，以下の 4 通りで試した．

EkingStep-Both 入玉勝ち棋譜で両者の手を学習．

EkingStep-Winner 入玉勝ち棋譜で勝った方の指し手のみを学習．

EkingStep-Random-Both 全棋譜から乱数で抽出した棋譜で両者の手を学習．

EkingStep-Random-Winner 全棋譜から乱数で抽出した棋譜で勝った方の手を学習．

また，今回の実験では，Bonanza 本来の特徴に関する重みパラメータは変更せずに，「入玉ステップ数」の値の 0〜6 に対応する 7 つの重みパラメータだけを変更するように Bonanza の学習プログラムを修正した．

Bonanza version 6.0 では重みパラメータの発散を防ぐために，L1 正則化相当の正則化を行っている．本研究で加えた特徴に関する重みパラメータに関しては，正則化を行わなかったが，重みパラメータは発散しなかった．既存の特徴に関する重みパラメータを固定したことが発散の抑制につながったと考えられる．

なお，「入玉ステップ数」が ∞ のときの重みパラメータは 0 に固定した．また，Bonanza の評価関数では，3 駒間の関係の重みパラメータは，駒割の重みパラメータの 1/32 の値を用いているが，「入玉ステップ数」に対応する重みパラメータは駒割に匹敵するほど大きいことが予想されたため，1/32 の値を用いずに駒割と同様に使うようにした．

収束の様子を調べるために，前の実験では target の値を用いたが，今回は target の値の変化が小さく，学習する重みパラメータが 7 個と少ないため，**図 7** のように，学習の反復回数ごとの 7 つの重みパラメータの変化を直接示すことにする．図のように反復回数 16 回までは単調に増加する傾向にあったが，それ以降は収束していると考えられたため，反復回数 30 回で学習を打ち切った．図 7 は EkingStep-Both の場合の収束の様子を示すが，他の 3 つに関しても同じような傾向が見られた．

得られた重みパラメータを**表 2** に示す．駒割に換算すると，歩 1 枚の重みが 87 となっているので，EkingStep-Both の場合で入玉した状態の価値は約歩 3.5 枚と学習されたことが分かる．入玉勝ち棋譜で勝った方の指し手のみを学習した EkingStep-Winner は，入玉したときの値と入玉ステップ数 6 の値での差が大きい．また，乱数で抽出した棋譜で学習

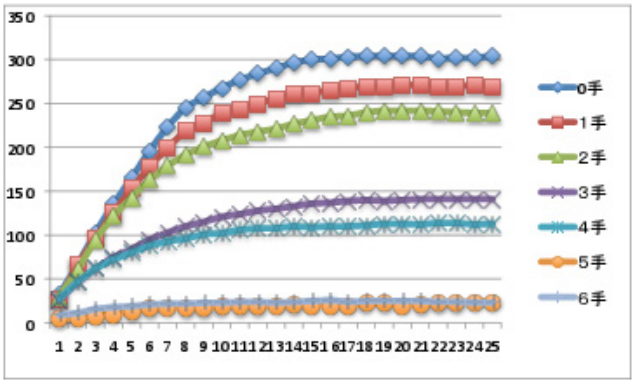

図 7 「入玉ステップ数」の重みパラメータの学習（EkingStep-Both）
Fig. 7 Learning the weight parameter of “entering-king step”.

表 2 「入玉ステップ数」の重みパラメータ
Table 2 The weight parameter of “entering-king step”.

入玉ステップ数	0	1	2	3	4	5	6
EkingStep-Both	303	270	241	143	115	22	24
EkingStep-Winner	706	497	339	161	122	1	8
EkingStep-Random-Both	17	141	157	117	109	39	37
EkingStep-Random-Winner	73	136	169	132	109	40	25

した EkingStep-Random，EkingStep-Random-Winner はともに入玉したときの値が小さくなっている．

この表の EkingStep-Both，EkingStep-Winner の行を見ると，手数 0〜2，3〜4，5〜6 の 3 段階ではっきりとした差があることが分かる．手数 5〜6 は自分の陣地の 1 段目と 2 段目であり，自玉がまだ囲いの中にいる状態である．囲いによって自玉が自分の陣地の 1 段目にいるか 2 段目にいるかが異なるので，一概に入玉まで近い歩数 5 の方が手数 6 より評価が高いとはいえず，そのために差がほとんどないと思われる．手数 3〜4 は，自玉が囲いから出てきて危険な状態が多く，入玉ができるならば目指すべき状態であるといえる．0〜2 は入玉までの手数が短く，入玉できる確率も大きいことが予想でき，そのために 3〜4 よりも差が出ていると思われる．全体としては入玉までの手数が 0 のときが一番評価値が大きく，手数が大きくなるにつれて，評価値が小さくなる傾向にあることが分かる．これは入玉をすると寄せられにくくなり，勝ちやすいという人間の感覚と合致している．

4.4 対戦実験（入玉ステップ数）

「入玉ステップ数」を考慮したプログラムについても対戦実験を行う．対戦相手はすべて標準の Bonanza（BonanzaStd）とする．対局数は BonanzaStd どうしも含めて，2,000 局に増やしたが，その他の実験条件は先ほどのものと同じである．ここで，「入玉ステップ数」を学習する際の学習対象として，入玉勝ち棋譜が有効であるか，勝った側の手だけを学習することが有効であるかを調べるために，実験に用いる将棋プログラムは以下の 5 通りとした．

表 3 対戦結果（棋譜・学習対象変更）
Table 3 Match results (variations of EkingStep).

プログラム	先手番			後手番			勝率	入玉勝率（%）
	勝ち (入玉)	負け	引き分け	勝ち (入玉)	負け	引き分け		
BonanzaStd	974(46)	994	32	980(60)	988	32	0.5	2.7
EkingStep-Both	503(67)	487	10	557(55)	431	12	0.536	6.1
EkingStep-Winner	482(80)	502	16	560(90)	430	10	0.528	8.5
EkingStep-Random	476(30)	510	14	489(28)	486	25	0.492	2.9
EkingStep-Random-Winner	484(37)	502	14	530(31)	459	11	0.513	3.4

BonaznaStd 標準の Bonanza.

EkingStep-Both 入玉勝ち棋譜で両者の手を学習した「入玉ステップ数」の特徴を評価関数に加えたもの．重みパラメータは反復回数 30 回後のもの．

EkingStep-Winner 入玉勝ち棋譜で勝った方の指し手のみを学習した「入玉ステップ数」の特徴を評価関数に加えたもの．重みパラメータは反復回数 30 回後のもの．

EkingStep-Random-Both 乱数で抽出した棋譜で両者の手を学習した「入玉ステップ数」の特徴を評価関数に加えたもの．重みパラメータは反復回数 30 回後のもの．

EkingStep-Random-Winner 乱数で抽出した棋譜で勝った方の指し手のみを学習した「入玉ステップ数」の特徴を評価関数に加えたもの．重みパラメータは反復回数 30 回後のもの．

これらのプログラムの対戦結果を**表 3** に示す．

入玉勝ち棋譜で両者の指し手を学習した「入玉ステップ数」の特徴を加えた EkingStep-Both の入玉勝率は，フィッシャーの正確確率検定により BonanzaStd の入玉勝率を有意に上回った．また，EkingStep-Both の勝率は先手番，後手番ともに 5 割を上回り，カイ二乗検定によって有意に勝ち越すことが確認できた．

入玉勝ち棋譜で勝った方の指し手のみを学習した EkingStep-Winner は，入玉勝率は全対戦中最高の 8.5%となり，フィッシャーの正確確率検定により EkingStep-Both を有意に上回った．勝率に関しては 5 割を上回ったが，カイ二乗検定によって有意に勝ち越すほどの差は見られなかった．

乱数で抽出した棋譜で学習した EkingStep-Random-Both は勝率は 0.496 で有意に負け越さず，入玉勝率は 2.9%で有意に向上したとはいえない．

フィッシャーの正確確率検定により，乱数で抽出した棋譜で勝った方の指し手のみを学習した EkingStep-Random-Winner は勝率こそ有意に勝ち越してはいないが，入玉勝率は BonanzaStd の入玉勝率を有意に上回った．

「入玉ステップ数」は評価関数が呼ばれるたびに双方の玉に関して計算するので，「入玉ステップ数」を特徴に加えることにより，単位時間あたりの探索ノード数は減少する．「コンピュータ将棋の進歩 2」[10] の問題集のうちの最初の 10 問を解かせたところ，Opteron 2376（2.3 GHz）1core では標準の Bonanza で 1.53×10^5 Nodes/second の探索が行われたのに対して，「入玉ステップ数」を加えたものは，1.39×10^5 Nodes/second と 10%近くの速度低下が見られた．実装の工夫により速度低下は抑えられる可能性はあるが，現在の実装で持ち時間で制限した場合は勝率では標準の Bonanza を下回る可能性が高い．

入玉ステップ数を特徴として重みパラメータを学習したプログラムが，勝率を落とさずにオリジナルよりも高い入玉勝率を達成したが，以下のような疑問をいだく人もいるだろう．

- 「入玉ステップ数」は，「入玉可能」と判断されたときの値は玉の段数によって決まる．「玉の段数」を特徴として，重みパラメータを学習させれば同様の結果が得られるのではないか?
- 「入玉ステップ数」を特徴として学習した結果得られた重みが，入玉確定時に歩 3.5 枚分というのは小さく感じられる．学習された重みパラメータよりも大きな値を与えれば，もっと入玉率が上がるのではないか?

この疑問に答えるために以下のような比較実験を行った．実験に用いるプログラムは以下のとおり．

EkingStep*2 EkingStep-Both の重みパラメータを 2 倍して加えたもの．

EkingStep*4 EkingStep-Both の重みパラメータを 4 倍して加えたもの．

EkingStep*10 EkingStep-Both の重みパラメータを 10 倍して加えたもの．

KingPosition 「玉の段数」を入玉ステップ数と同様に 7 段階に分けて特徴に加え，この重みパラメータだけを学習したもの．

この対戦結果を**表 4** に示す．参考のため，表 3 の EkingStep-Both の結果を再掲している．

「入玉ステップ数」を定数倍した EkingStep*{4,10} は EkingStep-Both より入玉勝率が下回り，BonanzaStd より勝率が下回っていることはカイ二乗検定により確認できた．KingPosition もほぼ同様の結果となった．「入玉ステップ数」を定数倍した EkingStep*2 は EkingStep-Both より入玉勝率が下回り，勝率が 5 割を超えたが，有意な差は見られ

表 4 対戦結果（特徴変更）
Table 4 Match results (EkingSteps whose parameters are amplified and program with evaluation function with simpler features).

プログラム	先手番			後手番			勝率	入玉勝率（%）
	勝ち (入玉)	負け	引き分け	勝ち (入玉)	負け	引き分け		
EkingStep-Both	503(67)	487	10	557(55)	431	12	0.536	6.1
EkingStep*2	477(32)	511	12	523(32)	458	19	0.508	3.2
EkingStep*4	413(24)	572	12	473(27)	518	8	0.448	2.6
EkingStep*10	378(23)	610	12	424(28)	564	12	0.401	2.6
KingPosition	462(36)	523	15	509(23)	478	13	0.493	3

ない．これらの結果により，「入玉ステップ数」に関して学習した重みパラメータは勝率を落とさずに高い入玉勝率をあげるという意味では適切な値に学習されたことが分かる．

5. おわりに

本稿では，オープンソースの将棋プログラム Bonanza をベースに入玉指向の将棋プログラムを作成するために，元の評価関数の特徴を増やさずに，入玉勝ちの棋譜のみから重みパラメータを学習する方法，「入玉ステップ数」という新たな特徴を導入して，重みパラメータを学習する方法を提案し，実験を行った．どちらの方法でも入玉勝率は有意に上昇したが，特に「入玉ステップ数」を導入したものに関しては，勝率を落とすことなく入玉勝率を増やすことに成功した．

2 人ゲームにおいて，「入玉ステップ数」のように評価関数に軽い探索の結果得られる特徴を用いて，機械学習により重みを調整するのは従来，一般的に用いられなかった手法だが，将棋においては「玉の広さ」を表す特徴，また，将棋以外のゲームへの適用が期待できる．

なお，持将棋における勝ち宣言のルールがコンピュータ将棋どうしの対局ではポピュラーなものとなっている．本稿では入玉するまでを目的として特徴を導入したが，相入玉した後で，宣言勝ちをするために味方の駒を敵陣に入れたり，大駒を捕獲したりするなどの戦略に対応した特徴は扱っていない．そもそも，現在公開されている Bonanza には宣言勝ちを行う機能が組み込まれていないが，それを含めて入玉後に勝ち切るための改良は今後必要になると思われる．

また，今回は時間の関係で，学習対象の棋譜の数や学習時の探索深さが十分得られなかったため，新たな特徴の重みパラメータのみを学習して，既存の重みパラメータは固定化した．しかし，文献 [11] に提案されている既存の重みパラメータを活かす形で学習する方法も適用可能であり，これを用いて十分な棋譜数，十分な探索深さで学習を行えば，さらに良質の評価関数が得られる可能性も期待できる．

参考文献

[1] 松原 仁（編）：あから 2010 勝利への道，情報処理学会誌，Vol.52, No.2, pp.152–190 (2011).
[2] 古作 登：コンピュータ将棋の不遜な挑戦：6．最強将棋ソフト「激指」との戦いに学ぶ—対コンピュータ戦略の必要性，情報処理学会誌，Vol.51, No.8, pp.1018–1021 (2010).
[3] 久米 宏：将棋倶楽部 24 万局集，ナイタイ出版 (2002).
[4] 久米 宏：最強の棋譜データベース，成甲書房 (2004).
[5] 片上大輔，山本一成：コンピュータは七冠の夢を見るか?，将棋世界 2010 年 3 月号，pp.160–165 (2010).
[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 2006, pp.78–83 (2006).
[7] Hoki, K. and Masakazu, M.: Efficiency of three forward-pruning techniques in shogi: Futility pruning, null-move pruning, and Late Move Reduction (LMR), *Entertainment Computing* (online), DOI: 10.1016/j.entcom.2011.11.003 (2011).
[8] Lieberum, J.: An evaluation function for the game of amazons, *Theoretical Computer Science*, Vol.349, pp.230–244 (2005).
[9] 生井智司，伊藤毅志：将棋における棋風を感じさせる AI の試作，情報処理学会研究報告，Vol.2010-GI-24, No.3, pp.1–7 (2010).
[10] 松原 仁（編著）：コンピュター将棋の進歩 2，共立出版 (1998).
[11] 矢野友貴，三輪 誠，横山大作，近山 隆：既存評価関数のパラメタを活かした適応学習，第 14 回ゲームプログラミングワークショップ 2009，pp.1–8 (2009).

滝瀬 竜司 （学生会員）

1987 年生．2011 年東京農工大学工学部情報工学科卒業．現在は東京大学大学院総合文化研究科広域科学専攻広域システム科学系修士課程に在籍．

田中 哲朗 （正会員）

1965 年生．1987 年東京大学工学部計数工学科卒業．1992 年東京大学大学院博士課程修了．博士（工学）．東京大学工学部助手，東京大学教育用計算機センター助教授を経て，現在は東京大学情報基盤センター准教授．日本ソフトウェア科学会，ACM 各会員．