静止探索を用いた Arimaa 評価関数の比較学習

川上　裕生[1,a)]　鶴岡　慶雅[1]

概要：Arimaa はチェス・将棋・囲碁などのゲームと比べてゲーム木のサイズが非常に大きく、コンピュータにとって難しいゲームである。チェスや将棋などのゲームでは、評価関数の学習手法として比較学習がよく用いられており、 Arimaa においても実用的な評価関数を比較学習を用いて学習することを目指した。学習時に探索を行わない場合と学習時に静止探索を行う場合で実験を行った結果、どちらでも人手で作成された評価関数よりも優れた評価関数を学習することができた。しかし、対戦実験では静止探索の有無による評価関数の勝率の差は殆ど見られず、静止探索の手法に改善の余地が残されていると考えられる。

Comparison Training of Arimaa Evaluation Functions with Quiescence Search

Yusei Kawakami[1,a)]　Yoshimasa Tsuruoka[1]

Abstract:
Arimaa is a game that is more difficult for computers than chess and shogi. This is partly because the number of legal moves is much larger in Arimaa. Comparison training is often used for learning an evaluation function for a shogi program. In this work, we attempted to learn a practical evaluation function for Arimaa from game records. We built evaluation functions by comparison training with and without search. The evaluation function learned without search and the one with quiescence search are stronger than the original handmade evaluation function. There is little difference between the evaluation function with search and the one without search. There is room for improvement in quiescence search.

1. はじめに

ゲーム AI は人工知能研究の題材のひとつとして古くから研究が行われている。ゲームを研究題材として用いる理由は、ルールが明確であること、勝敗により性能の評価が行い易いこと、人間にとって面白いものであることなどがあげられる。

機械学習を用いてコンピュータゲームの性能を向上させる研究はゲーム AI の研究の初期の頃から行われてきている [1]。機械学習を用いることで、そのゲームに対する深い知識を持たない人でもゲーム AI の作成が可能になり、手作業でヒューリスティックを組み込む必要もなくなるため手間も削減できる。また、機械学習により人間がうまく説明できないような “直感”をコンピュータに理解させることや、まだ人間が理解していないゲームの深い知識を得ることが期待できる。

本論文では Arimaa というゲームを対象とする。 Arimaa はチェス・将棋・囲碁などと同じ、二人零和有限確定完全情報ゲームに分類されるゲームである。 Arimaa は 2002 年に Omar Syed 氏が、AI にとっては難しいものの、ルール自体は子供でも理解できるというコンセプトで作成したゲームである。このゲームが考案された契機として、1997 年にチェスプログラム、ディープブルーが世界チャンピオン相手に勝利したことがあげられる。

Arimaa が AI にとって難しい理由を説明する。まず一つ目に、一ターンごとの合法手の数が非常に多いことがあげられる。 Arimaa では一ターンに 4 回の移動を行うことができ、そのため一ターンに平均して 18,000 通りの合法手が存在する [2]。類似した他のゲームにおける一ターンの合法手の数の平均は、チェスでは 35 通り、将棋では 80 通り [3]、囲碁では 250 通り程度であり、 Arimaa の合法手の数は桁違いに多いということが分かる。合法

[1] 東京大学大学院工学系研究科
Graduate School of Engineering, The University of Tokyo
[a)] kawayu@logos.t.u-tokyo.ac.jp

手の数が多いことによりゲーム木のサイズが非常に大きくなるため、単純な力任せの探索方法では深い探索を行うことができない。二つ目に、駒の初期配置をプレイヤーが自由に決められるということがあげられる。ゲーム開始時に、先手番のプレイヤーが手前二列に好きなように駒を配置したあとに、後手番のプレイヤーも駒を配置する。この駒の初期配置の方法だけでも 4.2×10^{15} 通り存在し、このゲームを複雑にする一因となっている。また、初期配置が一つに定まっていないため、チェスや将棋などのゲーム AI で行われている定跡の利用が難しくなっている。これらの要因により、 Arimaa はコンピュータにとって難しいゲームになっている。Arimaa の AI の研究を行うことにより、 AI が人間の頭脳に追いつき、追い越すための鍵となることが期待される。

Arimaa の評価関数を機械学習を用いて行った研究を 3 つ紹介する。まず一つ目に強化学習による Arimaa の評価関数の学習である [4]。この研究では、 TD, TD-Leaf, Rootstrap, Treestrap の 4 つの学習手法を用いて評価関数を学習している。対戦実験において学習前の評価関数を利用したものと比較して有意に勝率が向上しており、強化学習を利用することで評価関数を改善できることが分かる。二つ目に、将棋プログラム Bonanza [5] で用いられている比較学習 (comparison training) [6] という学習手法を Arimaa に適応したもの研究がある [7]。しかしこの論文では、駒の位置に対する重みを学習したにとどまっており、Arimaa の評価関数に有効であると考えられる他の特徴量を使用していない。そのため、プログラムの強さも、既存のプログラムより劣るものとなっている。三つ目に、Hřebejk らの研究 [8] では、TD 誤差学習と比較学習を組み合わせつつ、線形計画法で解ける問題として定式化することで、深さが 1 より大きい探索を行わずに有用な評価関数の学習に成功している。

本研究では、静止探索 [9] と比較学習 [6] を用いて Arimaa の評価関数を学習することを目指す。チェスや将棋などのゲームでは、棋譜を利用した学習手法である比較学習を利用することで、有用な評価関数を学習することに成功している。そのため、Arimaa においても比較学習をうまく利用することで、有用な評価関数の学習が可能なのではないかと考えられる。

本稿の構成を以下に述べる。第 2 章でゲーム AI に関する一般的な話を行う。第 3 章で Arimaa のルールについて簡単に説明する。第 4 章で評価関数の学習に関する関連研究の紹介、第 5 章で本研究の提案手法の説明を行う。第 6 章で評価の結果を示し、第 7 章で考察、第 8 章でまとめと今後の課題について述べる。

2. ゲーム AI

2.1 ゲーム木

ゲームのある局面をノード、選択される指し手をエッジとした木構造で、ゲームの進行を表すことができる。

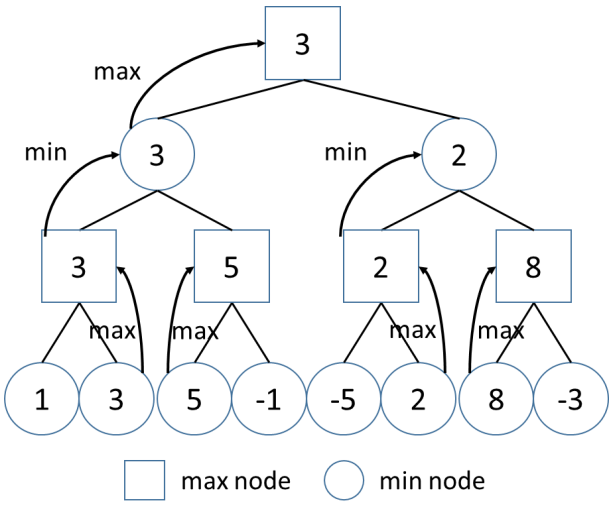

図 1 Minimax 探索

これをゲーム木と言う。ゲーム木を探索することで、先の展開を読むことが可能になる。代表的な手法として、Minimax 探索が存在する。

ゲームの複雑さを示す指標の一つとして、ゲーム木の大きさがあげられる。ゲーム木が大きくなるほど、コンピュータにとって難しくなることが多い。例えばゲーム木の大きさが 10^{30} 程度と比較的小さいチェッカーでは、両プレイヤが最善手を指し続けた場合に引き分けになることが既に示されている [10]。Arimaa のゲーム木の大きさは 10^{400} 程度であり [4]、チェスの 10^{120}、将棋の 10^{220} と比べて非常に大きい。

代表的なゲーム木探索の手法の一つとして Minimax 探索があげられる。Minimax 探索は、自分も相手も常に最善手を指すという前提で探索を行っていくアルゴリズムである。Minimax 探索の様子を図 1 に示す。評価値が大きいほど自分にとって有利な局面であることを示しており、自分の手番 (max node) では評価値が大きい方を、相手の手番 (min node) では評価値が小さい方が選択される。

2.2 評価関数

ゲームが終局となるゲーム木の末端まで、全ての局面を探索することができれば、そのゲームの勝敗を完全に予想することが可能になる。しかし、ゲームが終了するまで読み切ることは現実的な時間では不可能であることが多い。そのため、ゲームが終了していない局面についても有利・不利の判断を行いながら、探索をする必要がある。局面の有利・不利を数値化したものを評価値と呼び、評価値を計算するための関数を評価関数を呼ぶ。完璧な評価関数が分かれば、最善手が何であるかわかることになるため探索を行う必要がなくなる。しかし、完全な評価関数を作成することも多くの場合不可能であるため、探索と評価関数を組み合わせて利用する必要がある。精度の高い評価関数の計算には時間がかかるため、探索できるノード数が減少してしまう。そのため、評価関数

の精度と探索の深さのバランスをとる必要がある。実際には、短時間で計算できるような評価関数を用いて、長時間かけて探索を行うことがよく行われる。

評価関数は、局面の特徴ベクトルと重みベクトルの内積の形で表されることが多い。局面からどのように特徴量を抽出するかは、多くの特徴量の中から学習に有用な特徴を選択するような研究も行われている [11] が、ゲームに関する知識を持った人の手によって選択されることが多い。重みベクトルは、機械学習を利用して調整されることが多い。

3. Arimaa のルール

Arimaa では、8 × 8 のマス目を持った盤を用いる。駒はプレイヤーごとに象とラクダが各 1 個、馬と犬と猫が各 2 個、ウサギが 8 個である。盤のサイズがチェスと同じであり、また駒の種類と数もチェスと同じであるため、チェスの駒と盤を流用できるようになっている。駒の強さは強い方から順に象、ラクダ、馬、犬、猫、ウサギの順になっており、これは後に述べる PUSH や PULL、FREEZE に影響する。8 個のウサギのうちいずれか 1 個を反対側の一番奥の列まで先に到達させたプレイヤーの勝利となる。

駒の初期配置

Arimaa ではチェスや将棋などとは異なり、駒の初期配置は決まっておらず、プレイヤーが最初に決める必要がある。まず最初に先手のプレイヤーが手前二列に自分の駒を好きなように配置する。先手が全ての駒を配置したら、後手も同様に自分の駒を配置する。その後は交互に手番を消費して、駒を動かしていく。駒の初期配置の一例を図 2 に示す。

駒の動き

プレイヤーは一度の手番において最大 4 ステップまで動かすことができる。自分の駒を一度動かす行動は 1 ステップ分である。各駒は隣接する上下左右の空きマスに移動することができる。しかしウサギは後ろに移動することはできない。一度の手番で一つの駒を何度も動かすこともできるし、複数の駒を動かすこともできる。しかし、自分よりも強い相手の駒が隣接しており、かつ味方の駒が隣接していない場合、その駒は FREEZE し動かすことができない。4 ステップ全てを消費せずにパスすることも可能であるが、最低でも 1 ステップは消費しなければならない。

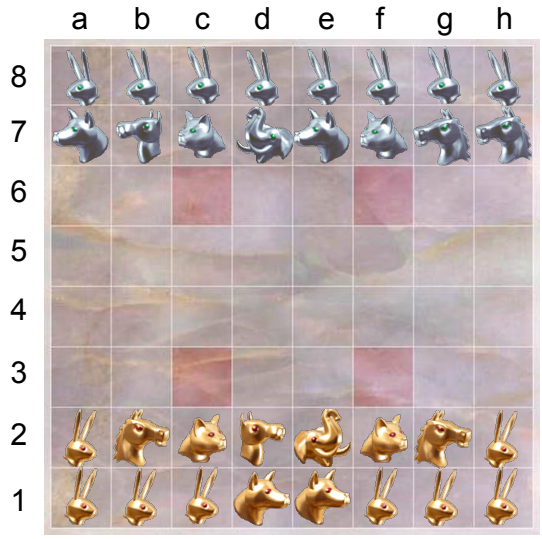

図 2 Arimaa の駒の初期配置の一例 [12]

Arimaa には PUSH や PULL という特別な動きが存在し、これにより相手の駒も動かすことができる。 PUSH は、自分の駒と隣接している相手の駒を任意の方向の空きマスに移動させ、相手の駒が移動する前の位置に自分の駒を移動させる。 PULL は相手の駒と隣接している自分の駒を移動させた後に、隣接していた相手の駒を自分の駒が移動する前の位置に移動させる。 PUSH や PULL は自分の駒と相手の駒の移動で計 2 ステップ消費したことになる。PUSH や PULL は自分より弱い駒に対してのみ可能であり、自分と同じ強さの駒や自分より強い駒に対しては行うことができない。

トラップ

c3, c6, f3, f6 の 4 つのマスにはトラップが存在する。トラップ上にいる駒は、味方の駒が隣接していない場合ゲームから除外される。PUSH や PULL を用いて相手の駒をトラップに落とすことができ、そのようにして相手の駒を減らしていくことがこのゲームの序盤、中盤における基本的な戦術となる [13]。

4. 関連研究

4.1 静止探索

静止探索は、通常の探索の後に評価値が大きく変動しそうな指し手に限って探索を行い、得られた局面の評価値を最終的な評価値として利用することで、局面の評価値を安定させる手法である [9]。チェス、将棋などのゲームでは、駒の奪い合いや、駒を成る手が生じると大きく評価値が変動する。そのため、数手先で大きく評価値が変動するにも関わらず、水平線効果により正確な評価値を求めることができないという状況が発生する。探索する指し手を限定して、駒の取り合いなどのない “静止した局面” になるまで探索を行なうことで、水平線効果を防ぎ、短時間で安定した評価値を得ることが可能になる。

Arimaa における静止探索は、Wu の研究 [4] や、ネット上で公開されている Arimaa プログラムの一つである OpFor [14] などで実装が行われているものの、定量的な評価は行われていない。静止探索を行なう際には、どのような指し手を探索の対象に含めるかが問題となる。 Wu の研究では駒を取る手と駒を取られるのを防ぐ手、 OpFor では駒を取る手を静止探索に利用している。

Arimaa では、水平線効果により、数手先で両方の駒を取られてしまうにも関わらず、本来取られずに済む自分の弱い駒をトラップの隣に移動させることで、強い駒が取られるのを遅らせるような手を選択してしまうことが考えられる。静止探索を行なうことで、取られてしまう駒を意味もなく増やしてしまうような手が選択されることを防ぐことが期待される。

4.2 比較学習

Arimaa に類似したゲームであるチェスや将棋の評価関数学習手法としてよく用いられている、比較学習 [6] の研究について述べる。比較学習は上級者の棋譜を利用した評価関数の学習手法であり、プログラムが上級者の指し手を真似するように重みベクトルが調整される。

チェスにおける比較学習では、棋譜中の局面から棋譜の指し手を一手指した局面と、他の手を一手指した局面を比較してもうまくいかないことが明らかになっている。指し手の目的、狙いをプログラムに理解させるために、静止探索や浅い探索を行なうことで、有用な評価関数の学習に成功した [15]。

将棋では、 Bonanza [5] が評価関数をプロ棋士の棋譜から学習することに成功して以来、教師あり学習がよく用いられている。Bonanza で用いられている学習手法は、比較学習をベースとした手法である。数式で表すと以下の誤差関数 J を最小化する特徴ベクトル $\mathbf{v}$ の最適化問題となる。

$$J(P_0, ..., \mathbf{v}) = \sum_{i=0}^{N-1} \sum_{m=1}^{M} T[\xi(p_m, \mathbf{v}) - \xi(p_{m=0}, \mathbf{v})] \quad (1)$$

ただし N は学習データの局面数、P_i は局面、M はその局面における合法手の数、ξ は局面の評価関数、T は評価値の差を棋譜の指し手との一致度に変換する関数であり、シグモイド関数などが用いられる。p_m は局面 P の指し手 m の後の局面を探索して得られた最善応手手順の末端局面である。$m = 0$ は棋譜の指し手である。これに、過学習を防ぐための正則化項を加えたものを目的関数として、最急降下法により目的関数 J を最小化することで重みベクトルの学習を行う。

また、激指 [16] では比較学習と平均化マージンパーセプトロンを利用した手法が用いられている [17]。この手法では、ある局面 P に対する指し手 m のあとの最善応手手順を探索し、得られた末端局面を利用した学習を行う。その局面に対する評価値を返す関数を $\xi(p_m, \mathbf{v})$ とし、全ての合法手の集合を M とする。まず棋譜の指し手が他の指し手と比べて相対的に高く評価されているかどうかを確認し、棋譜の指し手より評価値が高い、もしくはある margin を持って評価値が近いものを発見する。具体的には局面 P における棋譜の指し手 $m = 0$ に対して

$$M' = \{m \in M | \xi(p_m, \mathbf{v}) + \text{margin} > \xi(p_{m=0}, \mathbf{v})\} \quad (2)$$

となる指し手の集合 M' を求める。M' に含まれている指し手は棋譜の指し手と比較して評価値が十分小さくはないということになる。

全ての $m \in M'$ の特徴ベクトルについて棋譜の指し手の特徴ベクトルとの差を取り、その平均を重みベクトル $\mathbf{v}$ に加える。式で表すと式 (3) のようになる。

$$\mathbf{v} \leftarrow \mathbf{v} + \frac{1}{|M'|} \sum_{m \in M'} (\phi(p_{m=0}) - \phi(p_m)) \quad (3)$$

$\phi(P)$ は、局面 P の特徴ベクトルである。それを全ての局面に対して繰り返し、最後に学習途中に現れた重みベクトルの平均を取り、それを最終的な重みベクトルとしている。

4.3 Arimaa の評価関数の学習

Arimaa の評価関数の機械学習として、Hřebejk らの研究 [8] について述べる。TD 誤差学習と比較学習を組み合わせつつ、線形計画法で解ける問題として定式化している。まず以下のように a,b,c,d を定義する。

$$a = |f(\text{current}) - f(\text{played})| \quad (4)$$

$$b = |f(\text{current}) - f(\text{best})| \quad (5)$$

$$c = f(\text{best}) - f(\text{played}) \quad (6)$$

$$d = f(\text{played}) - \text{average} \quad (7)$$

f(current) は教師データの局面の評価値、f(played) は教師データの指し手を指した局面の評価値、f(best) は最も評価値が高い指し手を指した局面の評価値、average は全指し手の評価値の平均である。a,b,c,d を用いて各教師データの局面に対する誤差関数を以下のように定める。

$$l(p) = \alpha a + \beta \max(a, b) + \gamma c + \delta(0, G - d) \quad (8)$$

α,β,γ,δ,G はそれぞれ学習のパラメータである。第一項と第二項は TD 誤差学習、第三項は比較学習のような動作を行なう。第四項は評価値のスケーリングに用いる。最終的な誤差関数は各局面について $l(p)$ を足しあわせ、正則化項を加えた

$$L = \sum_{p \in \text{training positions}} l(p) + \rho R \quad (9)$$

のような形となる。正則化項である R は各重みベクトルの要素 w_i の絶対値の和である。

$$R = \sum_i |w_i| \quad (10)$$

このようにして定式化した誤差関数を線形計画問題に変形し定式化することで学習を行っている。これは学習時に深さが 2 手以上の探索を行わないため、評価関数の重みベクトルが変化しても各局面の評価値を調べることが容易なため可能である。例えば第一項である αa は、各局面 i に対して変数に a_i,a'_i を導入し、制約として

$$(x_{j,\text{current}} - y_{j,\text{played}}) \tag{11}$$

を用いる。線形計画法を用いることによって誤差関数の最適化にかかる時間を減らし、有用な評価関数の学習に成功している。

また、川上らの研究では比較学習を用いた評価関数の学習を行っている [18]。この研究では教師データに対する一致率は上昇しているものの、最終的な強さの向上には至らなかった。その原因として、特徴量が不足していることや、学習時の探索に時間がかかってしまうことがあげられている。

5. 静止探索を用いた Arimaa 評価関数の比較学習

本研究では、 Arimaa の実用に足る評価関数を棋譜から学習することを目指す。

Arimaa において将棋と同様の手法で評価関数の学習を行おうとした場合の問題点として、将棋では浅い探索を非常に短い時間で行うことができるが、 Arimaa では浅い探索を行うことにすら時間がかかってしまうことがあげられる。

学習時に探索を行わない場合と、1 手探索＋静止探索を行なう場合の 2 つの手法で学習を行った。探索を行わない場合は、棋譜の指し手を指した局面の特徴量と、ランダムな合法手を指した局面の特徴量の比較による学習を行った。この場合は探索を行っていないため、多くの局面との評価値の比較を行なうことができる。静止探索を行なう場合は、棋譜の指し手を指した局面から静止探索を行った局面と、1 手探索を行い得られた局面から静止探索を行った局面の比較による学習を行った。静止探索は最大 2 手である。学習の手順を図 3、図 4 に示す。

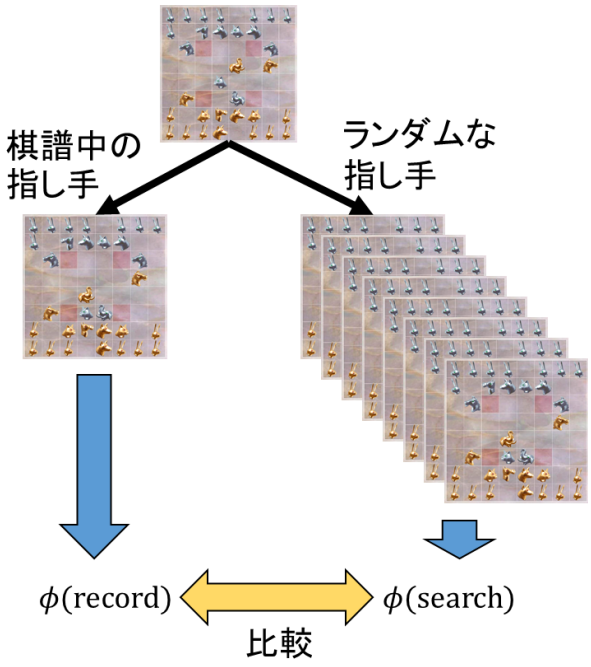

図 3 探索を用いない学習

Arimaa における静止探索では、1 手の指し手の中にも、評価値を大きく変動させる静止探索に利用すべきステップと評価値を殆ど変動させないステップが混在していることが考えられる。そのため、今回は駒を取ることに関連しそうなステップのみを生成し、それらのステップを組み合わせて駒を取る手を生成し、探索に用いた。生成した各ステップの駒の動きを以下に示す。

- 相手の駒を PUSH または PULL することでトラップ上に移動させる動き
- 相手の駒を PUSH または PULL することでトラップに隣接するマスに移動させる動き
- トラップに隣接している相手の駒に、その駒よりも強い自分の駒で隣接する動き
- トラップ上にいる相手の駒を守っているトラップに隣接している相手の駒を PUSH または PULL する動き

6. 評価

6.1 評価設定

インターネット上で公開されている Arimaa プログラムである Faerie を用いて実験を行った。 Faerie の評価関数は人間が手でつけた重みが用いられている。学習用の棋譜としては、 Arimaa のウェブサイト [12] からダウンロードできる棋譜約 30 万局 (2015 年 9 月時点) のうち、対戦者の Rating が 2,200 を超えている棋譜 2,321 局 (186,775 局面) を利用した。

学習の手法は 3.2 節で述べた平均化マージンパーセプトロンによる学習手法をベースとした。探索を行わない

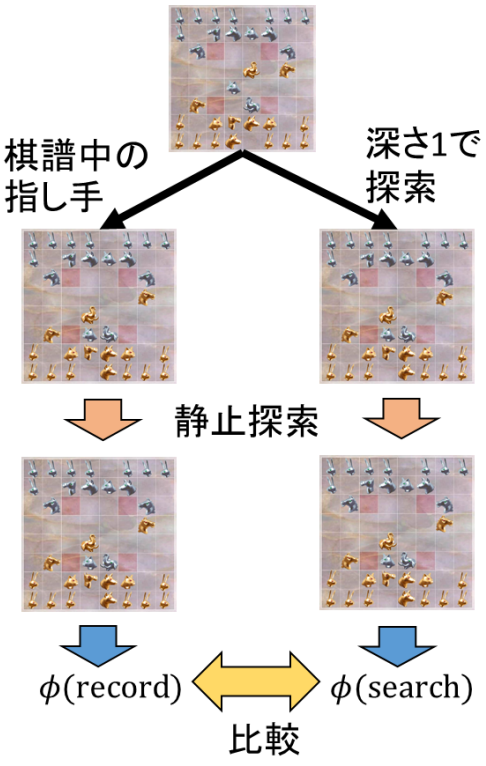

図 4 静止探索を用いた学習

場合は、棋譜中の各局面について棋譜の指し手とは別に 1,000 手を比較対象として用い、186,775 局面を学習に用いた。静止探索を行なう場合は 100,000 局面を学習に用いた。重みベクトルの初期値として、 Faerie で利用されている重みを用いた。

学習した評価関数は棋譜との指し手の一致率と、対戦実験によって性能を測定した。対戦実験では、評価関数を学習していないオリジナルの Faerie 、探索を行わずに学習した評価関数、静止探索を用いて学習した評価関数のそれぞれを対戦させた。対戦者の Rating が 2,000 を超えている棋譜から、重複のないように初期局面を選び出し、先手後手を入れ替えて対戦を行った。対戦時は探索の深さを 2 に固定した。

棋譜との指し手の一致率はテストデータ用の棋譜として Rating が 2,200 を超えている棋譜 126 局 (8,819 ターン) を利用した。学習した評価関数で探索を行い得られた最善手と、棋譜中の指し手との比較を行った。 Arimaa では一度の手番において最大 4 ステップ分の動きが可能であり、異なる指し手が 4 ステップ中 3 ステップは同じ動きをしていたり、異なる指し手であっても手番終了時の局面が同一であったりすることが考えられる。そのため、各手番において同じステップが出現する回数と、手番終了時の局面が一致する回数を調査した。

6.2 特徴量

評価関数の特徴量は、 Faerie で用いられていた特徴量に、 Arimaa の評価関数に有用であると考えられる特徴量を追加して学習を行った。Faerie の特徴量は大きく分けて「トラップに関する特徴量」、「駒の価値に関する特徴量」、「ウサギに関する特徴量」の 3 つから成り立っている。

トラップ に関する特徴量

- 各トラップに隣接している先手の駒の強さの合計
- 各トラップに隣接している後手の駒の強さの合計
- 各トラップに隣接している駒のうち最も強い駒を持つプレイヤー

駒の強さは象から順に 6, 5, 4, 3, 2, 1 とする。

駒の価値に関する特徴量

Faerie の駒の価値に関する評価関数は、駒の種類をベースとして、その駒に対して各フラグが立っているかどうかを利用して重みを計算しており、特徴量の線形和としては設計されていない。そのため、学習を行う際には “フラグ 1, 2, 4 が成立している象” の重みといったように特徴量を書き換えている。

各フラグを以下に示す。

- その駒が FREEZE しているか
- 隣接する空きマスがあるか
- トラップに隣接している場合、他に同じトラップに隣接する味方の駒が他にいるか
- トラップから二マス離れた地点にいる場合、トラップの隣が空きマスか
- トラップ上にいる場合、複数の敵の駒に隣接されているか
- 犬や猫の駒の場合、その駒のいる列が 3 列目以降か

ウサギに関する特徴量

- 各ウサギのいる列
- 七列目にいるウサギの前と左右が空きマスか、味方の駒か、それともそれ以外か

これらの Faerie で元々利用されていた特徴量に加えて、 Wu の研究 [4] を参考に、以下の特徴量を追加した。

- 各自分の駒より強い相手の駒がいくつあるか
- 残っているウサギの数
- 駒の種類と列
- 駒の種類とその駒のいるマス
- あと何ステップでゴールできるか
- トラップの周囲にいる駒の数と象が含まれているか否か
- あと何ステップで駒を取ることができるか
- 象が 3,4 ステップ以内にいくつのマスに移動可能か

6.3 実験結果

探索を行わない場合は 186,775 局面に対して学習を 10 回、静止探索を行った場合では 100,000 局面に対して 1 回行った。[*1] 学習時間は探索を行わない場合で約 19 時間、静止探索を行った場合で 80 時間であった。CPU は Intel Xeon CPU X5680 3.33GHz を用いた。

実験の結果を表 1 に示す。探索なしで学習した評価関数、静止探索を用いて学習した評価関数は共に Faerie に対して有意に勝ち越している。棋譜を利用して学習を行なうことで、 Arimaa の有用な評価関数を学習することができたと言える。

テストデータとの一致率は、 Faerie 、探索なしで学習した評価関数、静止探索を用いて学習した評価関数のどれも大きな差は見られなかった。この理由としては、棋譜中で最善手が指されていないため一致率と強さの間の相関が低くなっている、Arimaa では最善手と次善手の差が殆どない局面が多く存在するなどの可能性が考えられる。

探索なしで学習した評価関数と静止探索を用いて学習した評価関数との対戦における勝率は統計的に有意な差が見られなかった。チェスや将棋などのゲームでは、学習時の静止探索は有効に働くことが知られており、Arimaa における静止探索の手法に改善の余地があると考えられる。

7. 考察

今回の実験では、学習時に探索を行わない場合の評価関数と学習時に静止探索を行った場合の評価関数の対戦

[*1] 1 回の学習とは、ランダムシャッフルされた全局面に関して式 2、3 による重みベクトルの更新を行うことを意味する。

表 1 勝率と一致率

評価関数	勝率		一致率	
	vs Faerie	vs 探索なし	ターン	ステップ
Faerie	50.0%		7.8%	21.8%
探索なし	56.7% (565 - 433)	50.0%	7.8%	21.8%
静止探索あり	60.2% (1203 - 795)	49.1% (981 - 1017)	7.4%	22.0%

実験では統計的に有意な差が見られなかった。

しかし、チェスにおける比較学習の実験では、1 手動かした局面での比較を行った場合は良い評価関数を学習できなかったものの、1 手＋静止探索や、4 手＋静止探索をおこなうことで、有用な評価関数を学習できたという結果が出ている [15]。浅い探索や静止探索を加えることで、指し手の表面的な特徴だけでなく、その指し手の狙いや目的のようなものを捉えることができるのではないかと考えられる。

今回の実験では、相手の駒を取ることに関連するステップのみを生成し、相手の駒を取る指し手を生成して静止探索を行なうことを目指したが、Freeze している味方の駒に隣接して移動可能な状態にする、移動の妨げとなっている自分の駒を移動するなどのステップを行わなければ相手の駒を取れない状況が存在するため、相手の駒を取る手の全てを生成しているわけではない。しかし、相手の駒を取る手の全てを生成すると、最初の 2 ステップで相手の駒を取ることが可能な場合に残りの 2 ステップのパターンが膨大になってしまい、静止探索の効率が悪化してしまうなどの問題点が考えられる。また、静止探索中に味方の駒が取られることを防ぐ手を生成するためには、“味方の駒が取られることを防ぐ手”であるかどうかを識別する方法が必要となってくる。これには相手の手番に駒が取る手が存在するかどうか、指し手の生成を行って確認する必要があるため、計算に時間がかかると思われる。

今後は、指し手や指し手中の各ステップの前後での評価値の変動を見ることで、静止探索を行なう際に生成すべき指し手と生成する必要のない指し手を判別するなど、Arimaa に対するヒューリスティックを利用せずに静止探索に利用する指し手を選択するを検討している。

8. 終わりに

本稿では、 Arimaa の評価関数の精度を高めるために、静止探索を用いて Arimaa の評価関数を学習することを目指した。探索を行わずに比較学習を行った場合と、静止探索を用いて比較学習を行う場合の二つの手法について評価を行った。実験の結果、探索を行わない場合と静止探索を行った場合の共に、学習前の評価関数よりも良い評価関数を学習することができた。しかし、静止探索の有無による精度の違いは見られなかった。今回の実験で行った静止探索はヒューリスティックな実装であり、改善の余地が残されていると考えられる。今後は、静止探索の手法を改善することで、評価関数の学習の改善を行っていきたい。

参考文献

[1] Samuel, A. L.: Some Studies in Machine Learning Using the Game of Checkers, *IBM J. Res. Dev.*, Vol. 3, No. 3, pp. 210–229 (1959).

[2] Brian, H.: A Look at the Arimaa Branching Factor, http://arimaa.janzert.com/bf_study/ (2006).

[3] 松原　仁，半田剣一：ゲームとしての将棋のいくつかの性質について，情報処理学会研究報告. 人工知能研究会報告， Vol. 94, No. 83, pp. 21–30 (1994).

[4] Wu, D. J.: Move Ranking and Evaluation in the Game of Arimaa, PhD Thesis, Harvard University (2011).

[5] 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).

[6] Tesauro, G.: Connectionist Learning of Expert Preferences by Comparison Training, *Advances in Neural Information Processing Systems 1* (Touretzky, D., ed.), Morgan-Kaufmann, pp. 99–106 (1989).

[7] Kanjanapa, T., Kanako, K. and Yoshiyuki, K.: Design and implementation of Bonanza Method for the Evaluation in the Game of Arimaa, Technical Report 4, Department of Computer and Information Sciences, Tokyo University of Agriculture and Technology (2012).

[8] Hřebejk, T.: Arimaa challenge-static evaluation function, Master's thesis, Charles University in Prague (2013).

[9] Harris, L. R.: The Heuristic Search and the Game of Chess - A Study of Quiescene, Sacrifices, and Plan Oriented Play, *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, Georgia, USSR, 3-8 September 1975*, pp. 334–339 (1975).

[10] Schaeffer, J., Burch, N., Björnsson, Y., Kishimoo, A., Müller, M., Lake, R., Lu, P. and Sutphen, S.: Checkers is solved, *science*, Vol. 317, No. 5844, pp. 1518–1522 (2007).

[11] 矢野友貴，三輪　誠，横山大作，近山　隆：ゲーム構成要素を組み合わせた特徴の最適化，第 15 回ゲームプログラミングワークショップ 2010，pp. 15–22 (2010).

[12] Syed, O.: Arimaa - Intuitively simple ... intellectually challenging, http://arimaa.com.

[13] Choksi, V., Ebrahim-Zadeh, N. and Mohan, V.: Leveraging Game Phase in Arimaa (2013).

[14] Brian, H.: OpFor - The "Opposing Force" Bot, Information Page, http://arimaa.janzert.com/opfor/.

[15] Tesauro, G.: Comparison training of chess evaluation functions, *Machines that learn to play games* (Fürnkranz, J. and Kubat, M., eds.), Nova Science Publishers, Inc., Commack, NY, USA, pp. 117–130 (2001).

[16] 鶴岡慶雅：将棋プログラム「激指」のページ，http://www.logos.t.u-tokyo.ac.jp/~gekisashi/.

[17] 鶴岡慶雅：3 選手権優勝記：激指の技術的改良の解説

(<ミニ特集>コンピュータ将棋の不遜な挑戦)，情報処理，Vol. 51, No. 8, pp. 1001–1007 (2010).
[18] 川上裕生，鶴岡慶雅：多様な特徴量を用いた Arimaa 評価関数の比較学習，ゲームプログラミングワークショップ 2014 論文集，Vol. 2014, pp. 151–156 (2014).