2008－GI－20 (2)
2008／6／27

進行度を用いたボナンザメソッドの提案

松井利樹† 橋本剛† 橋本隼一† 野口陽来†

†北陸先端科学技術大学院大学

本研究ではボナンザメソッドに進行度を用いた評価関数の学習法を提案する。将棋では一般的に序盤は駒の損得、終盤では寄せの速度が重要と言われており、ゲームの進行に応じて適切な値が変動していると考えられる。しかし、序盤から終盤まで単一な局面評価ではどうしても値が平均化されてしまうので良い値を得られないことがしばしばある。そこで従来の将棋プログラムで使われてきた局面の進行具合を表す変数(進行度)を使って表現し、より精度の高い評価関数を設計した。自己対戦の結果より提案した手法の有効性を確認した。

An Enhancement of The Bonanza Method by Game Progress Criteron

Toshiki Matsui†, Tsuyoshi Hashimoto†,

Junichi Hashimoto†, Haruki Noguchi†

†Japan Advanced Institute of Science and Technology

In the domain of shogi, the most important factors fluctuate depending on game progress. However usual evaluation functions consist from only one coefficient vector which is calculated as average of entire games. In this paper we propose a new method to handle the problem using a game progress criterion, called shinkoudo. The results of self-play games show that the proposed idea works efficiently.

1　はじめに

本研究は名人に勝つ将棋プログラムの作成を目標としている．現在のコンピュータ将棋のレベルはもはやアマチュアの人間が勝つことは不可能な領域にまできている．コンピュータ将棋が持つ大きな課題のひとつとして局面の形勢判断を行う評価関数の設計の難しさがあげられる．今までコンピュータ将棋の評価関数はプログラマによって調整されてきた。しかし2006年に世界コンピュータ将棋選手権において，強化学習によって評価関数を設計した保木のプログラムが優勝し，将棋のような複雑なゲームでも機械学習を用いる事が有効であると始めて実証された。そこで本研究では保木による提案手法を参考に評価関数の機械学習を行い、北陸先端大学飯田研究室で製作しているコンピュータ将棋ソフトTACOSに組み込む事で強化をはかる[1]。

保木による提案手法を簡単に説明すると強いプレイヤと同一の指し手を選択する評価関数の発見を目指し，古典的な勾配法を用いて最適制御を行うというものである．[2]この手法によって作られた評価関数は従来の評価関数にはない人間らしさと安定感を持っており非常に高く評価された。しかし保木の手法が持つ欠点としてゲームの進行に対応しない評価関数であることがあげられる．将棋では一般的に序盤は駒の損得、終盤では寄せの速度と言われており、ゲームの進行に応じて適切な値が変動していると考えられる。しかし、序盤から終盤までを単一のテーブルで表現してしまうと、どうしても値が平均化されてしい良い値を得られないことがしばしばある。本来はゲームの進行に合わせて適切な値を取る評価関数の設計が望ましい。もちろん存在しうる全ての局面に適切な値を割り振ることなど不可能である。局面に応じて変動する値を表現するために、従来の将棋プログラムで使われてきた局面の進行具合を表す変数(進行度)を使って表現し、より精度の高い評価関数を作ることを目標とする。

2　設計

2．1　進行度について

進行度は従来の将棋プログラム[3]でよく使われてきた概念で、ゲームの進行具合を表すための指標である。本研究でも序盤から終盤へ値を滑らかに変動させるために使っており、非常に重要な変数となっている。計算は主に駒の位置関係や成駒の数、持ち駒の数などで計算され、値域は0~1までの値で与えられる。

2．2　理論的枠組み

この節では提案手法の理論的な枠組みを述べる。序盤から終盤にかけて各特徴の評価値の変動を再現するために係数を序盤と終盤で分けて、進行度を使って評価関数を計算する事を行う。従来の評価関数は各係数の値を進行度によって0%から100%まで与えるものであったが、全ての項目の初期値が0というのは明らかにおかしい。今回は学習によって初期値の部分を得ることができるので、局面 p が与えられた時の評価関数を以下のように定義できる。

$$f(p)=(1-\alpha)l(p)^T v+\alpha l(p)^T w$$

局面：p
進行度：$\alpha(0 \le \alpha \le 1)$
特徴ベクトル：$l(p) := [l_1(p), \cdots, l_n(p)]^T$
序盤の係数ベクトル：$v := [v_1, v_2, \cdots, v_n]^T$
終盤の係数ベクトル：$w := [w_1, w_2, \cdots, w_n]^T$

特徴ベクトルは$[0,1]$で表現される関数である。その局面で存在する特徴が 1 で、存在しなければ 0 となる。従来の評価関数では係数ベクトルvは全て 0 であった。評価値は進行度を変数とした序盤と終盤の係数ベクトルと特徴ベクトルの内積の線形和として計算される。序盤と終盤で同じ特徴ベクトルを使用する事で、評価関数の計算速度を落とさずにすむ（ただし係数ベクトルの参照回数が 2 倍になる）。実際 node/sec でもほとんど差は現れなかった。目的関数は以下のように設計される。

$$J(v,w) = \sum_{i=0}^{N} \sum_{g=1}^{G} T\left[h(p_{i,g}) - h(p_{i,0}) \right] \to \min$$

サンプルする棋譜の数：N
その局面での合法手数：G
シグモイド関数：$T[\cdot]$
訓練集合：$p_{i,g}(g != 0)$
教師信号：$p_{i,g}(g = 0)$

$$h(p_{i,g}) = \begin{cases} l(p_{i,g})^T v & \alpha < Q \\ l(p_{i,g})^T w & \alpha \ge Q \end{cases}$$

進行度の閾値：Q

当初は最適化を行う際に評価値の計算に$f(p)$を使用していたが、勾配ベクトルが複雑になりすぎてうまく収束させる事ができなかった。そこで最適化する際は評価値の計算に$f(p)$ではなく進行度を使用せずに特徴ベクトルと係数ベクトルの内積を求める$h(p)$を計算し、定数Qを境に進行度がQより小さい時は序盤を、それ以外は終盤の係数ベクトルを最適化した。以下に勾配ベクトルを示す。

$$\Delta J(v,w) = \sum_{i=0}^{N} \sum_{g=1}^{G} \begin{bmatrix} l_1(p_{i,g}) - l_1(p_{i,0}) \\ \vdots \\ l_n(p_{i,g}) - l_n(p_{i,0}) \end{bmatrix} \times T[X](1 - T[X]) + M_1 + M_2$$

$$X = \frac{1}{K}\left(h(p_{i,g}) - h(p_{i,0}) \right)$$

拘束条件：M_1, M_2
スケーリング係数：K

求めた勾配ベクトルを勾配方向にそって値を更新する。計算時間のほとんどがこのΔJの計算に注がれることになる。

2．3　拘束条件

係数ベクトルが必要以上に大きくなる事を防ぐために目的関数に拘束条件を課す必要があるが、本研究では学習中に探索を行っていないせいか保木によって提案された手法は今回の場合有効に働かなかった。そこで序盤では駒割を重要視させるために駒割以外の値(Positional Value)に大きな拘束を課し、終盤になるにつれて Positional Value の拘束を緩めるという設計思想で新しい拘束条件M_1を提案する。

$Barrier = A \cdot \alpha + B;$
$Penalty = |PositionalValue| - Barrier;$
$if(Penalty > 0)$
　$for(i$；その局面の特徴ベクトル全て$)$
　　$if(sign(PotisionalValue) == sign(v_i))$
　　$M_1 = \tanh\left(\frac{Penalty}{C}\right) \times \left(\frac{v_i}{Barrier}\right);$
　　$\Delta J_i += M_1;$
　　係数：A, B, C　　進行度：α

この拘束条件は与えた局面の Positional Value がバリア値を超えたときにだけ与えられる。バリア値は進行度を変数とした一次関数で与えられ、局面が進むにつれて大きくなっていく。Positional Value とバリア値との差分が大きいほど強いペナルティが与えられる。またペナルティは係数の絶対値が小さくなる方向でしか与えられず、係数の値が大きいほど強いペナルティとなる。これにより、自然と序盤は駒割が尊重され、終盤は Positional Value が尊重されることになり、値の増減を抑える事ができる。また係数ベクトルを分離して学習する今回のような場合、ひとつの式で各々の拘束を与える事ができて都合がよい。また序盤と終盤の係数ベクトルを分けることでサンプル数の不足からくるオーバーフッティングが起こってしまう。終盤にしか現れないような特徴が序盤においてオーバーフッティングを起こす事を抑止するために、サンプル数の少ない特徴に対して大きなペナルティを課す拘束条件 M_2 を加えることにした。

$$M_2(v_i)=\frac{P}{S(v_i)}v_i$$

係数 : P

サンプル数 : $S(v_i)$

2．4　特徴ベクトル

今回の学習プログラムで考慮している特徴ベクトルは以下のとおり

- 駒割
- 持駒の価値
- 王とその他の駒の位置関係
- 隣接している駒の位置関係
- 香角飛馬竜王の動ける枡の数
- 角飛馬竜の相手玉方向の利きの数
- 相手玉周辺の利きの勝ち負け
- 香桂角飛馬竜が当てている駒の種類
- 王の段
- 王の自由度
- ピン駒の種類、方向、利きの勝ち負け
- パターンマッチングによって得られた特徴
- パターンが存在しなかった時の特徴

基本的には保木にならって特徴ベクトルを生成した。しかし保木の構成だと王とその他の駒の位置関係に関して、特に王が７段目より上にいる時は極端にサンプルが少なくなってしまう。そこで棚瀬が考案した縦と横を分離した絶対座標系で考える手法を適用した[4]。これにより特に入玉に関してははっきりと良い性能が発揮された。また保木の特徴ベクトルにはない相手玉周辺の利きの勝ち負けは本稿では非常に重要な特徴となっている。

パターンマッチングは棋譜中に頻出する三駒パターンを抽出して取得した。特徴の数は全部で 85000 個ほどあり、かなり膨大である。係数はその約２倍存在する事になるので、もっと少ない表現を再考する必要があると思われる。

0	0	0	0	0	0	0	0	0
-21	玉	-32	-137	-141	-83	-151	150	-17
277	0	-82	-96	-107	73	25	-16	86
-102	126	61	131	131	86	105	32	-39
-36	-7	-30	51	-10	72	38	9	-50
5	-8	-7	-5	2	-9	5	-28	-1
-27	-13	-26	-18	-30	-17	-26	-20	-16
-29	-88	-109	-65	-41	-90	-88	-74	-65
-1	-32	-106	-124	-105	-134	-126	-152	-6
0	0	0	0	0	0	0	0	0
177	玉	39	42	33	-128	-184	-118	-153
332	0	266	118	110	32	-90	-94	-240
169	330	267	143	123	71	30	48	-79
26	58	22	41	6	3	-11	-19	-42
-5	-13	1	-5	-2	0	-7	-35	-7
-5	-4	10	-5	-9	-5	-11	-31	-2
117	-37	15	47	30	-4	-49	-82	-53
-1	-123	-40	-11	9	-25	-44	-145	-129

図２.相手玉が８二にある時の味方の歩の位置に対する得点(上:序盤 下:終盤)

0	0	-6	-6	5	-1	0	0	0
-1	0	-9	-1	-2	-1	-1	玉	0
-1	-2	-1	-1	-2	1	0	0	0
2	-2	59	9	33	-43	70	23	90
-55	-26	34	-20	58	55	96	173	113
-32	55	-11	17	28	22	42	40	-29
-18	-10	-25	-18	-3	-6	-15	20	-50
-23	-14	-54	-29	0	-46	3	4	-27
-1	-17	-78	-74	-52	-28	-52	-59	-2
-534	-434	-423	-479	-307	171	0	-47	0
-213	-344	-257	-252	12	158	15	玉	-93
-261	-177	-216	-172	-63	124	0	0	0
-160	-151	-59	-84	28	116	171	100	-50
-152	-28	-15	-14	10	25	51	-26	-103
-88	-7	-2	-11	-9	-11	9	-35	-127
-57	-27	-16	-16	-23	-21	-31	-24	-137
-149	-29	-6	-27	-31	-45	-6	-12	-127
-177	-105	-106	-113	-118	-71	-67	-111	-184

図 1.相手玉が２二にある時の味方の銀の位置に対する得点(上:序盤 下:終盤)

２．５　実験環境

今回の実験に使用した棋譜は 28621 局で、そのうちプロの対局棋譜はおよそ 12000 局であり、残りは将棋クラブ 24 でレーティング 2300 以上の人々の対局データである。今回の実験では駒割の価値は固定して行っている(歩一枚の価値は 100 点)。また評価値を計算する際に、最大交換値を計算して加える事を行っている。最大交換値は YSS[5] などでも実装されており、実際 TACOS でも実装されているので加えることにした。

また保木による手法と違って今回の学習は探索を行っておらず全て棋譜から一手生成した局面から比較して学習を行っている。探索を行わなかった理由として時間的制約があったという理由が大きいが、探索無しでもどのくらい学習する事が可能かを調べて見たかったというのもある。探索を行っていない事もあって、3 万近い棋譜を使用しているにもかかわらず、学習にかかる時間は一時間にも満たない。

３　まとめ

３．１　実験結果

拘束条件 M_1 を課す事で値がうまく抑えられ探索中に乱暴な手が少なくなり強くなった。拘束条件 M_1 を課さなければ自己対戦における勝率が 3 割近く低下する事から、重要な式であると言える。次に棋譜から学習された係数ベクトルの一部を図１～図４に示す。序盤と終盤ではかなり傾向が違っていて興味深い。図１や図２で玉の前方が 0 なのは学習の仕様上、王手局面では評価関数を学習しないためである。図１において序盤のテーブルの 1～3 段目の数値や図３の右側の値が非常に小さいのはオーバーフッティング対策の拘束条件 M_2 のせいである。図２より序盤は２二の地点に歩を打つ桂馬を当てに行く手が高くなっているが、終盤では大きな減点となっている。

−1	−6	−14	9	−14	6	−1	1	−1
−1	−33	32	−18	−11	6	8	1	−1
−43	−17	−7	9	−64	−3	−41	−15	0
0	43	27	−12	−29	−90	−115	0	0
王	43	44	4	−23	−56	−158	0	0
−193	−115	−31	−20	−99	−132	−199	−237	−259
−111	−11	−3	−31	−41	−76	−144	−160	−282
−20	23	−12	−16	−52	−78	−122	−213	−284
109	50	26	−25	−44	−89	−139	−212	−280
王	114	25	−29	−55	−68	−152	−242	−257

図３.自玉が９九にある時の味方の金の位置に対する得点(上:序盤 下:終盤)。

−24	3	−54	−43	−11	−21	2	14	−53
59	−104	−108	−127	−72	−56	−92	−61	117
−26	−51	−206	−137	−78	48	−6	−103	−81
−30	−82	−25	−75	39	5	0	玉	−69
−49	−70	−136	27	−41	19	0	0	−96
22	22	10	7	−23	−14	−5	9	−37
31	20	12	−20	−25	−63	−178	−65	−113
31	−2	−9	−35	−88	−152	−484	−611	−447
16	0	4	−19	−92	−198	−420	玉	−326
−12	−162	−162	−162	−162	−162	−162	−162	−162

図４.自玉が２八にある時の各枡に対する利き負けの得点 (上:序盤 下:終盤)

図４より序盤は突破を図る事が非常に効果的であるが、終盤になると玉から離れた位置に利きを集めるのは意味がなくなってくるという結果をみてとれる。

また全体的に序盤の数値は終盤と比べて小さくなっていて、意図した結果となっている事がわかる。総じて序盤では位取りや突破などが重要で終盤は玉近辺のみが重要であるというだいたい人間の直感と一致するよいデータを学習する事ができたといえるだろう。

本稿では提案した進行度を考慮した評価関数を TACOS へ実装する事が間に合わなかった。代わりに係数ベクトルを分けずに学習した評価関数(拘束条件などの項目は全て提案した手法を使った)を実装した TACOS と従来 TACOS を途中局面から先後入れ替えで322戦行った。対戦成績は177勝144敗となり自己対戦の結果から統計的に強くなったといえる。

３．２　考察

今回の学習では探索を行っていないので、学習の収束精度が危ぶまれたが、自己対戦の結果を見ると実際には従来を上回る性能があると言える。もちろん探索を行う事でもっと性能が向上する可能性も十分にある。また今回提案した拘束条件 M_1 はかなり感触がよく、評価関数の性能向上に大きく貢献した。保木の提案した拘束条件と比べても自己対戦で大きく勝ち越すことが出来たので今回の実験環境では優位さははっきりでたと言ってよいだろう。提案した評価関数は途中局面から自己対戦という形式での性能評価は難しいが(序盤がほとんどあらわれないため、ほとんど中終盤のみの評価値になってしまう)、単純な棋譜の一致率は従来の方法と比べて向上している。

３．３　今後の課題

今後は学習に探索を取り入れて、今回提案した手法がどういった振る舞いをするのか検討し、保木の手法と性能対比を行う事が一番の課題である。また考慮すべき特徴ベクトルも再考する必要があると思われる。特に囲いを表現するためにはどうしても複数駒に関する特徴を組み込む必要性があるが、ほとんどの複数駒の特徴は棋譜中に十分なサンプルが現れない。そこでパターンマッチングを行い頻出するパターンのみを抽出して組み込む事を現在は課題として行っている。また現状の評価関数の一番の弱点は終盤であり、特に王の安全度をうまく理解できていない。従来の TACOS で使用

していた王の安全度を学習した評価関数でも使用すると勝率は 1 割以上も向上した。よって王の安全度に関する特徴をうまく設計する事ができれば勝率はさらに向上すると考えられる。

参考文献

[1] 橋本. 将棋アルゴリズム TACOS のアルゴリズム, コンピュータ将棋の進歩 5, pp. 33-66, 2005

[2] 保木. 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, Nov. 2006.

[3] 鶴岡. 将棋プログラム「激指」, コンピュータ将棋の進歩 4, pp. 1-17, 2003

[4] 棚瀬. 棚瀬のページ. http://tanase.yasushi.googlepages.com/2007csa

[5] 山下. コンピュータ将棋 2 以降の改良点, コンピュータ将棋の進歩 5, pp. 1-32, 2005

[6] 金子. 兄弟節点の比較に基づく評価関数の調整. 第 12 回ゲームプログラミングワークショップ, pp. 9-16, Nov. 2007.