(GPW07 の後に誤字や図の位置等を修正)

兄弟節点の比較に基づく評価関数の調整

金　子　知　適 †

本稿では兄弟節点の比較に基づく評価関数の調整方法を提案する．棋譜を教師例とした評価関数の調整は古くから行われており，最近では兄弟節点の比較に基づくものが注目されている．特に，探索により最善応手手順を求めることと，求めた手順を元に重みを調整することを繰り返すことで良い評価関数が得られることが知られているが，計算時間がかかるという問題点がある．その計算時間は最善応手手順を求める時間がほとんどであるため，最善応手手順を効率良く求めることと，数値計算において求めた最善応手手順を有効に活用することで効率の改善を図り，Bradley-Terry モデルに基づいた指し手の並び替え及び絞り込みとロジスティック回帰を利用した重みの調整を行った．将棋を題材にした実験において，提案手法が有効であることを確認した．また，駒の重みを固定することで，さらに効率が向上することが分かった．

Learning Evaluation Functions by Comparison of Sibling Nodes

TOMOYUKI KANEKO †

This paper presents a framework for learning of evaluation functions. It compares the selected move and other moves in a database of game records, and adjusts weights by using logistic regression. The goal of the adjustment is that the evaluation function assigns higher value for the move selected by human player than the other legal moves. In the naive implementation of this framework, we needs to compute the PV nodes for each legal move in each position in the database, and its computational cost is very large. In order to remedy this problem, we selectively used the effective moves instead of using all the legal moves. The effectiveness of moves was automatically determined by the method based on Bradley-Terry model. The experiments on Shogi showed the efficiency and effectiveness of our method.

1. はじめに

ゲームプログラミングにおいて評価関数を自動的に求めることは重要な研究課題であり広く研究されてきた．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，特徴と呼ばれる．将棋の場合，駒の損得や玉の危険度などが知られている[12]．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．一方で，将棋の様々な評価項目について自動的な調整を行うことは長らく困難と言われていた．

2006 年に発表された保木[15] の手法では，棋譜に指された手を実際に探索が指す方向に重みを調整することで 1 万を越える重みの調整に成功したことから，実用的な評価関数の調整方法についての注目が集まっている．保木の手法はプログラムが実際にコンピュータ将棋選手権で優勝していることから有用性は疑う余地がないものの，計算時間が長くかかる点に改善の余地がある．☆そこで本研究では収束の速さを目標として，(1) ロジスティック回帰にもとづく重みの調整，(2) 指し手の順位づけを利用した訓練例の選択を組み込んだ学習の枠組を提案し，実際に効果的であることを将棋を題材にした実験で示す．

2. 関連研究

オセロでは多数のパターンの重みを最小二乗法で調整した評価関数が早くからトップレベルの強さを実現している[4]．これは，局面に対し評価関数がとるべき

† 東京大学大学院総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo
kaneko@graco.c.u-tokyo.ac.jp

☆ 2006 年のゲームプログラミングワークショップの質疑では，3ヶ月程度かけたとのこと．

表 1　アルゴリズムの比較

	比較	最善応手手順	計算手法
TD	親子節点	なし	最急降下法
TDLEAF[3]	親子節点	最善応手手順	最急降下法
駒の関係[14]	兄弟節点	なし	共役勾配法
保木[15]	兄弟節点	最善応手手順	最急降下法
本研究	兄弟節点	最善応手手順	ニュートン法

評価値の教師例を与えて調整するものである．一方，将棋やチェスのトップレベルのプログラムでは手で調整した評価関数が使われていた．その理由の一つは評価関数がとるべき評価値の教師例を用意することが難しいことと考えられる．将棋の神であれば，任意の局面に対して勝ち，負け，引き分けのラベルをつけることができる．しかし，評価関数を作成する場合には同じ勝ちの局面でも，プログラムが勝ちやすい局面と難しい局面を区別する必要があり，そのための適切な指標として評価の定まった方法は存在しない．一方，オセロでは終盤の評価関数に対しては，探索で読み切った値を，中盤の局面に対しては終盤の評価関数を用いて探索した評価値が，学習のための適切なラベルとして機能することが知られている[4] 点が将棋やチェスとは異なる．

そのため，別の考え方として，局面の比較に基づく手法が比較的早くから着目されてきた．その一つは，人間が選んだ指し手の局面は，選ばなかった手の局面より評価が高いべきであるという考え方に基づくものである．この考え方は，少なくともプログラムが人間に近付くためには妥当であると考えられる．

表 1 に，局面の比較に基づく学習手法を掲載する．TD 法は，手を指す前と指した後の局面 (本稿では親子と呼ぶ) を比較して重みを調整する手法である．改良版の TDLEAF[3] では，親子局面を直接比較する代わりに，それぞれで探索を行い最善応手手順後の局面を求め，それらを比較する．

将棋では，親子の比較より兄弟の比較の方が優れているという報告があり[14]，金子らの駒の関係[14] と保木[15] の手法では，棋譜で指された手の後の局面と他の合法手の後の局面 (兄弟) を比較し，棋譜で指された手の方が評価値が高くなるように重みを調整する．

計算手法の観点では，駒の関係以外は上記の研究では最急降下法を用いており収束が遅いという問題点がある．一方，駒の関係では収束は速いものの，駒組みのみの学習に留まり駒の価値を含めた学習は行われていない．

そこで本研究では，最善応手手順を考慮した上で兄弟節点の比較を行い，ロジスティック回帰により重みを調整する．ロジスティック回帰には，ニュートン法に基づく高速なアルゴリズムが存在する．

3. 兄弟節点の比較に基づく重みの調整

提案手法では，棋譜で指された手が探索により選択されるように評価関数を調整する．すなわち，棋譜で指された手の評価値が，他の合法手全てよりも高くなることを目的とする (もちろん，それは実現不可能なのでなるべく近付けることが目的となるが，詳しくは後で議論する)．一般にゲーム木探索では min-max 法に基づいて選択された局面 (そこに至る手順を最善応手手順と呼ぶ) の評価値に基づいてルートの局面の評価値が定まる．そこで，指された手の評価値を他の手の評価値より高くするためには，それぞれの手の最善応手手順後の局面を取り出し，前者の局面の評価値が (手番にとって) 高くなるように調整すれば良い．実際には，最善応手手順は用いている評価関数に依存し，評価関数を調整すると最善応手手順が変化する．そのため，この調整がうまく動作するためには，評価関数の調整の範囲では最善応手手順が変化しないか，変化したとしても実用的な解に収束することを仮定する必要がある．これは一見大胆な仮定であるが，実際的にはうまく動作することが保木[15] により示されている．

具体的な手順を，図 1 に掲載する．大きく分けて，最善応手手順を求める部分と重みを調整する部分の二つからなり，全体を収束するまで繰り返す．何も工夫をしない場合，計算時間は，最善応手手順を求められることにほとんど費される．そこで，一度得た最善応手手順を効率良く利用することと，不要な探索を削減することが必要となる．

最善応手手順を求める探索は，保木[15] の場合同様に，全幅で一手を読みその後静止探索を行うこととした．但し，静止探索を長手数行うことは計算時間がかかるため，4 手で打ち切っている．多くの将棋プログラムでも静止探索の深さに制限を設けているため，この変更は妥当であると考えている．また，最善手の評価値と掛け離れた値は学習への影響が少ないため，最善手でない手を探索する際には，保木の手法にならって最善手の評価値の前後に歩 3 枚分程度のウィンドウを用いる．(1) の (a) で合法手の一部を捨てる理由は，訓練例を絞り込むことで計算時間を削減する工夫であり，続く節で説明する．

続いて，重みをロジスティック回帰で調整する．具体的な計算については続く節で説明する．保木の手法では重みは定数値だけ徐々に変化するが，ロジスティック回帰では重みは必要に応じて大きく動くため収束が

(GPW07 の後に誤字や図の位置等を修正)

(1) 棋譜データベース中の各局面について，以下の計算を行う
  (a) 合法手を全て作成する．(*) そのうちの一部を間引きする
  (b) 棋譜で人間により指された手の後の局面と，他の手の後の局面のペア $\langle s_i, t_i \rangle$ を作る
  (c) 各ペアのそれぞれの局面に対して探索を行い，最善応手手順を求め，ディスクに格納する．
(2) 格納した最善応手手順を利用して，棋譜で人間により指された手の評価値が他より高くなるように，以下の手順で重みを更新する
  (a) 各ペアについて，それぞれの局面の最善応手手順後の局面 $\langle \bar{s}_i, \bar{t}_i \rangle$ を作成し，各局面での特徴ベクトル $\langle x_{s,i}, x_{t,i} \rangle$ を求める．
  (b) ペアの差分 $x_i = x_{\bar{s},i} - x_{\bar{t},i}$ を説明変数の特徴ベクトルとする．また，従属変数 y_i は先手番なら 1, 後手番なら 0 とする．
  (c) x, y を用いてロジスティック回帰で w を求める．
(3) (1) に戻り，新たな重みで探索し，その結果を元に w を調整し直すことを繰り返す．

図 1 提案アルゴリズム

速いことが期待できる．以下，ロジスティック回帰を説明した後，訓練例の絞り込みについて説明する．

3.1 ロジスティック回帰

特徴ベクトルを x，重みベクトルを w とすると，次式は，wx が $-\infty$ から ∞ を動く時に 0 から 1 まで単調に増加しロジスティック曲線を描く．

$$P(x) = \frac{1}{1+e^{-wx}}.$$

ロジスティック回帰では，特徴ベクトル x_i に対する $y_i \in \{0,1\}$ が与えられた場合に，尤度

$$\Pi_i \left(y_i P(x_i) + (1-y_i)(1-P(x_i)) \right)$$

を最大にする w を求める手法である．得られた w は，$wx > 0$ なら $y = 1$，$wx < 0$ なら $y = 0$ と予測する分類器として使われ，wx の絶対値は予想の確からしさを表す．本稿での計算では，wx は二つの局面の評価値の差，$P(x)$ を比較の確からしさと捉えて重みを調整する．得られた w はそのまま評価関数として用いることができる．

単純な線形回帰と比較するとロジスティック回帰は $\{0,1\}$ の予測に向いている．また，値が極端である部分があまり影響しないという特性がある．すなわち今回の目的である指し手の比較のモデル化に適している．保木[15] が指摘しているように，棋譜の中には単純な探索では理解できない指し手もあれば，ポカのような指し手も存在する．値の調整の際にはそれらの影響を最小限に留めることが望ましい．また，既に最善手より低い点数がついている手の評価をより下げることや，順位が変わらないにもかかわらず最善手より遥に高い点数の手の評価を下げようとすることも無駄である☆．

ロジスティック回帰には様々な計算手法が存在するが，このような学習に用いる場合には，訓練例の数が多いため係数行列がメモリにもハードディスクにも収まらないことに注意が必要である．そのため，係数行列への任意のアクセスを仮定する手法は適さない．今回の実装では最善応手手順のみディスクに収め，係数行列は必要に応じて順次生成している．また，特徴の種類が多くなると (準) ニュートン法に必要な勾配行列もメモリに収まらない．このような場合には，内部で共役勾配法を用いる実装が良い．過学習を避けるための l_1-あるいは l_2-regularization に対応した手法もあるが[9][7]，今回は実装の容易さから IRLS[11] を用いた．他に，[8][6][1] などの手法もある程度大規模なデータに対して用いられている．

IRLS では内部で共役勾配法[2] を用いる．その計算では，X を訓練例を行ベクトルとする行列，D を何らかの対角行列，p を何らかのベクトルとした時に積 $q = X^t D X p$ を計算する必要が生じる。この際に，$X^t D X$ がメモリに収まらないため，X のベクトルを順に生成しながら q を順次計算する必要がある．

3.2 レーティングに基づく指し手の絞り込み

将棋の合法手は終盤では 200 手を超えることもあるが，それらのほとんどは人間のプレイヤに取っては考慮の対象とならない指し手である．また，コンピュータ将棋プログラムの場合にも，ルートに近いところ以外では前向き枝刈りを行っているプログラムが多い．一方で，提案する評価関数の学習においては，考慮の対象とする候補手の数が計算時間に大きく影響するために，候補手の数を減らすことが望ましい．そこで，評価関数以外の方法で指し手の順位づけを行い，人間があまり指さないと判断された指し手は間引きすることを行う．囲碁においてはレーティングに基づく指し手の順位づけが成功を収めている[5] ため，それに倣って棋譜を教師として学習を行った．

ELO レーティングは，プレイヤの強さを数値化する

☆ 評価関数の学習において近いことは古くから考えられていたようである[10]

表 2　駒の初期値 (1)

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

際に用いられる方法の一つで，各プレイヤ i が γ_i という正の値を持ち，プレイヤ i が j に勝つ確率を

$$\frac{\gamma_i}{\gamma_i + \gamma_j}$$

とモデル化する．いわゆるレーティング r_i は $400 \log_{10} \gamma_i$ と変換したものである．

Bradley-Terry モデルは複数人の試合を扱える一般的なモデルで，1 から n までのプレイヤの中で $i(1 \leq i \leq n)$ が勝つ確率を

$$\frac{\gamma_i}{\sum_{j=1}^{n} \gamma_j}$$

とモデル化する．チームを組むプレイヤの γ_i を掛け合わせた積が一人のプレイヤであるとして比較することでチーム戦も扱うことができる．例えば、プレイヤ 1,2,3 と 4,5 と 6 の 3 チームで最初のチームが勝つ確率は，

$$\frac{\gamma_1\gamma_2\gamma_3}{\gamma_1\gamma_2\gamma_3 + \gamma_4\gamma_5 + \gamma_6}$$

とモデル化される．

将棋の指し手には,「駒得」や「王手」,「飛車の前に歩を打つ」など様々な特徴が考えられる．それらの特徴の強さ (例えば，王手の指し手を他の手より優先する度合) に点数をつけ，指し手は特徴のチームであると考えると，ある局面で合法手のうちのどの手が指されるかを上記のモデルを当てはめて考えることができる．そこで，棋譜で指された指し手のレーティングが高くなるように，各特徴のレーティングを定めた．詳しくは文献[5] を参照されたい．

図 1 の手順中で合法手を作成した後に，あらかじめ求めておいた指し手の特徴毎のレーティングを用いて，指されそうな順に合法手を並べ替える．並べ替え後，順位が後ろの方の指し手は，評価関数の学習への影響が少ないと考えて，何手かに一手のみ訓練例に加える．

4. 実 験 結 果

4.1 駒割の学習

まずは予備的な実験として，駒の価値のみの評価関数を作成し，その重みを学習させた．棋譜は将棋クラブ 24[13] の 40 試合を用いた．訓練例は 4 万程度であった．3.2 節で説明した訓練例の絞り込みは行っていない．

初期値として表 2 の価値を与えたところ，図 2 のように妥当と思われる値に落ち着いた．横軸が反復回数，

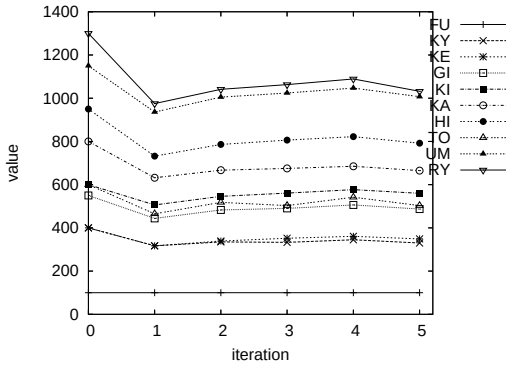

図 2　反復数と駒の価値の変化 (初期値は表 2)

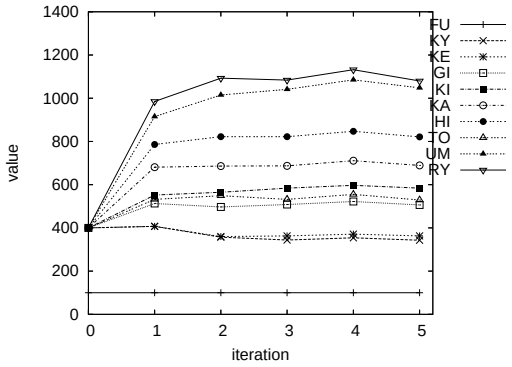

図 3　反復数と駒の価値の変化 (初期値 400, 歩のみ 100)

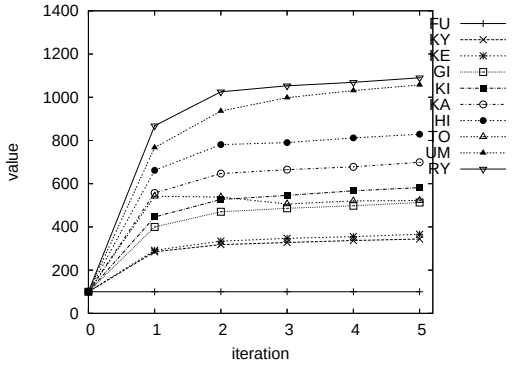

図 4　反復数と駒の価値の変化 (初期値 100)

縦軸が駒の重みで歩を 100 点に正規化した値で示している．また，初期値を変更し，歩を 100 点他を 400 点にした場合と，全てを 100 点とした場合の二種類について実験を行ったところ，それぞれ図 3, 4 に示すようにほぼ同じ値となった．全ての場合で探索し直した回数は 5 回，共役勾配法を起動した回数で数えても 10 回程度で計算は収束した．そこで，これ以降の実験で

(GPW07 の後に誤字や図の位置等を修正)

表 3　直接の取り返しによる駒の損得関する指し手の特徴

特徴	数	値の範囲	詳細
駒割	13	-821 – 581	(-∞ ≤) -821 (-1050 ≤) -634 (-850 ≤) ... (-50 ≤) -4.19 (51 ≤) 175 ... (1051 ≤) 581
影利き	2	97.4 – 213	1 つ 97.4，2 つ 213

表 4　利きに関する指し手の特徴

特徴	数	値の範囲	詳細
取り返し	2	321 – 482	取り返し 321，連続取り返し 482
連続駒取り	1	74	
ピンされた駒への利き	2	12 – 30	自分の駒の駒 30，相手の駒 12.2
王手	4	389 – 746	直接 389，直接 (かつ長い利きの筋を開ける) 416，開き王手 547，両王手 746
取ると一手詰	1	99.6	
パスすると一手詰	1	172	
送りの手筋	1	237	
移動元の利きの数	9	-81.2 – 427	なし -17.3, 敵 1 427, 敵 2 345，自分 1 -63.6，自分 1 敵 1 237, ... 自分 2 敵 2 188
移動元への敵の利きの種類	224	-324 – 441	角に歩 441，飛車に桂馬 331，角に桂馬 308 ... 歩に竜 -291，と金に金 -324
移動先に自分の利き	1	68.3	
移動先の長い利き	3	-64.8 – 75.9	敵のみ 75.9，自分のみ -64.8，両方 47.6
移動元の長い利き	16	-131 – 173	

表 5　位置に関する指し手の特徴

特徴	数	値の範囲	詳細
空成り	1	-163	
振り飛車で飛車を合わせる	1	104	
歩で止められる香打ち	1	-21.5	
敵玉との相対 X 座標	189	-496 – 293	差 0 に飛車打ち 293，差 0 に竜が移動 175 ... 差 8 に金打ち -496
敵玉との相対 Y 座標	357	-446 – 192	
自玉との相対 X 座標	189	-178 – 89.4	
自玉との相対 Y 座標	357	-566 – 138	
X 座標	105	-312 – 141	5 筋の香移動 141 ... 1(9) 筋に桂馬移動 -312
Y 座標	189	-951 – 358	
前のマスの状態	3780	-516 – 767	角の前に歩 767 ... 成銀の前に角 -516
斜め前のマスの状態	3780	-607 – 756	
横のマスの状態	3780	-605 – 721	
下のマスの状態	3780	-535 – 535	成香の下に桂馬 535 ... 飛車の下に敵の成香 -535
斜め下のマスの状態	3780	-565 – 514	
桂馬の位置のマスの状態	3780	-557 – 671	
長い利きの駒の先 (前)	2615	-542 – 518	
長い利きの駒の先 (その他)	2092	-692 – 565	
駒を挟んだ長い利き (前)	145	-136 – 96.5	
駒を挟んだ長い利き (その他)	116	-199 – 114	角が敵の竜を睨む 114 ... 飛車が敵のと金を睨む -199
二手前とのバイグラム X 座標	3125	-534 – 375	
二手前とのバイグラム Y 座標	3125	-467 – 433	
一手前とのバイグラム X 座標	3125	-455 – 429	
一手前とのバイグラム Y 座標	3125	-295 – 514	

は，駒割の初期値は全て表 2 のものを用いた．また，他の特徴の初期値は 0 とした．

4.2　指し手の順位づけ

続いて 3.2 節で説明した指し手を順位づけする関数を棋譜に基づいて作成した．4 万棋譜を用いて，計算時間は一日程度であった (計算機は Xeon 5300 を搭載した Mac を利用し 8 並列で動かした)．

指し手の特徴としては，直接の取り返しの駒得/駒損に基づくもの，王手や戦術的な利きに関するもの，位置を考慮したものの 3 つのカテゴリを用意した．具体的な特徴と得られたレーティングの一部をそれぞれ表 3，表 4，表 5 に掲載する．値は $400 \log_{10}$ の変換をした後のものである．「駒割」では、駒得の指し手ほどレーティングが高い，「王手」や「取り返し」は重視される，など妥当な点数がついていると考えられる．表中で，「移動元への利きの数」は，駒が逃げるかに関係する特徴でプレイヤ毎の利きの数を数え (最大 2 で打ち切り)，3×3 で 9 通りに分類した．位置に関しては，玉との相対座標や，絶対座標においては，駒の種類と移動かどうかを分類した．「マスの状態」は，自分の駒

表 6　指し手の並べ替えに関する性能

特徴	平均順位	相対所要時間
駒割 (表 3)	25.5	1.00
駒割+利き (表 3+表 4)	14.4	5.64
駒割+利き+位置 (表 3+表 4+表 5)	5.03	11.3

表 7　評価関数の種類

名前	特徴の数	中身
piece	16	駒割
tactical	249	piece+駒への利きの数, 種類
position	1305	tactical+絶対位置，相対位置
tactical2	234	tactical で駒の重みを固定
position2	1290	position で駒の重みを固定
pattern2	3978	position2+各駒の 10 近傍の種類

の種類，隣の { 駒の種類と所有者，空白，盤外 } かどうかと，盤外以外の場合にプレイヤ毎の利きの数を数え (最大 2 で打ち切り) て分類している．「長い利きの駒の先」は空白であるかぎり先を調べて行き止まった場所を調べる特徴で，飛角とその成駒及び香のみに用いる．駒を挟んだ長い利きは，田楽刺しなどを判断するための特徴である．バイグラムは，移動先が敵玉の玉の 25 近傍に連続する場合を判定する特徴である．

得られたレーティングの性能を測るために，訓練に使ったものとは別の 2,000 棋譜について，各局面でレーティング順に並べ替えた場合に指された手が何手目になるかを調べた．表 6 の中央の列に結果を掲載する．駒割のみではあまり役に立たず，位置に関する特徴を加えると約 5 手目と性能が急激に良くなることが分かる．また，最適化したコードではないので参考程度の価値であるが，駒割の判定を 1 とした時の他の特徴の相対的な所要時間を示した．特徴が増えるほど並べ替えに時間がかかり，正確さと速度のトレードオフがあることが分かる．実際に，この並べ替えは将棋プログラムの強さの改善には有効だったものの，図 1 の提案手法の (1)(c) で探索を行う際の指し手の並べ替えには，位置を使うものは有効ではなく駒割のみを用いる方が良かった．これは探索ウィンドウが狭いことと，残り深さが浅いためと考えられる．

以降の実験では，全ての特徴と得られた値を用いて図 1 の提案手法の (1)(a) における指し手の絞り込みを行った．具体的には，8 手目までは全て用い，16 手までは $\frac{1}{2}$，32 手までは $\frac{1}{4}$，64 手までは $\frac{1}{8}$，それ以降は $\frac{1}{16}$ を用いた．それぞれの範囲で何手目を選ぶかは局面毎に機械的に切替えた．これにより安定して約 3 倍程度の効率化を実現した．具体的には後で議論する．

4.3　評価関数の調整とその評価

続いて，6 種類の評価関数を作成し，重みの調整を行った．それぞれの評価関数の特徴の数と内容を表 7 に示す．まずは，前半の 3 種類 (piece, tactical, position) について説明する．

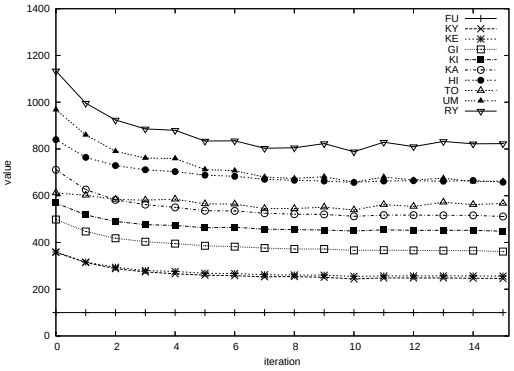

図 5　反復数と駒の価値の変化 (position)

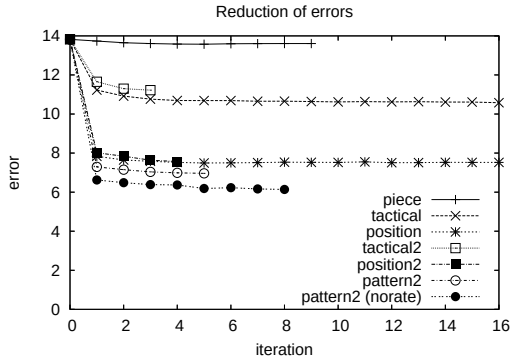

図 6　反復数と誤差の減少

400 局の棋譜を用い，訓練例の数は 40 万強であった．初めに，図 5 は，position における駒の重みの変化である．l_2 正規化をしていることもあり，特徴を増やしても駒の重みが急激に乱れることがないことを確認することができる．他の評価関数でもほぼ同様のグラフとなっている．

続いて，評価関数の性能を測るために，探索で選ばれた最善手が棋譜で指された指し手とどの程度一致するかを測定した．局面は，訓練に用いたものとは別の 100 棋譜中の 4 手毎の局面を利用した．指標としては，文献[15] を参考に以下の式を用いた．

$$\frac{1}{\text{局面数}} \sum_{\text{合法手}} T\left(-(\xi(\text{指された手}) - \xi(\text{他の手}))\right)$$

ξ は最善応手手順後の局面の評価値，T(x) はシグモイド関数 $\frac{1}{1+exp(-x \times 3/100)}$ で 100 は歩の点数である．この指標は，評価値の差が十分ある場合は 0 または 1 となり，各局面で最善手より評価値が高い指し手の数を数えることに相当する．一方，評価値が近い指し手に

(GPW07 の後に誤字や図の位置等を修正)

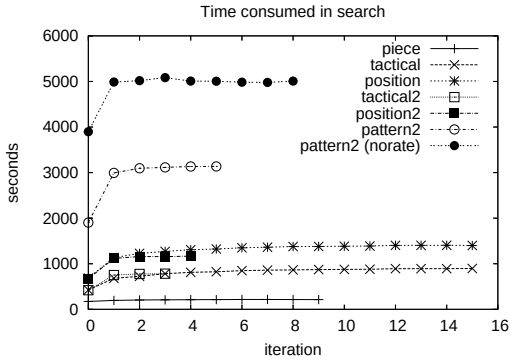

図 7　反復数と最善応手手順の計算時間

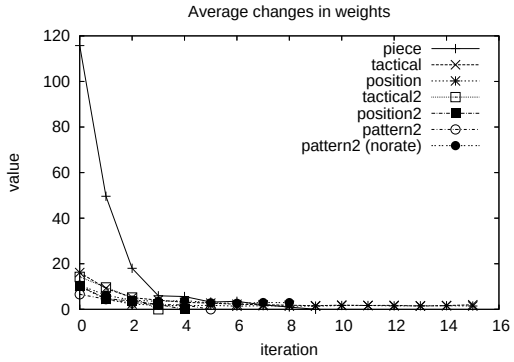

図 10　反復数と重み平均更新量

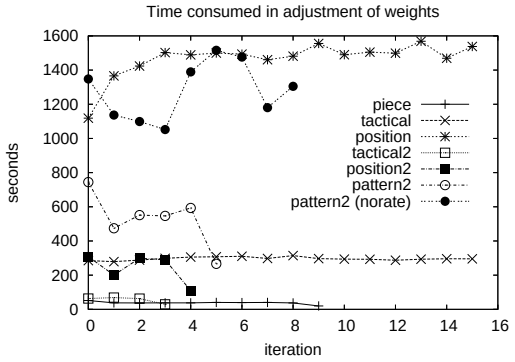

図 8　反復数と重みの調整の計算時間

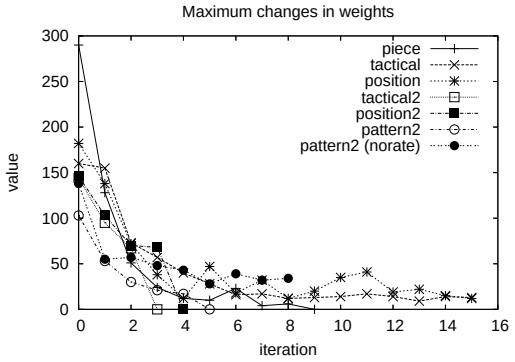

図 9　反復数と重み最大更新量

ついては 0.5 近辺となるため，最善手より評価が低くても値が近いものがあれば評価は下がる．この指標 (以後，簡単に誤差と呼ぶ) について，各反復毎の値を図 6 に示す．横軸が反復回数，縦軸が誤差である．piece, tactical, position のどの評価関数でも反復の度に減少しているが，値は数回の反復で落ち着いている．また，特徴の数が多い方が，少ない誤差を実現している．

続いて，計算時間について考察する．図 7 に，最善応手手順を求めるための計算時間を各反復毎に示す．横軸が反復回数，縦軸が計算時間である．計算時間は秒数 (cpu 時間) で測定した．なお，実際には 8 並列で動かしているため所要時間はもっと少ない．評価関数の特徴の種類が多い方が探索に要する時間は長くなる．また，反復が進むほど，計算時間が増えている．これは駒割以外の初期値が 0 なので初めは局面の差がつきにくいが，反復が進むほど局面に細かい評価値がつくためである．図 8 にロジスティック回帰に要した時間を各反復毎に示す．計算時間は，最善応手手順の場合同様に秒数 (cpu 時間) である．この時間は，共役勾配法が収束に要する時間に大きく影響される．一般的には，特徴の数が増えるほど時間が長くかかり，piece, tactical, position の比較でもそのようになっている．

反復毎の重みの変化量について，最大値と平均値をそれぞれ図 9 と図 10 に示す．最大値については多少ぶれがあるものの，両者共，反復が進む毎に 0 に近付いている．平均値と最大値の差は，重み毎に動かすべき幅が特徴毎に異なることを示している．

4.4　駒の評価値の固定

駒の評価値は初期値でほぼ妥当であると考えられるため，これを固定にして他の特徴のみを調整する実験を行った．tactical と position について駒の価値を固定にした評価関数が，それぞれ tactical2 と position2 である．それらの調整は数回で収束した．誤差を図 6 で比較すると，tactical と tactical2 は少し差が見えるものの position と position2 は差がなく，駒の評価値を固定にしても評価関数の正確さに影響がないことが示唆される．一方，計算時間については，図 7 の最善応手手順を求める部分は差がないものの，図 8 に示したロジスティック回帰の計算は圧倒的に速くなっている．また全体としても少ない反復回数で終了するため計算

時間については大きなメリットがある．

以上のように，駒の評価値を固定にすれば特徴の数を増やしても現実的な時間で計算が終わることが分かったために，特徴の数を増やした新たな評価関数 (pattern2) の実験を行った．表 7 に記載したように約 4,000 の特徴を持つ．その結果，図 6 の白丸のように，今迄に説明したどの評価関数よりも正確となった．

4.5 指し手の絞り込みの影響

指し手の絞り込みによる効率の向上と得られる評価関数の性能の関係を調べるために，指し手の絞り込みを行わない調整を行った．対象としては，今迄でもっとも正確だった pattern2 の評価関数を用いた．但し，実験時間の都合で棋譜の数は 200 にした (今迄は 400)．訓練例の数は約 65 万であった．400 棋譜の場合を外挿すると約 130 万程度になり，一方で指し手の絞り込みを行った場合が 40 万強であるから，指し手の絞り込は訓練例の数を約 1/3 に減らしていると考えられる．

この実験 (pattern2 (norate)) の計算時間を，図 7 及び図 8 の pattern2 と比較すると 1.5 倍以上の開きがあり，訓練例の数を揃えると 4.5 倍以上の時間がかかることが想像される．このことから，指し手の絞り込みは重要であると言える．しかし，図 6 を見ると分かるように，性能としては絞り込みを行わない方が向上している．従って，絞り込みを行うことで計算時間を節約して特徴を増やす方法と，絞り込みを行わない方法とどちらが優れているかは今後の研究が必要である．

5. おわりに

本稿では，兄弟節点の比較に基づく評価関数の調整方法を提案し，将棋における実験で効果を示した．ロジスティック回帰の採用，Bradley-Terry モデルに基づく指し手の順位づけを利用した訓練例の選択，駒の重みの固定の 3 つの組み合わせにより，従来手法と比較して短い時間での重みの調整を実現した．ロジスティック回帰で動く重みは特徴毎に幅があり，勾配に従って固定値を動かす場合と比較して素早い収束を実現している．また，駒の重みを固定することで，計算時間が極めて短縮される一方，学習で得られた評価関数の性能は変わらないことが実験的に示された．指し手の順位づけを利用した訓練例の選択では，計算時間を $\frac{1}{3}$ 程度にすることに成功したが，学習で得られた評価関数の性能に多少影響した．この影響の評価及び，改善については更なる研究が必要である．

参考文献

1) G. Andrew and J. Gao. Scalable training of L1-regularized log-linear models. In Z. Ghahramani ed., *Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007)*, pp. 33–40. Omnipress, 2007.
2) 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.
3) J. Baxter, A. Tridgell, and L. Weaver. Learning to play chess using temporal-differences. *Machine Learning*, 40(3):242–263, Sept. 2000.
4) M. Buro. Improving heuristic mini-max search by supervised learning. *Artificial Intelligence*, 134(1–2):85–99, Jan. 2002.
5) R. Coulom. Computing elo ratings of move patterns in the game of go. In *Computer Games Workshop*, Amsterdam, 2007.
6) A. Globerson, T. Koo, X. Carreras, and M. Collins. Exponentiated gradient algorithms for log-linear structured prediction. In Z. Ghahramani ed., *Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007)*, pp. 305–312. Omnipress, 2007.
7) K. Koh, S.-J. Kim, and S. Boyd. A method for large-scale l_1-regularized logistic regression. In *22nd National Conference on Artificial Intelligence*, pp. 565–571, 2007.
8) S.-I. Lee, H. Lee, P. Abbeel, and A. Y. Ng. Efficient l1 regularized logistic regression. In *AAAI*. AAAI Press, 2006.
9) C.-J. Lin, R.C. Weng, and S.S. Keerthi. Trust region newton methods for large-scale logistic regression. In *Proceedings of the 24th international conference on Machine learning*, Vol. 227, pp. 561–568, 2007.
10) T. Marsland. Evaluation function factors. *ICCA Journal*, 8(2):47–57, 1985.
11) A. M. Paul Komarek. Fast robust logistic regression for large sparse datasets with binary outputs. In *Artificial Intelligence and Statistics*, 2003.
12) 鶴岡. 将棋. 情報処理, 44(9):900–904, 2003. 特集ゲーム情報学.
13) 久米. 将棋倶楽部 24 万局集. ナイタイ出版, 2002.
14) 金子, 田中, 山口, 川合. 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, Nov. 2003.
15) 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.