情報処理学会論文誌 Vol. 51 No. 12 2141–2148 (Dec. 2010)

創立 50 周年記念論文

将棋の棋譜を利用した大規模な評価関数の学習

金 子 知 適[†1] 山 口 和 紀[†1]

本稿では兄弟節点の比較に基づく評価関数の調整方法について大規模な実験を行い議論する．将棋では近年，棋譜を教師とした評価関数の調整が注目を集めている．現在有望とされる手法では，最善応手手順の探索と，最善応手手順後の局面を比較した評価値の調整を繰り返すことで，棋譜の指手が探索結果と一致するように評価関数を調整する．本稿では，学習の際の損失関数の違いや訓練例の与え方が評価関数の学習に与える影響を，実験を通じて考察し議論する．さらに，実際に 100 万を超える特徴で学習させ，強いプログラムの評価関数の作成に成功したことを報告する．

Learning of Evaluation Functions by Game Records in Shogi

Tomoyuki Kaneko[†1] and Kazunori Yamaguchi[†1]

This paper discusses practical issues in learning of evaluation functions, based on comparison of moves that appeared in a database of game records. In the framework recently researched in shogi, the weights of evaluation functions are adjusted in such a way that the value of the move selected by human players become higher than that of alternative moves, where the value of a move is the evaluation value of a leaf position of the principal variation. In this paper, the effectiveness of four kinds of loss functions as well as other parameters in the framework are discussed. We also show that the weights of more than a million of the weights of the features were successfully adjusted in the method.

†1 東京大学大学院総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo

1. はじめに

ゲームプログラミングにおいて実用的な評価関数を自動で作成す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る．評価関数は，有利・不利を表す評価値を局面に対して与えるもので，特徴と各特徴を重みづけるパラメータからなる．特徴とは局面の一部に注目して数値化するもので，将棋の場合には駒の損得や玉の危険度などが用いられる[17]．評価関数が広く用いられる線形結合の場合は，特徴の重み付きの和が評価値となる．複雑な評価関数において，プログラマが各特徴に適切なパラメータを設定することは難しい．一方で，機械的な調整も容易ではない．

2006 年になって，棋譜の指手が探索で選ばれるように評価関数のパラメータを調整する手法が保木[22] により提案され，実際に強いプログラムが作られた．以来，40 以上のプログラムが強さを競い合う世界コンピュータ将棋選手権においても，学習を採用したプログラムの躍進が目覚ましい．2009 年には優勝した GPS 将棋[20] をはじめ上位 5 位までが，さらに 2010 年には上位 8 位までが学習を採用したプログラムにより占められた．

実用的には，保木の手法の計算時間の短縮が期待されている[⋆1]．学習時間はパラメータの具体的な調整方法に強く依存すると考えられるため，本稿では勾配を用いた調整方法を利用し，実際に 100 万を超える特徴を用いて十分に強い評価関数の作成に短期間で成功したことを報告する．また，この学習の枠組みの個々の要素については，用いる損失関数や訓練例の与え方などに複数の選択肢が考えられる．本研究では，簡単に実行が可能な小規模な実験において，損失関数や訓練例の与え方などの影響についても議論した．

2. 関連研究

オセロでは多数のパターンのパラメータ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に到達している[5]．これは，局面に対し評価関数がとるべき評価値の教師を与えて調整するものである．一方，将棋や，DeepBlue も含めて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[6]．その理由の 1 つは学習のための評価値の教師を用意することが難しいことと考えられる．オセロでは学習のためのラベルとして，終盤の評価関数に対しては探索で読み切った $[+64, -64]$ の値が，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それぞれ用いられた[5]．チェスや将棋におい

⋆1 2006 年のゲームプログラミングワークショップの質疑で 3 カ月程度と報告された．

ても，一部の終盤では局面に対して勝ち，負け，引き分けのラベルをつけることも可能ではある．しかし，勝ちの局面を細分化してプログラムにとっての勝ちやすさを区別したラベルをつけることができないという点で，オセロと比較して学習が難しくなっている．

チェスや将棋の研究では，局面の比較に基づく学習手法が研究されてきた．TD 法は，手を指す前と指した後の局面を比較してパラメータを調整する手法である．改良版の TDLeaf[3] では，指手の直後の局面を比較する代わりに，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．TDLeaf ではそれなりの強さを実現したが，トップレベルのプログラムには到達できていない．他に探索木に注目して探索後の評価値を教師にする手法も提案されている[16]．しかし学習後の強さは TDLeaf のものに劣っている．実用的な学習は保木により将棋で初めて実現された[22]．次の章で説明するがその主要なアイデアは，棋譜の指手の後の局面と他の合法手の後の局面（兄弟）を比較し，棋譜の指手の方が評価値が高くなるようにパラメータを調整することである．アイデアの一部は様々な研究に古く遡ることができる[1),11),12),15),21)]．しかし，反復を行っていない，損失関数の形が適していないなどそれぞれ問題があり，実用的な結果は得られていなかった．

3. 兄弟節点の比較に基づくパラメータの調整

初めに，棋譜の指手が探索により選択されるように評価関数を調整するための一般的な枠組みについて説明する．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よい．実際には，最善応手手順は評価関数に依存し，評価関数のパラメータを変更すると最善応手手順が変化しうる．そのため，棋譜の指手が探索で選ばれる方向にパラメータを調整したとしても，調整後に再度探索した場合に調整前よりも棋譜の指手が探索で選ばれやすいという理論的な保証はない．しかし，実データに対しては一致率が向上することが保木[22] の提案以降，多数の実験によって肯定的な結果が得られている．

具体的な手順を，**図 1** に掲載する．大きく分けて，手順 (1) で最善応手手順を求める部分と手順 (2) でパラメータを調整する部分の 2 つからなり，収束に近づくまで全体を繰り返す．

3.1 最善応手手順の探索

手順 (1) では，棋譜の各局面ですべての合法手に対して最善応手手順を求める．ここでは，使う棋譜や，最善応手手順を求めるための探索手法などに選択肢がある．棋譜については，本稿ではプロ棋士のものとアマチュアのものの比較を行った．探索手法の選択肢には，探索の幅や深さと $\alpha\beta$ 探索における探索窓がある．前者については保木の提案[22] にならって，全幅で一手読み[⋆1]と静止探索を行った．静止探索は探索の末端で評価値を安定させる技術で，ここでは取る手と成る手の中で駒損しない指手を対象とした．ただし，静止探索を長手数行うことは計算時間がかかるため，4 手で打ち切った．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，妥当であると考えている．なお，このような学習で静止探索を行うことの重要性は，過去にチェスの小規模な実験でも報告されている[15]．探索窓に関しては，棋譜で指されなかった手の探索を行う際に棋譜で指された手の評価値を中心とした適切な窓を設定することで，$(-\infty, \infty)$ の探索を行うよりも探索時間を節約することができる[22]．しかし，幅を狭くしすぎると評価値が正確でなくなるという問題点もあるため，その適切な幅は実験的に定める必要がある．

(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)　合法手をすべて作成する．
　(b)　棋譜の指手の後の局面 s_i と，他の手の後の局面 t_i のペア $\langle s_i, t_i \rangle$ を作る
　(c)　各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，棋譜の指手の評価値が他より高くなるように，以下の手順でパラメータを更新する
　(a)　各ペア $\langle s_i, t_i \rangle$ 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)　ペアの差分を x_i とする（$x_i = x_{\bar{s},i} - x_{\bar{t},i}$）．また，評価値の差が正であるべきか負であるべきかを表す変数を y_i とする（先手番なら 1, 後手番なら -1）
　(c)　x, y を用いて，$yw^{\mathrm{T}}x > 0$ となるように w を調整する．
(3)　(2) に戻り，新たなパラメータで探索し，その結果をもとに w を調整し直すことを繰り返す．

図 1　評価関数調整の枠組
Fig. 1　Framework for learning of evaluation functions.

3.2 パラメータの調整

手順 (2) では手順 (1) の結果を利用してパラメータを調整する．この段階で，局面のペア $\langle \bar{s}_i, \bar{t}_i \rangle$ と，手番を表す係数 y_i がすでに求まっている．$\bar{s}$ は棋譜で指された手の局面で

⋆1 なお，その後の 2009 年のコンピュータ将棋協会の例会において，全幅探索の深さを一手から二手に増やしたところ強さの向上がみられたとの報告が存在する．

(1)	$\sum_i \log(1+e^{-y_i w^{\mathrm{T}} x_i})$
(2)	$\sum_i \max(1-y_i w^{\mathrm{T}} x_i, 0)$
(3)	$\sum_i e^{-y_i w^{\mathrm{T}} x_i}$
(4)	$\sum_i \frac{1}{1+e^{y_i w^{\mathrm{T}} x_i}}$

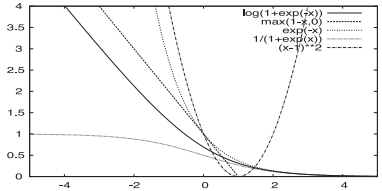


図 2　損失関数の形状
Fig. 2　Loss functions.

求められた最前応手手順の末端の局面，$\bar{t}$ は棋譜とは異なる指手の局面で求めた最前応手手順の末端の局面，y_i は先手番ならば 1，後手番なら -1 とする．これらの局面の選好関係をなるべく満たすように評価関数のパラメータ（ベクトル）w を調整する．簡単のため評価関数は線形であるとして，局面 p の特徴ベクトルを x_p とすると，局面 p の評価値は w と x_p の内積 $w^{\mathrm{T}}x_p$ である．したがって，局面 p は局面 q より評価値が高いという条件は，$w^{\mathrm{T}}x_p > w^{\mathrm{T}}x_q$ と表せるが，それは $x_i = x_p - x_q$ を用いて $w^{\mathrm{T}}x_i > 0$ と変形できる．ルート局面が後手（Min プレイヤ）の場合は棋譜で選択された手の評価値を低くすることが目的となるので，最終的に手番を表す変数 y_i を掛けて $y_i w^{\mathrm{T}}x_i > 0$ と表現される．パラメータの調整は，このような不等式が多数与えられたときにそれらをなるべく満たすような w を求める問題となる．機械学習では上で説明した不等式への違反の度合いを「損失」として定量化し，それらを最小化するようにパラメータを調整する．具体的には，個々のデータの損失を関数 f で表すとして，その総和である関数 $L = \sum_i f(y_i w^{\mathrm{T}}x_i)$ を最小化する．ここで，f にどのような関数を用いるかと，パラメータを具体的に動かす方法について選択肢がある．

3.2.1　損 失 関 数

図 2 左に本稿の実験で用いる損失関数を掲載する．関数 (1) がロジスティック回帰に相等することをはじめ，(1) から (3) の関数は一般的によく使われる関数である[4]．また関数 (4) は保木[22] が使用したものである．図 2 の右側には，それらに最小二乗誤差を加えたものを掲載する．横軸が $y_i w^{\mathrm{T}}x_i$ の値で，正に大きいほど良い評価関数である．縦軸は，横軸に対応する損失の大きさで，最小化する対象である．図から分かるように，関数 (1) から (3) は左上から右下に向けて小さくなるという形状をしている．関数 (4) は，それと似ているが，左端の部分で損失の増大が頭打ちになるという特徴がある．さらに，最小二乗誤差は右側の部分でも損失が大きくなるという違いがある．

損失関数の選択の際に重要なことは，棋譜には，コンピュータの浅い探索では理解できないような高度な指手も，ポカのような指手も存在する点である．それらはどちらも，計算機の探索結果とは一致しにくい．保木が選択した関数 (4) は，学習の際にそれらの指手の影響を最小限にとどめるためとされている[22]．同様に，すで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，その逆を実現するために，パラメータを動かすことの価値は小さい．最小二乗法は，正にも負にも離れた評価の損失を重視しすぎるために有効でないと考えられる．この点で，関数 (1) から (3) は正の部分より負の部分の方が傾きが大きいため，評価値について正しい順序となった局面のペアについて，評価値の差をより広げる改善よりは，誤った順序となったペアを正しくする方向を重視するという望ましい性質がある．

3.2.2　パラメータの調整

続いて，定義された損失関数を最小化するための具体的なパラメータの調整方法について議論する．将棋の評価関数のパラメータの次元は，Bonanza 4 の場合は 1 億，GPS 将棋でも 400 万以上と大規模である．また訓練例もそれ以上に用意される．したがって，直接解法ではなく反復解法を用いる必要がある．さらに，係数行列はもとより分散共分散行列等も，メモリやハードディスクに収めることが難しいという点に注意を払う必要がある．近年，大規模なロジスティック回帰においても様々な計算手法が提案されている[2),7)–10),13)] が，本稿では様々な損失関数について実験するためにより一般的な勾配法を用いた．

勾配法では，t を反復回数とし，以下の式に基づいてパラメータ w_t を順次更新する：

$$w_{t+1} = w_t - \eta_t \nabla_t.$$

ここで，∇_t は w_t の付近での L の勾配を表し，η_t は学習の速度を制御するスカラーである．保木[22] は，各パラメータを動かす幅に勾配を直接利用せずに，勾配の符号に基づいた定数幅の変更を行っている：$w_{t+1} = w_t - \eta_t sign(\nabla_t)$．これは，目的関数が滑らかでない（$w$ を変更すると最善応手手順が変化しうる）ことを重視した設計と述べている．

学習の効率を高めるためには η_t が重要な役割を果たすことが，著者らの予備的な実験では分かっている．このことは，たとえば歩の点数を 10 点変更すると探索木が大きく変わるが，位置評価の 10 点であればそれほどでもないといった直感的な理解とも一致する．そこで本稿の実験では，stochastic meta descent[14] という手法を導入した．η_t はベクトルに拡張され，w の更新の際には以下のように要素別に幅が制御される：

$$w_{t+1} = w_t - diag(\eta_t)\nabla_t.$$

なお diag は対角行列を作る演算子とする．ベクトル η_t は，以下のように更新される：

$$\eta_t = diag(\eta_{t-1}) Max_{0.5}(1 - \mu\, diag(\nabla_t) v_t),\ v_{t+1} = \lambda v_t - diag(\eta_t)(\nabla_t + \lambda H_t v_t). \quad (1)$$

ただし μ はメタ学習定数，$Max_{0.5}$ は 0.5 より小さい要素を 0.5 で置き換えたベクトルを返

す演算，また λ は $[0,1]$ の範囲の値で，慣性のような効果を持つパラメータである．H は L の w に関するヘッセ行列で，v は η を動かした場合の w への影響を近似するベクトルである．詳しくは文献 14) を参照されたい．実装上は H をメモリ上に持つ必要はなく，∇ の計算と同時に Hv を求めることができるため，計算コストはほとんど増えない．

さらに，予備的な実験において初期値 η_0 の値も重要であることが分かった．この適切な値は，用いる特徴集合によって異なるため，すべての実験で共通の値を用いることは難しい．そこで，今回の実装では，パラメータの更新により損失関数 L の値を小さくすることに連続して成功している間は η_{t+1} を式 (1) で求めた値からさらに 1.2 倍に，失敗した場合は半分にするというヒューリスティックな更新を採り入れた．これにより初期値の差をかなり吸収できることが分かったため，本稿の実験ではすべて 10^{-6} を初期値として用いた．最後に，以上の設定では $\lambda = 0$ で問題なく動くことが分かったため，式 (1) は結局，1 つ前の勾配と今回の勾配の積に応じて η を増やすという意味になる．

4. 実験結果

4.1 駒の価値の学習

はじめに，駒の価値のみの評価関数を作成し，そのパラメータを学習させた．まず，**表 1** の上部に記載の初期値で探索して訓練例を作成した．この値は手調整では最も優れた評価関数を持つと思われる激指の駒の価値について歩の価値が 128 になるようにスケールしたものである[17]．続いて，パラメータを初期値 0 から調整してどのような値となるかを調査した．実験条件を単純にするため，図 1 における手順 (1) と (2) を 1 度ずつ行った段階での結果を示す．手順 (2) においてはパラメータの更新を 100 回行ったところで打ち切った．初期値に近い値が復元できれば，学習に成功したと考えられる．学習は浮動小数で行ったが，結果は歩の値が 128 点となるように全体をスケールして表示する．最善応手手順を求める際の探索窓は，歩の枚数に換算して 3 枚分，8 枚分，16 枚分の 3 通りを試した．また棋譜は，将棋クラブ 24[18] の棋譜，同将棋クラブ 24 で先手後手双方のプレイヤのレートが 1,500 以上のもの（以降では簡単に有段の棋譜と呼ぶ），プロの棋譜の 3 種類をそれぞれ約 200 試合を用いた．

結果をまとめて表 1 に掲載する．全体の傾向として似たような値になっているが，関数 (2) は値が小さい（歩の価値が相対的に高い）傾向にある．また，棋譜による差はあまりない．探索窓は狭い方が飛車や龍などの価値の高い駒の値が高くなり，駒の値の差がはっきり出るようである．代表として，飛車の値をグラフに描いたものを**図 3** に掲載する．飛車の

表 1　駒の価値：初期値と学習後（歩の値を 128 としたときの相対的な値）
Table 1　Initial and learnt values (scaled so that pawn = 128).

棋譜 窓関数		香	桂	銀	金	角	飛	と	成香	成桂	成銀	馬	龍
初期値		512	512	704	768	1,024	1,216	768	768	768	768	1,472	1,664
24	3 (1)	401	409	589	678	816	977	650	618	633	661	1,195	1,347
	(2)	260	260	388	408	520	648	392	242	314	364	776	904
	(3)	381	392	569	659	775	904	654	660	630	627	1,141	1,255
	(4)	380	398	556	636	785	948	598	417	558	580	1,145	1,301
24	8 (1)	342	350	514	606	684	825	575	574	568	608	1,027	1,153
	(2)	240	243	348	409	449	546	384	213	302	299	677	768
	(3)	339	347	507	596	680	799	578	610	576	589	1,017	1,110
	(4)	324	346	491	583	666	812	543	433	499	562	999	1,140
24	16 (1)	323	330	489	579	642	773	549	483	527	566	964	1,082
	(2)	215	220	314	370	385	467	356	162	236	249	565	630
	(3)	322	317	479	559	625	733	543	562	553	555	940	1,022
	(4)	301	322	460	546	616	746	509	356	447	489	918	1,043
有段	3 (1)	409	422	604	696	838	1,010	658	598	652	664	1,220	1,385
	(2)	257	257	385	418	513	641	385	223	312	319	769	897
	(3)	386	399	573	661	775	928	638	599	639	632	1,122	1,280
	(4)	377	397	559	642	787	946	586	319	518	461	1,126	1,284
有段	8 (1)	340	358	521	614	697	845	577	538	579	590	1,039	1,186
	(2)	246	252	360	426	462	562	386	246	312	298	703	790
	(3)	341	356	515	603	687	821	567	573	589	594	1,014	1,149
	(4)	317	346	491	582	666	804	541	427	495	489	994	1,126
有段	16 (1)	324	342	500	592	661	801	561	510	558	562	990	1,126
	(2)	218	226	324	384	400	490	384	171	243	230	596	658
	(3)	313	329	479	563	625	742	541	547	567	557	925	1,041
	(4)	310	338	481	570	647	779	531	417	482	478	965	1,089
プロ	3 (1)	413	428	605	698	879	1,030	677	563	657	699	1,247	1,404
	(2)	257	257	385	387	514	642	385	157	276	336	770	897
	(3)	400	414	588	674	843	962	663	587	638	679	1,192	1,267
	(4)	376	388	546	631	799	951	606	300	508	550	1,121	1,285
プロ	8 (1)	344	362	516	601	743	853	590	450	564	597	1,051	1,168
	(2)	238	248	338	386	468	546	387	145	266	273	642	712
	(3)	359	373	531	610	764	855	602	554	603	614	1,071	1,137
	(4)	296	318	449	538	653	767	531	280	419	459	921	1,053
プロ	16 (1)	329	346	495	577	708	809	575	421	536	573	999	1,109
	(2)	203	212	287	340	392	446	337	103	179	206	494	536
	(3)	331	344	499	568	701	777	569	503	565	580	978	1,031
	(4)	299	321	453	543	658	771	537	322	438	495	930	1,061

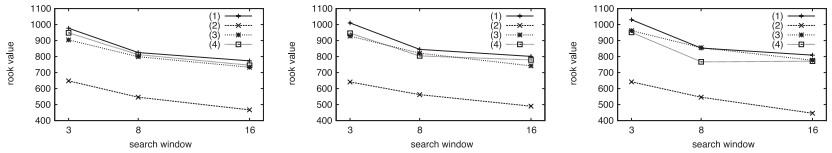


図 3　パラメータを変えた場合の飛車の値の変化（左：将棋クラブ 24，中央：将棋クラブ 24 有段，右：プロ）
Fig. 3　Rook values for each record set (Left: Shogi Club 24, Center: dan players in Shogi Club 24, Right: professionals).

価値が 1,000 点を超えたのは，関数 (1) を用いた場合のみ（将棋クラブ 24 有段者の棋譜とプロの棋譜で探索窓が歩 3 枚分のとき）である．以上より，探索時の評価関数に近いパラメータが求められる調整法が適切だとすれば，関数 (1) が適していると考えられる．

4.2　進行度等を加味した駒の価値の学習

続いて進行度を用いる評価関数において，先ほどの傾向が変化するかどうかを調査した．具体的には，進行度 p が 0 から 1 の値をとるとして，終盤の駒の価値と玉の安全度の差[19]を加えた以下の評価関数を用いた：

$$\text{評価値} = \text{駒の価値} + p \cdot (\text{終盤用駒の価値} + \text{玉の安全度の差}).$$

進行度 p は玉の周りの利きや持駒を考慮して決められる．

また，文献 22) を参考に，学習後の棋譜との不一致度を以下のように測定した：

$$\text{不一致度} = \frac{1}{\text{局面数}} \sum_{\text{合法手 i}} T\left(\xi(\text{i}) - \xi(\text{棋譜の手})\right).$$

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 $T(x) = 1/(1+\exp(-3x/128))$ で 128 は歩の点数である．関数 T は，評価値の差が十分ある場合は 0 または 1 となるため，各局面で最善手より評価値が高い指手の数を数えることに相当する．一方，評価値が近い場合は T の値は 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不一致度は大きくなる．この不一致度は小さいほど良い評価関数と推測される．なお，学習において探索窓を絞った場合も，棋譜との不一致度を測る際には共通の広い窓を用いた．

表 2 に結果をまとめる．紙面の都合で，将棋クラブ 24 の棋譜を使ったもののみ掲載し，小駒の成駒を省いた．全体にわたり，ほぼ駒の強さに応じた点数の順番になっている．また，終盤になると，歩の価値は下がり金の価値は上がる傾向がある．しかし，個々の値はかなり異なったものとなっている．棋譜との不一致度に注目すると，**図 4**(左) に示したように，全体として探索窓を広げる方が微差ではあるが良い成績 (値が低い) となっている．また，損失関数の違いに着目すると関数 (4) が最も成績が良い．これは不一致度と同じ形式の関数を最適化の対象としていることを考えれば自然である．(4) に続いては (1) の成績が良い．玉の安全度の差についても，図 4(右) に示したように探索窓や損失関数によって傾向が異なる．文献 19) では角 2 枚分程度に調整されたプログラムが強かったことから大きい方が良い調整であると考えると，探索窓は広くとる方が良く，また損失関数は関数 (4) や (1) が向いていると考えられる．最後に代表として飛車の値についてグラフにしたものを，**図 5** に示す．opening は序盤（進行度 $p = 0$）の値で，ending は終盤（進行度 $p = 1$）の値である．前節同様に探索窓が広がるほど値は下がる傾向にある．また，前節とは異なり関数 (1) よりも関数 (4) の方が高い値を保っている．図 3 と比較すると序盤の値ははっきり低いが終盤の値は似たようなものになっている．

表 2　学習結果（歩の値を 128 としたときの相対的な値）
Table 2　Learnt values (scaled so that pawn = 128).

窓関数特徴	歩	香	桂	銀	金	角	飛	馬	龍	安全度差	不一致度
3 (1) 全体	128	337	336	475	517	649	772	864	994	56	12.36
終盤	−29	38	21	66	113	96	137	296	310		
3 (2) 全体	128	251	249	356	370	472	585	685	792	48	12.52
終盤	−40	41	31	60	113	111	130	209	226		
3 (3) 全体	128	320	332	509	544	704	827	925	1,064	50	12.37
終盤	−29	31	6	−13	46	−31	−32	141	102		
3 (4) 全体	128	382	358	453	526	649	789	977	1,102	85	12.21
終盤	−27	41	59	180	192	251	334	368	478		
8 (1) 全体	128	293	293	413	460	555	642	727	819	64	12.20
終盤	−33	9	−8	32	74	29	91	238	263		
8 (2) 全体	128	207	209	276	306	349	422	505	567	69	12.25
終盤	−44	30	5	68	108	102	145	220	263		
8 (3) 全体	128	296	304	461	512	631	733	805	913	49	12.34
終盤	−32	−1	−27	−46	−13	−81	−69	102	74		
8 (4) 全体	128	323	305	382	471	539	683	868	970	101	11.97
終盤	−39	34	42	175	188	215	308	323	420		
16 (1) 全体	128	276	276	387	425	511	584	660	745	67	12.14
終盤	−36	0	−21	23	71	13	82	227	248		
16 (2) 全体	128	196	194	258	278	316	382	464	529	79	12.15
終盤	−45	28	4	64	117	91	142	205	240		
16 (3) 全体	128	284	295	438	480	587	675	722	820	49	12.29
終盤	−37	−19	−62	−66	−35	−110	−95	94	60		
16 (4) 全体	128	322	316	388	469	533	674	873	967	102	11.96
終盤	−32	40	29	169	188	218	308	283	385		

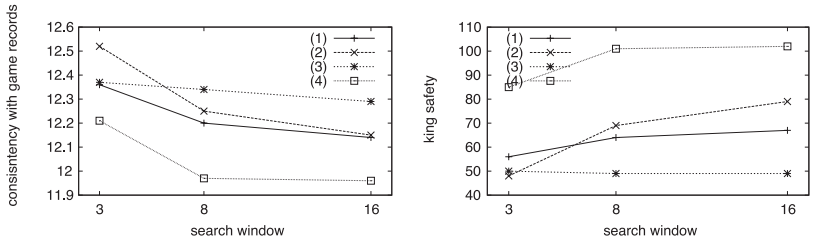


図 4　棋譜との不一致度 (左) と玉の安全度の評価 (右) の探索窓との関係
Fig. 4　Inconsistency with game records (left) and weights for king safety (right) for each search window.

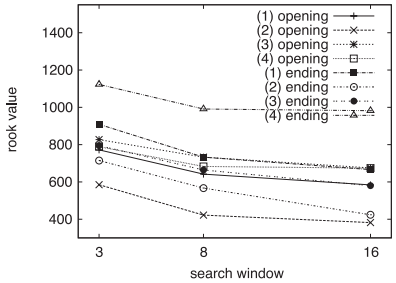


図 5　探索窓と飛車の値の関係
Fig. 5　Rook value for each search window.

これだけの実験で明確な結論を出すことは難しいが，まず，駒の価値のような基本的な特徴の値ですら，実験条件により大きく左右されることが分かる．詳細には，探索窓を狭くするとデメリットがある可能性があり，また損失関数は関数 (1) や (4) が良いと推測される．

4.3　実用的な評価関数の調整とその評価

続いて，学習手法の性能評価を，実用的な正確さの評価関数を対象に行った実験について報告する．まず強いプログラムの評価関数を用意し，駒の価値以外のパラメータを 0 に初期化したのちに，本稿で提案した手法で学習させた．評価関数は 2009 年の世界コンピュータ将棋選手権で優勝した GPS 将棋の最新版（2010 年 1 月 17 日時点，OSL revision 4061）のものを用いた．文献 20) で説明されているように，GPS 将棋の評価関数には将棋の様々な概念が組み込まれている．多くの特徴は序盤，中盤，終盤に別々のパラメータを持ち，この実験で使ったバージョンでは合計で約 471 万種類のパラメータがある．序盤，中盤，終盤の評価関数の使い分けは進行度を p として

$$\text{評価値} = \begin{cases} (0.5-p)\cdot \text{序盤} + p\cdot \text{中盤} & (0 \le p < 0.5) \\ (1-p)\cdot \text{中盤} + (p-0.5)\cdot \text{終盤} & (0.5 \le p < 1.0) \end{cases}$$

と進行度に基づく加重平均が用いられる．進行度の値の計算についても GPS 将棋のものをそのまま用いた．

表 3　反復の進行と成績の変化
Table 3　Improvements on performance for each iteration.

反復	不一致度	一致率 (%)		有効特徴	平均	対戦勝敗		学習時間 (hours)		最大変化
		含タイ	除タイ			手調整	2009	探索	調整	
0	13.18	68.1	15.3	13	-	-	-	-	-	-
1	4.25	37.7	36.6	419209	3.5	16–24	4–36	7.18	10.2	295
2	3.31	41.4	40.4	581912	5.4	26–14	4–36	8.03	10.3	151
3	3.01	42.8	41.9	653366	7.4	31– 9	7–33	7.57	11.7	175
4	2.83	43.5	42.6	692974	9.6	32– 8	11–29	7.15	11.8	137
5	2.69	43.8	43.0	716255	10.7	29–11	18–22	6.68	11.8	93
6	2.62	44.3	43.5	733482	12.7	35– 5	13–27	6.50	11.9	86

棋譜は，プロ棋士が指したとされる約 3 万の棋譜から，反復ごとに 3,000 の棋譜をランダムに選んで使用した．これは学習時間を減らすためである．また損失関数はロジスティック回帰のものを用い，各反復においては 50 回パラメータを更新した時点で次の反復に移行して探索木を作り直した．ただし，パラメータが大きく動いた場合は 50 回を待たずに打ち切った．その条件としては，変化量のベクトルのノルムが元のベクトルのノルムを超えた場合と設定した．これは経験的に良いと予測される設定ではあるが，探索木の更新とパラメータの更新をどのようにバランスさせるかは未解決の問題である．また正則化は用いなかったためパラメータが 0 となる場合は，値が非常に小さく整数に変換した際に 0 となった場合か，そもそも棋譜の探索結果に表れなかった場合である．

結果について，学習の所要時間とともに得られた評価関数の性能について評価を行った．主な内容を**表 3** に掲載する．左から順に，反復の回数，棋譜との不一致度と指手一致率，0 でないパラメータ（有効特徴）の数，それらの絶対値の平均，対局の勝敗，その回の反復の所要時間，最も大きく変化した値の変化量を掲載した．

不一致度は反復が進むにつれて単調に減り，またその値は前節の実験での値よりも大幅に低い．同様に一致率は反復が進むほど単調に上昇し，この値は今までに報告されたどの文献よりも高い．これらの結果から，用いた特徴が実際に有効なものであり，また GPS 将棋

の強さとの関連が示唆される．なお，棋譜との不一致度と指手の一致率の測定に関しては，その回の反復において学習に使わなかった棋譜からランダムに選んだ棋譜を用いた．テスト用の棋譜を完全に切り分けなかった理由は，棋譜により不一致度や一致率がかなり異なるため，テストのための標準的な棋譜集合を選ぶことが難しかったためである．

対局による評価は，2 種類のプログラムを相手に実験を行った．1 つは，2009 年の世界コンピュータ将棋選手権で優勝したバージョンの GPS 将棋[⋆1]である．このバージョンの評価関数は，特徴の追加と追加前のパラメータを初期値とした学習を繰り返して作られていて，本稿のように 0 から学習させたものではない．こちらは現時点で最も強いとされるプログラムという位置づけである．もう 1 つは 2008 年まで GPS 将棋が使っていた手調整に基づく評価関数で，floodgate という自動対局場[⋆2]では gps_normal として知られるプログラムである．こちらは 2005 年にはコンピュータ将棋選手権の決勝に進み他の年も次点近い成績を残すなど，トップレベルではないもののそれを追いかけるグループの強さといえる．定跡の影響を排除して勝敗の比較を行うために，事前に様々な戦型の定跡を 20 棋譜選び，初形から 30 手まで進めた局面を用意した．また先手後手の影響を除くために，1 つの局面で両方の手番を割り当てて合計 40 試合を行った．持時間は一手 30 秒とした．環境は Linux を使用し，CPU はそれぞれ AMD Opteron 285（対 2009 年版），280（対手調整版）である．CPU の違いは，使い分けに意味があるということではなく，空いている機材で並行して実験を進めたためである．また駒価値のみの評価関数では勝負にならないため，初期値では対局を行っていない．手調整版に対しては，2 回目の反復ですでに勝ち越し，その後も勝率を伸ばしている．2009 年版に対しては，手調整版よりも成績が悪いがそれでも勝率はゆっくりと改善されている．**図 6** に反復ごとの勝率を，二項分布を仮定した場合の 5%信頼区間とともに，グラフに示す．各反復ごとに勝率が向上しているとまではいいきれないが，たとえば手調整版に対する勝率では 4 回目以降は統計的に有意に 0.5 を超え，初回の勝率と比べても統計的に有意に向上しているといえる．最後の反復では，一致率や不一致度は改善されているにもかかわらず，2009 年版に対する勝率が悪化した．これは対局数が足りなかったという偶然の可能性もあるが，複数のプログラムに安定して強さを向上させる難しさの一端を示唆していると考えられる．対局させた 2 つのプログラムの評価関数は，それぞれ前述の floodgate での gps_normal と gps_l に近いと考えられるが，floodgate においてもレートのわりに 2 つのプログラムのどちらかを苦手や得意とするプログラムが実際に複数存在する．

⋆1 http://gps.tanaka.ecc.u-tokyo.ac.jp/cgi-bin/viewvc.cgi/tags/wcsc19/ から入手可能である．
⋆2 http://wdoor.c.u-tokyo.ac.jp/shogi/ プログラムの強さの測定に用いられている

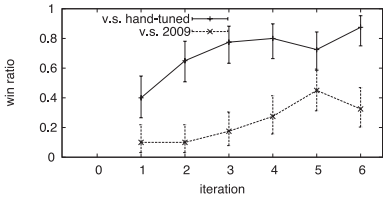


図 6　反復ごとの勝率
Fig. 6　Win ratio for each iteration.

総合して，数回の反復でかなりの強さを実現したといえる．また反復に要する時間は，Intel Xeon X5470 の PC の 8 コアを利用して，1 回あたり 19 時間未満である．合計 1 週間未満でこの強さを実現したのは初めての報告である．具体的に，最も大きく変化したパラメータに注目すると，初回は 200 を超えその後も 100 近辺の変化がある．一方，変化の平均値は小さいことから，変化が小量にとどまる特徴も多数存在すると考えられる．保木の手法ではすべての特徴が固定幅で変化する[22]ためこのような柔軟な調整は難しい．よって少ない回数の反復で強さの向上を実現できた背景には，stochastic meta descent[14]による調整が有効に働いていると考えられる．最後に有効特徴の数は，反復が進むほど数が増え大きくなる．しかしその増え方は緩やかで，実験対象の GPS 将棋の最新版では 120 万を超える特徴に 0 でないパラメータがついていることと比較するとかなり少ない．これは数回の反復では，まだ十分に収束していない可能性が示唆される．

5. おわりに

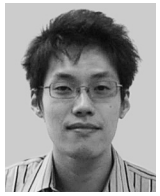
兄弟節点の比較に基づく評価関数の調整方法は，近年成果をあげている有望な手法である．これは探索木の生成とパラメータの調整を繰り返す複雑な手法であり，個々の詳細に関してはこれまで一部の組合せでしか報告がなかった．本稿ではまず小規模な評価関数を対象に様々な条件での学習結果の比較を行い，複数の種類の棋譜と損失関数で現実的な値が得られることから，手法のある程度の頑健さを確認した．続いて，100 万を超える特徴と進行度を含む複雑な関数形を持つ大規模な評価関数を用いた実験において，パラメータ調整にロジスティック回帰を用いた場合でも効果的な学習が行われることを確認した．このことは，手法全体を探索木の生成とパラメータ調整を 2 つの別の問題として扱うことで，他の高速な調整手法を組み合わせることが可能になったという意義がある．実際に学習で得た評価関数

を用いたプログラムを GPS 将棋と対局させたところ，初回の反復から勝率の上昇が確認された．そして 1 週間程度の調整後には，手で調整した評価関数には有意に勝ち越し，2009 年の世界コンピュータ将棋選手権で優勝したバージョンとも十分戦えるものとなった．

参 考 文 献

1) Anantharaman, T.: Evaluation Tuning for Computer Chess: Linear Discriminant Methods, *ICCA Journal*, Vol.20, No.4, pp.224–242 (1997).
2) Andrew, G. and Gao, J.: Scalable training of L1-regularized log-linear models, *ICML*, Ghahramani, Z. (Ed.), pp.33–40, Omnipress (2007).
3) Baxter, J., Tridgell, A. and Weaver, L.: Learning to Play Chess Using Temporal-Differences, *Machine Learning*, Vol.40, No.3, pp.242–263 (2000).
4) Bishop, C.M.: *Pattern Recognition and Machine Learning* (*Information Science and Statistics*), Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006).
5) Buro, M.: Improving heuristic mini-max search by supervised learning, *Artificial Intelligence*, Vol.134, No.1–2, pp.85–99 (2002).
6) Campbell, M., Joseph Hoane, Jr., A. and Hsu, F.: Deep Blue, *Artificial Intelligence*, Vol.134, No.1–2, pp.57–83 (2002).
7) Globerson, A., Koo, T., Carreras, X. and Collins, M.: Exponentiated gradient algorithms for log-linear structured prediction, *ICML*, Ghahramani, Z. (Ed.), pp.305–312, Omnipress (2007).
8) Koh, K., Kim, S.-J. and Boyd, S.: A method for large-scale l_1-regularized logistic regression, *22nd National Conference on Artificial Intelligence*, pp.565–571 (2007).
9) Lee, S.-I., Lee, H., Abbeel, P. and Ng, A.Y.: Efficient L1 Regularized Logistic Regression, *AAAI*, pp.401–408, AAAI Press (2006).
10) Lin, C.-J., Weng, R.C. and Keerthi, S.S.: Trust region Newton methods for large-scale logistic regression, *Proc. 24th International Conference on Machine Learning*, Vol.227, pp.561–568 (2007).
11) Marsland, T.: Evaluation Function Factors, *ICCA Journal*, Vol.8, No.2, pp.47–57 (1985).
12) Nowatzyk, A.: (2000). http://tim-mann.org/DT_eval_tune.txt
13) Komarek, P. and Moore, A.: Fast Robust Logistic Regression for Large Sparse Datasets with Binary Outputs, *Artificial Intelligence and Statistics* (2003).
14) Schraudolph, N.N.: Fast Curvature Matrix-Vector Products for Second-Order Gradient Descent, *Neural Computation*, Vol.14, No.7, pp.1723–1738 (2002).
15) Tesauro, G.: Comparison training of chess evaluation functions, *Machinesthat Learn to Play Games*, Furnkranz, J. and Kumbat, M. (Eds.), pp.117–130, Nova SciencePublishers (2001).
16) Veness, J., Silver, D., Uther, W. and Blair, A.: Bootstrapping from Game Tree Search, *Advances in Neural Information Processing Systems 22*, Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C.K.I. and Culotta, A. (Eds.), pp.1937–1945 (2009).
17) 鶴岡慶雅：将棋，情報処理，特集 ゲーム情報学，Vol.44, No.9, pp.900–904 (2003).
18) 久米　宏：将棋倶楽部 24 万局集，ナイタイ出版 (2002).
19) 竹内聖悟，金子知適，林　芳樹，山口和紀，川合　慧：勝率に基づく評価関数の評価と最適化，情報処理学会論文誌，Vol.48, No.11, pp.3446–3454 (2007).
20) 金子知適：コンピュータ将棋の新しい波：3. 最近のコンピュータ将棋の技術背景と GPS 将棋，情報処理，Vol.50, No.9, pp.878–886 (2009).
21) 金子知適，田中哲朗，山口和紀，川合　慧：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 (2003).
22) 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，pp.78–83 (2006).

(平成 22 年 1 月 25 日受付)
(平成 22 年 5 月 6 日採録)



金子　知適（正会員）

1997 年東京大学教養学部卒業．2002 年東京大学大学院総合文化研究科博士課程修了．博士（学術）．2002 年東京大学院総合文化研究科助手．2007 年助教．思考ゲーム，知識処理に興味を持つ．

山口　和紀（正会員）

1978 年東京大学理学部数学科卒業．1981 年東京大学理学部助手．1985 年理学博士．1989 年筑波大学電子情報工学系講師．1992 年東京大学教養学部助教授．1999 年東京大学情報基盤センター教授．2007 年東京大学総合文化研究科教授．モデリング全般に興味を持つ．ACM 会員．