

将棋での少数の棋譜からの評価関数の学習における 拘束条件の研究

大森 翔太郎^{1,a)} 金子 知適^{1,b)}

概要: 本研究では、棋譜を教師に将棋の評価関数のパラメータを学習する手法と既存の評価関数のパラメータに近づける手法を組み合わせることを提案する。実際に Bonanza (version 6.0) というプログラムと羽生 4 冠の棋譜で実験を行ったところ、拘束条件の違いによって学習させた棋譜との一致率や学習した後の成駒 (と, 成香, 成桂, 成銀) の価値に大きな違いが現れることが観察された。

Research on Constraints in Learning of Evaluation Functions in Shogi with a Limited Number of Records

SHOTARO OMORI^{1,a)} TOMOYUKI KANEKO^{1,b)}

Abstract: In this research, I proposed that to learn evaluation functions combines learning of evaluation functions in shogi with a number of master's records with adaptive learning utilizing parameters of existing evaluation function. I tried it that I used Bonanza (version 6.0) and Habu Yosiharu's who is the best shogi player records. Then I saw very different piece values (propawn and prolance and proknight and prosilver) and prediction on learning records when constraints were different.

1. はじめに

現在までに、プロ棋士にコンピュータ将棋ソフトウェアが大幅に勝ち越している。この結果などから、既にコンピュータ将棋ソフトウェアの棋力は、プロ棋士と同等以上に到達したと考えられている。このことからプロ棋士と同じ棋力に達しているのならば、特定のプロ棋士の指し手を模倣するコンピュータ将棋ソフトウェアが実現できるのではないかと期待がある。もし実現したならば、タイトル戦の局面や自分の対局で自分の教わりたいプロ棋士の考え方をコンピュータ将棋ソフトウェアにアドバイスを求めて、自分の棋力向上に役立てることができる。文献 [3] などからも分かるように、現在の将棋ソフトウェアは、プロ棋士の棋譜を教師とした機械学習で評価関数を作っている。そ

こで、真似したい棋士の棋譜のみを学習に用いれば、その棋士の棋風を評価関数に反映させることができないかと考えられる。しかし、実際には学習に使える棋譜が減ってしまうことから、棋風を持たせる以前に弱くなってしまふなどの弊害があり、現段階では成功したという報告はない。そこで本研究では、次のような手法を提案する。まず大量の棋譜から評価関数を作成する。次に特定の棋士の棋譜を学習し、評価関数を微調整する。その際に、パラメータの重みを 0 に近づける項 (文献 [3] を参照) と多数の棋譜から学習した評価関数からなるべく離れないようにする項 (文献 [4] を参照) を組み合わせる。パラメータの重みを 0 に近づける項と既存のパラメータに近づけようとする項の大きさを変えて、一致率、駒の価値、評価関数の重みの絶対値の平均などの指標を手がかりに評価関数の学習状況において議論した。パラメータの重みを 0 に近づける項を大きくとった場合には、成駒の価値の判断においてもともと他の成駒 (成香, 成桂, 成銀) より値の高いとが他の成駒 (成香, 成桂, 成銀) より小さくなってしまふことや学習した棋譜との一

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

a) omori@graco.c.u-tkyo.ac.jp

b) kaneko@graco.c.u-tkyo.ac.jp

致率が下がってしまうことが分かった。またパラメータの重みを 0 に近づける項を小さくとした場合には、学習した棋譜との一致率が高くなり過ぎてしまい過学習を起こしているような状態になってしまうことが分かった。しかし既存のパラメータに近づける項をパラメータの重みを 0 に近づける項と組み合わせることで、パラメータの重みを 0 に近づける項を大きくとした場合に起きた成駒（と、成香、成桂、成銀）の学習の問題を解決できる。また学習した棋譜との一致率が低い問題やパラメータの重みを 0 に近づける項を小さくとした場合に起きた過学習を起こしているような状態になってしまう問題を解決できることが分かった。ただし既存のパラメータに近づける項の値を決める抑制パラメータ C がある範囲を超えると、どんなに学習回数を増やしても既存の評価関数の駒の価値、学習の一致率を示し続けてしまうということが分かった。

2. 関連研究

評価関数の学習と個性に関する関連研究としては、まず、入玉に関連する局面の評価の改善に取り組んだ文献 [1] が上げられる。入玉の評価はコンピュータ将棋が苦手とすることで知られており、文献 [1] の中で、入玉に対しての弱点を改善する手法を Bonanza を題材にして提案している。コンピュータ将棋ソフトウェアが、入玉に対して弱い理由は学習する棋譜の中で、入玉に関する棋譜が少ないからだと考えられている。この文献の中では、入玉の特徴の重みだけを学習し、元の Bonanza の評価値そのものは変えず、入玉に関する評価を改善した Bonanza を作成している。私の研究でも、Bonanza を用いて学習を行っており、また元の Bonanza の評価値を既存のパラメータとして扱っている。また文献 [2] では、プロ棋士の個性を実現するために、真似したい棋士の棋譜だけで評価関数の機械学習を行うという実験を、Bonanza を題材に行っている。この文献では、特定のプロ棋士の序盤に関しては実現できているが中終盤に関しては実現が難しいと結論づけている。私の研究でも、プロ棋士の棋譜を元に学習を行っており、最終的な目標として、プロ棋士の個性の実現を目指している。

3. 評価関数の学習における複数の正則化パラメータの活用

本研究では、現在の評価関数の学習の標準的な手法である文献 [3] の学習手法を基本に、まず大量の棋譜から評価関数を作成する。次に特定の棋士の棋譜を学習し、評価関数を微調整する。その際に、パラメータの重みを 0 に近づけるパラメータの項（文献 [3] を参照）と多数の棋譜から学習した評価関数からなるべく離れないようにする項（文献 [4] を参照）を組み合わせる。まず文献 [3] の手法を説明する。

$$J^P(w) = J(P, w) + J_C(w) + J_R(w) \quad (1)$$

と定義された関数である。 $J^P(w)$ は目的関数、 $J(P, w)$ は学習を行う棋譜の指し手を将棋ソフトウェアが指せるように評価関数の値を調整するための関数、 $J_C(w)$ は駒の重みに関して制約を加える関数、 $J_R(w)$ はパラメータの重みを 0 に近づける項で駒の位置関係の重みに関して制約を加える関数である。 $J(P, w)$ は

$$J(P, w) = \sum_{p \in P} \sum_{m \in M'_p} T(s(p, d_p, w) - s(p, m, w)) \quad (2)$$

と定義された関数である。 P は学習する棋譜の局面の集合、 p は P 中の 1 つの局面、 M'_p は局面 p での棋譜の指し手を除いた全ての集合、 m は M'_p から選ばれた指し手、 $s(p, w)$ は局面 p で木を探索することによって得られたミニマックスの値、 p, m は局面 p から m 動かした後の局面、 d_p は局面 p での棋譜の指し手、 $T(x)$ は $1/(1 + \exp(ax))$ のことで左右反転したシグモイド関数であり、 $a > 0$ である。 $J(P, w)$ は式 (2) より棋譜の指し手を探索したときに得られた値と合法手の指し手を探索したときに得られた値の差が大きいときには、全体が 0 に近づくので、和を取ると小さくなる。一方で差が小さいときには、全体が 1 に近づくので、結果的に和を取ると大きくなる。そもそも目的関数は全体の値を小さくする w を見つけるのが役目なので、目的関数の項が、 $J(P, w)$ だけだと全体の値を小さくするために $s(p, m, w)$ を $s(p, d_p, w)$ に近づけようとする。 $J_R(w)$ は

$$J_R(w) = \lambda_1 |w''| \quad (3)$$

と定義された関数である。 $\lambda_1 > 0$ 、 w'' は駒の位置関係の重みベクトルである。 w を駒の重みベクトル w' と駒の位置関係の重みベクトル w'' と分けることができる。 $J_R(w)$ の式 (3) では、 w の絶対値が大きければ、目的関数の式全体が大きくなってしまいうので、 $J_R(w)$ の式の値を小さくするために w の値を小さくしようとする。こうすることで、 w の値を抑えて過学習がおこらないようにしている。さらに文献 [4] では、式 (1) に新たな拘束条件の項を加えることを提案していて、

$$J_{R'} = \frac{C}{2} \|w - w_0\|^2 \quad (4)$$

と定義されるものを式 (1) に加えて

$$J_{MMTO}^P(w) = J(P, w) + J_C(w) + J_R(w) + J_{R'} \quad (5)$$

と定義している。 w_0 は既存のパラメータである。こうすることによって既存パラメータとの差が小さいほど追加した式の値が小さくなり、全体の値が小さくなる。つまり全体の値を小さくするために w の値を w_0 に近づけるようになる。本研究では、学習の基本的な枠組みは文献 [3] に倣い、また既存の評価関数との差分を小さくする手法として文献 [4] で提案された手法を用いている。本研究の主要な貢献は、2種類の正則化項すなわち式 (5) の $J_R(w)$ と $\frac{C}{2} \|w - w_0\|^2$ の組み合わせについて、実験的に調査したことである。

4. 評価関数の学習実験

4.1 実験環境

今回の実験では、羽生善治 4 冠の棋譜 1878 局を使用して、Bonanza (version 6.0) で学習を行う。それぞれ元の Bonanza (version 6.0) に配布されている fv.bin, param.h を初期値として 25 回学習を行った。

4.2 元の Bonanza による目的関数の制御

Bonanza で評価関数の機械学習を行う際の評価関数の拘束条件に使われている $J_R(w)$ を変えることで一致率、駒の価値、評価関数の重みの絶対値の平均のグラフがどのように変わるか、の観察を行う。これまでに得られている知見として、Bonanza のソースコード内の shogi.h における FV_PENALTY の初期値である $1/160$ を $1/16000$, $1/1600$, $1/160$, $2/64$, $3/64$, $4/64$ と設定した場合の学習の経過について結果を紹介する。

4.2.1 学習した棋譜との一致率

一致率とは、学習に用いた棋譜とどの程度同じ手を指すかを示したものである。一致率の変化のグラフでは、横軸を Iteration (学習回数) として、縦軸を Prediction (一致率) とした。拘束条件として、用いられているパラメータを 0 に近づける項のみを変更した結果を示す。まず一致率の変化について図 1 に示す。図 1 を見れば分かるように拘束条

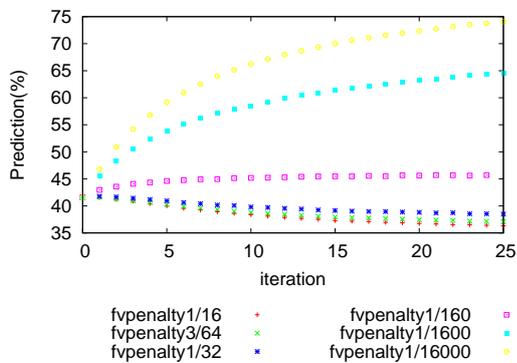


図 1 一致率の変化

件であるパラメータを 0 に近づける項の値が、 $1/16$, $3/64$, $1/32$, $1/160$, $1/1600$, $1/16000$ と小さくなるほど学習回数を増やすたびに、学習した棋譜との一致率が上昇することが分かる。しかし学習した棋譜との一致率が高い場合には、過学習が発生して学習に使用しなかった棋譜での一致率が下がってしまうことが多い。文献 [3] の中では、 $J_R(w)$ がない場合の目的関数とある場合の目的関数を比較している。文献の図を読み取るとパラメータを 0 に近づける項がある場合の目的関数は、学習した棋譜との一致率が学習回数を増やしていくと 40 パーセント程度、学習に使用しなかった棋譜との一致率が学習回数を増やしていくと 36

パーセント程度になる。一方 $J_R(w)$ がない場合の目的関数は、学習した棋譜との一致率が学習回数を増やしていくと 64 パーセント程度、学習に使用しなかった棋譜の一致率が学習回数を増やしていくと 33 パーセント程度と低くなる。私の実験でもパラメータを 0 に近づける項が小さい場合には、学習した棋譜との一致率で文献 [3] での $J_R(w)$ がない場合の目的関数を使用したときの学習した棋譜の一致率のグラフと同じようなグラフを描いている。一方でパラメータを 0 に近づける項が $1/160$ のときには、学習した棋譜との一致率で文献 [3] でのパラメータを 0 に近づける項がある場合の目的関数を使用したときの学習した棋譜との一致率のグラフと同じようなグラフを描いている。これらのことから学習に使用しなかった棋譜で一致率を出せば、同様のことが起こることが予想される。またパラメータを 0 に近づける項の値が $1/16$, $3/64$, $1/32$ と大きいときにも、学習回数を増やすたびに、学習した棋譜との一致率が下降してしまうという問題も図 1 から読み取れる。これらのことからパラメータを 0 に近づける項の値を変えるだけでは、元々の Bonanza の拘束条件の良さを超えることが難しいと考えられる。

4.2.2 評価関数の重みの絶対値の平均

評価関数の重みの絶対値の平均のグラフでは、横軸を Iteration (学習回数) として、縦軸を average (平均の重み) とした。評価関数の重みの絶対値の平均のグラフを図 2 に示す。図 2 を見れば分かるように拘束条件であるパラ

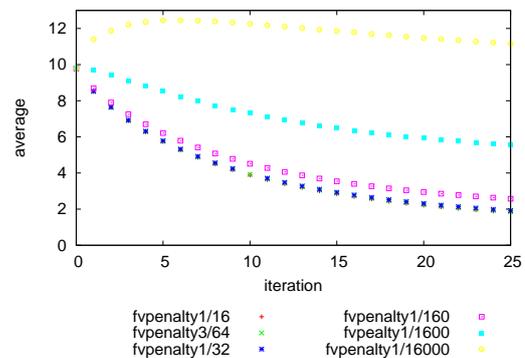


図 2 重みの絶対値の平均の変化

メータの重みを 0 に近づける項の値が、 $1/16000$, $1/1600$, $1/160$, $1/32$ と大きいほど学習回数を増やせば、重みの絶対値の平均を示す average の値が下降し、パラメータの重みを 0 に近づける項の値が $1/32$ 以上大きい $3/64$, $1/16$ になると $1/32$ のときと全く同じグラフを描くことが分かる。これは式 (3) による影響だということが分かる。なぜならパラメータの重みを 0 に近づける項の値が大きいということは、それだけ w の値を小さくしようとするので結果的に平均は小さくなるからである。

4.2.3 駒の価値

駒の価値について、大駒（飛車、角、龍、馬）と小駒（歩、香車、金、銀、桂馬）に関しては、どのパラメータの重みを0に近づける項の値でもあまり変化が見られなかった。なので、ここでは成駒（と、成香、成桂、成銀）とこれらの成駒（と、成香、成桂、成銀）と同じ動きをする金を加えたグラフで違いを見ていく。図3、図4、図5、図6、図7、図8は、パラメータの重みを0に近づける項の初期値である1/160を1/16000、1/1600、1/160、2/64、3/64、4/64と設定した場合の成駒（と、成香、成桂、成銀）と金それぞれの価値の違いを表しているグラフである。グラフを見れば分かるように最初に一番成駒（と、成香、成桂、成銀）の中で価値の高かったとが、図5、図6、図7、図8では、他の成駒（成香、成桂、成銀）の価値よりも低くなってしまっている。これは学習した棋譜の中に、成香や成桂や成銀の出現頻度が少ないからだと考えられる。これらに対して、図3の成駒（と、成香、成桂、成銀）の価値は安定している。この結果から、拘束条件が強いと駒の価値をうまく学習できていないのではないかと考えた。また学習した棋譜との一致率に関して、拘束条件が弱いと過学習のような状態になってしまう問題や拘束条件が強いと学習した棋譜との一致率が低くなる問題がある。そこで、過学習も起こさず、駒の価値もうまく学習するために式(4)を導入した式(5)の目的関数を用いて、次のような実験を行った。

4.3 元の Bonanza の目的関数に既存のパラメータに近づく項を加えた新たな目的関数による制御

Bonanza に文献 [4] で提案された手法を応用し式(1)に式(4)を導入して目的関数を式(5)とし、 w_0 を元の Bonanza (version 6.0) に配布されている fv.bin として、抑制パラメータ C の値を変更して、 $J_C(w)$ と $J_R(w)$ による制御と同様にパラメータの重みを0に近づける項の初期値である1/160を1/16000、1/1600、1/160、2/64、3/64、4/64と設定した場合の学習の経過について結果を出した。 $J_C(w)$ と $J_R(w)$ だけ拘束条件としたときには、パラメータの重みを0に近づける項が4/64のときに成駒（と、成香、成桂、成銀）の価値の変化が著しかった。なのでここでは、パラメータの重みを0に近づける項が4/64として抑制パラメータ C の値を変化させて成駒（と、成香、成桂、成銀）と同じ動きをする金を加えたグラフから成駒（と、成香、成桂、成銀）の価値の観察を行った。その結果、パラメータの重みを0に近づける項が4/64でも抑制パラメータ C が0.005のときには、駒の価値が水平で全く変化しなかった。実際に図9では、グラフが水平で駒の価値に変化がない。

ここで示すグラフ以外にも、学習した棋譜との一致率のグラフで同様に水平なグラフになった。これは、抑制パラメータ C の値が大きいと式(4)で示したように w_0 の既存パラメータに w を近づけようと拘束条件が強く働くために

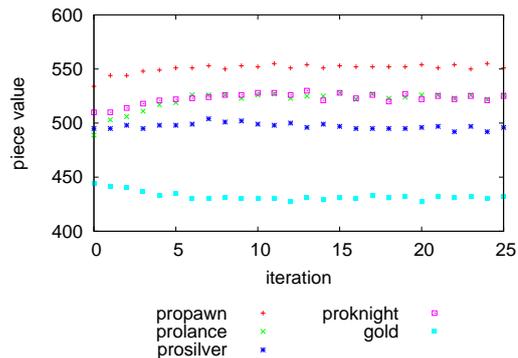


図3 駒の価値の変化 (パラメータの重みを0に近づける項が1/16000)

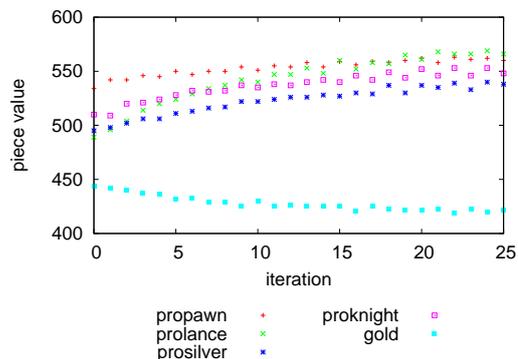


図4 駒の価値の変化 (パラメータの重みを0に近づける項が1/1600)

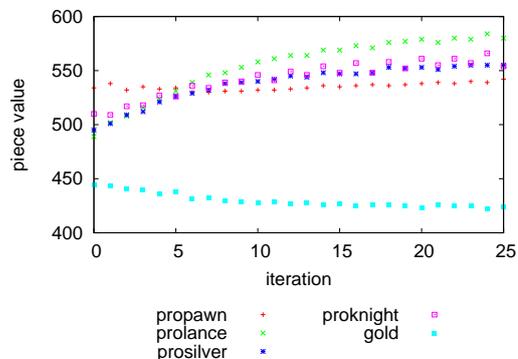


図5 駒の価値の変化 (パラメータの重みを0に近づける項が1/160)

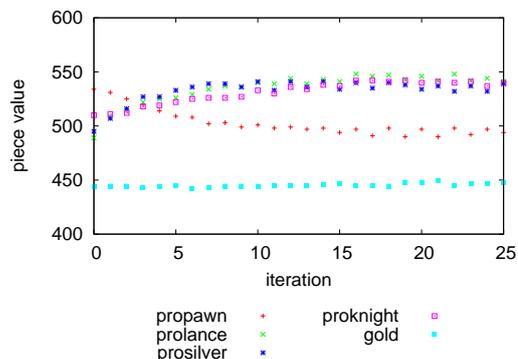


図6 駒の価値の変化 (パラメータの重みを0に近づける項が2/64)

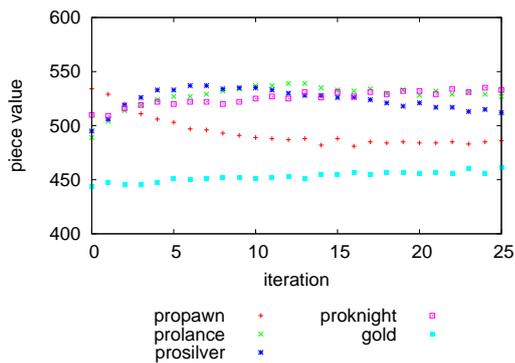


図 7 駒の価値の変化 (パラメータの重みを 0 に近づける項が 3/64)

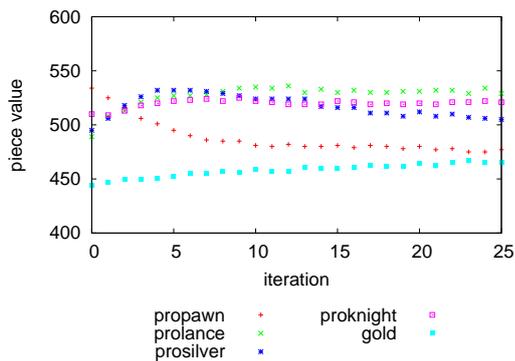


図 8 駒の価値の変化 (パラメータの重みを 0 に近づける項が 4/64)

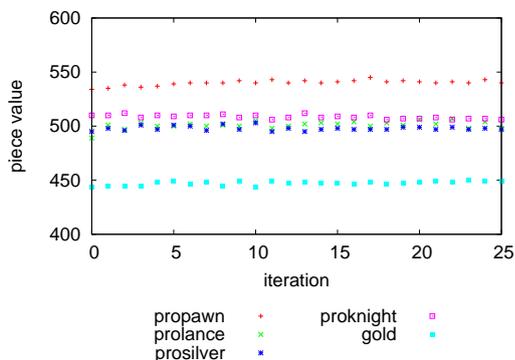


図 9 駒の価値の変化 ($C = 0.005$)

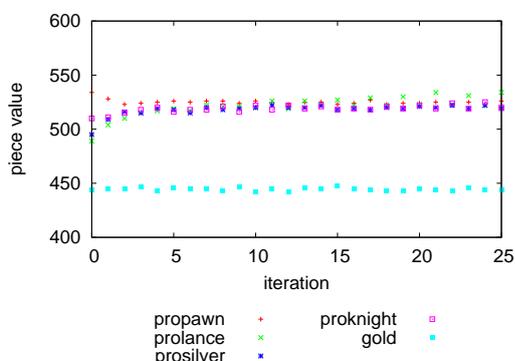


図 10 駒の価値の変化 ($C = 0.001$)

起こる。実際に抑制パラメータ C が 0.005 以上のときには、同様の結果になる。一方で、抑制パラメータ C を 0.001 で

パラメータを 0 に近づける項を同様に 4/64 としたときには、図 10 を見れば分かるように、との価値を他の成駒 (成香, 成桂, 成銀) が追い抜こうとしている。また学習した棋譜との一致率のグラフについてもほんの少し変化があった。

抑制パラメータ C を 0.001 から小さくしていくと徐々にパラメータの重みを 0 に近づける項だけを拘束条件に用いたときと同じグラフになっていった。学習した棋譜との一致率での図 11 や駒の価値に関する実験の図 12, 図 13, 図 14, 図 15, 図 16 で示す。

4.3.1 学習した棋譜との一致率

$J_C(w)$ と $J_R(w)$ を拘束条件としたときには、パラメータを 0 に近づける項を 1/16000 としたときに学習した棋譜との一致率のグラフの変化が著しかったので、パラメータを 0 に近づける項を 1/16000 と固定する。抑制パラメータ C を 0, 0.00000005, 0.0000005, 0.000005, 0.0005, 0.005 としたときのグラフを図 11 に示す。図 11 を見れば分

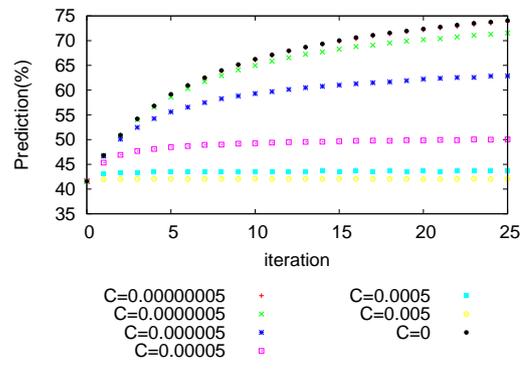


図 11 一致率の変化

かるように一致率の値が既存のパラメータに近づけようとする項の影響で小さくなるのが分かる。このこと $J_C(w)$ と $J_R(w)$ を拘束条件として行った学習の際に起こっていた過学習を防ぐことができると考えられる。しかし抑制パラメータ C の値が 0.005 より大きいときには、初期のパラメータと全く一緒のものになってしまうという欠点がある。さらに、抑制パラメータ C の値が 0.00000005 より小さいとパラメータを 0 に近づける項だけで学習したときと同じ結果になってしまう。だから、既存のパラメータの影響をうまく考えて抑制パラメータの値を扱わなければならない。

4.3.2 駒の価値

$J_C(w)$ と $J_R(w)$ を拘束条件としたとき同様、大駒 (飛車, 角, 龍, 馬) と小駒 (歩, 香車, 金, 銀, 桂馬) に関して目立った変化はなかったので、成駒 (と, 成香, 成桂, 成銀) とこれらの成駒 (と, 成香, 成桂, 成銀) と同じ動きをする金を加えたグラフで違いを見ていく。 $J_C(w)$ と $J_R(w)$ を拘束条件としたときに、パラメータの重みを 0 に近づける項を 4/64 としたときの成駒 (と, 成香, 成桂, 成銀) と同じ動きをする金を加えたグラフの変化が著しかったので、パラメータの重

みを 0 に近づける項を $4/64$ と固定する。さらに抑制パラメータ C の値を 0.00000005 , 0.0000005 , 0.00005 , 0.0005 , 0.005 として, 図 12, 図 13, 図 14, 図 15, 図 16 に示した。駒の価値のグラフでは, 図 12, 図 13, 図 14, 図 15 が似たようなグラフになった。これらの図は, $J_C(w)$ と $J_R(w)$ を拘束条件としたときにパラメータの重みを 0 に近づける項を $4/64$ として駒の価値のグラフを描いた図 8 と似たようなグラフである。このことからパラメータの重みを 0 に近づける項が大きいと既存のパラメータに近づけようとする項があまり機能しないことが分かる。パラメータの重みを 0 に近づける項を $4/64$ として元々の駒の価値を維持しながら学習しようとしても, 既存のパラメータに近づけようとする項を大きくするしかなく, 結局初期のパラメータと一致率, 駒の価値など何もかも同じになってしまう。

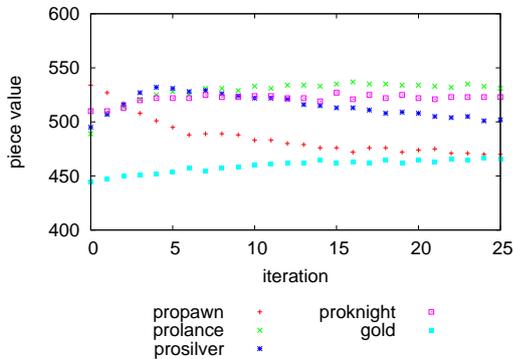


図 12 駒の価値の変化 ($C = 0.00000005$)

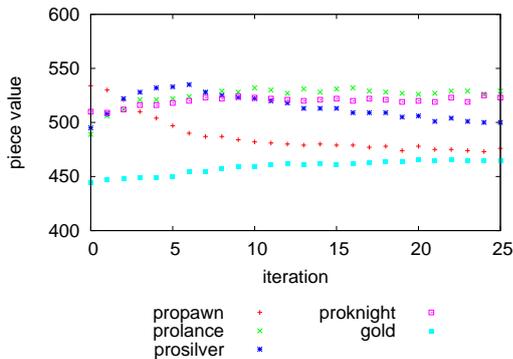


図 13 駒の価値の変化 ($C = 0.00000005$)

4.3.3 2 種類のパラメータの相互作用

既存のパラメータに近づけようとする項である式 (4) の学習をした重みのパラメータと既存のパラメータの差を表した $\|w - w_0\|^2$ の値を学習回数を増やした時に抑制パラメータ C の値によってどうなるかを図 18 と図 19 に示した。図 18 は, 一致率の変化の図を示すときに扱ったパラメータの重みを 0 に近づける項を $1/16000$ としたときの抑制パラメータ C についてのパラメータの差をとったものである。図 19 は, 成駒の価値の図を示すときに扱ったパラ

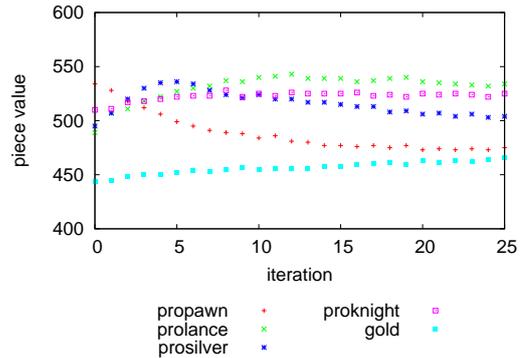


図 14 駒の価値の変化 ($C = 0.0000005$)

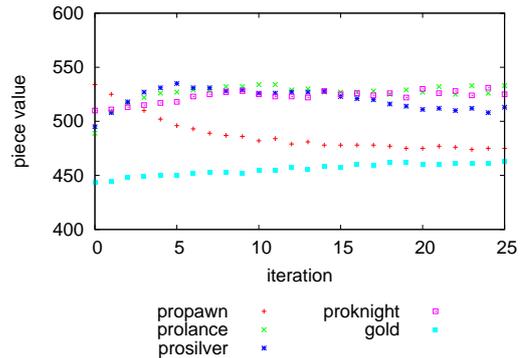


図 15 駒の価値の変化 ($C = 0.000005$)

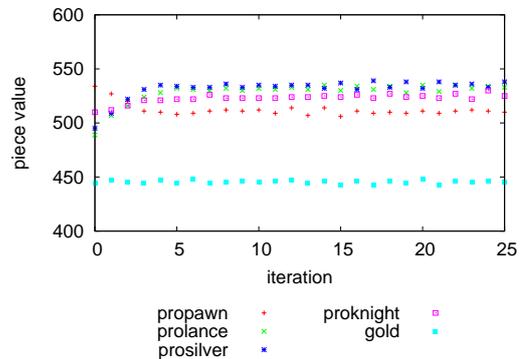


図 16 駒の価値の変化 ($C = 0.0005$)

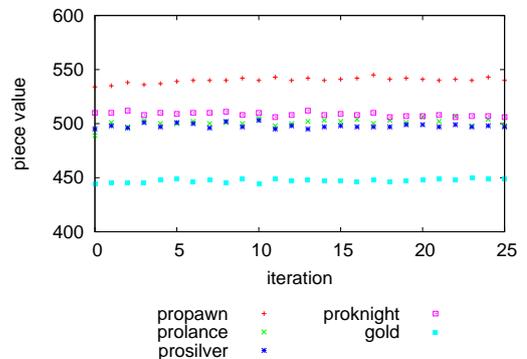


図 17 駒の価値の変化 ($C = 0.005$)

メータの重みを 0 に近づける項を $4/64$ としたときの抑制パラメータ C についてのパラメータの差をとったものであ

る。図 18 と図 19 を比べると抑制パラメータ C の値が同じでも、パラメータの差の値は大きく異なることになることが分かる。これは、図 18 では、パラメータの重みを 0 に近づける項が $1/1600$ と小さいため目的関数全体への影響が小さくなり一方で、既存のパラメータに近づけようとする項の影響が大きくなっているからと考えられる。だから図 19 においても同様に考えて、今度はパラメータを 0 に近づける項が $4/64$ と大きいため目的関数全体への影響が大きくなり一方で、既存のパラメータに近づけようとする項の影響が小さくなっているからと考えられる。また元々のパラメータよりも勝率が高いものを得るならば、2 種類のパラメータの範囲が 0 以外のもので強くなっているものを得たい。そのためには、駒の価値の判断が正確で未知の棋譜つまり学習していない棋譜の一致率が高いものを得たい。図 18 では、抑制パラメータ $C=0.00000005$ のときと 0.000005 のときにその可能性がある。また図 19 では、 $C=0.00005$ のときにその可能性がある。実際に対戦をさせ、勝率などから強さを判断する必要がある。

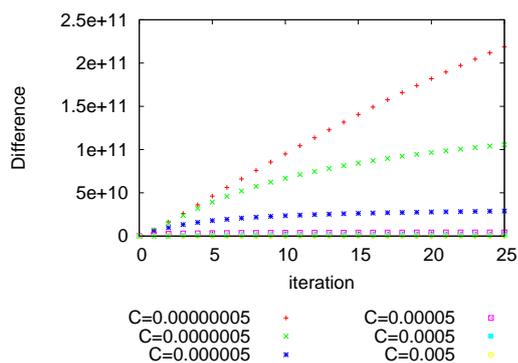


図 18 2 種類のパラメータの相互作用 (0 に近づける項 $1/1600$ のとき)

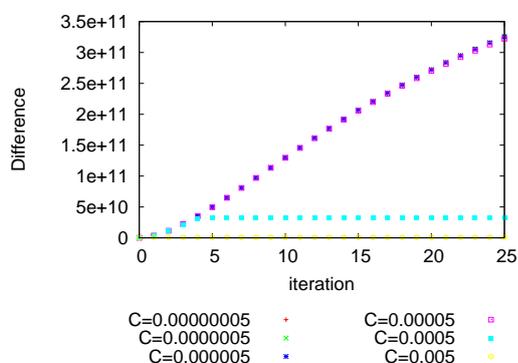


図 19 2 種類のパラメータの相互作用 (パラメータを 0 に近づける項 $4/64$ のとき)

5. まとめ

今回の実験から、拘束条件の違いによって成駒 (と, 成香, 成桂, 成銀) の価値や学習した棋譜との一致率に大きな違いが出るのが分かった。またその際に新たに既存のパラメータに近づける拘束条件を追加して、成駒 (と, 成香, 成桂, 成銀) の価値や学習した棋譜との一致率を調整できることが分かった。特に既存のパラメータに近づける拘束条件内の抑制パラメータ C をどう取れば、効果がどう現れるのかということを示した。パラメータの重みを 0 に近づける項が $1/1600$ のときには、抑制パラメータ C が 0.005 より大きいときには、既存のパラメータの値の一致率、駒の価値どちらにおいても学習回数を増やしても同じ値を取り続ける。また抑制パラメータ C が 0.00000005 より小さいときについて、 $J_C(w)$ と $J_R(w)$ を拘束条件としたときに、パラメータの重みを 0 に近づける項が $1/1600$ のときの一致率と同じ値になり続ける。これらの違いは、パラメータの重みを 0 に近づけようとする項と既存のパラメータに近づけようとする項どちらの影響をより強くするかということに関係している。2 種類のパラメータの相互作用の実験で、駒の価値や学習した棋譜との一致率から良い値なのではないかと推測していたが、実際に対局をして勝率を測ったり、学習に使用しなかった棋譜との一致率から求める必要がある。今後は、今回の実験の結果を元に、勝率やテストデータの一致率も合わせて最適な拘束条件を見つけ出し、示す必要がある。また今回は学習回数が 25 回と少ないので、もっと学習回数を増やして同様の実験を行う必要がある。

参考文献

- [1] 滝瀬竜司, 田中哲朗: 入玉指向の将棋プログラムの作成, 第 16 回ゲームプログラミングワークショップ 2011, pp. 25-31 (2011).
- [2] 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).
- [3] K. Hoki, T. Kaneko: Large-Scale Optimization for Evaluation Functions with Minimax Search, JAIR, Vol. 49, pp. 527-568 (2014).
- [4] 矢野友貴, 三輪誠, 横山大作, 近山隆: 既存評価関数のパラメー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, pp. 1-8 (2009).