情報処理学会論文誌 Vol.57 No.11 2382–2391 (Nov. 2016)

教師データが不足した環境での機械学習結果改善手法

金澤 裕治[1,a)]

受付日 2016年2月19日, 採録日 2016年9月6日

概要：厳密解を求めるのが困難でヒューリスティクスによって解かれている問題で，計算機が熟練者を上回ることが困難なものが存在する．そのような問題において，ヒューリスティクス手法を多数のパラメータで制御できるようにしておき，そのパラメータを機械学習によりチューニングすることで，熟練者の判断を再現できれば，解法の性能向上が期待できる．そのために解決しなければならない課題の 1 つが，教師データの不足である．本論文では，教師データが不足した環境で学習結果に含まれる誤りを改善する強化学習類似手法を提案する．提案手法を将棋プログラム Bonanza 6.0 の機械学習テーブル改善に適用し，1 回の適用でイロレーティングが平均 25 程度，繰り返し適用することで，最終的には 150 程度向上した．

キーワード：教師あり機械学習，ヒューリスティクス，将棋

Refinement of Machine Learning Results Generated from Insufficient Sample Data

Yuzi Kanazawa[1,a)]

Received: February 19, 2016, Accepted: September 6, 2016

Abstract: There are some problems where human experts can produce better result than heuristics methods on computers. Performance of such heuristics methods may be improved significantly by machine learning on result by human experts. An issue that must be solved to make it possible is sample data shortage. This paper proposes a reinforcement-learning–like method to fix errors in machine learning result generated from insufficient sample data. The method was applied to refine parameters used by the shogi program Bonanza 6.0. Experimental results show that Elo rating of Bonanza 6.0 with refined parameters was improved by 150 points.

Keywords: supervised machine learning, heuristics, shogi

1. はじめに

厳密解を求めるのが困難な組合せ最適化問題では，ヒューリスティクスを用いた解法を採用することが多い．それらの解法では，問題を解く過程をいくつかの段階に分解し，なんらかの評価値を定義して，次の段階に移るための簡単な最適化問題を解くことを繰り返すようなものが使われるものが多い（**図 1**）．たとえば，LSI 配置問題は，論理素子とそれをつなぐネット，配置対象の面と面積制約を与えられて，ネットが使用する配線リソース量を最小にするような論理素子の位置を決定するものであるが，これを厳密に解くことは困難であるため，Min-cut 配置手法 [1] のようなヒューリスティクスを使用する．これは，**図 2** のように領域を分割しながら，論理素子を分割した領域のどちらかに割り当てる処理を繰り返すもので，各分割の際に，面積制約を満たしながら領域をまたぐネット数が最小になるようなグループ分けを選択する．結果的に，最終的なネットの長さが小さい配置を得ることを期待するものである．最初に述べたヒューリスティクスの一般論にあてはめると，領域分割が次の段階に移るための操作で，分割されるネットの本数を評価値として最適化を行うことに相当している．

人間の熟練者の判断による結果がこのような解法による結果を上回る場合が存在する．上にあげた単純な評価値を

[1] 株式会社富士通研究所
Fujitsu Laboratories Ltd., Kawasaki, Kanagawa 211–8588, Japan
[a)] ykanazawa@jp.fujitsu.com

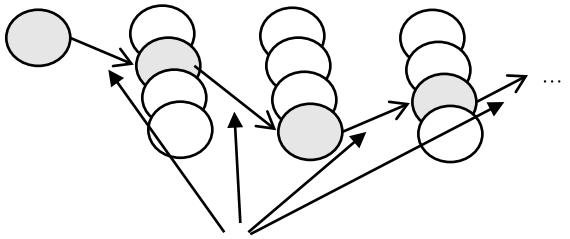


図 1 ヒューリスティクスによる組合せ最適化問題の解法モデル
Fig. 1 Model of heuristics technique to solve combinatorial optimization problem.

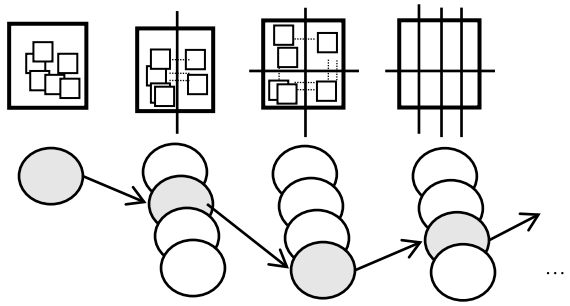


図 2 ヒューリスティクスによる解法の例：Min-cut 配置手法
Fig. 2 Example of heuristics technique: Min-cut method of LSI placement.

使用している配置手法では，一部に配線が集中する領域が発生し，迂回が必要になって配線リソースを限度以上に消費することも多い．回路の規模や複雑度によるが，回路の規則性や全体の構成に関する情報が利用できるときは，熟練設計者は自動配置では達成できない配線性の良い配置を作ることができる．アルゴリズムを多数のパラメータから構成された中間評価値により制御できるようにしておき，熟練者の判断を再現するようにパラメータをチューニングする手法を採用できれば，自動配置の性能を高めることが可能となるはずである．しかし，チューニング作業がネックとなり，これまではそのような手法を採用することが困難であった．

近年の計算機能力と機械学習技術の向上により，チューニング作業を自動化することで，ヒューリスティクスによる解法の大幅な強化が可能になりつつある．そのために解決しなければならない課題の 1 つが教師データの不足である．熟練者による設計結果を教師データにしようとしても，最終的な設計結果として残されているものの数は，機械学習が必要とする数と比較して多くはない．これは，機械学習用のパラメータを増加させると，組合せの数が指数関数的に増えるのに対し，熟練者による設計結果は線形でしか増えないためである．このため，単純な機械学習結果を使うだけでは，様々な条件に対応して高い精度を持つ中間評価値を作ることは難しい．

教師データが不足する状況自体を変えることは困難なので，対象としている問題の性質を抽出し，境界条件として使うことで学習結果を改善する技術が必要になる．

本論文では，中間評価値に基づいたヒューリスティクス解法において教師データ不足を補う手法についての仮説と，それに基づく手法の提案，さらに提案する手法のコンピュータ将棋への適用例について述べる．将棋は 2 名のプレイヤーが交互に指すという特徴があるため，最適化問題とは探索が異なる面はあるが，真の最適解を見つけるのが非常に困難であるため，中間評価値を使って次の状態を選択する手続きを繰り返すという特徴は共通している．また，ルールが単純であることや，プロ棋士によって作られた棋譜を入手可能であること，優れた将棋ソフトであり機械学習を用いた評価値生成を行う Bonanza 6.0 のソースコードが公開されていることなど，手法の評価に都合のよい点があるため採用している．

2. 対象とする探索問題

2.1 定義

本論文では以下の特徴を持つ解法を対象としている．

- 解を得る手続きを複数の段階に分割している．
- 中間評価値を最適化することで次の段階の状態を選択する．
- 処理の初期において最終的に「良い状態」に到達できる可能性を高くするような選択を熟練者は行うことができる．
- 熟練者による判断を反映した中間評価値を教師あり機械学習によって求めている．
- 教師データの不足により評価値に誤りが含まれている場合があり，そのような評価値が得られる局面から処理を進めると期待した結果が得られないことがある．

2.2 Bonanza 6.0

本節では，Bonanza 6.0 [2] の評価関数の構成とその学習手法について簡単に述べる．

Bonanza 6.0 では，将棋のある局面に対する評価値を，駒得と 3 駒関係と呼ばれる手法によって計算する．

駒得は先手後手が持つ各駒のポイントを合計したもので，任意の局面で容易に計算でき，局面の評価値としては，ある程度信頼できるという特徴がある．しかし，詳細な評価を得ようとすると，駒得に加えて駒の配置を考慮する必要がある．配置に基づいた評価値は 3 駒関係によって求められる．

3 駒関係は，さらに KPP と KKP と呼ばれる 2 種類のパラメータ群に分類される．KPP とは玉（King）が存在する位置と他の駒（Piece）2 枚の種類・位置の組合せのことである（持ち駒は枚数ごとに別の駒と見なして扱う）．玉と他の駒 1 枚の組合せは，KPP で 2 つの P が同じ場合とし

て表現される．KKP は同様に，自玉，敵玉の位置と他の駒一枚の種類・位置の組合せになる．すべての KPP，KKP の組合せに対してポイントを格納したテーブルがメモリ上に存在し，ある局面の評価値は，盤上に存在するすべての KPP，KKP の組合せに対して，テーブル上の値の和をとることで求められる[*1]（**図 3**）．

駒得計算における各駒の価値と 3 駒関係のテーブルの値は以下のような機械学習手法によって決定されている．プロ棋士が採用した手と，採用しなかった手を指した後，それぞれの局面から 2〜3 手探索した結果を求める．プロ棋士の指した手から到達する局面と，それ以外の手から到達する局面の評価値の差が一定値 T 以上つくように，駒得や 3 駒関係の値を求める．これは，プロ棋士の指した手から到達した局面の評価値を Vpro，それ以外の手から到達した局面の評価値を Vnonpro としたとき，$\mathrm{Vpro} - \mathrm{Vnonpro} < \mathrm{T}$ の場合をエラーと見なし，エラーの値を最小化することで求められる．最適化においては，すべてのプロ棋士が指した手/それ以外の手のペアから得られるエラーの合計値に過学習を防ぐための正則化項を加えたものを最適化の目的関数として，この値を最小にするような各駒の価値，KPP，KKP の値を最急降下法で求める．正則化項に関しては，Bonanza 6.0 では一次の正則化項を使用している．この手法は，細かい点で様々な違いはあるものの，評価関数が一次式で構成されていることもあって，サポートベクタマシン [3] による学習手法に類似している．

仮に理想的な評価値が得られるなら，一手先の評価値を最大にするような手を選択すればいいということになるが，駒得と 3 駒関係から得られた評価値ではそこまで正確ではないため，Bonanza 6.0 は，$\alpha\beta$ 探索手法によって，各プレイヤーが互いに現在の局面の n 手先の局面で自分の評価値を最大にする処理を n を増加させながら繰り返すことで次の指し手を決定する．この詳細は本論文で扱う範囲を越えているのでここでは述べない．興味がある読者は [4] を参照されたい．

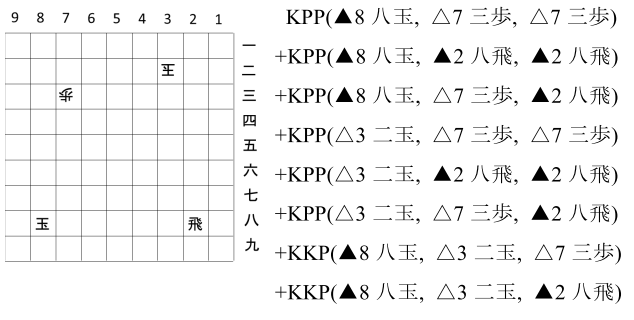


図 3 3 駒関係の評価値計算

Fig. 3 Board evaluation based on KPP (king/piece/piece)/KKP (king/king/piece).

[*1] Bonanza の実装では，先手後手を反転させたときに評価値の正負が反転するという性質を利用してテーブルを圧縮しているため，実際の計算は，ここに書かれたものより多少複雑になる．

3. 課題

3.1 教師データの不足

前節で述べた手法において，問題となるのは教師データの不足である．たとえば，1 章で述べた LSI 配置問題では，実際の設計例を教師データとすることは考えられるが，最終的な設計物として残っているものを数千件以上入手するのは困難である．

コンピュータ将棋においても同様の問題が存在する．Bonanza ではプロ棋士の指し手を教師データとして使用しているが，プロ棋士による公式戦の棋譜数は 5 万局程度である．一局につき百数十手の指し手があり，各指し手ごとに数十手以上の合法手があるため，教師データの数は数億程度あるが，初期局面が同一であるなどの条件があるため，かなりの部分が重複している．Bonanza 6.0 の 3 駒関係のパラメータは先手後手と左右の対称性を考慮して圧縮しても 4,500 万以上あり[*2]，中盤以降の局面の中には教師データが不足する局面が存在すると推測できる．その局面に関しては，得られたパラメータに不適切な値が残っていることになる．

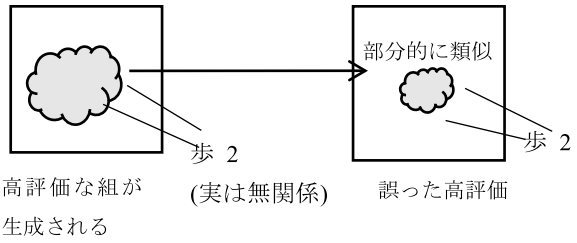
教師データが不足しているため生じる誤差の例としては以下のようなものが考えられる．

(1) 高い評価値を生み出す配置の範囲の認識エラー
教師データを通じて，対局者がある駒の配置を選択することが分かったとき，そこに共通する配置を選択の原因と判断して高い評価値がつくことになる．教師データが少ないと，このグループの範囲に誤りが生じる（**図 4** (a)）．たとえば，対局で先手がある配置を選ぶとき，教師データの中では偶然常に先手が歩 2 枚を持っていたとすると，この持ち歩が実際には戦局に関係なかったとしても，盤上の各駒の配置と 2 枚の持ち歩の組合せに高い評価値がつくことになる．対局中に持ち歩 2 枚と当該配置の一部が出てきたとき，誤った評価値が得られてしまうことになる．

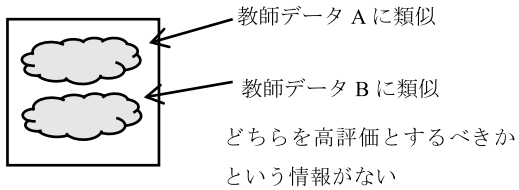
(2) 2 つ以上の教師データの混合した局面での評価値バランスエラー
ある局面の敵陣内の配置が教師データ中のある対局に類似し，自陣内の配置が別の対局に類似していたとしよう（図 4 (b)）．両者の評価値は別々に求められたものであるため，評価値を比較した結果は正しくない可能性がある．評価値の低い方の配置の方が，実は重要だった場合，優勢の判断を誤ることになる．

上にあげた例に限らず，教師データが不足している状況では，誤った評価値がついている局面は多数生じているものと考えられる．このような誤りを減少させることができれば，評価値の正確性を向上することが可能になる．

[*2] Bonanza 6.0 のテーブルでは左右対称の圧縮は行っていないため，格納されているパラメータ数は 9,000 万程度になる．



(a) 実際には有利不利に無関係でも教師データに存在した駒を含むペアに高得点



(b) 評価値のバランスの誤り

図 4 教師データ不足による評価値エラーの例

Fig. 4 Example of evaluation error caused by insufficient samples.

4. 提案手法

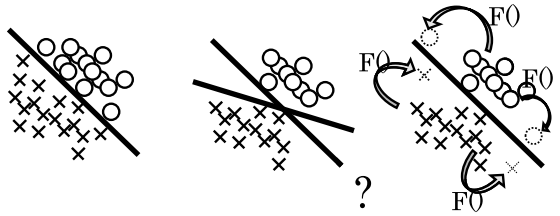
4.1 仮説

本節では，将棋という具体例から一度離れて，機械学習の理論から見た教師データ不足について考えてみることにする．

教師あり機械学習は，与えられた教師データに基づいて，未知の入力に対する答えを生成することを目的としている．教師データが不足する場合，機械学習の理論としては，対象とする問題の性質を前提にすることはできないので，教師データが与えられていない点の評価値としては何も仮定せず，なるべく偏りのない答えを返すようになっている．

単純な例として，2 種類の特徴量に基づいたサンプルを特徴量空間上の直線でクラス分けする場合を考える．教師データが十分あれば，クラスを分ける直線はほとんど誤差なく求めることができる（**図 5** (a)）．教師データが少ない場合，教師データを分割するような直線は多数存在するため，その中から最もらしいものを 1 つ選ぶ必要がある（図 5 (b)）．サポートベクタマシンの場合は，直線とそれに最も近い教師データとの距離が最大になるように選択する．

機械学習の理論としては対象とする問題の性質に依存しないことが要求されるので，そのような手法を取るわけだが，現実の問題では，対象とする問題が持つ制約条件を使って結果を改善できるなら，それでもかまわない．たとえば，問題を解析すると，なんらかの変換 F() が存在し，与えられた教師データに基づいて新規の点における分類が判断できるものを得ることができる場合がある．そのような変換 F() を求めることができれば，より正確な境界面を求めることが期待できる（図 5 (c)）．



(a) 教師データが十分存在する場合　(b) 教師データが十分存在しない場合　(c) 教師データが生成できる場合

図 5 2 次元の特徴量空間でのクラス分けの例

Fig. 5 Example of classifying in 2D feature space.

ここで変換 F() が，個別の問題を詳細に解析しないと得られないものでは，それを見つけるのに手間がかかってしまうので，できるだけ普遍性のあるものが欲しい．そんな都合のよい変換があるのかというと，我々は 1 つ候補を知っている．2.1 節で述べた「次の段階の状態を選択する手続き」である．

次の段階の状態を選択する手続き（ここでは SelectNext() と呼ぶことにする）を使って得られた状態の評価値と，現在の状態の評価値が満たすべき性質を考えてみよう．評価値の判断に従って SelectNext() によって次の段階に進めた結果，実は評価が悪くなってしまったという場合，最初の評価値には問題があり，悪い評価値を返すべきだったと判断できる．これは，3 章で述べたコンピュータ将棋で評価値を誤るケースで考えてみると，誤った評価値がついた局面から数手進めてみた結果，実際には駒得や，他の高評価な局面につながらないことが判明することに相当する．この場合，最初の状態での評価値を下げる必要があるということが分かる．そのような変更を加えた結果，評価値が正しい部分まで変化しないようにする必要があるが，これは SelectNext() 後に評価値が変動しなかった部分も含めて多数の状態に対する評価値を計算することで実現可能である．

以上の考え方を推し進めると，様々な状態に対して SelectNext() を適用し，それによって得られた状態の評価値を理想の評価値と考え，元の状態の評価値がその値になるように学習しなおすことで，教師データが不足しているために誤差がある機械学習結果を改善することができるのではないかという仮説が成立する．あとは，何をすれば「元の状態の評価値がその値になるように学習」できるかが問題だが，これは回帰分析を実行すれば良い．

さらに，与えられた局面と一部が異なる局面を生成し，同様に SelectNext() を適用することで，図 4 (a) で述べたような評価値の誤りがあったときに，評価値の変動に関係がある配置の範囲を確定し，正確に修正することが可能になる．一部だけ異なる局面は，将棋の場合は一手前の局面

から実戦では指されなかった手を適用して駒を動かした局面を作ることで得られる．

4.2 強化学習との類似性

4.1 節で述べた手法は SelectNext() を適用して得た評価値を利得と考えれば強化学習 [5] と同じ形式になっている．実際に，強化学習の効果の一部に，教師データ不足を補う効果が含まれている可能性はあると思われる．

しかし，本手法が目標とするのは与えられた教師データをよく反映した学習結果の生成であり，強化学習というよりは，教師あり機械学習の改善手法であると筆者は考えている．この違いのため，強化学習では考えにくい実装を採用することになる．

たとえば，提案手法においては，学習で参照するのは，現在の局面から 1 段階から 3 段階程度進めた局面の評価値としており，実際に評価値どおりの結果に帰着したかどうかを確認していない．評価値の誤った部分を直そうとしているのに，その評価値に基づいた学習を実行していいのかという疑問が生じるが，その背景には前節で述べた仮説が存在している．また，提案手法では，読みを浅くして実戦では使用できないような状態で探索させた結果を積極的に使うことで，教師データが不足している部分を改善する効果が期待できる．このような実装は強化学習の効果を期待する場合は採用しにくい．

一方，提案手法では，与えられた教師データに含まれていなかった情報を学習することは期待していない．したがって，提案手法を使用して与えられた教師データの範囲内で機械学習結果を改善した後で，本来の強化学習を適用することを考慮するのが良いのではないかと考えられる．そのためにも，両者の効果を区別しておくことは重要と考える．

4.3 提案手法の特徴と既存手法との比較

提案手法の処理を以下に記述する．

(1) 機械学習で得られたパラメータを入力とする．
(2) 自己対戦その他の手段により，評価値が一定値以上変動した局面を選択し，解析対象として現在の変数テーブルを生成するのに使用した教師データに追加する．
(3) 教師データ内の局面に対し，現在の評価値に従って d 手先まで最善手によって得られた局面の評価値を計算し，それを現在の局面から静止探索[*3]で到達した局面の理想の評価値とする（**図 6**）．
(4) 選択した局面の直前の局面において全合法手を指すことで 1 手分だけ異なる局面を生成し（n 個の合法手があった場合は n 局面生成される），各局面に対して，(3) と同様に理想の評価値を求める（ここでは (3) と (4) を区別して別に書いているが，与えられた棋譜に現れた局面に対して全合法手を生成し，その先の局面に対して理想の評価値を求めればよい．実際に指した手は合法手の中に入っているため，（棋譜の最終手を除いて）同じ結果になる）．
(5) 各局面に対して，理想の評価値に近い評価値を返すようなパラメータを回帰分析によって求める．
(6) 新規に得られたパラメータを自己対戦により検証し，勝率が向上していなかったら終了，向上していた場合はそれを採用して (2) に戻る．

[*3] 当たりになっている駒を取り合って安定した局面に到達する探索．

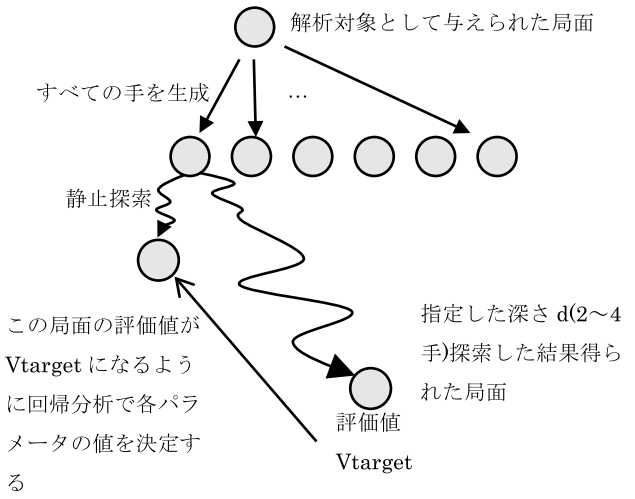


図 6 提案手法における目標評価値の求め方
Fig. 6 Method to determine target evaluation value in proposed method.

類似した手法を使う既存技術として，数手先まで探索した結果を何らかの形で学習に使用する手法は，文献 [6], [7] など，複数の提案が存在する．文献 [6] は，数手先の評価値と現在の局面の評価値の差の自乗の和を最急降下法で最小化しようとするものであった．文献 [7] は，ステージごとに終盤から逆に学習を進めるため，理想の評価値は固定されるという点で提案手法と類似している．教師データ不足を補うという観点から作られたものではないため，1 手異なる手を指した局面を対象にするという手法は採用していない．自己対戦結果を学習に使用するものとしては文献 [8] があり，熟練者による棋譜に自己対戦で得られた棋譜を追加し，学習する手法について論じている．

本論文の提案手法では，教師データの不足によって学習結果に含まれている誤りを修正することを指向しており，その結果，与えられた局面と 1 手異なる局面を生成して解析対象とし，得られた解析対象局面に対して理想の評価値を固定して，それに近い評価値を与える変数テーブルを回帰分析によって求めるという処理につながっている．評価値のスナップショットをとることにより，教師データ不足による誤差を解消する能力が向上すると考えた理由は以下のとおりである．

局面 A から数手先を探索し，局面 B に到達した場合を

考えよう．動いたのはたかだか数手なので，局面 B と局面 A の配置には共通する部分が多い．局面 A よりも局面 B の評価値が高かったとする．局面 A から探索した結果，評価値が高い局面 B に到達したということは，局面 A の配置の一部に評価値を増加させるものが存在することを意味する．このとき，評価値増加につながった配置の一部は局面 B でも残っている可能性が高い．局面 A，B のペアだけを見て現在の評価値の差の自乗の和を減らすように最急降下法を行った場合は，共有された部分の変数には影響は発生しないが，提案手法では，評価値を変動させる配置を正確に抽出することを目的として，1 手異なる局面を生成して同じ処理を実行している．これが原因で，評価値増加につながった配置全体が抽出され，それに対応する変数の評価値が上昇することになると考えられる．結果的に，局面 B の評価値も上昇する可能性が高い．局面 A の評価値を修正する過程で，局面 B の評価値が変動し，それに向かって局面 A の評価値を近づける実装を仮に採用したとすると，正のフィードバックが働いて評価値が不安定になるケースが発生する．値が発振しないように制御を加えるような手法を考えるという方向もあるが，評価値増加につながる配置を正確に抽出するという目標を満たしながらそのような制御を実行するのは困難をともなう．局面 B の評価値を最初に求めたら，その値を目標として固定し（処理中に局面 B の評価値がどのように変わっても目標の値は変わらない），局面 A の評価値をそれに近づけるのは単純に実装可能である．

類似した状況は，提案手法を適用する前に学習が完了していた局面でも発生していると考えられる．処理を始める前の時点では，以前の機械学習で教師データとして使用した局面に対して得られる評価値はほぼ正しい値がついていると考えてよいだろう．しかし，教師データが不足している場合においては，評価値として最終的に得られる値は正しくても，それを計算するための変数テーブルの値の一部に間違った値がついていて，他の部分がその誤差を補償することで全体として正しい答えが返るようになっている可能性が考えられる．たとえば，図 4 (b) に示した例の場合，新規に解析対象に追加された局面の評価値が修正されると，学習済み局面の評価値も変動することになる．手法を適用した後に得られた評価値が適切な値のまま維持されていることを保証するために，もともと正しい評価値がついていた局面に対しても，評価値が大きく変動している部分と同様，固定された目標評価値を与えることが望ましい．結果的に，過去に学習に使用した局面を含め，与えられたすべての局面に対して目標評価値を最初に求めて固定し，それに近い評価値を返す変数テーブルを求めることになる．

提案手法において，目標とする評価値を固定するのは，以上述べた評価値の変動から影響を受けることを回避するのが理由である．

以上述べたように，提案手法は，教師データ不足により，機械学習結果が誤っている場合に特化して作られた手法である．したがって，教師データが十分提供されている場合には適用しても効果はなく，既存手法の方が良い成績を収めるものと考えられる．

4.4 手法の詳細について

4.4.1 解析対象とする棋譜

入力とする棋譜に関しては，対局中に評価値が大きく変動するなどの問題があったものを使用することが望ましい．前節で述べたように，正しい評価値がついている部分が誤った値にならないように，初期の変数テーブルを求める際に Bonanza メソッドで使用した棋譜も含める．

自己対戦結果，他プログラムとの対戦結果以外に，棋譜には現れないが探索中に内部的に現れた局面を使うことが考えられる．Bonanza 6.0 の場合は，読みの深さを増加させながら探索する際に，各深さごとに現在の手番のプレイヤーにとって最善の結果となる手の応酬とその評価値がログに出力されるので，それを解析することで，一段深く読んだときに評価値の変動があった部分を抽出することが可能である．現在の局面 A から n 手読んだときに局面 B に到達，n + 1 手読んだときに局面 C に到達したとする．|eval(B) − eval(C)| の値が一定値以上だったとき，B から 1 手指したところと C の 1 手前のどちらも変動が生じている可能性があるので，初手から A を経由して B に到達する棋譜と，初手から A を経由して C に到達する棋譜の両方を生成し，解析対象とする．

4.4.2 回帰分析処理

回帰分析処理は，Bonanza 6.0 のように評価関数が線形の場合は，問題としてはサポートベクタマシンの回帰分析手法である ε-SVR と同じ形式になる．変数とデータが多いため，ε-SVR で一般的に使われる双対問題に変換してラグランジュの未定乗数法で解く手法ではなく，評価関数の各係数を直接最急降下法で求める手法を使用する．

回帰分析を開始する前，目標評価値を求める時点で，入力局面の各子局面から静止探索で到達した局面 Q の評価値を Vcurrent，d 手探索して到達した局面の評価値（目標評価値）を Vtarget とする．回帰分析の処理中は，Vcurrent，Vtarget は変化しない．局面 Q の評価値 x が Vtarget に近づくよう，以下のエラー関数 Error(x) を適用して得られた値を入力として与えられたすべての局面の子局面に対して和をとり，2 次の正則化項を加えた関数 targetFunc を最急降下法で最小化する．

$$\mathrm{Error}(x) = \begin{cases} C(x - V_0)^2 & (x < V_0) \\ 0 & (V_0 \le x \le V_1) \\ C(x - V_1)^2 & (x > V_1) \end{cases}$$

$$V_0 = \min(\mathrm{Vcurrent} + m, \mathrm{Vtarget} - \varepsilon)$$

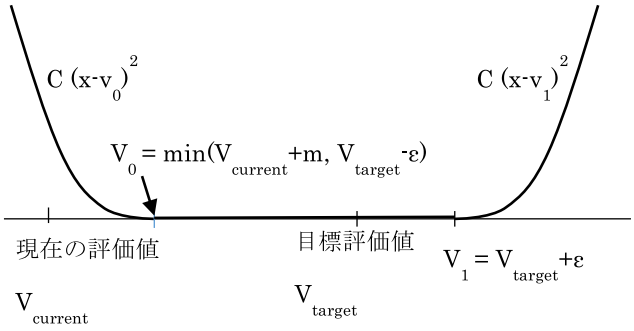


図 7 回帰分析で使用するエラー関数の例
Fig. 7 Error function used for regression of proposed method.

$$V_1 = \max(\mathrm{Vcurrent} - \mathrm{m}, \mathrm{Vtarget} + \varepsilon)$$
$$\mathrm{targetFunc}() = \sum_{Q} Error(eval(Q)) + \sum_{v=kpp,\, kkp} v^2$$

このエラー関数は，詰みが存在するなどの理由で Vcurrent と Vtarget の差が非常に大きい局面が一部にあったときに，それが他の局面に与える影響を一定以下に抑えるため，評価値 x が Vcurrent から一定値以上 Vtarget に近づいたときはエラーの値が 0 になるように作られている．Vtarget>Vcurrent の場合のエラー関数を**図 7** に示す．評価値が m 以上動いたものについてはエラーと見なさないように，目標評価値の周辺のエラーなしと見なす区間を拡大している．

また，目標とは反対方向に評価値が大きく変動するのを防ぐため，通常 ε-SVR は 1 次のエラー関数を使用するところ，2 次式を使用している．

詰み，千日手など特殊なケースへの対応に関しては，詰みは最高限度の評価値が得られたものとして扱っている．静止探索で到達した局面で詰んでいた場合は回帰分析の対象にはしていない．千日手かどうかのチェックは行っていない．

4.4.3 自己対戦による検証

回帰分析のための最急降下法の処理中に，最新のパラメータを一定間隔（1 日 1 回程度）で取り出し，リーグ戦に投入して勝率を検証する．筆者の環境では，この検証は収束判定も兼ねている．収束判定は別の形でもできるが，自己対戦による検証は次の世代で入力として使用する棋譜を生成する手段にもなっているため，このような形を採用している．

5. 実験

本論文の提案手法は，教師データが不足している部分の学習結果の改善に効果があると考えられる．したがって，何回も実行すると，教師データ不足の状態が解消され，しだいに改善効果が弱まっていくことが予想される．教師データが不足している領域がほかに存在するなら，それに対応する棋譜を供給できれば，そこに含まれていた修正が必要な局面の数に応じて改善効果が復活することが予想される．

実験では，Bonanza 6.0 の学習結果に提案手法を繰り返し適用し，レーティングの変化を観察する．目標評価値を求め，修正する局面の供給元として，自己対戦棋譜，インターネットで公開されている対戦システム Floodgate [9] での対戦結果[*4]，さらに，4.4.1 で述べた探索中に現れた局面をログから抽出する手法を順に使用し，レーティング向上効果が見られるかどうかを確認した．

実験で使用したパラメータは，探索深さ d = 3，使用したエラー関数の ε，m の値は，それぞれ $\varepsilon = 16$，m = 528，Error(x) 内の 2 次の正則化項に対する大きさを決める定数 C = 4.0e10 である．最急降下法については，与えられた棋譜の全局面の子局面に対してループしながら目的関数の偏微分を求め，それに比例して各 kpp/kkp パラメータをアップデートする処理を繰り返す素朴なものを使っている．複数の棋譜に同一局面が現れるケースは，Bonanza 6.0 の機械学習処理で使用しているハッシュによるチェックを流用してスキップしている．したがって，全局面をループして偏微分を求めているときに，各局面は 1 回だけ解析されるようになっている．

出発点として使用した学習結果は，Bonanza 6.0 で使用している 1 次の正則化項を 2 次の正則化項に置き換え，正則化項の係数 1.0e-10，プロ棋士の指した手から到達する局面と，それ以外の手から到達する局面の評価値の差として要求される値 T = 256，駒得パラメータは Bonanza 6.0 の学習で使用している初期値，kpp/kkp の初期状態はすべて 0 として得られたものである．

本実験では，提案手法を 9 回適用した．各回で使用した棋譜数とその内訳を**表 1** に示す．

最初に，Bonanza 6.0 の一部として配布されている変数テーブルを使用する original と，前段落で述べた出発点として使用した学習結果 base に対し，主に探索深さ限度 3 として短時間で終わる自己対戦を多数実行して得られた棋譜の局面と，base を生成するのに使用した棋譜を解析対象として，提案手法を適用した．2〜4 回目の適用では，それ以前の適用で得られた変数テーブルと base，original で，4.4.3 で述べた強さ検証のためのリーグ戦（持ち時間 2 分切れ負け）と，深さ限度 3 の自己対戦を行い，その結果の中で，最新版とその直前の版のプログラムが負けた棋譜，あるいは最終的に勝った棋譜であっても，評価値が一次的に −1000 以下になったものを選択して解析対象に追加し，提案手法を繰り返し実行している．このような選択基準を採用したのは，この時点では，解析対象の棋譜が少ないこともあり，評価値の変動を厳密にチェックして選別するよ

[*4] Floodgate でのプログラム名は PuppetMaster，アクセス期間は 2012 年 11 月〜2013 年 4 月である．当時の対戦相手の中でレーティング上位は tsutsukana，maybe_tomorrow，ponanza，Gekisashi，gpsfish などであった．

表 1 各処理で使用した棋譜数[*5]
Table 1 Number of sample matches in each iteration.

	棋譜数	自己対戦 (カッコ内は持ち時間 2 分の対局結果)	Floodgate (カッコ内は自プログラムの対戦数)	ログから生成
1st	141,815	92,901 (1,990)	0	0
2nd	267,419	218,515 (63,265)	0	0
3rd	314,791	265,887 (85,544)	0	0
4th	362,355	313,451 (110,790)	0	0
5th	434,891	320,815[*6] (153,332)	65,172 (1,613)	0
6th	525,157	390,776 (209,462)	85,477 (2,696)	0
7th	616,096	484,615 (257,693)	82,577 [*7] (3,954)	0
8th	1,171,529	678,103 (405,284)	86,129 (8,204)	358,393
9th	1,489,998	746,382 (453,877)	91,715 (9,622)	602,997

りは，手間をかけずに多めに棋譜を選択できる手法が望ましかったためである．同一の棋譜が生成される可能性に関しては，得られた棋譜と同一のものが存在するかどうかをmd5 ハッシュによって確認し，はじいている．

自己対戦結果を使用した改善を繰り返して 4 回目で勝率検証リーグ戦での成績向上が頭打ちになったため，新規局面の導入を目的に，5〜7 回目の適用では Floodgate での自己の全対戦結果の棋譜と，Floodgate の高レーティングプログラムの対戦結果を対象に加えた．Floodgate の棋譜では勝敗や対局中の評価値は考慮していない．

8，9 回目の適用では，さらに局面の増加を目指して，対局には現れなかったが，対局中の検討に現れた局面をログから抽出した．Floodgate での全対戦結果と，持ち時間 2 分の自己対戦結果のログを対象に，1 手深く読んだときに 256 ポイント以上の変動があったときに変動の前後の局面両方を抽出している．1 回の対局から約 40 程度の局面が抽出され，各局面に対して，それを最終局面とする一局の棋譜を生成して回帰分析処理に渡している（回帰分析処理ではログから生成した棋譜と通常の棋譜とを区別せずに同じ処理をしている）ため，使用棋譜数が 7 回目の 62 万局に対し，8 回目では 117 万局に増加している．

[*5] 合計の棋譜数には base を作成した Bonanza メソッドで使用した棋譜 48,904 が含まれている．
[*6] 5th では，探索深さ限度 3 の自己対戦結果のうち，古いものを削除しているため，棋譜数が減少している．
[*7] 6th から 7th で floodgate の棋譜数が減少しているのは，gps などの実験結果の棋譜が入っていたことに気づき削除したため．同様の理由で，自己対戦の局数を除いた floodgate の局数が 8th で減少している．

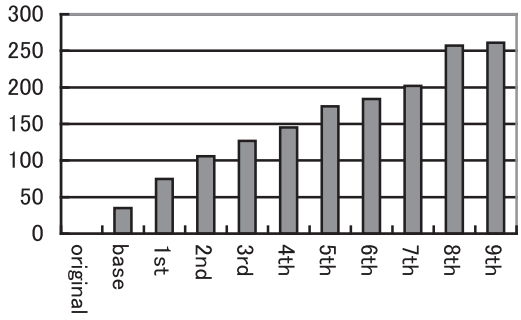


図 8 Bonanza 6.0 の一部として配布されている機械学習結果（original）からのレーティング差
Fig. 8 Elo rating difference from Bonanza 6.0 (original): calculated from results of matches against each other.

5.1 実験結果

提案手法を適用して得られた機械学習結果を使ったプログラム同士で，すべての組合せに対して先手後手それぞれで最低 200 局持ち時間 2 分切れ負けで対戦した結果得られたイロレーティングの変化を**図 8** に示す Original，base は，それぞれ Bonanza 6.0 の一部として配布されている変数テーブルを使用したもの，出発点として使用した学習結果，1st〜9th は，それぞれ n 回本手法を適用して得られた変数テーブルを使用したものである．

図 8 では，9th のレーティング向上が 250 を超えているが，これは実際よりも大きな値となっていると考えられる．将棋プログラムの改良では，改良前のプログラムと改良後のプログラムを直接対戦させるとレーティング向上効果が大きく見え，改良を継続してさらに強くなったプログラムや異種のプログラムと対戦させると，レーティング向上効果が小さくなる現象がしばしば見られる．その効果を排除するため，より強いプログラムを加えたリーグ戦，異種プログラムとの対戦によって提案手法の効果を評価した．

より強いプログラムを加えたリーグ戦では，[10] に述べている手法を用いて改良した変数テーブルを使用している．プログラムの変更は，評価値を求める部分のみで，探索部分は Bonanza 6.0 と同じものを使用している．使用した手法は

(1) KPP テーブルのパラメータに共通する特徴を抽出するため各パラメータを複数のパラメータの和に分解する手法
(2) 手番を考慮した評価値
(3) 飛角香の利きがどこまで到達しているかを考慮した評価値

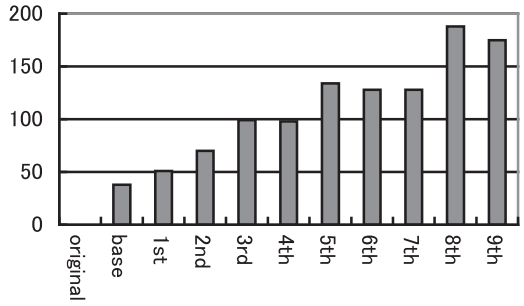


図 9 より強いプログラムを含めたリーグ戦で得られたレーティング差

Fig. 9 Elo rating difference from Bonanza 6.0 (original): calculated from results of league including stronger programs.

で，(1) のみを採用したもの，(1)，(2) を採用したもの，(1)〜(3) を採用したものの 3 種類の局面評価方式が存在する．それぞれの評価方式に対して，本論文で述べた手法を使用して変数テーブルを改善したため，段階に応じて，2 種，4 種，3 種のプログラムが存在し，これらの合計 9 種類のプログラムを加えたリーグ戦を行っている[*8]．各プログラムを最低二千局以上対局させて得られた base，1st〜9th の orignal とのレーティング差を**図 9** に示す．レーティング向上は図 8 と比較して小さくなるが，向上効果は残っている．

さらに，gpsfish_minimal[*9]を使用し，異種プログラムとの対戦による評価を行った．この実験の目的は本手法を使用した学習の段階による強さの比較であり，gpsfish_minimal との強さの比較ではないため，gpsfish_minimal のパラメータ調整は行っていない．使用した環境（gpsfish_minimal はコンパイルが Xeon X5690 3.47 GHz，実行が Xeon E5-2697 2.7 GHz 2 スレッド，bonanza はコンパイル，実行ともに Xeon E5-2690 2.9 Hz 1 スレッド）が，gpsfish_minimal には不利な条件になっているらしく，gpsfish_minimal のレーティングは original よりも 140 低くなっている．各段階のプログラムと gpsfish_minimal を最低 600 回対局して得られた勝率をレーティングに換算したものを，**図 10** に示す．全体の傾向は図 9 とほぼ同じであり，ばらつきを考慮（図 10 に示すレーティング差の標準偏差は 20 程度）してもレーティング向上効果が出ていることが確認できる．図 8，図 9 の結果から，base と 9th の間のレーティング向上は，150 程度と考えられる．

最も対局数が多い図 9 の結果では，Floodgate での対局を加えた 5th とログから生成した棋譜を使用した 8th で

[*8] 追加プログラムのレーティングは，パラメータテーブルの種別ごとに最もレーティングが高かったものは，(1) のみを採用したものが 240，(1)，(2) を採用したものが 400，(1)〜(3) を採用したものが 430，それぞれ orignal よりも高かった．同種のプログラムでの評価結果であることから，これらの数字も高めに出ているものと考えられる．

[*9] http://gps.tanaka.ecc.u-tokyo.ac.jp/gpsshogi/pukiwiki.php?Gpsfish_minimal（参照 2016-08-28）

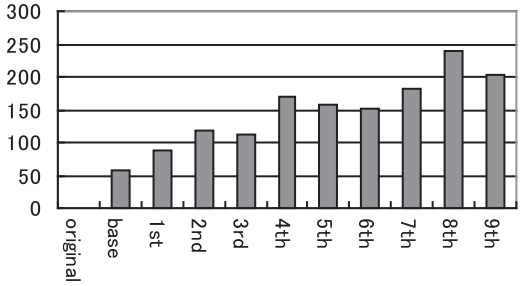


図 10 異種プログラムとの対戦から得られたレーティング差

Fig. 10 Elo rating difference from Bonanza 6.0 (original): calculated from results of matches against a non-Bonanza program.

レーティングが向上している．6th，7th のレーティングはあまり向上していない理由は，棋譜数の増加量が少ないためと考えている．棋譜の増加が少ない理由は，Floodgate での対局は他プログラムによる対戦を入れても月に 1 万局程度しか増やせないためである．新規のソースから棋譜を追加することで頭打ちだったレーティングが向上していることから，誤った評価値がついている局面を供給できた数と効果が連動しているように見える．仮説に沿った現象ではあるが，試行数が少ないため，これだけで結論することはできない．

5.2 実行時間

実行時間に関しては，本実験では，収束の判断を慎重にしたのと，次の段階で使用する棋譜の蓄積を待つため，Xeon E5-2690（2.9 GHz）2 ノード（物理 16 コア）の Linux マシンを使用して改善処理 1 回の適用に 1 カ月程度かけている．これは時間のかけすぎで，50 万局程度の棋譜数であれば，最急降下法処理を SGD-QN [11], [12] などで置き換えることで 2〜3 日程度に抑えることが可能である．

6. おわりに

教師データの不足を補うことを目標に，強化学習類似の手法を提案し，それに基づいてレーティングを 150（配布されている Bonanza 6.0 のテーブルと比較すると 190）改善できることを示した．さらに，繰り返し適用したときの効果の減少，新規データの供給による効果など，仮説から予想される結果が得られている．

将棋の場合，初期状態が一定であるため，探索範囲を限定することが可能であるなど，本手法の効果を強める性質もある．そのため，現実の様々な問題への本手法を適用するには，本手法を適用する状態を選択する手法など，まだ解決すべき問題もあると思われるが，実用化に向け検討を進めたいと考えている．

参考文献

[1] Breuer, M.A.: A class of min-cut placement algorithms, *Proc. 14th Design Automation Conference*, pp.284–290 (1977).

[2] Hoki, K. and Kaneko, T.: Large-Scale Optimization for Evaluation Functions with Minmax Search, *Journal of Artificial Intelligence Research*, Vol.49, pp.527–568 (2014).

[3] Boser, B.E., Guyon, I. and Vapnik, V.: A training algorithm for optimal margin classifiers, *Proc. 5th Annu. Workshop Comput. Learn. Theory*, pp.144–152 (1992).

[4] Hoki, K. and Muramatsu, M.: Efficiency of Three Forward-Pruning Techniques in Shogi: Futility Pruning, Null-Move Pruning, and Late Move Reduction (LMR), *Entertainment Computing*, Vol.3, No.3, pp.51–57 (2012).

[5] Sutton, R.S. and Barto, A.G.: *Reinforcement Learning*, MIT Press (1988).

[6] Veness, J., Silver, D., Uther, W. and Blair, A.: Bootstrapping from Game Tree Search, *Advances in Neural Information Processing Systems*, Vol.22, pp.1937–1945 (2009).

[7] Buro, M.: From Simple Features to Sophisticated Evaluation Functions, *Computers and Games, LNCS*, pp.126–145, Springer-Verlag (1998).

[8] 林 伸也，浦 晃，三輪 誠，田浦健次朗，近山 隆：自己対戦棋譜を利用した半教師あり学習による将棋の評価関数の学習，*GPW2011* (2011).

[9] 森脇大悟，金子知適：自動対戦サーバ「Floodgate」，コンピュータ将棋協会誌，Vol.20, pp.3–10 (2010).

[10] 金澤裕治：第 25 回世界コンピュータ将棋選手権アピール文書，コンピュータ将棋協会（オンライン），入手先 ⟨http://www.computer-shogi.org/wcsc25/appeal/NineDayFever/NDF-2015.txt⟩（参照 2016-08-28）.

[11] Bordes, A., Bottou, L. and Gallinari, P.: SGD-QN: Careful quasi-Newton stochastic gradient descent, *J. Mach. Learn. Res.*, Vol.10, pp.1737–1754 (2009).

[12] Bordes, A., Bottou, L., Gallinari, P., Chang, J. and Smith, S.A.: Erratum: SGD-QN is less careful than expected, *J. Mach. Learn. Res.*, Vol.11, pp.2229–2240 (2010).

金澤 裕治 （正会員）

1990 年東京大学大学院工学系研究科情報工学専攻修士課程修了．同年 (株) 富士通研究所入社．現在，設計自動化関連の研究に従事．