将棋における勾配ブースティング木を用いた評価関数

渡辺 敬介[1,a)]　金子 知適[2,3,b)]

概要：本論文は，将棋における勾配ブースティング木を用いた局面評価関数を実証する．現在，殆どの将棋プログラムでは線形モデルを用いた評価関数が使用されている．一方で，機械学習分野では様々な非線形モデルを用いた手法が提案されており，これらの手法をうまく将棋に適用できれば既存手法より正確な評価関数を作成できると期待される．本研究は，勾配ブースティングを用いることにより評価関数の改善を試みた．1 手当たりの探索局面数を固定して対局実験を行った結果，提案手法は基本手法に対して勝率 6 割以上で勝ち越し，提案手法が有力な手法であることが示された．しかし探索速度では提案手法に従来手法に大きく劣り，さらなる改善が必要であると考えられる．

Evaluation function with gradient tree boosting for shogi

Keisuke Watanabe[1,a)]　Tomoyuki Kaneko[2,3,b)]

Abstract: This paper explores an application of gradient tree boosting into evaluation functions of shogi. Currently, evaluation function with a linear model is implemented in most shogi programs. Meanwhile, many non-linear models have been proposed in the field of machine learning research, and if these non-linear approaches could be successfully adopted to evaluation function, it would give more accurate evaluation of positions than linear approach does. Therefore, this research tries to improve an evaluation function by incorporating gradient boosting. Self-play experiments with fixed search nodes per position show that the effectiveness of our approach in terms of evaluation accuracy by achieving over 60 percent win ratio against the baseline program. Our future work is to improve its execution speed, that is much slower than the baseline for now.

1. 背景

強い将棋プログラムを作成するためには，正確な局面評価関数を作成することが重要である．

現在，多くの将棋プログラムの評価関数では，盤面を何らかの特徴ベクトルで表現し，特徴の重み付き線形和により評価関数を構成，重みを機械学習を用いて最適化している [4] [9]．

一方で，機械学習の分野では様々な非線形モデルが提案されており，これらの手法をうまく将棋の評価関数に適用できれば，従来の線形モデルに基づく評価関数より正確な評価関数が得られると期待される．

しかし将棋の評価関数の最適化では，文献 [4] や [9] のように探索と組み合わせたパラメータの最適化が必要であると考えられるため，現在将棋以外の分野で成功している機械学習手法をそのまま将棋に適用可能かは未知である．

非線形モデルの機械学習手法を将棋の評価関数に適用した例としては，カーネル SVM を評価関数に用いる研究 [7] があるが，この研究ではカーネル SVM を用いた評価関数を探索に組み込んでおらず，今後の課題としている．

本研究は，探索を用いたパラメータ最適化と gradient boosting [2] を組み合わせることで，既存手法よりも優れた評価関数の作成が可能であるかを実証することを目的としている．

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
2 東京大学大学院情報学環
Interfaculty Initiative in Information Studies, the University of Tokyo
3 国立研究開発法人科学技術振興機構さきがけ
JST, PRESTO
a) watanabe-keisuke518@g.ecc.u-tokyo.ac.jp
b) kaneko@acm.org

2. 関連研究

本章では，gradient boosting [2] と，その弱学習器として用いる回帰木について述べる．

2.1 Gradient boosting

Gradient boosting は，ブースティングアルゴリズムの一種であり，複数の弱学習器を統合して全体の学習器を構成する手法である．弱学習器には回帰木を用いることが多く，本研究では弱学習器として回帰木を用いる．

入力ベクトルを x, ラベルを y として, 全体の学習器

$$F(x) = f_0(x) + f_1(x) + f_2(x) + ... + f_M(x)$$

を損失関数 $L(y,\ F(x))$ が最小になるように弱学習器 $f_m(x),\ m = 1,\ 2,\ ...\ M$ を逐次的に学習, 統合していく．すなわち, 学習開始時に関数 $F_0(x) = f_0(x)$ が与えられるとし, m ステップ目の学習では m 個の弱学習器からなる学習器

$$F_m(x) = F_{m-1}(x) + f_m(x)$$

を損失 $L(y,\ F_m(x))$ が最小になるように弱学習器 $f_m(x)$ を決定する．Stochastic gradient boosting [3] では，弱学習器を最適化する際に訓練集合中の全てのデータを使わず，ランダムに一定数のデータをサンプリングして用いる．

具体的には，以下のアルゴリズムにより学習器 $F(x)$ を得る．

(1) 損失を最小化するような定数関数 $F_0(x)$ を得る．
(2) For $m = 1$ to M
  (a) 訓練集合から N 個のデータをサンプリングし，集合 D を得る．
  (b) 集合 D の各要素 $(x_1, y_1),\ (x_2, y_2),\ ...\ (x_N, y_N)$ について, 勾配 $y'_i = -\frac{\partial L(y_i,\ F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}$ を計算する．
  (c) 得られた勾配を予測する回帰木 $T(x)$ を生成する．すなわち，$\sum_{i=1}^{N} (y'_i - T(x_i))^2$ を最小化する回帰木を生成する．
  この回帰木が弱学習器 $f_m(x)$ である．
  (d) 損失 $\sum_{i=1}^{N} L(y_i,\ F_{m-1}(x_i) + T(x_i))$ が最小になるように回帰木 T の葉の重みを最適化する．
  (e) $F_i(x) = F_{m-1}(x) + \nu T(x)$ とする．ν は shrinkage パラメータで，$0 < \nu \leq 1$ を満たす定数．
(3) $F_M(x)$ を $F(x)$ として出力する．

第 3 章で詳しく述べるが，本研究では shrinkage パラメータ ν は 1 とし，$F_0(x)$ は定数関数から変更した．

2.2 回帰木

2.2.1 回帰木による予測

回帰木は木構造を用いた予測モデルであり, 特徴ベクトル x に対して予測値 y, $y \in \mathbb{R}$ を出力する．予測を行う際には，根節点から葉へと木をたどっていくことにより，予測値を決定する．木の内部節点には特徴ベクトル x の要素と閾値が設定されており，これと特徴ベクトルを比較することで入力を各子節点へと分類する．葉節点には値が設定されており，たどり着いた葉の値が予測値となる．

これは，特徴ベクトルの空間を軸に垂直な超平面でいくつかの領域に分割し，各領域ごとに予測値を割り当てることと等価である．

例えば図 1 の回帰木の例では，8 八に先手玉があり 7 八に先手の金がある場合には予測値は 110, 8 八に先手の玉がなく 1 二に後手の金がある場合には予測値は 250 となる．

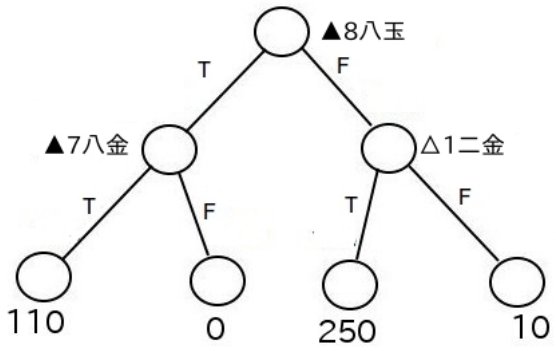

図 1 回帰木の例

2.2.2 回帰木の学習

次に回帰木の学習方法について述べる．

自乗誤差の総和を最小となるるように回帰木の構造を決定したいが，内部節点での分類に用いられる変数と閾値の組み合わせは膨大となるため，厳密解を得るのは現実的には不可能であることが知られている．そのため，単一の根節点からスタートし，各節点で最も損失が下がるように子節点を生成することで貪欲最適化を行い木を成長させる手法が広く用いられている．

子節点を生成し木を成長させるステップでは，各節点で最も損失が下がるような変数としきい値を選択する，という動作を再帰的に実行する．

また，本研究では，根節点からの距離が一定以上になった時と訓練集合の中で葉節点に分類される訓練例の数が一定以下になった時に木の成長を停止させることとした．

木の複雑さを制御し過学習を抑制する手法として，正則化により葉節点の数を抑制する手法や木を成長させた後に情報量基準を用いて枝刈りを行う手法も存在するが，本研究ではこれらの手法は実装していないため，割愛する．

3. 手法

本研究では，同一のコンピュータプレイヤで対局を行い，その対局結果を予想するように評価関数の最適化を行う．

このような手法は Logistello [1] の評価関数の最適化や AlphaGo [5] での ValueNet の作成にも用いられている手法であり，将棋においても適用された例がある [6]．オセロや囲碁における先行研究では損失関数には対局結果と予測値の自乗誤差を用いており，将棋における先行研究では負の対数尤度を損失関数としている．

また，実験に用いるプログラムの詳細や，データセットの生成条件については第 4.1 章で述べる．

3.1 損失関数

提案手法では以下の損失関数を最小化する．ここで，ラベル Y_i は, 局面 p_i の手番側から見た対局結果であり, 手番側プレイヤの勝ちならば 1, 負けならば 0, 引き分けならば 0.5 となる．この関数は，実際の対局結果が引き分けの局面をデータセットから除けば実際の勝敗 Y_i と予測された勝率 y_i の交差エントロピーをデータセットの数で割った値となっており，引き分けの局面では評価値の絶対値が 0 に近いほど小さくなる関数である．

$$
\begin{aligned}
L(F) &= \frac{1}{n}\sum_{i=1}^{n} -(Y_i \log(y_i) + (1-Y_i)\log(1-y_i)) \\
y_i &= \sigma(q(F, p_i)) \\
F &: \text{評価関数} \\
q(F, p_i) &: \text{局面 } p_i \text{で静止探索して得られる評価値} \\
\sigma(x) &= \frac{1}{1+\mathrm{e}^{-kx}} \\
k &= 1.7 \times 10^{-3}
\end{aligned}
$$

3.2 基本手法

基本手法として，駒の価値と 2 駒の位置の組み合わせ特徴の線形和とした評価関数を用いた．

駒の価値は予め定めた定数に固定し，2 駒の位置の組み合わせ特徴の重みをバッチサイズ 32 のミニバッチ勾配降下法により最適化した．

すなわち，各 epoch で

(1) 訓練集合から，この epoch 中に選択されていない局面をバッチサイズ分だけ選択する．
(2) 選択された局面で最新の評価関数を用いて静止探索を行い，最善応手列の末端局面を得る．
(3) 得られた末端局面で勾配を計算し，勾配降下法により重みを更新する．

という手順を訓練集合中の全ての局面を選択するまで繰り返す．また，正則化は行っていない．

3.3 提案手法

基本手法を Stochastic gradient boosting で拡張した評価関数を作成した．基本的なアルゴリズムは第 2.1 章で述べた通りであるが，関数 F_0 を定数関数ではなく基本手法により得られた評価関数とした点と，3.1 で定義した損失関数では勾配を求める際に静止探索を行い得られた末端局面を用いる点で異なる．

また，通常の機械学習であれば，文献 [3] で述べられているように回帰木の葉の重みを決定するステップでは損失を最小化するような値を各葉ごとに独立に計算すれば良い．しかし本研究では葉の重みを変更すると静止探索により得られる最善応手列の末端局面が変わってしまうため，上記の手法では損失を最小化できるとは限らない．

そのため，葉の重みを最適化する際には基本手法と同様バッチサイズ 32 のミニバッチ勾配降下法を用いた．これは，より新しい評価関数での静止探索結果を用いることで，重みを変更した最新の評価関数での静止探索結果を最適化することと近い結果が得られると期待されるためである．

4. 実験

4.1 実験条件

4.1.1 実験に用いたプログラム

本研究では，筆者が開発中の将棋プログラム「おから饅頭」に実験に必要な実装を施した．このプログラムは，実現確率探索をベースにした探索を実装しており，第 27 回世界コンピュータ将棋選手権で 14 位となったバージョンに若干の変更を加えたものである．

4.1.2 データセットの生成

本項では，データセットを作成する際に行った対局について述べる．本研究では，駒の価値のみの評価関数に乱数を加えたプログラムによる自己対戦を 400 万局行い，データセットを作成した．また，256 手経過した時点で対局を打ち切り，引き分けとして扱った．

乱数は文献 [8] 同様，局面のハッシュ値に対して正規分布に従って生成した値を割り当てた．標準偏差は歩の価値と同じ大きさとした．また，対局時の探索深さを 5 とした．

これらの棋譜のうち 5 万局をテスト用，残りの棋譜をトレーニング用とし，各棋譜から 10 局面ずつ復元抽出した．

以上の手順により作成されたテストセット 50 万局面, トレーニングセット 3,950 万局面のデータセットを用いた．

また, 棋譜生成にかかる計算コストを見積もるため, CPU が AMD Ryzen 7 1700 (3GHz, 8cores) である計算機上で並列数を 8 として同様の条件で 2,000 局の棋譜を生成した所, 3 分 58 秒で棋譜の生成が終了した．

4.2 回帰木の設定

回帰木に入力として与える特徴ベクトルは，表 1 に示す binary vector とした．また，盤面は手番側プレイヤから見た状態，すなわち後手番の場合は 180 ° 反転した状態で各マスの座標を扱った．さらに，手番プレイヤの玉が右側に

ある場合，盤面を左右反転したものとして各マスの座標を扱った．

表 1 回帰木の入力とする特徴ベクトル

特徴	次元	id		
持ち駒	76	0	-	75
手番プレイヤの玉の X 座標	5	76	-	80
非手番プレイヤの玉の X 座標	9	81	-	89
手番プレイヤの玉の Y 座標	9	90	-	98
非手番プレイヤの玉の Y 座標	9	99	-	107
手番プレイヤ側の玉以外の盤上の駒の位置	81	108	-	188
非手番プレイヤ側の玉以外の盤上の駒の位置	81	189	-	269
盤上の歩または銀の位置	81	270	-	350
盤上の香車の位置	81	351	-	431
盤上の桂馬の位置	81	432	-	512
盤上の銀，金または成金の位置	81	413	-	593
盤上の角または馬の位置	81	594	-	674
盤上の飛車または龍の位置	81	675	-	755
盤上の馬または龍の位置	81	756	-	836
各マスにおける効き数の優劣	162	837	-	998
合計	999			

次に，学習時に用いるハイパーパラメータについて述べる．一つの回帰木の生成に用いるサンプル数を 2 千万局面，回帰木の最大深さは 10 とし，分類されるデータ数が 10 以下の節点はそれ以上分割を行わないこととした．

4.3 実験結果

4.3.1 損失の比較

基本手法と提案手法での損失の値をそれぞれ図 2,3 に示す．

基本手法のテストセットにおける損失は 8 epoch 終了時点で 0.553 であり，提案手法でのテストセットにおける損失は回帰木を 40 本生成し終えた時点で 0.551 であることから，提案手法が有効であると考えられる．

しかし図 3 では，トレーニングセットにおける損失は回帰木を増やすにつれて下がっているものの，テストセットでは損失が減少しているのは最初の数本から 10 本だけである．この事から，提案手法では過学習が起きていると考えられ，データセットを増やす，正則化を導入するなどにより過学習を抑制する必要があると考えられる．

次に，テストセットに含まれる局面について，手番側プレイヤ側から見た実際の勝敗と，基本手法と提案手法での予測勝率について集計した．予測勝率を 0.05 刻みの区間に分割し，対局結果ごとに予測勝率の出現頻度を測定，ヒストグラムにしたものが図 4 である．このヒストグラムでは提案手法と比較手法で大きな差は見られないが，実際の対局結果が引き分けでなかった局面では提案手法では基本手法より予測勝率が 0.5 となった局面が僅かに減っている．

また，本実験で用いたデータセットは予測勝率が 0.5 付近の値となる局面，すなわち形勢が互角であると判断されるような局面が多く含まれていることがわかる．

また，学習により得られた最初の回帰木を `https://game.c.u-tokyo.ac.jp/ja/kwatanabe/` に掲載する．

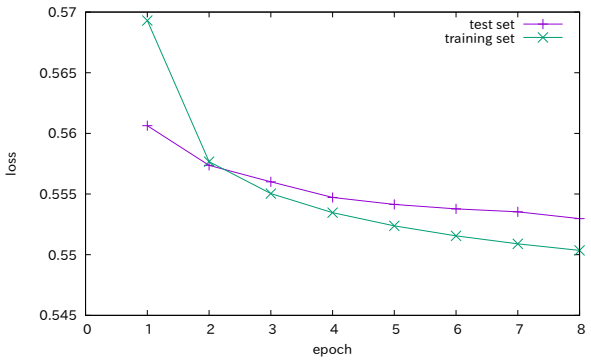

図 2 基本手法における損失

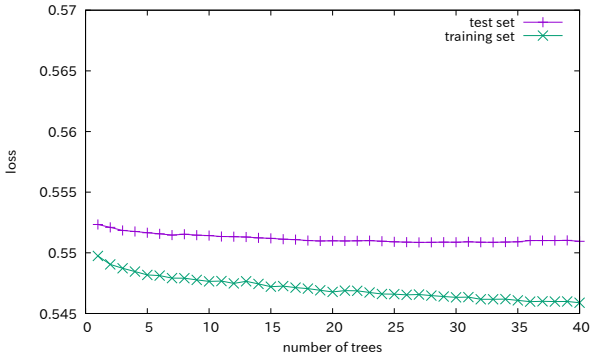

図 3 提案手法における損失

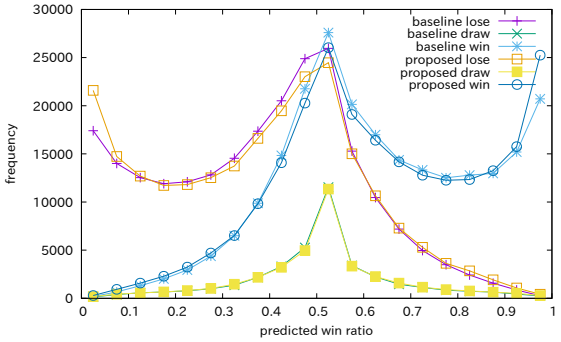

図 4

4.3.2 対局実験

提案手法と基本手法で 1,000 局の対局実験を行った．提案手法の回帰木の数は 10, 20, 30, 40 の 4 通りについて計測した．双方 1 スレッドで探索を行うこととし，一手当たりの探索局面数を 100 万局面とした．また，双方のプレイヤは同一の定跡ファイルを用いており，序盤はこの定跡に従ってランダムに指させることで対局が同一の進行になることを防いでいる．また，256 手経過した時点で対局を打ち切り引き分けとした．

対局実験の結果を表 2 に示す．回帰木の数が 10, 20, 30,

40 のいずれの場合も提案手法は基本手法に勝率 6 割以上で勝ち越している．提案手法の勝率は有意水準 5 %の片側検定で有意に 0.5 より高く, 提案手法が有効であると言える．

表 2 対局結果

回帰木の数	基本手法に対する勝敗
10	660 - 37 - 303
20	665 - 34 - 301
30	661 - 38 - 301
40	631 - 34 - 335

4.4 探索速度の測定結果

初期局面で 10 秒間の探索を行い，探索局面数を測定した．CPU が AMD Ryzen 7 1700 (3GHz, 8cores) である計算機上で実験を行った．また，探索では 1 スレッドのみを用いた．

測定結果が 3 である．この表から, 提案手法は基本手法に比べて探索速度が遅く，回帰木の数を 10 本とした時点で探索速度が基本手法の約半分となっていること，回帰木の数を増やすとさらに実行速度が低下することがわかる．

表 3 探索速度

手法	回帰木の数	探索局面数
基本手法	-	15,440,576
提案手法	10	7,615,372
提案手法	20	6,022,416
提案手法	30	4,646,500
提案手法	40	4,135,237

4.4.1 探索速度を考慮した対局実験

第 4.4 章では，提案手法では回帰木の数を 10 本とした時点で探索速度が約半分となることが分かった．そこで，提案手法と基本手法で探索局面数に差をつけ，1 手あたりの思考時間が概ね等しくなるように対局実験を行った．

提案手法は回帰木の数を 10 本，1 手当たりの思考局面数を 50 万とし，基本手法は 1 手当たりの思考局面数を 100 万とした．その結果，提案手法から見た勝敗は 403 勝 556 敗 41 分となった．提案手法の勝率は有意水準 5 %の片側検定で有意に 0.5 より低く，提案手法における探索速度の低下が大きな問題であることがわかる．

5. まとめ

本論文では，将棋における gradient boosting を用いた局面評価関数を提案し，その評価を行った．テストセットにおける損失および対局実験において提案手法は基本手法の性能を上回り，提案手法の有効性が示された．

しかし，提案手法では過学習を起こしていることが示唆され，また，探索速度においても提案手法は大きな問題があることが分かった．今後これらの課題を解決することができればより性能の高い評価関数を実現できると考えられる．

また，評価関数の精度を改善するための手段として，データセットや損失関数を変更することが考えられる．その時に提案手法と基本手法で評価関数の精度がそれぞれどのように変化するか調査する必要がある．

謝辞

この研究の一部は，JSPS 科研費 16H02927 と JST さきがけの支援を受けています．

参考文献

[1] Buro, M.: Improving heuristic mini-max search by supervised learning, *Artificial Intelligence*, Vol. 134, No. 1–2, pp. 85–99 (2002).

[2] Friedman, J. H.: Greedy Function Approximation: A Gradient Boosting Machine, *The Annals of Statistics*, Vol. 29, No. 5, pp. 1189–1232 (online), available from ⟨http://www.jstor.org/stable/2699986⟩ (2001).

[3] Friedman, J. H.: Stochastic gradient boosting, *Computational Statistics & Data Analysis*, Vol. 38, No. 4, pp. 367–378 (2002).

[4] Hoki, K. and Kaneko, T.: Large-scale optimization for evaluation functions with minimax search, *Journal of Artificial Intelligence Research*, Vol. 49, pp. 527–568 (2014).

[5] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. et al.: Mastering the game of Go with deep neural networks and tree search, *Nature*, Vol. 529, No. 7587, pp. 484–489 (2016).

[6] 鶴岡慶雅，横山大作，丸山孝志，高瀬　亮，大内拓実：第 26 回世界コンピュータ将棋選手権激指 アピール文書，http://www.computer-shogi.org/wcsc26/appeal/Gekisashi/appeal.txt.

[7] 末廣大貴，畑埜晃平，坂内英夫，瀧本英二，竹田正幸：SVM による 2 部ランキング学習を用いたコンピュータ将棋における評価関数の学習，電子情報通信学会論文誌 D，Vol. J97-D, No. 3, pp. 593–600 (2014).

[8] 小幡拓弥，杉山卓弥，保木邦仁，伊藤毅志：将棋における合議アルゴリズム：既存プログラムを組み合わせて強いプレイヤを作れるか？，ゲームプログラミングワークショップ 2009 論文集，Vol. 2009, pp. 51–58 (2009).

[9] 金澤裕治：教師データが不足した環境での機械学習結果改善手法，情報処理学会論文誌，Vol. 57, No. 11, pp. 2382–2391 (2016).