第 32 回 世界コンピュータ将棋選手権
Novice アピール文章

中屋敷 太一　　熊谷 啓孝　　笹井 雄貴　　矢内 洋祐

2022 年 2 月 15 日

1　概要

本稿では，コンピュータ将棋ソフトウェア Novice の用いているアルゴリズムについて記述する．筆者らは，Novice を AlphaZero [1] に倣って開発した．Novice は，AlphaZero 同様，探索（先読み）にモンテカルロ木探索 (Monte-Carlo Tree Search, MCTS)，そして局面の評価（大局観）にニューラルネットワークを用いている．自己対戦による棋譜の生成と，その棋譜からのニューラルネットワークの学習を繰り返すことにより，強い評価関数を学習することを目指している．

本稿では，Novice が AlphaZero のアルゴリズムに加えた工夫について記述する．

2　Novice の工夫

本節では，ニューラルネットワークの学習手法について記述する．ニューラルネットワークの学習は，自己対戦により教師データを生成し，その教師データを用いてニューラルネットワークのパラメータを更新して行う．初めに，自己対戦の用いている工夫を記述する．そしてその次に，ニューラルネットワークのの学習に関する工夫について記述する．

2.1　自己対戦

AlphaZero の手法では，自己対戦による教師データから学習が行われているが，学習の際には膨大な計算資源が使われた [1]．そのため，この再現実験を行うことは容易ではなく，現実的な時間で強い将棋プログラムを作成するためには，いくつかの工夫が必要となる．

筆者らは，学習の予備実験の結果，3 つの次に記述する問題に遭遇した．1 つ目は，少なくない教師データが，千日手による引き分けによって終局していること，2 つ目は，優勢な局面でも勝ち切るまでに多くの手数を要していること，そして 3 つ目は，戦型選択が偏り，いくつかの戦法の学習が行えていないことであった．本小節では，これらの問題について，筆者らが用いた解決策を記述する．

引き分け率の調整

予備実験では，教師データ内に，引き分けで終局している棋譜が少なくなかった．筆者らは，この原因は，攻める際に評価値が一瞬下がることが多いこと（将棋では攻める際に先に駒を取られることが多い）に由来すると考えた．特に (Novice の) 自己対戦では，MCTS のシミュレーション回数が少なく (具体的には 800 回程度)，探索が浅いため，この仮説は有力であると考えられる．

この問題を解決するために，生成している教

師データ内で引き分け率が 2% を超えている場合には，その次に行う自己対戦の設定を，引き分けを先手勝ちまたは後手勝ちの設定で行うようにした．なお，教師データ内で先手勝率が 50% を超えている場合には引き分けを後手勝ち，後手勝率が 50% を超えている場合には引き分けを先手勝ちとした．

ニューラルネットワークがこれらの対局条件を区別して学習できるように，前述したようにニューラルネットワークに引き分けの際の点数（先手勝ちなら先手 +1 かつ後手 −1）を入力特徴量として与えた．

以上の工夫により，実際に生成される教師データ内の棋譜の引き分け率を常におおよそ 2% に保つことに成功した．

様々な最大手数設定の対局

予備実験では，勝勢な局面でも，終局まで多くの手数を要している棋譜が多く発見された．筆者らは，この問題は，終局時に勝てる棋譜であれば，要する手数の長さを考慮していないことが原因であると考えた．

この問題を解決するために，様々な最大手数の設定で自己対戦を行うことで解決を図った．なお，最大手数に達した場合には，即座に引き分けとして扱った．この変更により，勝勢の局面でも，長く手数を要する勝ち手順には，引き分けとして終局する可能性が生じる．そのため，できるだけ短い手順で終局する手順が学習できることが期待される．また，副次的に，劣勢な局面では最大手数まで粘ろうとする手順を学習していることが期待される．

ニューラルネットワークが様々な最大手数に対応できるように，前述したようにニューラルネットワークに現在の手数および最大手数を入力特徴量として与えた．

初期局面集の使用

予備実験として，将棋の初期配置のみから学習を行うと，安定しない評価値が出力されていたり，良い手を見落としたりしていることが発見された．そこで，初期局面集を用意し，自己対戦による棋譜生成の際には，対局ごとにその中からランダムで 1 局面を選び，その局面を初期局面として使用した．初期局面集には，筆者らが選んだ約 100 局面を使用した．なお，初期局面集を用いるという工夫はやねうら王[*1]やdlshogi[*2]に既に用いられており [2, 3]，筆者らは大いに先行事例を参考にした．

[*1] https://github.com/yaneurao/YaneuraOu.git

[*2] https://github.com/TadaoYamaoka/DeepLearningShogi

表 1　ニューラルネットワークの入力特徴量

特徴量	チャネル数
手番の駒配置	14
相手の駒配置	14
手番の持ち駒	24
相手の持ち駒	24
手番	2
王手	1
各列の歩の存在	2
現在の手数 / 最大手数	1
1.0 / 最大手数	1
引き分け点数	2
合計	85

2.2　ニューラルネットワークの構造と学習

本小節では，Novice の用いたニューラルネットワークの構造と学習手法について記述する．

ニューラルネットワークの構造

ニューラルネットワークの入力層は，1 局面あたり，85 チャネル，9×9 ピクセルで構成される．各チャネルが表現する特徴量を表 1 に示す．盤上の位置に依存する情報は，9×9 のうち，対応する場所のみが指定される値となる．それ以外のものは，9×9 ピクセル全て同じ値が指定される．駒配置については，先手後手を含めて駒種を区別した各チャネルを用意し，各駒の存在するマスに対応したピクセルに 1 を設定した．持ち駒については，上限を 4 枚とし，dlshogi と同じ手法 [4] で設定した．手番には 2 チャネル割り当てられており，与えられた局面の手番が先手番の時はこの 1 チャネル目のみが 9×9 全て 1，後手番の時はこの 2 チャネル目のみが 9×9 全て 1 と設定される．なお，AlphaZero では，手番を表すためには 1 チャネルのみが使用され，後手番の時にそのチャネルを 9×9 全て 1 にしていた．しかし，Leela Zero[*3]によって，先手番での局面を入力とするニューラルネットワークの実行の際に，ニューラルネットワークが盤面を把握しづらくなる可能性が指摘されている [5]．

> leela-zero/README.md より抜粋
> *There are 18 inputs to the first layer, instead of 17 as in the paper. The original AlphaGo Zero design has a slight imbalance in that it is easier for the black player to see the board edge (due to how padding works in neural networks). This has been fixed in Leela Zero.*

そのため，Novice も Leela Zero と同様の手法で 2 チャネル使用した．Novice では，学習の効率向上を期待し，現在の局面が王手されているかどうかと，それぞれの升の列に歩があるかどうかを追加の情報として与えた．また，後述する理由のため，現在の手数及び最大手数に関する情報と，引き分けで対局が終了した際に先手後手がそれぞれ得られる点数を入力に与えた．

中間層は SE block [6] 有りの Residual Network [7] で構成した．なお，12 blocks × 256 filters の構成を用いた．

最終的に，ニューラルネットワーク $f(\cdot;\theta)$ は 4 つの値を出力する (式 1)．

$$\boldsymbol{p}(s), \boldsymbol{v}(s), \boldsymbol{d}(s), \boldsymbol{z}(s) = f(s;\theta) \qquad (1)$$

ここで $\boldsymbol{p}(s)$ は局面 s での次の一手の確率分布，$\boldsymbol{v}(s)$ は { 勝ち，引き分け，負け } の確率分布，$\boldsymbol{d}$ は { 先手の宣言勝ち，後手の宣言勝ち，それ以外 } の確率分布，$\boldsymbol{z}(s)$ は 9×9 の各マスごとに先手後手それぞれ利きがあるかどうかの予測である．

[*3] https://github.com/leela-zero/leela-zero

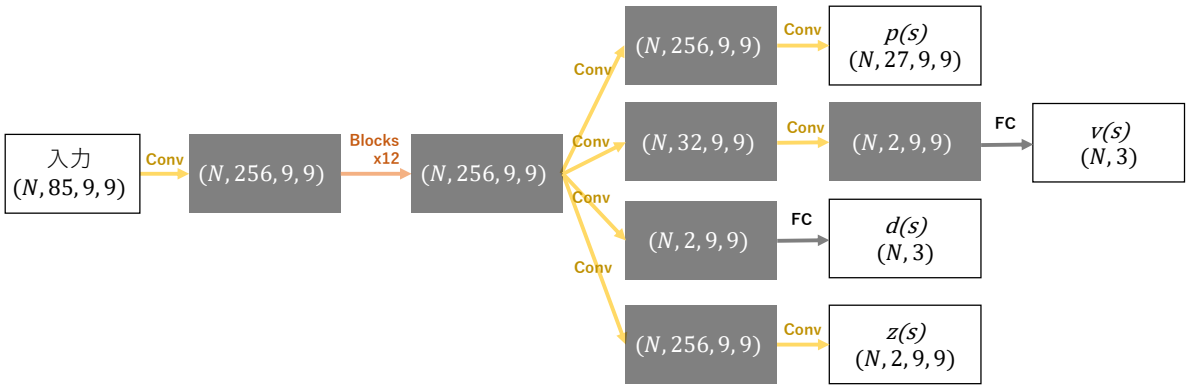

図 1　ニューラルネットワークの概略図

ニューラルネットワークの学習

ニューラルネットワークの重み θ の学習は，自己対戦の棋譜を教師データとし，Stochastic Gradient Descent を用いて次のロス関数 $\mathcal{L}$ を最小化をすることで行った．

$$\begin{aligned}\mathcal{L}(s;\theta) = &-\boldsymbol{p}'(s)\cdot\log\boldsymbol{p}(s;\theta)\\ &-v'(s)\cdot\log v(s;\theta)\\ &-0.1\cdot d'(s)\cdot\log d(s;\theta)\\ &-0.1\cdot\boldsymbol{z}'(s)\cdot\log\boldsymbol{z}(s;\theta)\\ &-0.01\cdot\mathcal{H}(\boldsymbol{p}(s;\theta))\end{aligned}$$

ここで $\boldsymbol{p}'(s), v'(s), d'(s), z'(s)$ はそれぞれ，教師データの，MCTS の訪問回数の分布，対局結果，入玉結果，盤面の利きである．また $\mathcal{H}(\cdot)$ は方策のエントロピーであり，方策が確定的になるのを防ぐ効果を期待できる [8]．

2.3　詰み探索による強化

モンテカルロ木探索と合わせて，PV Mate [9, 10] と末端局面での 5 手詰め探索を用いている．PV Mate では，現在の探索の中で最も現れそうな進行に現れる各局面について，証明数探索 (Proof Number Search, PNS) [11] を用いて詰みの有無を探索している．なお，実装を簡単にするため df-pn [12] は未使用である．また，モンテカルロ木探索に現れる全ての局面で深さ 5 の深さ優先探索 (Depth First Search, DFS) を行っている．なお，PNS および DFS は，それぞれ専用のタスクのキューを持ち，モンテカルロ木探索とは独立したスレッドで動作させることで高速化をしている．

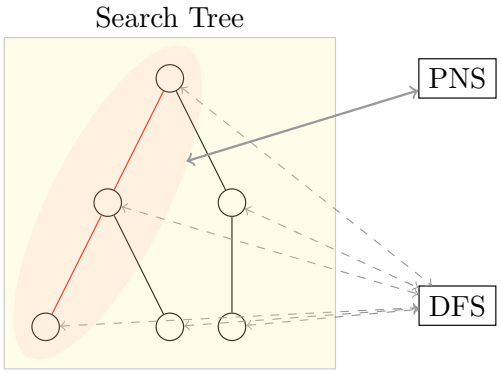

図 2　詰み探索の概略図．赤い線は PV を表す．PV 上の局面では PNS による詰み探索を行う．加えて，全ての局面で DFS による詰み探索を行う．

参考文献

[1] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, Vol. 362, No. 6419, pp. 1140–1144, 2018.

[2] Tadao Yamaoka. 将棋 AI の進捗 その 19 (初期局面集). https://tadaoyamaoka.hatenablog.com/entry/2018/04/06/001745. (2022/01/19 閲覧).

[3] やねうらお. Yaneuraou/source/learn/learner.cpp. https://github.com/yaneurao/YaneuraOu/blob/cc30a4323fa323be29d8513bb44269aaa4b4d39a/source/learn/learner.cpp#L3421. (2022/02/09 閲覧).

[4] Tadao Yamaoka. 将棋 AI の実験ノート（入力特徴量の数値の表現方法）. https://tadaoyamaoka.hatenablog.com/entry/2019/06/12/230710. (2022/01/19 閲覧).

[5] leela-zero. README.md. https://github.com/leela-zero/leela-zero/blob/e3ed6310d33d75078ba74c3adf887d18439fc2e3/README.md. (2022/01/19 閲覧).

[6] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[8] Tadao Yamaoka. 強化学習におけるバッチサイズとエントロピー正則化. https://tadaoyamaoka.hatenablog.com/entry/2019/05/10/234328. (2022/02/09 閲覧).

[9] Tadao Yamaoka. 第 2 回世界将棋 AI 電竜戦バージョン. https://github.com/TadaoYamaoka/DeepLearningShogi/releases/tag/denryu2021. (2022/01/19 閲覧).

[10] Tadao Yamaoka. Merge feature/pv_mate. https://github.com/TadaoYamaoka/DeepLearningShogi/commit/4f4d7eed209e7183384ee13425d4e3e42e1dcec5. (2022/01/19 閲覧).

[11] L.V. Allis. *Searching for solutions in games and artificial intelligence*. PhD thesis, Maastricht University, January 1994.

[12] 長井歩, 今井浩. df-pn アルゴリズムの詰将棋を解くプログラムへの応用. 情報処理学会論文誌, Vol. 43, No. 6, pp. 1769–1777, Jun 2002.