Policy-Value Monte-Carlo 木探索を用いた将棋の解説文に出現する手順の予測

池田大雅 [1]　鶴岡慶雅 [2]
[1] 東京大学工学部電子情報工学科　[2] 東京大学大学院情報理工学系研究科電子情報学専攻
{ikeda, tsuruoka}@logos.t.u-tokyo.ac.jp

概要

将棋などの完全情報ゲームのプログラムは任意の盤面から最善と考えられる指し手とその評価値（形勢判断）を正確に提示できるが、その理由まで説明することは難しく解釈性が課題となっている。本研究では解説文に出現する手順を予測する部分ゲーム木（解説木）をゲームエンジンを用いて生成する手法を提案する。Policy-Value Network を用いた Monte-Carlo 木探索 (PV-MCTS) によって得られたゲーム木を再帰的に枝刈りして解説木を生成する。この手法により、解説文のコーパスに依存しない解説木の予測が可能となる。また、PV-MCTS を制御する PUCT アルゴリズムを調整することによる予測精度の変化についても調査した。

1　はじめに

近年、将棋・チェスなどの完全情報ゲームのコンピュータプログラムは著しく棋力が向上している。特に AlphaGo [1], AlphaGo Zero [2], AlphaZero [3] などのように、教師あり学習または強化学習で深層学習を行うプログラムが台頭している。このようなプログラムは任意の盤面から評価値や最善の手順を正確に示すことができる一方で、指し手が最善である理由などの説明は難しく、解釈性に乏しいという課題がある。

将棋やチェスなどの解説文生成については、ルールベースの手法 [4], [5], [6] や n-gram 言語モデルを利用する手法 [7] が提案されているが、近年では大規模言語モデル (Large Language Model, LLM) の発展により、強力なゲームエンジンと LLM を組み合わせてゲームの解説文を出力する研究が進んでいる。例えば Lee ら [8] は盤面からタグと呼ばれる情報の単位を抽出し、盤面とともに LLM に入力する手法を提案した。この手法は人間的に好ましい解説文の生成に有効であった一方で、深い手順の説明が不正確になる場合があることが課題とされた。LLM に入力する盤面に関する情報を工夫することで、より正確な解説文を出力できる可能性がある。

本研究では、解説木と呼ばれるデータ構造に着目する。解説木とは、将棋などの完全情報ゲームにおいて、ある盤面の解説文中に現れた手順をゲーム木と呼ばれる木構造で示したものである [9]。解説手順をゲーム木と同様に表現することにより、自然言語による解説と局面の遷移を対応づけることが可能になる。例として、2024 年の第 9 期叡王戦五番勝負第 1 局より図 1 の盤面と図 2 の解説文[1)]が与えられる場合、解説文中の局面の遷移は図 3 の解説木で表現することができる[2)]。

亀甲ら [10], [11] はプロ棋士の対局と解説文のコーパスを教師データとして解説木を生成する手法を提案した。本研究では、Policy-Value Network (PVN) を用いた Monte-Carlo 木探索 (Policy-Value Monte-Carlo Tree Search, PV-MCTS) によって生成したゲーム木を再帰的に枝刈りすることで、ゲームエンジンを用いて解説木を生成する手法を提案する。本研究の手法は、解説木の生成に解説文のコーパスを必要としない点において従来の手法と異なるため、解説文が豊富に存在しない完全情報ゲームに応用できる可能性がある。

2　Policy-Value Monte-Carlo 木探索

2.1　Policy-Value Network (PVN)

PVN [3] は、Convolutional Neural Network (CNN) の一種である Residual Network (ResNet) [12] がベースとなっているニューラルネットワークであり、

1）　http://live.shogi.or.jp/eiou/kifu/9/eiou202404070101.html
2）　解説木に出現する手順は符号表現ではなく、SFEN 形式（チェスの符号表現である FEN 方式を将棋に応用したもの）に変換した状態で保存される。

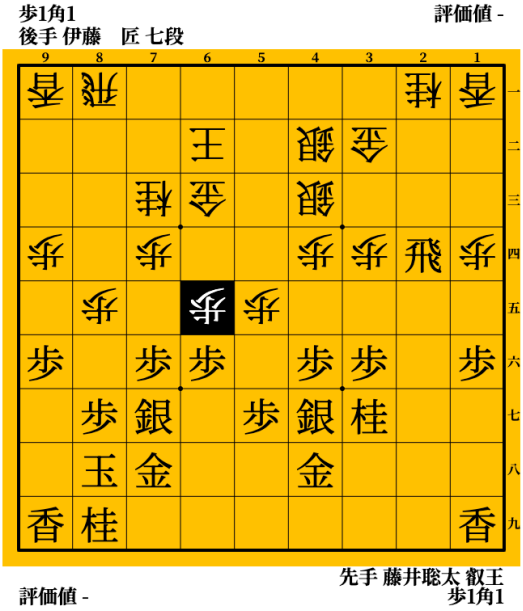

図 1　将棋の盤面の一例（第 9 期叡王戦五番勝負第 1 局 66 手目、▲藤井聡太叡王対△伊藤匠七段）

飛車が走ってきたところに△６五歩と反発。それを見て藤井は羽織を脱ぎ、丁寧に畳んで左後方に置く。▲６五同歩は△同桂で後手の攻めを呼び込む。以下、▲６六銀は△８六歩▲同歩△５九角、▲６八銀は△６六角▲７七桂△７五歩の要領で攻められる。次に△６六歩は駒に当たらないので、先手はこの手番で有効な手を指したい。

図 2　図 1 の局面に対応する解説文の一例

AlphaZero などのゲーム AI のゲーム木探索に使用されている。PVN の入力はゲームの盤面状態 s であり、出力は方策ベクトル (Policy, $\mathbf{p}$) および価値 (Value, v) の 2 つである。

$$(\mathbf{p}, v) = f_\theta(s) \tag{1}$$

方策ベクトル $\mathbf{p} \in \mathbb{R}^n$ は現局面から予測される次の指し手の確率分布であり、価値 v は現局面から予測される期待勝率を表している。本研究では棋譜のデータベースを取得し、教師あり学習によって PVN の学習を行う。

2.2　PV-MCTS の手順

PV-MCTS [3] は Monte-Carlo 木探索 (Monte-Carlo Tree Search, MCTS) の一種で、AlphaZero などで用いられる最善手の探索アルゴリズムである。従来の MCTS と比較すると、末端ノードで勝率のシミュレーションを行う代わりに PVN の出力結果を用いることから計算量を削減することができる。

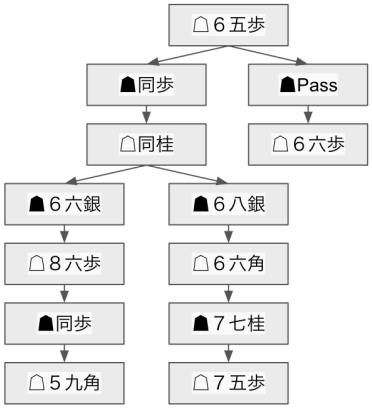

図 3　図 2 の解説文から抽出される解説木

PV-MCTS の手順は以下の通りである。

1. 初期状態: 現局面 s_0 に対応する根ノードおよび合法手に対応する子ノードのみを持つゲーム木を用意する。
2. 以下の手順を T 回繰り返す。
   (a) 選択: ゲーム木の末端ノードに到達するまで、PUCT スコアが最も高い指し手 a を再帰的に選択する。
   (b) 評価: 末端ノードの局面での価値 v を PVN から取得する。
   (c) 展開: 末端ノードの局面での合法手にあたる子ノードをゲーム木に追加する。
   (d) 更新: (a) で辿った経路上の各ノードの累計価値 $W(s,a)$ に v を加算し、訪問回数 $N(s,a)$ に 1 を加算する。
3. 最終的なゲーム木の子ノードのうち、訪問回数が最も多いノードを最善手とする。

PUCT スコアは主に v に依存する予測勝率の平均 $Q(s,a)$ と主に $\mathbf{p}$ に依存する $U(s,a)$ の和で表される。

$$\begin{aligned}\text{Score} &= Q(s,a) + U(s,a) \\ &= \frac{W(s,a)}{N(s,a)} + c_{puct} P(s,a) \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}\end{aligned} \tag{2}$$

式 (2) において $W(s,a)$ はノードの累計価値、$P(s,a)$ はノードの選択確率、$N(s,a)$ はノードの訪問回数[3)]を表している。定数 c_{puct} は、$Q(s,a)$ と $U(s,a)$ のバランスを調整する役割がある。

3　提案手法

PV-MCTS の探索結果として得られるゲーム木を枝刈りすることで局面 s_0 から解説木 T_c を生成する

3)　3 章以降は簡単のため n と表す。

手法を提案する。

3.1 PVN の学習用のデータセット

PVN の学習には、将棋のプログラム同士による対局のプラットフォームである Floodgate[4] から棋譜を引用した。棋譜は 2015 年 1 月から 2023 年 12 月のものを採用し、50 手未満の極端に短い棋譜およびレーティングが 3500 に満たないプログラムによる対局を除外した。また、投了・入玉宣言・千日手以外によって対局が異常終了した棋譜も除外している。学習に用いた対局数は 103,389 局、盤面数は 15,623,244 となった。

3.2 PV-MCTS の実装

Floodgate の棋譜で学習した PVN を使用して PV-MCTS を実行する。本実験の実装では PV-MCTS のステップ数 T を管理し、$T > 1000$ が満たされた時に探索を終了している[5]。

また、式 (2) で示した PUCT アルゴリズムの c_{puct} を小さくすれば v に、大きくすれば $\mathbf{p}$ に重きを置くように探索が進められると考えられる。本実験では解説手順の予測に最適な c_{puct} の値を検証する[6]。

3.3 枝刈りアルゴリズム

PV-MCTS によって生成されるゲーム木をそのまま解説木として用いるのは、ノード数が大きすぎる[7]ため不適である。ゲーム木の重要な手順のみを残しつつ解説木に適したノード数に減らすために、以下のアルゴリズムでゲーム木を再帰的に枝刈りし、訪問回数 n が $\alpha \cdot T$（α は閾値、$0 < \alpha < 1$）よりも多いノードのみを残す。PV-MCTS では尤もらしい手順を優先的に探索するため、n の大きいノードほど PV-MCTS によって高く評価されており、重要度も高いと考えられる。

1. ゲーム木の根ノードで開始する。
2. ゲーム木の現ノードについて各子ノードの探索回数 n を取得し、探索率 $\frac{n}{T}$ を計算する。
3. 各子ノードについて、探索率 $\frac{n}{T}$ と閾値 α を比較し、以下の操作のうち該当するものを行う。
   (a) $\frac{n}{T} < \alpha$ ならば、子ノードおよびその傘下のノードを枝刈りする。
   (b) $\frac{n}{T} > \alpha$ ならば、その子ノードに移動して手順 2〜3 を再帰的に行う。
4. 枝刈りされなかったノードの集合が求める解説木 T_c となる。

上記のアルゴリズムを実行すると、ゲーム木のノードのうち $n < \alpha \cdot T$ となるものがすべて枝刈りされる。これは $n > \alpha \cdot T$ を満たすゲーム木のノードのみが解説木に追加されることと等価になる。実際の実装でもゲーム木を枝刈りするのではなく、根ノードのみの解説木を用意して $n > \alpha \cdot T$ であるノードを追加している (Algorithm 1)。

Algorithm 1 Pruning of Monte-Carlo trees

```
function PRUNE_MCT(tree T_c, state s)
    for child_node in s.child_nodes do
        if child_node.n > α * T then
            Expand T_c with child_node
            Prune_MCT(T_c, child_node)
        end if
    end for
end function
function GENERATE_COMMENTED_TREE(state s_0)
    Initialize T_c with board state s_0
    Prune_MCT(T_c, s_0)
    return T_c
end function
```

4 評価

4.1 解説木抽出用のデータセット

提案手法によって生成された解説木が実際の解説文に出現する手順をどれほど正確に予測できているかを評価するために、実際の解説文から解説木を抽出した。評価に用いた解説文は 2024 年の第 9 期叡王戦五番勝負の第 1 局〜第 5 局のものである。解説木を抽出する際には現局面を除く局面数が 3 未満である極端に短い解説文は除外し、現局面以前の分岐（代えて▲7 三歩成は〜となっていた）やパスが含まれる手順（詰めろ・必至の解説など）も除外している。解説文から抽出された解説木の数は 87、解説木の平均ノード数は 7.30 となった。

4） http://wdoor.c.u-tokyo.ac.jp/shogi/floodgate.html
5） PV-MCTS のステップは並列で処理されるため、厳密に $T = 1000$ で探索が停止するとは限らない。
6） PV-MCTS を用いるゲーム AI では $c_{puct} = 1$ で実装される例が多い ([13], [14])。
7） PV-MCTS では $n > 0$ であるノード数が T と等しくなる。

図 4　開始局面から生成される解説木の一例

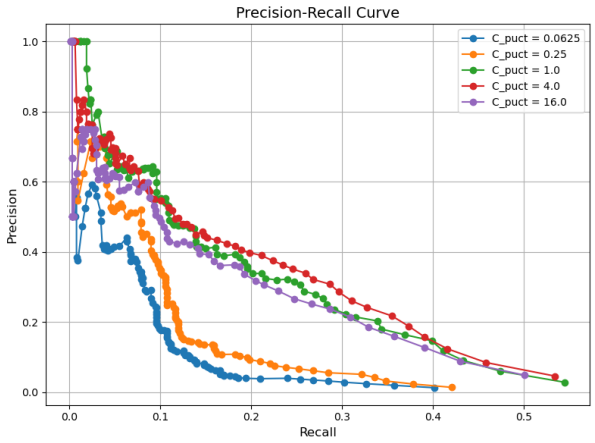

図 5　c_{puct} の各値における Precision-Recall 曲線

4.2　解説木の定量的評価

生成される解説木を定量的に評価するために、解説木に出現する手順のリストを取得する。解説木の各ノードについて、根ノードからのパスを取得して手順のリストに追加する。例えば開始局面から図 4 のような解説木が生成された場合、手順のリストは [[7g7f], [7g7f, 3c3d], [7i7h]] のように表現される。

提案手法により生成された解説木が正しく予測できた手順を正例、それ以外の手順を負例とし、Precision、Recall およびそれらの調和平均である F_1 値を取得する。また、枝刈りの閾値 α を 0.01〜1 の範囲で調節し、Precision-Recall 曲線を取得する。なお、α を大きくすると枝刈りされるノードが単調に増加するため、生成される解説木のサイズおよび Recall は単調に減少する。

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

$c_{puct} = 0.0625, 0.25, 1, 4, 16$ における Precision-Recall 曲線を図 5 に示した。また、c_{puct} の各値について、最大の F_1 値およびその時の閾値 α を表 1 に示した。本実験で抽出した解説木の評価では、$c_{puct} = 4, \alpha = 0.10$ の F_1 値が 0.297 で最も高かった。また、$c_{puct} = 4$ で生成される解説木の平均サイズも抽出された解説木と最もよく一致した。

PVN による選択確率 $P(s,a)$ が高い指し手は、深い読みなどは考慮せず直感的に良い指し手である可能性が高いと考えられる。そのため、c_{puct} をある程度大きくして $P(s,a)$ の高い指し手の探索を優先することで、最善の手順だけでなく直感的に優れていて解説文に採用されやすい手を解説木に含むことができていると考えられる。

表 1　c_{puct} の各値における最大の F1 値、およびその時の閾値 α と解説木の平均サイズ

c_{puct}	0.0625	0.25	1	**4**	16
F_1 score	0.138	0.160	0.278	**0.297**	0.259
α	0.56	0.51	0.16	**0.10**	0.07
Size	2.90	2.36	6.03	**6.80**	8.84

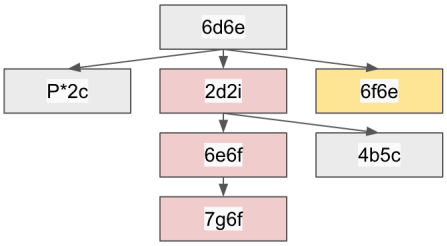

図 6　Monte-Carlo 木の枝刈りによって生成される解説木の例。黄色のノードは実際の解説文に出現した手順、赤色のノードは実際の棋譜に出現した手順である。

4.3　生成される解説木の例

本実験にて最も高い F_1 値が得られた条件 $c_{puct} = 4, \alpha = 0.10$ にて解説木を予測し、実際の解説文から抽出された解説木との比較を行う。例えば図 1 の局面からは図 6 のような解説木が予測される。図 3 の解説木と比較すると、[6f6e]（▲6 五同歩）など解説文に出現する手順をある程度予測できていることが確認できる。また、[2d2i, 6e6f, 7g6f]（▲2 九飛、△6 六歩、▲6 六同銀）のように、解説文には出現しないものの実際の棋譜の進行を予測できている手順もある。

5　おわりに

本研究では、PV-MCTS で生成されるゲーム木を枝刈りすることで解説文に出現する手順を予測する手法を提案した。この手法は、PVN を棋譜から学習すれば解説文のコーパスが不要である点で従来の手法と異なる。そのことから、解説文が十分に存在しない完全情報ゲームにも応用できる可能性がある。また、PUCT アルゴリズムの定数 c_{puct} を調整することで解説文の予測精度が向上する可能性があることを示した。

参考文献

[1] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. **Nature**, Vol. 529, pp. 484–503, 2016.

[2] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, L. Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. **Nature**, Vol. 550, pp. 354–359, 2017.

[3] 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.

[4] Aleksander Sadikov, Martin Možina, Matej Guid, Jana Krivec, and Ivan Bratko. Automated chess tutor. **Lecture Notes in Computer Science**, p. 13–25, 2007.

[5] Tomoyuki Kaneko. Evaluation of real-time commentary generated by computer shogi program. **Transactions of Information Processing Society of Japan**, Vol. 53, pp. 2525–2532, 2012.

[6] Koki Ishiwaki and Tatsuya Arakawa. Automatic generation of novice-oriented tsume-shogi commentary including information about seemingly good moves. **IPSJ SIG Technical Report**, Vol. 2015-GI-34, pp. 1–7, 2015.

[7] Hirotaka Kameko, Shinsuke Mori, and Yoshimasa Tsuruoka. Learning a game commentary generator with grounded move expressions. **2015 IEEE Conference on Computational Intelligence and Games (CIG)**, Vol. 53, p. 177–184, Aug 2015.

[8] Andrew Lee, David Wu, Emily Dinan, and Mike Lewis. Improving chess commentaries by combining language models with symbolic reasoning engines. **arXiv e-prints**, p. arXiv:2212.08195, December 2022.

[9] Hirotaka Kameko, Shinsuke Mori, and Yoshimasa Tsuruoka. Mapping between move expressions and positions for shogi commentary grounding. **The 19th Game Programming Workshop 2014**, Vol. 2014, pp. 202–209, 2014.

[10] Hirotaka Kameko, Shinsuke Mori, and Yoshimasa Tsuruoka. Predicting moves in comments using realization probabilities. **The 21st Game Programming Workshop 2016**, Vol. 2016, pp. 28–35, 2016.

[11] Hirotaka Kameko, Shinsuke Mori, and Yoshimasa Tsuruoka. Predicting moves in comments for shogi commentary generation. **Transactions of Information Processing Society of Japan**, Vol. 58, pp. 2070–2079, 2017.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, June 2016.

[13] 布留川英一. AlphaZero 深層学習・強化学習・探索人工知能プログラミング実践入門. Born Digital, Inc., 第 1 版, 2019.

[14] 山岡忠夫, 加納邦彦. 強い将棋ソフトの創りかた Python で実装するディープラーニング将棋 AI. マイナビ出版, 第 1 版, 2021.