お前、CSA 会員にならねーか？ アピール文書

ザイオソフト コンピュータ将棋サークル
野田久順 岡部淳 鈴木崇啓
河野明男 伊苅久裕

目次

- お前、CSA 会員にならねーか？
- 改良点
- 使用ライブラリ

お前、CSA 会員にならねーか？

- 奈川トモ先生の漫画『お前、タヌキにならねーか？』が元ネタです。

『お前、タヌキにならねーか？』をイメージして AI で生成した『お前、CSA 会員にならねーか？』のイメージ画像

改良点（1）

- nnue-pytorch を用いて学習しています。
  - nnue-pytorch は NNUE 評価関数の学習器です。
  - nnue-pytorch は、コンピュータチェス思考エンジン『Stockfish』の開発チームが開発しました。
  - コンピュータ将棋思考エンジンの NNUE の学習に対応させるため、やねうら王を組み込んでいます。
  - レーティングが向上し、学習時間も短くなりました。

改良点（2）

- Optimizer に Momentum SGD を使用しています。
  - Momentum なしで学習させた評価関数と比べ、R30.2 程度レーティングが高くなることを確認しました。

改良点（3）

- Suisho10Mn_psv を用いて Fine-tuning しています。
  - Suisho10Mn_psv は水匠シリーズ開発者 たややん様が公開されている学習データです。
  - 学習率を 1e-7 、λ＝0.0 で学習しています。
    - λ＝0.0 は勝敗項のみを見る設定です。

使用ライブラリ

- やねうら王
  - やねうら王を元に改造した思考部を使用しています。
    - 独自の工夫を加えるにあたり、改造しやすく、レーティングも高いためです。
- 水匠
  - 学習データを使用しています。
    - Fine-tuning により、レーティングが高くなることを確認したためです。
- tanuki-
  - 棋譜の生成に使用ています。
    - 過去に開発した資産の再利用のためです。
- nnue-pytorch
  - 評価関数の学習に使用しています。
    - レーティング向上と学習時間の高速化のためです。

大事なことは、CSA 会員になったら見つかるかも。

第34回世界コンピュータ将棋選手権 「東横将棋」アピール文書

2024.3.31

東横コンピュータ将棋部

定跡とNNUE評価関数の極北を目指しています。

・従来の強化学習手法に加え独自にNNUE評価関数を強化。今回からマメットブンブク評価関数（halfkp_1024x2-8-32）を使用しています。

・手作業による定跡の生成。角換わり定跡と相掛かり定跡に主眼を置いています。

・探索部はやねうら王、いわゆるやねうら王チルドレンです。最新のやねうら王の使用を予定しています。

よろしくお願いいたします。

ノミというのがおりますなｗｗｗちっぽけな虫ケラのノミですなｗｗｗ

あの虫は我々巨大で頭のいい人間にところかまわず攻撃を仕掛けて戦いを挑んでくるんですなｗｗｗ

巨大な敵に立ち向かうノミｗｗｗこれは「勇気」と呼べるんですかなｗｗｗ

ノミどものは「勇気」とは呼べませんなｗｗｗ

では「勇気」とはいったい何なんですかなｗｗｗ

「勇気」とは「怖さ」を知ることｗｗｗ「恐怖」を我が物とすることなんですなｗｗｗ

人間讃歌は「勇気」の讃歌ｗｗｗ人間のすばらしさは勇気のすばらしさｗｗｗ

いくら強くてもこいつら屍生人は「勇気」を知らんのですなｗｗｗ

ノミと同類ですなｗｗｗ

んんんｗｗｗ

今年も性懲りもなく出るんですなｗｗｗ

・役割論理とは

第33回世界コンピュータ将棋選手権で新人賞を受賞する快挙()で完全かつ最終的に確立された論理ですなｗｗｗ

しょぼい定跡とマメットブンブク評価関数の強化と元高級スリッパ（粗大ゴミ）の微妙な火力によって評価値ダメージレースを制する必勝の戦術ですなｗｗｗ

クラウド重課金(笑)はもうやめましたぞｗｗｗ
おうちパソコン3990Xでどこまでやれるか見ものですなｗｗｗ

・定跡について
floodgateその他で収集した棋譜やら何やら適当にぐちゃまぜにして脳を焼きながら手作業で調整修正した居飛車定跡「1.21ジゴワット火葬砲定跡」を使用しますぞｗｗｗ

・戦型について
角換わり、相掛かり共に先手必勝が確定しましたなｗｗｗ後手番は対策が急務となっておりますぞｗｗｗ

ぶっちゃけ対策なんかできませんなｗｗｗ
しっかり定跡をキメてこられるとまったく太刀打ちできませんぞｗｗｗありえないｗｗｗ
特に角換わりは絶対に拒否した方がいいですなｗｗｗ

ちなみに振り飛車は先手後手ともに必敗ですなｗｗｗ埴輪定跡は徹底的に対策するしかありえないｗｗｗ

横歩取り：先手必勝ですなｗｗｗ
一手損角換わり：先手必勝ですなｗｗｗ
雁木：先手番でわざわざ指す必要はなさそうですなｗｗｗえぐいですなｗｗｗ
矢倉：先手番でわざわざ指す必要はなさそうですなｗｗｗ矢倉は終わりましたｗｗｗ
相振り飛車：なんで相手がありがたくも不利飛車を指してくれているのにこちらも振ってやる必要があるんですかなｗｗｗありえないｗｗｗ
筋違い角：後手の振り飛車党への嫌がらせの精神攻撃ですなｗｗｗそれ以上でもそれ以下でもありませんぞｗｗｗだいたい負けますなｗｗｗ

まずは後手番でどこまでダメージを最小限にするかが重要ですなｗｗｗ
現状ほぼ先手後手が同じ割合になるらしいので後手番でも勝てそうな相手(酷い)に必然力で対戦すること

が重要ですぞｗｗｗ

互角の分かれで逃げられればNNUE型評価関数の終盤の強さと元高級スリッパの火力で踏み潰すだけですなｗｗｗ

もちろん定跡を整備しなければdl系やや○こま王()や水○匠などの強豪には手も足も出ませんぞｗｗｗ

・必然力とは

論者を圧倒的勝利に押し上げる力ですなｗｗｗ

強豪に 絶 対 に 先手を引く、後手番での当たりが良いｗｗｗ裏街道最高ｗｗｗなどは必然力とされていますなｗｗｗ

ヤーティ神への信仰によって得られる加護とされていますぞｗｗｗ

やはりこれが最も重要なファクターとなっていますなｗｗｗ

結局「評価関数の強化」と「定跡の強化」という極めて当たり前の結論に至る訳ですなｗｗｗ

将来的にはdl系とマメットブンブク形式（halfkp_1024x2-8-32）のハイブリッド＋強力な定跡が主流になっていくのではないですかなｗｗｗ知らんけどｗｗｗ

現時点では定跡強化が勝利の鍵になりそうですなｗｗｗ

・評価関数について

ついに待望のマメットブンブク評価関数（halfkp_1024x2-8-32）を使用しますぞｗｗｗPytorchｗｗｗ

もうどこもhalfkp_1024x2-8-32ばかりで特にアドはありませんなｗｗｗ入玉どうするんですかなｗｗｗ

知らんがなｗｗｗ

なお標準NNUE評価関数はもう完全にサチっててオワコンでもはや観る将が藤井聡太の将棋を観戦する時にしか使い道がありませんなｗｗｗ

標準NNUEで粘る場合でも水匠5はもちろんHaoやBLOSSOMより強くないとお話になりませんなｗｗｗゴミｗｗｗ

ﾅﾝﾄｶ改？知りませんなｗｗｗ

もちろん振り飛車評価関数は総合的にロジックするまでもなくありえないｗｗｗ

・使用予定の評価関数

Grampus_HD2_20220228：とんいる人民共和国()でこの世の誰も体験したことのない最も厳しい無慈悲な鉄槌を受けたマメットブンブク評価関数ですなｗｗｗ

誰の評価関数なんですかなｗｗｗ

もう評価関数に手を付けている暇などなさそうなので苦渋の選択ですぞｗｗｗ

やっぱりちょっとだけ手を入れる予定ですぞｗｗｗ

・シードについて

今回は第6シードですなｗｗｗ感謝しかありえないｗｗｗ

全体の実力が底上げされているので一次予選から修羅場になる可能性も十分にありますなｗｗｗそして1枠は某7995WX枠が確定しておりますぞｗｗｗ

・東横将棋について

A.東横将棋はGrampusですか？

Q.違いますなｗｗｗ東横将棋は東横将棋であってそれ以上でもそれ以下でもありませんぞｗｗｗ

十六式いろは煌（きらめき）

第 34 回世界コンピュータ将棋選手権
アピール文

メンバー

末吉竜介、（増える予定）

「十六式いろは 煌」の由来

昨年 2022 年に 2 期生の先輩達が決めた「十六式いろは煌 (きらめき) 」を今年も引き継ぎます。

以下、 2022 年当時の由来の記載です。

様々な名前の候補が上がり最終的に決まったのが考え始めてからなんと 1 か月かかりました！。
皇 (すめらぎ) 、煌 (きらめき) 、日本工学院、かまトゥ（学校のマスコットキャラ）…などなど。
「日本工学院の名前があった方がよいのではないか」や
「ローマ字で書いた方がかっこいい！」などかなりの意見などがありましたが
最終的には末吉先生の「十六式いろは」と生徒達で考えた「煌 (きらめき) 」を
組み合わせて決定しました！

ソフトの概要

採用予定

- やねうら王
- dlshogi
- KomoringHeights
- Electron 将棋

ソフトの説明（予定）

- やねうら王での評価関数は第４回電竜戦の
「十六式いろは煌（きらめき）」内部の
「十六式いろは幻（まほろ）」を改良したもの。
探索速度重視で標準 NNUE から軽量化する予定
- dlshogi のネットワークモデルは昨年に引き続き軽量なもの
（GhostNet・ゴーストネット）を採用
- 定跡ファイルは floodgate、wcsc、電竜戦、AobaZero の棋譜を
利用して作成する
- 詰将棋エンジン（KomoringHeights・コーモリンハイツ？）を
含めて合議制の見直し

2024.03.31

wcsc34 での実装について

（次のページから記します）

基本的な動作

―昨年 2022 年の wcsc32 と同様。

やねうら王（★１）と dlshogi （★２）、詰将棋エンジンの各エンジンによる合議がこのソフトの最大の特徴。 Ayane を使用して両エンジンを呼び出し、評価値を比較して指し手を決める。

★ １　評価関数は、第 4 回電竜戦の「十六式いろは煌（きらめき）」内部の評価関数「十六式いろは幻（まほろ）」（やねうら王の評価関数）を改良したもの。
★ ２　ネットワークモデルは第 4 回電竜戦の「十六式いろは煌（きらめき）」内部のネットワークモデル「十六式いろは幽（ほのか）」（ dlshogi のネットワークモデル）

wcsc33 時との違い

1) やねうら王の評価関数の変更（十六式いろは幻（まほろ））
2) dlshogi のネットワークモデルの変更（十六式いろは幽（ほのか））
3) 定跡ファイルの変更（内容は 03/31 現時点では未定）
4) 詰将棋エンジンを追加

変更１）やねうら王の評価関数の変更

第 4 回電竜戦での「十六式いろは幻（まほろ）」（★）から
さらに自己対戦で作成した教師局面データで追加学習したもの。
探索速度の向上を目的にし標準 NNUE ではなく軽量化した NNUE にする予定。

（★）十六式いろは幻（まほろ）
既存の教師局面データを使わず、自己対戦と学習を繰り返して
合計 50 億局面の教師局面データから学習したものに、
水匠 5 、 Hao 、 BLOSSOM で補完し、さらに自己対戦＆追加学習したもの。

変更２） dlshogi のネットワークモデルの変更

「十六式いろは幽（ほのか）」

探索速度の向上を目標に、軽量化と精度維持を目指したネットワークモデル。

具体的には、精度を維持しつつパラメータ数を大幅に削減し軽量化した
GhostNet （ゴーストネット）をメインに、
SENet （エスイーネット）、 Bottleneck （ボトルネック）を追加したもの。

変更３）定跡ファイルを新規で作成

WCSC、電竜戦、 AobaZero、 floodgate の棋譜を元に作成する予定。
方針は現時点（2024-03-31）では未定。

変更４）詰将棋エンジンを追加

詰将棋エンジン KomoringHeights も搭載する予定。

棋力（wcsc33 後に floodgate で測定）

十六式いろは煌（きらめき）　wcsc33（一次予選　7 位）　：　R3524
　マシン： GALLERIA XF （Core i7-9700K , GeForce RTX 2070 SUPER）

以下、内部エンジン単独
　十六式いろは幻（まほろ） - 第 4 回電竜戦：
　　VS Hao-wcsc33　：R-123.8
　　VS Suisho5　　：R-112.9
　十六式いろは幽（ほのか）：　未測定

使用ソフト、参考文献等、その１

sueyoshiryosuke/16shiki-Iroha_kirameki:
https://github.com/sueyoshiryosuke/16shiki-Iroha_kirameki
←wcsc32 時のもの。

TadaoYamaoka/DeepLearningShogi
https://github.com/TadaoYamaoka/DeepLearningShogi

yaneurao/YaneuraOu
https://github.com/yaneurao/YaneuraOu

Electron 将棋
https://sunfish-shogi.github.io/electron-shogi/

Home · yaneurao/YaneuraOu Wiki
https://github.com/yaneurao/YaneuraOu/wiki

tttak/GougiShogi: 合議将棋
https://github.com/tttak/GougiShogi

使用ソフト、参考文献等、その２

オープンソースの詰将棋エンジン「KomoringHeights」を作った・コウモリのちょーおんぱ
https://komorinfo.com/blog/komoring-heights/

yaneurao/Ayane:
https://github.com/yaneurao/Ayane

AobaZero
http://www.yss-aya.com/aobazero/

floodgate
http://wdoor.c.u-tokyo.ac.jp/shogi/floodgate.html

次世代の将棋思考エンジン、 NNUE 関数を学ぼう（その１．ネットワーク構造編）
- コンピュータ将棋 Qhapaq
https://qhapaq.hatenablog.com/entry/2018/06/02/221612

tanuki- 2022-06-07 やねうら王学習部リグレッション調査 - nodchip のブログ
https://nodchip.hatenablog.com/entry/2022/06/07/000000

使用ソフト、参考文献等、その３

人間の棋譜を用いずに評価関数の学習に成功 | やねうら王 公式サイト
https://yaneuraou.yaneu.com/2017/06/12/%E4%BA%BA%E9%96%93%E3%81%AE%E6%A3%8B%E8%AD%9C%E3%82%92%E7%94%A8%E3%81%84%E3%81%9A%E3%81%AB%E8%A9%95%E4%BE%A1%E9%96%A2%E6%95%B0%E3%81%AE%E5%AD%A6%E7%BF%92%E3%81%AB%E6%88%90%E5%8A%9F/

lambda 混合絞りについて | やねうら王 公式サイト
https://yaneuraou.yaneu.com/2017/08/21/lambda%E6%B7%B7%E5%90%88%E7%B5%9E%E3%82%8A%E3%81%AB%E3%81%A4%E3%81%84%E3%81%A6/

テラショック定跡の生成手法 | やねうら王 公式サイト
https://yaneuraou.yaneu.com/2019/04/19/tera-shock-book-generation/

ShogiGUI
http://shogigui.siganus.com/

ai5/BookConv
https://github.com/ai5/BookConv

使用ソフト、参考文献等、その 4

ak110/Blunder.Converter: 棋譜変換ツール。
https://github.com/ak110/Blunder.Converter

各種将棋ソフト間での教師データの変換ツールの開発 - コンピュータ将棋 Qhapaq
https://qhapaq.hatenablog.com/entry/2017/12/25/002820

たややん /ToSfenpack20210122
https://twitter.com/tayayan_ts/status/1338443561272950787?s=20

将棋 AI の進捗 その 31(cuDNN による SENet の推論処理の実装）
- TadaoYamaoka の開発日記
https://tadaoyamaoka.hatenablog.com/entry/2019/07/24/011150

強い将棋ソフトの創りかた | マイナビブックス
https://book.mynavi.jp/ec/products/detail/id=126887

精度を維持したままパラメータ数を大幅に削減「 GhostNet 」 | AI-SCHOLAR | AI ：（人工知能）論文・技術情報メディア
https://ai-scholar.tech/articles/image-recognition/ghostnet-ai-383

Polonaise アピール文書

谷合廣紀

2024 年 5 月 4 日

1　独自の工夫

基本は dlshogi/ふかうら王などのいわゆる DL 系で採用されている、policy+value Network + MCST のアプローチを取っています。dlshogi/ふかうら王と大きく異なるのは、モデル構造とその入出力です。

1.1　モデル入力のエンコード

モデルに盤面を入力するにあたって、まずは盤面情報を数値行列である入力特徴量に変換する必要があります。dlshogi では駒の位置や利きなどを 9x9 の 2 次元行列にエンコードしていき、最終的に 9x9x 特徴数の大きさを持つ入力特徴量を得ています。この入力特徴量は CNN を使い推論されていくため、dlshogi は画像処理的なエンコードと捉えることができます。

一方の Polonaise では、盤面を 1 一から順に見ていき、1 一、1 二 ･･･ 9 九の駒と先後の持ち駒（7 種 x2）を並べた 95 字の文字列にエンコードすることで入力特徴量を得ます。この入力特徴量はモデルの最初の層で埋め込み層によりベクトルに変換されて推論されていくため、自然言語処理的なエンコードと捉えることができます。

具体的に Polonaise のエンコード方法を図1の盤面を使って示します。

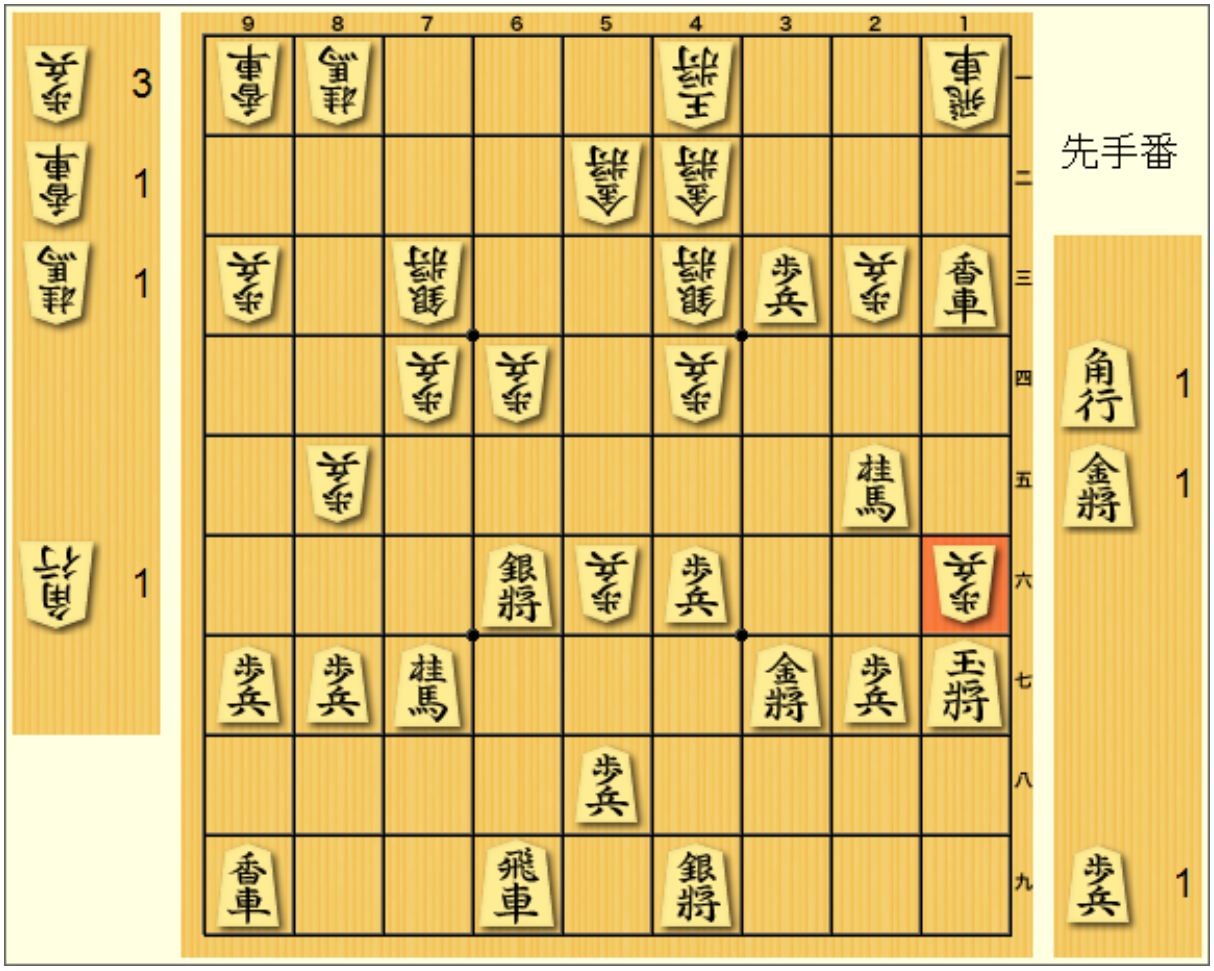

図1: 入力盤面

この盤面の駒を 1 一から順に見ていくと、「後手の飛車」、「空きマス」、「先手の香車」･･･ と続きます。また、持ち駒は先手は歩が 2 枚、金が 1 枚、角が 1 枚です。盤面 81 マスの情報はそのまま駒情報が入り、持ち駒はそれぞれの駒を何枚持っているかが入ります。したがって図1をエンコードすると、図2の文字列が得られます。

実際には文字列だと扱いづらいため、各文字が対応する数値 ID に変換されて、95 個の数値列がエンコードされた入力特徴量となります。

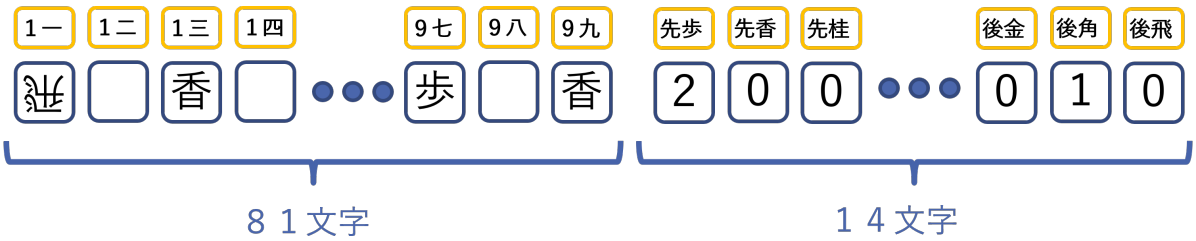

図2: 文字列エンコード

1.2　モデル構造

95 個の数値列は、最初の埋め込み層によって 95x256 の大きさの行列に変換されます。これまでは gMLP[?] と呼ばれる MLP ベースのモデルを採用していましたが、今回は BERT-large 程度の中規模なモデルを採用しています。

1.3　モデル出力

dlshogi で採用されている policy の出力は、「着手するマス」と「その駒はどの方向から来たか」の組み合わせで表現されます。「着手するマス」は 81 マスあり、「どの方向から」は 27 通りあるため、その組み合わせは 2187 通りです。したがって policy は 2187 通りのクラス分類問題として表現されています。

しかし、「1 一のマス」に「下がる」や「左に寄る」といった動きは将棋の合法手として存在しません。このように dlshogi の policy 表現の中には決して現れない組み合わせがいくつかあります。それら非合法手を数え上げていくと 691 通りあり、約 32% が非合法手となっていることがわかります。

実験の結果、policy の出力を 2187 クラスの分類問題として解くよりも、非合法手 691 通りを除いた 1496 のクラス分類問題として解いた方が、policy の学習がうまくいくことがわかりました。そのため Polonaise では 1496 のクラス分類問題として policy の学習を行っています。

1.4　PVM ネットワーク (5/5 追記)

モデルの出力として policy, value に加えて mate_prob すなわち入力局面が詰むかどうかを出力しています。この mate_prob がある閾値 (大会では 0.5) を越えたらその局面が詰み探索キューに追加されて、待機している複数の df-pn ソルバーで詰み探索を行い、詰みと判定されたらそのノードの情報を勝ちに更新します。これによって dl 系が苦手とする終盤において、見落としが少なくなったように見えます。(実装できたばかりのため、計測データはなし。)

2　使用ライブラリ・使用データ

- ふかうら王
- AobaZero 公開データ
- Suisho-10Mn 公開データ

参考文献

WCSC34 koron アピール文章

野田煌介

2024/03/28

1 前回までの課題

現在、NNUE 系の将棋 AI では表現力の限界と序盤-終盤の棋力トレードオフが課題になっていると考えられる。

以前の koron はこれらの課題に対して、「ある一定の手数を経過した場合に評価関数を切り替える」といった手法をしばしば採用してきた。

しかしながら、大会結果からも分かる通りこれらは大した成果を上げることは出来なかった。

原因として、局面がどのような状況であるにも関わらず、手数で序盤終盤を判断し、評価関数を切り替えるといった手法に問題があったと考えた。

2 改善点

そこで今回は評価値に応じて評価関数を切り替える手法に変更した。

一定の評価値以内では重いネットワークを使用し、評価値が有利不利どちらかに偏った場合、軽いネットワークを使用する。また評価値に関係なく、どちらかが入玉した場合も軽いネットワークに変更する。

これにより、序盤では精度の向上、終盤では読み抜けの防止や入玉将棋におけに棋力の向上などを両立できると考える。

3 使用ライブラリ

やねうら王

　　優秀な思考エンジンであるため。

nodchip 氏が公開している教師データ群

　　質、量ともに優秀な教師データであるため。

たややん氏が公開している教師データ群

　　Fine Tuning において優秀な教師データであるため。

「技巧」アピール文書

2024 年 3 月 31 日

出村　洋介

1. Gumbel AlphaZero を用いた効率的な強化学習

AlphaZero [1]の学習を効率化した Gumbel AlphaZero [2]を用いて，既存の棋譜を使わずに強化学習でゼロから学習を行なっています.

オリジナルの AlphaZero では難しかったような軽量な実験条件（1 手 16〜64 シミュレーション程度）でも Gumbel AphaZero では比較的安定した学習ができるため，学習に要する計算資源が小さく済むようになり，各種実験を行いやすくなりました.

選手権用には，さらに学習時間を増やしたバージョンで参加予定です.

2. 強化学習時の初期局面を多様化

元々の AlphaZero では強化学習時の初期局面は 1 種類（平手初期局面）のみであり，手元の実験では自己対局で似たような展開になりやすい傾向が見られました.

そこで，今年の技巧では，自己対局時にさまざまな局面を積極的に経験させるため，学習時の自己対局で用いる初期局面の多様化を行なっています. 具体的には，2 手ランダムに指した局面（下図（b））や，駒の配置を一定の制約のもと並べ替えた初期局面（下図(c)）を初期局面として強化学習を行なっています（下図(c)の並べ替えでは駒の配置が左右対称を除き約 81 万通りあるため，「チェス 960」[3]にならって「将棋 81 万」と呼んでいます[4]).

このように強化学習時の初期局面を多様化することにより，実験では平手初期局面のみでの強化学習と比較して同一対局数で一定の勝率向上が確認できています. 学習順序としては，学習初期には「将棋 81 万」（下図(c)）で幅広い形を学習させてから，2 手ランダムの初期局面（下図(b)）で学習を行う方法が調べた中では効果的でした.

(a) 平手初期局面　(b) 2手ランダムの初期局面(例)　(c) 将棋81万の初期局面(例)

3. Rust と Python による独自実装

主要な開発言語には Rust を利用しており，既存の将棋ライブラリを使用しない独自実装

となっています．実装上は以下のような工夫がされています．

・Rust 組込みの 128 bit 整数型を用いた Bitboard

・利き数を保持するデータ構造

・利き情報を活用した高速な 1 手詰関数［5］

・ニューラルネットワークへの入力に将棋独自の特徴量を追加

また，学習部の開発は主に Python を用いて行っており，高速化や安定性向上のために次のような実装上の工夫がされています．

・PyTorch Lightning の混合精度学習による学習の高速化［6］

・Torch-TensorRT を用いた推論の高速化［7］

・Python 側と Rust 側のデータの受け渡し時に安全な NumPy 形式で転送［8］

参考文献

[1] D. Silver, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science, 362(6419): 1140–1144, 2018.

[2] I. Danihelka, et al. Policy improvement by planning with Gumbel. In International Conference on Learning Representations, 2022.

[3] Wikipedia, https://ja.wikipedia.org/wiki/チェス 960.

[4] 出村，金子．将棋 81 万: 強化学習のための多様性を持った将棋初期局面集．第 28 回ゲームプログラミングワークショップ, pp. 111–118, 2023.

[5] 金子，田中，山口，川合．新規節点で固定深さの探索を行う df-pn の拡張．情報処理学会論文誌, 51(11): 2040–2047, 2010.

[6] PyTorch Lightning, https://lightning.ai/docs/pytorch/stable/common/precision_intermediate.html.

[7] Torch-TensorRT, https://github.com/pytorch/TensorRT.

[8] Rust-numpy, https://github.com/PyO3/rust-numpy.

やねこま王チーム PR 文章

ペタショック定跡

以下のブログ記事に書いたように 3000 万局面程度生成した。

将棋 AI の 2024 年は大規模定跡時代
https://yaneuraou.yaneu.com/2024/01/14/the-era-of-large-scale-book-in-shogi-ai/

DL 系の将棋 AI で読み抜けする件の改良

今回、探索部には DL(Deep Learning)系の将棋 AI であるふかうら王(自作の dlshogi 互換エンジン)を用いる。

ふかうら王は AlphaZero 型の将棋 AI である。

AlphaZero 型の DL 系の将棋 AI ではしばしば読み抜けするという問題がある。

これは Policy Network(方策予測のためのニューラルネットワーク)が指し手の実現確率を出力するのだが、好手に対して極めて 0 に近い確率を出力することがあるからだと思う。

このような場合、その指し手の先の変化を全く読まないので読み抜けする。0 に近い確率が出力された場合であっても少しぐらい読むべきだと思うが、そうしてしまうと読まなければならない枝が増えすぎてしまい、Policy Network を用意している意味がなくなってしまう。

そこで特定の条件に合致した場合のみ、0 に近い確率が出力された場合であってもその先の変化を読むようにしようかと思っている。

第 34 回世界コンピュータ将棋選手権
dlshogi with HEROZ アピール文章

山岡忠夫
加納邦彦
大森悠平
2024/3/26

※下線部分は、第 33 回世界コンピュータ将棋選手権からの差分を示す。

1　dlshogi のアピールポイント

dlshogi は、ディープラーニングを使用した将棋 AI である。

2017 年より、AlphaGo の手法を参考に開発を行っている。

ディープラーニング系の将棋 AI は、大局観に優れており、序中盤の形勢判断が従来型将棋 AI と比べて正確であるという特徴がある。一方、終盤の読みが重要になる局面では、従来型将棋 AI の方が正確な場合がある。

dlshogi は、終盤の課題に対処するために、独自の工夫を行っている。

具体的には、「MCTS の葉ノードでの短手数の詰み探索」、「ルート局面で df-pn による長手数の詰み探索」、「勝敗が確定したノードのゲーム木への伝播」、「PV 上の局面に対する長手数の詰み探索」、「強化学習時に初期局面集を使用して局面の多様性を確保する」、「強化学習時に df-pn により詰み探索を行い詰みを報酬とする」という工夫を行っている。

これらのいくつかは、現在、dlshogi 以外のディープラーニング系の将棋 AI には取り入れられているが、dlshogi 以前にこれらを導入しているディープラーニング系の将棋 AI はなかった。

<u>今大会に向けては、新しい手法で定跡の自動生成を行った。</u>

2　チーム参加について

今大会では、HEROZ チームとして参加する。

3　dlshogi の特徴

- ディープラーニングを使用
- 指し手を予測する Policy Network
- 局面の勝率を予測する Value Network
- 入力特徴にドメイン知識を積極的に活用
- モンテカルロ木探索
- 未探索のノードの価値に親ノードの価値を使用
- GPU によるバッチ処理に適した並列化

- 自己対局による強化学習
- 詰み探索結果を報酬とした強化学習
- 既存プログラムを加えたリーグ戦による強化学習
- 既存将棋プログラムの自己対局データを混ぜて学習
- 既存将棋プログラムの自己対局データを使った事前学習
- ブートストラップ法による Value Network の学習
- 引き分けも含めた学習
- 指し手の確率分布を学習
- 同一局面を平均化して学習
- 評価値の補正
- SWA(Stochastic Weight Averaging)
- 末端ノードでの短手順の詰み探索
- ルートノードでの df-pn による長手順の詰み探索
- 勝敗が確定したノードのゲーム木への確実な伝播
- PV 上の局面に対する長手数の詰み探索
- 序盤局面の事前探索（定跡化）
- 定跡作成時に floodgate の棋譜の統計を利用した確率分布を方策に利用
- マルチ GPU 対応（NVIDIA A100×8 を使用予定）
- TensorRT を使用
- Optuna による探索パラメータの最適化
- 確率的な Ponder
- ノードのガベージコレクションとノード再利用処理の改良
- 飛車と角の利きのビット演算
- 2 値の入力特徴量を 1 ビットで転送することで推論のスループットを向上
- Stochastic Multi Ponder

4　使用ライブラリ

- Apery[1]（WCSC28）
  →局面管理、合法手生成のために使用

4.1　ライブラリの選定理由

本プログラムは、将棋におけるディープラーニングの適用を検証することを目的としており、学習局面生成、局面管理、合法手生成については、使用可能なオープンソースがあれば使用する方針である。そのため、学習局面を圧縮形式(hcpe)で生成する機能を備えていて、合法手生成を高速に行える Apery を選定した。

[1] https://github.com/HiraokaTakuya/apery

5　各特長の具体的な詳細（独自性のアピール）

5.1　ディープラーニングを使用

DNN(Deep Neural Network)と MCTS を使用して指し手を生成する。

従来の探索アルゴリズム(α β法)、評価関数(3 駒関係)は使用していない。

5.2　Policy Network

局面の遷移確率を Policy Network を使用して計算する。

Policy Network の構成には、Residual Network を使用した。

入力の畳み込み 1 層と、ResNet 30 ブロック(畳み込み 2 層で構成)と出力層の合計 62 の畳み込み層で構成した。フィルターサイズは 3（入力層の持ち駒のチャンネルのみ 1)、フィルター数は 384 とした。

5.3　Value Network

局面の勝率を Value Network を使用して計算する。

Value Network は、Policy Network と出力層以外同じ構成で、出力層に全結合層をつなげ、シグモイド関数で勝率を出力する。

5.4　入力特徴にドメイン知識を積極的に活用

Alpha Zero では、入力特徴に呼吸点のような囲碁の知識を用いずに盤面の石の配置と履歴局面のみを入力特徴とすることで、ドメイン知識なしでも人間を上回ることが示された。しかし、その代償として、入力特徴にドメイン知識を活用した AlphaGo Lee/Master に比べて倍のネットワークの層数が必要になっている。AlphaGo Zero の論文の Figure 3 によると、ネットワーク層数が同一のバージョンでは Master を上回る前にレーティングが飽和している。

強い将棋ソフトを作るという目的であれば、積極的にドメイン知識を活用した方が計算リソースを省力化できると考えられる。

そのため、本ソフトでは、入力特徴に盤面の駒の配置の他に、利き数と王手がかかっているかという情報を加えている。それらの特徴量が学習時間を短縮する上で、有効であることは実験によって確かめている。

5.5　モンテカルロ木探索

対局時の指し手生成には、Policy Network と Value Network を活用したモンテカルロ木探索を使用する。

ノードを選択する方策に、Policy Network による遷移確率をボーナス項に使用した PUCT アルゴリズムを使用する。PUCT アルゴリズムは、AlphaZero の論文[2]の疑似コードに記述さ

[2] http://science.sciencemag.org/content/362/6419/1140

れた式を使用した。

また、末端ノードでの価値の評価に、Value Network で計算した勝率を使用する。

通常のモンテカルロ木探索では、末端ノードから複数回終局までプレイアウトを行った結果（勝率）を報酬とするが、将棋でランダムなプレイアウトは有効ではないため、プレイアウトを行わず Value Network の値を使用する。

5.6　未探索のノードの価値に親ノードの価値を使用

モンテカルロ木探索の UCB の計算時に、未探索の子ノードがある場合、そのノードの価値に何らかの初期値を与える必要がある。子ノードの価値は親ノードの価値に近いだろうという将棋のドメイン知識を利用し、それまでの探索で見積もった親のノードの価値を動的に初期値として使用する。ただし、ノードの訪問回数が増えるに従い、その価値の減衰を行い、幅より深さを優先した探索を行う(FPU reduction)。

5.7　GPU によるバッチ処理に適した並列化

複数回のシミュレーションを順番に実行した後、それぞれのシミュレーションの末端ノードの評価をまとめて GPU でバッチ処理する。その後、評価結果をそれぞれのシミュレーションが辿ったノードにバックアップする。以上を一つのスレッドで行うことで、マルチスレッドによる実装で課題となる GPU の計算後にスレッドが再開する際にリソース競合が起きる問題（大群の問題）を回避する。

GPU で計算中は、CPU が空くため、同じ処理を行うスレッドをもう一つ並列で実行する。2 つのスレッドが相互に CPU と GPU を利用するため、利用効率が高い処理が可能となる。

5.8　自己対局による強化学習

事前学習を行ったモデルから開始して、AlphaZero[3]と同様の方式で強化学習を行う。自己対局により教師局面を生成し、その教師局面を学習したモデルで、再び教師局面を生成するというサイクルを繰り返すことでモデルを成長させる。

2018 年の大会で使用した elmo で生成した教師局面で収束するまで学習したモデルに比べて、自己対局による強化学習によって有意に強くすることができた。

5.9　詰み探索結果を報酬とした強化学習

自己対局時に終局まで対局を行うと、モンテカルロ木探索の特性上、詰むまでの手順が長くなる傾向がある。勝率予測に一定の閾値を設けることで、終局する前に勝敗を判定することで対局時間を短縮できるが、モデルの精度が低いうちは誤差が大きいため、学習精度に影響する。

この課題の対策として、df-pn による高速な長手数の詰み探索の結果を報酬とした。単純に

[3] https://arxiv.org/abs/1712.01815

すべての局面で詰み探索を行うと、自己対局の実行速度が大幅に落ちてしまう。自己対局は複数エージェントに並列で対局を行わせ、各エージェントからの詰み探索の要求をキューに溜めて、詰み探索専用スレッドで処理するようにした。エージェントが GPU の計算待ちの間に詰み探索が完了する。エージェントが探索している局面は別々のため、時間のかかる詰み探索の要求が集中することは少ない。これにより自己対局の速度を大幅に落とすことなく長手数の詰み探索を行えるようになった。

5.10 既存プログラムを加えたリーグ戦による強化学習

自分自身のプログラムのみで強化学習を行うと戦略に弱点が生まれる可能性がある。弱点をふさぐには多様なプログラムによるリーグ戦が有効だが、複数のエージェントを学習するにはエージェント数の分だけ余分に計算資源が必要になる。

計算資源を省力化して、リーグ戦の効果を得るために、オープンソースで公開されている既存の将棋プログラムを 1/8 の割合でリーグに加えて強化学習を行うようにした。

5.11 既存将棋プログラムの自己対局データを使った事前学習

本プログラムを使用して、Alpha Zero と同様に、ランダムに初期化されたモデルから強化学習を行うことも可能だが、使用可能なマシンリソースが足りないため、スクラッチからの学習は行わず、既存将棋プログラムの自己対局データを教師データとして、教師あり学習でモデルの事前学習を行う。

教師データには、elmo で生成した自己対局データを使用した。

5.12 既存将棋プログラムの自己対局データを混ぜて学習

以前の dlshogi は、入玉宣言勝ちできる局面でなかなか入玉宣言勝ちを目指さないという課題があった。

自己対局では入玉宣言勝ちの棋譜が少ないため、それを補うため既存将棋プログラム(水匠)の自己対局で、入玉宣言勝ちの棋譜を生成し、dlshogi の自己対局のデータに混ぜて学習した。

5.13 ブートストラップ法による Value Network の学習

Value Network の学習の損失関数は、勝敗を教師データとした交差エントロピーと、探索結果の評価値を教師データとした交差エントロピーの和とした。

このように、本来の報酬（勝敗）とは別の推定量（探索結果の評価値）を用いてパラメータを更新する手法をブートストラップという。

経験的にブートストラップ手法は、非ブートストラップ手法より性能が良いことが知られている。

5.14 引き分けも含めた学習

将棋はルールに引き分けがあるゲームであるため、引き分けも正しく学習できる方が望ましい。そのため、自己対局で引き分けとなった対局も学習データに含めて学習した。

5.15 指し手の確率分布を学習

以前の dlshogi では、指し手のみを学習していたが、AlphaZero と同様に、自己対局時で MCTS で探索した際のルート局面の子ノードの訪問回数に従った確率分布を学習するように変更した。確率分布を学習することで、最善手と次善手の行動価値が近い場合に、次善手の行動価値を正しく学習できるようになる。

確率分布を学習することで、floodgate の棋譜に対する一致率が向上することが確認できたが、対局して強さを計測すると弱くなるという現象が確認できた。原因は、モデルの方策の出力の性質が変わるため、探索パラメータの調整が必要なためであった。Optuna を使用して探索パラメータを最適化(5.27 参照)することで、指し手のみを学習したモデルよりも強くすることができた。

5.16 同一局面を平均化して学習

自己対局では、序盤の同一の局面の教師データが多く生成される。それらの重複した局面を別のサンプルとして学習すると、モデルの学習に偏りが起きる。

局面の偏りをなくするために、同一の局面を集約し、指し手の確率分布と勝敗を平均化し、1 サンプルとして学習した。

5.17 評価値の補正

自己対局で生成するデータには、MCTS で探索して得られた勝率(最善手の価値)を局面の評価値を記録し、学習時にブートストラップ項(5.13 参照)として使用している。記録した評価値(勝率)が、実際の対局の結果から算出した勝率と一致しているか調べたところ、乖離しているという現象が確認できた。そのため、評価値を実際の自己対局での勝率に合うように、補正を行った。

5.18 SWA(Stochastic Weight Averaging)

画像認識の分野でエラー率の低減が報告されている手法である、SWA(Stochastic Weight Averaging)をニューラルネットワークの学習に取り入れた。一般的なアンサンブル手法では、推論結果の結果を平均化するが、SWA では学習時に一定間隔で重みを平均化することでアンサンブルの効果を実現する。

5.19 末端ノードでの短手順の詰み探索

モンテカルロ木探索の末端ノードで、5 手の詰み探索を行い、詰みの局面を正しく評価できるようする。並列化の方式により、GPU で計算中の CPU が空いた時間に詰み探索を行う

ため、探索速度が落ちることはない。

5.20 ルートノードでの df-pn による長手数の詰み探索

モンテカルロ木探索は最善手よりも安全な手を選ぶ傾向があるため詰みのある局面で駒得になるような手を選ぶことがある。

対策として、詰み探索を専用スレッドで行い、詰みが見つかった場合はその手を指すようにする。

詰み探索は、df-pn アルゴリズムを使って実装した。優越関係、証明駒、反証駒、先端ノードでの 3 手詰めルーチンにより高速化を行っている。

5.21 勝敗が確定したノードのゲーム木への確実な伝播

モンテカルロ木探索で構築したゲーム木の末端ノードで詰みが見つかった場合、その結果をゲーム木に伝播して利用する。つまり、モンテカルロ木探索に、AND/OR 木の探索を組み合わせて、詰みの結果を確実にゲーム木に伝播するようにする。

5.22 PV 上の局面に対する長手数の詰み探索

ディープラーニング系の将棋 AI は、選択的に探索を行うために、終盤の局面で読み抜けがあると、頓死することある。

頓死を防ぐため、PV 上の局面に対して、df-pn による長手数の詰み探索を行い、詰みが見つかった場合、局面の価値を更新するようにする。

5.23 序盤局面の事前探索（定跡化）

出現頻度の高い序盤局面は、対局時に探索しなくても、事前に探索を行い定跡化しておくことができる。また、事前に探索することで、対局時よりも探索に時間をかけることができる。

ゲーム木は指数関数的に広がるため、固定の手数までの定跡を作成するよりも、有望な手順を選択的に定跡に追加する方が良い。自分が指す手は、1 つ局面につき最善手を 1 手（または数手）登録し、それに対する応手は、公開されている定跡や棋譜の統計情報を使って確率的に選択する。その手に対して、また最善手を 1 手（または数手）登録する。この手順により、頻度の高い局面については深い手順まで、頻度の低い局面については短い手順の定跡を作成することができる。

5.24 定跡作成時に floodgate の棋譜の統計を利用した確率分布を方策に利用

定跡を自分自身の探索のみで作成した場合、読み抜けがあった場合に定跡を抜けた後に不利な局面になる恐れがある。そのため、モンテカルロ木探索の PUCT の計算で、方策ネットワークの確率分布と floodgate の棋譜の統計を利用した確率分布を平均化した確率分布を利

用し、致命的な読み抜けを防止する。

5.25 マルチ GPU 対応

複数枚の GPU を使いニューラルネットワークの推論を分散処理する。

「5.7 GPU によるバッチ処理に適した並列化」の方式により、GPU ごとに 2 つの探索スレッドを割り当てることで、GPU を増やすことでスケールアウトすることができる。ノードの情報は、すべてのスレッドで共有する。

確認できている範囲で 4GPU までほぼ線形で探索速度を上げることができている。

5.26 TensorRT を使用

モデルの学習にはディープラーニングフレームワークとして PyTorch を使用しているが、対局プログラムには、推論用ライブラリの TensorRT を使用する。

TensorRT を使うことで、事前にレイヤー融合などのニューラルネットワークの最適化を行うことで、推論を高速化することができる。TensorCore に最適化されており、TensorCore を搭載した GPU では CUDA+cuDNN で推論を行う場合より、約 1.33 倍の高速化が可能になる[4]。

また、対局の実行環境にディープラーニングフレームワークの環境構築を不要とすることを目的とする。

5.27 Optuna による探索パラメータの最適化

PFN により公開された Optuna[5]を使用して、モンテカルロ木探索の探索パラメータ（PUCT の定数、方策の温度パラメータ）を最適化した。

Optuna は、主にニューラネットワークの学習のハイパーパラメータを最適化する目的で利用されるが、将棋エンジン同士の連続対局の勝率を目的関数として、探索パラメータの最適化に使えるようにするスクリプト[6]を開発した。Optuna の枝刈り機能により、少ない対局数で収束させることができる。

5.28 確率的な Ponder

モンテカルロ木探索は確率的にゲーム木を成長させる。その特性を活かして、相手が思考中に、相手局面からモンテカルロ木探索を行うことで、確率的に相手の手を予測して探索を行うことができる。予測手 1 手のみを Ponder の対象とするよりも、効率のよい Ponder が実現できる。

---

[4] https://tadaoyamaoka.hatenablog.com/entry/2020/04/19/120726

[5] https://optuna.org/

[6] https://github.com/TadaoYamaoka/DeepLearningShogi/blob/master/utils/mcts_params_optimizer.py

5.29 ノードのガベージコレクションとノード再利用処理の改良

世界コンピュータ将棋オンライン大会でノード再利用に 10 秒以上かかる場合があるこがわかったため、ノード再利用の方式の見直しを行った。

以前は、オープンアドレス法でハッシュ管理を行っており、ルートノードから辿ることができないノードをすべてのハッシュエントリに対して線形探索してノードの削除をおこなっていた。

これを、Leela Chess Zero のゲーム木の管理方法[7]を参考に、ゲーム木をツリーで管理を行うようにし、ルートの兄弟ノードをガベージコレクションする方式に変更した。ノードの合流の処理が行えなくなるという欠点があるが、ノード再利用を短い時間で行えるようになった。

5.30 飛車と角の利きのビット演算

第 31 回世界コンピュータ将棋選手権の Qugiy のアピール文章[8]による、飛車、角の利きをビット演算により求める方法を実装した（実装はやねうら王のソースコードを参考にした)。ZEN2 の CPU で NPS が約 1% 向上した。

5.31 2 値の入力特徴量を 1 ビットで転送することで推論のスループットを向上

マルチ GPU を使用した場合、4GPU 以上では CPU と GPU 間の帯域がボトルネックになるため、2 値の入力特徴量を float の代わりに、1bit で表現し、GPU にビットで転送後、GPU 側で CUDA のプログラムでバッチ単位に並列に float に戻す処理を実装した。こうすることで、転送量が削減でき、NPS が 36.6%向上した。

5.32 Stochastic Multi Ponder

相手番に、相手番の局面から探索を行う Stochastic Ponder（5.28 参照）と、次の相手の指し手を複数予測し、並列に分散して探索を行う Multi Ponder を組み合わせて使う。

shotgun で実装されていた Multi Ponder[9]では、技巧 2 の Multi PV の結果を利用しているが、dlshogi の Stochastic Ponder では、ほとんどの場合、相手局面でのゲーム木が展開済みであり、ルートの子ノードの訪問回数を参照することで、有望な予測手を N 手取得することができる（ゲーム木が未展開の場合は、方策ネットワークの推論結果を使用する)。

また、予測した N 手以外の手が指された場合、Stochastic Ponder でも並列に探索を行っているため、Multi Ponder を使用しない場合と遜色のない手を指すことができる。

ponderhit した場合、次の局面の指し手予測の第一候補をその ponderhit したエンジンに

[7] https://tadaoyamaoka.hatenablog.com/entry/2020/05/05/181849
[8] https://www.apply.computer-shogi.org/wcsc31/appeal/Qugiy/appeal_210518.pdf
[9] http://id.nii.ac.jp/1001/00199872/

割り当てることで、前回の探索結果を再利用する。