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。

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。

WCSC34 W@ndre7 アピール文書(仮)

Dated 2024/02/07

気が付くと今回で世界コンピュータ将棋選手権に7回目の出場だそうです。あっという間ですね。
これまでのアピール文等を振り返ってみると、ディープ系に取り組むといいながら、なかなか実践していないなというのを再認識しました。
そこで、今年はdlshogi系をベースとし、いろいろ触りながら改造していく予定です。今はResNetと戯れている最中。

過去のアピール文

WCSC33 W@nderER 1 2 3(pdf)
WCSC32 W@nderER 1 2 3(pdf)
WCSC31 W@nderER 1 2(pdf) 2(docx)
WCSC30 W@nderER 1 2 3 (※WCSC30は中止で代わりにWCSOC2020が開催)
WCSC29 W@ndre 1 2 3 4
WCSC28 W@ndre 1 2
SDT5 W@ndre SDT5(pdf)

二番絞り

二番絞りの誕生経緯については2022年のアピール文などを参考に頂ければ幸いであるが基本的には最高精度を目指す大きなモデルを作成しようとする試みである.

2022年度は幸いにも準優勝という好結果を得た．準備段階では手ごたえがなかったがその後の計測で大変高精度なものが完成していたことが分かった．今年度は新しい試みを加えたがまともな計測に至っていない。もし旧バージョンより弱いと判断した場合は旧バージョンで出場する可能性がある.

ちなみに，2022度版の局面評価精度は驚愕の域に達しており，既発表であるが一手の局面展開も行わず将棋倶楽部24でレート2949，八段認定頂いている．前人未到の領域といって過言ではないだろう.

また，電竜戦ハードウェア統一戦においても準優勝となり，記念に2017年に行われたハードウェア統一戦の第5回電王トーナメントを思い起こしGTX1080Tiの二番絞りをfloodgateに投入したところ短期レート4500台を記録した．（もちろん一時的なものでありしばらくしてレートは4100〜4200程度で落ち着いた）

今年度はこれを上回ることを目指しているが上記の通り昨年に続き未計測である.

参考：

芝，「将棋のPV-MCTSに向けた深層学習モデルの最適化」，第45回ゲーム情報学研究会

芝，「探索アルゴリズムに適した時間利用に関する研究」，第46回ゲーム情報学研究会

第32回世界コンピュータ将棋選手権，https://bleu48.hatenablog.com/entry/2022/05/06/145915

芝，「コンピュータ将棋における高精度な深層学習モデル」，ゲームプログラミングワークショップ2022

二番絞り＠将棋倶楽部24の戦型分析，https://bleu48.hatenablog.com/entry/2023/03/08/062634

二番絞りの計測の件（GX1080Ti編），https://bleu48.hatenablog.com/entry/2023/03/09/132936

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

第３４回世界コンピュータ将棋選手権
「あすとら将棋」アピール文書

令和６年４月
恒岡 正年

1. 全体の構成
dlshogi の探索部、学習部を改造して利用。独自構造の model を採用。

2. 独自に実装した部分
Network 構造、学習方法、探索部の改造、model の生成の工夫等。

3. 開発動機
深層学習の勉強を兼ねて dl 系将棋 AI の model の学習を始めた。
WCSC33 に参加して決勝戦で全敗。上位ソフトと良い勝負をしたい。

4. 主な開発内容
独自構造の model の作成・学習方法
棋譜生成：5kpo～20kpo 程度の物を中心に現在約 55 万個の棋譜作成
学習には、上記生成した棋譜以外に GCT の学習用データを利用
探索パラメータの調整(optuna 利用)

５. 定跡生成
前回は手入力の定跡だったが、今回は条件を満たす棋譜を集め定跡を生成した。
条件付きで選別した自己生成棋譜及び Floodgate の棋譜から作成した。

千日手の棋譜を含めるべきかどうかを検討した結果、Draw_Value_Black/White を調整すれば千日手の棋譜を含めなくても良いとの結論になった。
この方法は手軽な割に効果が大きい。定跡の有無でレートが R100 位変わる。

「2 段仕込み定跡」を試している。これは、計算量と引き換えに上記の手法で作成した定跡に含まれている可能性のある悪手を排除する手法である。この手法の問題点は上記で登録された「対振り飛車定跡」が削除されてしまう点である。

最終的には「2 段仕込み定跡」か上記で生成した定跡との「ブレンド定跡」の良い方を使う予定。（対振り飛車には「ブレンド定跡」の方が良さそう。）

6.思考時間制御
手数と推定勝率に応じて思考時間を最大５倍まで変化させる様にした。

7.探索部

Hybrid 探索

２つの異なる学習を行った model(異なる極小値へ収束した可能性が高い)を用意し これらを A,B とする時 A=>A=>A または B=>B=>B の様に単独で使用し探索するよりも、A=>B=>A=>B の様に交互に探索する方が良い事を確認した。

V235(k=256,36b)と V252(k=384,24b)の２つの model を使って Hybrid 探索を行う。

8.高速化

network 構造の工夫、及び GPU が rtx4090・rtx3090 の場合に最適化した探索部の工夫により、同じパラメータ数を持つ ResNET 構造の model と dl 将棋標準の探索部の組み合わせに対して約２倍の高速化を達成した。

今回使用する予定の model:V235 と model:V252 の Hybrid 探索で、約 48000nps 程度の探索速度になった。

9.予想レート

以上の工夫により WCSC33 では R4200-R4300 程度だったが、今回は同じハードウェアで R4400-R4500 になったと推定している。

以下は少し細かな内容

10. Network 構造

Network を次の３つに分ける。

(1)データ入力部 (2)ResNET 部 (3)結果の出力部(Dual Head 部)

1）データ入力部

入力データの構成は標準と同じ。構造はオリジナルとは異なる。

2）ResNET 部

標準の ResNET とは異なる構造を採用。（ 以下 AstraNET と称する。）

WCSC33 では、カーネル数を k, ブロック数を b とする時、(k=256,b=5) + (k=224,b=20) の２段構造の通常の ResNET(合計 25 層)を採用した。

今年は、ResNET を AstraNET(自称)に置き換えた。また 20b~36b まで種々のサイズの model を作成し強さを比較した。

k=256 で 20b 及び 24b の model では思考時間を延ばしても R4400-4500 のソフトに読み負ける事を確認。これよりも重い Network が必要。

現時点では k=256,36b 及び k=384,24b の寸胴型の物を使う予定。NVIDIA 製 GPU のハードウェアの構造上２段構造にしても探索速度へのメリットは小さいと判断し寸胴型にした。

3）結果の出力部 (Policy/Value Network の分岐後)
dlshogi と同じ構造。
4）その他
AstraNET の活性化関数は主に ReLU を、全結合層には ELU を使っている。
(去年は全結合層に SiLU を使った)

11. 学習方法
１）学習率(lr)：0.2 から始めて 0.000003 まで等比数列的に減少させ学習した。減少させる割合は 0.80～0.95。d 値(=Loss-AverageAcc) の下がり方を見ながら調整した。
２）weight decay を 0.00010～0.00003 まで振って評価した。0.00003 は小さすぎる可能性がある。学習初期は大き目の物を使って学習終盤に小さくするのが良さそう。
３） 学習の途中で AstraNET のブロックの順序の入れ替えを適宜行っている。（去年も同様）

12. model の「追加工」
学習後に model の「追加工」をおこなった。
必ずしも Loss 値や d 値の最も小さい物、PolicyAcc,ValueAcc,Entropy の最も大きい物がベストな model とは限らない。
学習終盤では上記の指標を参考に良さそうな model を選択しベンチを実施。ベンチの結果の良かった複数の model のブレンド(例えば、3:2:1、100:1、100：-1 等)や乱数による揺らぎを与え派生 model を作成した。これらの中から最もベンチ結果の良かった物を最終的に採用した。
ベンチは棋譜生成も兼ねており次回の学習に使用する為 10k po~40k po で行った。
「追加工」により R+60 程度向上する場合も有った。

10. Optuna での探索パラメータの調整
去年の model は手作業である程度絞り込んでから Optuna を使った。
今回は 7 つある探索パラメータから以下の様なグループ A、B、C、D を作り、
A=>B=>C=>A(不十分な場合 B=>C=>A を追加)=>D=>E の様に調整した。
１回の Trial 数は 8～15 程度。全てを合計しても 100Trial は超えていない。

A: Softmax_Temp., root パラメータ 計 4 個
B: Softmax_Temp., non-root パラメータ 計 4 個
C: Softmax_Temp., C_fpu_reduction, C_fpu_reduction_root 計 3 個
D: 全パラメータ 計 7 個
E:Softmax_Temp., C_fpu_reduction, C_fpu_reduction_root を個別に振って妥当性を確認

一回の調整毎に上位 2~3 個のベンチを取り、ベンチ結果の最も良い物を次の中心値とした。Optuna での相手は Hao(3.5e6 nodes),ベンチでの相手は Hao(~6.0e6 nodes)と

Tanuki-dr4(~5.0e5 nodes)を使った。評価される側の勝率が 50~65%になる様に po 数を調整した。1 Trial の games 数は 250 とした。

11.中終盤の棋力向上

WCSC33 では、80~100 手付近で読み負けるケースが多かった。その対策として以下の 3 点を行った。

1) 自己生成棋譜は、去年は意図的に 40～80 手を重複させ 30 手～120 手の局面を学習データとした。121 手以降は未使用であった。今回は、意図的な重複を排した 30～150 手の範囲の局面を使用した。

2) 学習終盤では全学習データ内の重複局面を削除して学習した。これにより序盤～中盤初期の棋譜の割合が減る。

全データを毎回全て学習するのは時間が掛かりすぎるため、学習時間が 12 時間を超えると次の学習条件に移る様にした。経験的に学習条件を変更した直後から 3 ～ 4 時間の間に有望な model が得られる事が多く、逆に 8 時間以上同一条件で学習を続けても良い model が得られる事は少なかった。

3) 持ち時間の使い方を変更し、70～95 手の思考時間を延ばした。また思考時間を延長する場合、従来 2 倍までだった物を条件によって最大 4 倍まで延長するようにした。

-以上-

名人コブラ アピール文書

松山洋章

概要

ディープラーニング評価関数とNNUE評価関数をベースとしたアンサンブル評価関数を使用します

評価関数

複数のNNUE評価関数をマージして
利用します

定跡

意図的にランダムな手を混ぜた自己対戦により定跡を作成します

使用ライブラリ

- **dlshogi**

  局面評価のベースとして。

  序中盤の局面評価に優れるため。

- **やねうら王**

  局面評価のベースとして。

  読みの速さに優れるため。

ソフト名の由来

劇団鋼鉄村松

「二手目8七飛車成り戦法」

登場人物より

参考:
https://stage.corich.jp/stage_main/23036

ponkotsu アピール文書

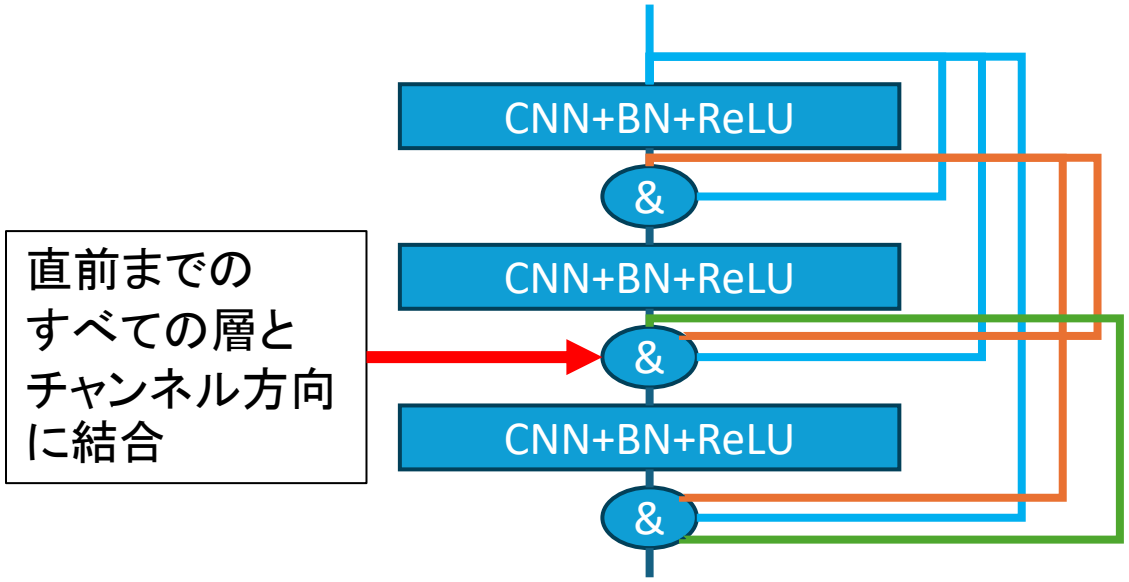
WCSC33からの改良内容

- ネットワーク構造の改良
- 学習率スケジューラの変更
- 最適化関数の変更
- 学習データのアップデート

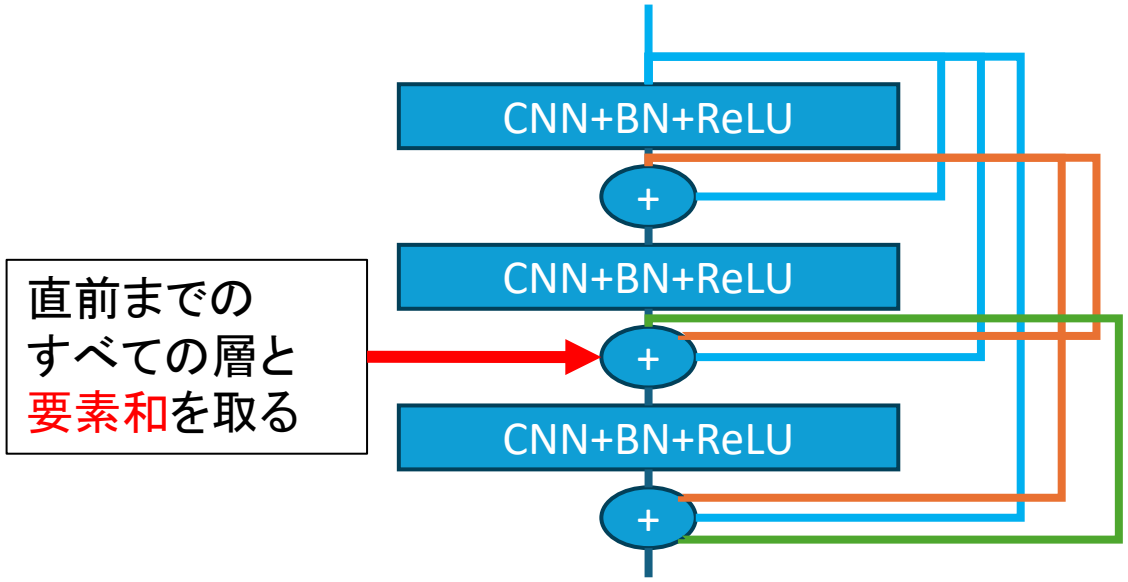
ネットワーク構造の改良

- 従来版で問題となっていた層が深くなるにつれチャンネル数が増大する問題の対策として、スキップ接続をチャンネル方向に結合するのではなく要素和を取るようにした

従来版(DenseNet)



改良版



※オーバーフロー対策として要素和を取った後平均を取っている

ネットワーク構造の改良(識別精度の比較)

- 学習データ: DL将棋本付属訓練データ
- epoch数: 22 epoch
- テストデータ: DL将棋本付属テストデータ
- Policy・Valueともに改良後のほうが高精度

Model	Policy Accuracy(%)	Value Accuracy(%)
DenseNet10	43.4	72.2
改良版DenseNet10	**44.3**	**72.9**

学習率スケジューラの変更

- 学習率スケジューラをReduceLROnPlateauに変更

ReduceLROnPlateauでは

$\mathrm{loss}_{\mathrm{best}}$: 最も良かったloss
threshold: 閾値
patiance: 改善が見られなかったepoch数

としたときlossがpatience epoch連続で

$$\mathrm{loss}_{\mathrm{best}} * (1 - \mathrm{threshold})$$

以上だった場合学習率を下げる

最適化関数の変更

- 最適化関数をSAM(Sharpness-Aware Minimization)に変更

以下の手順でパラメータを更新する

1. パラメータwの周辺で損失が最大となる$w + \hat{\epsilon}(w)$を算出
2. $w + \hat{\epsilon}(w)$における損失・勾配を算出
3. 2.で算出した勾配でwを更新

最適化関数の変更(識別精度の比較)

- 学習データ: DL将棋本付属訓練データ
- epoch数: SGD 22 epoch、SAM 11epoch
- テストデータ: DL将棋本付属テストデータ
- Policy・ValueともにSAMを使用したほうが高精度

Model	Policy Accuracy(%)	Value Accuracy(%)
DenseNet10+SGD 22 epoch	43.4	72.2
DenseNet10+SAM 11 epoch	**44.0**	**72.7**

学習データのアップデート

- DL将棋本の付属データのうちfloodgateの棋譜と水匠の棋譜を以下のようにアップデート

種類	従来の内容	アップデート後の内容
floodgateの棋譜	floodgateの2019年～2021年5月のレーティング3500以上のプレイヤーの棋譜	floodgateの2019年～2024年3月のレーティング3800以上のプレイヤーの棋譜
水匠の棋譜	水匠3改を使用して、1手120万ノードで自己対局した棋譜	たややんさん作成の最新の水匠を使用して1手1000万ノードで自己対局した棋譜(リンク)

その他

- 以上の改良に加え、ネットワークのブロック数を10→15に増やす予定
- 後日昨年のバージョンとレーティングを比較する予定

参考文献

- ReduceLROnPlateau — PyTorch 2.2 documentation (2024年3月27日閲覧)
- "Sharpness-Aware Minimization for Efficiently Improving Generalization", Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B. (2020)
- SoTAを総なめ！衝撃のオプティマイザー「SAM」爆誕&解説！ #機械学習 – Qiita (2024年3月27日閲覧)

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。

アピール文

花井祐　2024 年 3 月 14 日記

プログラム名：いちびん

プログラムの内容は以下のとおりです。

1.概要

NNUE を基本としています。思考部は、自作の教師データを使って学習を行っています。探索は、やねうら王のソースコードに手を入れて時間管理の部分を書き換えています。定跡は WCSC33 に参加したバージョンを基本に作成しています。

2.プログラムの内容

2.1　思考部分

今では少々時代遅れになってしまった NNUE 路線を守っています。教師データは、自己対戦で数年前に作成した深さ１２～１４の教師データを数億局面作成し、評価関数を作成しています。１年間、努力しましたが WCSC33 バージョンを上回る強さにはできなかったと考えます。

2.2　探索部分

WCSC33 で、比較的良い成績を残せたのは、時間管理を少し工夫したことだと自己評価しています。WCSC34 では、時間管理をさらにブラッシュアップしました。その内容は、局面の評価点がマイナス６００点より悪化した場合には、「ほとんど負け」として、それ以前に思考時間の山場をもってくるようにしています。

2.3　定跡

WCSC33（2023 年 5 月開催）終了後の翌日から、ほぼ 1 年間、機械を動かし続け、ひたすら定跡を作り続けました。「いちびん WCSC33 参加バージョン」を 1 手あたり１０億局面読ませる作業をつづけ、消費する電気代に涙しつつ、今日に至っています。

３　その他

WCSC33（2023 年 5 月開催）終了後に、DL について導入を考えましたが、現在のコンピュータ将棋にかける情熱と予想される高額な費用とを考えて DL への挑戦をあきらめました。頭を使いながら評価関数を手作りしていた時代は遠い過去のことになってしまいました。

以上

Polonaise アピール文書

谷合廣紀

2024 年 3 月 31 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。

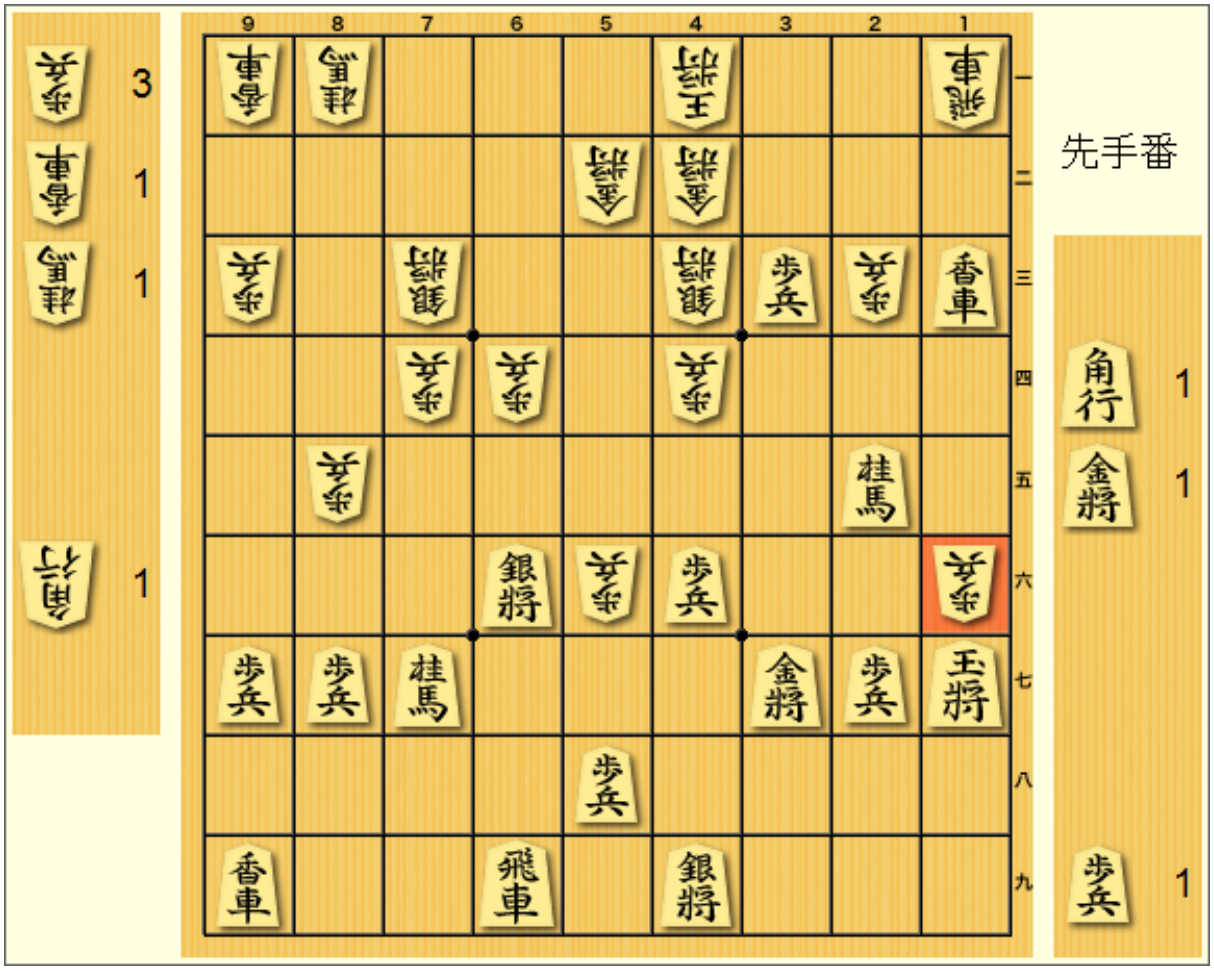


図1: 入力盤面

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。

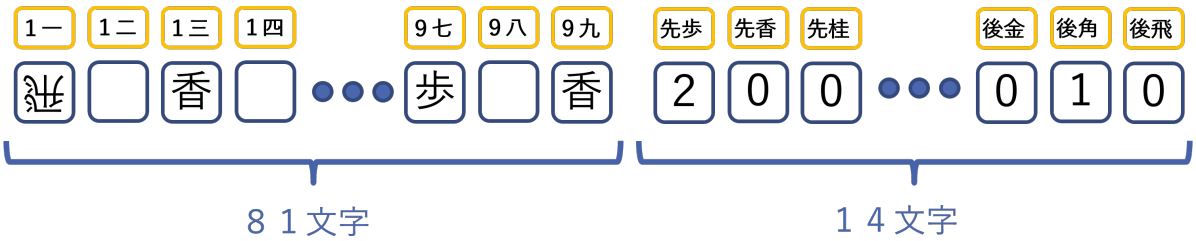


図2: 文字列エンコード

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。

2　使用ライブラリ・使用データ

- ふかうら王
- 「強い将棋ソフトの創り方」公開データ

参考文献

Daigorilla WCSC34 アピール文章・・・急いで書きました泣

工夫した点

① 後手番勝率の強化：　後手は角換わりを避け、雁木もしくは力戦にさすように model のパラメータを調整した。

② 探索部の高速化及びパラメーターの調整:ふかうら王を CUDA12.3　TensorRT8.6、cudnn9.0 に最適化し、後手の勝率を疑似調整した棋譜で自動調整を行った。ちなみにやねさんのスポンサーには入っていないのでバージョンは 8.00

③ Model の最適化：特に至って工夫はしていないが 20b_swish で学習を進めている。現時点では手動と自動で調整している定跡を用いて標準型 NNUE 関数「Hao」を用いて depth14～18、約 5 億局面を学習させている。学習率を徐々に下げている。

④ 定跡の強化：秘密にしておきたいが上記の通りかなり手間をかけて生成している。無駄な局面を減らすため実現確立が高いものから局面を誘導し、様々な評価関数に対応できるようにしている。詳しくは決勝に残ったら書きたい。

Team Novice

wcsc34

Update..

・昨年までDL部を中心に開発していた中屋敷さんが別ソフトとして出場します

　→　nshogi として出場しています！

・これに伴い、昨年のバージョンからDL部を切り離し、NNUEのみでの出場となります

・NNUE部は昨年から開発が止まっているため、目安ソフト的な立ち位置として出場します

・定跡は例年通り新たに作成する予定です

・フルスクラッチについては、開発が止まっていることもあり本年は申請していません

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。

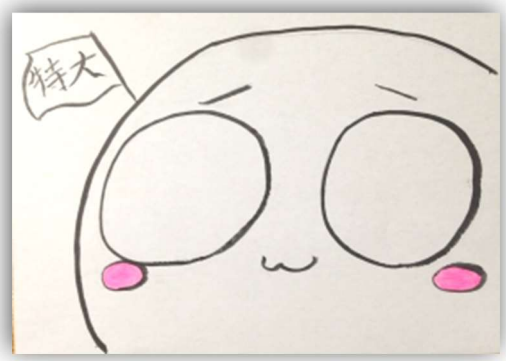
たややん氏が公開している教師データ群

　　Fine Tuning において優秀な教師データであるため。

TMOQ アピール文書

2024 年 03 月 20 日 作成
2024 年 04 月 30 日 改訂

【ソフト名】TMOQ（特大もっきゅ）
“TMOQ”と書いて「特大もっきゅ」と呼びます、
愛娘が命名しデザインしたものです



【コンピュータ将棋大会実績】
2016 年の WCSC26 以降、ほぼ全ての大会に出場、
そこそこの成績でやってきてます

【WCSC34 の目標】
少なくとも 1 回は入玉宣言勝ちする！

【TMOQ の特徴 および WCSC33 との違い】

1. 戦いを避ける平和主義者（入玉宣言勝ち狙い）
2. dlshogi をベースに複数エンジンを使用
3. メインの思考部は『Lí（Ponder 無し)』(昨年は ResNet 9b(Ponder 有り))
4. 勝勢になったら、駒得エンジンに切替えて入玉を目指す（入玉可能性を増やすべく、昨年の駒得エンジンをチューニング）
5. 約 2 千 5 百万手に近い定跡ファイル（昨年より約 300 万手増）
6. Note PC を使用、莫大な計算資源がなくてもコンピュータ将棋は楽しめる！

※ 必ずしも強くなることを目指している訳ではありませんが、強豪ぞろいの二次予選で入玉宣言勝ちすべく、メインの思考部を独自の ResNet 9b から『Lí』に切り替えました。結果、昨年 WCSC33 版とのテスト対局で約 76 %の勝率となりました

【使用ライブラリ】

- 『DeepLearningShogi』(Commit 790e2f4 on 3 Feb 2022) (GPL)
- 『Lí』（tanuki- 第 33 回世界コンピュータ将棋選手権バージョン）
- 『やねうら王』 V7.00

【御礼】

今回も山岡氏、加納氏、野田氏＆やねうらお氏を中心に、多くの方々の公開情報により参加できました。この場を借りて御礼申し上げます

AobaZero の 2024 年のアピール文書

山下 宏
yss@bd.mbn.or.jp

1 AlphaZero の追試が最初の目的

AobaZero は Bonanza、LeelaZero のコードをベースに AlphaZero の追試をするべく MCTS ＋ディープラーニングで実装されてます。ネットワークは 3x3 のフィルタが 256 個の 20 block の ResNet でパラメータの個数は 2340 万個。棋譜生成をユーザの皆様と協力して行う分散強化学習です。オープンソースです[*1]。

2 AlphaZero の追試は 2021 年 4 月に終了

AlphaZero の将棋の追試は、2019 年 3 月から開始し、2021 年 4 月に 3900 万棋譜を作成して終了しました[*2]。2024 年 3 月 29 日現在、6746 万棋譜を作成しています。

3 追試終了後から +260 ELO、去年の選手権から +30 ELO

追試終了後からは +260 ELO、去年の選手権からだと +30 ELO ほど強くなっています。追試終了時では AlphaZero より +150 ELO 弱い、という推定でしたが現在は +260 なので AlphaZero を +110 ELO 程度、超えた棋力かもしれません。

4 昨年の選手権からの主な変更点

- 学習棋譜の平均 playout 数を 1600 から 3200 に変更
- 先手勝ち局面の学習確率を減らす
- NVIDIA のドライバ更新時のエラー修正

4.1 学習棋譜の平均 playout 数を 1600 から 3200 に

ほぼ重みの棋力が停滞していたので playout 数を倍にしてより強い棋譜を生成するようにしました。当然生成時間は倍かかるので、生成棋譜数も 11000 棋譜/日から 9700 棋譜/日に下がってます。・・・ほとんど下がっていないですね。実は倍に変更してからかなりの資源を投入して棋譜生成に協力してくださった方がいたためです。この場をお借りして感謝いたします。棋譜の棋力はほぼ同じ条件で floodgate で走らせた限りでは平均 1600 の 3630 から平均 3200 で 3740 と +110 ELO 程度向上しているようです。この平均、とは 400playout ごとに Root の着手の訪問数の分布に変化がないなら打ち切る、という手法です。最小 400、最大 12800 です。この KL 情報量を利用した探索の打ち切りは LeelaChessZero で使われています[*3]。この探索量の調整で 1 手 3200 固定より +100 ほど強くなります。

4.2 先手勝ち局面の学習確率を減らす

playout 数を倍にして棋力を上げたのですが、生成棋譜の先手勝率が 0.724 と 7 割を超えていました。PUCT の性質で勝率が悪い局面だと多くの候補手を調べて、さらに弱くなってるケースがありそうです。そこで先手が勝った局面の学習確率を減らして初期局面の勝率が 5 割になる重みを再学習で作ってみたところ、棋力も +14 ELO 程度強いものが出来たので、これに差し替えました。その結果、先手勝率は 0.632 と 1 割近く下がりました。形勢判断としては正しくないですが棋譜生成においてはこちらの方が望ましいかもしれません。

4.3 NVIDIA のドライバ更新時のエラー

NVIDIA のドライバを最新にすると AobaZero が起動時に動かなくなる、という現象が発生しました。これは初回起動時に OpenCL の最適な設定を見つける行列計算を行う際に、OpenCL を動的にコンパイルしていたのですがコンパイラのエラーチェックが厳しくなったせいでした。芝さんなどの指摘で気づきました。ありがとうございます。

5 dlshogi 互換モデルで出場予定

AobaZero の棋譜で dlshogi のモデルを作ってみました。AobaZero と同じ 20 ブロック、256 フィルタの ResNet、Swish だと学習速度は 5.7 倍！探索速度の NPS は 3 倍ほど速いです。学習は 3090 だとミニバッチ 256 が最大だったのですがミニバッチ 2048 もあっさり動いたのにビックリでした。

1 手 1playout と 1 手 100playout、nps(RTX 3090) は表 1 です。ミニバッチはすべて 1 です (nps 測定では AobaZero が mb=17、dlshogi が mb=256)。dlshogi dr2 には Policy の強さ (1p)、探索 (100p) でもまだはっきり負けています。重みサイズが 2 分の 1 程度にも関わらず。384x30b にしてもまだ Policy の精度で負けてるので何か学習が正しくない気はします。ただ実時間だと nps が 3 倍近いのもあり (3090

[*1] https://github.com/kobanium/aobazero
[*2] https://github.com/kobanium/aobazero/issues/54
[*3] https://lczero.org/dev/wiki/technical-explanation-of-leela-chess-zero/

で AobaZero は 6000 程度) +40 ELO 程度 dlshogi モデルが強いので、これで選手権は参加する予定です。AobaZero は TensorRT に対応してないのと、複数 GPU での性能がいまいちなのもあって。計測は 24 手までの互角局面集利用で先後入れ替えて 800 局からです。

5.1 局面選択は Value に効果的？

学習は通常の AobaZero の学習と同じように学習される局面を選択しています[*4]。

先手勝率を 5 割になるように先手勝ち局面の選択割合を減らす、序盤の学習確率を減らす、勝率が 80 %程度の手が 8 倍程度で学習されやすく、など。全局面の 50 %ほどが選択されるぐらいで。それを csa 形式に変換した後、aoba_to_hcpe3.py を改造して hcpe3 形式に変換しています。この選択をなしで hcpe 形式 (着手のみ) で学習させた場合は Policy の精度はいいのですが 1 手 100playout で 251 差、とはっきり弱く、局面の選択をするのは Value の精度に効いてるかもしれません。なお重複局面を平均化して削除する dlshogi の学習機能はどちらも有効にしています。

学習は 4000 万から 6690 万までの 2690 万棋譜から 5 億 7000 万局面を抽出してミニバッチ 2048 で学習率 0.1 から 0.001 まで Cosine Annealing で学習させました。さらに 0.00001 ぐらいまで下げた方がもう少し性能が上がるようです。

表 1　AobaZero(w4357) の dlshogi モデルに対する ELO

	1p/手	100p/手	nps
dlshogi dr2 (224x15b)	-61	-100	25300
AobaZero 256x20b	186	100	20500
AobaZero 256x20b(選択なし)	157	251	20500
AobaZero 384x30b	61	-58	5500

6　定跡は floodgate の AobaZero の棋譜から

定跡は floodgate で走らせている AobaZero の棋譜の局面を 30 万局面ほど探索させて作成する予定です。

7　対振り飛車の弱さが課題

floodagte で HoneyWaffle との棋譜を見ると相手が飛車を振っただけで評価値が +400 (勝率 65 %) と大きく出ます。学習棋譜でも振り飛車の割合は 2 %しかなく、ミレニアムや振飛車穴熊、といった囲いの知識もないので多様性に欠ける感じはします。この影響もあり特に対振り飛車は弱い感じです。

8　人間の知識は使っていない、をおそらく継続

利きの情報の追加や 3 手詰、dfpn 詰などで AlphaZero からは離れてきましたが、まだ全体としては「人間の知識は使っていない」を継続していると考えています。

9　棋力の推移

図 1 が ELO の推移です。floodgate での測定レートの方が若干高めなのは Kristallweizen での棋力測定に用いている互角局面集 (24 手まで) には AobaZero が指さない穴熊や振飛車が多数含まれているせいです。局面集を使わずに初手から指させた方が +100 ELO ほど強くなります。

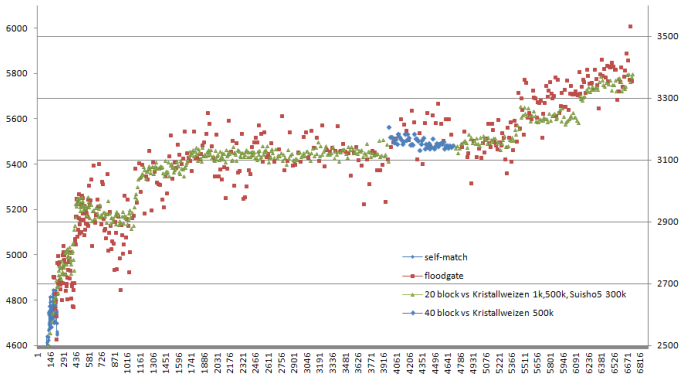


図 1　棋力の推移。右軸が floodgate のレート、横軸は棋譜数 (万)

10　5 年で 6700 万棋譜

6700 万棋譜、という膨大な棋譜を 5 年間で生成してきました。棋譜生成に協力していただいてる皆様に感謝いたします。

[*4] https://www.apply.computer-shogi.org/wcsc33/appeal/AobaZero/2023csa_appeal.pdf

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

第34回世界コンピュータ将棋選手権「水匠」アピール文書

令和６年３月３１日

たややん

第１　定跡作成プログラム

１　ＷＣＳＣ３３においては、以下の定跡作成手法を採用していました。

⑴　指定局面から連続対局させ、指し手を全て定跡として登録する。

⑵　任意の局面が定跡に登録されていた場合、ミニマックス法で定跡データ内を探索し、勝ちの枝があるか、負けの枝しかないか判別する。

⑶　勝ちの枝があれば、その手を指し、負けの枝しかない、又は定跡に登録されていない局面であれば、探索エンジンで思考させる。

２　これに対し、ＷＣＳＣ３４における定跡作成手法は、以下の手法により、ミニマックス法で探索する必要なく、前項⑴～⑶と同様の定跡が作成できる仕組みを採用しました。

⑴　対局が終わった際、勝った側の指し手は全て定跡データベースに登録する。

⑵　負けた側の指し手は、末端の局面から、定跡データベースに今回選択された指し手と別の手が登録されている局面が現れるまで、定跡データベースから削除する。

⑶　定跡データベースに指し手が登録されていればその手を指し、登録されていなければ、探索エンジンで思考させる。

３　コードは公開されています。

https://github.com/tayayan/HiraganaSuisho/blob/main/makebook2.py

第２　評価関数の学習

第３２回世界コンピュータ将棋選手権に出場されたマメットブンプクの評価関数からファインチューニングをした評価関数を使います。

学習手法での工夫は、以前とほぼ同様（学習させる局面の評価値に閾値を設ける、学習時FV_SCALEの調整、勝敗項の教師信号の低減等）です。

教師データとして、日本AMD株式会社様よりお借りした、AMD EPYC™ 9654プロセッサーやAMD Ryzen™ Threadripper™ PRO 7995WXプロセッサーを活用し、水匠の1手1000万ノードで対局して作成したデータを使用しました。

当該教師データも一部（約１億局面）公開しています。

https://drive.google.com/file/d/1VyP4MX_AuQhvy8sesymgPVf9sUnQoGPl/view?usp=drive_link

第３　評価関数リレーの採用

上記学習方法を採用することによって、入玉宣言が苦手な評価関数となっているため、一定の評価値を超えたら水匠5にスイッチする仕組みを採用します。

第４　探索部の改善

やねうら王を使用し、Stockfishに実装された探索部の変更部分の採用及びパラメータ調整をしています。

以上

「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.

第34回世界コンピュータ将棋選手権

Ari Shogi and フレンズ　アピール文書

兵頭優空

概要

Ari Shogi and フレンズは、DL系のAI「Ari Shogi」とNNUE系のAIを組み合わせる事で「比較的低コストであらゆる状況に対応できるAI」を目指しているハイブリッド型の将棋AIです。

(12月末に今まで使っていたノートPCが壊れたので)1月末から新しいPCに変わり、計算資源にかなり余裕ができたので、NNUEの学習など今まで計算資源の都合で取り組めなかった事にも取り組みつつ、良い成績を目指したいと思っています。

目次

名前について

「Ari Shogi and フレンズ」という名前の「Ari Shogi」の部分は牡羊座(Aries)から、「and フレンズ」の部分は、複数のAIで協力するイメージからつけています。

牡羊座から名前を付けたのは、「自分の星座が牡羊座だから」と「某電気羊のように、何回か大きく進化していつか電竜になってほしいという願いを込めている」の2つの理由があります。(2つ目は後付けです)

アピールポイントなど

・DL系のAIの「終盤が弱い」など弱点を単体で克服するのを放棄し、苦手な部分はNNUE系AIに任せます。

・DL評価関数では、精度を上げるために複数の評価関数を使ってアンサンブルを行います。

第4回電竜戦(以降、DR4)でAri Shogi and フレンズは、「複数の評価関数の出力の平均を1つの評価関数の出力として扱う」という機能を大会直前に実装し、「自分が学習させた中で精度の高いモデル上位3つ」を搭載して参加していました。

この機能は、

1. かなり前から「複数の評価関数で行えば(速度は大きく落ちるものの)簡単に精度を上げられそう」という事を考えていた(実際に手を付けることはなかった)

2. 11月に「夏頃から新しく使い始めた入力特徴量」に大きなバグが見つかった。その特徴量を使って学習している途中だったモデルは良い感じの精度まで上がっていたが、結構酷いバグだったので新しいモデルを1から学習させる事になった。

3. 2のせいで予定が狂い、大会で使うモデルの精度が予定よりもずっと低くなってしまった。(これ関連で入玉モデル、進行度モデルの学習もほとんど行えなかったので、それらの出力による合議のモード切替は全然上手くいかなかった)

4. 大会直前に「今から精度を上げる方法はないか」と考えていたところ、アンサンブルの件を思い出したので、それ関連で実装が1番簡単そうだった「出力を平均する」というのを実装した。(当時は学習はGoogleColab無料版などの「ネットで無料で使えるGPU」に頼ってい

たが、使える分を全部使いきってしまった後だったのでそんなにできる事がなかった。当時のメインマシンのノートPCにはGPUが乗っていたが、GPUに負荷をかけるとバッテリー周りの挙動が怪しくなるようになっていたので、GPUに負荷が猛烈にかかる学習は怖くてできなかった。)

という経緯で実装された急場しのぎのものでしたが、評価関数の精度が(少しだけですが)ちゃんと上がり、デメリットの速度低下も

・探索部がPythonで書かれているので、C++で書かれているものに比べると元々かなり遅い

・(速度低下が1番悪影響を及ぼしそうな)中終盤はNNUE系AIに丸投げしている

・最善手をNNUE系AIでチェックし、NNUE系AIが考える最善手との評価値から大きく悪化している場合は、NNUE系AIの最善手を指すという機能がある

といった理由であまり気にならなかったので、割と上手くいった改造だったと考えています。

WCSC34バージョンのAri Shogi and フレンズは、DR4バージョンで上手くいった方針を発展させ、より強力な手法を採用等に取り組み、精度のさらなる向上を目指します。

また、アンサンブルに使うモデルはある程度多様性があったほうが良いと考えているので、今使っているResNet系以外の系統のモデルも学習させてアンサンブルに使う予定です。

(3月末の時点では、
https://tokumini.hatenablog.com/entry/2021/09/04/120000

を参考にVision Transformer(ViT)を実装するところまで終わっているが、計算資源が足りなくて学習は回せていない)

・NNUE評価関数を自作の学習部で学習させます

DR4までは、「NNUEの学習もやってみたいなー」とは思ってはいたものの、リソースの都合等から取り組む事はなく、公開されているNNUE評価関数をそのまま利用していましたが、今回からは自分で学習させた標準型のNNUE評価関数を使用する予定です。

学習については、普通に学習させるだけでなく色々な実験(DLモデル

からの蒸留とか)をしたいと考えています。(が、リソースの都合であまりできなさそう)

使い慣れたプログラムをベースにしている方が効率が良く実験できると考えたので、実験用プログラムのベースとしてpython-dlshogi2の学習部をベースにしたNNUE学習部を製作しました。

学習部は公開してほしいという要望があったため、テストで作った振り飛車評価関数と学習ログと一緒に
https://github.com/YuaHyodo/Ari_Shogi_NNUE_train

https://github.com/YuaHyodo/Haojian_nnue

で公開しています。(厳密には、非公開のものから分岐させて個人用のメモとかを消した後のものなので同じものではないです。また、元々公開する予定のなかったものなので、超絶汚いです)

上の学習部(のもとになった非公開の学習部)でHáoを水匠1000万ノードのデータで1epochだけ追加学習した"Haojian_240317"という評価関数が、CPU: i5-13400F, 探索部: やねうら王7.50,1スレッド, ハッシュ512MB, たややん互角局面集24手目～という設定で、

```
///
VS Háo / 1手1000ms
対局数5000 先手勝ち2606(52.7%) 後手勝ち2336(47.3%) 引き分け58
engine1
勝ち2558(51.8% R12.1 +-9.6) 先手勝ち1349(27.3%) 後手勝ち1209(24.5%)
宣言勝ち8 先手宣言勝ち3 後手宣言勝ち5 先手引き分け33 後手引き分け25
engine2
勝ち2384(48.2%) 先手勝ち1257(25.4%) 後手勝ち1127(22.8%)
宣言勝ち3 先手宣言勝ち2 後手宣言勝ち1 先手引き分け25 後手引き分け33
///
VS Háo / 1手5000ms
対局数5000 先手勝ち2478(50.0%) 後手勝ち2475(50.0%) 引き分け47
engine1
勝ち2630(53.1% R21.4 +-9.7) 先手勝ち1315(26.5%) 後手勝ち1315(26.5%)
宣言勝ち15 先手宣言勝ち10 後手宣言勝ち5 先手引き分け21 後手引き分け26
engine2
```

勝ち2323(46.9%) 先手勝ち1163(23.5%) 後手勝ち1160(23.4%)

宣言勝ち6 先手宣言勝ち3 後手宣言勝ち3 先手引き分け26 後手引き分け21

///

VS BLOSSOM / 1手1000ms

対局数5000 先手勝ち2486(50.2%) 後手勝ち2470(49.8%) 引き分け44

engine1

勝ち2914(58.8% R61.2 +-9.8) 先手勝ち1461(29.5%) 後手勝ち1453(29.3%)

宣言勝ち22 先手宣言勝ち13 後手宣言勝ち9 先手引き分け22 後手引き分け22

engine2

勝ち2042(41.2%) 先手勝ち1025(20.7%) 後手勝ち1017(20.5%)

宣言勝ち1 先手宣言勝ち0 後手宣言勝ち1 先手引き分け22 後手引き分け22

///

VS BLOSSOM / 1手5000ms

対局数5000 先手勝ち2597(52.5%) 後手勝ち2346(47.5%) 引き分け57

engine1

勝ち2876(58.2% R56.7 +-9.8) 先手勝ち1502(30.4%) 後手勝ち1374(27.8%)

宣言勝ち9 先手宣言勝ち4 後手宣言勝ち5 先手引き分け30 後手引き分け27

engine2

勝ち2067(41.8%) 先手勝ち1095(22.2%) 後手勝ち972(19.7%)

宣言勝ち1 先手宣言勝ち0 後手宣言勝ち1 先手引き分け27 後手引き分け30

///

VS 水匠5 / 1手1000ms

対局数5000 先手勝ち2580(52.5%) 後手勝ち2331(47.5%) 引き分け89

engine1

勝ち2548(51.9% R12.9 +-9.6) 先手勝ち1338(27.2%) 後手勝ち1210(24.6%)

宣言勝ち56 先手宣言勝ち30 後手宣言勝ち26 先手引き分け47 後手引き分け42

engine2

勝ち2363(48.1%) 先手勝ち1242(25.3%) 後手勝ち1121(22.8%)

宣言勝ち1 先手宣言勝ち0 後手宣言勝ち1 先手引き分け42 後手引き分け47

///

VS 水匠5 / 1手5000ms

対局数5000 先手勝ち2515(51.1%) 後手勝ち2402(48.9%) 引き分け83

engine1

勝ち2596(52.8% R19.1 +-9.6) 先手勝ち1323(26.9%) 後手勝ち1273(25.9%)

宣言勝ち56 先手宣言勝ち29 後手宣言勝ち27 先手引き分け34 後手引き分け49

engine2

勝ち2321(47.2%) 先手勝ち1192(24.2%) 後手勝ち1129(23.0%)

宣言勝ち2 先手宣言勝ち1 後手宣言勝ち1 先手引き分け49 後手引き分け34

///

計測はTanuki Coliseum(https://github.com/nodchip/TanukiColiseum / 使ったのは古いバージョン)で行った。

いずれもengine1がHaojian_240317

FV_SCALEは、水匠5とBLOSSOMが24、Háoが20、Haojian_240317は軽く計測して1番強そうだった値に設定。

対BLOSSOMは設定をミスっているのか、相性の問題なのか、開始局面の問題なのか、異様に成績が良い。

という結果になっていて、floodgateでの計測でもHáoより少しだけレートが高いという結果で、少なくとも弱くはなっていさそうなので学習部に致命的レベルのバグはないと思います。多分

Haojianという名前は、ベースになっているHáoと、某ゲームに出てくる災いの剣からつけています。

・自動定跡生成時の局面評価で、「方策出力のみで指すDL系AIによる連続対局の結果」と「浅い探索のNNUE系AIによる連続対局の結果」と「長い時間探索したAIによる連続対局の結果」を組み合わせる予定です。

「方策出力のみで指すDL系AIによる連続対局」には「GPUを用いる事でそこそこの棋力で同時に何千局も行える」という大きなメリットがあるため、定跡生成や教師データ生成の一部で使用できると考えています。

詳細は未定ですが、

「方策出力のみで指すDL系AIによる連続対局の勝率」と「浅い探索の

NNUE系AIによる連続対局の勝率」が大きく違なる場合、局面の難易度が高いと判断し「長い時間探索したAIによる連続対局」を多めに行う

といった事も行いたいと考えています。

本番までに余裕があったらやる事

・持ち時間管理部を改造する

形成に大きな差がついてからは持ち時間に差があっても逆転する事は難しいと思うので、それを念頭に置いた持ち時間管理を行いたいと考えています。

詳細は決まってないですが、DR4バージョンの序盤モードの持ち時間管理をベースに作ろうと今のところは考えています。

[DR4バージョンの序盤モードの持ち時間管理(ざっくり)]

1.事前に決めておいた「中終盤のために残しておく時間」を現在の残り時間から引く

2.「現在の局面の終局までの推定手数」から、事前に決めておいた「終局までの推定手数がこの値以下になった時に中終盤モードに移行する値」を引いたものを「中盤までの推定手数」とする

3.残り時間を「中盤までの推定手数」で割ったものに、加算時間や秒読みを足したものを「今の手番で使う時間(のベース)」に設定

4.一定の条件を満たした場合は予定の時間より早く打ちきったり、逆に延長したりする。(時間は余裕を持って計算しているので足りなくなる事はあまりない)

ちなみに、本番では「終局までの推定手数」の精度が低かったせいで思うように行きませんでした。

~~・局面生成AIを教師データ生成などに応用する~~

~~DR4では簡単な局面を生成できるところまではいったものの、色々あってGPU資源がなくなってしまったために結局応用する事はできませんでした。~~

~~いつかは覚えてないですが、DR4のアピール文書~~
~~(https://drive.google.com/file/d/16P0Gcv4K0yo8v4GGH2zo2QQ290EnWeHd/view)に書いていた「評価関数の弱点の克服方法」について、より良い案を思いついたので余裕があったら実装したいと考えています。~~

=> 間に合いそうにないので今回はやりません。

ただ、個人的には局面生成AIにゼロからの強化学習くらいロマンを感じるので、いつかリベンジする予定です。

使用予定のライブラリ

・python-dlshogi2(https://github.com/TadaoYamaoka/python-dlshogi2)

シンプルな構造でそれなりに強く、かつ使い慣れているため、Ari ShogiやNNUE学習部のベースとして採用しています。

・やねうら王(https://github.com/yaneurao/YaneuraOu)

自分が知っている「NNUEが利用できる探索部」のなかで最強なので、NNUE系の探索部として採用しています。

・dlshogi(https://github.com/TadaoYamaoka/DeepLearningShogi)

探索部のパラメータ調整スクリプトや、教師データの変換スクリプトが便利なので、その部分だけ利用しています。

・KomoringHeights(https://github.com/komori-n/KomoringHeights)

USIプロトコルで利用できて、かつ性能が良さそうなので、詰め将棋エンジンとして採用しています。

使用予定のデータ / 公開モデル

・AobaZeroのデータ(http://www.yss-aya.com/aobazero/)

・floodgateのデータ(http://wdoor.c.u-tokyo.ac.jp/shogi/index.html)

・水匠1手200万手のデータ
(https://drive.google.com/file/d/1R9kI3xDKeoIjyFPDORS6K-1wwck

o75fr/view ）

・水匠1000万ノードの公開データ
（ https://drive.google.com/file/d/1VyP4MX_AuQhvy8sesymgPVf9sUnQoGPl/view ）

・dlshogi_with_GCTのデータ
（ https://tadaoyamaoka.hatenablog.com/entry/2021/05/06/223701 ）

・書籍「強い将棋ソフトの創りかた」のデータ

いずれのデータも、質も量も十分以上なため採用しています。

・Qhapaq Pretty Derbyのデータ
（ https://qhapaq.hatenablog.com/entry/2021/11/23/220251 ）

採用理由：振り飛車評価関数を簡単に作れるから

振り飛車をコンセプトにするわけではないですが、振り飛車評価関数を一部で採用する予定なので、その振り飛車評価関数の学習に利用します。

~~・shogi_suisho5のデータ_depth9（ https://huggingface.co/datasets/nodchip/shogi_suisho5_depth9 ）~~

~~・shogi_hao_depth9のデータ（ https://huggingface.co/datasets/nodchip/shogi_hao_depth9 ）~~

~~ゼロからNNUEを学習させる事ができる量/質である事が分かっているデータなので、ゼロからNNUEを学習させる場合は採用します。~~

SSDの容量が足りないし、ゼロからNNUEを学習するのを断念したので多分使いません。

・Háo（ https://github.com/nodchip/tanuki-/releases/tag/tanuki-.halfkp_256x2-32-32.2023-05-08 ）

・BLOSSOM（ https://twitter.com/senninha_a/status/1654675205464395

9872 ）

・水匠5(https://github.com/mizar/YaneuraOu/releases/tag/v7.5.0 ）

・Lí(https://github.com/nodchip/tanuki-/releases/tag/tanuki-wcsc33-2023-05-04 ）

~~ゼロからのNNUE評価関数を学習させるのを断念した場合は、NNUE評価関数のベースとして採用する予定です。(単体でベースにする以外にも、キメラ化したものをベースにする可能性もある ）~~

~~特にHáoは、3月末の時点で1番強い評価関数(Háoよりレート10～20ほど強い)のベースになっているので、採用される確率が1番高いです。~~

採用理由：強いから

3月末の時点の状況だと、NNUEをゼロから学習させるのは無理そうなのでこれらの評価関数を組み合わせて作ったキメラ評価関数をベースにする予定です。

標準型NNUEを作るのに標準型以外のNNUEを素材として使用しり予定になっているのは、ネットワーク構造がある程度違うNNUEを使ったキメラも作れる方法を採用する予定だからです。

その他

・本当は会場に行って参加したいですが、家から遠いので今年もオンライン参加です。

・アピール文書は後で追記したものに差し替える予定です

以下、おまけなので文章とかより一層雑です

おまけ1：モデル精度について色々

モデル名	パラメータ数	方策正解率	価値正解率	備考
WCSC33のモデル	1687306	0.443736	0.7185835	テストデータの一部が学習データに含まれている
23年11月モデルA	3213186	0.445989	0.7198876	テストデータの一部が学習データに含まれている
23年11月モデルB	3213186	0.4552351	0.7279302	テストデータの一部が学習データに含まれている
DR4で使ったアンサンブルモデル	8113678	0.463778	0.7344486	WCSC33のモデル + 23年11月モデルA + 23年11月モデルB
SplatBrella_231219のアンサンブルモデル	12296782	0.475275	0.7415995	WCSC33のモデル + 23年11月モデルB + python-dlshogi2のモデル
WCSC34に向けて育成中のモデルの1つ	7164218	0.463992	0.727845	学習途中でまだまだ伸びそう
python-dlshogi2のリポジトリのモデル	7396290	0.4785397	0.7409203	
GCT電竜	?	0.46163161	0.73495564	データは https://tadaoyamaoka.hatenablog.com/entry/2021/09/11/155938 より引用
dlshogi with GCT	?	0.48964214	0.75278598	データは https://tadaoyamaoka.hatenablog.com/entry/2021/09/11/155938 より引用
dlshogi dr2_exhi	?	0.52322546	0.76564239	データは https://tadaoyamaoka.hatenablog.com/entry/2021/09/11/155938 より引用
tanuki-wcsc28	32117089	-	0.66378727	方策出力が無いので価値正解率のみ
水匠5	32117089	-	0.68430669	方策出力が無いので価値正解率のみ
BLOSSOM	32117089	-	0.68488171	方策出力が無いので価値正解率のみ
Háo	32117089	-	0.68491119	方策出力が無いので価値正解率のみ

・データはGCTの公開ノートブック
(https://tadaoyamaoka.hatenablog.com/entry/2020/11/26/203912)で使われているテストデータと同じ

・パラメータ数は
https://rheinmetall.hatenablog.com/entry/2021/05/14/000000 のコードで測定

・「23年11月モデルA」は「入力特徴量がバグっていたのに気づいて途中で学習を止めたモデル」

・「python-dlshogi2のリポジトリのモデル」は
https://github.com/TadaoYamaoka/python-dlshogi2/blob/main/checkpoints/checkpoint.pth のモデルの事。精度を比較したり、アンサンブルの効果を確かめたりするために利用している。

・比較のために、GCT電竜、dlshogi with GCT、dlshogi dr2 exhiのデータを https://tadaoyamaoka.hatenablog.com/entry/2021/09/11/155938 から引用している。

・昔の記録を発掘して、そこに書いてある数値を確認せずにコピペしているだけなので、間違っている可能性がある

・SplatBrella_231219は、モデルとパラメータを軽く弄って(ついでにいらない機能をOFFにして)floodgateに放流していた Ari Shogi and フレンズ。ペタショック定跡に(有利な先手番でだが)1発入れた時のもの。

http://wdoor.c.u-tokyo.ac.jp/shogi/view/2023/12/19/wdoor+floodgate-300-10F+SplatBrella_231219+YOV800-peta4M-3700X+20231219190001.csa

SplatBrellaという名前は、某イカのTPSゲームで私がここ数年愛用している傘の英名からとっています。

[感想とか考察とか]

・DR4のモデルでは、単体での精度が1番高いモデルから正解率が方策/価値ともに上昇している。探索速度は落ちたものの、DR4での構成では探索速度の低下によるデメリットをあまり感じなかったので、大会直前にこれを実装したのは成功だったと思う。

・SplatBrellaのモデルは、単体での精度が1番高いモデルからほとんど変わっていない。(むしろ若干下がっている)

1番精度の高いモデルと2番目以降のモデルの精度に大きな差があったので、足を引っ張ってしまったのかもしれない。(が、より複雑な方法でアンサンブルを行った場合は、これでも精度が上がる可能性があると自分は考えている。この辺は余裕があったら実験する。)

=>(追記)「1番精度の高いモデルと2番目以降のモデルの精度に大きな差があったため、足を引っ張って精度が上がらなかった」という説を確かめるための実験を行った。

モデル名	パラメータ数	方策正解率	価値正解率	備考
WCSC33のモデル	1687306	0.443736	0.7185835	テストデータの一部が学習データに含まれている
23年11月モデルA	3213186	0.445989	0.7198876	テストデータの一部が学習データに含まれている
23年11月モデルB	3213186	0.4552351	0.7279302	テストデータの一部が学習データに含まれている
DR4で使ったアンサンブルモデル	8113678	0.463778	0.7344486	WCSC33のモデル + 23年11月モデルA + 23年11月モデルB
SplatBrella_231219のアンサンブルモデル	12296782	0.475275	0.7415995	WCSC33のモデル + 23年11月モデルB + python-dlshogi2のモ
WCSC34に向けて育成中のモデルの1つ	7164218	0.463992	0.727845	学習途中でまだまだ伸びそう
python-dlshogi2のリポジトリのモデル	7396290	0.4785397	0.7409203	
GCT電竜	?	0.46163161	0.73495564	データは https://tadaoyamaoka.hatenablog.com/entry/2021/09/11/155
dlshogi with GCT	?	0.48964214	0.75278598	データは https://tadaoyamaoka.hatenablog.com/entry/2021/09/11/155
dlshogi dr2_exhi	?	0.52322546	0.76564239	データは https://tadaoyamaoka.hatenablog.com/entry/2021/09/11/155
tanuki-wcsc28	32117089	-	0.66378727	方策出力が無いので価値正解率のみ
水匠5	32117089	-	0.68430669	方策出力が無いので価値正解率のみ
BLOSSOM	32117089	-	0.68488171	方策出力が無いので価値正解率のみ
Háo	32117089	-	0.68491119	方策出力が無いので価値正解率のみ
WCSC34に向けて育成中のモデルの1つ(その後)	7164218	0.4717394	0.7350589	伸びしろはもうそんなに残ってなさそうだが、まだ少しは伸び
2024年3月30日の実験モデル	14560508	0.4842954	0.7458868	pydlshogi2のモデル + 育成中のモデル(その後)

(表の赤色のところが新しく調べた場所)

「2024年3月30日の実験モデル」は、「WCSC34に向けて育成中のモデルの1つ(その後)」と「python-dlshogi2のリポジトリのモデル」の2つのモデルの出力を単純に平均しただけのものだが、ちゃんと正解率が上昇している。

なので、「SplatBrellaのモデル精度が、python-dlshogi2のモデル単体からほとんど上がらなかったのは、他のモデルが足を引っ張っていたから」という説は多分あっている。

ただ、あるモデルが"有望ではない"としてほぼ0%を出力した指し手でも、他のモデルがある程度の値を付けた場合は探索される可能性が出てくるため、実際に探索に組み込んだ際はテストデータでの精度以上に違いが出る(良い方向でも悪い方向でも)のではないか、と考えている。(試せてはない)

・DR4で使った3つのモデルは「2023年に作ったDL評価関数ランキングtop3」だが意外と精度が出ていない。

「WCSC34に向けて育成中のモデル」は、学習させ始めてから3日くらいしか経ってないのに、DR4のアンサンブルモデルと同じくらいの精度が出ている。(DR4のモデルはテストデータの一部が学習データに含まれているというハンデがあるので、実際の精度は育成中のモデルのほうが高い可能性もある)

(2024年1月の末にPCが新しくなるまでは)GoogleColab無料版などの「無料で利用できるGPU」を使ってチマチマと学習を進めていた(一応、メインのノートPCにはGPUが載っていたが、酷使しすぎたせいか、GPUに負荷をかけるとバッテリー周りの挙動が怪しくなるようになっていたので、あまり学習には投入できなかった)のが、新しくなってからはRTX4060Ti(メモリ16GB)がほぼ24時間フル稼働、と環境が大きく変わったのはあるが、それにしても「3日学習させたモデル >= 去年1年間の集大成ともいえるモデル」というのは少し残念。

おまけ2: DR4バージョンのAri Shogi and フレンズのfloodgateでの計測

ハードウェア含め本番と同じ構成で計測。対局数が少ないのでレートはあまり信頼できないが、少なくとも4000は超えていると思う。

asa35	4257	61	35	0.635	2023-08-08	4168
SV-037	4254	75	47	0.614	2023-09-03	4165
test_13900k	4252	39	15	0.712	2023-06-29	4163
b22sd	4242	45	30	0.598	2023-12-05	4153
ECLIPSE-20230523_R9-5900HX	4241	24	9	0.728	2023-07-25	4152
GloriousMoment	4240	22	5	0.799	2023-11-28	4151
FAREWELL_R9-5900HX	4234	221	136	0.619	on line	4145
Sagittarius	4232	104	68	0.603	2023-12-02	4143
VW201	4229	16	7	0.691	2023-06-18	4140
Ari_Shogi_and_Friends_DR4	4221	12	4	0.735	2023-12-05	4132
S.novi-belgii	4219	16	1	0.933	2023-10-15	4130
GGG	4217	52	29	0.640	2023-09-14	4128
ECLIPSE-20230717_R9-5900HX	4215	31	16	0.659	2023-07-25	4126
ECLIPSE_Ryzen9-5900HX	4214	26	14	0.648	2023-07-03	4125
dltest	4210	14	4	0.758	2023-10-17	4121
KANROJI_MITSURI	4209	31	15	0.662	2023-11-03	4120
Hao-s-book_black-i5-13400	4204	21	2	0.913	2023-12-06	4115
nnue-tanuki-dr3_5800u	4203	51	32	0.616	2023-11-22	4114
SV-q044	4200	222	36	0.859	2023-10-19	4111
B1112	4199	343	240	0.588	on line	4110
HoneyWaffle-37-003	4195	879	666	0.569	2023-12-02	4106

引用元：http://wdoor.c.u-tokyo.ac.jp/shogi/x/rating/players-floodgate-20231207.html

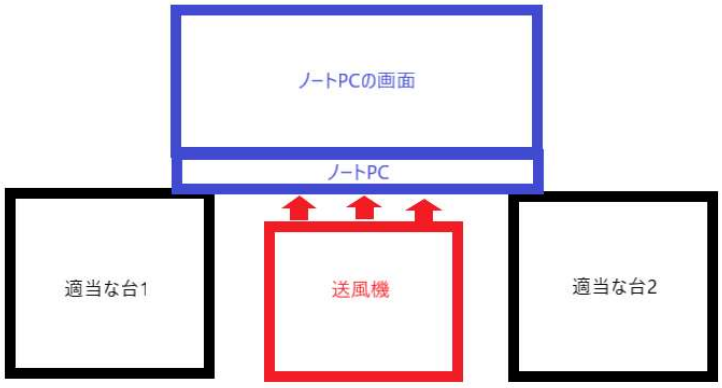
おまけ3. 第4回電竜戦本戦の感想とか

・大会前は「モデル精度が低い」「モード切替機能が微妙」「定跡がない」「探索部や合議パラメータを本来は連続対局の結果を元に自動調整する予定だったが、リソース不足でちゃんとできなかった」「無理な運用を続けたせいでハードウェアが劣化してきている」といった理由から、「あまり良い結果にはならなさそう」と予想していた

・特にハードウェアは、大会前/大会中に故障しないかヒヤヒヤしていた。長期間無理な運用をしてきたせいなのか、GPUに強い負荷をかけると「バッテリーが認識されなくなる」や「電源に接続されているはずなのにバッテリーで稼働中(電源に接続されていない)と表示される」といった現象が発生したし、比較的軽めの負荷でも長期間稼働させると「いつの間にか電源が落ちている」や「画面が真っ暗になり、一切の操作を受け付けなくなる(が、

本体は高温になっているしファンも稼働しているので恐らく電源はついている)」といった現象が発生した。

・大会中も、ずっと電源に接続しているのにも関わらず「バッテリーが検出されません」みたいな表示と「充電中」という表示が高速で切り替わる、という事が起きていた。多分そのせいで、大会中は下の画像のようにしてPCを冷やしていたのに、CPUのクロックがかなり落ちてしまっていた。



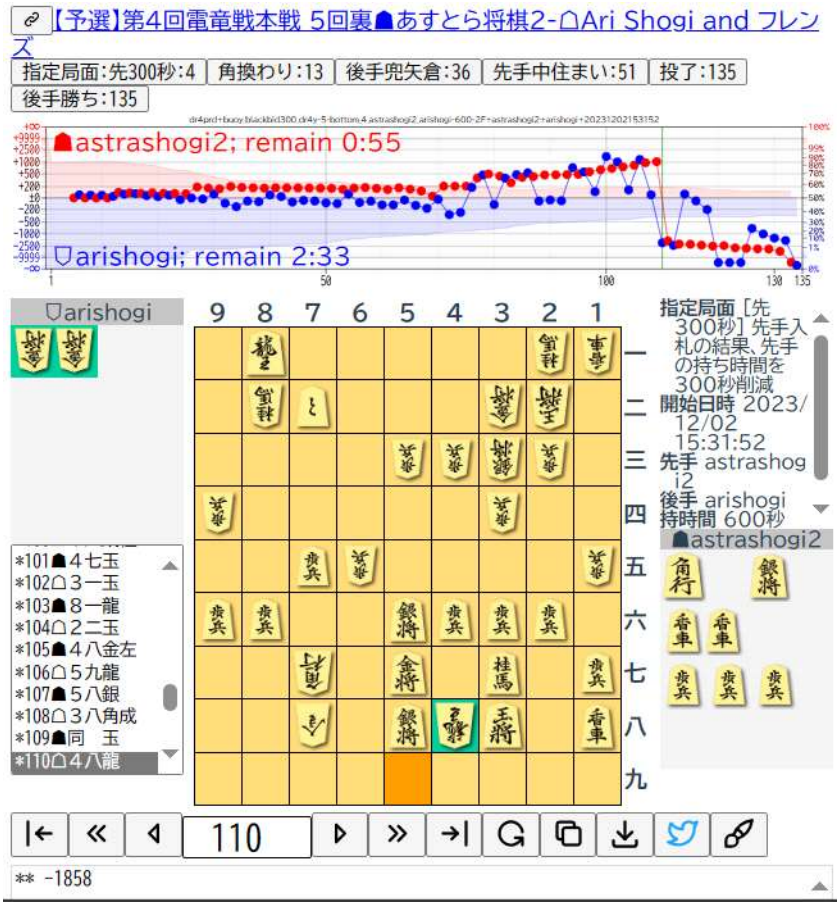
・大会で使ったPCは、大会後の12月の末に「動作中に異臭がする」という事が起きてしまったので封印する事になった。一応起動もできるし、データは無事で取り出す事もできるが、火事になるのは絶対に避けなければならないので、(データを取り出すなどの)理由がない限りは起動しないようにしている。

PCが壊れたせいで、1ヵ月くらい開発速度が大きく低下し、モチベーションも激減したが、「気づかないうちにPCから出火して家が火事になる」とか「大会前/大会中に壊れる」とか「(バックアップ取ってないのに)データが全部消える」とか、そういう事が無かったのでまだ良かったと思う。

・それでもB級に進出し、しかもB級最下位を回避したのは、非常に運が良かったからだと思う。(実際、C級上位陣や、2日目に参加しなかったQhapaqチームやアストラ将棋チームのほうが強いと思う)

WCSC33も強運で1次予選を突破していたし、2023年は将棋AIに関しては非常に運がよかった。(2023年の初詣で引いたおみくじの結果はたしか"凶"だったけど)

・個人的に1番嬉しかった勝ちは1日目5回裏の対あすとら将棋2戦。相手が格上だったので、5回表の先手千日手で満足していたのだが、終盤に相手が読み抜けしたので、2戦1.4勝0.6敗という想像もしなかったレベルの非常に良い結果に終わった。



引用元: https://denryu-sen.jp/denryusen/dr4_production/dist/#/dr4prd+buoy_blackbid300_dr4y-5-bottom_4_astrashogi2_arishogi-600-2F+astrashogi2+arishogi+20231202153152/110

・個人的に1番悲しかった負けは2日目の5回表の対ECLIPSE戦。後手番で600以上まで行ったのに、そこから逆転して負けた。

引用元：https://denryu-sen.jp/denryusen/dr4_production/dist/#/dr4prd+buoy_blackbid300_dr4b-5-top_4_eclipse_arishogi-600-2F+eclipse+arishogi+20231203141035/91

nshogi アピール文章

中屋敷 太一

概要

- nshogiはAlphaZeroの手法をベースとしています．
- AlphaZeroの手法に加え，いくつかの工夫を行っています．
- 指し手生成や探索，評価関数の学習など，主なコンポーネントはフルスクラッチで実装しています．
- 中屋敷は去年までTeam Noviceのメンバーで出場していました．
  - 今年は，新たに，Noviceとソースコードを一切共有しないコンピュータ将棋ソフトウェアを実装しました．
  - それをnshogiとして，Noviceとは別にエントリーしています．
  - （技術的な交流などはTeam Noviceのメンバーと引き続き行っています．）
- 学習データにWCSC 33までのNoviceのデータを使用しているため，フルスクラッチ申請を行いません．

探索

- AlphaZeroと同様，pUCTを用いたMonte-Carlo Tree Searchを行っています．
- ゲーム木のリーフ局面の評価に，ニューラルネットワークによる評価に加え，Depth First Searchによる5手詰みの探索を行っています．
  - 探索スレッドとは別のCPUスレッドで詰み探索を行います．
- AlphaZeroがゲーム木内の各ノードの勝率を保持していることに加え，nshogiは各ノードの引き分け確率を保持しています．

評価関数

- 評価関数にはニューラルネットワークを用いています．
- 30 Block 256 Filter のResidual Networkの構成のニューラルネットワークで局面を評価しています．
- 現局面を入力に取り，その局面の勝率，次の一手の選択確率分布，引き分け確率，現局面の利きの4つを出力します．
  - 現局面の利きは学習時の補助タスクとしてのみ用い，対局時には用いません．
- 学習に542,709,386局面（重複あり）を用いています．

定跡

- floodgateの棋譜から生成しています．
- 勝率などの指標をもとに，定跡として登録する手を選んでいます．

その他

- 探索におけるメモリ確保/解放が頻発したため， segregated free listを用いたメモリ管理を行うことで，メモリ確保/解放のオーバーヘッドを減らしています．
- リーフ局面を展開する際に行われるメモリ確保を，ニューラルネットワークによる局面の評価中に，別スレッドにより行います．

Last updated 2024-03-29 20:25:01 +0900

第 34 回世界コンピュータ将棋選手権 参加ソフト

ねね将棋 アピール文書

日高雅俊

2024/04/29

概要

iPhone 上で機械学習専用チップを活用し、深層学習ベースの将棋 AI を高速に動作させる。最新の iPhone 15 Pro を採用し、過去に使用していた iPad(第 9 世代)の約 5 倍の計算速度が得られる。探索速度は約 5000NPS (nodes per second)である。Floodgate でのレートは 3712（26 勝 24 敗、2024 年 4 月 29 日時点)。

図 1. 動作画面

技術要素

図 2. ソフトウェアの構造

ねね将棋は iPhone 上で動作するソフトウェアである。構造を図 2 に示す。GUI は、対局中

の盤面の表示や、対局サーバへの接続設定の受付を行う。UI フレームワーク SwiftUI により実装されている。通信部は、現在の盤面を把握し、思考部と USI プロトコルで通信し、対局サーバと CSA プロトコルで通信する。Swift 言語で実装された盤面表現を用いて、プロトコルの変換を行っている。思考部は指し手を決定する機構であり、C++言語で実装されたふかうら王（やねうら王の一種で、深層学習系評価関数を用いたモンテカルロ木探索機構を有する）を、iOS プラットフォーム向けに独自ビルドしたものを用いる。通信部との連携は、C++言語の標準入出力（通常、将棋所等とのプロセス間通信の手段となる）を Swift 言語のコールバック関数呼び出しに置換することによりプロセス内で完結させる。Core ML アダプタは、評価関数である深層学習モデルをふかうら王から呼び出すための機構である。Core ML は、Apple が提供する深層学習モデルの実行エンジンであり、機械学習専用チップ Neural Engine を利用することでモデルを高速に実行可能とする。Core ML アダプタは、Objective-C++で実装され、ふかうら王の一部としてビルドされる。評価関数には、dlshogi 系評価関数を使用（書籍「強い将棋ソフトの創りかた」サンプルコードにより学習した 20 層 192 チャンネルの CNN）する。PyTorch で学習されたモデルを、Core ML で使用される Apple 独自のモデル形式に変換して用いる。

対局サーバとの通信では、会場に用意される有線 LAN（イーサネット）を用いることで無線と比べ信頼性の向上を図る。iPhone を有線 LAN に接続することは一般的ではないが、USB 接続のイーサネットアダプタ ETX3-US2 (I-O DATA)により実現できる。

ねね将棋は iPhone の計算能力を最大限活用するものであるため、稼働中の発熱が大きい。長時間の対局を安定させ速度低下を避けるため、外付けファンによる強制空冷を実施する。

図 3. iPhone にファンを取り付けた状態

過去との差分

最新の iPhone 15 Pro を採用。ソフトウェアは、Swift 言語のみで思考部含め実装した WCSC32 と、ふかうら王を iOS 向けにビルドした WCSC33 (ただし USI と CSA の変換は Mac 上で動作する将棋所を利用)の実装を統合し、USI と CSA の変換機構を新規実装。

使用予定ライブラリ

- ふかうら王（やねうら王の一種。深層学習系評価関数により主に機械学習専用チップ Neural Engine で思考）
  - dlshogi 系評価関数を使用（書籍「強い将棋ソフトの創りかた」サンプルコードにより学習した 20 層 192 チャンネルの CNN）

ソースコードを公開しています。
https://github.com/select766/NeneShogi2024
各技術要素の詳細は、ブログをご覧ください。
https://select766.hatenablog.com/archive/category/%E3%82%B3%E3%83%B3%E3%83%94%E3%83%A5%E3%83%BC%E3%82%BF%E5%B0%86%E6%A3%8B

iPhone で将棋 AI を動かすノウハウ

単に将棋が指せるソフトができたあと、大会で安定して動作させるためのノウハウです。

- 機内モード・おやすみモードにする
  - 電話が着信すると困る。
- 有線 LAN へ接続する
  - USB 接続できる有線 LAN アダプタが存在する。
- プロビジョニングプロファイルの有効期限に注意
  - App Store で配布していない独自アプリは、一定期間で起動不能となる。
- 端末を冷却する
  - 端末が熱くなると計算速度が低下する。ゲーム向けの冷却ファンが使える。
- 充電器への接続を確認する
  - 接続忘れを見落としやすい。

「習甦」

【探索】

1スレッドはdf-pnにより詰みを確認，他のスレッドはStockfishから取捨選択したシンプルなα-β探索

【評価関数】

玉の位置に対する各駒の位置と全升目の利き数（先後別3まで）を特徴量とするニューラルネットワーク

【機械学習】

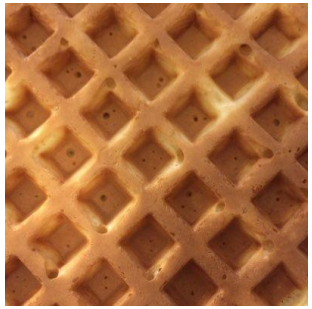
自己対戦において初手〜4手目までMultiPV24〜16に飛車を振る手を入れることによりオールラウンダーをめざす

ゼロベースから評価関数を効率良く学習するため，初期段階では入玉も考慮した駒割り関数の値を報酬に加算

HoneyWaffle

第34回
世界コンピュータ将棋選手権
アピール文書
開発者　渡辺 光彦

開発者



氏名: 渡辺 光彦

職業: プログラマー

棋力: 将棋ウォーズで2級、ぴよ将棋でR900-1000程度の振り飛車党

Twitter: @shiroi_gohanP (https://twitter.com/shiroi_gohanP)

ニコ生の電王戦をきっかけにコンピュータ将棋を始める。

将棋連盟Liveやニコニコ生放送、AbemaTVの将棋中継が好き。

note書いています！ → https://note.com/honeywaffleshogi

文春オンラインのインタビュー記事 → https://bunshun.jp/articles/-/14921

HoneyWaffle (ハニーワッフル) 名前の由来（去年と同じ）

・四角いワッフルは将棋盤と似ている

・ゆるふわスイーツ的なスナック感覚の軽さを表現

　元々タブレット向けに開発していたので物理的に軽いこと、振り飛車の軽い捌きができるようになるといいなという想いから命名しました。

　コンピュータ将棋といえば、人名 +将棋と命名するのが格調高いと思っています（森田将棋とか）。私が有名ではなく、将棋界で渡辺といえば渡辺明先生なので、「渡辺将棋」とは命名すべきではないと思いました。ということで、渡辺がだめなら光彦 →みつ→Honey、Waffleは上記のとおり将棋盤の意味。 いい命名じゃないです か？ すごくないですか？

以下のリンク先で出せるものは公開しています。使い方がおかしいのはいつものこと。

https://github.com/32hiko

コンセプト

青字は使用予定ライブラリ

「毎日毎日練り続けた振り飛車定跡でいい将棋をお見せしたい」

（1）振り飛車定跡

ほぼ毎日4時間程度、floodgateでの実戦譜をもとに定跡を作成する作業をしています。複数台のミニPCで参戦していますが、検討にはAWS EC2のm6a.48xlargeを使用しています。1日約60局前後の棋譜を検討すると月に10万円以上かかります。（昨年8月からなので、高いマシンを買うのと同じくらいの出費が…）前回大会では先手は三間飛車、後手は四間飛車が中心でしたが、先手では初手56歩系、後手では角交換系の定跡を追加しています。

（2）評価関数とエンジンはBLOSSOMとやねうら王7.61を使用

無償で使用できる最高クラスの評価関数とエンジンを使用。（有償化に反対する意図は全くありません、念のため） こちらも自分自身で開発したい思いもありますが、全然手が回っていません。

（3）マシンは定跡の作業でも使っているm6a.48xlargeを使用（1台）

最後に

コンピュータ将棋では振り飛車の評価値が低く出る傾向なのは以前から言われていますが、飛車を振って下がった評価値をそれ以上下がらないようにキープすることは、評価値の割には難しくありません。（相居飛車で100の差がついたら大変なイメージがありますが、振り飛車ではそうでもない実感があります）

また、振り飛車を指すことで相居飛車の深い研究（特に先手が必勝に近い戦型）を必ず回避でき、その上で逆にこちらの深い研究に誘導できるのはメタ的に無視できない利点だと思います。

楽しくやれるうちはコンピュータ将棋と振り飛車を続けていくつもりですが、仮に今大会が最後になってしまっても悔いのないようにがんばります。

第34回世界コンピュータ将棋選手権 アピール文書
プログラム名「タンゴ」
開発者　渡邊敬介
2024年3月30日

概要

実現確率探索による深い読みと、機械学習により最適化された正確な評価関数により、強力なコンピュータプレイヤの実現を目指します。

本プログラムの大きな特徴は、局面評価関数および実現確率用の着手確率にFactorization Machinesを使用している点です。2023年以前のコンピュータ将棋選手権でこのような手法を用いていたチームは、私の知る限りでは私のチーム以外に存在しません。

局面評価関数

本プログラムでは、駒の損得以外に局面評価関数の特徴量に下記の3つを採用しています。先後の対称性を保つため、先手から見た盤面と後手から見た盤面それぞれについて下記の特徴量を入力とするFactorization Machineによりスコアを計算し、算出された先後のスコアの差と駒の損得を合算してその局面の評価値としています。

1. 2駒の位置関係 (俗に言うKP + PP + KK)
2. 自玉周辺25マスの双方の効き
3. 手番

Factorization Machinesを使用することで、これらの入力特徴の組み合わせ特徴を取り込んだ極めて表現力の高い評価関数となっていると考えています。例えば、2駒の位置関係同士の組み合わせは4駒の位置関係(俗に言うKKPP + KPPP + PPPP)に相当します。

実現確率探索用の着手予測

Factorization Bradley-Terry モデル[1]を使用しています。通常のモデルと小規模で高速な簡易計算用モデルの2種類を用意し、探索中の末端付近ではこの簡易計算モデルを使用することで高速化を図っています。

また、Factorization Bradley-Terry モデルでは着手確率を計算するためにはその局面の全ての着手のスコアを計算した上でソフトマックス関数に通す必要があります。そこで、「駒の位置」や「王手がかかっている」などの盤面共通の特徴量を最初に計算し、各着手のスコア計算で再利用できるようにしています。さらに、駒の位置については差分計算可能なので、簡易計算モデルで使用される駒の位置の特徴量については差分計算を行っています。

追試可否

再現実験を可能とするため、対局用プログラムだけでなく学習に使用したデータセットも大会後1年間保管する予定です。

参考文献

[1]Xiao, Chenjun, and Martin Müller. "Factorization ranking model for move prediction in the game of Go." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. No. 1. 2016.

恭祐 アピール文書 WCSC34

開発者 ペンギンクミマヌ

2024年3月10日　作成
2024年5月3日　改訂

プログラム情報

プログラム名:恭祐

プログラム名の由来:知人がどうしても自分の名前を冠して欲しいと言ってたので

特徴: dlshogiベース

初参加:WCSCは初参加　第4回電竜戦でコンピューター将棋大会デビュー

使用ライブラリ: dlshogi,cshogi “強い将棋ソフトの創りかた”に準じる

採用している手法: “強い将棋ソフトの創りかた”に準じる

探索部:ふかうら王

コメント

第4回電竜戦のときは,強い将棋ソフトの創りかたの付属データとAobaZeroの棋譜から評価関数を作成しました。強さは定跡なしでfloodgate 3600-3800程度でした。電竜戦前は、定跡で棋力をカバーできるかと思っていましたが、本戦で棋力の必要性を痛感しました。ネットで公開されている棋譜からこれ以上の棋力向上に限界を感じたので、電竜戦終了後から自前で教師局面生成に取り組んでいます(1日1万局面しか作れず😂)。4月頃から~~その教師局面を混ぜて学習させ~~ようと思いましたが、先日たややん氏が公開された教師データも活用して学習させ,WCSC34で使いました。目標は2次予選進出です。

参考文献

・強い将棋ソフトの創りかた

・やねうらおうwiki

開発者情報

開発者名:ペンギンクミマヌ　(本名)山下公誠

SNS: X note

第34回
世界コンピュータ将棋選手権
なのはアピール文書

2024年3月31日 川端一之

■**なのは**ってなんだよ

- 熱血魔法バトルアクションアニメ「魔法少女リリカル**なのは**」シリーズの主人公**高町なのは**を由来にし、さまざまな称号を冠する彼女のような強さを盤上で実現したいという願いを込めています。

- よく「名前の割に強い」という声をいただきますが、その認識は逆で、「名前負けしている」や「名前の割に弱すぎる」というほうが妥当な評価です。

■作者はどんな人？

- 静岡県出身 愛知県在住
- とあるメーカーに勤務(ちょっとAWS使う)
- 好きな食べ物は焼肉、しゃぶしゃぶ、寿司
- 好きなアニメは魔法少女リリカルなのは、りゅうおうのおしごと！、痛いのは嫌なので防御力に極振りしたいと思います。、くまクマ熊ベアー、とある科学の超電磁砲、ラブライブ！、五等分の花嫁

  最近は「お隣の天使様にいつの間にか駄目人間にされていた件」がお気に
- 将棋ウォーズ 1級
- 囲碁は日本棋院 初段(アマチュア)

■開発環境

- こんなPCで開発しています

  Lenovo Thinkpad T14

  CPU：AMD Ryzen 7 PRO 4750U

  プレゼントでいただきました

- 出場PC

  Minisforum UM790Pro

  CPU：AMD Ryzen 9 7940HS

  RAM：32GB

  OS：Windows 11 Pro

■なのはの構成

- MSYS2のg++で開発
- 手生成では歩、角、飛の不成も生成
- 探索はStockfish使用
- Bitboard未使用(盤情報は配列)
- 定跡部は実戦での出現数および勝率を考慮して手を選択
- 評価ベクトルは3駒関係(KPPT)

…と、数年前に見たような平凡な構成

■今回の変更点

- 出場PCのパワーアップ(Ryzen 7 4750U → Ryzen 9 7940HS)
- [予定]探索部の強化(Stockfish 8改→ Stockfish 15.1改)
- [予定]評価関数の強化(配布教師データでの学習)

■意気込み

- 現地会場から参加！
- 0次予選突破！
- 出来れば勝ち越したい！
- 詰めルーチンを間に合わせたい！

■今後の野望(?)

- Ryzen AIの活用
- 評価関数のNNUE化
- ミクロコスモスを解く
- 非コピーレフト系ライセンスを採用して開発

■使用ライブラリについて

- なのはmini　Stockfishをベースに独自に将棋化しました。

■使用データについて

- 評価ベクトルの学習には、一般に流布された教師データを使う予定
  - やねうらおさん配布の110億教師データ
  - nodchipさん配布の教師データ

■最後に

- **なのは**アピール文書は以上です
- 最後まで読んで頂きありがとうございます

絵： COCOさん

■参考文献

- 小谷善行、他:「コンピュータ将棋」, サイエンス社, 1990.
- 松原仁 編著:「コンピュータ将棋の進歩」, 共立出版, 1996.
- 松原仁 編著:「コンピュータ将棋の進歩2」, 共立出版, 1998.
- 松原仁 編著:「コンピュータ将棋の進歩3」, 共立出版, 2000.
- 松原仁 編著:「アマ四段を超えるコンピュータ将棋の進歩4」, 共立出版, 2003.
- 松原仁 編著:「アマトップクラスに迫るコンピュータ将棋の進歩5」, 共立出版, 2005.
- 池泰弘:「コンピュータ将棋のアルゴリズム」, 工学社, 2005.
- 金子知適, 田中哲朗, 山口和紀, 川合慧:「新規節点で固定深さの探索を併用するdf-pnアルゴリズム」, 第10回ゲーム・プログラミングワークショップ, pp.1-8, 2005.
- 脊尾昌宏:「詰将棋を解くアルゴリズムにおける優越関係の効率的な利用について」, 第5回ゲーム・プログラミングワークショップ, pp. 129-136, 1999.
- 保木邦仁:「局面評価の学習を目指した探索結果の最適制御」 http://www.geocities.jp/bonanza_shogi/gpw2006.pdf
- 岸本章宏:「IS 将棋の詰将棋解答プログラムについて」, http://www.is.titech.ac.jp/~kishi/pdf_file/csa.pdf, 2004.
- 橋本剛, 上田徹, 橋本隼一:「オセロ求解へ向けた取り組み」, http://www.lab2.kuis.kyoto-u.ac.jp/~itohiro/Games/Game080307.html

■参考Web

- やねうら王 公式サイト: http://yaneuraou.yaneu.com/
- 千里の道も一歩から: http://woodyring.blog.so-net.ne.jp/
- 小宮日記: http://d.hatena.ne.jp/mkomiya/
- State of the Digital Shogics [最先端計数将棋学]: http://ameblo.jp/professionalhearts/
- ながとダイアリー: http://d.hatena.ne.jp/mclh46/
- 毎日がEveryday: http://d.hatena.ne.jp/issei_y/
- Bonanzaソース完全解析ブログ: http://d.hatena.ne.jp/LS3600/
- aki.の日記: http://d.hatena.ne.jp/ak11/
- FPGA で将棋プログラムを作ってみるブログ: http://blog.livedoor.jp/yss_fpga/

※読めなくなったサイト含む

Argo（アルゴ）WCSC34 アピール文 2024.3.31

ソフト名/soft-name：Argo（アルゴ）

開発者/developer：市村豊（いちむらゆたか/Yutaka Ichimura）

X（旧Twitter）:@argonworks

Blog：http://blog.livedoor.jp/argon1/

2024.1.31時点

WCSC32のときの評価関数をそのまま使って参加させて頂こうかと思っています。

WCSC32の作文に記載した内容を以下に転載します。

＊＊＊＊＊＊＊＊

「水匠 3 改」を元にして以前やねさんが配布していた 150GB の教師データを 3 つ使って学習させたものが、水匠 5 に対して 50 回くらい対局させて勝率が 40%くらい（20 勝 30 敗くらい）になりました。現状はそれでやろうかと思っています。

使用ライブラリとそれを使用した理由

やねうら王：使いやすくて改造をしやすいから。

水匠 3 改：色々と評価関数を作っていてこれを元にしたのが現状で一番良さそうだったからです。

＊＊＊＊＊＊＊＊

**********以上です。以下はおまけです。

この文章を読んでいただきましてありがとうございます。あなた様の大変貴重な時間を使ってこの文章を読んでいただけること、感謝しております。

以下の文章は無理に読む必要はありません。個人的にはコンピュータ将棋選手権のアピール文を読む人にとって興味深く読めそうな作文を以下に書いたつもりでおります

2024.1.31時点 気が向いたら2024.3.31までにもうちょっと加筆修正するかもしれないです。
以下は、アピール文のおまけの作文の草稿です。現状の分を載せておきます。
読みたくなければ無理をして読まなくて良いです。

この作文を読んでいるということはあなたは頭が良い人だということです。
コンピュータ将棋の大会のアピール文を読んでいるという、そんなことをしている時点であなたは頭が良い人だということです。
あなたが、自分が頭が良いということを自覚しているのならばそれで良いのですが、自覚していないのならば自分は割りと頭が良い人間であるということを自覚してもらって良いです。

> 人生は他人との競争ではない話。
囲碁の名人と将棋の名人は争っていない。それと同じように私やあなたは他人とは争っていない。人生は他人との競争ではない。将棋の名人が囲碁の名人よりも劣っているということがないように、あなたが他人よりも劣っているということはない。

仮に競争であるとしたら、一番になるよりも最下位になる方が良いのではないかと言う風に思う。一番になるということは「動物園のサルの飼育員」とか「幼稚園の先生」みたいなもので、周りの人の面倒を見ないといけなくて非常に面倒でしかない。

そもそも大体の人は一番ではない。
「一番以外は一番ではなく、大体の人は一番ではない。なので、一番ではないことは標準的なことであり一番の人が例外的」ということ。
あなたが今どういう状況かは知りませんが、「自分は今現在すごく良い状態で勝ち組です」というのでしたらそれは良いことだと思いますが、そもそも世の中の大体の人は多分勝ち組ではありません。だから私やあなたが勝ち組ではないとしてもそれは全くもって標準的な状態であるということです。
私がこれまで根拠なく誤解していたことですが、大体の人は負け組であって、勝ち組のほうが例外的な少数派ということです。だから私やあなたが負け組の状態にあるとしてもそれは単に人間として標準的

な状態にあるというただそれだけの話しです。人間的に立派であるとか、能力が高いとか、価値があるとか言うことは大体の人はない。だから劣等感を感じて苦しむ必要はまったくない。

もちろん可能であるならば勝ち組になる方が良いとは思います。立派であるとか能力が高いとか価値があるとかいう人間になるために努力する方が良いとは思います。最終的に死ぬまでずっと負け組で終わるとしても、それでも努力するのは意味があることだと思う。というか、負けても負けても闘い続けるというその事それ自体が十分に立派なことだと思います。結果的に勝つかどうかというのは副次的なことだと思います。

具体例として明治維新のときの新選組の土方歳三という人はかなり人気がある人だと思います。ご存知のように明治維新において新選組は最初から最後までずっと負け続けます。土方歳三は敗走を重ねながら最後までずっと戦い続けた人です。それだから人気なのです（「ゴールデンカムイ」という人気のマンガがありますが、読んでもらえばわかりますが土方歳三が影の主人公の話です）。

日本のアニメやマンガのストーリーは「序盤から中盤にかけて負けているが最後に勝利する」という内容のものが多い印象を受けますが、実際のところは、「大体の人は、人生は最初から最後までずっと負け続ける」ものなのではないかと私は思っています（それなので、私やあなたが人生において最初から最後までずっと負け続けるとしてもそれは人間として標準的であるというだけであって、標準よりも劣っているということはない）。それでも結果的に勝つかどうかはともかくとして、勝ちを目指して挑戦を続けることそれ自体に意味があることだと思います。やったことがある人はわかるでしょうが、そもそも勝つことよりも負けても負けても闘い続けることのほうが難しいことです。勝っているから続けるというのは技術的には難しくとも精神的にはそこまで難しくはない。負け続けているのに闘い続けることのほうが技術的には難しくなくとも精神的には難しいことです。それなので、負け続けているのに闘い続けている人は勝っている人と同じくらいに立派なのです。

他人の不幸を願う理由がない

「他人の幸福を自分の幸福のように喜び、他人の悲しみを自分のことのように悲しむこと」それが人間にとって最も重要なことである、というのが「ドラえもん」の結婚前夜のエピソードに描かれている藤子・F・不二雄のメッセージです。そして、それができない状況のときというのは、大体は自分の側に問題があることが多い。そのことが自覚できているのならばまだ大丈夫だと個人的には思っています。

他人に対する評価というのは自分に対する評価の反映だと思います。やたら他人に文句を言っている人

というのは、本人が問題を抱えていることが多いです。そもそも悪口というのは自分が言われたら嫌なことを言うわけです（それなので悪口というのはそれを言われている人ではなくて、それを言っている人の欠点を表していることが多い）。本人が自分に満足していたら他人に文句を言わないと思います。我ながら、そもそも他人の批評という生産性があまりないことにリソースを投じている時点で負けている感じがします。興味関心は自分の未来に向けたほうが良いと思います。過去と他人にこだわるべきではない。変えることができるのは未来の自分だけなので、自分の未来のことに意識を集中させたほうが良いと思います。

自分が得をすることをする。

現在の日本において経済学と言ったら大体は新古典派経済学を指すと思うのですが、その基本的な考え方は「人間は自分が得をすることをする（人間は自分が損をすることはしない）」「人間は常に合理的に振る舞う」ということを仮定して社会現象を分析するものだと個人的には思っています（社会全体を大きな将棋盤だとみなして、人間一人一人が駒であると考えて盤上の動きを予想すると考えると、経済学というのは将棋の研究と本質的には同じことのように感じます）。この「人間は常に自分が得をすることを目的に合理的に振る舞う」という仮定は、当然ながら実際とは多少ズレています（実際の人間は道徳心や利他心を多少は持っているし、いつも合理的に振る舞うわけではない。それなので、人間はいつもいつも合理的に振る舞うわけではない、として分析する行動経済学の考え方があってこれはこれですごく面白いです。行動経済学は、経済学に心理学を組み合わせて社会現象を分析する感じ）。

その「人間は自分が得をすることをする」という考え方ならば、「他人が不幸になることで自分が得をするならばそれをする」。しかし、それは例外的なことだと思う。多くの場合は他人が不幸になっても自分は得をしない。それなので、他人の不幸を願う理由はない。

そして、他人が幸福になると自分も幸福になるのであれば、他人の幸福を願う理由になる。

社会科学的に考えて、他人がパフォーマンスを上げると自分も得をするのではないか？ ということを私はなんとなく思っている。

他人が全部やってくれるならば自分は何もしないで毎日ゴロゴロと寝て暮らすことができる。それなので、他人がパフォーマンスを上げることは歓迎するべきことだと思う。

他人がパフォーマンスを上げることを邪魔をする理由はない。

「将棋」というゲームは「二人零和有限確定完全情報ゲーム」、ゼロサムゲームです。それなので、相手が得をするとその分だけ自分が損をするというゲームになっていますが、それは将棋というゲームが特殊なのであって、私たちが今いるこの社会というのは「他人が得をするとその分だけ自分が損をするというゲーム」ではないと思います（このことを学術的に論証してみたらどうなるのかというのは気になります。社会科学なのか、ゲーム情報理論なのかどのジャンルになるのかもよくわかりませんが）。

私は生まれてから現在まで一貫して勤労意欲がない人間です。
日本国憲法第二十七条一項で規定されている「勤労の義務」を果たす意志が皆無という、存在そのものが憲法違反な私です（ついでに言えば日本国憲法第三十条で規定されている「納税の義務」も可能ならば果たしたくないとも思っています。義務というものが嫌いな割に権利意識はやたら高いという、我ながら私は模範的な日本国民なのです）。
一体何をどうすれば「勤労意欲」などという気の利いたものが湧いて出てくるのかというのが不思議でしょうがないと私は心から思います（最も勤労意欲がある人からすれば「何でそんなに働くのが嫌なんですか？」と心から思っていることでしょうが）。
しかしながら、若さ故の過ちから7年9ヶ月の間、私は勤労と納税をしてしまいました。実際に勤労をしてみて、自分に勤労が向いていないことを再認識することができました。
そんなわけで、いい感じに無職になった私は、少し調査をしてみて市議会議員は基本的に一ヶ月に一回駅前のゴミ拾いをすることさえしていれば大体の日はのんびり過ごせそうだと思うようになりました。
そんなわけで、市議会議員だったら私のような勤労意欲がない人間でもできるだろうと思って、市議会議員になろうと2023年4月の統一地方選のときに行われた広島県呉市の市議会議員選挙に立候補してみたのですが、30万円の供託金を没収される結果になったのは我ながら遺憾に思います。
そんなわけで、市村さんは会社を解雇されていい感じの無職になって市議会議員選挙に立候補したら落選したどころか30万円の供託金を没収されたりしているけど今のところ人生を前向きに生きているので、あなたが今どういう状況かは知りませんが少々のことで人生を後ろ向きになる必要は全くないのです。

もし「私が市村さんの分まで二人分の勤労と納税をするので、市村さんは勤労と納税をしなくても良い

のですよ」と、言ってくださる方がいるのでしたら、私はその方に大いに感謝すると同時にその方を応援したいと心から思います。

他人が勤労をして納税することを妨げるべきではない。

余談ですが、市議選に立候補してみて、「選挙公報」という新聞みたいなのが発行されるのですが、それの広告代だと考えると30万円は十分にペイするなと思いました（供託金を没収された私が言うのもなんですが、市議選の場合は供託金没収ラインを超えるのは一般的にはそこまで難しくはないと思います）。自分が選挙に立候補するまでは東京都知事選挙に立候補されている独立系候補の方々について、「この人たちは一体何なんだ？」と思っていたのですが、市議選に立候補してみると、都知事選に立候補するのは300万円であれだけ広告が打てると考えるとむしろ「安い」と思うようになりました。都民向けの商品を持っていて都民向けに広告を打つことが有効な人は都知事選に立候補するのは全然ありだと思います。

> 囲碁・将棋のような競技の有益性。
「人生は他人との競争ではない」と私は思っています。
「『人生は他人との競争ではない』ということを前提にするから、将棋のような競技を安心して行うことができる」というのが私の認識です。個人的には別になんの役に立たなくたって良いじゃないか、とは思っていますが、幸か不幸か将棋のような競技はおそらく相当になにかの役に立っているのだろうという感じがします。
オリンピックとか、サッカーのワールドカップとか、あれは「代理戦争」なのでしょう。人が死なない戦争をしている。それをすると人が死ぬ戦争が発生する頻度を減らすことができるのだろうと私はなんとなく思っています。
個人的に私はサッカーが好きという以上にJリーグが好きなのですが、それは「地域間の代理戦争」をしているところが見ていて面白いというふうに思っています。バスケットボールのBリーグとかプロ野球とかもそういう「地域間の代理戦争」という側面があるような感じがします。私が好きなモータースポーツについて言えば、自動車のレースというのは自動車メーカー同士の代理戦争なんですね。

ゲームセンターを運営されていた方がTwitterで「ゲームセンターは年齢や性別や社会的地位に関係なく対等な立場で交流することができる稀有な場所」という投稿をしていて、私は「なるほどー」とすごく納得しました（現在はクレーンゲームが多いかもしれませんが、かつての格闘ゲームの筐体がおいてあった頃のゲームセンター）。囲碁とか将棋もそういう「年齢や性別や社会的地位に関係なく対等な立場で交流することができる稀有な場所」なのだろうなと。そういうのは非常に重要だし、非常に有益だと思います。

なんというか、戦争をしているときの「中立地帯」な感じがします。言い方を変えると、サードプレイスですね。個人的にそういう場所は非常に重要だと私は思っている人なのでそれは大事にしたいと思います。

学校とか会社とかでしんどくても、将棋の大会で上位に入賞したというようなことがあれば、それを支えにして人は生きていけると思うんですよ。そういうことを思えば、おそらくは囲碁や将棋のような競技はマイナスよりはプラスのほうが大きいのだろうと思う。

> ゲームの有益性について。

ゲーム雑誌「ファミ通」連載のデジタル技術のコラム、西川善司「ゲームのムズカシイ話」において「日本においてはゲームは子供が遊ぶもの、というふうな認識のされ方をすることが多いが、ゲームを『その時点におけるコンピュータ・サイエンス技術の集大成』と捉えている文化圏は意外と多い。特に最近はゲーム技術を産業応用することが行われるようになってきていて、学術界からも注目されている」というふうなことが書いてあった気がします（注）。たしか2023年の5月頃の号だった気がする。

今日の日付の新聞に書いてあるようなことを含めて「歴史」というのは「社会実験の記録」だと私は思っています。

そして「SFは未来についての思考実験」という言われ方をする事があるのですが、それを言えばSFに限らずアニメ・漫画・小説・ドラマ・映画、のようなフィクションの物語というのは、「思考実験の論文」だと私は捉えています（注）。

ゲームは現実世界のシミュレーターという風に私は捉えています。わかりやすい例で言えば、将棋と本質的には同じゲームである「大戦略」とか「ファミコンウォーズ」のようなゲームは戦争のシミュレー

ションとして行われていた兵棋演習を娯楽にしたもののようです（注）。2023年に公開された映画「グランツーリスモ」は、レースゲームのチャンピオンから本物の自動車のレーサーになったヤン・マーデンボロー選手の実話の物語です。

日本はもうダメなのではないか、という話をたまに聞くことがありますが、「日本がこれからの30年間でほぼ間違いなく世界ナンバーワンを取れる分野が一つだけあって、それが「マンガ・アニメ・ゲーム」のポップカルチャーの分野だ」という話を聞いたことがあります（どこで聞いたかは覚えていないのですが）。
個人的には、そう簡単にトヨタ・ホンダ・日産・スズキ・マツダ・SUBARUが自動車産業において敗北するだろうかと思いますが、仮に自動車産業でトヨタ・ホンダ・日産・スズキ・マツダ・SUBARUとかの日本メーカーが負けるにしても「マンガ・アニメ・ゲーム」がある限りこれからの30年間は日本は大丈夫だという風に私も思っています。

注：ゲーム技術の産業応用
https://xtech.nikkei.com/atcl/nxt/column/18/00001/07867/
> 物流や工場にデジタルツイン、AmazonやBMWが活用
> 現実空間の物体や環境を仮想空間に再現した「デジタルツイン」を利用して、最新の大規模施設の設計や構築、検証を行う動きが活発になってきた。先行するのは海外勢だ。米Amazon.com（アマゾン・ドット・コム）は物流施設、ドイツBMWは電気自動車（EV）工場を仮想空間で構築し、生産性向上やコスト削減を図る。

https://blogs.nvidia.co.jp/2021/05/10/nvidia-bmw-factory-future/
> NVIDIA と BMW、現実世界と仮想世界が融合された未来の工場を実演

注：「SFは未来についての思考実験」
松本健太郎「スマホアプリはなぜ無料？ 10代からのマーケティング入門」河出書房新社 P161-162
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

特にインターネットが浸透した今の社会では、デジタルでしか実現できない体験を作ることが、大きなお金を生み出すことに直結します。ですから大人たちは、デジタルでイノベーションを生み出すのに必死です。

ただ、先ほどお話ししたように、イノベーションが起こったあとの未来を、イノベーションが起こる前に具体的に想像するのは、なかなか難しいわけです。

こうした状況で、大人たちは一体どんなことをしていると思いますか？

その答えは、「SF思考」です。これまでは物語の一ジャンルであったSFのような考え方を元にして、新しい体験を作ろうとしています。

（中略）

といっても、素人がSFの未来の世界を描くのは簡単ではありません。そこで大人たちは、すでに世の中に出ているSF小説やSF映画、SFマンガも参考にしながら、頭を柔らかくして、まだ世の中にない新しい体験を生み出そうとしているのです。

こうした背景もあって、14歳の皆さんが、今からたくさんSFものを見たり読んだりしておくのは、SF思考を身につけるという意味でも、将来的に役立つんじゃないかと思います。

そう言われても「SFってなんだか難しそう」と敬遠してしまう人もいるかも知れませんね。一見とっつきにくそうなジャンルですが、先に上げた「ドラえもん」もSFです。アニメなら「機動戦士ガンダム」シリーズや「サマーウォーズ」など、親しみやすい作品もたくさんあります。

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

注：兵棋演習

> チェスター・ニミッツは第二次大戦時の対日戦は海軍大学校で学んだ兵棋演習の再演であり、予期できなかったのは神風攻撃のみだったと語っている。

https://ja.wikipedia.org/wiki/%E5%85%B5%E6%A3%8B%E6%BC%94%E7%BF%92

アピール文 240331追記

> 囲碁・将棋ソフトの競技会は実質的にLLM（大規模言語モデル）の性能を競っている。
大雑把に書きます。

「囲碁ソフトや将棋ソフトは、割と容易にLLM（大規模言語モデル）に転用が可能ではないか？」と現在の私は思っています（「容易」かどうかは主観の問題ですが）。棋譜の代わりにウィキペディアのテキストのような文章を大量に学習させたら、それはLLMになるのではないか。

　それなので、囲碁・将棋ソフトの競技会は実質的にLLMの性能を競っているのに近い状態だと思う。そしてそれをやっていたら最終的にはAGI（汎用人工知能）に到達するのではないか？　と私は思っているわけです。
　オープンソースプロジェクトとしてLLMやAGIの開発をしているのが囲碁・将棋ソフトの競技会、という風な捉え方をしてみたらどうかということです。
　OSについて、Windowsに対してのリナックスOSがオープンソースプロジェクトとして開発されたように、LLMやAGIにおけるオープンソースプロジェクトなのではないかという風に思う。
　それなので、これを続けていたら、OSにおけるLinuxのようなポジションに、LLMやAGIにおいて囲碁・将棋ソフトの競技会がなるのではないかと思っているわけです。

> 個人的に、知っておくと何かの役に立つかもしれないと思っていることを書いておきます。

　スマートフォンで、音声入力をすると、テキストを簡単に入力できます。

　LLM（大規模言語モデル）で、検索というか調べ物をすると良いです。
　マイクロソフトのCopilotとグーグルのGeminiを（無料で使えるために）私は使っているのですが、分からないこととか困ったなと思ったときにはCopilotとGeminiに自然言語で質問文を入力すると、割と有

益な答えが返ってくるのですごく良いと思います。

　例として「会社を解雇されて途方に暮れているけど、これからどうしたら良いですか？」「市議会議員選挙に立候補して当選するためには何をするとよいですか？」というふうな文章を入力すると割と有益な答えが返って来ます。LLMは種類によって返答の内容が違うので可能ならば複数の種類のLLMに同じ質問をして比較してみると良いです。

　個人的にはマイクロソフトのCopilotとbingのアプリをスマートフォンに入れておいて、いつでも使えるようにしておいたほうが良いと思っています。

・マイクロソフトのbing。ここからCopilotが使えます（ここからだとGPT-3.5じゃないかと思いますが、スマートフォンのアプリからだとGPT-4が使える）
https://www.bing.com/
・GoogleのGemini
https://gemini.google.com/app

　2045年にシンギュラリティが実現してもおかしくない状況になってきたと思っていますが、現在でもカメラからの映像をリアルタイムで処理して、話し出すようになればそれは殆ど人工知能と言ってよい状況だと思うのですが。

　元はゲームで2018年4月にアニメが放送された「シュタインズ・ゲート ゼロ」という作品のヒロインがそういう感じです。ヒロインのクリスは人工知能でスマートフォンの中にいて、スマートフォンのカメラから外界の映像を得て、主人公と会話をする。それが近い将来に実現すると思います。

　端的に言えば、男性の場合で言えば、「AI彼女」が近い将来に実現するということです。

　個人的に良いと思っているサービス。

・無料で音楽が聞ける
Spotify（スマートフォンで聞くとシャッフル再生しかできないけど、パソコンだとプレイリストの順番

通りに聞ける。無料アカウントだと広告がたまに入ります）

https://open.spotify.com/

・無料で利用できるインターネット・テレビ（知っている人も多いでしょうけど）

ABEMA

https://abema.tv/

TVER

https://tver.jp/

・無料で漫画が読める

ニコニコ静画

https://seiga.nicovideo.jp/manga/

コミックNEWTYPE

https://comic.webnewtype.com/

少年ジャンプ+

https://shonenjumpplus.com/

マガジン

https://pocket.shonenmagazine.com/

サンデー

https://www.sunday-webry.com/

カドコミ

https://comic-walker.com/

　スマートフォンの漫画アプリ。マンガ・アプリのランキングの上位10個のマンガアプリを全部入れてもよいです。大雑把に言えば、一つのマンガアプリにつき「（マンガを）一日に4話無料で読めます」というのならば、マンガアプリを10個スマートフォンとかタブレットに入れておくと、（マンガを）一日に40話無料で読めるようになります。

・国立国会図書館デジタルコレクション
https://dl.ndl.go.jp/
　著作権が切れた書籍を公開している。現在は書籍だけだけど、近い将来に雑誌も公開対象にすることを予定しているみたいです。
　個人的には、仏教についての重要な経典（漢訳された大蔵経を日本語に翻訳したもの）は恐らくは全部ここに無料で公開されていると思うのでそれがありがたいです。

> 大蔵経(だいぞうきょう)とは？ 意味や使い方 - コトバンク
仏教のすべての典籍を集録したもので，一切経・蔵経・三蔵聖経ともいう仏教の経（仏の教説集）・律（仏弟子の生活規範）・論（インド仏教学者による経の解釈）の３部（三蔵）を含む。 サンスクリット語系とパーリ語系に分かれ，前者には漢訳・チベット訳，後者にはビルマ訳・タイ訳などがある。 後者は『南伝大蔵経』として伝わる。

・無料ではないけど低価格で良いと思うサービス。

　新しかったり人気だったりする漫画を低価格で読む方法として、
レンタル・コミック
漫画喫茶・インターネットカフェ
　で読む。

ブックオフ・オンラインで注文した本とか雑誌とかをブックオフの店頭受け取りにすると一冊でも送料無料です。ブックオフ・オンラインは、書籍だけではなくて雑誌も扱っているので中古の雑誌が安くなっているのを近くのブックオフの店頭受取で送料無料で購入するのは結構良いと思います。あと、ブックオフはゲーム・ソフトとかDVD・ブルーレイディスクとかの映像ソフトとか、音楽CDも取り扱っています（個人的には10年くらい前に発売されたアニメのイベントとかライブのDVD・BDが500円くらいで売っていたりするのをたまに買います）。

・予測市場のポリマーケット

https://polymarket.com/

Wisdom of Crowds（群衆の叡智）・集合知による未来予測にして、「保険」を中央管理者なしで実現している（「過去のことはGoogleに聞け、未来のことはAugerに聞け」とインターネットで見たことがありますが、そのAuger（英語で「占い師」）の互換サービスです。）。

英語のサイトですがGoogleのクロム・ブラウザで開くとブラウザの標準機能で日本語に翻訳してくれます。

Wisdom of Crowds（群衆の叡智）については、ジェームズ・スロウィッキー著『「みんなの意見」は案外正しい』（角川書店）という本がすごく面白いです。

> 予測市場 (よそくしじょう、prediction market) とは、将来予測をするための先物市場である。

https://ja.wikipedia.org/wiki/%E4%BA%88%E6%B8%AC%E5%B8%82%E5%A0%B4