お前、 CSA 会員にならねーか？

詳細アピール文書

ザイオソフト コンピュータ将棋サークル 野田 久順

開発動機

小学生の頃、クラスメイトの家で遊んだスーパーファミコンの将棋ゲームで、コンピュータ将棋に興味を持った。その後、フリーの将棋ソフトをいくつか使ったあと、 2006 年に Bonanza に触れ、実装に興味を持った。 Bonanza のソースコードをダウンロードし読もうとしたが、当時の自分のプログラミング力では、まったく読み進めることができなかった。 2015 年に Apery がオープンソース化した際、ダウンロードして読んだところ、ある程度読むことができた。これをきっかけにコンピュータ将棋ソフトの開発に興味を持った。

開発過程

2015 年に第 3 回電王トーナメントへの出場を目指して開発を始めた。最初に Apery の高速化を行った後、クラスタリング、評価関数の強化学習、詰将棋ルーチンなどを開発し、2018 年には世界で初めて NNUE 評価関数を搭載したコンピュータ将棋ソフトを公開した。2021 年にはこの NNUE 評価関数をオープンソースのコンピュータチェスソフト Stockfish に移植した。その後、Stockfish チームが開発した nnue-pytorch を将棋用に改造し、NNUE 評価関数の学習を行った。そして、nnue-pytorch に対して数々の改良を加え、現在に至る。

独自の工夫

nnue-pytorch に対し、以下の変更を加えた。

- チェスに関する部分をやねうら王に差し替えた。
- やねうら王の学習器に実装されている Newbob 風の学習率スケジューラーを移植した。
- 評価値と勝率の変換に使用する Ponanza 定数を指定できるようにした。
- 各ステップ毎に学習率を調整するため、Warmup を独自に実装した。
- ネットワークパラメータを各エポック毎にクリッピングするようにした。

- 入玉時にボーナス点を加えるようにした。
- 学習率や lambda の値を、収束時に切り替えて再度学習を始められるようにした。

また、以下の工夫を行った。

- 最適化手法に Momentum SGD を使用した。
- Hao で生成した教師データと、水匠 5 で入玉将棋の局面から対局させて生成した教師データを混ぜて学習した。
- ネットワークアーキテクチャに halfkp_1024x2-8-64 を採用した。

実験結果

『Lí-VENGE』（tanuki- 第 2 回マイナビニュース杯電竜戦ハードウェア統一戦 決勝トーナメントバージョン）との自己対局を行い、レーティングを測定した。持ち時間は 5 分、1 手ごとに 2 秒加算、スレッド数 1、千日手の評価値は-2 とした。開始局面は、dlshogi の互角局面集から角換わりの割合が 10%になるように間引いたものからランダムに選択し、5000 局の対局を行った。結果、評価関数のみで R45.6、評価関数と探索部の組み合わせで R122.0 のレーティング向上が確認された。

追試可能か

上記の内容の詳細は、以下のブログに公開している。
nodchip のコンピューター将棋ブログ https://nodchip.hatenablog.com/
このブログの内容に従って追試を行うことで、結果の再現が可能だと考える。

謝辞

第 34 回世界コンピュータ将棋選手権が恙なく終了したこと、関係者の皆様に深く感謝申し上げます。また、「お前、CSA 会員にならねーか？」の開発・運用に関わった皆様に厚くお礼を申し上げます。最後に、本文書を作成するにあたり、校正に尽力してくださった ChatGPT さんに心から感謝申し上げます。

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

山岡忠夫
加納邦彦
大森悠平
2024/5/12

1　独自に工夫した点

1.1　自動定跡作成

直近の大会では、先手勝率が高い角換わりの先手定跡が出回ったため、先手角換わり定跡を搭載したチームに後手で角換わりを受けるとなかなか勝てないという状況が起きている。

先手角換わりが先手優勢であれば、初期局面から自動で定跡を作成することで、自動的に角換わりを避ける定跡ができるはずと考えて、1 年間かけて定跡を作成した。

定跡の作成方法は、手番側は定跡に登録された局面を α β 探索した時の最善手を指し、相手側は、dlshogi の訪問回数に応じた分布と NNUE 系ソフトの指し手からサンプリングするという方法である。定跡に未登録の局面に達したら、大会と同じくらいの思考時間で思考した結果を登録している。また、末端の局面は NNUE 系と評価が乖離していないかチェックを行い、乖離している場合は定跡を延長する。

これにより、指されやすい手は、100 手以上の深さまで掘ることができて、定跡を抜けた時点の評価は dlshogi と NNUE 系で一致する局面となる。

この手法で定跡作成を続けたところ、先手はほとんどのケースで最善手を変更する必要がないが、後手は有望と予測された手順がすべて先手に潰されてしまうため、手を変える必要があり、あまり深くは掘られないという状況になった。評価値では互角と判断していても、100 手目付近まで調べないと分からないということが多く、後手番で精度の高い定跡を作るのは現実的に困難である。深くまで調べた手順も先手有利の結論になると回避してしまうため、大会では有効かもしれない手順を自ら回避してしまうという課題が発生した。

結局、後手番は角換わりを受けて初期局面以上の差を広げないか、多少不利になっても角換わりを避けて相手の定跡がないことを期待して時間の有利を築く以外の有効な戦略は発見できなかった。

対して先手の定跡は、比較的効果があり、2 次予選と決勝合わせて、定跡を抜けた時点の評価値が 500 点以上が 4 局、300 以上が 6 局あった。また、定跡の手数は先手で平均 66.5 手、後手で平均 33.75 手であった。

1.2　モデル

前大会と同様の 30 ブロック 384 フィルタのモデルを使用した。しかし、2 次予選で終盤の読み抜けで負けることがあったため、決勝では後手のみ終盤読み抜けが比較的少ない 20 ブロック 512 フィルタのモデルを使用した。

2　開発動機

2016 年 3 月に行われた AlphaGo とイ・セドル九段の対局で、AlphaGo が従来と異なるディープラーニングという手法で勝利したことに衝撃を受けた。囲碁の盤面を画像として入力して指し手を予測するという仕組みを理解したいと考えて、論文を読み AlphaGo のクローンの実装を行った。プロの棋譜を学習して対局するところまで実装できたが、囲碁 AI は、すでにアメリカ、中国の巨大企業も取り組んでいたため、個人で囲碁 AI を開発するモチベーションは続かなかった。将棋 AI では、まだディープラーニングの手法が実験レベルでしか試されていなかったため、自分が開発する意義があると考えて取り組むことにした。

3　開発過程

2017 年からはじめは教師あり学習でのモデル学習から始め、PV-MCTS の実装をしたことで、GPSFish に勝てることできたため、ディープラーニングの手法が将棋 AI でも有効であることに確信が持てた。その後 AlphaZero が発表され、将棋でもトップレベルの強さにできることが報告された。AlphaZero は、膨大な計算リソースで実現されていたため、工夫を行うことで個人レベルの計算リソースでも強くできることを目標にして改良を続けた。

今大会では、HROZ の将棋 AI 開発者のチームで参加した。社内の計算リソースを使って強化学習を行った。チームメンバと共に、戦型・定跡といった大会に向けた作戦を準備した。

4　実験結果

今大会では主に、定跡作成に取り組んだため、モデルのレーティング向上はない。正確に測定していないが、定跡を使用した場合、定跡を使用していないバージョンと比べて、先手は 100%近い勝率、後手は 50%程度の勝率となる。

5　追試可能か

dlshogi の探索部、学習部のソースコードは GitHub[1]で公開している。学習方法については、ブログ記事[2]や書籍「強い将棋ソフトの創りかた[3]」で解説している。dlshogi の学習に使用しているデータも書籍に付属している。それらを元に、ある程度の追記を行うことは可能である。また、定跡作成は計算リソースをかけているため、同様の規模での追試は難しいと考える。

---

1 https://github.com/TadaoYamaoka/DeepLearningShogi
2 https://tadaoyamaoka.hatenablog.com/
3 https://honto.jp/netstore/pd-book_31311287.html

東横将棋　WCSC34 大会後詳細アピール文書

東横コンピュータ将棋部

概要及び開発動機

WCSC34 版の東横将棋は、最新開発番やねうら王探索部＋マメットブンブク型評価関数（halfkp_1024x2-8-32）のコンピュータ将棋エンジンです。ディープラーニング系以外の NNUE 評価関数系としては現状では主流の構成と言えるでしょう。一般家庭で検討用途に使用される際には、標準 NNUE 型評価関数またはマメットブンブク型評価関数が選択されると思われます。
この構成と定跡の強化で現在の主流ソフトにどこまで対抗できるかというのが主な開発動機です。
WCSC34 版の東横将棋は、全てにおいて第 4 回電竜戦本戦で使用したマメットブンブク型評価関数をそのまま使用しています。探索部については同じく全てにおいて最新の開発版やねうら王をそのまま使用しています。

使用ハードウェア

選手権すべてにおいて、AMD Ryzen Threadripper 3990X を使用しました。このハードウェアはかつて「高級スリッパ」とも称され、最強の家庭用パソコンとして名を馳せましたが、発売から 4 年が経過し、既に完全に時代遅れのものとなっています。
WCSC34 においてはクラウド利用はそもそも想定しておらず、上位チームのモンスターマシンにどこまで対抗できるかに興味があり、全対局において Ryzen Threadripper 3990X を使用することに決めておりました。

評価関数について

昨年からマメットブンブク型評価関数の強化に注力しております。しかし、マメットブンブク型評価関数は入玉に難点があることが指摘されており、現状では一定の評価値で標準 NNUE 型にリレーするのが最善かと思われます。

定跡について

角換わりと後手番相掛かりを重視した定跡の作成に力を入れました。昨年度の定跡をベースに floodgate で数え切れないほどの対戦を積み重ねて定跡を作成しました。この過程で、ディープラーニング系のライセンスに抵触するソフトの使用は避け、s-book_blackフリーおよびライセンスフリーの原則を遵守しました。
昨年度の定跡をベースに tanuki-wcsc33 定跡を参考に、さらに floodgate の棋譜を参考に手動で定跡を作成しました。しかし、結果的に大会までに納得のいく定跡が完成したとは言えませんでした。

評価関数の実験結果等

大量の自己対戦の結果、電竜戦で使用した評価関数から多くの強化は望めないと判断いたしました。

追試の可否について

ハードウェア及び使用したソフトウェアは全て手元にありますので、いつでも追試可能です。

謝辞

大会中は本当に楽しい経験をさせていただきました。運営の皆様、スポンサー様、他の参加者の方々には深く御礼申し上げます。
私は開発者ではありませんが、現地での開発者の方々との交流は刺激的で、充実した時間を過ごすことができました。次回の大会では、今回の経験を生かし、さらに上位入賞を目指したいと強く思っています。
大会に参加することは私にとって非常に貴重な経験であり、皆様の温かいサポートや応援があったからこそ、3 位入賞という素晴らしい結果を得ることができました。心からの感謝を込めて、改めてお礼申し上げます。コンピュータ将棋界でのさらなる成果や活躍をお届けできるよう、精一杯努力し続けます。引き続き皆様のご支援と応援をよろしくお願い申し上げます。

koron 詳細アピール文

2024/5/18 野田煌介

・開発動機

小学生の頃、将棋と Python に触れていた自分は将棋 AI に興味を持った。その後、やねうらお氏のブログ[1]を見たことがきっかけで、当時の KPTT 雑巾絞りなどで遊ぶのが新たな趣味となった。それがきっかけとなり、実際に世界コンピューター将棋選手権に向け、コンピューター将棋の開発を行うこととなった。

参加するにあたって、多くの将棋 AI では単一の AI を用いて対局を行っているが、これらはどうしても序盤と終盤のトレードオフの問題が付きまとう課題点があると考えた。koron はリレー式合議を行うことによって、序盤・中盤・終盤全てに対して最適な状態で対局できる AI の開発を目的として開発を行っている。

・独自に工夫した点

以前は手数で評価関数を切り替える手法を採用していたが、将棋は双方の戦法次第で 40 手～60 手程度で終盤になることもあれば、100 手以上指しても形勢が変わらないこともあるなど、一概に手数で対局の進み具合を決めることは困難であり、評価関数を切り替える指標としては適切できなかった。

そのため、今回の koron は評価値を対局の進み具合を示す値であるとみなして、それに応じて評価関数を切り替えることにした。これにより、以前よりも適切な場面での評価関数の切り替えが行えるようになり、棋力の向上につながったと考えている。

評価関数では序盤の互角時には halfkp-1024x2-8-16、中盤～終盤には halfkp-256x2-16-32 を採用した。これにより、評価値によるリレー対局を行うことで、自分が優勢な場合は読み抜けによる頓死を防ぎ、自分が不利な場合は相手の読み抜けを咎めやすくなる効果が期待できる。また勝勢・不利のどちらでも入玉将棋が強くなるメリットがあると考えた。

現状、切り替えの最適なパラメーターの探索が十分とは言えないので、最適な値を求めて引き続き研究していきたい。

・開発過程

電竜戦バージョンの koron へ追加学習を行った。また序盤で使用する halfkp-1024x2-8-16 では、与える教師データを序盤のみに絞り序盤の精度向上を狙った。

リレープログラムの開発では、評価関数の学習に時間がかかったため、切り替えの最適なパラメーターを十分に探索することはできなかった。そのため、本番では評価値 1000 を上回った場合、もしくは評価値-300 を下回った場合に評価関数を切り替えることとした。

一定の基準値を設け、評価値が再び互角になった場合、評価関数を元（halfkp-1024x2-8-16）に切り替える機能も開発したが、本番では採用しなかった。これは切り替え時に多少の時間的損失が生じてしまい、それが複数回ともなると棋力に少なからず影響が出ると考えたことによる。

---

[1] https://yaneuraou.yaneu.com/

・実験結果

条件

開始局面：Floodgate32-80.sfen[2]
持ち時間：5 分 フィッシャー2 秒
Ryzen 7950x(24T 使用) Hash 4GB
320 手超えは持将棋
評価値 1000 もしくは-300 を超えた時点で切り替え

結果

Li と Hao のリレー vs Li
194-71-153 (55.91% R41.24)
koron vs Li
231-86-167 (58.04% R56.36)

考察

持ち時間を多くした分、対局数が少なくなってしまったため追加学習によるレート向上効果は不明だが評価関数の切り替えによって有意にレートが向上したと考えられる。

・追試可能か

学習などにはランダム性があるため完全に同じ結果を得ることは難しいが、ある程度の追試は可能と考えられる。

・使用ライブラリ

やねうら王[3]

nodchip 氏が公開している教師データセット[4]

たややん氏が公開している教師データセット[5]

・謝辞

koron は上記のライブラリを学習時の教師局面や探索部として利用した。これらのライブラリが存在しなければ、入賞はおろか世界コンピューター将棋選手権への参加も叶わなかっただろう。

また、今回も CSA 関係者やスポンサーの皆様方、その他多くの方々によって、世界コンピューター将棋選手権が無事に終了したことに感謝を申し上げたい。

[2] https://gist.github.com/TadaoYamaoka/41a09638fcf07145101e1c716978fb76
[3] やねうら王最新版：https://github.com/sponsors/yaneurao/
[4] https://huggingface.co/datasets/nodchip/shogi_hao_depth9
[5] https://drive.google.com/file/d/1VyP4MX_AuQhvy8sesymgPVf9sUnQoGPI/view

「技巧」詳細アピール文書

2024 年 5 月 19 日

出村　洋介

1　独自に工夫した点

1.1　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を行いました．Gumbel AlphaZero のアルゴリズムを用いたことにより，少ないシミュレーション数で計算時間を抑えた効率的な強化学習が可能となりました．

今年の技巧では，乱数で初期化されたネットワークから約 5000 万対局の自己対局で学習を行いましたが，学習時間は 1 週間程度に抑えられています（学習には NVIDIA 社の GeForce RTX 2080 Ti〜RTX 4090 クラスの GPU を 20 台程度利用）．

1.2　強化学習時の初期局面を多様化

自己対局時に様々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いました．具体的には，平手初期局面から 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).「チェス 960」にならい「将棋 81 万」と呼んでいます）を初期局面として強化学習を行いました [3].

(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)

1.3　ネットワークの改良

推論に用いるネットワークは，Gumbel AlphaZero 論文 [2]で用いられた bottleneck blocks をベースに独自の改良を行いました．（この改良については別の機会に発表させていただければと考えております．）

2　開発動機

深層強化学習を用いた将棋 AI の作成には大規模な計算リソースが必要という状況を少しでも改善できればと考え，今年の技巧では，各種効率化により計算資源を抑えつつ，既存の棋譜を使わずにゼロから強化学習を行うことを目指して開発を行いました．また，既存の棋譜や将棋用のライブラリを利用しなかったのは，コンピュータの自己対局のみで学習するとどのような将棋を指すのか純粋に見てみたいという気持ちもありました．

3　開発過程

2016 年の選手権後に C++版の技巧を公開していましたが, 新たなプログラミング言語に挑戦したいと考え，心機一転 Rust と Python を使ってフルスクラッチで書き直しました.

今回の開発では，計算コストを下げた実験を多数行い開発サイクルを速くして各種改良点を地道に探しました．普段の実験では，棋譜数を 1000 万局程度とし，さらにネットワークのチャネル数やブロック数を減らすなどして，教師あり学習ならば 1 GPU で 1 日程度, 強化学習なら 10 GPU 程度で 1 日程度で 1 回の実験が終わる計算量に抑えていました.

4　実験結果

選手権に向けては，学習棋譜数は 5000 万対局程度まで増やし，ほぼ飽和状態となるまで自己対局による強化学習を行いました．自己対局のシミュレーション数は，概ね 1000 万対局まで 1 手 32 シミュレーション，1000 万〜4000 万対局まで 64 シミュレーション，4000 万対局以降は 128 シミュレーションと段階的に増加させています.（今回は学習途中に手動で学習条件を調整したため実際の対局数には若干誤差が出ています.）

今年の決勝リーグで用いたネットワークと探索部の実験結果は以下のとおりです.

技巧（100 ノード）対 水匠 5（40,000 ノード）[1]	+202 Elo
技巧（800 ノード）対 水匠 5（250,000 ノード）[1]	+127 Elo
初期局面の探索速度（15 秒間の平均 NPS）	100,153 局面/秒（RTX 4090） 58,493 局面/秒（RTX 3090）

5　追試可能か

強化学習ではランダムな要素が複数あるため（ネットワークの重み初期化，自己対局時の開始局面の選択，自己対局時の指し手など），完全な再現は難しいですが，以下の参考文献などを参考に概ね追試可能と考えています．ネットワーク関係の改良は後日別の機会に改めて発表させていただければ幸いです.

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]　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.

[1] 対局は各 3000 局．MCTS 探索のバッチサイズは 1．水匠 5 との対局実験では，公開されている dlshogi の互角局面集（https://tadaoyamaoka.hatenablog.com/entry/2022/12/31/114258）を 24 手目まで利用.

やねうら王 featuring Ryfamate
WCSC34 大会後アピール文

チーム やねこま王

やねうらお [*], 駒の書体 (Komafont) [†], よみぃ [‡]

1 開発動機

筆者 (Komafont) は、これまで NNUE 型評価関数[*1] と Deep Learning (DL) 系評価関数を組み合わせた合議プログラム Ryfamate として単独出場していたが、DL の発展に伴い大規模な計算資源の必要性が増していることや、持病のため現地参加を求める新ルールへの対応が困難であったことから、単独出場を断念しチームで参加することとなった。そこで、Ryfamate の合議に用いる独自開発した評価関数 Ryfamate Cross Network (RyfcNet) [*2] のうち、序中盤向けに学習を進めていた 30 ブロック Attention [*3]付きのモデルをチームに提供した。これに、やねうらお氏が作成したペタショック定跡[*4] および探索エンジン ふかうら王[*5] を組み合わせ、よみぃ氏が担当したインフラのもとでさらに学習を進めたうえで出場した。

2 開発過程

近年、生成 AI などの分野で目覚ましい成果をあげる Transformer であるが、そのまま将棋用の評価関数とした場合は推論速度が問題となる。そこで、RyfcNet では、現在将棋の評価関数として活躍している ResNet [*6] の一部ブロックを、Transfomer の中核的要素である Attention を用いたブロック (A-Block) に置き換え、従来のブロックと同程度の推論速度になるよう調整した。これにより、12 ブロックの比較実験において、Attention を用いない RyfcNet と比べ、わずかながら精度の向上を確認した。加えて、Attention を用いた RyfcNet のほうが、チャンネル数を増加させたときに精度が大きく向上することなどから、序中盤向けの大型モデルに Attention を採用した。

さらに、本年は Positional Encoding (PE) を工夫し、Relative Position Representations [*7] などを参考に、Attention 中の各 Softmax 関数への入力値に対して学習可能パラメータを加算した。従来の ResNet では、角の進む方向に離れたマスの認識に複数のブロックが必要であったが、この工夫により、これを 1 つのブロックで認識できるような head が学習の結果として獲得できた。[*8]

---

[*] やねうらお https://yaneuraou.yaneu.com/

[†] 駒の書体 (Komafont) https://x.com/komafont

[‡] よみぃ https://www.youtube.com/@WoooRen1006

[*1] 那須悠. 高速に差分計算可能なニューラルネットワーク型将棋評価関数. 2018.

[*2] https://www.apply.computer-shogi.org/wcsc33/appeal/Ryfamate/appeal_ryfamate_20230421.pdf

[*3] Attention : 本文では Multi-Head Self-Attention のことを指す。

[*4] https://yaneuraou.yaneu.com/2024/01/14/the-era-of-large-scale-book-in-shogi-ai/

[*5] https://github.com/yaneurao/YaneuraOu

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

[*7] Shaw et al. Self-attention with relative position representations. 2018.

[*8] https://x.com/komafont/status/1787014338286653808

なお、Attention を用いたモデルでは、これを用いないモデルに比べ、早い段階で過学習の兆候が現れた。これは、Attention が Convolution に比べて帰納バイアスが弱く、同程度のモデルサイズでもより多くの教師が必要となるためと思われる。この点については、チームメイトから提供された計算資源によって大いに助けられた。また、今年は、たややん氏によって公開された NNUE 1000 万ノードの教師[*9] が特に終盤の Value の精度向上に貢献した。

3 実験結果

本年使用した Attention つきの RyfcNet(Ryfc30_WCSC34)、昨年合議のメインとして使用した Attention なしの RyfcNet(Ryfc20_WCSC33)、お前、 CSA 会員にならねーか？[*10] (tanuki-WCSC34) の三者を総当り式に対局させた。計測には、Ryzen 5950X + GeForce RTX 4070 ×1 のマシンを使用した。Ryfc30_WCSC34 と Ryfc20_WCSC33 は 1 手 4 秒とし、これらとほぼ互角になるよう tanuki-WCSC34 の秒数を設定した。

結果、Ryfc30_WCSC34 は、Ryfc20_WCSC33 に対して R+28.4 (±33.9, n=792) であった。[*11] 事前の実験により、長時間・高ノードの対局であれば優位性が拡大することは確認しているが、当初期待したほどの差ではなかった。Attention については、さらなる改良が必要であると考える。

4 追試可能性

すでに RyfcNet の基礎技術は再現可能な内容を公開しており、昨年公開して以来、複数の方より実装・実験したとの報告をいただいた。なお、C-Layer については、形状が (C,H,W) の入力値に対して、(C,H,1),(C,1,W),(1,H,W) といった形状の出力値となる。このため、形状が (C,H,W) のテンソルを得るためには、これらのうち複数の形状の出力値を加算することが有効である。[*12] また、A-Block については、PyTorch の標準的な方法で実装可能であるが、FP16 で学習や推論を行うとオーバーフローによって失敗することがある。このため、2024 年 5 月 5 日時点では、dlshogi の学習器・探索エンジンなどに、Attention 対応の修正を施す必要がある。当チームでは、オーバーフローの原因となる演算箇所を BF16 または FP32 で演算させることで対応した。

5 おわりに

大会主催者、スポンサー、関係者、そして開発者と開発者を応援してくださる多くの方々に、厚く御礼申し上げます。

[*9] https://x.com/tayayan_ts/status/1767133487847645457

[*10] https://lessertanuki.booth.pm/items/5713519

[*11] 実験の条件や結果の詳細は X(Twitter) で公開した。 https://x.com/komafont/status/1792004038596481499

[*12] 具体的な応用例として、2 種類の C-Layer の出力値を加算するネットワーク図を公開している。
また、C-Layer の出力値と S-Layer の出力値を加算することも有効と思われる。
https://x.com/komafont/status/1787011012945928504

十六式いろは煌（きらめき）　詳細アピール文
2024-05-19

開発動機
末吉が十六式いろはシリーズを作る始まりのきっかけは2005年頃、書籍「コンピュータ将棋のアルゴリズム」（著者、池泰弘）を購入したことでした。個人でも将棋AIを開発できる可能性を感じ、胸が踊ったことを今でも覚えています。
しかし、開発するためのまとまった時間が作れず10年以上経った2016年に、自身の将棋のスキルをアマチュア初段のレベルまで引き上げるための補助ツールを目指しつつ、またプログラミングの勉強を再開する目的で、開発を開始しました。

そして、2021年には新たな展開がありました。日本工学院専門学校の学生達と共同で、この将棋AIの開発をプロジェクトとして進めることになったのです。学生達が学んできた機械学習の知識を活かし、プログラムを作り機械学習の成果を実感してもらうという教育的なことが新たな目的となりました。またこのプロジェクトは、これから機械学習を学ぶ人の為になるように、学習データやその成果を公開していくことも行っています。

開発過程
末吉が一人で開発していた2021年までは、書籍「コンピュータ将棋のアルゴリズム」を参考にして、C言語やLuaによるフロムスクラッチで開発していました。
学生達と開発し始めた2021年からは、書籍「強い将棋ソフトの創りかた」（著者、山岡忠夫、加納邦彦）を参考にして、ディープラーニングによる将棋AIの開発にシフトしました。まずは、学生と共に書籍にある将棋AI「python-dlshogi2」の仕組みを学ぶところから始めました。その成果物として2022年に、ディープラーニング将棋AI制作ソフト「将スタ -将棋ソフトスタジオ-」を作り公開しました。その間のWCSC32では、dlshogiのネットワーク部分を変更することにし、デフォルトのResNetのブロック数を減らし、学習時間が少ない軽量なものに変えてみました。

WCSC33では、SENetに変更してからWideResNetを参考にチャネル数を増やして、ブロック数を減らして軽量化を試みたところ、想定以上の成果が出ました。そして、2023年始めにChatGPTが公開されたことによって、軽量なネットワークのGhostNetに変更することができました。

その一方で、人による棋譜を用いずにAIが生成した棋譜のみで将棋AIを開発しようと思い、やねうら王の標準NNUE型の評価関数を一から作成し始めました。ある程度の棋力の評価関数が作成できれば、蒸留による軽量なNNUEの評価関数を作成しようと考えています。
ちなみに2024年現在、dlshogi系の将棋AIよりもNNUEによる将棋AIの方が、教師局面の生成は速いこともあって、しばらくはNNUEの評価関数を育てることも行っていく予定です。

実験結果
今回のWCSC34では、やねうら王の標準NNUEで戦い抜いたので、その評価関数の育て方の実験結果を記そうと思います。

棋力測定の方法
- 対局数：500～3000
- 開始手数：24
- 互角局面集：
  互角局面集作成スクリプト - TadaoYamaokaの開発日記
  https://tadaoyamaoka.hatenablog.com/entry/2021/09/20/222018
  上記で公開されている互角局面集。
- 定跡：なし
- 思考ノード数：300000（平手局面 depth 13くらい）

WCSC33まで
WCSC33の十六式いろは煌（きらめき）のアピール文のP8～11を参照してください。

https://www.apply.computer-shogi.org/wcsc33/appeal/16_Shiki_Iroha_KIRAMEKI/2023-05-12_16-168kirameki_wcsc33.pdf

学習20億局面くらいから50億局面
- パラメータを複数変えながら実験を行った。
- 学習率eta0.01、lambda0.3など、設定が同じ値でも生成するたびに棋力がR+-30くらいで変更する。
- 学習局面が増えるにつれ、etaを下げていくと効果がある。（だんだんとfitしていっているため？）
- 同じ局面を2週学習しても、棋力が向上する。だが、3周目はほとんど効果がない。
- 1周目eta0.1、2周目eta0.01など、重ねて学習するときは学習率を下げると効果的。
- lambdaは「0.1～0.5」（勝率項は控えめに参照）が効果的で、特に「0.1」は良い結果が出やすい。
- mirror_percentageは、50から10に変更すると効果があった。
- newbob_decayは、0.2から0.1に変更すると効果があった。
- 評価値の上限を1200としたのは、中盤力を短時間の学習で上げる狙いがあったが検証できておらず。
- 30億局面以降では、評価値の上限は1200よりも32000の方が効果があった。（30億局面以下は未検証）

学習80億局面以降
- 学習率eta0.000001くらいにしないと、棋力に変化が起きなくなってくる。
- 教師局面の深さを徐々に深くしていっても、まだ効果がありそう。

追試可能か
可能です。

棋風
対局の解説等を聞いている限り、かなり攻めっ気が強いようです。
2次予選時は定跡を積んでいないのですが、矢倉っぽいのが好きそうな感じがしました。

以上です。

Polonaise アピール文書

谷合廣紀

2024 年 5 月 15 日

1　開発動機

近年の将棋 AI 開発において、dlshogi をはじめとした AlphaZero 系のアプローチは非常に強力な手法として注目されています。しかしながら、将棋の盤面を画像として捉え、ResNet などの畳み込みニューラルネットワーク（CNN）を用いる手法には疑問を感じていました。

CNN の特徴の一つに位置不変性があります。これは、画像中の特徴がどの位置にあっても同じ特徴量に変換される特性です。囲碁においてはこの特性が有効ですが、将棋においては必ずしも有効とはいえません。例えば、美濃囲いは２八玉・３八銀・４九金という特定の配置だからこそ囲いとしての固さを発揮します。これが２七玉・３七銀・４八金や３八玉・４八銀・５九金とタテやヨコにずれた配置では、同じ意味を持たなくなります。さらに、将棋には飛・角・香といった飛び駒があり、これらは最大で８マス離れた位置に利きを作ることができます。これを特徴量として捉えるためには、3x3 の畳み込み層では 3 層必要になります。

このように将棋において CNN は必ずしも最適なモデルとは言えません。そのため自然言語処理で大きな成果を上げている Transformer を将棋 AI に応用することに着目しました。将棋の盤面を文字列に変換し、自然言語処理モデルで学習・推論を行うことで、従来の手法とは異なる新しいアプローチを模索しました。これが、Polonaise の開発動機です。

2　開発過程

最初の実装は、YouTube チャンネル『予備校のノリで学ぶ「大学の数学・物理」』の動画に出演した際に作成した BERT-MCTS でした[*1]。動画出演のオファーを受けた際に、インパクトのあるコンテンツを提供したいという思いと、自然言語処理モデルを将棋 AI に応用するというアイデアを持っていたことから、これを良い機会と捉えて開発に着手しました。

BERT-MCTS は、実装の一部に誤りがあったり、モデルの学習が不十分だったため、あまり強力な将棋 AI にはなりませんでした。しかし、このアプローチ自体には大きな可能性を感じました。このため、さらにこのアプローチを成長させ、第 32 回世界コンピュータ将棋選手権に出場することを決めました。探索部には、ふかうら王のアルゴリズムをベースに実装しました。

第 32 回から大きなアップデートは探索部にはありませんが、主にモデルの改良を重ねてきました。そして今回の第 34 回世界コンピュータ将棋選手権では、BERT-large という大規模なモデルに挑戦しました。これ

[*1] プロ棋士自作の将棋 AI と戦ったら色々とヤバかった: https://www.youtube.com/watch?v=2Vl6Ao4GaSQ
BERT-MCTS のコード: https://github.com/nyoki-mtl/bert-mcts-youtube

が可能になったのは、多くの開発者が無料で良質な教師データを公開してくださったおかげです。また、モデルの学習には Google の TPU v3-8 を利用しています。

3　独自の工夫

3.1　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。

3.2　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。

3.3　PVM ネットワーク

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。(実装が大会直前だったため、計測データはありません。)

4　追試可能か

公開データによる教師あり学習しか行っていないため、同じ入力エンコードとモデル構造を用いれば追試可能です。