CGP アピール文章

2021/3/28
大熊　三晴

主な特徴

・無駄に一から作成
・非ビットボード型
・無駄に高 NPS を目指してるけど最近この部分はさぼり気味
・局面構造体に各マスへの利きの状態を保持
・局面構造体に評価関数の演算途中結果のうち変化の頻度が少ないものを中心に保持
・評価関数も自力で学習
・AVX-512 命令をはじめとした拡張命令を活用
（ただし今回、非対応 CPU のマシンを使用する可能性があるため、
無効化する可能性あり）
・大会までにはアピール文書を書き直すくらいに開発が進んでて欲しいです。
・一般に流布している定跡データや、一般に流布している「局面と評価値のセット」、読み筋等は使用しておりません。無駄なこだわりだとは思いますが。

・１から作成

強さをあまり考えずに高 NPS を目指して自作したプログラムをベースとしております。並列化手法は現在は LazySMP を微修正したものです。

・非ビットボード型，利き等を保持

非ビットボードだとビットボードに比べ遅くなる処理もありますが、複雑な情報を持てることにより速く処理できる可能性もあります。ビットボードに比べ遅い処理をうまく避けるために利きを保持したり、局面構造体の配置をビット位置を含めて工夫しております。AVX-512 でかなりの並列化が出来そうですがまだ Core i9 7940X で Ryzen9 3950X を上回るほどではありません。(1 スレッドあたりでは AVX-512 有効の Core i9 7940X のほうが上)

また利きの保持以外にも、演算途中のデータを保持することによりメモリアクセス待ち時に演算を回す事により高速化を狙っております。

・評価関数

現在は手番付き KPP です。ただし持ち駒周りの評価を一般的な 3 駒関係より拡張しております。

・SIMD等の活用

高速化のためSIMDを活用しております。SIMDは現在評価値の算出、オーダリング、構造体のコピーが主な使用箇所です。ただし構造体のコピーに関してはx64のストリング命令が高速化されてきているので変更するかもしれません。

第31回
世界コンピュータ将棋選手権
なのはアピール文書

V0 2021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。

- 本来の**なのは**の強さを思うと情けないくらい弱く「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 東京都在住
- とあるメーカーに勤務
- 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！、とある科学の超電磁砲、ラブライブ！、五等分の花嫁
- 将棋ウォーズ 1級
- 囲碁は日本棋院 初段(アマチュア)
- ！職業プログラマ∩！研究者　　∩！東大∩！学生

■開発環境

- こんなPCで開発していますｗ

- 出場PC(予定)

  CPU：AMD R7 2700X

  メモリ：32GB

  OS：Windows10Pro

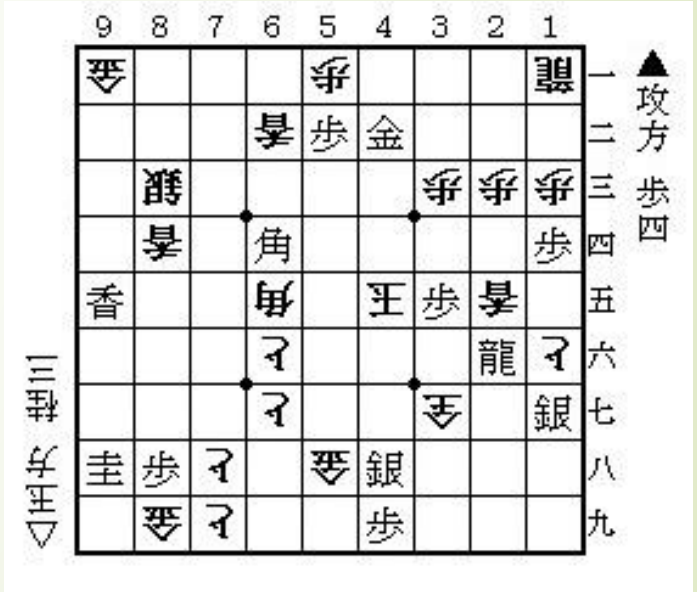
■なのはの構成

- Visual Studio Community 2019(C++)にて開発
- 手生成では歩、角、飛の不成も生成
- 探索はStockfish使用
  - 評価ベクトル縮小、詰めルーチン(df-pn)削除したものを、**なのは**miniとして公開
- 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
- 評価ベクトルは一般に流布された棋譜データを使用して学習する予定
  - 周回遅れの3駒関係による評価

…と、どこにでもあるような平凡な構成

■**なのは**の特徴は？

- 強力な詰めルーチン搭載(**なのは**詰めとして公開)
- **なのは**詰めは江戸時代の名作 611手詰めの「寿」を解く
- 常に詰みを狙い、**一発逆転するポテンシャル**を秘めています！



■なのはの新たな特徴(予定)は？

- 特になし

■意気込み

- DLは無理なので、ゆるく参加したいと思います。
- ０次予選突破が目標です。
- あわよくば予選で勝ち越ししたい

(今後の課題)

- 周回遅れですが、NNUE形式に対応

■使用ライブラリについて

- なのはmini　自作です。

■使用データについて

- 自前で用意するのはリソース不足なので、一般に流布されたデータを使用予定です。

■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

第 31 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

ねね将棋(NEural NEtwork Shogi)は、深層学習(Deep Learning)を用いた評価関数により思考する将棋ソフトです。従来の 3 駒関係+α β 探索に代わるアーキテクチャで強くすることを目指しています。

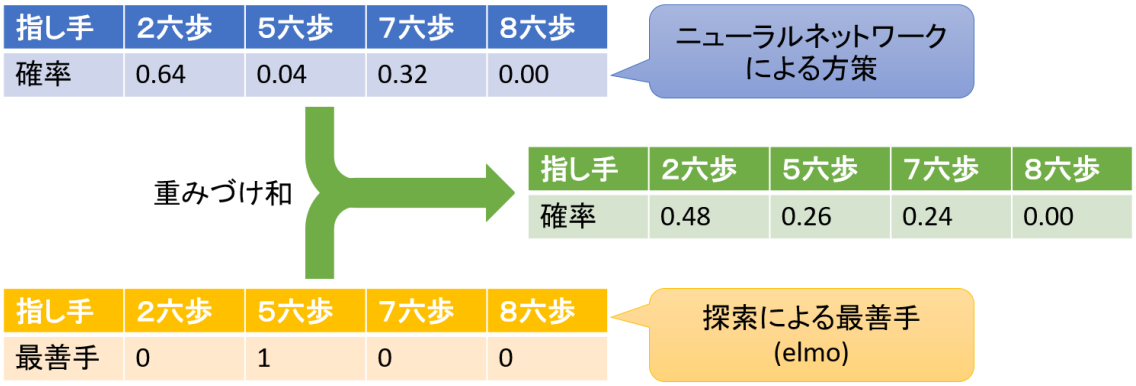
昨年との大きな変更点

昨年まで独自の探索部・評価関数を用いていましたが、dlshogi ライブラリに全面的に依存するよう方針を変更しました。主な理由は、私が強化学習の計算資源を捻出できない点、改良箇所を第三者が利用しやすくなる点です。

使用ライブラリ

- dlshogi [1]
- GCT 評価関数 第 1 回世界将棋 AI 電竜戦版 (2020 年) [2]
- やねうら王 [3]
- elmo 評価関数 世界コンピュータ将棋選手権 29 版(2019 年、NNUE 型) [4]

ポイント



dlshogi は MCTS 型の探索部を搭載しており、深層学習を用いた方策関数により探索する手を絞ります。この手法の欠点として、有望な手に対する方策関数の出力値（確率）が小さい場合、その手を指した後の局面が全く評価されず大きな誤りにつながる場合があります。これを補償する手段として、深層学習による評価と同時に全幅探索ベースの従来型の将棋 AI（やねうら王+elmo 評価関数 2019 年版）を用いて 1,000 ノード分の探索を行い、その最善手を方策関数の結果とブレンドする手法を開発しました。この手法は、広義の評価関数アンサンブルとみることができます。棋力は、無改造の dlshogi と比較してレート+70 程度の

向上を得られました。一方、提案の改造を施した dlshogi の elmo に対する勝率は、無改造の dlshogi の elmo に対する勝率よりわずかに下がる結果となり、相性があるようです。

ハードウェア

ミドルレンジのデスクトップパソコンを使用します。
CPU: Intel Core i5-7600K (4 コア、HT なし)
GPU: NVIDIA GeForce GTX 1060 6GB
dlshogi は 2,500nps 程度、やねうら王は 1 コアで 900,000nps 程度となります。

目標

昨年までとハード・ソフトともに大きく異なるため強さの比較はできません。棋力がないので自分での判断はできないのですが、深層学習型とも従来型とも違う棋風が出たら面白いなと思っています。

[1] 山岡忠夫 https://github.com/TadaoYamaoka/DeepLearningShogi
[2] 加納邦彦、山岡忠夫
https://github.com/TadaoYamaoka/DeepLearningShogi/releases/tag/denryu2020
[3] 磯崎元洋 https://github.com/yaneurao/YaneuraOu
[4] 瀧澤誠 https://mk-takizawa.github.io/elmo/howtouse_elmo.html
2021 年 3 月 18 日作成 日髙雅俊 https://github.com/select766

AobaZero のアピール

AobaZero は、AlphaZero の将棋の実験を、市販で出回っているような GPU を使って再現することを目標としたプロジェクトです。計算資源の提供を世界中のユーザにお願いして棋譜を生成しています。先駆的プロジェクトである LeelaZero（囲碁）や LCZero（チェス）のようにソースコードを公開しています（注 1)。

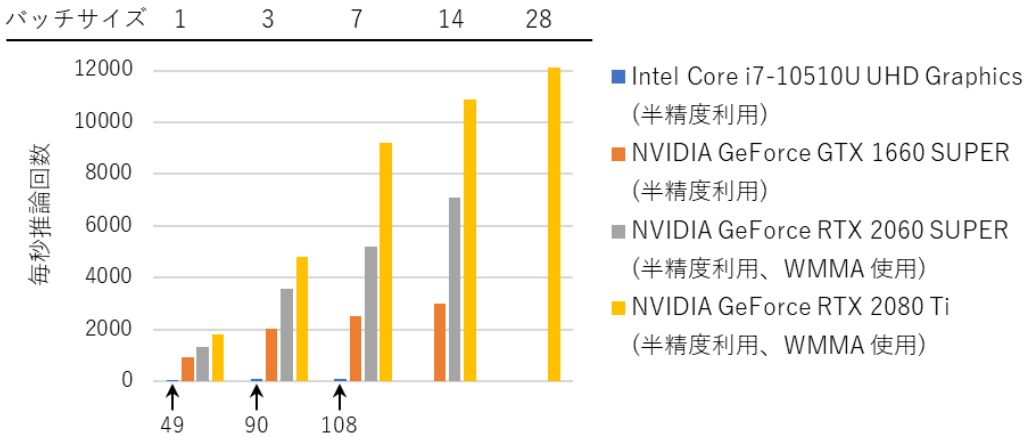
棋譜生成数は 3600 万（昨年 730 万程度）に達し、AlphaZero の実験で生成された棋譜数(2400 万）を超えました。このプロジェクト開始から約 1 年 11 か月たち、AlphaZero の学習アルゴリズムの実装も一通り書き終え、開始時に計画していた計算実験を遂行することができました。ここまでこのプロジェクトに参加して頂いた皆さまに感謝いたします。

モンテカルロ木探索のシミュレーション数を制限して、AlphaZero の対局実験の 10 秒相当の思考を模倣し、Elo レーティングで AobaZero の棋力を推定しました（注 2)。このような条件下で AobaZero の棋力は AlphaZero に、レーティング差にして後手番で 50、先手番で 320 程度まで迫っているのではないかと予想しています。また、floodgate(注 3)のレーティングはシミュレーション数 800 で着手を決定した場合、3150 点程度には達することも分かりました(昨年は 3000 点)。AobaZero の実装や実験セットアップの妥当性には今後、さらなる検証が望まれます。

出力チャネル数 256、ブロック数 20 の ResNet からなるニューラルネットワーク（NN）の順伝播に関しては、2020 年ごろ市場でよく出回ったような GPU でこれを効率よく処理するため、cuDNN などの汎用な深層 NN ライブラリを使わずに、大きさ 9×9 の盤面に特化したコードも独自に実装しました(注 2)。

この実装は、基本的には OpenCL1.2 の API を使っていて、Intel UHD Graphics のようなノート PC の CPU に内蔵されている非力な GPU でも動作できるように書かれています。また、NVIDIA 社の CUDA API「Warp Matrix Multiply Accumulate (WMMA)」も使うことができ、同社の Volta 以降の GPU がもつ混合精度の行列演算ユニット「Tensor Core」を利用可能です。

図に現状の性能を示します。順伝播を行う GPU のタスクはバッチを組んで処理することによって効率は向上し、バッチの大きさが 28 に達したあたりで GTX 2080 Ti の性能はピークに到達 、毎秒推論回数は 1 万 2 千を超えます。



図：2020 年ごろに市場によく出回った GPU を使った AobaZero の NN の推論効率。CPU 側のメモリにストアされている入力値から、同メモリに出力値をストアするまでを推論 1 回とした。

注 1：https://github.com/kobanium/aobazero
注 2：山下宏、保木邦仁、小林祐樹、AobaZero の高速化と現在の状況、CSA 会誌、vol.32, 2021 年
注 3：コンピュータ将棋連続対局場所（floodgate),
http://wdoor.c.u-tokyo.ac.jp/shogi/floodgate.html

付録：
将棋のルールだけを教えて自己対戦だけで学習した AI が将棋の知識を獲得していく過程が分かってきました。結果として、一般的に指されている戦型や囲いはほぼ再発見できたように思えます。

確認された囲い：雁木、矢倉、美濃囲い、高美濃、銀冠、左美濃、中住まい、右玉、
矢倉穴熊
確認された戦型：相掛かり、横歩取り、横歩取り青野流、角換わり棒銀、
角換わり早繰り銀、角換わり腰掛け銀、後手四間飛車、
矢倉脇システム、ひねり飛車

先手番では振飛車は指しませんが、後手番では四間飛車を好んで指していました。
穴熊も矢倉から穴熊に組み替える矢倉穴熊は指します(▲78 金▲67 金型)。
▲78 金▲79 金の正統な？穴熊は見つけていません。

これらを再発見できたのは、これらの囲い、戦型が「勝ちやすい」指し方なのだと思います。その一方で、一目散に穴熊を目指す指し方や、多彩な振飛車(中飛車、三間飛車)などは見つけていません。人間によって長く指されている指し方を一部発見できていないのは、この手法が万能ではないことも示していると思います。

棋風としてはすぐ殴り掛かる居飛車党で相掛かりや角換わりを好み、これは AlphaZero で公開された 100 棋譜と似ています。序盤の微妙な駒組の評価や、王が三段目以上に上がったときの形勢判断、入玉できるか、の見切りの力などが優れていると思います。その一方で、終盤での読み抜け、飛角の長い利きの両取りをうっかり、などが弱点です。

他のソフトに対して宣言勝ちをするのが上手いです。特に先手番での宣言率が高く、対 elmo(WCSC27)だと持ち時間が長いと勝ち将棋の 99%は宣言勝ちになります。5 手詰があっても宣言勝ちを選ぶことがあります。自己対戦でも先手の宣言率が高く、27 点法(先手は 28 点で勝ち、後手は 27 点で勝ち)は(先手は 29 点で勝ち)など、より先手に厳しくすべきなのかもしれません。学習棋譜の先手の勝率は 54%程度です。

棋力の推移や現在の棋譜生成速度などはこちらのページで公開しています。
http://www.yss-aya.com/aobazero/
ノイズなしの自己対戦のサンプルはこちらのページで公開しています。
http://www.yss-aya.com/aobazero/no_noise/sample.html

「芝浦将棋 Softmax」のチーム紹介

2021 年 3 月 24 日
芝浦工業大学情報工学科
岩本裕大，粂川叶，村上陽大，五十嵐治一

1. はじめに

本稿は，第 31 回世界コンピュータ将棋選手権（2021 年 5 月 3 日～5 日開催）に出場予定の「芝浦将棋 Softmax」（シバウラショウギ　ソフトマックス）のアピール文書です．本チームは昨年に引き続いて 5 回目の出場です．本チームの原型は 2016 年まで出場していた「芝浦将棋 Jr.」チームですが，探索手法が従来の Min-max 探索（$\alpha\beta$ 探索）とは異なる MC Softmax 探索である点が大きく異なります．棋力的には従来の $\alpha\beta$ 探索手法のチームにはまだまだ及びませんが，アルゴリズムが単純でコーディングの容易さや並列性に優れています．以下，簡単に本チームの特徴を紹介していきます．

2. 開発メンバー

五十嵐は芝浦工業大学工学部情報工学科に勤務する教員です．岩本と粂川は五十嵐研究室の大学院生，村上は 2020 年度の卒業生です．

3.「芝浦将棋 Softmax」の特徴

本チームの特徴を，以下の１）～４）のようにまとめました．このうちの１)～３）が本チーム独自の探索方式である「MC Softmax 探索」[1]に関する説明です．この探索方式は, 文献[2][3]の研究が基になっています．

１）モンテカルロ・ソフトマックス（MC Softmax）探索を使用

現在のチェスや将棋のプログラムは「Min-max 探索」が伝統的な探索方式です．この探索方式では探索木のすべてのノードを探索する必要があります（全幅探索）が，$\alpha\beta$ カットなどの枝刈りの処理により探索にかかる計算時間を短縮しています（$\alpha\beta$ 探索）．この全幅探索に対して，そのゲーム特有の知識（ヒューリスティクス）を用いて探索するノードを限定したり，優先順位をつけて選択的に探索する「選択探索」という探索方式があります．コンピュータ囲碁で有名な AlphaGo はこの探索方式を採用しています．

本チームの探索方式は後者の選択探索に分類されます．特に，ノードの選択方式としてノード評価値の min-max 演算ではなく，確率分布に基づく選択（Softmax 探索）を使用して

います．したがって，探索木をルートノード（実際の盤面の局面）から選択して降りていく（読んで行く）際には，実際にサイコロをふりながら確率的に選んで末端局面まで降りていきます．この確率的選択方式は，AlphaGo のようなコンピュータ囲碁ソフトで用いられている「モンテカルロ木探索」(Monte Carlo Tree Search)における決定論的な木の選択方法（UCT など）とは一線を画しており，我々は「モンテカルロ・ソフトマックス探索方式」(Monte Carlo Softmax Search)と呼んでいます．

２）ノードの局面評価関数と探索深さ指標を用いた確率的なノード選択方策

前項で述べた MC Softmax 探索によりルートノードから探索木を降りていく際には，本チームでは指し手の良さを用いたボルツマン分布を利用します．すなわち，各ノードでの指し手の選択確率を次の式で計算し，その確率に従ってノードを選択していきます．

$$\pi(a|s) = exp(E_a(a;s)/T) / \sum_{x \in A(s)} exp(E_a(x;s)/T) \qquad (1)$$

ただし，s は局面（ノード），a は指し手，$E_a(a;s)$は局面 s における指し手 a の良さですが，指した後の局面ノード s'の評価値 $E_s(s')$で置き換えることにします．$A(s)$は s における合法手の集合，T は温度と呼ばれているパラメータです．温度が低ければ最良優先探索に，温度が高ければランダム探索に近づきます．ノードの評価値は，探索木の末端ノード（leaf）であればそのノードの局面評価関数により計算します．（今回のバージョンでは，leaf でさらに静止探索を行っています）

一方，内部ノードであれば子ノード $v(x;s)$の評価値 $E(v)$をその子ノードの選択確率 $\pi(x|s)$で重み付けた期待値

$$E(s) = \sum_{x \in A(s)} \pi(x|s) E\big(v(x;s)\big) \qquad (2)$$

で定義します．したがって，読んだ先（子孫ノード）に評価の高い手があるような手は高く評価されます．また，十分探索が進んで探索木をすべて展開した後では，(1)で T をゼロに近づける（低温化）と，Softmax 探索による探索結果は Min-max 探索の探索結果に近づいて行きます．

ここまでは昨年までのバージョンでした．今年は，「探索深さ指標」$d(s)$を Softmax 探索のための深さの指標として新たに定義し，これを用いて探索を行います．探索深さ指標は以下の式で定義しています．

$$d(n) \equiv \sum_{l \in leaf(n)} depth(n,l) \prod_{i=1}^{depth(n,l)} P(n_{i+1}|n_i) \qquad (3)$$

ここで，$leaf(n)$はノードnを根とする探索木の末端ノードの集合，$depth(n,l)$は根nから末端lへの経路の長さ，n_iは根$n=n_0$から末端 $l=n_{depth(n,l)}$までの経路上のノード，$P(s'|s)$はsからs'への選択確率（$s \xrightarrow{a} s' \Rightarrow P(s'|s)=\pi(s'|s)$）です．

この式(3)を探索の中で効率的に計算するため式変形します．末端ノードの探索深さ指標を0として，内部ノードの探索深さ指標を以下の式で再帰的に求めます．

$$d(s)=\sum_{s'\in C(s)}\{P(s'|s)d(s')+1\} \qquad (4)$$

C(s)はsの子ノードの集合です．この式変形による誤差は10^{-10}程度とごく僅かです（予備実験で確認済み）．

この探索深さ指標$d(s)$を用いて，ノード選択時の評価値に次のような補正を掛けることで，深さを考慮した探索を行います．

$$E_s(s)=E(s)\left\{1-\frac{1}{2d(s)}\right\}+E_0(s)\frac{1}{2d(s)} \qquad (5)$$

$E_0(s)$は局面 s の局面評価値です．静止探索を行わないなどの精度の低い評価関数を用いた場合は特に効果が大きく，用いない場合よりもレートにして300程度棋力が向上しました．

３）バックアップ操作

MC Softmax 探索の全体の流れを図１に示します．ルートノードから，２）の選択法に従ってノードを選択し，末端ノードまで到達すると，そのノードの子ノードを一段階だけすべて展開します．展開後は新たな末端ノードの評価値を局面評価関数で計算し，その値をルートノードへ向けて(2)の計算を繰り返し，ルートノードまでの経路上のノード評価値を更新していきます．我々はこの更新操作を「バックアップ操作」と呼んでいます．また，(1),(2)で定義される期待値操作の方法を「バックアップ方策」と呼び，２）で述べた「ノード選択方策」と区別しています．

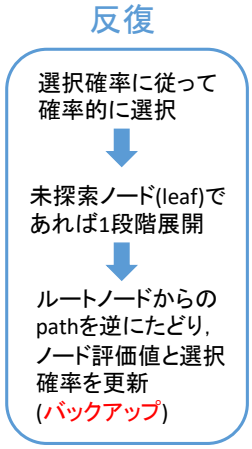


図１

また，MC Softmax 探索の名前の由来は，ルートノードから末端ノードへ到達するまで，(1)の選択確率に従って確率的にノード選択を行って経路が生成される２）の過程は，ルート局面における各指し手の良さを求めるためのモンテカルロ・サンプリング（一種のシミュレーション）に相当するからです．

上記のモンテカルロ・サンプリングを一定回数あるいは一定時間行った後，確率値の最も高い子ノードだけを次々に選択して得られた手順を最善応手手順であると決定します．

４）評価関数について

現在のところ，評価関数は既存の公開された評価関数をそのまま使用しています．今年度も昨年と同じく，Apery の KPPT 型評価関数を利用します．tanuki-の NNUE も実装していますが，KPPT の方が勝率が良かったためこちらを利用しています．

なお，末端ノードでの局面評価には静止探索（駒の取り合いだけを考慮する探索）を行って，その結果を局面評価値として返す処理を行っています．現バージョンのプログラムでは，この静止探索においては高速化のために従来のαβ探索を使用しています．

４．昨年のバージョンからの更新箇所のまとめ

昨年のバージョンからの更新個所を以下のように簡潔にまとめました．

(1)序盤定跡の使用

通常の序盤定跡を実装してそのまま使用することもできますが，独自の定跡データを作成しています．この方法は，序盤探索木を作成しておき，開始時に読み込んで，それを対局時に成長させ，終了時には元の序盤探索木へ格納します．これの反復により，対局を通して序盤定跡を成長させることができます．現在，実験中ですが，良い結果が出れば大会で使用します．

(2)ノード選択方策中の温度パラメータの調整

探索時のノード選択方策(1)において，温度パラメータをスレッドごと，あるいはノードごとに調整する方式を検討中です．大会までに間に合えば実装します．

(3)その他変更点

細かいリファクタリングを行っています。子ノードの保持方法を、ノードのポインタの動的配列としていたのを、ノードの可変長配列で直接保持するように変更し、ノードの生成・削除の処理を簡素化しました。

５．今後の課題

今年は，64 コア(128 スレッド)のワークステーションを使用する予定です．各スレッドが探索木を共有し，図１に示した処理を独立に行っています．

さらに，親ノードとそれ以下の探索木とをスレッドに分散して割当て，完全な並列分散化処理を行うことも可能です．上記のスレッド割当と探索木の分割処理とをうまく動的に行うことが今後の課題の一つです．今のところ，最善応手手順や有力手順の近傍を中心に，スレッドと探索木を探索途中で動的に割り当てることを考えています．

また，MC Softmax 探索方式は，ニューラルネットワークモデルによる評価関数表現と非常に相性が良いとされています．実際，2017 年 11 月開催の第 5 回将棋電王トーナメントでも mEssiah というチームが採用してくれました[5]．今後，ディープラーニングを用いた学習方式がコンピュータ囲碁だけではなく将棋へも波及して来ると予想されます．mEssiah の開発者の話によれば，ニューラルネットワークモデルによる評価値計算にはかなりの時間的コストがかかるが，GPU などを用いると多くの局面の評価値計算を一度に並列化して計算することができ，例えば，図 1 における子ノードの一斉展開と評価値計算には適しているとのことです[5]．我々の研究グループもこのメリットを実験により確認しています[6]．このように，局面評価関数としてニューラルネットワークモデルを用いることも本チームの今後の課題の一つです．

評価関数については，独自の学習法を考案しており，今後実装していく予定です．将棋では「Bonanza メソッド」と呼ばれる教師付学習方式が有名ですが，我々の研究グループは，この方式をより一般化した「方策勾配を用いた教師付学習法」を提案しています[7]．通常の教師付学習では，棋譜の着手を正解手として，この正解手の情報だけを用いますが，本学習法では，正解手以外の手（例えば有力な次善手）の評価値も学習データとして利用することが可能です．

さらに，３）で述べたモンテカルロ・サンプリングで生成された探索木において，全 leaf に出現する特徴量の重みを探索時と同様なモンテカルロ・サンプリングとバックアップ操作だけで学習することが可能です[1]．将来的にはこの学習法も実装していく予定です．

また，上記のような教師付学習だけではなく，報酬の最大化を目的とする強化学習（TD 法や方策勾配法），勝敗の予想確率を学習する回帰法，深い探索結果を利用する Bootstrap 法（RootStrap 法や TreeStrap 法）も，Softmax 探索とモンテカルロ・サンプリングの組合せで実行することが可能です．これにより，最善応手手順だけでなく，有力変化手順の近傍局面に出現する特徴量パラメータも，その重要度に応じて積極的に学習できるので，学習の精度や速度の向上に繋がると期待しています[8]．

６．おわりに

現在のコンピュータ将棋プログラムの多くは，探索方式（Minimax 探索の高速版である $\alpha\beta$ 探索）からソースコードのレベルまで，Stockfish[8]などのチェスプログラムから大きな影響を受けています．それに対して，本チームは Softmax 探索とモンテカルロ・サンプリングをベースにしています．本探索方式は囲碁プログラムで用いられているモンテカルロ木探索の一種と思われますが，プレイアウトを行わない点や，確率的選択を行っている点が異なっています．また，探索と複数局面の評価に関する並列化の効果も高く，特に局面評価では GPU を用いたニューラルネットワークモデルの並列計算に向いています．さらに，プログラム作成が容易で，他のゲームプログラムへの適用も容易あることから汎用性にも優れて

いると考えています.

まだまだ問題点も多いのですが，新しい探索方式と学習方式を研究する上では面白さが多く，開発者自身，今後の展開を楽しみにしております．最終的には，プロ棋士の棋譜を用いることなく，コンピュータ自身が自己対局を（あるいは他者との他流試合も）通して，探索法や局面評価関数を学習し，人類の棋力を超えて，新しい定跡や戦法を創出し，棋士や将棋ファンを大いに楽しませてくれることを目標としております.

参考文献

[1] 桐井杏樹，原悠一，五十嵐治一，森岡祐一，山本一将，“確率的選択探索の将棋への適用”，第 22 回ゲーム・プログラミング・ワークショップ 2017 予稿集，pp.26-33（2017）.

[2] 五十嵐治一，森岡祐一，山本一将，“方策勾配法による静的局面評価関数の強化学習についての一考察”，第 17 回ゲームプログラミングワークショップ 2012 予稿集, pp.118-121 (2012).

[3] 原悠一，五十嵐治一，森岡祐一，山本一将，“ソフトマックス戦略と実現確率による深さ制御を用いたシンプルなゲーム木探索方式”，第 21 回ゲーム・プログラミング・ワークショップ 2016 予稿集，pp.108-111(2016).

[4] 岩本 裕大 ，五十嵐 治一，“コンピュータ将棋における MC Softmax 探索のための探索深さの指標”，第 25 回ゲーム・プログラミング・ワークショップ 2020 予稿集，pp.46-52（2020.11.14-15），

[5] コンピュータ将棋ソフト mEssiah の内部構造, https://qiita.com/sakuramaru7777/items/ebb397eef94fc02be2d8

[6] 吉野 拓真 ，五十嵐 治一 ，川島 馨,“MC Softmax 探索における局面の並列評価：GPU とニューラルネットワークモデルの利用”，第 25 回ゲーム・プログラミング・ワークショップ 2020 予稿集，pp.16-21 （2020.11.14-15）

[7] 古根村光，山本一将，森岡祐一，五十嵐治一，“方策勾配を用いた将棋の局面評価関数の教師付学習：静止探索の導入と AdaGrad の適用”，第 22 回ゲーム・プログラミング・ワークショップ 2017 予稿集，pp.1-7（2017）.

[8] 五十嵐治一，森岡祐一，山本一将，“MC Softmax 探索における局面評価関数の学習”，第 23 回ゲーム・プログラミング・ワークショップ 2018 予稿集，pp.212-219 （2018），

==========

2021年

柿木将棋のアピール文章

今年は昨年と殆ど変わりません。そのため、昨年のアピール文章も添付します。

ここ数年、開発は行っておりません。

1985年頃に開発を始めました。

前向き枝刈を行う選択探索で、アルファベータ法を使っていました。

評価関数は、手作業で作成・調整していました。当時は、みんなそうでした。

2007年、ボナンザの影響を受け、全幅探索を行い、ボナンザ法で学習した評価関数を使うプログラムを新しく作成しました。

2008年から2012年までは、従来の選択探索のプログラムと新しい全幅探索のプログラムを組み合わせていました。

2013年、新しい全幅探索のプログラムだけとしました。

次のような手法を使っています。

・全幅探索

・局面の評価関数は、ボナンザ法で学習

・利きデータを使用し、bit-boardは使っていない。

・局面の評価項目は、独自のもの

・山下さんの 0.5手延長方式を採用

・並列化 PVS

・NULL move 枝刈

・王手延長

・静止探索

今年、フィッシャールールの時間は、昨年と同じで、次です。

>対戦開始当初の持ち時間は１５分とし、自らの手番となるごとに５秒加算される。

320手で引き分けとなるのは、多分、2020年からで、多分、未対応なので、今後、対応する予定です。

2012年から、2018年は、次のCPUを使った自作PCで参加しました。

Core i7-3960X Extreme 3.3GHz 6core

2019年もその予定でしたが、起動しなかったので、急遽、次のCPUを使ったノートPCで参加しました。

Core i7-6600U 2.6GHz 2core

今年は、初めて、リモートの開催なので、次のCPUを使った自作PCで参加の予定です。

Core i9 9900K 3.6GHz 8core

2021/03/27 柿木

==========

2020年

柿木将棋のアピール文章

今年は昨年と殆ど変わりません。そのため、昨年のアピール文章も添付します。

1985年頃に開発を始めました。

前向き枝刈を行う選択探索で、アルファβ法を使っていました。

評価関数は、手作業で作成・調整していました。

2007年、ボナンザの影響を受け、全幅探索を行い、ボナンザ法で学習した評価関数を使うプログラムを新しく作成しました。

2008年から2012年までは、従来の選択探索のプログラムと新しい全幅探索のプログラムを組み合わせていました。

2013年、新しい全幅探索のプログラムだけとしました。
次のような手法を使っています。

・全幅探索
・局面の評価関数は、ボナンザ法で学習
・利きデータを使用し、bit-boardは使っていない。
・局面の評価項目は、独自のもの
・山下さんの 0.5手延長方式を採用
・並列化 PVS
・NULL move 枝刈
・王手延長
・静止探索

今年、フィッシャールールの時間は、昨年と同じで、次です。
>対戦開始当初の持ち時間は１５分とし、自らの手番となるごとに５秒加算される。

2012年から、2018年は、次のCPUを使った自作PCで参加しました。
Core i7-3960X Extreme 3.3GHz 6core

昨年もその予定でしたが、起動しなかったので、急遽、次のCPUを使ったノートPCで参加しました。
Core i7-6600U 2.6GHz 2core

今年もこのノートPCで参加の予定です。

2020/03/26 柿木

==========

2019年

柿木将棋のアピール文章

今年は昨年と殆ど変わりません。そのため、昨年のアピール文章も添付します。

1985年頃に開発を始めました。
前向き枝刈を行う選択探索で、アルファβ法を使っていました。
評価関数は、手作業で作成・調整していました。

2007年、ボナンザの影響を受け、全幅探索を行い、ボナンザ法で学習した評価関数を使うプログラムを新しく作成しました。

2008年から2012年までは、従来の選択探索のプログラムと新しい全幅探索のプログラムを組み合わせていました。

2013年、新しい全幅探索のプログラムだけとしました。
次のような手法を使っています。

・全幅探索
・局面の評価関数は、ボナンザ法で学習
・利きデータを使用し、bit-boardは使っていない。
・局面の評価項目は、独自のもの
・山下さんの 0.5手延長方式を採用
・並列化 PVS
・NULL move 枝刈
・王手延長
・静止探索

今年、フィッシャールールの時間が変わりましたので、今後、その点だけ対応する予定です。

2019/03/25 柿木

==========

2018年

柿木将棋のアピール文章

今年は昨年と殆ど変わりません。そのため、昨年のアピール文章も添付します。

1985年頃に開発を始めました。
前向き枝刈を行う選択探索で、アルファβ法を使っていました。
評価関数は、手作業で作成・調整していました。

2007年、ボナンザの影響を受け、全幅探索を行い、ボナンザ法で学習した評価関数を使うプログラムを新しく作成しました。

2008年から2012年までは、従来の選択探索のプログラムと新しい全幅探索のプログラムを組み合わせていました。

2013年、新しい全幅探索のプログラムだけとしました。
次のような手法を使っています。

・全幅探索
・局面の評価関数は、ボナンザ法で学習
・利きデータを使用し、bit-boardは使っていない。
・局面の評価項目は、独自のもの
・山下さんの 0.5手延長方式を採用

・並列化 PVS

・NULL move 枝刈

・王手延長

・静止探索

参加する意味はあまりありませんが、ほぼ同じソフトを同じハードで参加しているので、

そういう意味のベンチマークはなるかと思います。

2018/03/25 柿木

追記

EXEファイル自体、昨年と同じで、まったく同じプログラムとハードで参加しました。

2018/05/10 柿木

==========

2017年

柿木将棋のアピール文章

今年は昨年と殆ど変わりません。そのため、昨年のアピール文章も添付します。

1985年頃に開発を始めました。

前向き枝刈を行う選択探索で、アルファβ法を使っていました。

評価関数は、手作業で作成・調整していました。

2007年、ボナンザの影響を受け、全幅探索を行い、ボナンザ法で学習した評価関数を使うプログラムを新しく作成しました。

2008年から2012年までは、従来の選択探索のプログラムと新しい全幅探索のプログラムを組み合わせて

いました。

2013年、新しい全幅探索のプログラムだけとしました。

次のような手法を使っています。

・全幅探索

・局面の評価関数は、ボナンザ法で学習

・利きデータを使用し、bit-boardは使っていない。

・局面の評価項目は、独自のもの

・山下さんの 0.5手延長方式を採用

・並列化 PVS

・NULL move 枝刈

・王手延長

・静止探索

2017/03/28 柿木

追記

選手権前に floodgate で、しばらく対戦させたところ、レーティングは 約2300 です。

2014/8/30 には 2234 でしたが、その後、殆ど改良はしてなくて、ハードも同じです。

2017/04/26 柿木

==========

2016年

柿木将棋のアピール文章

今年は昨年と殆ど変わりません。そのため、昨年のアピール文章も添付します。

変わる予定なのは、次の点だけです。

・フィッシャールールに対応（予定）

2016/03/26 柿木

==========

2015年

柿木将棋のアピール文章

今年は昨年と殆ど変わりません。そのため、昨年のアピール文章も添付します。

変わったのは、次の点です。

・秒読みルールに対応

・バグの修正

・評価関数を少し改良したつもり

昨年の選手権直後の 2014/5/7、floodgateでのレーティングは 2177 でした。

上記改良を行い、2014/8/30 には 2234 になったので、57 上がりました。

その後は、改良できなかったので、この 1 年の改良点は、これだけです。

2015/04/02 柿木

==========

2014年

柿木将棋のアピール文章

今年は昨年と殆ど変わりません。そのため、昨年のアピール文章も添付します。

変わったのは、次の点です。

・バグの修正

・評価関数を少し改良したつもり

2014/03/26 の時点のfloodgateでのレーティングは、2178 と昨年とほぼ同じです。

2014/03/26 柿木

==========

2013年

柿木将棋のアピール文章

今年は、去年と大きく変わりました。

昨年まで、全幅探索のプログラムと選択探索のプログラムの２種を組み合わせていました。

今年は、全幅探索のプログラムだけにしました。ようやく、その方が強くなったからです。

全幅探索のプログラムは、2007年に新しく開発を始めたものです。

ボナンザの影響を受け、全幅探索を行い、評価関数はボナンザ法の学習で作成しているものです。

ただし、ボナンザとは色々違っています。

例えば、bit-boardは使わず、利きデータを使っています。

評価関数の評価項目は独自のものです。

floodgate では、今年のプログラムは昨年のプログラムに対して、７割程度の勝率があります。

ただし、floodgateでのレーティングの点数は、後述するように、少し不可解な点があります。

昨年のプログラムは、昨年のfloodgateでの点数は 2145点、

ほぼ同じプログラムが今年は、2089点と少し下がりました。

全幅探索のプログラムは、昨年に対して、次の改良を行いました。

１．並列化

２．山下さんの 0.5手延長方式を採用

３．評価関数の調整

全幅探索のプログラムのfloodgateでのレーティングは、改良前 2094点、

改良後 2184 点と 90点上がった程度です。

2013/04/18 柿木

==========

カツ丼将棋アピール文書

ーー開発者紹介ーー

代表者：松本浩志
所属：NPO法人AI電竜戦プロジェクト　理事長(代表)
(認証中 2021/3/21時点)

ーー今回の特徴ーー

●今回は完全フロムスクラッチではなく、公開ソフトをカツ丼将棋exeで包んで出場します。
●カツ丼将棋と公開ソフトでコラボ、リレーしながら指します。
●カツ丼将棋exeにmultiponder(複数候補先読み)を実装し、**予選から決勝まですべてノータイムで指します**。
※自分の手番で考えてしまうと、単に公開ソフトを代理で出しているだけになってしまうので。。。
●相手の強さを鑑みて、**穴角を目指す場合があります**。
●**不成を積極的に活用**し、相手の読み筋を外して時間を使わせます。
●今回使う公開ソフトはやねうら王v6.0.0+水匠3

ーーポエムーー

私は将棋倶楽部24でAIを出して、普通の指す将にも将棋AIに触れてもらおうということをずっとやってきている。それをやるためにカツ丼坊という将棋倶楽部24を自動で操縦するシステムを作った。これはUWSCを使ったもので画像認識を駆使したものである。画像認識を使うと人間のやる作業を直観的に再現できるのがよいところだが、先方のアップデートで画像作り直しリスクがあるのと、相手の指し手を認識するのに時間がかかる。遅い時は体感0.5秒か。そんな折に24がHTML5化された。いずれjava版は廃止されるので現在のカツ丼坊は作り直せばならなくなった。HTML5で動作しているので、python+seleniumで自動化ができるのと、chromの通信ログを見ることにより高速に指手を認識できる。中断対局の処理も簡単になった。せっかく指手が高速にできるなら極限まで光速で指すノータイム指しソフトを作ってみようと思った。それをやるためにはmultiponderを実装し、相手の手番中に複数手を考え、相手の指し手に対する応手を即指しする。候補になければ仕方がない0.2秒考えたものを指す。同様のことは第5回電王トーナメント準優勝のshotgunがやっているが、相手の時間を使わせるため微妙に最善手ではない、というのは今はしていない。いわばチョットgun方式である。この手法で現在24で光の速さで指す非人間的なソフトとして放流している。Multiponder、簡単そうで将棋所の標準機能では実現できないので通信も自分でつくってないと難しいのでそれなりに価値があるかもしれない。

公開ソフトを使うことについては、AMD5950x+RTX3090のマシンを購入し、これはディープラーニングの勉強のためにそちら方面に移行するつもりで、今のカツ丼将棋を改良する予定はない。なので前述した通り、24でノータイムで強いという非人間的な友達のなくなるソフトとシステムを作ったので、今回の選手権でもそのネタでいく。公開ソフトをカツ丼将棋exeが包むことで、中身を知らないという馬鹿を輩出すると同時に、中身をまるっと入れ替えられるメリットもある。

電竜戦のきふわるべは、駒得のみの評価関数でユニークな動きをしていた。もしカツ丼将棋を駒得だけにしたらどうなるかとためしてみたところ、取る手がないので探索順序が9九から調べ始める関係で、9八香、9九角、その後6九の金が7九と6九を反復横跳びする穴角戦法が偶然にもできてしまった。せっかくできたのなら使えるところで使いたい。

第31回世界コンピュータ将棋選手権

Miacisアピール文書

迫田真太郎

2020/03/24

Miacisの特徴

- 評価値を確率分布として出力

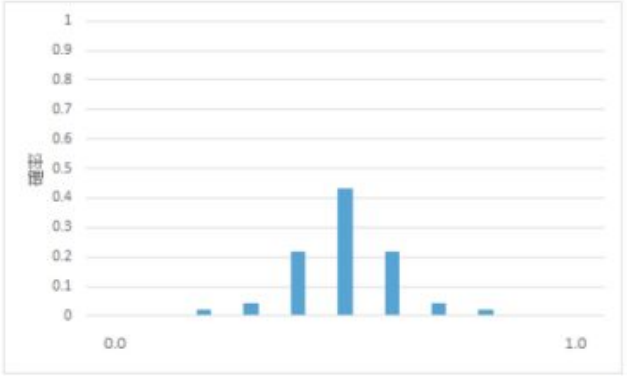
従来の大多数

評価関数

予測勝率 0.55

提案手法

評価関数

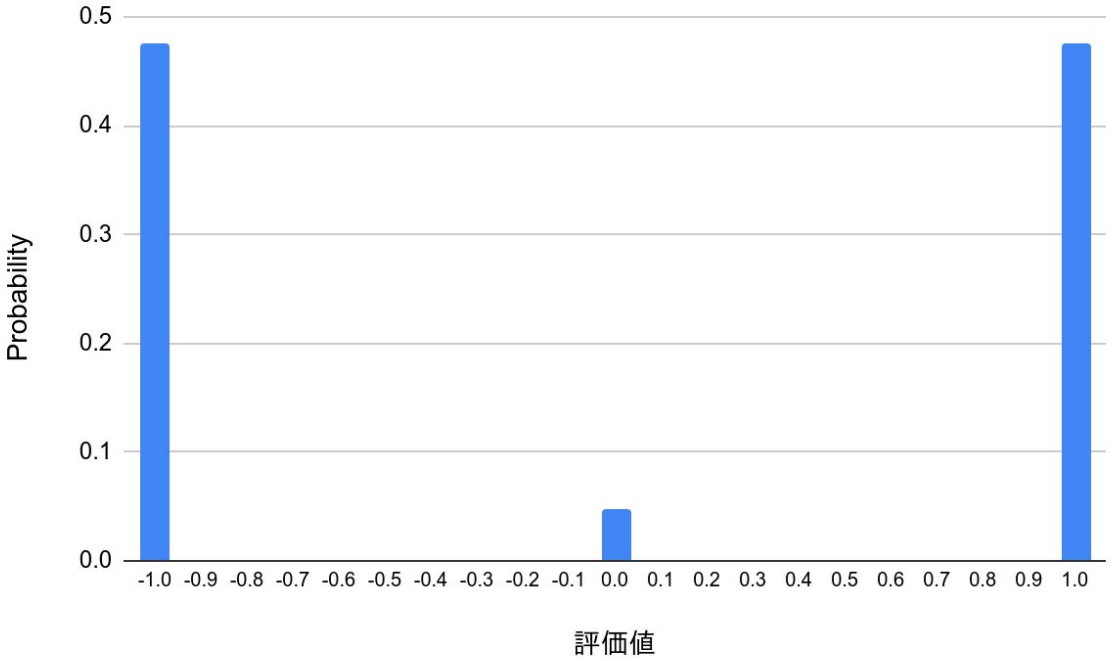


1年前からの改善点

- TensorRTの導入
  - LibTorchと連携したTRTorchを利用
  - 高速化によりR+100ほど
- 学習の改善
  - 強化学習の前に教師あり学習を導入
  - 評価関数の精度向上によりR+100ほど
- floodgateでR3800程度
  - ブログ記事

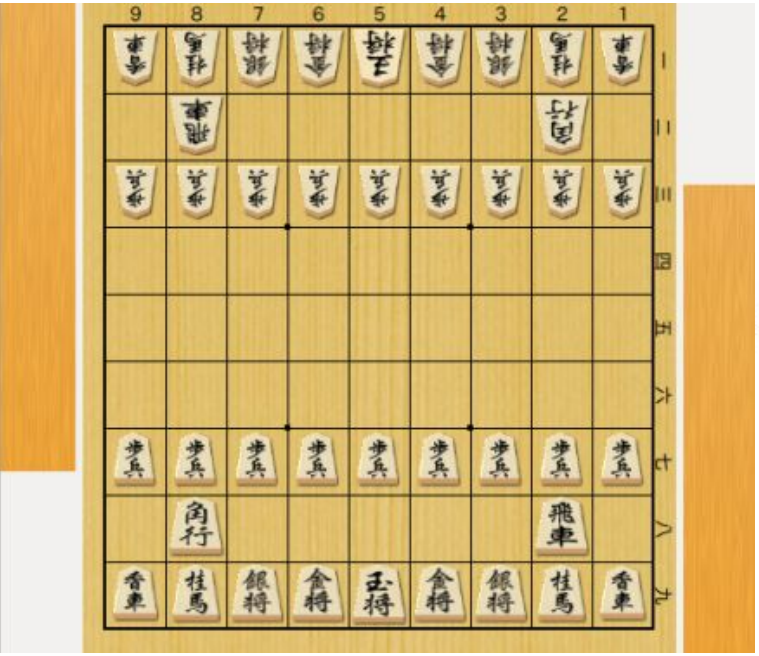
学習方法

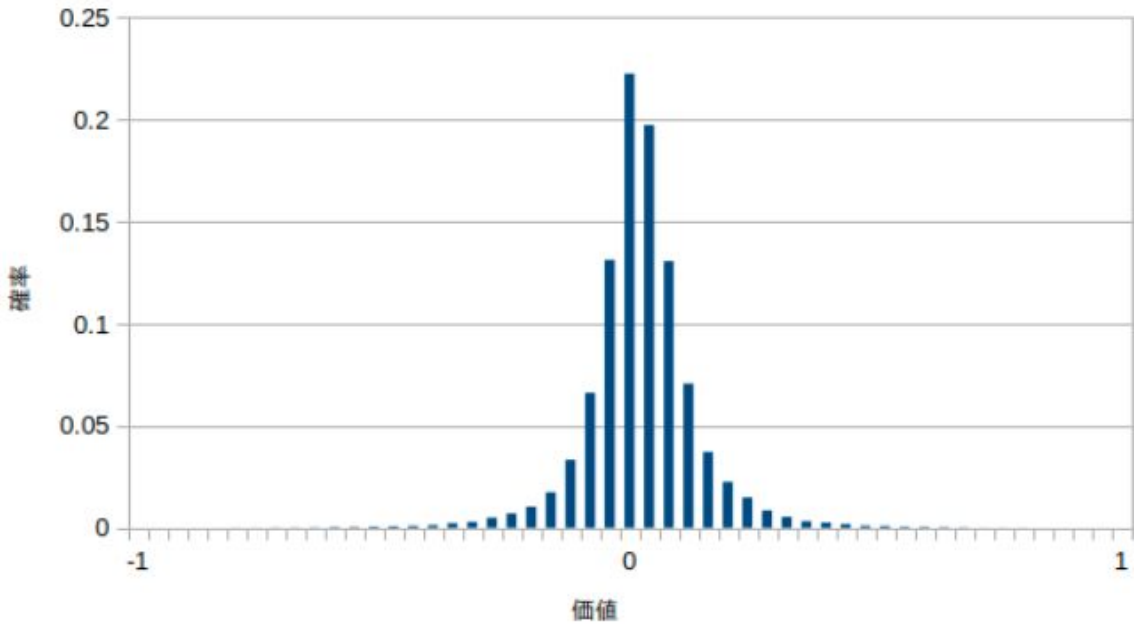
- 教師あり学習と強化学習を併用
  - 最終結果[-1, 0, 1]を予測する教師あり学習ではどの局面でも以下のような分布になってしまう



- 強化学習(TD学習)により数手先の評価値を予測
  - 評価値の安定度を学習

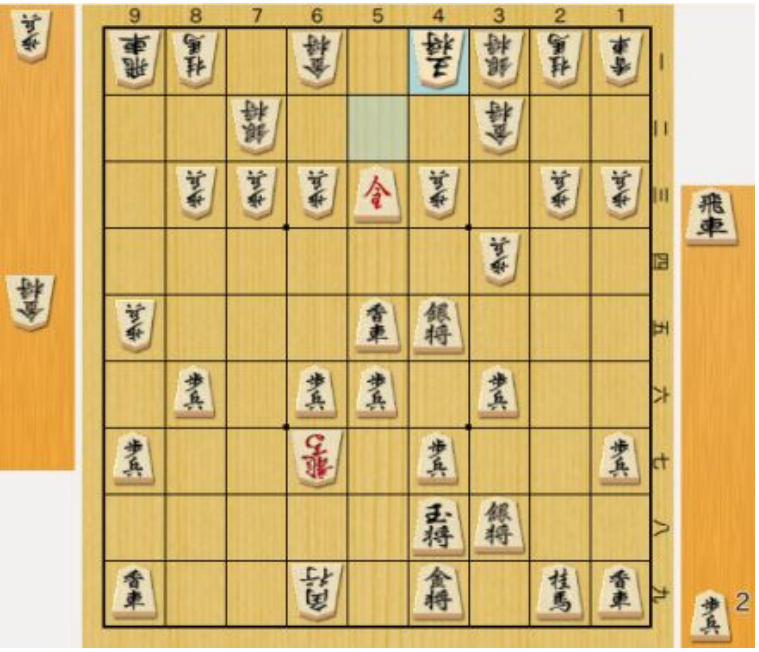
出力例1

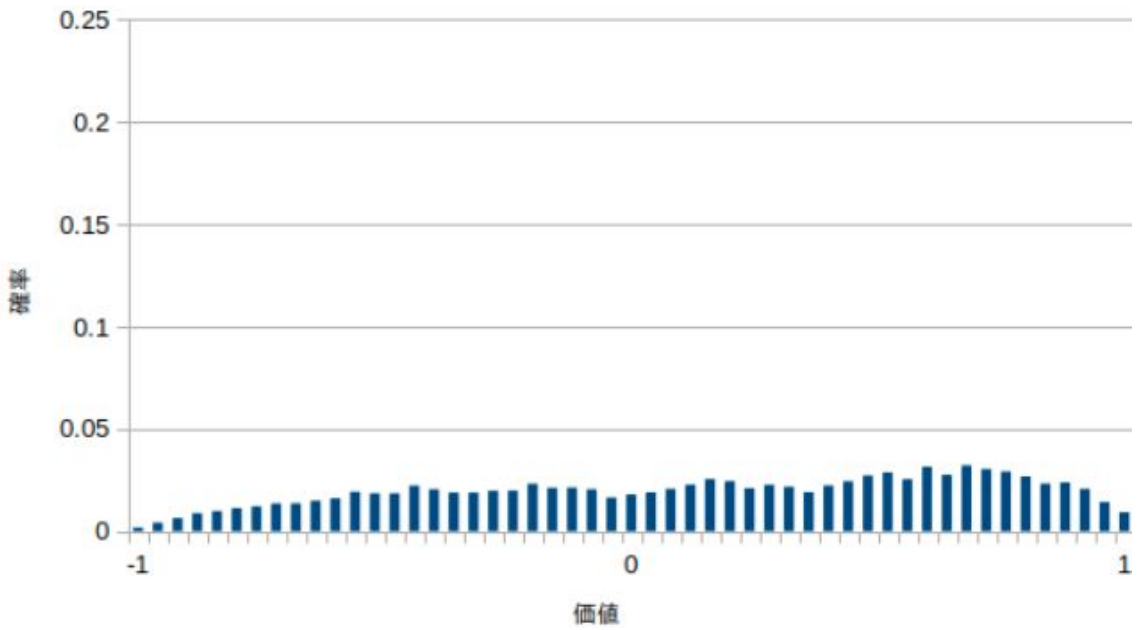




- 初期局面では 0 付近に高い確率を示す

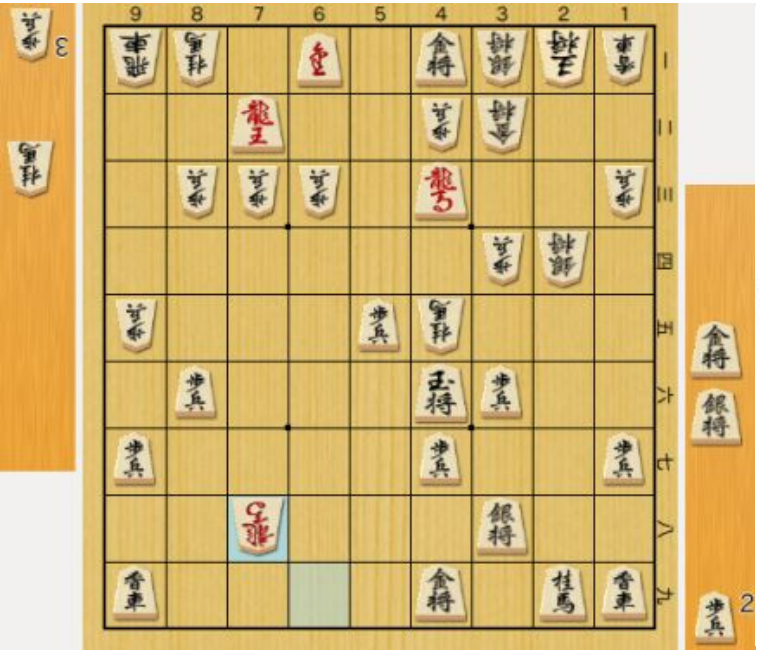
出力例2

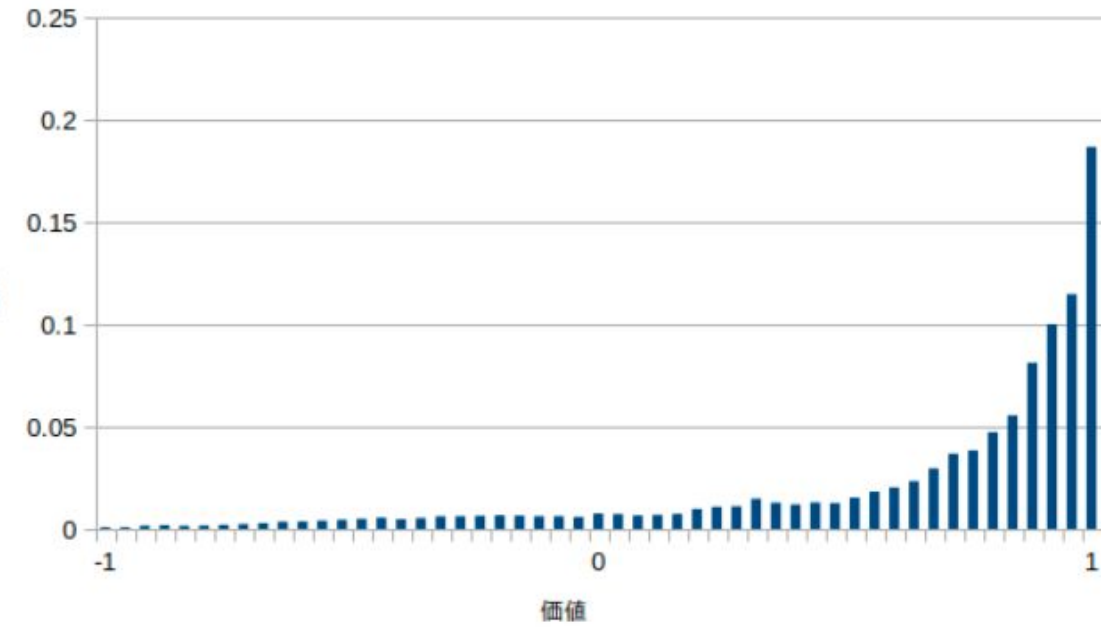




- 中盤の難所 (？) では、期待値はほぼ 0(互角) だが幅広い値に確率を示す

出力例2





- 勝ちになった終盤では 1 付近に高い確率を示す

手抜きについて

手抜きチーム

2021 年 3 月 28 日

手抜きは CSA プロトコルで対局を行うコンピュータ将棋プログラムです。開発者らが将棋プログラムの仕組みを理解するために作っています。
リポジトリ：`https://github.com/hikaen2/tenuki-d`

名前の由来

かっこいい名前が思いつかなかったため『手抜き』にしました。

特長

- 初段〜二段くらいの棋力
- α β 探索
- D 言語で実装

開発者

鈴木太朗：自転車と合唱と Emacs が好きです。誰か df-pn 教えてください。Twitter: @hikaen2
玉川直樹：好きなコードは add9 です。将棋ウォーズ 2 段。Twitter: @Neakih_kick

1　今年の目標

NNUE を理解する。

2　使用ライブラリ

- 『どうたぬき』(tanuki- 第 1 回世界将棋 AI 電竜戦バージョン) の評価関数ファイル nn.bin[*1]

[*1] `https://github.com/nodchip/tanuki-/releases/tag/tanuki-denryu1`

表 1　どうたぬきの nn.bin の内訳

No.	内容	開始位置	サイズ (バイト)	備考
1	version	0	4	0x7af32f16
2	hash	4	4	0x3e5aa6ee
3	size	8	4	0x000000b2(architecture のサイズ. 可変)
4	architecture	12	178	-
5	header	190	4	-
6	FeatureTransformer.biases	194	512	$\vec{\boldsymbol{b}}_1$: int16_t が 256 個
7	FeatureTransformer.weights	706	64,198,656	$\boldsymbol{W}_1$: int16_t が 256×81×1548 個
8	header	64,199,362	4	-
9	HiddenLayer1.biases	64,199,366	128	$\vec{\boldsymbol{b}}_2$: int32_t が 32 個
10	HiddenLayer1.weights	64,199,494	16,384	$\boldsymbol{W}_2$: int8_t が 32×512 個
11	HiddenLayer2.biases	64,215,878	128	$\vec{\boldsymbol{b}}_3$: int32_t が 32 個
12	HiddenLayer2.weights	64,216,006	1,024	$\boldsymbol{W}_3$: int8_t が 32×32 個
13	OutputLayer.biases	64,217,030	4	$\vec{\boldsymbol{b}}_4$: int32_t が 1 個
14	OutputLayer.weights	64,217,034	32	$\boldsymbol{W}_4$: int8_t が 1×32 個
			合計: 64,217,066	

3　メモ：nn.bin の構造について

どうたぬきの評価関数ファイル nn.bin は 64,217,066 バイトある。その内訳を表 1 に示す。値はすべてリトルエンディアンで格納されている。

表 1 の No.4: architecture には人間に可読な形式でニューラルネットワークの構造が書かれている。どうたぬきの場合は

```
Features=HalfKP(Friend)[125388->256x2],Network=AffineTransform[1<-32](ClippedReLU[32](
AffineTransform[32<-32](ClippedReLU[32](AffineTransform[32<-512](InputSlice[512(0:512)])))))
```

と書かれている。この文字列の長さはニューラルネットワークの構造によって変わるので，その長さ（バイト数）が No.3 size に格納されている。どうたぬきの場合は 0x000000b2 = 178 である。

No.7: FeatureTransformer.weights ($\boldsymbol{W}_1$) は 256 × 125388 行列である：

$$\boldsymbol{W}_1 = \begin{pmatrix} w_{1(0,0)} & \cdots & w_{1(0,125387)} \\ \vdots & \ddots & \vdots \\ w_{1(255,0)} & \cdots & w_{1(255,125387)} \end{pmatrix}$$

この行列は 125388 次元の特徴ベクトルを 256 次元ベクトルに変換する。ここで 125388 = 81×1548 は KP（K:自玉と P:玉以外の駒 の組み合わせ）が取りうる場合の数である。81 は K の取りうる場合の数（玉が 1 一から 9 九まで。盤面のアドレスを図 1 に示す），1548 は P の取りうる場合の数（内訳を表 2 に示す）である。この特徴ベクトルは駒のあるところが 1，駒のないところが 0 になる二値ベクトルである。将棋の駒は玉を除くと 38 枚あるので，特徴ベクトルは 125388 次元のうち 38 ヶ所だけが 1 で，のこりがすべて 0 のベクトルとなる。特徴ベクトルの具体例については次節に記す。

9	8	7	6	5	4	3	2	1	
72	63	54	45	36	27	18	9	0	一
73	64	55	46	37	28	19	10	1	二
74	65	56	47	38	29	20	11	2	三
75	66	57	48	39	30	21	12	3	四
76	67	58	49	40	31	22	13	4	五
77	68	59	50	41	32	23	14	5	六
78	69	60	51	42	33	24	15	6	七
79	70	61	52	43	34	25	16	7	八
80	71	62	53	44	35	26	17	8	九

図 1　盤面のアドレス

$\boldsymbol{W}_1$ は nn.bin に column-major で格納されている。すなわち最初の値は $w_{(0,0)}$，次の値は $w_{(1,0)}$，… という順番で格納されている。一方，後述する $\boldsymbol{W}_2$, $\boldsymbol{W}_3$, $\boldsymbol{W}_4$ は row-major で格納されている。すなわち最初の値は $w_{(0,0)}$，次の値は $w_{(0,1)}$，… という順番で格納されている。$\boldsymbol{W}_1$ とそれ以外で格納順が異なることに注意されたい。

No.6: FeatureTransformer.biases ($\vec{\boldsymbol{b}}_1$) は 256 次元ベクトルである：

$$\vec{\boldsymbol{b}}_1 = \begin{pmatrix} b_{1(0)} \\ \vdots \\ b_{1(255)} \end{pmatrix}$$

このベクトルは $\boldsymbol{W}_1$ の出力である 256 個のニューロンに足すバイアスである。

$\boldsymbol{W}_1$ に KP の 125388 次元の特徴ベクトル $\vec{\boldsymbol{x}}$ を掛けて，$\vec{\boldsymbol{b}}_1$ を足すと，256 次元のベクトル $\vec{\boldsymbol{y}}$ が得られる：

$$\underbrace{\begin{pmatrix} y_0 \\ \vdots \\ y_{255} \end{pmatrix}}_{\vec{\boldsymbol{y}}} = \underbrace{\begin{pmatrix} b_{1(0)} \\ \vdots \\ b_{1(255)} \end{pmatrix}}_{\vec{\boldsymbol{b}}_1} + \underbrace{\begin{pmatrix} w_{1(0,0)} & \cdots & w_{1(0,125387)} \\ \vdots & \ddots & \vdots \\ w_{1(255,0)} & \cdots & w_{1(255,125387)} \end{pmatrix}}_{\boldsymbol{W}_1} \times \underbrace{\begin{pmatrix} x_0 \\ \vdots \\ x_{125387} \end{pmatrix}}_{\vec{\boldsymbol{x}}}$$

この 256 次元ベクトル $\vec{\boldsymbol{y}}$ を先手分と後手分あわせて 512 次元にしたベクトルを $\vec{\boldsymbol{z}}_1$ とする。$\vec{\boldsymbol{z}}_1$ を No.10: HiddenLayer1.weights ($\boldsymbol{W}_2$：32×512 行列) と No.9: HiddenLayer1.biases ($\vec{\boldsymbol{b}}_2$：32 次元ベクトル) に入力すると 32 次元ベクトル $\vec{\boldsymbol{z}}_2$ が得られる：

$$\underbrace{\begin{pmatrix} z_{2(0)} \\ \vdots \\ z_{2(31)} \end{pmatrix}}_{\vec{\boldsymbol{z}}_2} = \underbrace{\begin{pmatrix} b_{2(0)} \\ \vdots \\ b_{2(31)} \end{pmatrix}}_{\vec{\boldsymbol{b}}_2} + \underbrace{\begin{pmatrix} w_{2(0,0)} & \cdots & w_{2(0,511)} \\ \vdots & \ddots & \vdots \\ w_{2(31,0)} & \cdots & w_{2(31,511)} \end{pmatrix}}_{\boldsymbol{W}_2} \times \sigma \underbrace{\begin{pmatrix} z_{1(0)} \\ \vdots \\ z_{1(512)} \end{pmatrix}}_{\vec{\boldsymbol{z}}_1}$$

表 2　P の取りうる場合の数

No.	内容	開始序数	場合の数
1	味方の持駒の歩の 0 枚目〜18 枚目	0	19
2	相手の持駒の歩の 0 枚目〜18 枚目	19	19
3	味方の持駒の香の 0 枚目〜4 枚目	38	5
4	相手の持駒の香の 0 枚目〜4 枚目	43	5
5	味方の持駒の桂の 0 枚目〜4 枚目	48	5
6	相手の持駒の桂の 0 枚目〜4 枚目	53	5
7	味方の持駒の銀の 0 枚目〜4 枚目	58	5
8	相手の持駒の銀の 0 枚目〜4 枚目	63	5
9	味方の持駒の金の 0 枚目〜4 枚目	68	5
10	相手の持駒の金の 0 枚目〜4 枚目	73	5
11	味方の持駒の角の 0 枚目〜2 枚目	78	3
12	相手の持駒の角の 0 枚目〜2 枚目	81	3
13	味方の持駒の飛の 0 枚目〜2 枚目	84	3
14	相手の持駒の飛の 0 枚目〜2 枚目	87	3
15	味方の歩が１一〜９九にいる	90	81
16	相手の歩が１一〜９九にいる	171	81
17	味方の香が１一〜９九にいる	252	81
18	相手の香が１一〜９九にいる	333	81
19	味方の桂が１一〜９九にいる	414	81
20	相手の桂が１一〜９九にいる	495	81
21	味方の銀が１一〜９九にいる	576	81
22	相手の銀が１一〜９九にいる	657	81
23	味方の金 *が１一〜９九にいる	738	81
24	相手の金 *が１一〜９九にいる	819	81
25	味方の角が１一〜９九にいる	900	81
26	相手の角が１一〜９九にいる	981	81
27	味方の馬が１一〜９九にいる	1062	81
28	相手の馬が１一〜９九にいる	1143	81
29	味方の飛が１一〜９九にいる	1224	81
30	相手の飛が１一〜９九にいる	1305	81
31	味方の龍が１一〜９九にいる	1386	81
32	相手の龍が１一〜９九にいる	1467	81
			合計: 1548

* と・成香・成桂・成銀は金として扱う

ここで σ はベクトルの各要素を 0〜127 の範囲にクリップする活性化関数 clipped ReLU である。具体的には 0 以下の値は 0 にクリップする。127 以上の値は 127 にクリップする。

次に $\vec{z}_2$ を No.12: HiddenLayer2.weights ($\boldsymbol{W}_3$：32×32 行列) と No.11: HiddenLayer2.biases ($\vec{\boldsymbol{b}}_3$：32 次元ベクトル) に入力すると 32 次元ベクトル $\vec{z}_3$ が得られる：

$$\underbrace{\begin{pmatrix} z_{3(0)} \\ \vdots \\ z_{3(31)} \end{pmatrix}}_{\vec{z}_3} = \underbrace{\begin{pmatrix} b_{3(0)} \\ \vdots \\ b_{3(31)} \end{pmatrix}}_{\vec{b}_3} + \underbrace{\begin{pmatrix} w_{3(0,0)} & \cdots & w_{3(0,31)} \\ \vdots & \ddots & \vdots \\ w_{3(31,0)} & \cdots & w_{3(31,31)} \end{pmatrix}}_{\boldsymbol{W}_3} \times\sigma \underbrace{\begin{pmatrix} z_{2(0)} \\ \vdots \\ z_{2(31)} \end{pmatrix}}_{\vec{z}_2}$$

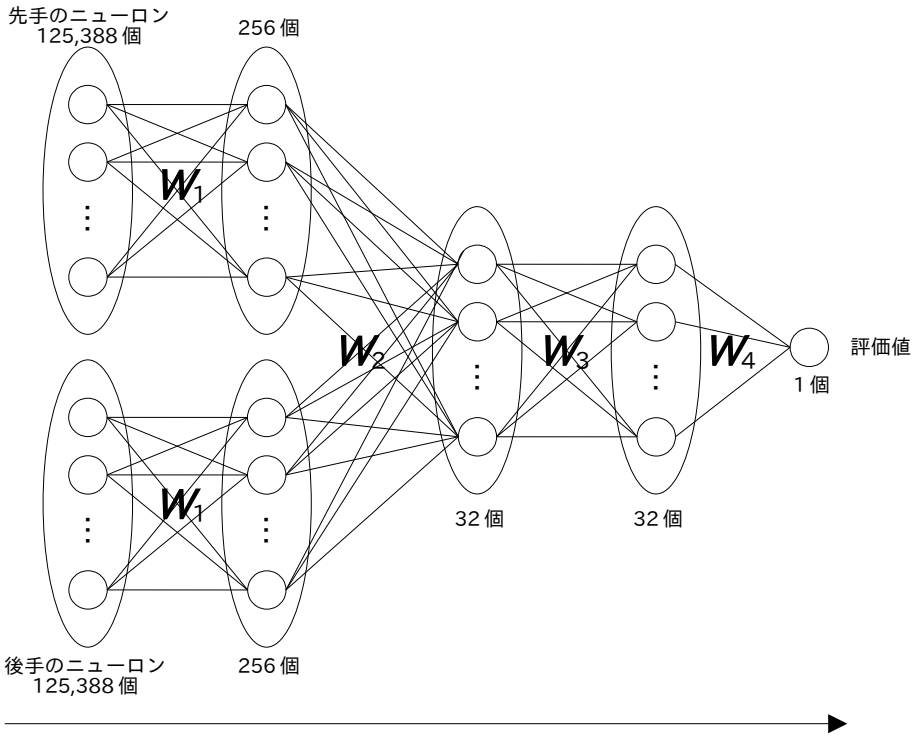


図 2　ニューラルネットワークの図（先手の評価値を求めるとき）

最後に $\vec{z}_3$ を No.14: OutputLayer.weights ($\boldsymbol{W}_4$：1×32 行列) と No.13: OutputLayer.biases ($\vec{\boldsymbol{b}}_4$：1 次元ベクトル) に入力すると 1 次元ベクトル $\vec{z}_4$ が得られる。これが評価値である：

$$\underbrace{\begin{pmatrix} z_{4(0)} \end{pmatrix}}_{\vec{z}_4} = \underbrace{\begin{pmatrix} b_{4(0)} \end{pmatrix}}_{\vec{b}_4} + \underbrace{\begin{pmatrix} w_{4(0,0)} & \cdots & w_{4(0,31)} \end{pmatrix}}_{\boldsymbol{W}_4} \times \sigma \underbrace{\begin{pmatrix} z_{3(0)} \\ \vdots \\ z_{3(31)} \end{pmatrix}}_{\vec{z}_3}$$

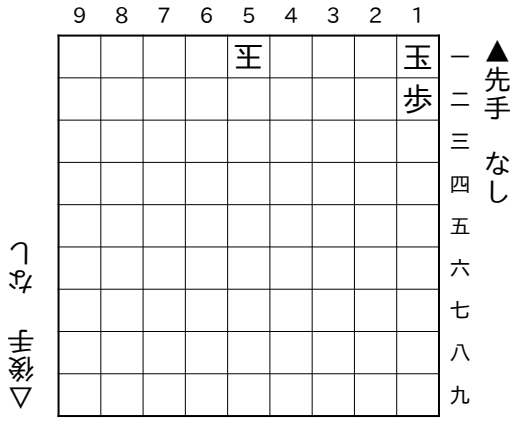
以上の計算を図に表すと図 2 となる（バイアス $\vec{\boldsymbol{b}}_n$ と活性化関数 σ の記載は省略している)。

以上の計算を行うサンプルプログラムを `https://github.com/hikaen2/nnue-test` に作成した（D 言語)。

3.1　特徴ベクトルの具体例

例 1

以下の局面を考える。



先手の特徴ベクトルを考える。先手玉が１一にいる。図 1 を見ると１一のアドレスは 0 なので, このときの K の序数は 0 になる。１二に（先手から見て）味方の歩がいる。表 2 を見ると味方の歩の序数は 90 から始まっている。これに１二のアドレスである 1 を足すと $90+1=91$ になり，１二にいる味方の歩を表現できる。K と P を合わせると $0\times 1548+91=91$ となる。これは 125388 次元ベクトルの（0 から数えて）91 番目の要素に 1 を立てることを意味する。したがって先手の特徴ベクトルはこのようになる：

$$\vec{x}=\begin{pmatrix} x_0 & = & 0 \\ \vdots & & \\ x_{90} & = & 0 \\ x_{91} & = & 1 \\ x_{92} & = & 0 \\ \vdots & & \\ x_{125387} & = & 0 \end{pmatrix}$$

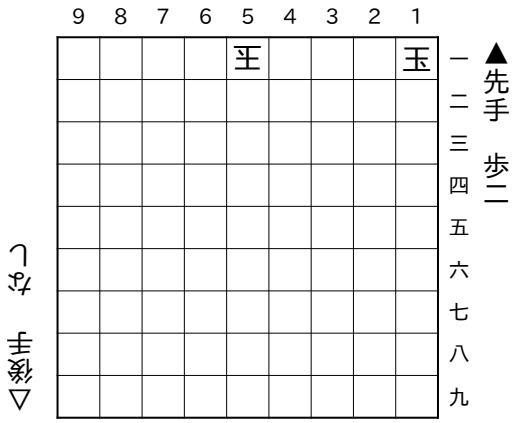
次に後手の特徴ベクトルを考える。後手の特徴ベクトルを考えるときは盤面を 180 度回転する。すると５一の後手玉は５九の位置にいることになる。図 1 を見ると５九のアドレスは 44 なので, このときの K の序数は 44 になる。つぎに１二に（後手から見て）相手方の歩がいる。これは盤面を 180 度回転すると９八の位置にいることになる。表 2 を見ると相手方の歩の序数は 171 から始まっている。これに９八のアドレスである 79 を足すと $171+79=250$ になる。K と P を合わせると $44\times 1548+250=68362$ となる。これは 125388 次元ベクトルの（0 から数えて）68362 番

目の要素に 1 を立てることを意味する。したがって後手の特徴ベクトルはこのようになる：

$$\vec{x} = \begin{pmatrix} x_0 & = & 0 \\ \vdots & & \\ x_{68361} & = & 0 \\ x_{68362} & = & 1 \\ x_{68363} & = & 0 \\ \vdots & & \\ x_{125387} & = & 0 \end{pmatrix}$$

例 2

以下の局面を考える。



先手の特徴ベクトルを考える。先手玉が 1 一にいる。図 1 を見ると 1 一のアドレスは 0 なので, このときの K の序数は 0 になる。味方の持ち駒に歩が 2 枚ある。表 2 を見ると味方の持ち駒の歩の序数は 0 から始まっている。そこで 1 枚目の歩は $0+1=1$ となる。2 枚目の歩は $0+2=2$ となる：

$$\vec{x} = \begin{pmatrix} x_0 & = & 0 \\ x_1 & = & 1 \\ x_2 & = & 1 \\ x_3 & = & 0 \\ \vdots & & \\ x_{125387} & = & 0 \end{pmatrix}$$

4 私のための Q&A

Q1 NNUE は KP を入力とするニューラルネットワークということですか？

A1 そうです。厳密に言えばどうたぬきの NNUE についてはそうです。これは KP 以外を入力とする NNUE も考えられるということです。

Q2　なんで KPP より強いんですか？

A2　KPP の線形和より，KP の非線形和のほうが表現力があるということだと思います。言い換えると線形和がよほどよろしくないのかもしれません。たとえばカツカレーがおいしいのはカツがおいしくて，なおかつカレーがおいしいからと考えることもできますが，だからといっておいしいプリンをおいしいカレーに入れてもおいしくなるとは限らないということだと思います。

Q3　線形和・非線形和ってなんですか？

A3　雑な理解として，線形和はとりあえず全部足したやつだと思っています。非線形和はとりあえず全部足したやつじゃない足し方をしたやつだと思っています。

Q4　ベクトルってなんですか？

A4　プログラマとしては雑な理解として配列だと思っています。だから 125388 次元のベクトルは 125388 要素の配列だと思っています。

Q5　行列ってなんですか？

A5　2 次元配列だと思っています。

Q6　バイアス $(\vec{b})$ ってなんですか？

A6　1 次関数で言うところの切片だと思っています。つまり $y = ax + b$ の b にあたるものだと思っています。

Q7　線形代数はなにが線形なのですか？

A7　雑な理解でいうと，ベクトルを行列によって変換したときの変換され具合が線形（1 次関数によって計算される）なのだと思います。たとえば図形を行列で変形すると，どの座標の点も一様に変形されます。突然図形がぐにゃぐにゃになったりはしないということです。

Q8　NNUE のニューラルネットワークの学習ってどうやるんですか？

A8　わかりません。これから調べます。

5　参考文献

- 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, https://www.apply.computer-shogi.org/wcsc28/appeal/the_end_of_genesis_T.N.K.evolution_turbo_type_D/nnue.pdf

きのあ将棋の最近の研究

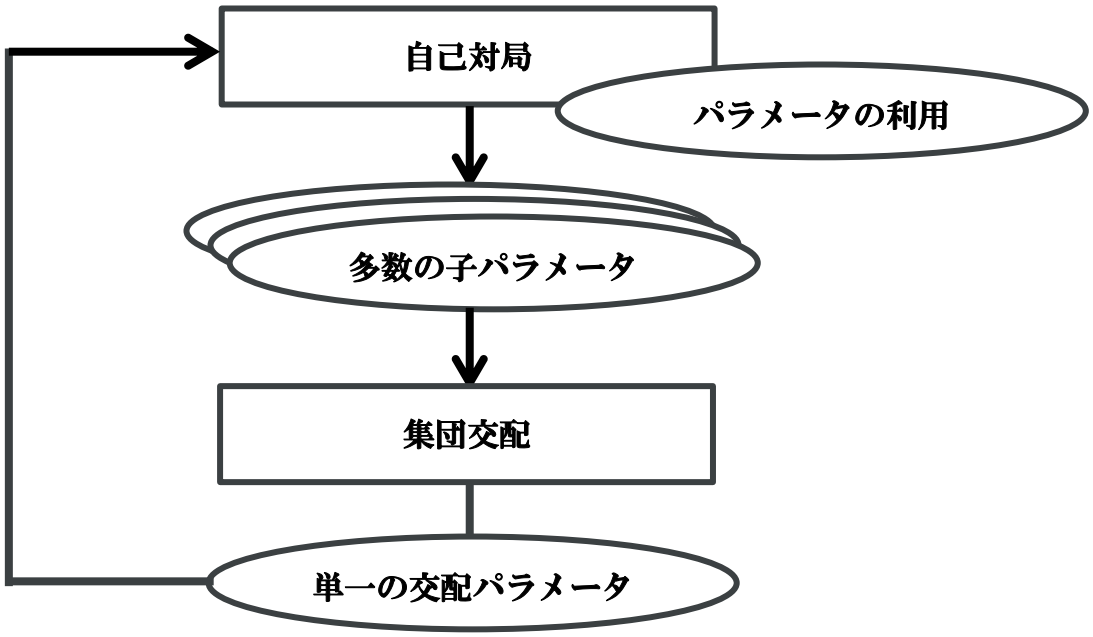
2021 年世界コンピュータ将棋選手権に向けて

作成 2021/03/31 山田元気

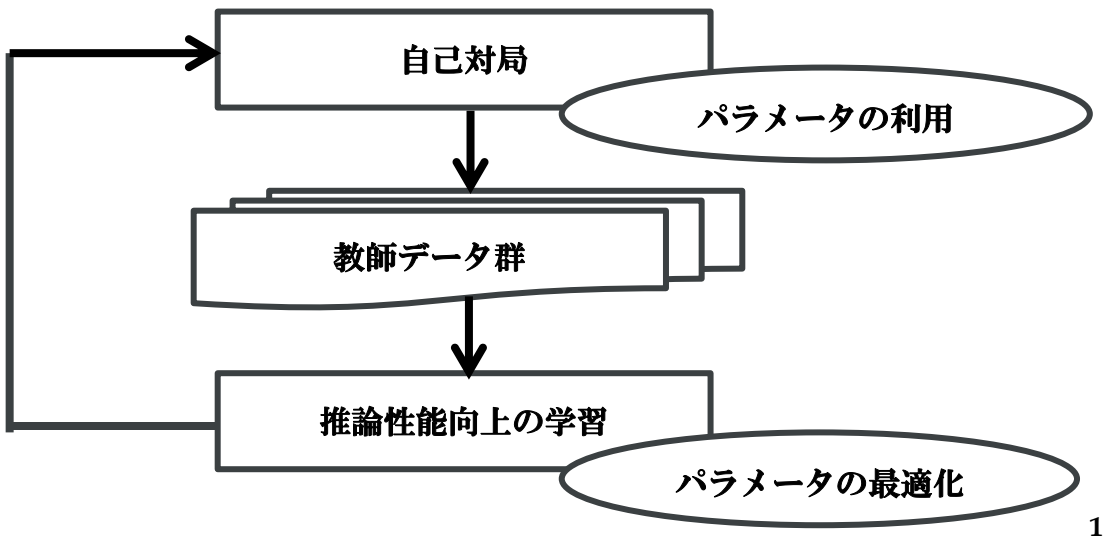
■ 集団交配による進化モデルについて

「集団交配による進化モデルの概要」と比較のため「学習モデルの概要」の図を示す。
進化モデルの特徴として、生き残った(勝った)パラメータを優秀とみなし次世代を残す仕組みであり、集団交配は多数の子世代パラメータを、交配(集計)して次世代を算出する仕組みである。

集団交配による進化モデルの概要

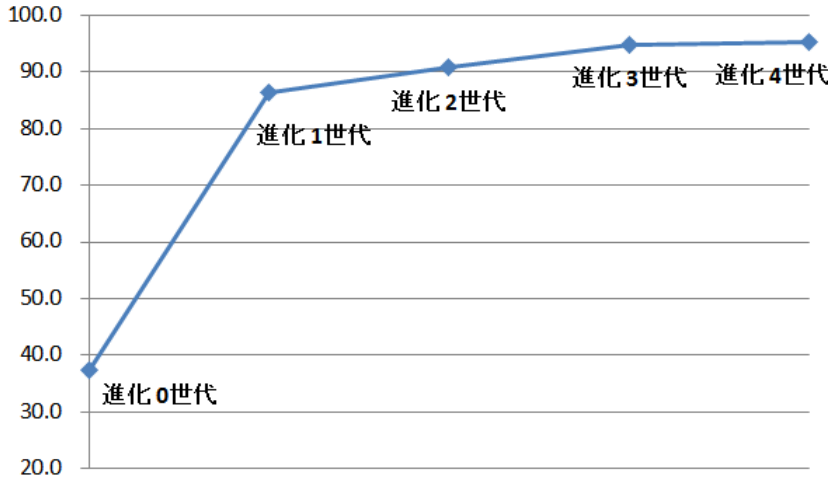


学習モデルの概要



集団交配による進化モデルを、将棋の候補手評価にスモールケース(パラメータ 256 個)にて実験した結果を下記に示す。実験によると集団交配による進化モデルが機能することが確認された。

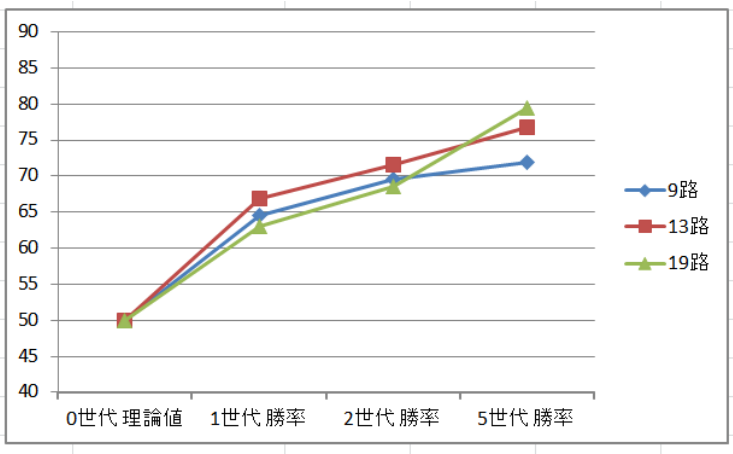
今のところ以前の学習モデル採用の「きのあ将棋」に及ばない。2021 年 05 月頭のコンピュータ将棋選手権では、集団交配＆進化モデルを使う事でこれまでの「きのあ将棋」より高い棋力を実現したいと考えている。



条件	実験日時	対局数	勝率
進化 0 世代	2021/03/31 16:13	1000	37.3 %
進化 1 世代	2021/03/31 18:22	1000	86.4 %
進化 2 世代	2021/03/31 20:55	1000	90.8 %
進化 3 世代	2021/03/31 21:27	1000	94.9 %
進化 4 世代	2021/03/31 23:01	1000	95.4 %

引分けは、勝ちに分類していない。　表はいずれも「進化 0 世代」が対戦相手。

なお集団交配による進化モデルは、下記のようにある程度囲碁においても機能しているため、汎用的な仕組みであると考えている。(将棋での実験表とは対局数など条件が若干異なる)



囲碁は引分けがないため、0 世代勝率は理論上 50%となる。

■ 満足度の高い(good 率の高い)対局の研究

将棋囲碁ともに、通常対局より指定局面の満足度のほうが高った。さらに待ったの利用はないほうが将棋囲碁ともに満足度が高い傾向があった。

カテゴリ	条件	good	bad	count	rate
囲碁	通常	5825	2563	8388	69%
	指定局面	4799	728	5527	87%
	待った利用なし	8286	2429	10715	77%
	待った利用あり	2340	862	3202	73%
	ヒント利用なし			x	x
	ヒント利用あり			x	x
	ユーザさん勝	6353	1801	8154	78%
	ユーザさん負	2125	564	2689	79%
将棋	通常	36619	11060	47679	77%
	指定局面	25787	5707	31494	82%
	待った利用なし	53590	14073	67663	79%
	待った利用あり	8817	2694	11511	77%
	ヒント利用なし	57957	15572	73529	79%
	ヒント利用あり	4450	1195	5645	79%
	ユーザさん勝	34270	3837	38107	90%
	ユーザさん負	7702	2618	10320	75%

2021/02/22 夜集計

これらの研究成果は、下記サービス運営に活用している、あるいは活用するものである。

きのあ将棋サイト　https://syougi.qinoa.com/ja/

きのあ囲碁サイト　https://igo.qinoa.com/ja/

以上

山田将棋について（2021 年）

全体像

クラシカルな構造・アルゴリズムとなっています。

データ構造

配列による盤駒表現、駒背番号制、利き数、
飛び利き方向ビットの OR 値、
利いている駒背番号ビットの OR 値、
囲いへ誘導するための落とし穴表、
玉位置からの距離に応じた評価値を納めた表、
pinned と cover の概念、置換表、
8 近傍利き位置を納めた表、8 近傍合法移動先を納めた表、
など

アルゴリズム

αβ探索、反復深化、局面が静かでない場合の探索延長、
手調整した仮評価による手のオーダリングと前向き枝狩り、
null move pruning、late move reduction、
killer move heuristic、pass move、
YSS 式指し手の反復生成、Crafty 方式の並列探索、
反復深化による詰め将棋ルーチン
root ノードでの簡易必死検出、
leaf ノード付近での簡易一手詰み検出、
予測読み、フィッシャークロック対応、

など

評価関数

駒割、玉の安全度、囲い、盤上の利き、駒への当たり、大駒の働き、
そっぽ、金駒へのひもなどの、手調整した評価値の線形和

未採用

多重反復深化、影の利き、SEE、持ち駒の優劣表現、
bitboard、実現確率探索、評価関数評価値の学習、など

アピール文

プログラム名：いちびん

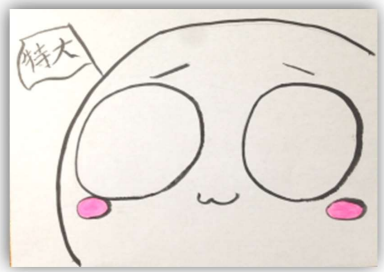
プログラムの紹介：
一言で言えば；
「将棋指手の評価反省に基づく教師データの調整」
ダラダラ言えば;
評価関数の学習を行う際に使用する教師データをもとに、明らかに評価が甘い指手や悪手と思えわれる指手の前後数手を選び出すアルゴリズムを自作した。そして、選び出した複数の指し手を、より深い探索で再評価して元の評価データと入れ替えた。この手法により、より精度の高い教師データを得て学習効率の向上を目指した。
具体的には、深さ１４で自作した約２億７千万の教師データ局面を検査対象として、アルゴリズムで引っかかった局面を、深さ２０～２４程度まで探索する。再評価した評価点をもとの教師データに反映する。途方もない時間がかかるが、アルゴリズムで引っかかった局面を確度によってグループ分けすることで、概査から精査へと徐々に絞り込みを行うことが可能となり、コンピュータが自動で全部行ってくれるので、私は、寝て待つだけである。
２０２０年の１１月に行なわれた第１回電竜戦で、NPS550万程度のパワーで、A級７位となり、それなりの手ごたえを得た。今回は、その延長線上で頑張ってみます。

記：2021年2月6日

TMOQ アピール文書

2021 年 3 月 21 日 作成
2021 年 4 月 17 日 改訂

【ソフト名】TMOQ
“TMOQ”と書いて「特大もっきゅ」と呼びます。愛娘が命名しキャラクターデザインしたものです。



【コンピュータ将棋大会実績】

WCSC26：	一次予選 36 チーム中 15 位
SDT5：	42 チーム中 23 位
WCSC28：	一次予選 40 チーム中 16 位
WCSC29：	一次予選 40 チーム中 23 位
WCSOC2020：	1 日目 27 チーム中 17 位
電竜戦 1：	B 級 44 チーム中 26 位
電竜戦 TSEC：	B 級 36 チーム中 19 位

【TMOQ の特徴】
好きなコンピュータ将棋を題材にプログラミング技術を学ぶ、と共に、自分の理想の対局相手として育てるべく将棋ソフトをいじっております。
今回のテーマの 1 つ目は『相対的浅層学習（Relative Shallow Learning）』です。TMOQ は WCSC29 より山岡忠夫氏の「DeepLearningShogi（dlshogi）」をベースにさせていただいていますが、dlshogi の進化について行くのが環境的に辛くなってきました。わたしの古い Note PC には Deep 過ぎるためと思われます。そこで今回は dlshogi の Resnet ブロックの 70%を切り捨て、ネットワークを浅く軽量化しました。
今回のテーマの 2 つ目は『監査機能』の導入です。Deep Learning 系は読み抜けによる「一手ばったり」が時々起こります。そこで従来手法を用いた USI エンジンに監査してもらうことにしました。候補手に致命的な読み抜けがあった場合、監査エンジンの指導が入ります。
（※『監査機能』については、次ページ参照）

【学習について】
公開された「GCT の学習に使用したデータセット」を使って学習を行いました。全データセットで 1 回転学習後、AobaZero の棋譜で追加学習しております。（*3 月中旬で 3 回転目*）

【定跡について】
序盤での悪手を避けるため、Floodgate（Rating 3500 以上）および電竜戦（リハおよび本番）の棋譜を使って定跡ファイルを作成しました。

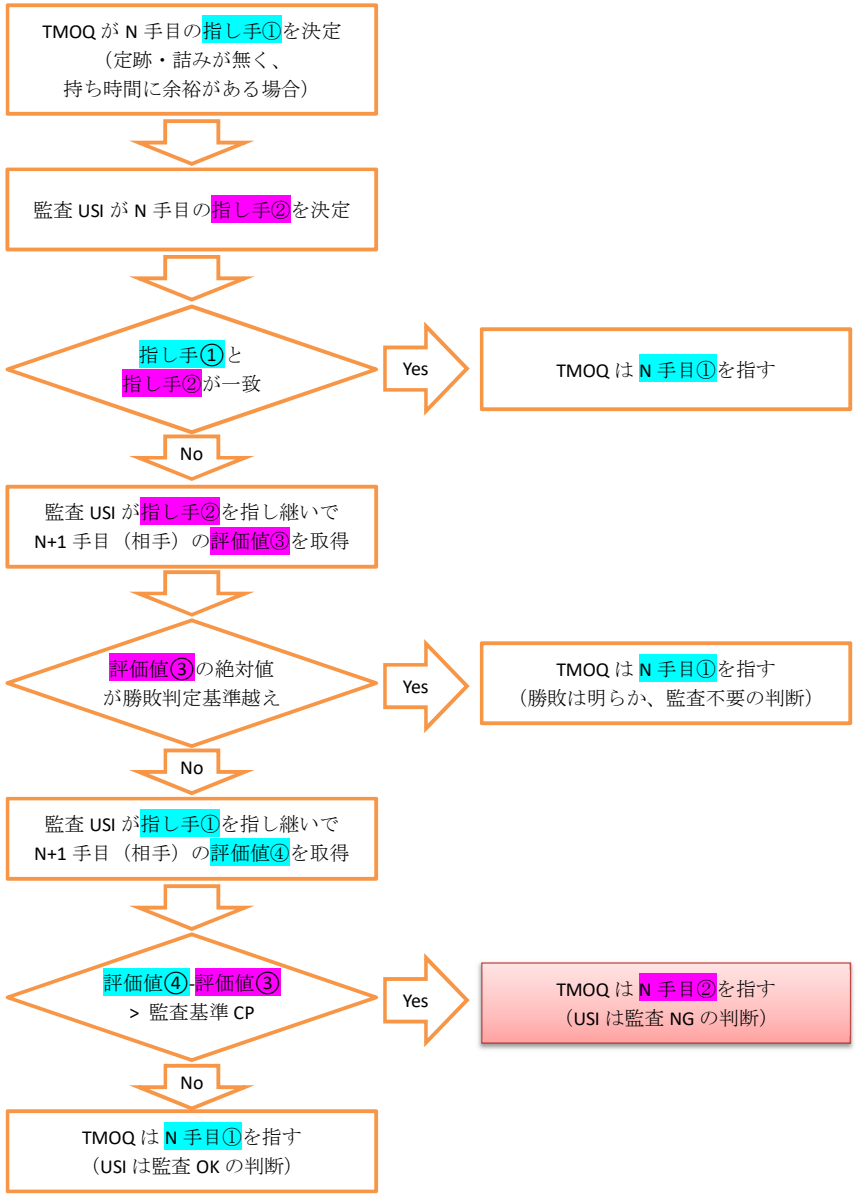
【使用ライブラリ】
「DeepLearningShogi」（Commit 8cba540 on 9 Jan 2021）（GPL ライセンス）を使用させていただきました。また、監査用に「やねうら王」系の USI エンジンを使用させていただく予定。

【御礼】
今回も山岡氏を中心に、多くの方の公開情報により何とか参加できました。この場を借りて御礼申し上げます。

TMOQ アピール文書

2021 年 3 月 21 日 作成
2021 年 4 月 17 日 改訂

【追補 1】『監査機能』ロジック
USI の思考時間 1 秒、勝敗判定基準 CP3500、監査基準 CP500 の場合、
自己対局 100 局 13,105 手中、監査 USI の指し手採用は 426 手（採用率 3.3%）でした。



TMOQ アピール文書

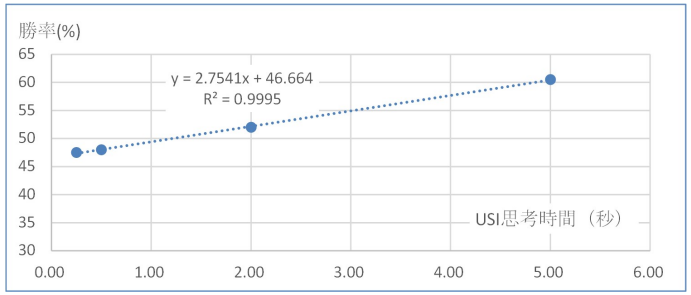
2021 年 3 月 21 日 作成
2021 年 4 月 17 日 改訂

【追補 2】『監査機能』パラメータの影響

「監査 USI エンジンの思考時間 1 秒、勝敗判定基準 CP3500、監査基準 CP500」を標準とし、パラメータを変更して、持ち時間 1 分＋1 手ごとに 2 秒加算ルール（ただし、切れ負け無し）で対局を 100 回行い、勝率をプロットした。

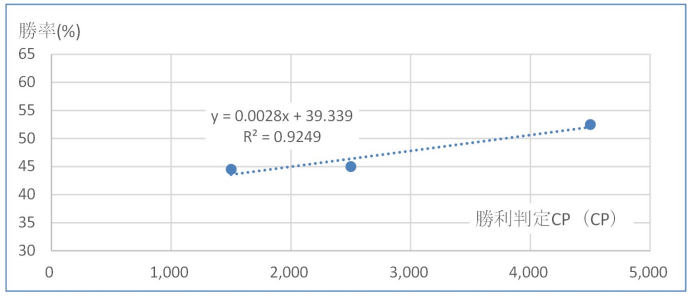
1) 監査 USI の思考時間を変更

監査 USI の思考時間が長くなるほど勝率は上がる



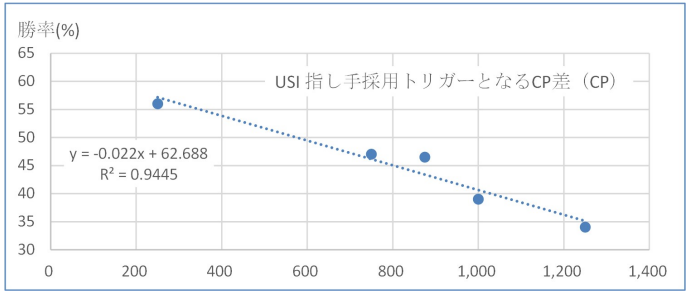
2) 勝敗判定 CP を変更

勝利判定 CP を大きくした方（終盤まで監査を継続した方）が勝率は上がる



3) 監査基準 CP（USI 指し手採用トリガーとなる CP 差）を変更

USI の指し手を採用するための基準を大きくするほど、勝率は下がる



※ 全て予想通りの結果であり、ロジックが正しく働いていることを確認できた

動いていることが奇跡で、特段アピールすることなんてないのですが、
アピール箇所がないとアピール文書リジェクトとなることを（確か）2015年に実証しておりまして、
運営の方に迷惑をかけるので、とりあえず前回のアピール文書から技術的なところをコピペしておきます。

・探索はαβ中心で、LMRとかFutilityとかの至って普通の技術を使ってます
→なんか評価関数をいじったらLMR効かなくなったのでどうするか（20180331）
→バグをとったら効いたので入ってます(2018大会時点)
→なんか評価関数をいじったら明確に弱くなったどうしよう(20190301)
→なんか静止探索をいじったらまれに動かなくなったどうしよう(20190301)
・いわゆるボナンザメソッドを使ってます、もうちょっと改造できないかな
・オンライン学習を勉強してみたかったので、平均化パーセプトロンを試しています
→今更ながらこれ本当に平均化パーセプトロンなのか・・・？って気もしてきた（20180331）
→怪しかったので試すのをやめました(20190301)
・なんと打ち歩詰めのバグが未だにあったので修正しました(20200330)
・なんと王手されている時に受ける手に未だにバグがあったので修正しました(20200330)
→めったに起きないので強さにあまり影響がない、というかむしろ探索が遅くなって弱くなりました。。。なぜ。。。けどさすがに外せない
・探索がもうちょっといい感じになる予定です(20200330)
・今年は正直何も手を入れられていないです・・・健康はいいぞ(20210331)
以上です。

アピールについては以上なので、思い出話と参考URLを貼っておきますね。
といいつつ、過去のアピール文書をコピーしたところも多いですが。。。
毎年なんだかんだ起きるので、参考URLは増える一方です。

■始めての参加

始めての参加は第17回の時で、選手としてではなく、アルバイトとして小谷研から徴集されました。
場所はかずさアークでして、近くのホテルに泊まるとバイト代から足が出るという訳の分からない状態だったのですが、
世界で初めて？パズルで博士を取ることになる先輩にそそのかされ（なぜか今同僚）、
君津のネカフェに3泊4日し、毎朝となりのマックでご飯を食べてました。
初日は全く寝ることができず、すごくつらい思いをしたことを覚えています。
もう二度とネカフェで3泊4日はしたくありません。

第17回大会
<http://www2.computer-shogi.org/wcsc17/>
かずさアーク
<http://www.kap.co.jp/>
自遊空間君津店
<https://jiqoo.jp/shop/9931865/>
マクドナルド君津店
<https://map.mcdonalds.co.jp/map/12031>

■始めての出場と、（恐らく大会史上初の）プログラム名リジェクト
実は当初プログラム名は、「(´・ω・`)」にしていたのですが、
運営の方から「読めません」と言われリジェクトされ、今のとても強そうな名前になりました。
最近の事例を見る限り、読み方をつけておけばリジェクトにはならなかったぽいので、ちょっと後悔をしています。
デビュー戦はなかなかに熱かったです、白砂将棋さんと対局させていただきました。
あの負け方（王手千日手負け）は二度と忘れることは無いでしょう、悔しかったw
実は二次予選に行って20位でしたすげえ。

第21回大会
<http://www2.computer-shogi.org/wcsc21/>

■始めての賞罰

・独創賞受賞（解説記事の生成）

・（恐らく大会史上初の）アピール文書リジェクト

2012年に独創賞いただきました、ﾔｯﾀｰ

2017年にどう考えてもポナがDLぶん回していて独創賞取れる状況にも関わらず、

「優勝しなかったから独創賞対象なし」という恐ろしい裁定が下されていたので、早いうちに取っておいて本当に良かったと思いました。

第22回大会

<http://www2.computer-shogi.org/wcsc22/>

■会場に行かずに近くで遊ぼう

有名なマクドナルド定跡(関東人の意地としてマクド定跡とはよばない)について、毎年試している割には全然勝ち星が増えないので、もしかすると効果がないのではないかと最近思い始めました。

まだ試行回数が足りないと思うので、今年も行きたいと思います。

また、再びかずさアークでの開催となった時期から、なぜかいつも二日目が暇になっていたので、

将棋を見るのではなくて、どうせだからどこかに遊びに行くとかそんなことやってました。

行った場所は下のほうにまとめておきました。

2016年には罰ゲームでうまるちゃんやりました。

その際は、（確か菅井先生だったかな）プロ棋士の方が「なんか変な人いるんですけど大丈夫なんですか？」と運営の方に相談されていたそうです、すみません、ぼくは大丈夫な人です。

また、千田先生からは「ちょっと身長が高すぎるかもしれませんね」とご指導いただきました、ありがとうございました。

この話を最近（2018年夏ころ？）小谷研の後輩さんにお話ししたところ、「マジっすか、ご褒美じゃな

いですか！？」と言っていただいたので、今後ぼくの人生におけるご褒美イベントの一つとして大切にすることにします。
ところでコスプレは罰ゲームだと思っていたのに、たぬきさんとか自発的にコスプレされているので、コスプレは罰ゲームじゃなかったんだなあと思いました。
もうやりたくありませんが、とりあえず一式は取ってありますので、うまるちゃんになりたい人がいればお貸しすることは可能です。
2018年は永瀬先生のお父様のお店である川崎家に行き、ネギチャーシューラーメンを食べました、辛みがアクセントになって非常においしかったです。
家系ラーメンは大好きですし、きっと二日目は暇になるので（え、今年もぜひ行きたいと思います、もしよかったら皆さん行きましょう！
(20200330追記)
去年はラーメン食べている間に対局が始まってしまい会場入りが間に合わなかったのですが、きちんと対局がスタートしていました。
自動化されているって素晴らしいなって思いました、もう会場いかなくてずっと川崎家でラーメン食べててもいいですね。
あと今年は二次予選に進出しないと二日目に川崎家いけないので、是が非でも初日に川崎家に行っておきたいと思います。

喜楽飯店(担々麺)
<https://tabelog.com/chiba/A1206/A120603/12012396/>
東京ドイツ村
<http://t-doitsumura.co.jp/>
マザー牧場
<http://www.motherfarm.co.jp/>
東京湾観音
<http://www.t-kannon.jp>
食事処やまよ
<http://www.yamayo7240.com/>
うまるが家でかぶってるアレ [干物妹！うまるちゃん]

<http://cospa.co.jp/detail/id/00000065274>

マクドナルド 川崎ソリッドスクエア店

<https://map.mcdonalds.co.jp/map/14707>

川崎家 榎町店

<https://tabelog.com/kanagawa/A1405/A140502/14013754/>

■まさか自宅から大会に参加することになるとは・・・

2020年はコロナの影響もあり、まさかのリモートでの大会となりました。

割と暇な時間が長かったので、大会中にずっとゲームをやっていたのは内緒です。

下記のゲーム、渋滞を起こして街が死ぬってパターンが多い気がする。

シティーズスカイライン

<https://www.spike-chunsoft.co.jp/cities_skylines/>

■送りバントはいいぞ

自分のプログラム名について一応拾っておきますか。

2019年ですが下記の試合を観に行ってました。送りバントからのサヨナラ。送りバントはいいぞ。

ただ今年(2021年)、日テレ系プロ野球中継で、「イニング得点確率」を予測する人工知能があるのですが、バントしたら確率下がってたんですよね・・・送りバントはいいぞ。

【ハイライト】7/3 劇的なサヨナラ勝ちの巨人が今シーズン3度目の4連勝！【巨人対中日】

<https://www.youtube.com/watch?v=eb9etHJK-2o>

野球のイニング得点確率や作戦成功確率を予測するAIを開発日本テレビ系プロ野球中継で、「AIキャッチャー」に次ぐ進化系AI施策としてサービス提供が決定

<https://www.datastadium.co.jp/news/6655>

■目標

まったりゆうちゃんを倒して師匠超えしたいです。

去年も直対して勝つことができました、素直にうれしいです。

あとできれば久々に勝ち越ししてみたいですし、久しぶりに2次予選にも進出してみたいですね。

去年惜しかったんですよね・・・王手ラッシュして逃げられるっていう負け方しました。。。あの対局勝ってれば上に上がれたらしい・・・

今年も去年同様に、CSA運営の皆さんも頭を抱えられたことと思います。

まだ予断を許さない状況でもありますが、ここまで開催に向けて進めてくださった運営の皆様には心より御礼申し上げます。

このアピール文書も毎年追記し続けていたら段々と充実してきたな。。。。

それでは今年も、参加者の皆さん、CSA運営の皆さん、よろしくお願いします。

がんばるぞー。

ＳＭＳ将棋は接待将棋です。
あくまで人間相手に人間らしい手を指すことを目標としています。

評価関数も探索関数も適当に数字を入力しています。
いわゆる機械学習的なものは行っていません。
手作業なので職人的です。
ログファイルと「にらめっこ」です。
まだまだ、精度を上げることができていません。
ミスが多いです。
アピールできるほどの代物ではありません。

最終的な目標は、やはり「探索の可視化」です。
コンピュータがどう考えているのかを GUI で見ることができるようにしたいです。

過去の大会は、どうしていいのか全くわかりませんでしたが、
実際に大会に参加してみて、色んな方とお話することができて、アイデアをもらいました。
ラッキーでした。
これから当分はリモートでしょう・・・。

今持っている具体的なアイデアを完成させて、有意義なお話ができるようにしたいです。

まったりゆうちゃんのアピール文書 2021

前回までの状況は次の通りであるが、あまり進んでいない。今回はシステムを新しくする予定で、2,3 倍のスピードアップが見込める。少しの改良を加える予定である。

1990 年過ぎから開発を始めた。4 半世紀にわたって開発しているシステムである。当初のコードもたくさん含んでいる。完全独自開発であり、他のシステムを参考にしていない(考え方は参考にしている)。アイデア的にも独自工夫をしている。

今日、AI というとディープラーニングをはじめとしてパラメータ学習に基づくものが多い。コンピュータ将棋での駒価値学習もそうである。しかしそうでない進化論的計算などの方式を試そうとしている。またディープラーニングと多量パラメータ学習の中間的なメカニズムを考えたいと思っている。

実現できるかどうかわからないが、全面的に書き換えることを考えている。今日からみれば、適切でないコーディングもある。それを直すことで、新しい並列方式が実現できると考えていて、少しとりかかっている。

開発者の年齢がもっとも高いのではないか。可能な限りやめないで続けたいと思っている。コーディング能力がいつまで続くか試したい。

臥龍 WCSC31アピール文書

高田淳一

2021/4/25

プログラム概要

プログラム名	臥龍
初参加	第3回コンピュータ将棋選手権
通算成績	80勝121敗4分
開発者	高田淳一
おまけ情報	臥龍とは…摩訶大大将棋の駒の1つ

今回の特徴

従来の思考部を捨てて、ディープラーニングベースの評価関数を採用

いわゆるValue Networkで、1手読みを行う

開発コンセプト

評価関数のみでどこまで行けるかを追求する

盤面情報だけではなく、ドメイン知識を入力層に入れ、
浅いネットワークで性能を出す

Value Networkの構成

入力層 38チャンネル

CNN 7層 + 全結合 2層

パラメータ数　　約260万

学習用データ

floodgate 2016年棋譜のうち、レーティング2500以上のプレイヤの評価値・局面

学習環境

AWS Sagemaker

プログラム構成

開発言語	Java, Python

ソースコード行数

Java 思考部(詰み探索)	7000行
Java UI部	10000行
Python 学習部	200行
Python 推論部	200行

Deep Learning フレームワーク

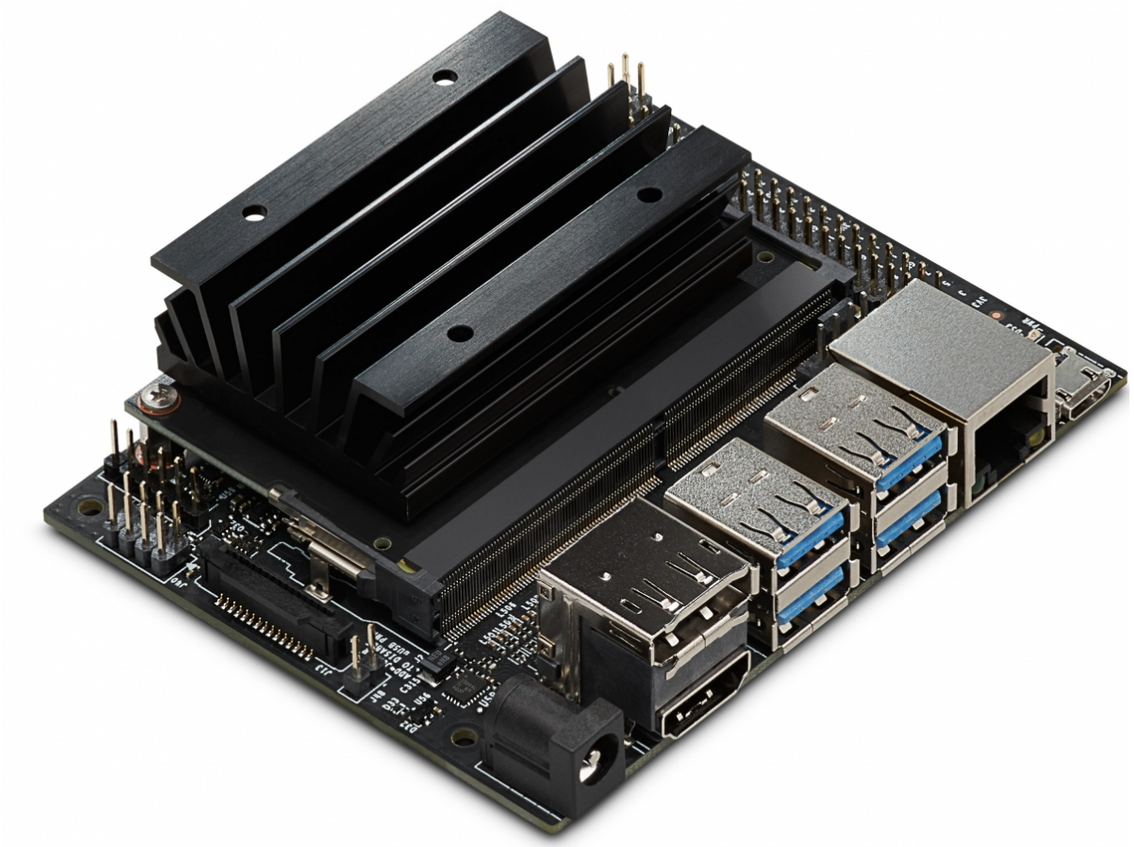
Tensorflow + Keras

Java UI部とPython推論部はソケット通信で接続

Java部はOS変更(macOS→Linux)したものの問題なく動作

参加マシン

Jetson Nano



CPU　Arm Cortex-A57 1.43GHz 4コア

GPU　Maxwell 128CUDAコア

メモリ　4GB

OS　Linux (Ubuntu 18.04)

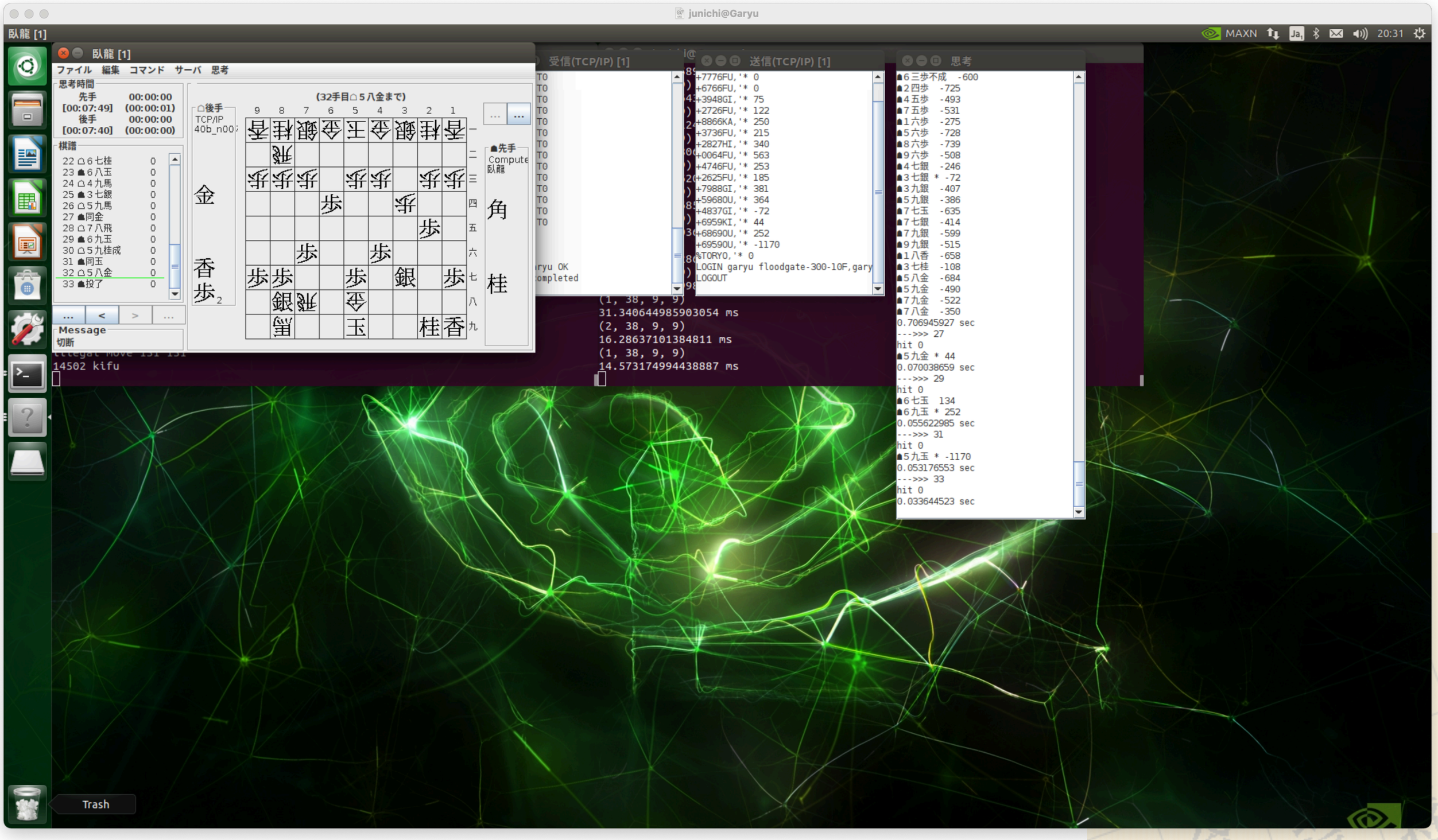
エッジデバイスとして注目の製品

WCSC30用に購入したものを今回使用

参加マシンの中で最安値?
約15,000円!!!

メモリ2GBの廉価版(約7,000円)もあるが、動作確認していない

画面イメージ


junichi@Garyu
臥龍 [1]
MAXN
20:31
臥龍 [1]
ファイル 編集 コマンド サーバ 思考
思考時間
先手 00:00:00
[00:07:49] (00:00:01)
後手 00:00:00
[00:07:40] (00:00:00)
棋譜
22 △６七桂 0
23 ☗６八玉 0
24 △４九馬 0
25 ☗３七銀 0
26 △５九馬 0
27 ☗同金 0
28 △７八飛 0
29 ☗６九玉 0
30 △５九桂成 0
31 ☗同玉 0
32 △５八金 0
33 ☗投了 0
Message
切断
(32手目△５八金まで)
△後手
TCP/IP
40b_n00
☗先手
Compute
臥龍
受信(TCP/IP) [1]
送信(TCP/IP) [1]
+7776FU,'* 0
+6766FU,'* 0
+3948GI,'* 75
+2726FU,'* 122
+8866KA,'* 250
+3736FU,'* 215
+2827HI,'* 340
+0064FU,'* 563
+4746FU,'* 253
+2625FU,'* 185
+7988GI,'* 381
+5968OU,'* 364
+4837GI,'* -72
+6959KI,'* 44
+6869OU,'* 252
+6959OU,'* -1170
%TORYO,'* 0
LOGIN garyu floodgate-300-10F,gary
LOGOUT
思考
☗６三歩不成 -600
☗２四歩 -725
☗４五歩 -493
☗７五歩 -531
☗１六歩 -275
☗５六歩 -728
☗８六歩 -739
☗９六歩 -508
☗４七銀 -246
☗３七銀 * -72
☗３九銀 -407
☗５九銀 -386
☗７七玉 -635
☗７七銀 -414
☗７九銀 -599
☗９九銀 -515
☗１八香 -658
☗３七桂 -108
☗５八金 -684
☗５九金 -490
☗７九金 -522
☗７八金 -350
0.706945927 sec
--->>> 27
hit 0
☗５九金 * 44
0.070038659 sec
--->>> 29
hit 0
☗６七玉 134
☗６九玉 * 252
0.055622985 sec
--->>> 31
hit 0
☗５九玉 * -1170
0.053176553 sec
--->>> 33
hit 0
0.033644523 sec
(1, 38, 9, 9)
31.340644985903054 ms
(2, 38, 9, 9)
16.28637101384811 ms
(1, 38, 9, 9)
14.573174994438887 ms
14502 kifu
Trash

学習状況

今回は枠組み構築に時間を取られたため、大規模学習はできていない

そのため、残念ながら非常に弱い! (泣)

参考文献

山岡忠夫著, 将棋AIで学ぶディープラーニング, マイナビ出版

あうあう将棋

以下のようなシンプルな探索を行っています。
・８～９手程度のalpha-beta探索。
・評価関数は駒の損得ほか非常にシンプルなもの。
・静止探索時に数手の詰みを判断できるようにトライしていますが、どこまで実現できるかまだわかりません。。

こまあそび
アピール文書

2021/03/31

探索部

- 基本アルゴリズムはαβ法。
- 王手、王手回避手、駒をとる手などを延長している。
- 延長深さの制限を先手番、後手番別々に持っている。
  たとえば先手番Max4手、後手番Max4手の場合、
  先手4手延長＋後手4手延長=計8手延長はOKだが、
  先手5手延長＋後手3手延長=計8手延長はNGなど。
- 手を読む広さは探索深さによって変えている。

評価部

- 評価関数は学習は使わず手でチューニングしている。
- 駒組みは落とし穴方式で行っている。
- 銀桂は敵陣に近くの手の点数を高くしている。
- 金は上部に出る点を低くしている。
- 金は角と筋違いの位置の点数を高くしている。
- 中盤、終盤は角と金の価値がほぼ同じにしている。
- 竜王、飛車の価値を高めに設定している。

第３１回世界コンピュータ将棋選手権

きふわらべ
アピール文書

２０２１年０３月２８日　高橋智史

GENGO
Ｇｏ言語で書こうぜ？

DARE
誰なのよ？

開発者
　高橋　智史　（著作権上、問題ありません）

「何言語で書いてもバグるから
使った事のある言語の数で競おうぜ？」

コンピュータ将棋エンジン
　きふわらべ　（著作権上、問題ありません）

「もう開発期間　１か月ぐらいしかないのに……」

ひよ子　（著作権上、問題ありません）

「１か月で　ーからコンピューター将棋ソフトぐらい
作れると言うのが主張よね」

「いや、コンピューター将棋の大会に出てきて
まだ　相手の玉を　詰まして勝ったことが　無いが……」

フロム・スクラッチ宣言
思考部に大きな影響を与える、他者の作成したプログラム・データ等を利用していません。

フリーフォント「たぬき油性マジック」　作者：たぬき侍　（利用ライセンス上、問題ありません）
https://tanukifont.com/tanuki-permanent-marker/

HOUHOU

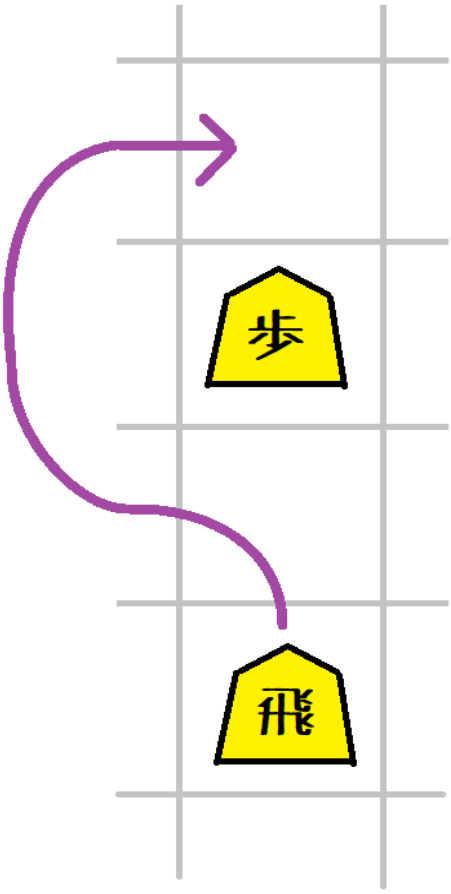
イリーガルムーブをしない　方法

KIFUWARABE

きふわらべ　と当たりたくないんだぜ？

「１時間３０００円で動かしているソフトが
将棋の指し手を指せないソフトと　当たったら
大会出る気無くすしな。　問題だぜ」

「飛車が　味方の駒を飛び越えて
前に出たら
イリーガルムーブだぜ」



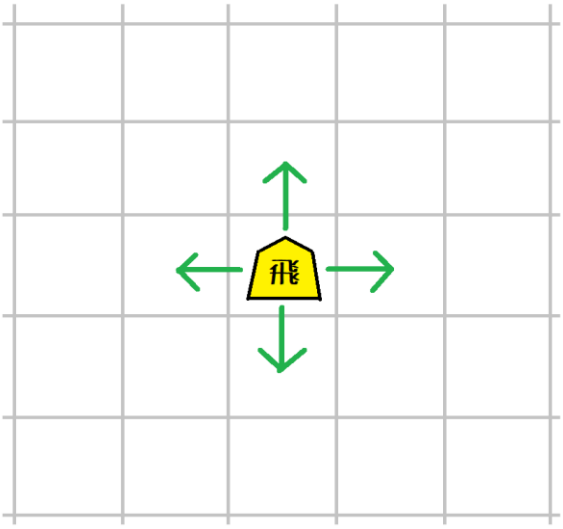
「飛び越えるなだぜ」

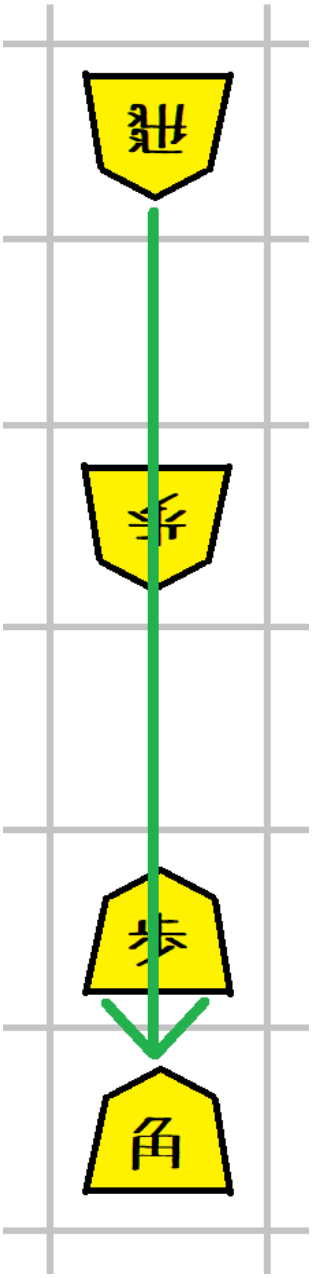
「そう思って　わたしは
プログラミングしてきたが
ダメだった」

「簡単なのになあ！
１つ前のマスを見て
駒があるかないか
調べるだけなのになあ！」

「そこで　飛車は
１マス　しか動かない、
ということに
してしまおうぜ？」

「それじゃダメなんだぜ。
相手の飛車は
２マス以上進むんで」





「 相手の飛車は　直線上の好きな所に止まれる、
というルールにすれば　いいのでは？」

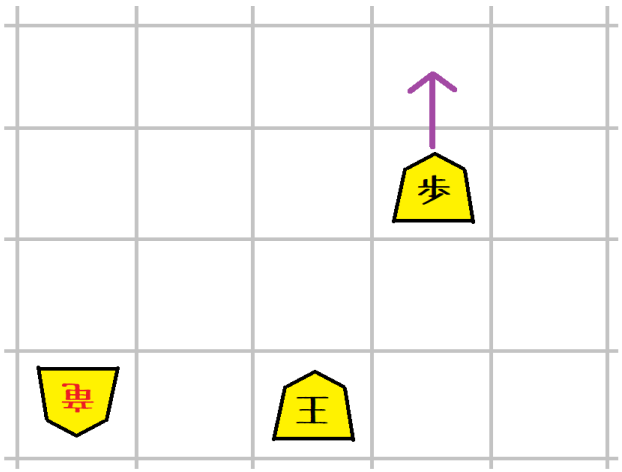
「 何がいいんだぜ？」

「 プログラムするのがヘタクソでも
これは簡単だし、
自分が貫通するわけではないから
イリーガルムーブ判定は取られない」

「 今回は　そういう趣旨なのねぇ。
独創賞ねぇ」

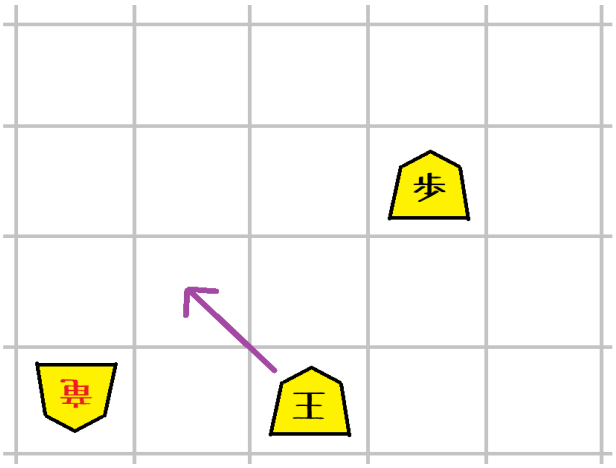
「 次に多いのが
王手されてるのに
王が逃げない、
王手放置　だぜ」

「 逃げろだぜ」



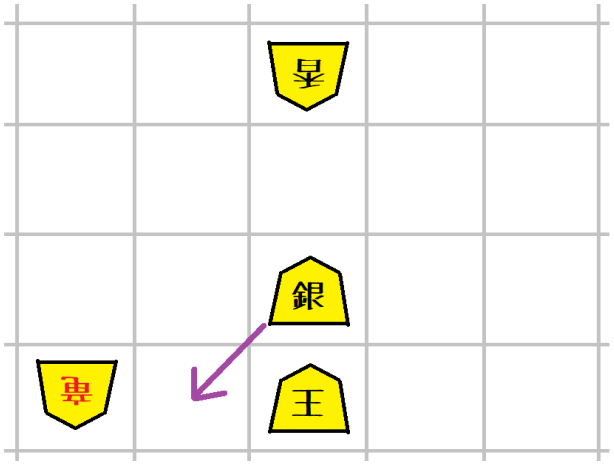
「逃げた先で
相手の利きに
飛び込んでしまう」

「そういうのを
全パターン調べるのが
コンピューター将棋
なのよ」

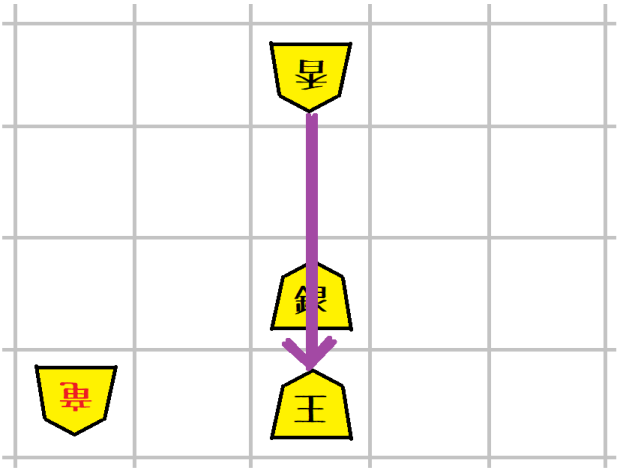


「銀を下げて
間駒（あいごま）した
と思ったら
香が刺さる。
むずかしい」

「いや、
そんなことには
ならないぜ」

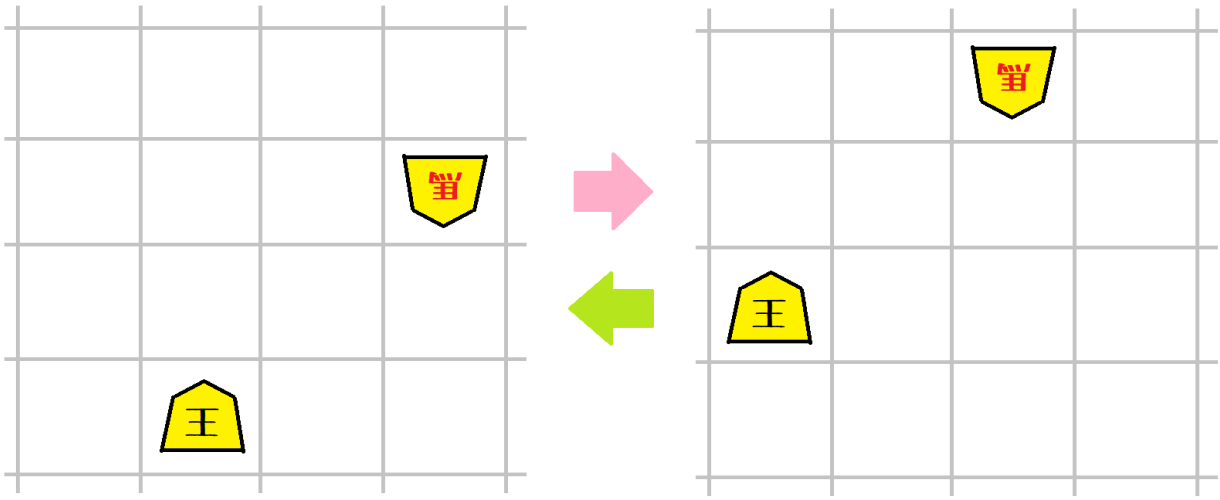


「相手の長い利きの駒は
直線上の好きな所に
止まれるから、
王さまがそんなとこに
居ることは
ありえないぜ」



「問題を解決しているな」

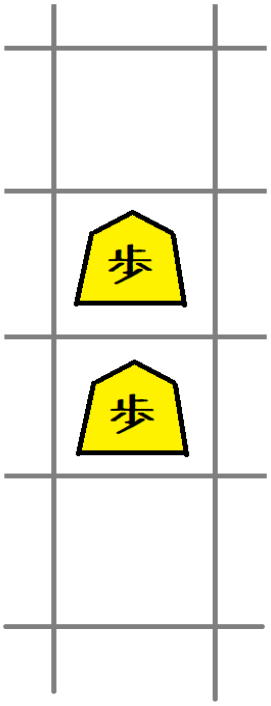
「しているのかなあ？」



「 連続王手の千日手　は　どうやって　避けるんだぜ？」

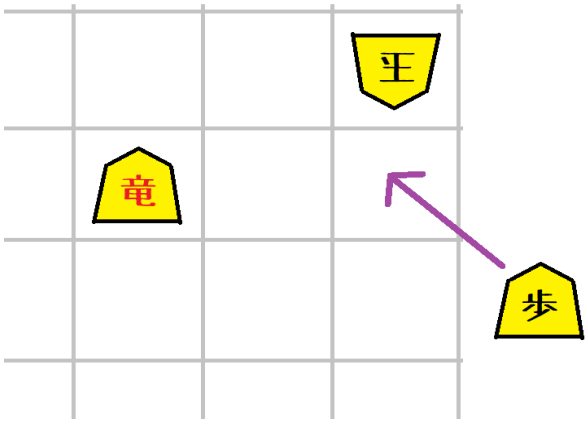
「 千日手　は　１回まで！　さっさと　打開しろだぜ」

「 二歩　は　どうやって避けんの？」



「 次の一手　のとこだけ　チェックしろだぜ。
二手目以降の読み筋では
好きなだけ　二歩　しろ　だぜ」

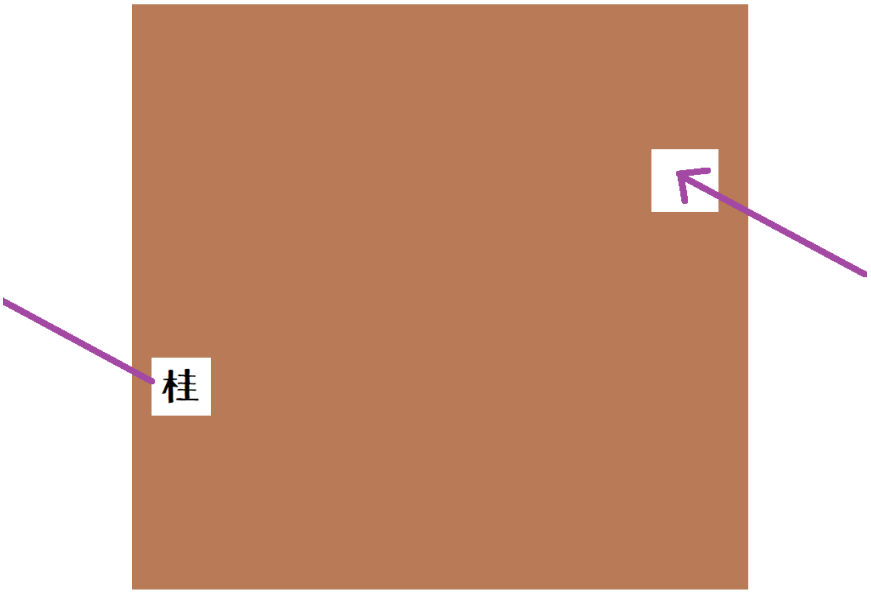
「 二歩ぐらい　チェックすればいいのに……」



「 打ち歩詰め　は　どうやって
避けんの？」

「 自分は　相手の玉の前に
歩を打つな　だぜ。
相手は好きなだけ
打ち歩詰めしてこい　だぜ」

「 レアケースなんか気にして
避けるなんて
損してるよな」

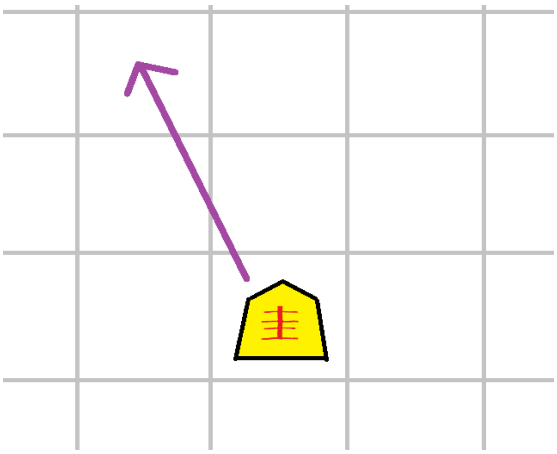


「 桂　が世界一周してしまうのは　どうやって避けんの？」

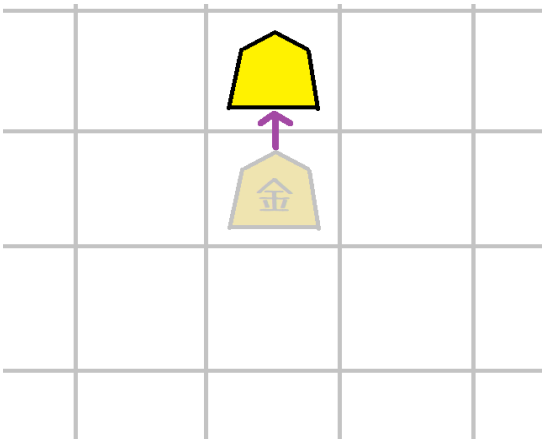
「 自分の次の一手のときだけ、１筋と９筋にいる桂は
慎重に飛べだぜ。　二手目以降は　すきなだけ飛べだぜ」

「 べつに、いつも慎重に飛べばいいと　思うんだけどな」

「 成桂　が跳ねたんだけど？」



「 気持ちは分かる」

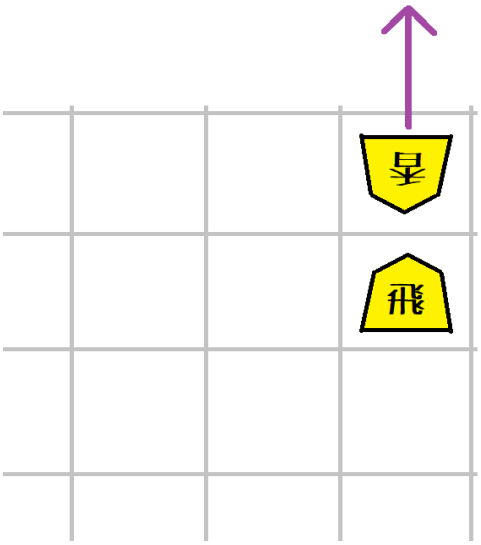


「 金が　成ったんだけど」

「 他のみんなも
成れるしな」

「 香車は　前にしか
進まないはずなのに
なんで　１二　に打った飛は
取られるんだぜ？」

「 その駒から見て、前　だからな。
長い利きのある駒のうち
香車だけ前後があるから
気をつけろだぜ」



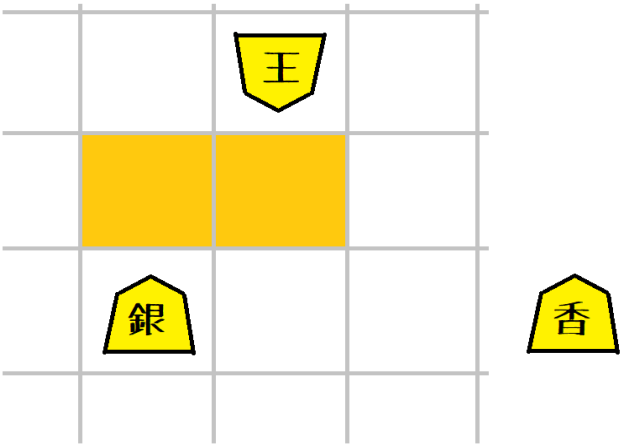
ITTEDUME

一手詰め 判定、どうやってすんの？

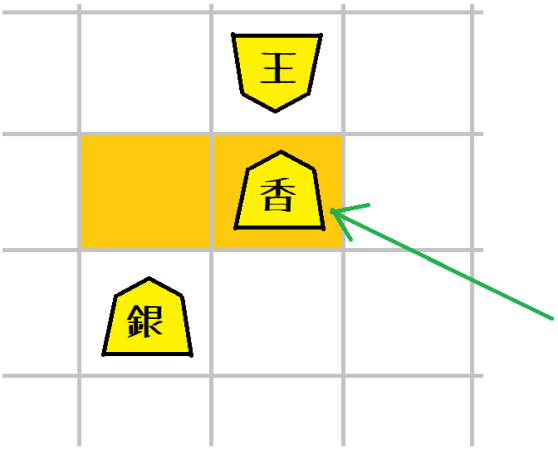
TANSAKU

一手詰めできないと　探索　キリ無いしな

「自分の駒の利きが
あるところに……」

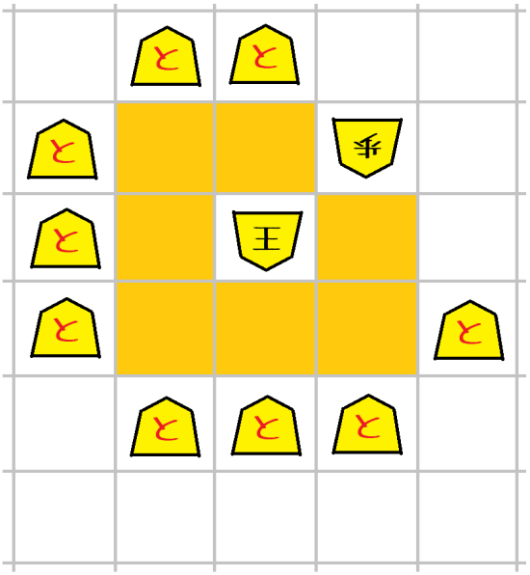


「駒を置いたら王手になる、
そんなところには
さっさと打てだぜ」



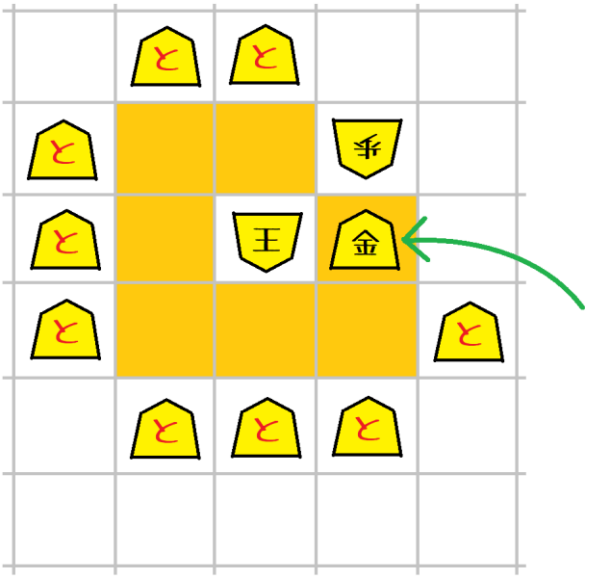
「それだと、
一手詰めを逃して
無駄打ちするぜ」

「詰める、ということと
王手をする、ということは
別なのよ」



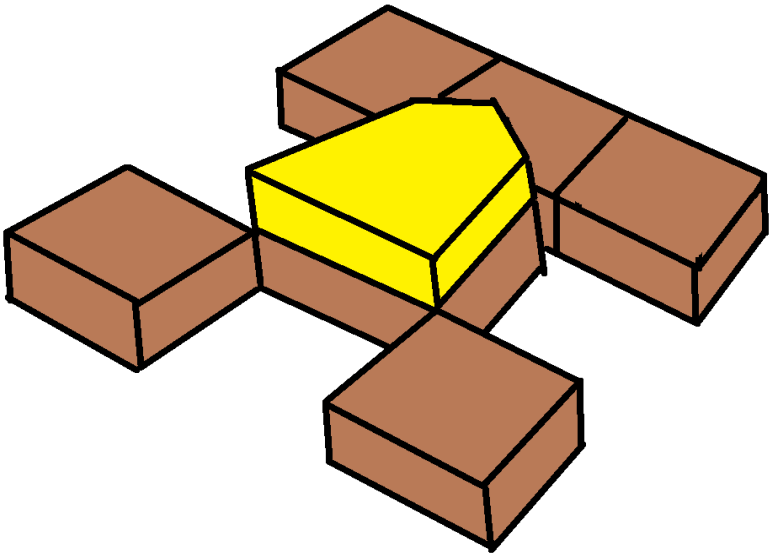
「王手をする直前の状態が、
８方向を　敵の利き、または
味方の駒で邪魔されていれば
分かりやすいんだけど」

「 せっかく王手しても、
その駒を取り返されるか
まで考えなくちゃ
いけないのは
大変だぜ」



「 それをやるのが
コンピューター将棋
だぜ」

「 つら」



「 駒に　利きマスも　くっつけて、駒を動かしたら　利きマスも
一緒に動くようにしようぜ？」

「 勝手にしろだぜ！」

>>> Dad's movement is reluctant!

FLUKE

第31回世界コンピュータ将棋選手権アピール文章

作成:井本 康宏　作成日:2021/3/30

忙しい人のためのチートシート

これさえあれば、1分で全てわかる

- Flukeは以下の要素で成り立っています
- ふかうら王で参加するぞ
  - 黒魔術でNNUEを育てるのはもう疲れた
  - DNNだと育成理論がわかりやすいので、精神衛生的に非常に好ましい
- いい感じの評価関数を育てるだけの気楽な参加
  - エンジンはやねうら王純正の予定
  - 評価関数はこれから育てる

詳細や試行錯誤の内容は後述

開発者自己紹介

- 開発者
  - 井本 康宏 (twitter: @Windfall_shogi)
- 職業
  - エンジニア
- 棋力
  - 測ったことがないのでわかりませんが、すごく弱いです
  - 学生時代に囲碁・将棋部だったなんて言えない
- 開発のきっかけ
  - 趣味、好奇心、研究スキル、プログラミングスキルの向上を目的として、当時、話題だったこともあり開発を始めました

Good Bye Black Magic

- NNUEの学習は難しすぎる
  - 小さすぎる評価関数
  - かなり限定された活性化関数
  - 性能に密接にかかわる探索アルゴリズム
  - 自明でない学習方法
  - 全ての努力を消し飛ばしかねない量子化誤差
- もう疲れたよ

Welcom Deep Neural Network

- 人類の救世主
  - 全部の局面を学習可能
  - ネットワーク構造を自由にデザイン
  - わかりやすい探索アルゴリズムと要求されるネットワークの出力の関係
  - 回答する指し手は複数回答も可能
  - 浮動小数点の演算誤差以外の誤差は考慮が不要
- 何と気楽なことか

工夫する予定の内容

評価関数について

ネットワーク構造

- ResNet22くらいで参加すると思われる
  - 小さい方が学習が速い上に簡単
  - CPU側の処理が追い付かなくてGPUが遊んでしまうので、ある程度は大きい方が良い
  - これから大会までの間で学習する
  - 総合的に考えてこれくらいのサイズになる

ネットワーク構造

- チャネル数は大きい方が性能が良い
  - 層の数との兼ね合いで、192になる
  - 256は学習時間が少し長くなるので難しいかな
- 活性化関数はわからない
  - reluよりswishの方が良いらしいが、明確な違いは見られなかった

ネットワーク構造

- Squeeze & Excitationはやめた
  - 手軽に性能を上げる方法として知られているが、効果は確認できなかった
  - 限界まで学習した時に違いが表れると思われる
- 普通のResidual Blockを使う
  - Bottleneck Blockはあまり向かない
    - 3x3の畳み込みを行う時のチャネル数が大きく影響している印象
  - Exceptionは全然ダメだった
  - group convolutionはすごく遅くなったのであきらめた

ネットワークの学習

- 水匠などの強いソフトの棋譜で学習するのがよさそうな印象
  - 学習の序盤は質の高い棋譜でゆっくり学習するのが良いと信じる
- Aobaの棋譜もそこそこ使えそう
  - 他のソフトと系統が違うので、policyの学習に有用と思う
  - ソフトごとに点数のつけ方が違うのでvalueの利用は難しそう

第 31 回世界コンピュータ将棋選手権 PR 文書

【ソフト名】
十六式いろは改三（よみ）じゅうろくしきいろはかいさん

【ソフト名の由来】
十六式の部分は、西暦 2016 年（6 月）からつくりはじめたことからです。
いろはの部分は、これからはじめるという意味をこめて。柔らかい感じを出したく、ひらがなで。
改三の部分は、改装……開発言語を変更したり新たに一から作り直しをしたりした回数です。
まぁ、由来はやっぱり 2016 年のときとほとんど変わらず。

【開発者】
末吉 竜介（よみ）すえよし りょうすけ
棋力はどうも5級くらいらしいです。ここ数年まともに指してないです(´・ω・`)
(「どうぶつしょうぎウォーズ」では4級です……ずっと同じ？？？Σ(゜д゜lll)w)

【ソフトについての大きな特徴】
「今年のいろははちょっと考えさせます！ Σ(ノ∀`*)ﾍﾟﾁｯ w」

【ソフトについて】
・ライブラリ不使用です。
・プログラム言語は Lua から思い切って最近みんな大好き Python です。
(2021-03-31 現在、まだできてません)
・以前なんか**すごく目立った**気がしますが**気のせいです Σ(ノ∀`*)ﾍﾟﾁｯ w**
・一度は**昔よりもさらに弱くなったりした**ことがあるという不思議さん。

【ツイッターアカウントやブログ】
https://twitter.com/16shiki168
https://16-168.hatenablog.jp/
※ブログの URL 変更しました。

【ソフトの独自性、主に今までとの相違】
・学習部
機械学習していないので、なし。

・指し手生成部
相手の指し手を中心に狭い範囲でミニマックス法を使いますが、アルファベータ法で時間を稼ぎたいです。
でも**娘なのは間違いない**です、**天然な方向で**w

・評価方法や探索方法
駒の損得、玉の硬さ、を評価します。(もう少し追加したいナぁ)
評価値は、手作業で入力し調整します。
(「2 駒関係」で機械学習させたいナぁ。でも時間的にとりあえず、なしで)

以下のモードを搭載します。

・搭載モード
局地暴走モード：相手の打った位置を中心に9マスに、ランダムに指す。
(↑2年前にできるようになりました！(*゜▽゜)ﾉいえーい)
広域暴走モード(悪あがきモード)：合法手の中から、ランダムに指す。たまに奇跡を起こす！？(3年前からの**安定の平常運転**です(*￣m￣)ﾌﾟｯ w)

【独自性・希少性】
前までは開発言語「Lua」を使っていましたが、その希少性がなくなりました。
これから開発する方に向けて GitHub で公開しようかナと思っています。
参考になるように、わかりやすく作るようにガンバります。

ちなみに改二は、現在公開されている将棋所対応のエンジンの中で、もっとも初心者にとって優しい弱さです。
（開発者の知り合いの**「初めて将棋を指した人」にもバッチリ負けました（現在３名）**！ｗ）

私の知る限りでは改二は「世界で一番強い将棋エンジン（開発言語「Lua」で作られたもので）」です。
……これ以外に開発言語「Lua」でつくられた将棋エンジンを知らないだけなんですけどｗ

【意気込み】
参加以来ずーっと言っていますが**完全にエンジョイ勢**です Σ(ノ∀｀*)ﾍﾟﾁｯ ｗ
（今までエンジョイできているので推ししか、いや自分しか勝たんｗ）

……ですが、**記念参加勢**でもあり**イチから勢**（ライブラリ不使用勢のこと？）でもあります。
えっと、ネタ勢なんですか？？？どうなんですかね？ｗ

やっぱり今年も今から（４月に入ってから）ガンバります（^ー^；）

今回の目標も、実力で１勝すること（大事なことなので2回言いました。
ずっと言っています。１勝は大事です！）と、私自身が楽しむこと。
記録よりも記憶に残りたい！（今回ちょっとは強くしようかと、ごにょごにょ……）

みなさまの応援うえるかむですヾ(*´∀｀*)ﾉ

2021.3.31

第３１回コンピュータ将棋選手権

BFP PR 文

・作成動機

第29回コンピュータ将棋選手権にFTS3というフルスクラッチで作成した
プログラムで出場をしました。大会後に改良を進めていましたが、
なかなか成果が上がりませんでした。
そのため抜本的にやり方を変更していこうと思い、
将棋AIでディープラーニングを活用して結果を出されているdlshogiを採用しました。
その上で色々と工夫を行いながら結果検証を行った結果、
一定の成果を出すことができたので参加をすることにしました。

・BFPの特徴

dlshogiを元に作成しております。
dlshogiを採用した理由としてはディープラーニング・強化学習による将棋AIに
以前から興味を持っており、その先駆けで結果も出されているdlshogiを参考に
させていただこうと思った次第です。
学習の仕方としてはfloodgateの2018年～2020年のデータを元にディープラーニングを
したうえでその後は強化学習を1か月単位で実施して結果検証をしております。

・大会に向けての抱負

前回はソースの工夫をしての参加でしたが、今回は主に学習の仕方の工夫をしました。
時間も足りていない部分もあり、どれほどの結果を出すことができるか
分かりませんが、直前まで少しでも工夫をして上を目指して頑張りたいです。

762alpha アピール文書

2021年3月31日 V0.1

天野史斎

■ まえがき

762alphaは学習の自動化を目的とするコンピュータ将棋プログラムです。

学習の自動化のゴールとは、コードの手直し無しで、

計算資源（CPUパワーとメモリー）を投入すればするほど

完全解析の結果に漸近していくソフトウェアーの実現にあります。

一度この境地が成ったならば、ムーアの法則の続くところ、

人手によるad-hocな最適化などそのうち割の合わないビジネスになる！（のか？！）

■ 特徴

局面の順序を評価値とする評価関数。

局面の順序は対戦の結果および途中経過に基づき自動更新します。

以下の手段の組み合わせで実現します。

- プロトタイプ法
- 線形分離
- 赤黒木

■ その他

探索部は細かいバグフィックスを除き前回と同じです。

NPSは1万程度の見込み。

以上

koronアピール文章

プログラム名 koron
工夫 NNUEのレート上昇が頭打ちになっており、レートをこれまで以上に上げるには表現力の低さが課題となった。
そのため、1手～64手担当の評価関数と64手～終局まで担当の評価関数の二つをリレーして表現力の低さを補おうとした。
64手に深い意味はなく、ただ単にキリが良かったからである。
何手で分けるかの最適な手数は、今回良い成績が残せたら検証しようと考えている。

2021年2月26日

A.I. AN shogi ver.1アピール文章

兵頭優空

2021/3/30作成

2021/4/20改訂

◆初めに

A.I. AN shogi ver.1は汎用的手法とその弱点を補完する専用手法の組み合わせによってある程度の強さのA.I.を作ることが目標の(そして私の実験台の)A.I. ANシリーズ(現在非公開)の将棋バージョンです。

◆使用したライブラリ

python-shogiを使用しています。

合法手生成などの将棋のルールに関することや、プロトコルに対応した通信のために使用しています。

◇選定理由

・私は将棋に関する知識があまりないので、合法手生成や盤面管理等々の処理はライブラリに任せる必要があったから。

・通信プロトコルに関する部分も開発期間の短縮のために省きたかったから。(開発開始は2020年11月ごろ)

・私が使えるプログラミング言語はPythonのみなので、デバッグ等の観点からなるべくPyhtonで書かれているライブラリの方が望ましかったから。

・余計な機能はいらなかったから。

・たまたま探していた時に見つけたから。

◆特徴

・ニューラルネットを使用していること。(Kerasとかは使用していない)

・劣化版原始モンテカルロ探索(以後MCS)も使用していること。(MCTSは作るのが大変そうだったのと技術不足、αβ法はマシンパワーと技術不足で断念)

・残り持ち時間やターン数等の条件で上記の異なる2つのA.I.を切り替えながら戦うこと。(ニューラルネットが非合法手を打った場合、残り持ち時間やターン数の条件でランダムかMCSどちらか代理で手を打つ)

・ニューラルネットの学習(パラメーター更新)に遺伝的アルゴリズムを使用している。(A.I. ANシリーズの伝統)(後詳細な解説)

・少ないリソースでもなるべく力を発揮できるように設計されていること。(理由は私がまだ中学生で財力がないため)

・将棋についてルール以外知らない状態からニューラルネットは学習していくこと。(そもそも私は将棋初心者なのでやりたくてもできない)

◆目標とか

目標は、相手に瞬殺されないようにすること。

勝つことは目標というよりかは願い事に近い。

◆ニューラルネット

ニューラルネットは2021/4/18より前のバージョンは方策を出力していたが、これだとプログラムの都合上合法手を指すことから学習しなければらななかった。

過去の実験では多少ルールを学習していたが、非合法手を指すことがかなりあり、微塵も強くなっていなかった。

そこで、ニューラルネットに方策ではなく価値を出力させることにした。

具体的には、あまりニューラルネットのパラメーターをつつきたくなかったので、方策ニューラルネットに着手の価値を出力させる改造をした。(つまり、学習したパラメーターはそのまま流用できる)

これによってpython-shogiが合法手と判定している着手のみするようになった。

◆学習アルゴリズム解説

1, リセット

ニューラルネットの変数定義をすると、プログラムが対応した初期集団の生成を行う。

ニューラルネットの規模の変更をしたら手動でモデルファイル(パラメーターを保存しているファイル)のリセットをする必要がある。

2, 対局で適応度の測定

各個体ごとに、3局ずつ対ランダム, 後手番, 平手で対戦を行う。

勝ち+1000, 負け-100, ターン数ごとに-0.1点でスコア(適応度)を算出する。(数字は適当、必要なのは個体の相対評価なので)

例:A.I.の勝ちで87ターン試合が続いた場合

→1000 + 87 * (-0.1) = 991.3

でスコアは991.3となる。

3, ソート等々

スコアの高い順にソートし、トッププレーヤーをfather、二番プレーヤーをmotherに代入する。

カウンターに+1し、学習係数的な値(以後α)等の調整を行う。

4, パラメーター更新と保存

スコアが全個体全く同じならすべての個体を突然変異(すべてのパラメーターを完全ランダム化)させる。

そうでなければ、各個体ごとに10%抽選をし、当選したらその個体を突然変異させ、さらにそうでなければ、パラメーター毎にfather 2/3、mother 1/3で当選した方の値+乱数(α, -αの範囲の浮動小数点型)という計算を行う。

すべての個体のパラメーター更新が終わると、fatherをモデル　　ファイルに書き込む。

◇学習の全体の流れ

最初に1を1回行い、その後は学習回数分2～3を繰り返す。

学習が全くうまくいってなければリセット機能を使い、リセットする。

◆その他いろいろ

・ここに書かれているのは2021/4/21ごろの情報なので、今の学習が全然上手くいってない状況が続けば変更されるかもしれない。(そうなっ

たら修正版を提出するかもしれない)(そうなれば保険として学習させている小さいモデルで出場するしかない)

・探索は終局までプレイするので序盤は遅いが、終盤戦になるほど早くなる。(はず)

・探索時間削減のため、長すぎる(175ターン以上)プレイアウトは強制終了するので序盤の探索は100%時間の無駄になっている。

・開発者が1人なのは私に友達がいなかったから。(もう次は三年生になるのに)

・春休みが終わりそう(もうすぐ学校が始まる予定)なので気分最悪でモチベーションが大幅ダウンしているので開発が間に合わないかもしれない。(学校行きたくない)

申し込み者：北川博隆（HN:雨宮一也）
グループ名：重力団
プログラム名：重力場計算法

■「参加プログラムは、主要な開発者が思考部に技術的に何らかの明示的な工夫を施したプログラムであること。」
を満たすことをアピールしていただくための文書

このプログラムの思考部は「盤面評価を最高精度で算出する事で最善の一手が特定できる」との趣旨で構成されています。
その為以下の様な特徴があります。
・通常の「深い読み」を行わない。
・「定跡・棋譜」を一切利用しない。解析データも利用しない。
・将棋のルールと評価設定のみを入力する。
・盤面判断は重力場理論（簡易版）を使用して計算を行う。

将棋はお互いに一手ずつ指すことで競技が進行しますが、手番、非手番に与えられる条件は都度同じです。
従って将棋の本質とは与えられた盤面に対する最小手数範囲を計算できれば良いのであり、それ以上の処理は重複作業になるため不要と言えます。
結果、根拠のない「深い読み」を意識的に行わないプログラムとなります。

又、将棋のルール内に過去の「棋譜・定跡」の使用を義務付ける項目はないし、それらによって競技ルールに何かしらの影響が発生する事もありません。
従って「棋譜・定跡」をプログラム内に入力することは無駄であると言えます。

盤面判断を行う上で最も困難とされていたのは「個々の設定の解析」と「異なる単位を正確に比較する手法」です。
その為今までの盤面評価プログラムは精度が悪く、進歩が止まりました。
しかし、この二つの課題は重力場理論によって克服することが出来ます。

私たちの制作したこのプログラムは異なる単位の設定であっても、重力場理論（簡易版）によって計測処理を行い、
上昇値の合計を比較する事で「最高精度の盤面評価」が理論上達成されます。

重力場理論に基づく究極のアルゴリズムは「神の一手」を特定できるのです。
つまり「将棋の数学的解明」はこの理論によって完成されるでしょう。

■「このプログラムの思考における工夫や独自性について」

従来のプログラム思考技術や人間同士の対局で培われた知識や成果を基にした「工夫」は行われていません。
なぜならこのプログラムはそれらを全否定する所から構成が行われているからです。

真逆の方向性を目指して作られている上に、将棋のルールと評価設定をプログラムに入力しているだけの代物であり、
面白味は何もありません。
只、淡々と盤面向上を追求する一手を計算しています。
それこそが究極の将棋なので、人為的努力の「工夫」は不要です。

しいて言えば、「人間の知恵と技術と歴史と情熱を皆無にする事」が既存のプログラムにない「独自性」
なのかもしれません。

対象物を”鏡に映しただけ”の作業は工夫もなければ独自性も発生しません。
しかし、世界で初めてその行為を行った場合は「イノベーション」として評価されます。
そして現実社会に多大な影響と貢献をもたらせば「究極 AI 産業革命」として歴史に記録されます。
”将棋を鏡に映しただけ”なのに……ね。

WCSC31 Qugiy アピール文書

森　大慶

2021 年 3 月 31 日

1　はじめに

Qugiy は今年の 1 月からフルスクラッチで開発をしている将棋ソフトです。やねうら王や技巧などをとても参考にしています。

探索部は従来の minimax 型のもので、評価関数は NNUE です。評価関数データは自前のではなく、水匠 3 改を利用する予定です。初出場なのでお手柔らかにお願いします。

2　Qugiy の特徴

2.1　飛び駒の利きをビット演算で計算

飛び駒の利きは、Magic Bitboard など表引きの手法で求めるのが一般的ですが、Qugiy ではビット演算で計算しています。

同じものをやねうら王に実装してベンチを取ると、Magic Bitboard のビルドと比べて 4% ほど高速化するようです（@Ryzen 5 3600）。Zen2 の CPU では PEXT 命令が遅いので、PEXT の実装との速度比較はしていません。

Magic Bitboard のように巨大なテーブルは必要ない（おかげで速くなる？）ですし、めんどくさい初期化も要らないので実装が楽なのもメリットです。

これらを全部公開するとやねうら王が 4% 速くなってしまうので、一部（香車の利き計算）だけ紹介することにします。

2.1.1　ビットボードレイアウト

Qugiy のビットボードのレイアウトと将棋盤は以下のように対応しています。

Bitboard と将棋盤の関係

9	8	7	6	5	4	3	2	1	
00	09	00	09	18	27	36	45	54	一
01	10	01	10	19	28	37	46	55	二
02	11	02	11	20	29	38	47	56	三
03	12	03	12	21	30	39	48	57	四
04	13	04	13	22	31	40	49	58	五
05	14	05	14	23	32	41	50	59	六
06	15	06	15	24	33	42	51	60	七
07	16	07	16	25	34	43	52	61	八
08	17	08	17	26	35	44	53	62	九
p[0]		p[1]							

いわゆる縦型ビットボードですが、縦型ならなんでも同じだろうと思っていたら、このレイアウトだと同一直線上にビットが 2 つ以上あるかを判定するときに p[0] と p[1] を or してから popcount... ができないという欠陥がありました。直すのがめんどくさいのでこのままにしてあります。このように、Qugiy のビットボードのレイアウトがやねうら王と異なっているため、以下のコードはそのままやねうら王に貼り付けても動きませんが、定数の部分を変えれば正しく動作します。

2.1.2 後手の香車の利き

このようなビットボードのレイアウトの場合、後手の香車の利きは次のように求められます。

```
template <Color C>
inline Bitboard lanceAttacks(int i, const Bitboard& occupancy) {
    if (C == BLACK) {
        ...
    }
    else if (C == WHITE) {
        __m128i mask = _mm_set_epi64x(0x3fdfeff7fbfdfeffULL, 0x000000000001feffULL);
        __m128i em = _mm_andnot_si128(occupancy.xmm(), mask);
        __m128i t = _mm_add_epi64(em, PAWN_ATTACKS[WHITE][i].xmm());
        return Bitboard(_mm_xor_si128(t, em));
    }
}
```

PAWN_ATTACKS[WHITE][i] は後手の歩がマス i にいるときの利きです。（後手の）香車であればたったこれだけで求めることができます。が、先手はこの方法で求められないのでトータルでどっちが速いかは知りません（適当）。4% の高速化の部分はおそらく飛車角の部分のおかげなので...

2.1.3 先手の香車の利き

先手の香車ですが、上のコードほど効率的ではありませんが一応ビット演算で求まります。コードは以下です。

```
if (C == BLACK) {
    __m128i mocc = _mm_and_si128(BLACK_LANCE_PSEUDO_ATTACKS[i].xmm(), occupancy.xmm());
    mocc = _mm_or_si128(mocc, _mm_srli_epi64(mocc, 1));
    mocc = _mm_or_si128(mocc, _mm_srli_epi64(mocc, 2));
    mocc = _mm_or_si128(mocc, _mm_srli_epi64(mocc, 4));
    mocc = _mm_srli_epi64(mocc, 1);
    return Bitboard(_mm_andnot_si128(mocc, BLACK_LANCE_PSEUDO_ATTACKS[i].xmm()));
}
```

BLACK_LANCE_PSEUDO_ATTACKS[i] はマス i にある香車の、盤上に何も駒がないときの利きを予め求めたテーブルです。

以上でビット演算による利き計算の紹介は終わりです。飛車角もこれの応用なので、ぜひ考えてみてください。

2.2 指し手生成の高速化

Qugiy の指し手生成の速度 (1 秒間での Capture(+ 歩の成り)、Quiet(− 歩の成り) の生成回数) は初期盤面で 15M 回/s、指し手生成祭りの局面で 8M 回/s、最大合法手局面で 4M 回/s ぐらいです。その主要な工夫をまとめます。

2.2.1 二歩判定の高速化

ビット演算ばかりで申し訳ないのですが、二歩判定もビット演算で高速化できました。全体の速度にはそこまで影響していない気がするのでコードを公開します。

```
inline Bitboard pawn_drop_mask(const Bitboard& pawns) {
    __m128i left = _mm_set_epi64x(0x40201008040201000ULL, 0x0000000000020100ULL);
    __m128i t = _mm_sub_epi64(left, pawns.xmm());
    t = _mm_srli_epi64(_mm_and_si128(t, left), 8);
    return Bitboard(_mm_xor_si128(left, _mm_sub_epi64(left, t)));
}
```

この関数は、歩のビットボードから、歩のいない筋を 1 で埋めたビットボードを返す関数です。香車の利き計算でのビット演算を応用して、複数の筋について同時に計算しています。同じ要領で複数の香車の利きを一発で求めたりできると思いますが、あまり高速化できる処理がなさそうなので使っていません。

なにげにお気に入りです。

2.2.2 SIMD による駒打ち高速化

駒打ちの生成において AVX2 命令で複数種類の駒を同時に処理すると、指し手生成祭りの局面での生成速度が 2 倍くらいになりました。上の二歩判定よりも圧倒的に効果があるのでおすすめです。

3 現時点での強さ

PVS、指し手のオーダリング、置換表での枝刈りなどの後ろ向き枝刈りに加えて、Futility Pruning, Null Move Pruning, ProbCut, Late Move Reduction などの前向き枝刈りは（条件がかなり適当ですが）実装が終わっています。一方マルチスレッドでの探索や、延長系はまだ実装していません。

Qugiy は floodgate に AisakaTaiga という名前で参加していて、現在のレーティングは 3000 程度です。作者は将棋のルールしか知らないのでどれくらいの強さなのかよく分かっていないのですが、レート的にはまだまだ弱いのでこれから強くしていきたいです。

4 選手権までに実装したい or 試したいもの

予告なく変更される場合があるので予めご了承ください。

1. Lazy SMP
2. ponder 対応
3. １手詰め
4. † Singular Extension †
5. Large Page？の利用
6. SIMD による指し手ソートの高速化
7. まともな時間制御
8. 宣言勝ち

5 最後に

ここまでご覧いただきありがとうございました！

第３１回世界コンピュータ将棋選手権「すいしょう」アピール文書

令和３年３月３１日

杉　村　達　也

第１　はじめに

参加プログラム「すいしょう」（以下，「本プログラム」といいます。）は，Pythonで動作する将棋プログラムです。Python入門書を途中で挫折したような方でも書くことができるような，簡単なコードのみで動作させています。

第２　合法手生成

91	81	71	61	51	41	31	21	11
92	82	72	62	52	42	32	22	12
93	83	73	63	53	43	33	23	13
94	84	74	64	54	44	34	24	14
95	85	75	65	55	45	35	25	15
96	86	76	66	56	46	36	26	16
97	87	77	67	57	47	37	27	17
98	88	78	68	58	48	38	28	18
99	89	79	69	59	49	39	29	19
100	90	80	70	60	50	40	30	20

図のように盤面に番号を振り，上に動かしたい場合1を引く，先手の桂馬の動きをさせたい場合11を引く又は9を足す，という計算で駒を動かす。計算の

結果，11未満となる，100以上となる，又は10の倍数の数値となる場合には合法手ではないため除外する。先手の桂馬の場合，下一桁が1又は2となる場所に行くことはできない。

といった具合で，合法手を生成します。

第３　盤面評価

駒得評価のみの予定です。

第４　探索

以下のルールで探索をします。

１　最も評価値が高い指し手を選択する。

２　選択した指し手の評価値Aと，当該指し手の２手前の指し手の評価値Bが異なる場合，評価値Bを評価値Aの値に変更して，ルートノードから再度探索をしなおす。

第５　おわりに

反則負けをしないよう頑張ります。

以上

チームビール工房HFT支店＠wcsc31アピール文

Have Fun Tech代表の曽根壮大と第28回優勝，第29回準優勝の芝世弐のチームです。

2020年，コンピュータ将棋の新棋戦「電竜戦」が開催されました。
サーバのテストなど多くの準備をする際にテストプログラムの多様性を目的に機械学習データ流用で新規に独自性のあるものを作ろうとしたプロジェクトが立ち上がりました。二番絞りプロジェクトです。
電竜戦の準備の度に強化され，本戦では上位に劣る計算機にて予選３位となりました。
その過程は第４５回ゲーム情報学研究会で報告してあり，またblogにも簡単にまとめてあります。
https://bleu48.hatenablog.com/entry/2021/03/09/075032

本チームはこの後継プログラムを選手権に投入するために結成されました。
面白そうに言うと二番シードを捨てて二番絞りのエントリーです。
ただ，目標は優勝です。

二番絞りプロジェクトにおいて電竜戦では流用元をAobaZeroとし，同サイズの20ブロックのモデルを作成しました。
その後，データ流用でそれより強化された15ブロックのモデルが完成されたためそれにより教師データを数億作成し
そのデータを40ブロックのモデルの学習に転用したものが現時点で最強となっています。
恐らく将棋の深層学習モデルとして過去最強であることは疑いがないところです。
この40ブロックのResNetモデルですが，二世代前のGPU1枚でfloodgateレートで3800近くをたたき出しています。
単純に例えれば，最後のハードウェア統一戦である第5回電王トーナメントにエントリーしていれば優勝するレベルであろうと思われます。

今のGPUでは探索速度が文字通り桁違いですのでどこまで伸びるかギリギリまで強化学習を進めようと考えています。

また，モデルに関してはブロック数，チャンネル数のみならずポリシーヘッド，バリューヘッドの部分変更など複数パターンをテスト中ですので本戦で用いるのが40ブロックになるかどうかは乞うご期待と言った感じです。

ちょっと囲碁に浮気していたとの噂があります。

http://entcog.c.ooco.jp/entcog/new_uec/

囲碁と将棋で技術転用が可能であるようです。

[English]

In the history of infectious diseases, a single drug has been easily resisted by mutations in the pathogen. In contrast, when multiple drugs are used simultaneously, each helps prevent the emergence of resistance to the others. Ryfamate derives from Rifamate which is the combination of two medicines for tuberculosis to retard the development of drug resistance. As with this method, Ryfamate aims to combine an NNUE alpha-beta search engine with deep learning MCTS outcomes to compensate for each other's shortcomings.

Recently, with the advent of deep learning, the opening of games has become more important. Ryfamate selects book moves with the highest hit rates based on the number of playouts instead of those with the best values. This selection not only improves the confidence of the move but also saves thinking time and guides the game to a specific pattern. As a result, Ryfamate strives to gain an advantage even against players with more computational resources.

[にほんご]

性能の良いマシンが参戦するって聞いたけど、定跡にいっぱいヒットさせて時間を稼げば、2950Xだって一世代新しい3950Xとも互角に戦えるはず。
あたいったら天才ね！

しかも、定跡で出てきやすい戦型に絞って学習させたら、二世代新しい5950Xや3090にも勝てるかも。
あたいったら最強ね！

[Library]

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

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

ponkotsu アピール文書

- プログラム名：
  ponkotsu
- 初参加：
  第 31 回コンピュータ将棋選手権
- ソースコード行数
  思考部&UI 部 796 行(2021/5/2 現在)
- 採用している手法
  Value Network
  Policy Network
  マルチタスク学習
  MCTS
  直前になりバグがたくさん取れて初手~10 手目くらいまではちゃんと指しますが，中盤以降はぽんこつなのでポンコツになります.
  土壇場でマルチタスク学習モデルの実装ができましたが見た目バグってそうです…
- 探索速度
  10k～200k nps
- 評価関数パラメータ
  Value Network & Policy Network 約 2000 万（ResNet50 を使用，マルチタスク学習を実施）
- 使用ライブラリ
  libtorch、cshogi

N駒関係を用いた評価関数を採用した.
現在，深層学習以外での主流な評価関数は，固定数の駒の位置関係に対して1つのパラメータを対応付ける2駒関係(PP)または3駒関係(KPP)を用いたものと思われるが，
これらの手法では駒の数を固定数としていることにより，評価能力の限界とパラメータの冗長性の課題が存在すると考えられる.
本プログラムでは任意の数の駒の位置関係を用いた評価関数を用いることでこれらの課題の解消を試みた.

任意の数の駒の位置関係を扱うにあたり，位置関係の集合の作成方法と評価関数の実装とで工夫を行った.
位置関係の集合の作成については，機械学習によるパラメータの調整時に，その時点で保持している位置関係を組み合わせて新たな位置関係を逐次生成することにより集合を自動生成している.
評価関数の実装では，「各局面に現れる位置関係の数は位置関係の総数よりも十分に小さい」というスパース性を考慮したアルゴリズムを用いることで高速化を図った.

EasyShogi　アピール文書
2021.3.30 記　高原順弥

以下の１・２どちらかで出場を予定します。

《１．強化学習をやりたい！》
強化学習の実験場であるOpenAiGymが提供するインターフェースに対応したcshogiで、
DQNアルゴリズムを用いて学習を行ったモデルを使用。
行動価値が最大またはMCTSにて探索（現時点でバグが多い）を行い指し手を決定。
明示的な工夫は特筆するものがなく、自己対局で報酬を得やすいように詰みあり局面からの学習を織り交ぜた。
当初は受け手側は環境でランダムに指し、攻め手側のみエージェントが詰め将棋を学習できるか実験を行ったが、
報酬を得る機会が少なく学習が進まないためエージェント同士の自己対局に戻した。
ローカル環境のCPUのみで学習を継続し、汎化能力は疑問だが、途中図からの対局を繰り返すことで局所的な詰み手順は
学習できることが確認できた。過学習との境の判断は今後の課題。
機械自らが試行錯誤を行う強化学習を体験できとても楽しかった。

《２．GPUを使ってみたい！》
安価にGPUが試せるNVIIA製のJETSON NANO を購入。
計88チャネルの簡易な特徴量（玉の位置、駒種を区別しない位置画像・駒の効き・遠くに及ぶ駒の効き・駒種ごと持ち駒）を用意し、
CNNによる移動後の座標のみを学習するモデルを作成中。
移動後の座標が決まれば、一手先の局面評価値を用いて一意に指し手を決められるようにしたい。
今後オフラインで大会があった場合は、Easyに持ち込めるエッジデバイスで参加をしたいです！