第29回
世界コンピュータ将棋選手権
なのはアピール文書

2019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
- 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！
- 将棋ウォーズ 1級
- 囲碁は日本棋院 初段(アマチュア)
- スマホゲーム非課金勢…スクフェス、デレステ、Fate/GO、ぷにぷに等
- ！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 1700

  メモリ：32GB

  OS：Windows10

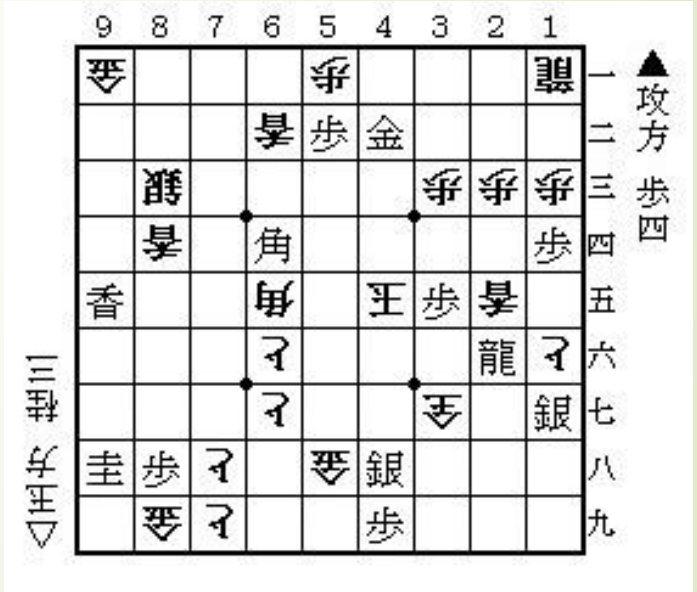
■なのはの構成

- Visual Studio Community 2017(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)
  - AMD RYZENに適したデータ構造
- 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
- 評価ベクトルは今年こそ自前学習で作成
- 詰めルーチン(df-pn)搭載

…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？

- 詰めルーチンの更なる強化
- 終盤の強化
  - コア数が増加したため、詰め探索用のスレッドを設定

ユ「詰め探索変更ってどんな風に変えたの？」

な「うん…あれって探索が遅くて高速戦では使えないから」

ユ「やっぱり高速化？」

な「ううん　逆！チャージタイムを増やして威力を大幅アップ！」

な「最大詰め手数の強化を最優先してみたの」

ユ「そ…　そう……」

※以下を参考に改変:「魔法少女リリカルなのはA's THE COMICS」pp.20-21, 原作 都築真紀, 作画 長谷川光司, 学習研究社, 2006.

■なのはの強さ

- 第24回世界コンピュータ将棋選手権で**1位！**(*1) (*2)
- 第25回世界コンピュータ将棋選手権で**1位！**(*1) (*3)
- 第26回世界コンピュータ将棋選手権で**2位！**(*1) (*4)
- 第27回世界コンピュータ将棋選手権で**1位！**(*1) (*5)
- 第28回世界コンピュータ将棋選手権で**2位！**(*1) (*6)

(*1) AMD製CPUをメインに使ったシステム構成での参加ソフトの中で(当者調べ)

(*2) トータルでは17位　　(*3) トータルでは11位　　(*4) トータルでは12位

(*5) トータルでは13位　　(*6) トータルでは20位

■意気込み

- できれば**シード権獲得！**
- あわよくば**入賞！**(あと、AMD勢で1位奪取したい)
- ライブラリ不使用者上位5位内

➡全力全開、手加減なしで！！

(今後の課題)

- 検討に使われるように振り飛車が不利飛車にならないようにする
- 詰めろ絡みの局面が続いても正着を続ける終盤力(即詰みがない時)
- 最長詰め手数の詰将棋の「ミクロコスモス」 (1525手詰め)を解く
- カットインやエフェクトなど**派手な演出**！！

■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

- やねうら王公式サイトで配布された教師データを使います。

■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

●NineDayFever

局面評価の極北を目指すといいつつ、アムンゼン隊のそり跡を見つけたスコット隊のような状況になっております(それは南極だ)。

2019年の相違点は以下の通りです。
・雑巾絞りで使用する対局結果の初期局面を生成する手法を実験中です。王以外の駒でも、違う場所に動かした局面を作ってそこから始めたいのですが、それだと候補が多すぎて破綻するので、どうやって動かす駒を選ぶかというのを実験中です。
・movecount pruning と lazy SMP がどうも相性が良いらしいのですが、この原因は、PVは transposition table経由で全スレッドで共有しつつ、2番目以降に評価する枝は確率的に選択するという、一種のモンテカルロ指向αβ探索になっているのではないかという仮説を立ててみました。2番目以降に探索する手を適当にばらけさせるため、ヒストリをスレッド別に持つのがよく行われているようですが、ばらけさせるのであれば、ヘルパースレッドのヒストリにノイズを入れてやることで同じ効果が得られそうだということで、そういう実装を実験してます。

・ライブラリ選択理由
2012年ごろにKPP/KKP テーブルで機械学習手法を試そうとした時点では bonanza 以外に選択肢がなかったので、そのまま使い続けています。惰性です。

以下は2018年の内容です。

・雑巾絞りやってます。
・三駒関係の各変数を分解して共通する要素を抽出したうえで機械学習することで未知の局面への対応能力を高めています(世間では次元落としとか呼ばれているらしい)。
・手番を考慮した評価値を使用しています(世間ではKPPTという名前がつけられているらしい)。
・定跡では自己対戦結果から各局面の勝率の分布を推定して各手の採用確率を決めています。生成した

局面からの対戦結果は雑巾絞りに使います。

・定跡では2017年にえらい目にあったので、Qhapacさんが公開していた定跡を取り込んでます。定跡の末端での勝率ベースで更新していく手法なので、他の定跡でもその末端の勝率を求めることで取り込むことができるようになってます。

・プログラムはbonanza 6.0にstockfish の手法を取り入れています。

第29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「あやめ」

昨年は「おから饅頭」という名前のプログラムで出場していましたが、今年は名前を変更しました。

今年の選手権に向けてプログラムを1から書き直し、将棋の盤面のデータ構造についても変更を加えています。
また、実現確率探索に用いる着手予測モデルや局面評価関数についても大きな変更を加えています。

第 29 回世界コンピュータ将棋選手権
dlshogi アピール文章

山岡忠夫
2019/3/25

※下線部分は、第 28 回世界コンピュータ選手権からの差分を示す。

1　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
- 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
- 既存将棋プログラムの自己対局データを使った事前学習
- REINFORCE アルゴリズムによる Policy Network の学習
- ブートストラップ法による Value Network の学習
- マルチタスク学習
- 末端ノードでの短手順の詰み探索
- ルートノードでの長手順の詰み探索
- 序盤局面の事前探索（定跡化）
- マルチ GPU 対応
- CUDA、cuDNN を直接使用
- GPU ごとに異なるモデルの読み込み
- Optuna による探索パラメータの最適化
- 確率的な Ponder（実装予定）

2　使用ライブラリ

- elmo[1]（Commits on May 29, 2017）
  →事前学習用の訓練データ生成に使用
- Apery[2]（WCSC28）

[1] https://github.com/mk-takizawa/elmo_for_learn
[2] https://github.com/HiraokaTakuya/apery

→局面管理、合法手生成のために使用

2.1　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る elmo、及び、合法手生成を高速に行える Apery を選定した。

3　各特長の具体的な詳細（独自性のアピール）

3.1　ディープラーニングを使用

DNN(Deep Neural Network)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。

3.2　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、Wide Residual Network[3]を使用した。

入力の畳み込み 1 層と、ResNet 10 ブロック(畳み込み 2 層で構成)と出力層の合計 22 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 192 とした。

3.3　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。

3.4　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

[3] https://arxiv.org/abs/1605.07146

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。

3.5 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[4]の疑似コードに記述された式を使用した。

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。

3.6 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。

3.7 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[5]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。

3.8 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。勝率予測が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ができるが、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。

---

[4] http://science.sciencemag.org/content/362/6419/1140
[5] https://arxiv.org/abs/1712.01815

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。

3.9 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。

3.10 REINFORCE アルゴリズムによる Policy Network の学習

単純に自己対局の指し手を学習するのではなく、学習局面の価値と勝敗データと関連付けて学習を行う。良い局面から負けになった手は、悪手として負の報酬を与え、悪い局面から勝ちになった手は善手として正の報酬を与える。学習アルゴリズムには、AlphaGo の論文に掲載されている REINFORCE アルゴリズムを使用した。

3.11 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。

3.12 マルチタスク学習

Policy Network と Value Network のネットワーク構成が同じ層を共通化し、出力層を分けることで、同時に学習を行う。

関連する複数のタスクを同時に学習することをマルチタスク学習という。タスク間に関連がある場合、単独で学習するよりも精度が向上する。

また、対局時に Policy Network と Value Network を同時に計算できるため、高速化の効果もある。

3.13 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、7 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。

3.14 ルートノードでの長手数の詰み探索

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手を抜くことがある。

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。

3.15 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。（実装予定）

3.16 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。

「3.6 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。

アムダールの法則により、増やせる GPU には上限があるが、確認できている範囲で 4GPU まで線形で探索速度を上げることができている。

3.17 CUDA、cuDNN を直接使用

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 Chainer を使用しているが、対局プログラムには、ディープラーニングフレームワークを用いず CUDA、cuDNN を直接

使用する。Chainer で学習したモデルを読み込み、推論を行う処理をスクラッチで開発した。

高速化と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。

3.18 GPU ごとに異なるモデルの読み込み

モデルごとに誤る確率が独立である場合、複数モデルが同時に誤る確率は、単一のモデルを使用する場合より低くなる。GPU ごとに異なるモデルを読み込むことで、探索の精度の向上が期待できる。

3.19 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[6]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の推移を目的関数として、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[7]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。

3.20 確率的な Ponder（実装予定）

モンテカルロ木探索は確率にゲーム木を成長させる。その特性を活かして、相手が試行中に、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 学習について

4.1 事前学習

- 事前学習データ：elmo(wcsc27)で深さ 8 で生成した 4.9 億局面

4.2 自己対局による強化学習

4.2.1 学習データ、パラメータ

- ミニバッチサイズ：64～1024（段階的に増やす）
- 学習アルゴリズム：Momentum SGD（学習率 0.01→0.001、慣性係数 0.9）
- 強化学習 1 サイクルで生成する局面：250～500 万局面
- 強化学習 1 サイクルで学習する局面：直近 10 サイクル分
- 強化学習のサイクル数：140（3 月末時点）

[6] https://optuna.org/

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

4.2.2 学習結果

2017 年～2018 年 6 月の floodate のレート 3500 以上の棋譜との一致率で評価

【事前学習を行ったモデル】

- Policy Network の一致率：40.9%
- Value Network の一致率：68.8%

【自己対局による強化学習を行ったモデル】

- Policy Network の一致率：44.0%
- Value Network の一致率：71.7%

以上

CGP　アピール文章

2019/3/29 初稿
2019/ 4 /25 改訂
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
・コンピュータ将棋では一般的にはあまり使われていない機能を使用
・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 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 AVX2 を使っていた部分からの置き換えと駒打ちの指し手生成くらいしか出来ておりません。

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。評価関数テーブルは駒割＋入玉時の位置評価のみの初期値から KPP 相対などの次元下げ＋ミニバッチ方式ボナンザメソッド＋Adam で自力で機械学習したものを元に、~~次元下げはそのままに elmo 方式をベースに変更を加えた方式(改悪か改良か中立的かは不明)＋Adam や AdaBound で学習したものです。~~
（追加学習失敗しました）

・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。

オーダリングの一部は VPMAX 命令で次に試す手を抜き出す方式を取っております。この方式は条件分岐なく複数の手を比較できるため、挿入ソートを通常の x86 命令で行うより高速化できております。SIMD を用いたソートの使用は試していないのでこちらのほうがより高速になる可能性もあります。

また置換表や評価値テーブルのページテーブルにラージページ(Windows での言い方 Linux 用語だと Huge Page、自動で Huge Page を使う Linux ディストリビューションもあるらしい)を使用し、高速化を図っております。

『ひまわり』

ひまわり 開発者
山本 一将，永塚 拓，高木 厚成

特徴

- AlphaZeroのアルゴリズムで新規に作成
  - 前年からの流用部分はルール部分のみ
- MCTSで探索
- 教師あり学習
  - 強化学習は間に合わない
- Tensorflowを使用して学習

Crazy Shogi is an implementation of the AlphaZero algorithm, as published by DeepMind. It is entirely original code developed from scratch, in C++. The neural network code uses the CUDNN library to run on Nvidia GPUs.

==========

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

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

==========

「芝浦将棋 Softmax」のチーム紹介

2019 年 3 月 22 日
芝浦工業大学情報工学科
五十嵐治一，横田直之，吉野拓真，岩本裕大

1. はじめに

本稿は，第 29 回世界コンピュータ将棋選手権（2019 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 3 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（α β探索）とは異なる Softmax 探索である点が大きく異なります. ただし, 合法手生成までは芝浦将棋 Jr.と共通で, 選手権公式ライブラリとして登録されている「芝浦将棋 Jr.合法手生成プログラム」[1]を使用しています. 棋力的には従来のα β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．横田，吉野，岩本は五十嵐研究室の大学院生と学部 4 年生(いずれも 2019 年度)です.

3.「芝浦将棋 Softmax」の特徴

本チームの特徴を，以下の１）～５）のようにまとめました．このうちの２)～４）が本チーム独自の探索方式である「MC Softmax 探索」[6]に関する説明です．この探索方式は, 文献[2][4]の研究が基になっています.

１）芝浦将棋 Jr. の合法手生成ルーチンを使用

芝浦将棋 Jr. では盤面表現のデータ構造を独自の Magic bitboard を用いて駒の利き場所での駒の配置状況などを計算しています[3]．この計算を含む合法手生成のプログラムは「芝浦将棋 Jr.合法手生成プログラム」の名称で選手権公式ライブラリとして登録されています. 芝浦将棋 Softmax はこの合法手生成プログラムをそのまま使用しています.

２）モンテカルロ・ソフトマックス探索を使用

現在のチェスや将棋のプログラムは「Min-max 探索」という探索方式をほぼ 100%採用しています．これに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，α β 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（α β 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．本チームは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」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」(MC Softmax 探索方式)と呼んでいます．

３）ノードの評価関数を用いたボルツマン分布による確率的なノード選択

前項で述べた Softmax 探索には，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（実際にはそこで静止探索も行っています）．

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

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

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．

４）深さ制御とバックアップ操作

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，３）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．

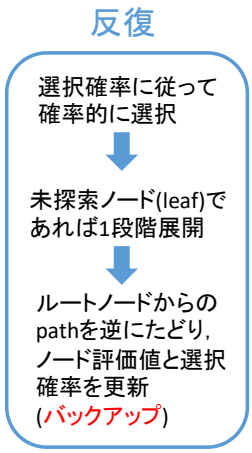


図１

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３）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．

５）評価関数について

現在のところ，評価関数の特徴量は，選手権公式ライブラリである Bonanza (Ver. 6.0.0) [5] のものをそのまま使用しています．Bonanza は「Bonanza メソッド」と呼ばれる教師付学習方式が有名です．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．本学習法では，正解手以外の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[6]．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[10]．

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 α β 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

今年のバージョンでは，以下の機能を追加しました．これまで探索木中に同一局面が存在しても全く別のノードを用意してきました．したがって，同じ局面に対してその局面からの探索処理を何度も繰り返してきました．そこで，同じ局面には一つのノードを割り当てる処理を追加しました．このノード（合流点）の管理のために，専用のハッシュテーブルを新たに用意しました．探索木中の各ノードは複数の親ノードを持つようになりますが，同一局面の展開処理を繰り返すことがないので，探索時間とノード数を削減することができます．ただ，現在は，動作が不安定な点があり完全に実装できていませんが，大会までに間に合えば実装します．

5. 今後の課題

今年は昨年同様，36 コア(72 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．しかし，有力手順を重点的に探索するためにはスレッドの割当方法を工夫する必要があります．また，各スレッドが同じ温度パラメータを持って枝を選択して降りて行く必要性は必ずしもありません．選択時に温度の高いスレッドや低いスレッドがあって，広く浅く探索するスレッドや狭く深く読むスレッドなどのバリエーションを持たせることも可能と考えています（ただし，バックアップ時の温度は一定としておく）．大会までに温度調整がうまくいけば実装します．

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 というチームが早速採用してくれました[9]．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[9]．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．

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，本探索方式は，プログラム作成が容易で，並列化の効果も高い上に，他のゲームプログラムへの適用も容易であるという点で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[10]，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] 「芝浦将棋 Jr.合法手生成プログラム」の機能説明書とプログラムは次のページからダウンロードできます：http://www2.computer-shogi.org/library/

[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] 例えば，http://www2.computer-shogi.org/wcsc26/appeal/Shibaura_Shougi_Jr./appeal.pdf に記載されています．

[4] 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).

[5] Bonanza のホームページ，http://www.geocities.jp/bonanza_shogi/

[6] 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）.

[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] Stockfish のホームページ，https://stockfishchess.org/

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

[10] 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），

第 29 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。

使用ライブラリ

やねうら王 [1] (ソースコードおよび教師局面) : ユーザ定義エンジンの追加がしやすいため
Apery, elmo : 使用の可能性があったため申請しましたが、不使用となりました

基本構成

USI 通信・合法手の生成までをやねうら王ライブラリで行い、探索部・評価関数は独自に実装しています。
探索部は MCTS を用い、選択的な探索を行います。評価関数において GPU を効率的に動作させるため、100~1000 局面を同時に評価できるようキューを用いたシステムになっています。前年度はシングルスレッドで探索を行っていましたが、これをマルチスレッド化し、末端局面での詰み探索を行うよう改良しました。優勢になってからちゃんと勝ち切れるようになった印象があります。
評価関数は Convolutional Neural Network (CNN)を用います。教師ありで学習しています。前年度はフレームワークとして Chainer を用いており、対局エンジンの動作に python 環境が必要でした。今回は C++から直接呼び出せる CNTK に変更したことで、python 環境が不要となり配布が容易となりました。
定跡はやねうら王の標準定跡です。

CNN

CNN の構造は、3×3 の convolution 2 層を 1 つのブロックとし、Batch Normalization、ショートカット接続を加えた residual 構造です。Convolution のチャンネル数は 192、ブロック数は 19 です。前年より深さを増やしました。192ch, 16 層→192ch, 42 層
入力は 119 チャンネル、9×9 サイズです。dlshogi [2]を参考にさせていただき、駒の配置およびある地点に効いている駒の数、王手かどうかを含めています。
出力は 2 系統あります。方策部分は 27 チャンネル、9×9 サイズです。CNNShogi [3]を参考にさせていただき、移動先マスごとに、駒の種類と移動元を表現する 27 チャンネルです。棋譜の正解の指し手に対して softmax cross entropy loss で学習します。静的評価値部分は 2 チャンネルあり、勝ちと負けに対応します。(1 チャンネルにして、tanh 関数で勝率にし

ても意味的には同じです。フレームワーク上実装しやすいほうを採用しているだけです。）
elmo と同様、棋譜の評価値を sigmoid で勝率に直したものと、対局の勝敗それぞれと softmax cross entropy loss をとり、その和を損失として学習します。
チャンネル数・ブロック数を増減させた結果として、現状より計算量が増える方向だと精度の増加よりも nps の低下の影響のほうが大きいようです。

思考時間制御

前回は予想残り時間を均等分割するだけの単純な思考時間制御で、明らかな手がある場面でも時間をかけていました。今回は、指し手の選択確率（MCTS におけるルートノードから子ノードへの訪問回数の比率）が特定の 1 手で十分大きくなり、残り思考時間内に大小関係が逆転する確率が一定以下になったらその時点で指す機構を実装しました。
「現在トップの指し手確率」×-5.326＋log2(「現在までのゲーム木のノード数」/1024)×-0.306＋log2(「予定残り思考時間内に評価できるノード数」/1024)×0.377＋0.798
という単純なモデルで、今後の思考で指し手が変化する確率を算出することとしました。パラメータは実際に思考を行った結果にフィッティングさせて決定しています。
本番で利用するハードウェアは強力なため、貧弱なハードウェアで作ったデータへのフィッティングの結果を外挿して活用することになります。数局試した感じでは悪くない挙動です。

ハードウェア

Amazon EC2 (クラウド)の p3.16xlarge インスタンスを利用予定です。Tesla V100 GPU を 8 台搭載しています。GPU をフル稼働させると 30,000nps 程度出ます。
本来は 16bit 演算で高速化を狙いたかったのですが、実装上の問題で実現できませんでした。
混雑しているとマシンが確保できない場合もあり、運次第です。クラウド利用不能時は方策関数のみで指すモードをローカルマシンで動かします。この場合は lesserkai に勝ったり負けたりする程度の強さです。

目標

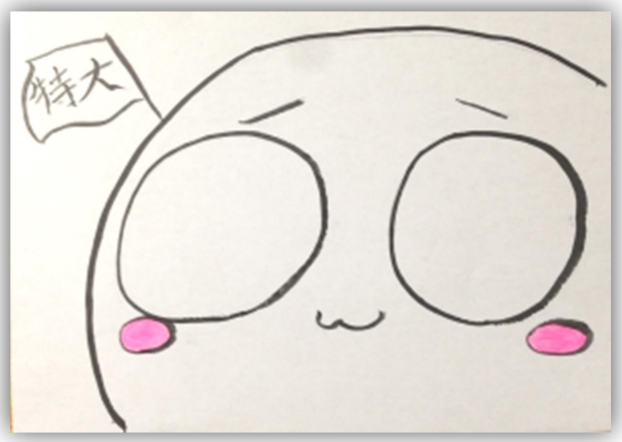
1 次予選突破。前回より少し強くなっているはずです。

[1] 磯崎元洋 https://github.com/yaneurao/YaneuraOu
[2] TadaoYamaoka https://github.com/TadaoYamaoka/DeepLearningShogi
[3] not522 https://github.com/not522/CNNShogi
2019 年 1 月 7 日作成（5 月 1 日改訂） 日髙雅俊 https://github.com/select766

TMOQ アピール文書

2019 年 3 月 30 日 作成
2019 年 5 月 02 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： 36 チーム中 15 位
WCSC27： 参加辞退
SDT5： 42 チーム中 23 位
WCSC28： 40 チーム中 16 位

【今回の趣旨・TMOQ の特徴】
好きなコンピュータ将棋を題材に深層学習を学ぼうというのが今回の趣旨です。前回までもそうですが、自分的に新しいことを試すのを主眼としております。
ベースとしているのは、山岡忠夫氏の『将棋 AI で学ぶディープラーニング』のサンプルプログラムです。ここに自己対局による強化学習・序盤定跡・詰め探索を加えたい！と考えております。また、強化学習において Apery/やねうら王の評価関数を利用して指し手を絞ることで強化学習を加速できるのではないかと妄想し、この辺りもチャレンジ！
まずは動くプログラムを準備して参加することが目標。

【使用ライブラリ】
「dlshogi」「python-shogi」「Apery WCSC26 版」「やねうら王 コンピューター将棋フレームワーク」を使う可能性があります。

【定跡について】
棒銀を指すよう改良した「まふ定跡」（WCSC28 で使用）に、千田先生の C-Book（20180802 版）と floodgate（2015-2017 年）の Rating 上位ソフトの棋譜をマージし、更に磯崎元洋氏公開の「100 テラショック定跡」で拡張し、100 万手を超える定跡ファイルを準備しました。これは強くするというよりは､序盤で変な手を指して 30 手前後で終るのを避けるためです。

【学習について】
平岡拓也氏作成の初期局面ファイルに floodgate（2015-2017 年）の Rating 上位ソフトの棋譜をマージし、瀧澤誠氏の elmo_for_learn で教師局面を生成して学習を行いました。
計算リソースが少ないと学習が遅いですね。とはいえ、学習が進むにつれ Floodgate の Rating が上がっていき、興奮しました。

【作者】
宮崎の生まれ、幼少の頃より将棋を嗜むも、指し将棋は 5 級程度（初段の免状は保有）。IT 企業勤務ですが、コンサルや管理系の仕事で海外を飛び回ることが多く、ここ 10 年は仕事で開発を行っておりません。大好きな開発を行うため、将棋のソフトに触り始めました。
今回も山岡忠夫氏をはじめ、多くの方が公開された情報により何とか参加できました。この場を借りて、感謝いたします。

山田将棋について（2019 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

dainomaruDNNcは第28回同様、DeepLearningShogi-wcsc28を使用しております。

DeepLearningShogi-wcsc28より、

1 詰み探索の深さを7から9

2 開始20手以上になった場合、探索深さ6 まで行い、3手詰めかどうか 判定を行うよう変更

3 探索回数最大の手を見つけるにおいて、詰め確認ができている場合は勝率設定をするよう変更

ディープラーニングによる強化学習のデータにおいては、DeepLearningShogi-wcsc28と変更なし。

ライブラリの選定理由

ディープラーニングの実装を調査していた際、DeepLearningShogi-wcsc28のベースとなった

DeepLearningShogi-masterの開発者の

「将棋でディープラーニングする」ブログを拝見させて頂き、実装方法や仕組みなどのご教授頂きました。

第28回出場の際、ライブラリーの使用のご了承頂いた経緯があります。

ブログでは、開発するにあたって環境構築や改善に対する試みが大変わかりやすい説明となっており、どのように改善してよいかの

検討をすることができました。また、開発者に対して私の質問に対し丁寧対応して頂き理解をすることができました。

DeepLearningShogi-wcsc28の開発者と自分の改善と比較し、ディープラーニングについて理解を深めたいと思ております。

■きのあ将棋について

○作成 :2019/03/31 ...過去のものを元に作成。

●きのあ将棋の特徴

1手ごと思考エンジンを実行し、次の着手を思考する方法を採用しています。(2011年採用)

・この方法のメリット

→１台のマシンで沢山の相手と対局する際にて、コンピュータ資源の節約。

→思考プログラムの実行を管理しやすくなる。など。。

・この方法のデメリット

→前回の思考時のハッシュの利用ができなくなる。

→相手の思考時間を有効利用できない。など。。

●使用ライブラリなど

現在のところ過去においても将棋ライブラリの利用ならびに利用予定なし。

他プログラム(手作成も含め)の評価値や読み筋、定跡データなどの利用もなし。

プロ棋士の棋譜データの利用のみあり。(椿原チーム様から。椿原チーム様ありがとうございます)

とはいえ研究開発を汎用化/効率化の目的から、きのあ将棋/囲碁で汎用ライブラリを利用。

●評価について

ここ数年と同じく、

・置換テーブルの値を元に、乗算、加算の単純な繰り返しで評価関数を構築を模索。

・ニューラルネットワークの考え方と独自のやり方のマージした方法を模索。

●機械学習について

「2の倍数固定値方式」や「ミニバッチのサイズ」などで、

数時間あるいは10分以内に収束するような簡易モデルで実験を繰り返して模索を繰り返しています。

●強さとは別に

UI、時間、待った、手数、初期局面、結果などと、good/bad投稿を

元にどういう内容がユーザさん満足度が高いかを本格的に研究を進めています。

10近く前？にも同じような研究を進めていましたが、

今回は汎用的な仕組みで、囲碁やその他にも共通で組み込める仕組みで分析システム構築を進めています。

また、good/badデータを元に

ユーザさんの満足度が上がるような評価関数を機械学習できるように研究を進めているのと、

そのgood率が高い局面とは何か、先後入れ替えても同じく高いと言えるのか(対称性の問題)などに興味があります。

現在の段階は、汎用的な仕組みでデータ集積段階で、

単純な統計分析でどのようなUIや、どのような利用方法が満足度が高くなるのかを分析している段階です。

5月までにデータ集積が間に合い、体力的余力があれば、

本項あるいは、きのあ将棋サイト上にてレポートを公開したいと思います。

●きのあ将棋サイト

https://syougi.qinoa.com/ja/

End

まったりゆうちゃんのアピール文書 2019

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。

手抜きについて

手抜きチーム

2019 年 3 月 25 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。作者らが将棋のプログラムの仕組みを理解することを目的として開発しています。リポジトリ：https://github.com/hikaen2/tenuki-d

1　作者の紹介

鈴木太朗

鈴木はプログラマです。会社員をしています。仕事のかたわら趣味の自転車と合唱に打ち込んでいます。そのかたわら手抜きを作っていますが完成させる自信がありません（夏休みの宿題を 9 月 1 日にやるタイプです）。棋力は 30 級程度です。Twitter: @hikaen2

玉川直樹

玉川は P 兼召喚士です。KPP とレーティングシステムを疑っています。将棋ウォーズ 2 段。手抜きの定跡を手入力で作りました。Twitter: @Neakih_kick

2　目標

2.1　今年の目標

- 探索をマルチスレッドにする

Ryzen を買っちゃったのでマルチスレッドにします。これから Lazy SMP を実装します。スレッドがよくわかりません。実装できても正しく実装できているか分からない予感がします。

- KPPT で評価する

去年はなのはさんの評価バイナリで KP + PP で評価しました。今年は Apery の評価バイナリ（大樹の枝形式）で KPPT で評価しようと思います。Apery の選定理由は評価バイナリが KPPT だからです。予想として，静的評価に手番が含まれると静止探索がいらないか，もしくは軽くていいのではないかと思っています。

評価バイナリの自作については来年挑戦します。

2.2　来年の目標

- 評価バイナリを自作する

2.3　去年までの目標と結果

第 28 回世界コンピュータ将棋選手権：

- ルール通りに指す → 結果：ルール通り指した（けど千日手のチェックをしてない）
- 投了する（第 27 回では王手放置していた） → 結果：投了した。一次予選 4 勝 4 敗
- KP で評価する → 結果：なのはさんの評価バイナリで KP + PP で評価した

第 27 回世界コンピュータ将棋選手権：

- 出場する → 結果：出場した
- 対局する → 結果：対局した。一次予選 2 勝 5 敗

3　プログラムの構造

この章では手抜きの構造の説明をします。

3.1　データ構造

盤面に 81 要素の 1 次元配列を使用しています。そのアドレッシングはこうです：

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　盤面のデータ構造

第 28 回世界コンピュータ将棋選手権では「壁」がありましたが，なくしました。壁がないと，壁のレイアウトを考えなくて済む，盤面がコンパクトになる，zobrist hash seed もコンパクトになる，静的評価が簡単になる（盤面のアドレッシングと評価バイナリのアドレッシングが合っていれば）といった利点があります。一方で合法手生成が複雑になる欠点があります（1 段目と 9 段目が繋がっているので油断すると駒がワープする）。

盤面の各要素は 0 から 28 までの値を取ります。値の意味はこうです（次表の A 列）：

表 1　値の意味

A		type(A)	base(A)	color(A)	promotable(A)	promote(A)	inv(A)
0	▲歩	歩	歩	▲	T	▲と	△歩
1	▲香	香	香	▲	T	▲成香	△香
2	▲桂	桂	桂	▲	T	▲成桂	△桂
3	▲銀	銀	銀	▲	T	▲成銀	△銀
4	▲金	金	金	▲	F	▲金	△金
5	▲角	角	角	▲	T	▲馬	△角
6	▲飛	飛	飛	▲	T	▲龍	△飛
7	▲王	王	王	▲	F	▲王	△王
8	▲と	と	歩	▲	F	▲と	△と
9	▲成香	成香	香	▲	F	▲成香	△成香
10	▲成桂	成桂	桂	▲	F	▲成桂	△成桂
11	▲成銀	成銀	銀	▲	F	▲成銀	△成銀
12	▲馬	馬	角	▲	F	▲馬	△馬
13	▲龍	龍	飛	▲	F	▲龍	△龍
14	△歩	歩	歩	△	T	△と	▲歩
15	△香	香	香	△	T	△成香	▲香
16	△桂	桂	桂	△	T	△成桂	▲桂
17	△銀	銀	銀	△	T	△成銀	▲銀
18	△金	金	金	△	F	△金	▲金
19	△角	角	角	△	T	△馬	▲角
20	△飛	飛	飛	△	T	△龍	▲飛
21	△王	王	王	△	F	△王	▲王
22	△と	と	歩	△	F	△と	▲と
23	△成香	成香	香	△	F	△成香	▲成香
24	△成桂	成桂	桂	△	F	△成桂	▲成桂
25	△成銀	成銀	銀	△	F	△成銀	▲成銀
26	△馬	馬	角	△	F	△馬	▲馬
27	△龍	龍	飛	△	F	△龍	▲龍
28	空	空	空	-	F	空	空

第 28 回世界コンピュータ将棋選手権では手番と成りをビット演算で管理していましたが，表で管理するようにしました。表で管理するとビットレイアウトを考えなくて済む利点があります。一方，表を引くのに（ビット演算よりは）時間がかかる欠点がありそうです。

表の type(A) の列から inv(A) の列までは A 列の関数です。意味はこうです：

- type(A): A から手番を除いた値
- base(A): A から手番を除いて成りを戻した値。駒を取ったときに使う。
- color(A): A の手番
- promotable(A): A が成れる駒かどうか
- promote(A): A が成った駒
- inv(A): A の手番を反転した駒

こうして見ると全部で 6 つもあって多いように思います。全部で 4 つくらいになってもいいような気がします。

3.2 探索

mini-max 探索をしています。

- $\alpha\beta$ 枝刈りをしています。最近やっと $\alpha\beta$ 枝刈りが何なのか分かるようになりました。
- null-move 枝刈りをしています。
- 静止探索しています。
- 指し手だけの置換表（指し手表?）があります。表のエントリは指し手だけです。一般的な置換表のエントリにはミスヒット検出のために局面のハッシュ値が入れてありますが，手抜きでは入れていません。指し手だけの表なのでミスヒットしても害がないと考えているためです。しかし Lazy SMP しはじめると問題があるかもしれません。
- Lazy SMP したいです。
- futility 枝刈りしていません。静的評価が差分評価でないからです。

3.3 静的評価

Apery, commit 3221627（2016 年 11 月 3 日）の評価バイナリを使って KPPT で評価する予定です。以下は Apery の評価バイナリについての覚書です（独自調査；間違ってたらすみません）。

覚書

Apery は commit a0a09de（2015 年 10 月 15 日）で手番評価を実装し KPPT 評価になった（それ以前は KPP 評価だった）。その評価バイナリには以下の 3 つのファイルがある：

- KK_synthesized.bin（52,488 バイト）
- KKP_synthesized.bin（81,251,424 バイト）
- KPP_synthesized.bin（776,402,496 バイト）

それぞれのファイルには，通常の評価値と手番を持つ側に加算される評価値の 2 つ組が，複数個格納されている[*1]。それぞれの評価値は 2 バイトもしくは 4 バイトの符号付き整数でリトルエンディアンである。

KK_synthesized.bin には 4 バイトの評価値を持つ 2 つ組（8 バイト）が 81（先手玉の位置）×81（後手玉の位置）＝ 6,561 個格納されている。

KKP_synthesized.bin には 4 バイトの評価値を持つ 2 つ組（8 バイト）が 81（先手玉の位置）×81（後手玉の位置）×1548（※）＝ 10,156,428 個格納されている。

KPP_synthesized.bin には 2 バイトの評価値を持つ 2 つ組（4 バイト）が 81（自玉の位置）×1548（※）×1548（※）＝ 194,100,624 個格納されている。

※ 次表の 1548 要素：

[*1] NineDayFever http://www2.computer-shogi.org/wcsc24/appeal/NineDayFever/NDF.txt

表 2　1548 要素の内訳

意味	from	to	要素数
味方の持駒の歩の 0 枚目〜18 枚目の評価値	0	18	19
相手の持駒の歩の 0 枚目〜18 枚目の評価値	19	37	19
味方の持駒の香の 0 枚目〜4 枚目の評価値	38	42	5
相手の持駒の香の 0 枚目〜4 枚目の評価値	43	47	5
味方の持駒の桂の 0 枚目〜4 枚目の評価値	48	52	5
相手の持駒の桂の 0 枚目〜4 枚目の評価値	53	57	5
味方の持駒の銀の 0 枚目〜4 枚目の評価値	58	62	5
相手の持駒の銀の 0 枚目〜4 枚目の評価値	63	67	5
味方の持駒の金の 0 枚目〜4 枚目の評価値	68	72	5
相手の持駒の金の 0 枚目〜4 枚目の評価値	73	77	5
味方の持駒の角の 0 枚目〜2 枚目の評価値	78	80	3
相手の持駒の角の 0 枚目〜2 枚目の評価値	81	83	3
味方の持駒の飛の 0 枚目〜2 枚目の評価値	84	86	3
相手の持駒の飛の 0 枚目〜2 枚目の評価値	87	89	3
味方の歩がアドレス 0〜80 にいるときの評価値	90	170	81
相手の歩がアドレス 0〜80 にいるときの評価値	171	251	81
味方の香がアドレス 9〜80 にいるときの評価値	252	332	81
相手の香がアドレス 0〜80 にいるときの評価値	333	413	81
味方の桂がアドレス 0〜80 にいるときの評価値	414	494	81
相手の桂がアドレス 0〜80 にいるときの評価値	495	575	81
味方の銀がアドレス 0〜80 にいるときの評価値	576	656	81
相手の銀がアドレス 0〜80 にいるときの評価値	657	737	81
味方の金がアドレス 0〜80 にいるときの評価値	738	818	81
相手の金がアドレス 0〜80 にいるときの評価値	819	899	81
味方の角がアドレス 0〜80 にいるときの評価値	900	980	81
相手の角がアドレス 0〜80 にいるときの評価値	981	1061	81
味方の馬がアドレス 0〜80 にいるときの評価値	1062	1142	81
相手の馬がアドレス 0〜80 にいるときの評価値	1143	1223	81
味方の飛がアドレス 0〜80 にいるときの評価値	1224	1304	81
相手の飛がアドレス 0〜80 にいるときの評価値	1305	1385	81
味方の龍がアドレス 0〜80 にいるときの評価値	1386	1466	81
相手の龍がアドレス 0〜80 にいるときの評価値	1467	1547	81
		合計:	1548

例：

1. KK[44][36][0]：先手玉がアドレス 44 にいて，後手玉がアドレス 36 にいるときの評価値
2. KK[44][36][1]：先手玉がアドレス 44 にいて，後手玉がアドレス 36 にいるときの，手番を持つ側に加算される評価値
3. KKP[44][36][1][0]：先手玉がアドレス 44 にいて，後手玉がアドレス 36 にいるときの，先手の持駒の 1 枚目の歩の評価値
4. KKP[44][36][1][1]：先手玉がアドレス 44 にいて，後手玉がアドレス 36 にいるときの，先手の持駒の 1 枚目の歩の，手番を持つ側に加算される評価値
5. KKP[44][36][2][0]：先手玉がアドレス 44 にいて，後手玉がアドレス 36 にいるときの，先手の持駒の 2

枚目の歩の評価値

6. KKP[44][36][20][0]：先手玉がアドレス 44 にいて，後手玉がアドレス 36 にいるときの，後手の持駒の 1 枚目の歩の評価値
7. KKP[44][36][576][0]：先手玉がアドレス 44 にいて，後手玉がアドレス 36 にいるときの，アドレス 0 にいる先手の銀の評価値
8. KKP[44][36][657][0]：先手玉がアドレス 44 にいて，後手玉がアドレス 36 にいるときの，アドレス 0 にいる後手の銀の評価値
9. KPP[44][1][2][0]：自玉がアドレス 44 にいるときに，味方の持駒の 1 枚目の歩から見た，味方の持駒の 2 枚目の歩の評価値
10. KPP[44][1][2][1]：自玉がアドレス 44 にいるときに，味方の持駒の 1 枚目の歩から見た，味方の持駒の 2 枚目の歩の，手番を持つ側に加算される評価値
11. KPP[44][1][20][0]：自玉がアドレス 44 にいるときに，味方の持駒の 1 枚目の歩から見た，相手の持駒の 1 枚目の歩の評価値
12. KPP[44][1][2][576]：自玉がアドレス 44 にいるときに，味方の持駒の 1 枚目の歩から見た，アドレス 0 にいる味方の銀の評価値
13. KPP[44][1][2][657]：自玉がアドレス 44 にいるときに，味方の持駒の 1 枚目の歩から見た，アドレス 0 にいる相手の銀の評価値
14. KPP[44][576][577][0]：自玉がアドレス 44 にいるときに，アドレス 0 にいる味方の銀から見た，アドレス 1 にいる味方の銀の評価値
15. KPP[44][576][658][0]：自玉がアドレス 44 にいるときに，アドレス 0 にいる味方の銀から見た，アドレス 1 にいる相手の銀の評価値

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)
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！

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/>

■目標

まったりゆうちゃんを倒して師匠超えしたいです。

前回も直対してないので今年こそ当たらんかな・・・、なんかいつも勝ち星並ぶんですが、順位で上に行けない。。。

あとできれば久々に勝ち越ししてみたいですねー。

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。

臥龍 アピール文書 (第29回世界コンピュータ将棋選手権)

プログラム情報

- 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 :
  - 77勝116敗4分
- 開発言語 :
  - Java
- ソースコード行数 :
  - 思考部 20100行、UI部 10971行
- 採用している手法 :
  - αβ探索、反復深化、トランスポジションテーブル、null move pruning
  - 探索の延長 : 王手
  - 詰み探索 : 深さ優先探索
- 採用していない手法 :
  - 並列探索、進行度、評価関数の学習、df-pn
- 探索速度 :
  - 約30万nodes/s (シングルスレッド Core i7 4960HQ 2.6GHz)
- 評価関数パラメータ :
  - 駒損得、成駒、当たり駒、駒の絶対位置、玉の自由度、各枡の利き、駒の働き、囲い、ピン、手番
  - 合計 約750

開発者情報

- 開発者 :
  - 高田淳一
    - Twitter
    - Facebook
- 関連Web Site :
  - コンピュータ将棋プログラム「臥龍」

前回からの改良

- 時間制御を変更。

2019/3/16記

日本将棋連盟TOP ＞ 電棋士データベース ＞ カツ丼将棋

電棋士データベース

五段 (将棋倶楽部24基準、R2400)

桜井昇八段門下 カツ丼将棋

Katsudon Shogi

電棋士番号	-9999
生年月日	2013年4月20日(4歳)
出身地	東京都千代田区
開発者	松本 浩志
師匠	桜井昇八段
評価関数	簡易4コマ(予定)
探索	カツ丼フィッシュ

前回からの進化した点

すみません、今回は育児等に追われてほとんど進化してません。。ただし、大会を盛り上げ、もっと大会の様子や開発者の苦労や工夫を深堀するため、将棋倶楽部 24 の最高 R2800 の高段 youtuber の Sugar さんとカツ丼将棋席で実況放送します。今回はそっちの方面で盛り上げますのでよろしくお願いします。

評価関数

簡易 4 コマ(予定)です。簡便であってもとりあえず 4 コマにしておけばいいかなと思ったのですが、肝心の棋力が弱くなっているので、本番までに調整します。間に合わなければ 2 駒で。

簡易 4 コマとは、まず将棋ソフトは 2 駒でもそこそこ強くて軽くて扱いやすい。3 駒はポテンシャルが高いが玉の差分とか面倒である。そこで、ルートにおける玉玉の位置で呼び出す 2 駒を決めてしまうイメージ。玉玉の位置関係は 81x81≒1600 通り。これを真面目に考

えると容量が莫大になるため、将棋盤を 3x3 で 1 ブロックにしてやることで 81 通り。これで容量・計算は現実的なものとなり、一応 KKPP ということで俺は 4 駒だーと言える。

+手調整を加えています。手調整何てはやらない時代ですが、将棋というゲームで勝利に近づくには以下の要件が考えられます・・・(①相手より駒得をする。②敵玉に迫る)。教師局面が少ないと、②がうまく学習されません。しかし我々は②の敵玉に迫れば勝利に近づくと知っているので、手調整で強制的に加点してやります。あとは探索先生におまかせしておけば自己対戦では強くなった気がしました。

探索

StockFish 風の探索に実現確率的なのを取り入れようとしたり、いろいろ努力をしています。が、一つもうまくいったものがないので、アピールするに至っておりません。。

将棋倶楽部 24 自動対局システムカツ井坊

将棋倶楽部 24 における自動対局システム「カツ井坊」を開発しました。これは画像認識・マウス操縦を駆使して汎用将棋 USI を将棋倶楽部 24 に 24 時間延々と自動対局ができるようにするシステムです。このシステムをフルに活用して、将棋倶楽部 24 のレートとソフトレートの直接比較を行いました。その詳細な仕組みや結果は、2018 年 3 月発行の CSA 会誌に掲載しておりますのでぜひご覧ください！。

レートを計るということ以外にも、将棋ソフトと人間がたくさん対局をして交流を深めるという形で将棋界に貢献ができたと思っています。

カツ井神解析

将棋ウォーズに棋神解析というのがありますけれど、将棋倶楽部 24 で対局してどこの手が悪かったか即座に教えてくれる機能があれば便利ではないかとカツ井神解析を発明しました。棋譜を強いソフトに食わせて解析してもいいんですが、即座に教えてくれるということと、観戦者も勉強になるというのが利点です。

カツ井坊は、JKishi18gou、Hefewizen の開発者のたまちゃんさんにもお渡ししておりまして、下図が昨年の覇者 Hefewizen に実装されたカツ井神解析です。

Hefeweizen > 26 手目の△62 飛(82)がよくなかったようで、-267 点 ⇒ -520 点となってしまいました。

Hefeweizen > 代えて△44 銀(53)とすると以下の進行が考えられます。

Hefeweizen > △44 銀(53)▲66 銀(67)△34 歩(33)▲38 銀(39)△42 金(52)▲68 金(69)

Hefeweizen > 30 手目の△64 歩(63)がよくなかったようで、-497 点 ⇒ -722 点となってしまいました。

Hefeweizen > 代えて△42 銀(53)とすると以下の進行が考えられます。

Hefeweizen > △42 銀(53)▲55 銀(66)△53 歩打(00)▲38 銀(39)△31 金(32)▲78 金(69)

Hefeweizen > 以上、カツ丼神解析でした。
< Hefeweizen >さんは、去りました(^o^)/~~~

これは一例ですけども、こんな感じです。チャット機能を使っているので観戦者も勉強できるということです。対局中の最もな悪手、あるいは良手を 1 つか 2 つ表示するのですが、何をもって悪手とすればいいのでしょうか。-2000 点から-9999 点になるのが最悪手でしょうか？これはなかなか面白いテーマだと思いますけれど、カツ丼神解析ではこうやっています。

「*ソフト(t-1)手目、相手(t)手目、ソフト(t+1)手目、において、(t+1)手目と(t-1)手目に際し、評価値の変化を勝率の変化に計算しなおして、これを勝率インパクトとします。勝率インパクトの大きい順に並べている。ただし、t 手目が PonderHit の手は除外する。最善手が悪手と認定されるのはおかしいという考え。MultiPV は使ってないので対局に支障がないというのはメリットです。一方で正確性は少々下がるかと思います。*」

です。勝率は、勝率=1/(1+exp(-600＊評価値))、たしかこんな感じで計算できます。-2000 点の時点で勝率はほとんど 0 なので、ここから詰み筋が生じても勝率のインパクトはほとんどなく、本当に反省したい手が表示されると考えました。他にももっといいやり方があるかもしれません。

こちらは JKishi18gou に実装されているカツ丼神解析です。あたかも JK がしゃべっているかのような口調に、簡単にカスタマイズできるようになっています。

ちなみに、GutIshiatama というのが現在の評価値をつぶやいていますけども、これもカツ丼坊の機能です。対局中は対局者が観戦者にチャットすることができないため、別アカウントを使う必要がありました。これで観戦者は今の局面を簡単に把握することができて、見る

だけで勉強ができる仕組みになっています。

このように、ソフトをうまく使ってもっと将棋を楽しんでもらいたいということを取り組んでおります。この点において、コンピュータ将棋の発展に貢献しているとも言えますので、ぜひとも CSA 貢献賞をお願いします m(_ _)m

最後に

永遠のネタ勢として楽しんできます。

年表

2013 年 4 月　第 2 回電王戦　Ponanza VS 佐藤慎一四段戦でえりりんから客いじりで「カツ井さん」と呼ばれて将棋ソフトを「カツ井将棋」に決める。

2013 年 10 月 第 1 回電王トーナメントデビュー
初陣は Ponanza。ぼこぼこにされる。
カツ井定跡で古豪スケルツォを破り念願の初勝利
永遠のライバル Labyrinthus に恥ずしめ詰めされる。

2014 年 5 月　選手権で全敗。Libshogi に全駒ステイルメイトされて Wikipedia にも掲載される。

2016 年 5 月　選手権でブービー。さすがに恥ずかしかったので少し強くすることを決意

2016 年 10 月　少々強くしたが、電王 T で竜の価値を 0 点にしてしまううっかりで惨敗。
メカ女子将棋に一手詰めのバグのうっかりで敗北。

2017 年 1 月　将棋倶楽部 24 で R2500 弱の成績を収め、五段に昇段。

2017 年 5 月　世界選手権一次予選で 12 位で次点。

いつものおまけ。

●毎度のことで恐縮なのですが、将棋の歌が少ないと思って将棋の演歌を作曲しました。

曲名　千駄ヶ谷エレジー

歌手　長沢千和子女流四段

作詞　袋小路宇治夫

作曲　カツ井将棋

日本将棋連盟推奨、桜井昇八段推薦

もしよかったらどうぞ。最近は長沢先生が老人会に遠征したりしているそうです。

Itune store、レコチョクから DL できます。itune store から購入可能）

(2019.3.19追記)

＊プログラム名称

Ichibinichi(イッヒビンイッヒ)、ドイツ語で「俺は俺だ」をイメージしたつもりです。昔は、「Wildcat」という名前で出ていました。

＊開発言語

javaです。Javaでの参加は２回目です。

＊プログラムの特徴

これぞという工夫がないのが残念です。古典書のような内容です。

探索部分は、αβ法を基本に、全幅探索、aspiration search、futility cut、null move search, 多重反復深化、並列探索（検討中）、ポンダー探索などで組み立てています。ビットボードは使っていません。

評価部分は、いわゆるKP＋王回りの利き＋悪形排除のための簡易型３駒関係（序盤のみ）の３つです。プロ棋譜33000を使ってボナンザ法で学習した評価関数（A）と、下記の参考文献を参考に、水平線効果の出現を極端に制限した特別バージョンを作って約10万の自己対戦棋譜を作成し、プロ棋譜では出現稀な局面を玉の盤面上の位置から判断して選出し、ボナンザ法で学習し?た評価関数（B）を作成しています。評価関数（A）と評価関数（B）をブレンドして評価関数（C）をつくり、これを最終的な評価関数にしています。

（参考文献）

・林伸也、浦晃、三輪誠、田浦健次郎、近山隆。自己対戦棋譜を利用した半教師あり学習による将棋の評価関数の学習。第16回ゲームプログラミングワークショップ2011

＊＊
＊

＊＊
＊

以下は、開発月報です。ご参考までに。

(5月)

・大会では３勝５敗の成績で１次予選敗退。Java、並列化なし、評価関数（要素数は約１万）は４年前作成のものの使いまわし、貧弱マシンパワー、まあこんなもんだなあという感じ。

・次回大会の目標を「１次予選で勝ち越し」ぐらいかなとボヤーと決める。

・データ構造を含めて、抜本的な見直しからの作り直し作業の開始。move生成のところで、前バージョンより２割弱のスピードを改善。

（６月）

・差手生成にバグはないか念入りに調べるために詰将棋用のプログラムを作成開始。桂馬、香車の非合法手生成のバグをつぶす。前年作成のｄｆｐｎを小修整して移植する。詰プログラムを作ったところで小休止。

（７月）

・オンライン学習に挑戦するために方策勾配法についてちょっかいを出し始める。下記の論文を目を皿にして読んでみるがチンプンカンプン状態（ボルツマン分布なんて、大学の授業だけの世界だと思っていたのに、こんなところで再会するとは…。あんまり真面目な生徒ではなかったからなと、今になってもっと勉強しておけばと後悔）。そうこうしているうちに忙しさにかまけて、コンピュータ将棋開発から一時的に遠ざかり始める。

（８月）

・今年の夏は異常に暑い。お盆は家族でカナダ旅行。仕事がメチャクチャに忙しくなり、しばらく休戦状態。

（９月）

・大槻知史さんがWEBで公開している「迷路の強化学習」には、C言語のソースコードが付属していた。これを自分なりにJavaに書き換えて走らせてみる。いろいろとパラメータを変えてみる。するとおぼろげながら方策勾配法とは何ぞえといういうことがわかり始める。再度、将棋関係の方策勾配法の論文再読を始める。なるほどなるほどだ。

（１０月）

・これとは別に、何年か前にボナメソで評価関数をつくるプログラム（ｃ/ｃ＋＋）があったので、javaに書き換えて、評価項目数を１００万程度に増やして走らせ始める。（ちなみに２９回大会バージョンは、KPの相対位置評価だけで項目数は約１万。）序盤の駒組はうまく行っているようなので学習はしているようだが、いまひとつシックリしない。

・大槻知史さんがWEBで公開している「迷路の強化学習」について、結果が収束しない。どうしてだろう。書いたプログラムは何回も見直した。しかし間違いはなし。１週間ほど、考え抜いたが理由はわからず。ただし、方策勾配法とは何ぞえには手ごたえを得る。

・鶴岡先生（劇指作者のひとり）の選手権優勝記―劇指の技術的改良の解説―を参考にプログラミングを開始する。

・最近、仕事がゴタゴタしていて１週間に１～２時間しかコンピュータ将棋にタッチできない。それに、眼球に疲労を感じる。今、一番、欲しいものはハイパフォーマンスのPCとその電気代、そして２４時間モニターを凝視できる頑強な視力かな。

（１１月）

・ボナメソによる評価関数でバグがボロボロと見つかる。致命的なのは、静止探索に入った後の読み筋を考慮していなかった。なんでこんな基本的なところで間違うのか。自分の注意力不足に愕然とする。静止探索に入る前に読みを打ち切っても学習は一応するはずだ。やけにトータルの学習時間が短いなあと思っていたが原因がわかった。

・「選手権優勝記―劇指の技術的改良の解説―」を細々とコード書き。考え自体がシンプルだから、意外と、早い。

・javaに限界を感じる。#define , #ifdefなどが使えないのがもどかしい。 c と比べるとマニュアル車とオートマチック車ぐらいの違いかな。

・大会事務局から連絡を受ける。ライブラリー使用が厳しくなる。私は他人様が作ったコードを丸ごと使うことには興味がない。富士山にヘリコプターを使って登っても楽しくないのではないだろうか。

（１２月）

・コンピュータ将棋大会開催事務局から、ルール変更の通知をもらう。思考時間の管理は、もう少しあとからやろう。

・大会用のハードウエアについて考慮し始める。ぜひとも新マシーンでチャレンジしたいものだ。

・生まれて初めて病院に入院する。１泊２日の検査入院。体のガタは日増しに増加、脳みそも同じように劣化しているかと思うと寂しいものだ。検査結果は癌にはノーヒット。

・教師なし学習について試行錯誤を続ける。もう半年は道草を喰っているなあ。理論上はうまくいくはずなのだが、プログラムにして動かしてみるが思い通りに行かない。次回大会も、ボナメソで参加濃厚。進歩なしか。

・コンピュータ将棋大会開催事務局から、大会開催の案内を受け取る。参加の手続きを行う。

（１月）

・家族で台湾旅行。楽しかった。大会参加費を新年早々に振り込む。1万円が消えた。５０００円の食事を２回喰ったと思って我慢。

・キラームーブの合法性をチェックするところで、前々からバグらしきものの存在を感じていたが、やっと明らかになる。気分的に少しすっきりした。

・探索の並列化を模索。例によってうまくいかない。

・並列探索がなんとか動くようになる。並列に探索していることは確かなのだが、なぜか速度があがらない。２スレッドで走らせても、１スレッドと変わらず。むしろ並列のためのオーバヘッドがかかって、遅くなってしまっている。これってJavaの宿命なのだろうか。ネットで調べると、物理メモリーの量が影響するとか。開発用にsurface pro 3（４GB）を使っているが、メモリーがあまりに貧弱なのが原因だろうか。

（２月）

・１０秒将棋の自己対戦棋譜を眺めて愕然とする。救いようがない指手ばかりだ。

・並列探索と格闘。動作が今一つだ。今のままでは、実戦投入には、ちょっと無理。

（３月）

・並列探索は、ひとまず休憩。先読み探索を考えてみる。これはすんなりとうまくいく。

・なんとこの時期になって、差し手生成のところでバグを発見する。自玉王が先手の場合は最上段（後手の場合は最下段）にいる時に、王手を掛けられた際の合い処理で動けない歩打（非合法手）を行って

いた。王手対応の処理がいい加減でした。めったに起こらないから今まで見落としていたようだ。

（今後の予定）

５月の大会までにやりたいことは、並列探索のブラシュアップ、LMRや各種パラメータの調整ぐらいかな。結局、強化学習は、半年ねばったができなかった。

平行してFloodgateに投入して武者修行も必要かなと考えています。それでは大会でお会いしましょう。川崎の産業会館へは歩いていけますので、とても楽です。

こまあそび
アピール文書

2019/03/29

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２９回世界コンピュータ将棋選手権　アピール文章

オッズの魔法使い　Season　２ -　Random Forest

作者　David Wada

〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜

０．Java　から　Rust　に乗り換えようと思いましたが、時間足らずで　来年に見送り　となりました。

１．今年は以前試した　Randomized Parallel　Best First　Minimax　Search　を採用。以前より賢くなりました。
それとも以前の実装が貧相だったと言うべきか

Best First　Minimax　Search
http://maxchickering.com/publications/aij96.pdf

Randomized Parallel　Best First　Minimax　Search
https://pdfs.semanticscholar.org/207b/87002a3d5785758e60870ec8835d4de259bd.pdf

評価関数の精度がここまで上がれば枝刈りをせず、単純に　最良優先探索　で十分なのでは？…と思う次第です。
これは　良さそうな手を進めて読む　人間の思考と酷似します。

但し、最良優先探索だけだと千日手ループに陥るのでランダムを仕込みます。

このアルゴリズムは並列探索向きです。CPU投入した分、効率が上がります。制限は予算のみ！

並列探索にはJPPFを使用しています。

https://jppf.org/

２．評価関数は　謹製Apery　に変更…ですが、

- 手番の加算は省略
- パラメターのみ使用

作者様に感謝です。

（（使用・選択理由））

それまではボナンザ使用でしたが、新しい評価関数に交換しました。
他の人がやらない探索が重点なので評価関数は外注…と言うことで。

３．定跡は　Qhapaq定跡（二代目）　を拝借。

http://qhapaq.hatenablog.com/entry/2016/12/14/222645

作者様に感謝です。

４．クラウドのマシン使用

AWS使用。一応、4台構成を予定しています。

〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜

2019年3月24日　ー　評価関数の使用理由

2019年3月23日　ー　初版

Miacis アピール文書

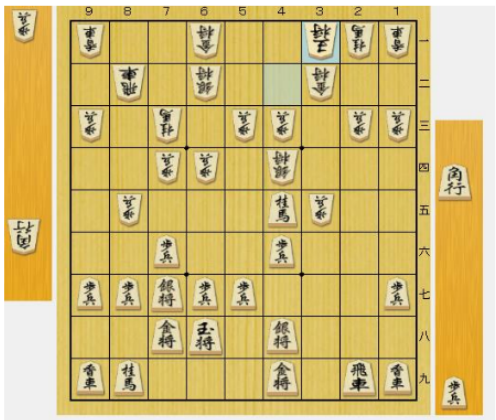
迫田真太郎

April 12, 2019

Miacisの特徴

◆ 評価値を確率分布として出力

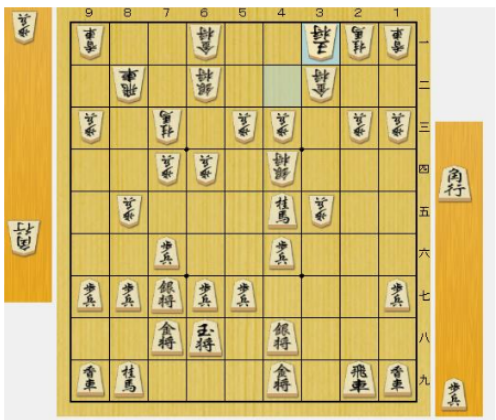
従来の大多数



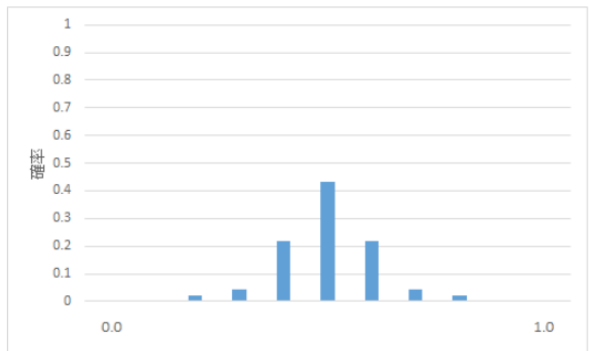
評価関数

予測勝率 0.55

提案手法



評価関数



背景

- 将棋ソフトにおける評価値 $\fallingdotseq$ 強化学習における状態価値
- 状態価値は「ある方策 π の下での累積報酬の**期待値**」
- 累積報酬の**分布を直接近似**する→分布型強化学習
  - 森村らの研究[i]
    - 累積報酬の分布からリスクを考慮した行動決定を実行
  - Categorical DQN[ii]
    - 深層強化学習においてカテゴリカル分布による累積報酬分布の近似を行う
  - Windfall[iii]
    - 形勢の分布を考慮した将棋ソフト

[i] Tetsuro Morimura, et al. "Parametric return density estimation for reinforcement learning." arXiv preprint arXiv:1203.3497 (2012).

[ii] Marc G. Bellemare, Will Dabney, and Remi Munos. "A distributional perspective on reinforcement learning." Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017.

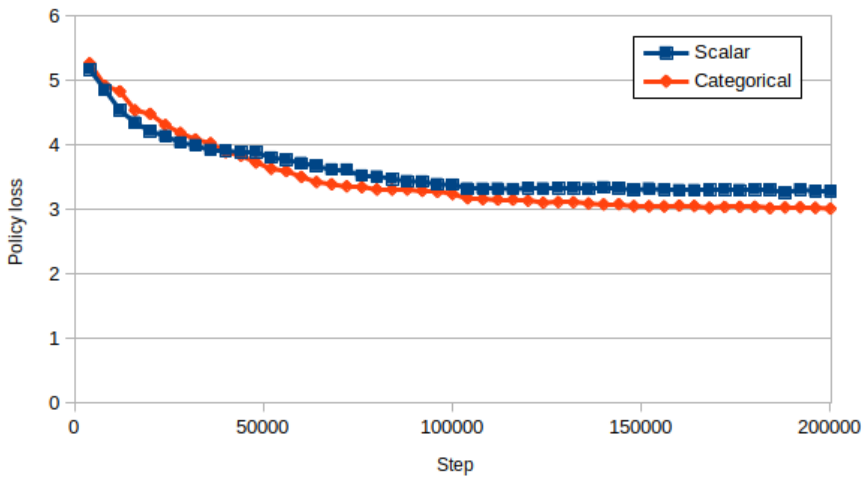
[iii] http://denou.jp/tournament2017/img/pr/windfall.pdf

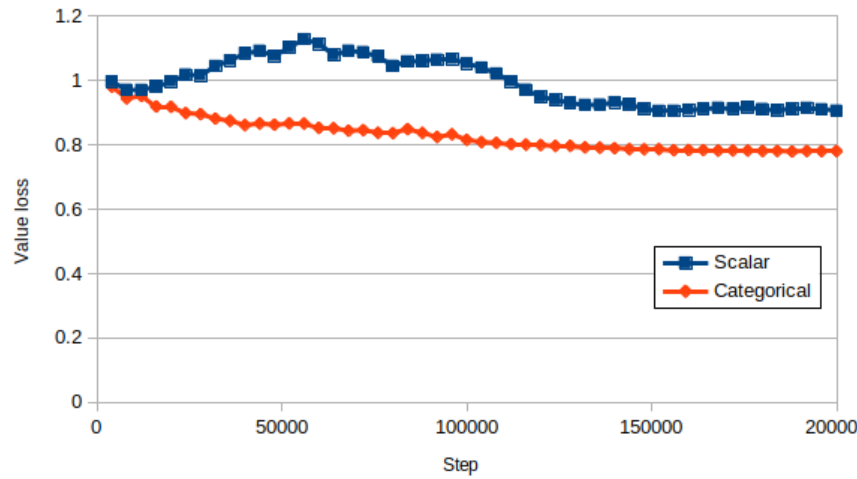
提案手法

- **評価値を確率分布として出力**
  - AlphaZero **モデルの** Value **を複数ユニットに拡張**
  - Categorical DQN **を参考に報酬** [-1, 1] **を** 51 **分割**
    - **状態価値が各値となる確率を出力**
- **確率分布を用いた効率的な探索**
  - MCTS **の選択ステップで良い価値となる確率を考慮**

実験結果(学習推移)

- ◆ 強化学習により自己対局から分布を学習
- ◆ 分布としたことで評価精度向上
  - floodgate の R2800 以上同士の棋譜に対する損失





実験結果 (対局)

- **技巧** 2[iv] (**深さ** 1, R1506)[v] **と** 1 **手** 1 **秒で** 100 **局対局**

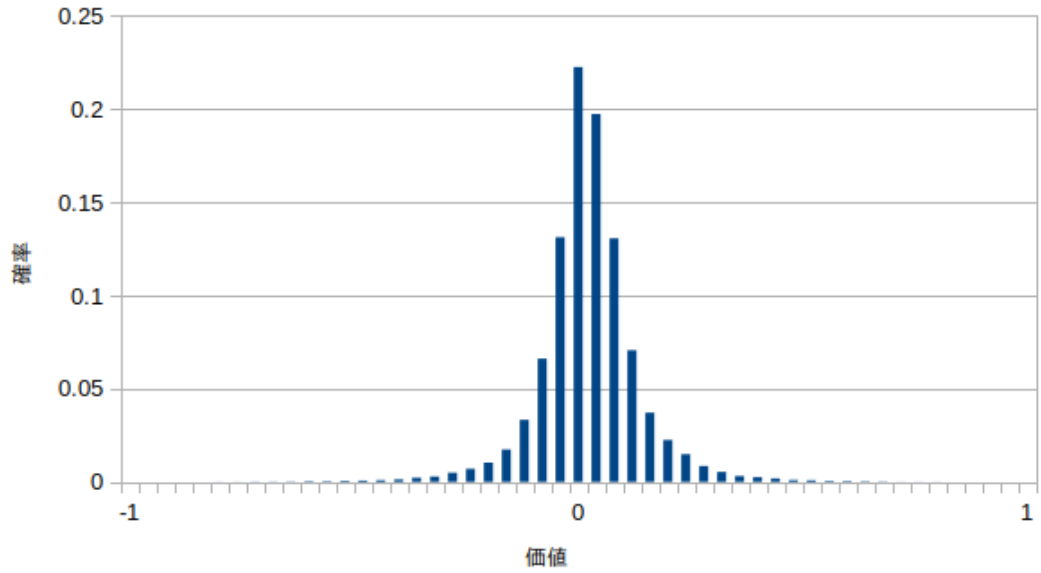
出力	勝率 (%)	推定レート
Scalar(**従来手法**)	10.0	1124.3
Categorical(**提案手法**)	72.0	1670.1

- **全体で** R+545.8 **の性能向上**
  - **評価精度向上で** R+468.1
  - **分布を考慮する探索によって** R+77.7

[iv] https://github.com/gikou-official

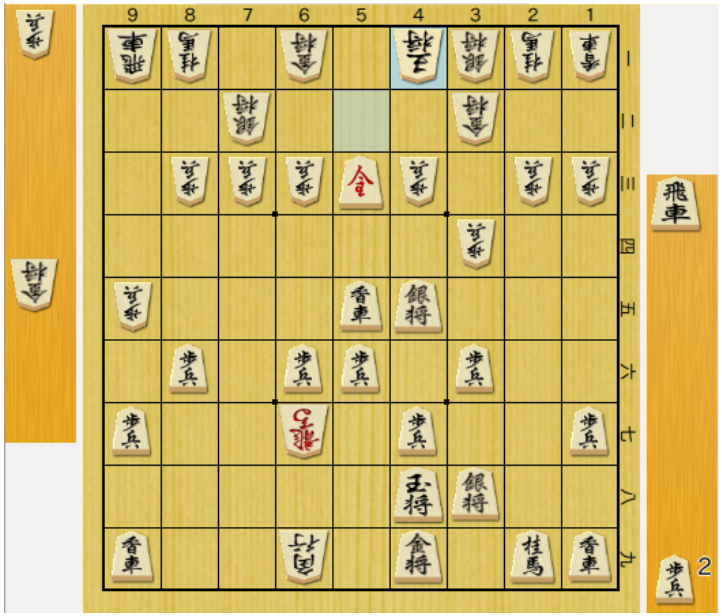
[v] http://www.uuunuuun.com/english

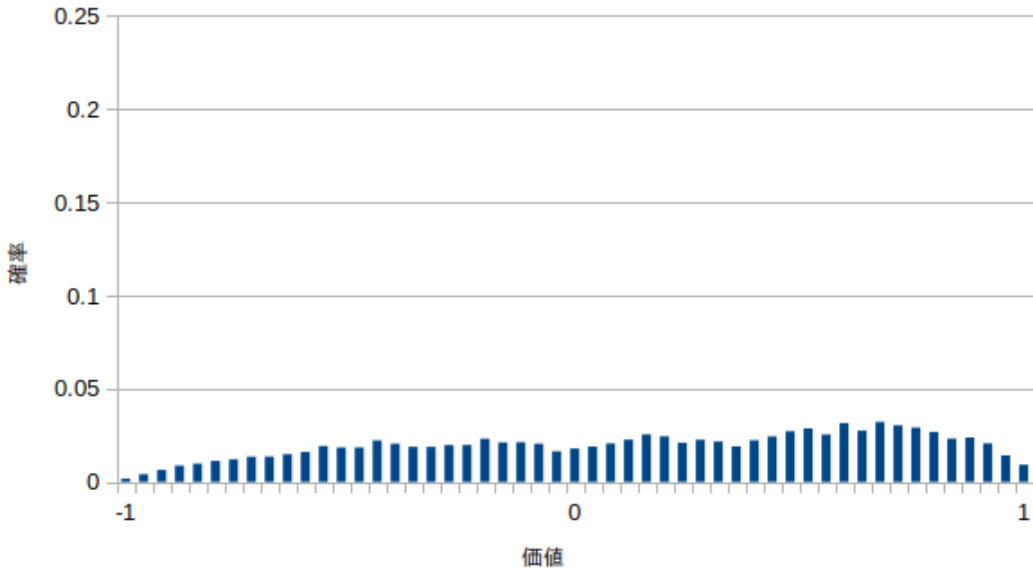
出力例 1



◆ 初期局面では 0 付近に高い確率を示す

出力例2

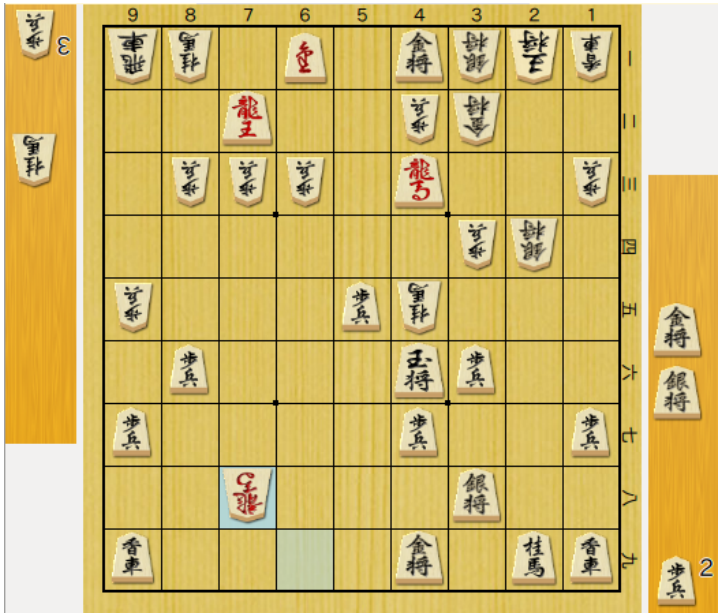


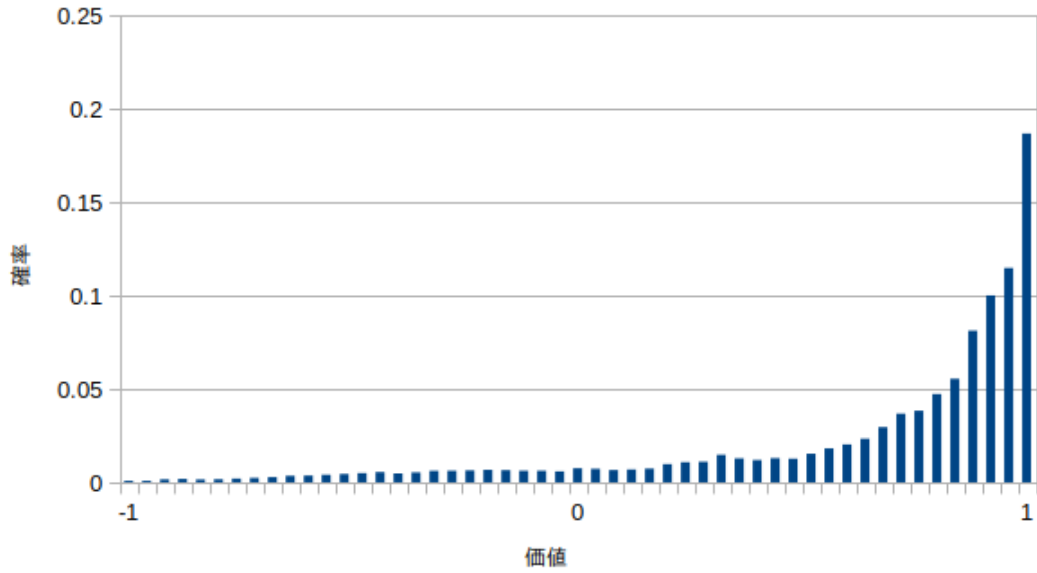


◆ **中盤の難所 (?) では、期待値はほぼ 0(互角) だが幅広い値に確率を示す**

- **「難解」あるいは「形勢不明」という判断？**

出力例3





◆ 勝ちになった終盤では 1 付近に高い確率を示す

SMS将棋のアピール文書 2019年

ライブラリの中かられさぴょんを選ばせて頂いた理由は、

ソースコードが理解しやすかったからです。

工夫した点は、稲庭将棋のアルゴリズムを採用した所です。

また、評価は手作業で行いました。

Windfall

第29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：2019/3/吉日

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

- 今回のWindfallは以下の要素で成り立っています
- 仕様：
  - 合法手からランダムに次の手を選ぶ
  - (実装が間に合えば、評価値が最良のものを選ぶ)
- プログラミング：
  - TensorflowからPytorchに変更
  - 局面の遷移、合法手の計算をYaneuraOuを参考に実装
- その他：
  - なぜか去年よりさらに後退した進捗
  - 一緒に開発してくれる方を募集中

去年も書いた

わからなかった方は、次項以降をどうぞ

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

去年より-3歩前進

- DNN関係で大幅な見直し
- この1年での変遷
  - 状態行動価値関数から状態価値関数に変更
    - DQNは行動の種類が多過ぎて無理
  - 大駒の利きを考慮するためには畳み込み層が多く必要なので、全結合層で遠くの駒も考慮して計算
    - 次元の呪い やばい
  - 長い利きの方向ごとに畳み込みを計算(例えば1x9のカーネルで畳み込み)
    - このタイミングでPytorchに変更
      - gatherやscatterがいい感じ

長い利きの方向ごとの畳み込み

- 広い範囲の畳み込みを行うことで、レイヤーが少なくても学習できる
  - 十字飛車を考えるとかなり広い範囲の計算が必要なので、3x3のカーネルでは辛そう
- 例えば、縦方向なら1x9のカーネルでストライドは0なので、実質的には全結合
- ラインごとに別のカーネルで計算
  - ラインとは、例えば、縦方向なら一つの筋のこと
  - 全て同じカーネルでは制約が強すぎると思う
  - gatherで集める→畳み込み(1x1 →1x9 →1x1)→scatterで9x9に並べなおす
    - メモリのコピーがかなりボトルネック
  - gather, scatterと同時に1x1の畳み込みを実行
    - pytorchのextensionでカスタムレイヤーを自作
    - ボトルネックは少し改善したが、各ラインの処理が逐次的に実行されるのであまり速くない

評価関数について

- Residual Networkを採用
  - 持ち駒を入力データのチャネルではなく、畳み込み後のバイアス項に対して計算
- 浅いネットワークから徐々に深くして学習
  - 同時に出力も大雑把なものから精緻なものへ段階的に学習
  - 長い利きを考慮した畳み込みレイヤーを使って学習
    - 浅いネットワークでも長い利きを学習する機会があるのはメリットだと思う
- 現状は挙動が明らかにおかしいので、間に合わないかも
  - 本番用PCで動かないのはつらい
  - ユニットテストは通るが、DNNに組み込むとエラー

探索について

- 今年もそこまで手が回らなかった
  - 今年は、合法手の中からランダムか評価値が最も高いものを選ぶ
- 本来の実装したい手法の詳細は最低限のものが出来上がった時に書きます
  - 過去のアピール文書で論文を参照しているので、どうしても気になる方は探してください

大会後の方針

- 長い利きの畳み込みレイヤーが動くなら
  - 本来の探索アルゴリズムの実装
  - DNNを深くしながら逐次的に学習
- 動かないなら
  - GPUは面倒が多いので、NNUEを改良しようかな
    - policyベースの学習とか面白そう
    - NNUEの本質は3駒以上の特徴量を手に入れたことで、非線形性ではないと思う
    - 具体的な方針は未定

2019年アピール文書

プログラムの基本は初回参加時から使いまわしていて

一般的な手法であるαβ法、ハッシュテーブル、null move等を使っています。

今回は、去年からの学習方法から一部変更して教師無し学習の３駒間の評価値を学習させています。

学習方法は強化学習(Q-Learn)のみです。

****

2018年アピール文書

プログラムの基本は初回参加時から使いまわしているものを使っています。

今回も、教師無し学習にて３駒間＋２駒間の評価値を学習させています。

学習は強化学習と遺伝的学習を組み合わせています。

前回に比較して大きな変更はありません。バグ修正レベルの変更です。

****

2017年アピール文書

プログラムの基本は初回参加時から使いまわしているものを使い、

今回(前回も)は、教師無し学習を試してみたくなったので

強化学習と遺伝的学習を組み合わせて作成している。

去年は、時間の都合で学習がほとんど進まなかったので

既存の評価値データでの参加だったが、

今回は学習に時間が取れそうなので学習した評価値での参加を考えている。

序盤については、以前から持っているの序盤データをそのまま使用する予定。

たとえ、以前のものよりも弱くなっていたとしても、

学習したデータで望むつもりである。

第２９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

２０１９年０３月０３日　高橋智史

THREADRIPPER

スレッドリッパーが　ただの箱になってしまうわよ

DARE

誰なんだぜ☆？

開発者
高橋　智史

「スレッドリッパーを買ったんだが、
きふわらべ　が将棋をルール通りに
指せないので何も始まらない……☆」

北白河ちゆり
／TOHO PROJECT
FANMADE　※

コンピュータ将棋エンジン
きふわらべ

「いのしし　を貫通しておいたぜ☆
３手目で　５段目に　いのしし
が居るから気を付けろよ☆」

「探索部だけ　ライブラリ　を使って、
思考部を作り始めた方が
いいんじゃないの？」

岡崎夢美
／TOHO PROJECT
FANMADE　※

この黒い箱の中にある
ヒートシンクの下に埋まっている☆

※北白河ちゆり、岡崎夢美は　東方夢時空　の登場キャラクター／
©上海アリス幻樂団　様の著作物です。

＼CHECK／

「借りてきた新幹線の自由席に　きふわらべ　が腰かけていたら
そいつは　きふわらべ　が速いのではなく　新幹線が速い……☆

今年の　きふわらべ　は　ただの箱　なのか、
まだ作っていないが　これから作る予定の　今年の一発ネタを
Ａ４　１９ページ　にわたって　紹介していこう☆」

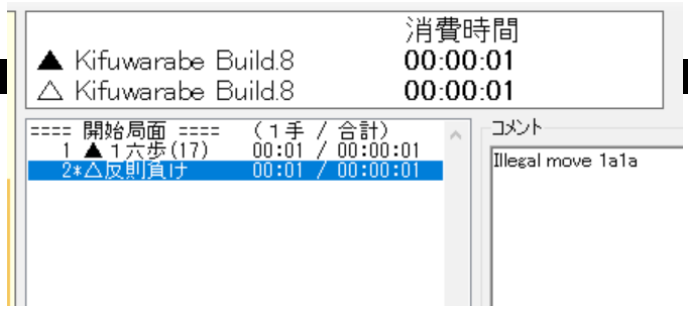
らーすぅー

Ｒｕｓｔ は今や きふわらべのフェイバリット だぜ☆

１年も　使い続ければ　UMAKU 上手く　なってたりしないの？

// Shogi dokoro

「つら……☆
指し手生成部
また書かないと
いけないのか……☆」



「右上端にあった駒を　右上端に移動だぜ☆」

```
// 移動先を開始地点にして、駒の位置を終了地点にする
use kifuwarabe_movement_picker::KmDir::*;
match *p_kmdir {
    // 東
    E(slider) => if slider {
        // 長東
        for i_east in 1..9 {
            if dx + i_east < SUJI_10 {
                let ms_src = suji_dan_to_ms(dx + i_east, dy);
                if gen_ky.has_ms_km(ms_src, km_dst) {
                    result.insert(ms_src);
                } else if gen_ky.exists_km(ms_src) {
                    break;
                }
            }
        }
    // 東
    } else if dx + 1 < SUJI_10 {
        let ms_src = suji_dan_to_ms(dx + 1, dy);
        if gen_ky.has_ms_km(ms_src, km_dst) {
            result.insert(ms_src);
        }
    },
    // 北東
    NE(slider) => if slider {
```

// Rust　Kifuwarabe WCSC28 (2018-05/Shogi)

「去年の　きふわらべ　ちゃんを使えば？
ライブラリ利用には　当たらないわよ？」

「じゃあ　去年のきふわらべ　を
アピールするのも　あり　では……☆
ランダムムーブだけど……☆」

// Kifuwarabe AIR2018 (2018-12/Go)　Rust

「囲碁に　走り駒　とか
無いだろ☆
何につまづく☆？」

「囲碁は　囲碁で
連の計算の遅さに
つまづいたりした☆」

```
// TODO 石が隣接していれば、連が変わる☆（＾～＾） 0～4つの連が隣接している☆（＾～＾）

// 連がつながるか調べたいので、自分の色と比較☆（＾～＾） 上、右、下、左。
// - [v] 連のIDの更新。
// TODO - [ ] 連の要素の更新。
let mut small_id = target as i16;
small_id = if pos.get_board().get_stone(top) == pos.turn {
    println!("Do move: 上とつながる。");
    // 置いた石と、隣の 連ID を見比べて、小さなID の方で塗りつぶす。
    refill_piece_distribution(target, top, pos)
} else {small_id};
small_id = if pos.get_board().get_stone(right) == pos.turn {
    println!("Do move: 右とつながる。");
    refill_piece_distribution(target, right, pos)
} else {small_id};
small_id = if pos.get_board().get_stone(bottom) == pos.turn {
    println!("Do move: 下とつながる。");
    refill_piece_distribution(target, bottom, pos)
} else {small_id};
small_id = if pos.get_board().get_stone(left) == pos.turn {
    println!("Do move: 左とつながる。");
    refill_piece_distribution(target, left, pos)
} else {small_id};
```

```
loop {
    // Standard input.
    // Be sure to add "info" before the output message.
    let mut line = String::new();
    io::stdin()
        .read_line(&mut line)
        .expect("info Failed: stdin.read_line.");

    // Excludes trailing newlines. The surrounding white
    line = line.trim()
        .parse()
        .expect("info Failed: stdin parse.");

    if line == "quit" {
        break;
    } else if line == "usi" {
        comm.println("id name Kifuwarabe Build.8");
        comm.println("id author Satoshi TAKAHASHI");
        comm.println("usiok");
    } else if line == "isready" {
        comm.println("readyok");
    } else if line == "usinewgame" {
    } else if line.starts_with("position") {
```

// Rust　Kifuwarabe WCSC29 (2019-xx/Shogi)

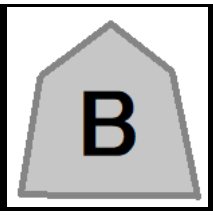
「コンピューター将棋イベントに　８回　も
選手出場してんのよ。
なんで　強くならないの？」

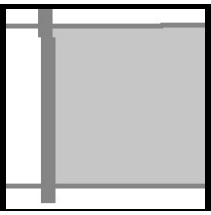
「指し手生成部　ばかり書いてるからな☆
探索部　とか　評価部　は
まったくだぜ☆」

ISOU

位相　から作りましょう

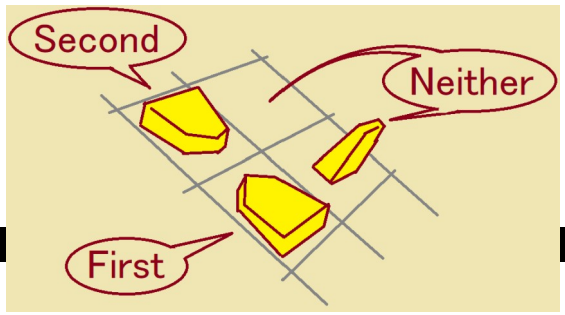
SENTE 先手 と GOTE 後手 と DOCHIRA DEMO NAI どちらでもない だぜ☆





```rust
#[derive(Clone, Copy, PartialEq)]
pub enum Phase {
    First,
    Second,
    Neither,
}
```

「もし将棋の神様が居るとして、
将棋の道具箱　の中で　１番最初に
創られたもの　が有ったとしたら、それは
きっと　先手、後手、どちらでもない
の　３つ　だと思うんだぜ☆」

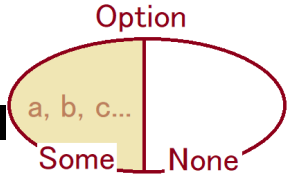


// Player

「どちらでもない　とは
何なのか……☆」

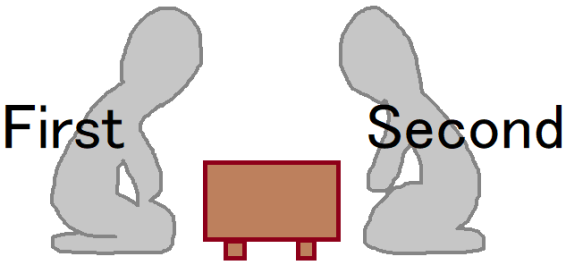
「空きマス　とか、　振り駒で立った駒　にも対応できるように
不測の事態を　表現できる　プログラムにしてあるのよ。
だからバグるのよね」

「どちらでもない　が有るのは　構わないが、
Ｒｕｓｔ言語で　無い　を表現するなら　Ｏｐｔｉｏｎ　を
使わないと 関数型プログラミング の簡潔さが生きないぜ☆」



// Option

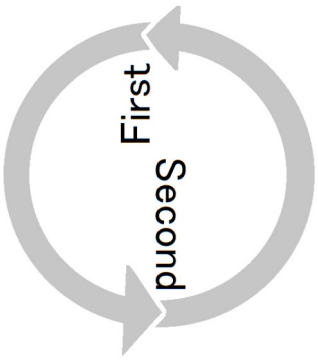
「将棋を知らない　ヒト科の動物　が
対局を見たら、　２人　が座りながら
寝ている、と思うかもしれない☆」



「たまに　駒を動かしているでしょ！」

「この交互に指すという動きは
歯車が食い違うことがない☆
つまり　手得　は貯金できる☆

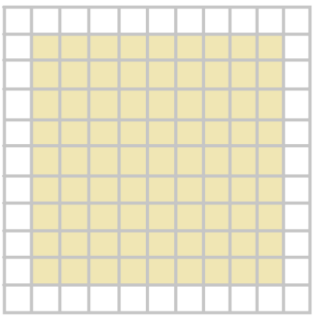
このような　盤面　に見えない　時系列　の
評価　を目指すのも方向性としてはあるよな☆」



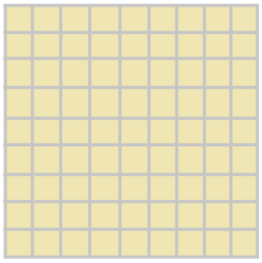
NIJIGEN

二次元　を　２人で　挟もうぜ☆

ARI
枠あり　なの？



NASHI
枠なし　なの？



「再帰関数で盤面をスキャンするときの番兵（ストップ）に使うのが　枠　だぜ☆

今回は　枠なし　で☆」

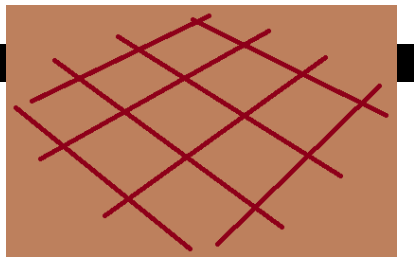
```
pub const FILE_LEN: i8 = 9;
pub const RANK_LEN: i8 = 9;
pub const BOARD_SIZE: usize = (FILE_LEN * RANK_LEN) as usize;

pub struct Position {
    pub board : [Piece; BOARD_SIZE],
}
```

「将棋の道具箱で　２番目に作られたのは　きっと　盤　だろう☆」

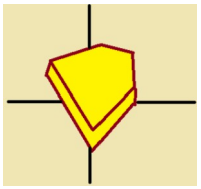
```
// Board
```

「二次元の直交座標は　そこらへんにあり、
２人が　それを挟んで向かい合ったとき　盤　になる。
部屋に置いているだけでは　ただの板　だぜ☆」



フロアタイル、金網、ハニーワッフル、
京都、並んだ本棚、スマホの画面、etc.

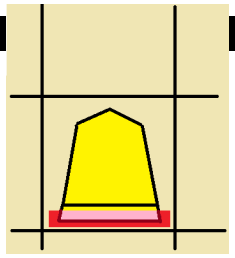
「そんなん　どうでもいいのに……☆」

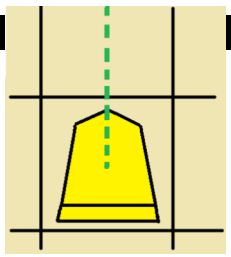


「盤上の　升目の中に　向きを揃えて置かれたときだけ、
駒 になるんだぜ☆ 持ってるだけでは ただの板☆ 」

「五角形なら　なんでも
将棋の駒　なんじゃないの？」

「下の線を揃えると
まっすぐ　上を向くぜ☆」





「この駒の形が　本将棋から
八方桂　を廃れさせたんだろ☆
ホモサピエンスサピエンスが 休暇の ON／OFF 分けたがるのも、
寝てる姿と　起きてる姿で　差が大きい　動物　だからだぜ☆」

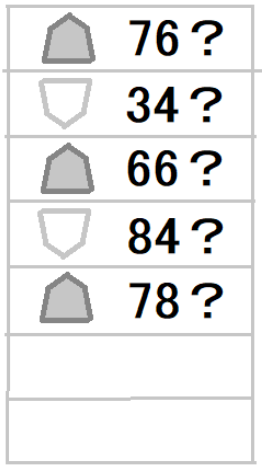
KIFU

駒の形より　棋譜　が大事なのよ

ISHI　か　KOMA　かより　でかい違いが　KIFU　だぜ☆

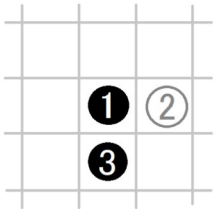
```
pub struct Record {
    pub items : Vec<Move>,
}
```

「１番目の　先後　と、
２番目の　盤上の升　の２つがある
棋譜　の創造が　３番目、
４番目は　駒のシープ　だぜ☆」



// Record

「囲碁の棋譜　みたいにすれば　いいのに……☆」



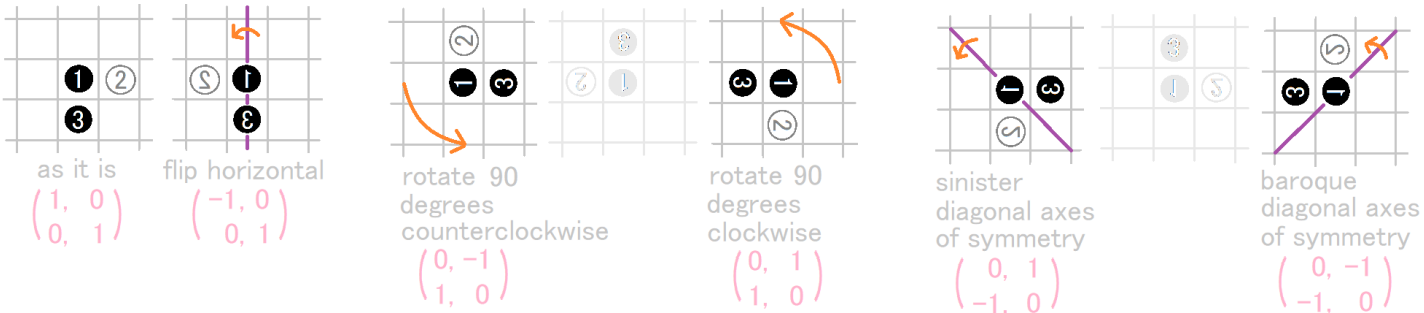
「棋譜　と　駒（シープ？）　は逆じゃないの？」

「ツイッターを眺めていると、
動かないのが石、動くのが駒
と 書いてる人がいたぜ☆」

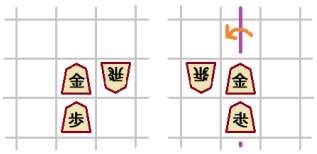
「造物神は　製造業じゃないんで☆
仕入れのことは　気にするなだぜ☆」

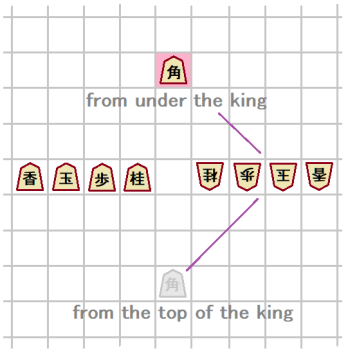
// Shape

「数えてみたが、囲碁には　８つ　の
同じ位相の　isomorphic　アイソモーゥフェック　がある☆」



「将棋は　たった　２つ☆」





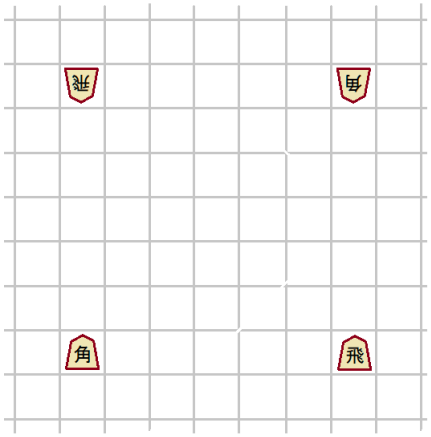
// Symmetric

「まったく同じ局面　が繰り返されると　千日手　で
盤は　同じこと　を許さないが☆、」

「盤上の対称形に　禁じ手　はない☆
盤は　アイソモーゥフェック　を許している☆」

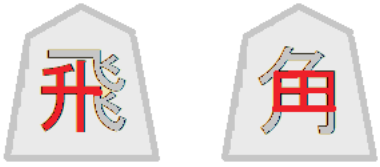
「しかし
左右対称は　将棋の　コォンジュゲーション☆
選ばなかった　共役(きょうやく)指し手　が　１つ☆
２つの　解(かい)　を　べつべつに研究しても
根(こん)　なものは同じだ、となってしまうぜ☆」

「無駄ねぇ」



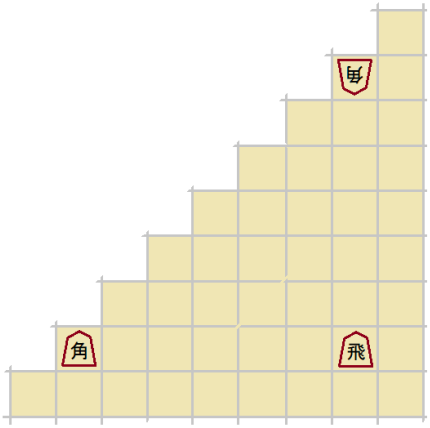
// Rook and Bishop

「初期配置の　角と　飛車は
コォンジュゲーション　を研究せずに済ますための
配慮　にも思えてくる……☆
なぜ　角と　飛車なのか……、と考えていたら☆」



「升田　が出てきた☆」

「升田さんは　関係なさそうねぇ」



「もし　根　に配慮するなら
ナナメ対称　にするべきだと思う☆」

「ライブラリーを使って
さっさと　評価部　に取り組んだらいいのに……☆」

IKUTU MONO KOTO

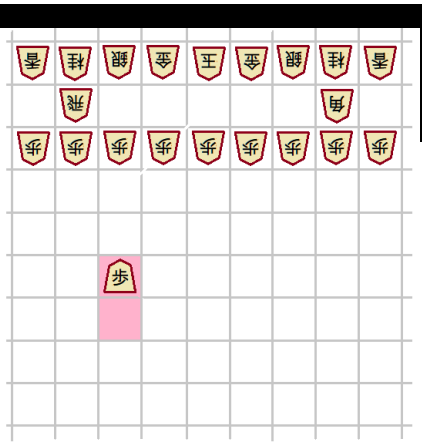
１度に　いくつものこと　をやれだぜ☆

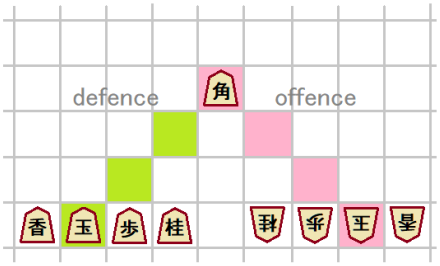
将棋のルール通りの　次の一手（TUGINO ITTE）　を選びなさいよ！

```
} else if line.starts_with("go") {
        let thought = Thought::new();
        println!("bestmove {}", thought.get_best_move(&mut position).to_sign());
}
```

「 創造の５番目は　一対多（いちたいた）☆」

「 もう　将棋用語　でもなくなった……☆」

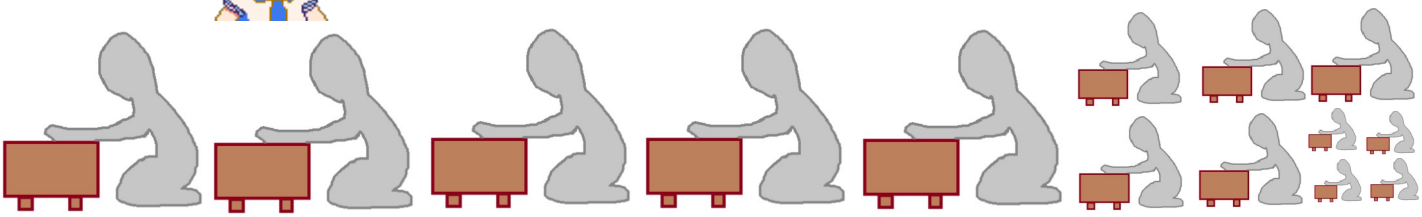




「 大勢相手に　動ける駒は１つの
ことでしょ」

「 １手で　攻防にも
視聴者にも　利かす、
ということだぜ☆」

「 １２時間かけて　１３０回　駒を前後させるゲームなら
一局やれば　満足して　止めてしまうだろう☆」



「 べつに……☆　駒しか　動かしていないが……☆」

「 コンピューター将棋は　評価部、　探索部　の２本足にすることが
最も　効率的な　資源の使い方だそうだぜ☆
わたしは　電車を速くするのではなく　地面を短くするために
将棋の指し手　と　共役　である対称的な　小さな空間　を
自作して、　そっちを　評価、探索　する狙いだぜ☆」

「 数学的には　そんな空間が無いのは１秒で分かるそうよ」

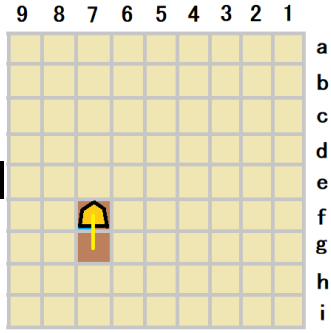
FUGO

探索空間の広がりは　符号　の掛け算なのね

式は　ALGEBRA　アルジェブラ　の始まりだぜ☆

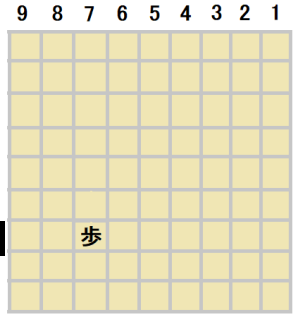
「わたしが　マァーテマティーシエン　数学者　じゃなくて良かった……☆
できないことが　分からないので　いつまでも　考え続けることができる……☆」

「時間が安い……☆」



// 7g7f

「創造の６番目は　FUGO　符号☆
将棋を　数式　で書いている人がいないので　わたしが考える☆」



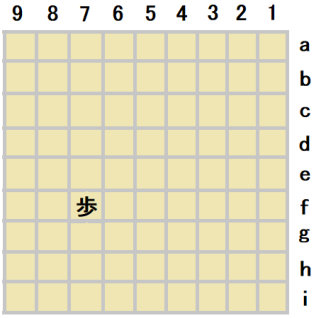
「局面　は　SU　数☆
歩が１個の局面は　フォーサイス エドワーズ記法で
以下のように書くぜ☆」

9/9/9/9/9/9/9/2P6/9/9 -

9/9/9/9/9/9/9/2P6/9/9 - × 7g7f

「局面　に　指し手の符号　を　KAKEZAN　掛け算　すれば☆」

「その答えは　次の局面だぜ☆」

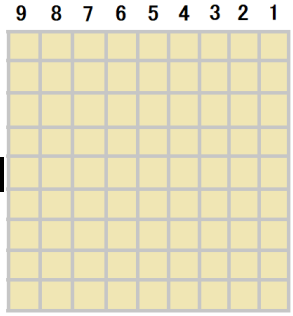


= 9/9/9/9/9/9/2P6/9/9/9 -

「つら……☆」

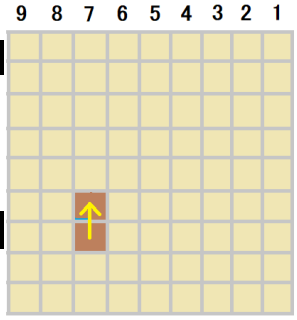
「なんで　掛け算　の記号を使うの？
足し算　の記号では　ダメなの？」

SHOGI SEKI
「 将棋積 だぜ☆
サイン コス
ｓｉｎ も ｃｏｓ も出てこない 二次元の デカルト座標だぜ☆」



REIGEN
「 じゃあ 空っぽ の局面を 零元 としよう☆」

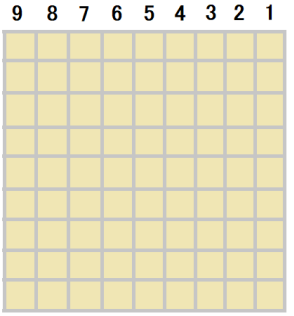
9/9/9/9/9/9/9/9/9 -



9/9/9/9/9/9/9/9/9 - × 7g7f

NANA ROKU MU
「 空っぽ の局面で 7六無 を突くと
何が起こるのか☆？
コンピューター将棋のルールでは 無い駒を突くのは ILLEGAL MOVE 反則手 だが……☆」

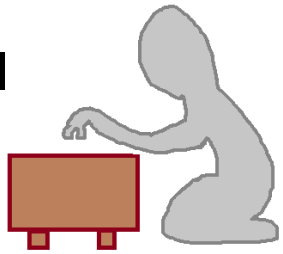
「反則手 なのよーっ！」



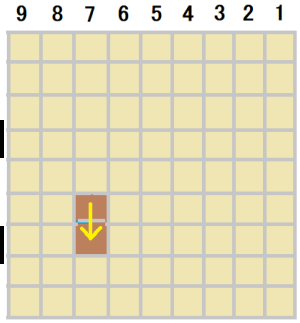
= 9/9/9/9/9/9/9/9/9 -

// Did you brush dust off?

「 そこは 柔軟に
何も変わらなかった、
と 定義しても いいじゃないか☆
位相も変わらない……☆」



WARIZAN
「 １手戻すのは 割り算 としよう☆」



9/9/9/9/9/2P6/9/9/9 - ÷ 7g7f = 9/9/9/9/9/9/2P6/9/9 -

9/9/9/9/9/2P6/9/9/9 - × 7f7g = 9/9/9/9/9/9/2P6/9/9 -

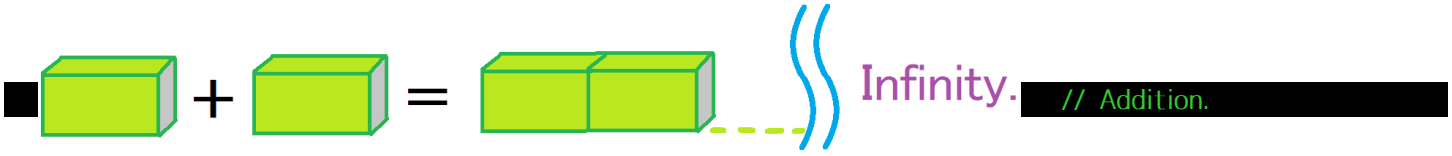
「 先手が ７六歩 を 待った するのは、
後手が相手の駒で ７７歩 するのと 同じだぜ☆
SENGO SHINTAI TENTAISYO NO HOSOKU
先後 進退 点対称の法則☆」

「 お父ん以外に 使う人がいない知識が増えていく☆」

「どこが　KAKEZAN　掛け算　なんだぜ☆？」

「足し算をすると
増えるのは☆」

「一番最初の　足し算　の動きは、永遠に　使い回せて、
この動きを　ヒト科　の動物が見ると　“増える”　という
感情を　もってしまうからなんだが……☆」



「増え続ければ　いつかはあふれる　コップの中の水とは異なり、
この理屈は　インフィニティ　であることを知った☆
こっちは　足し算　の物語のオープニングだぜ☆」

「聞いてないのに……☆」

「それに比べて
掛け算　には　常に　全体　を意識した　比率　のイメージがある☆
インフィニティ　は　足し算の物語　のイメージであって
掛け算の物語 に出てくるインフィニティは それを借りてるだけだぜ☆」

「将棋は　どこに行ったのよ！」

「例えば　数が　４　までしかない宇宙があって、
半分の２　と、　半分以上の３　を掛け算すると
全体が１６としたときの　半分以下６　に減る☆」

「イライラッ」

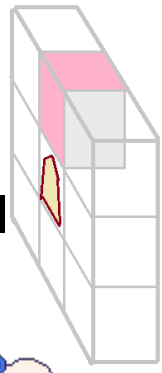
「そして　将棋の初期局面の初手が　３０通り　あることを
思い出せだぜ☆」

FUETE

次元　が　増えて　いる感情を持ってしまうわよ

JIGEN

平均　１３０次元　ぐらいかだぜ☆

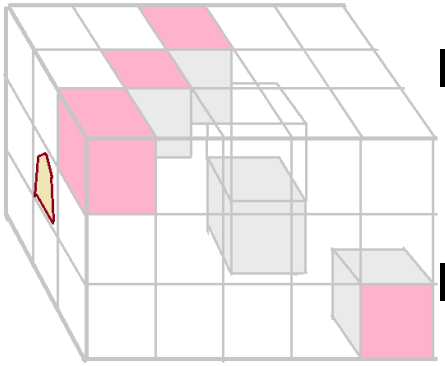


「将棋盤は　すでに　２次元　あるんで、
３次元　しか絵に描けない　わたし　は
あと　１次元　しか余裕がない☆」

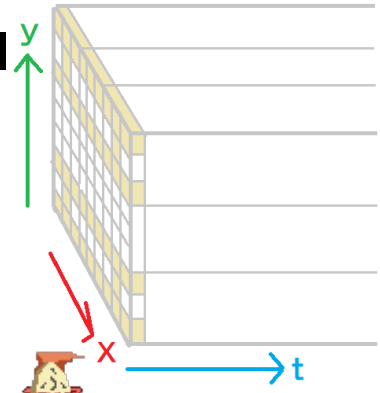
// Pawn moves.

「箱が立っているぜ☆」

// Silver moves.



「駒の動き　１つ１つ　を紙にして
本を作ったと　思ってくれだぜ☆」



// Moves.

「これを　将棋盤　でやって、
横から見た　と思ってくれだぜ☆」

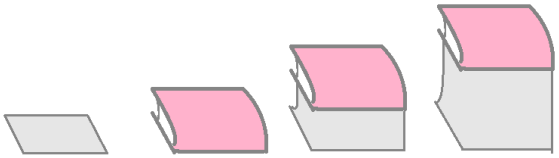
「ひふみんアイのタイムバージョンね」

「この板を全部　掛け算する☆
平均手数が１３０なら、
１３０次元☆」

「初手が　３０　通りとして、
２手目には　９００　通り☆」

「掛けても　掛けても　将棋盤は
ずっと　２次元　だろ☆
本譜　しか見てないからだぜ☆」

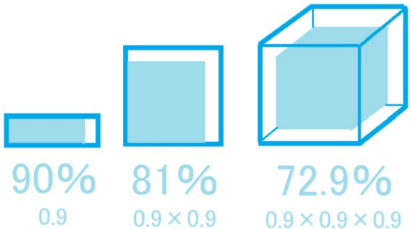
// Some positions.



「変化　を見ることができたなら
底（てい）　が変化するが、　n次元　だぜ☆」

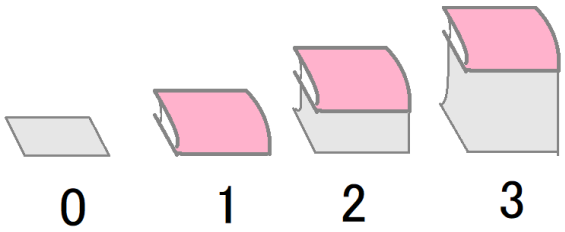
「変化を探索する　あらゆるゲームは
次元が上がるだけで　水が減っていく
JIGEN NO NOROI
次元の呪い　にかかっているのよ」

// The curse of dimensionality.

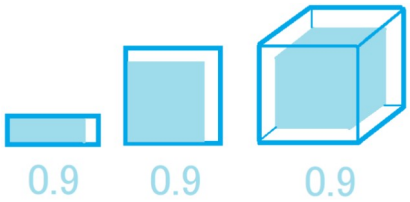


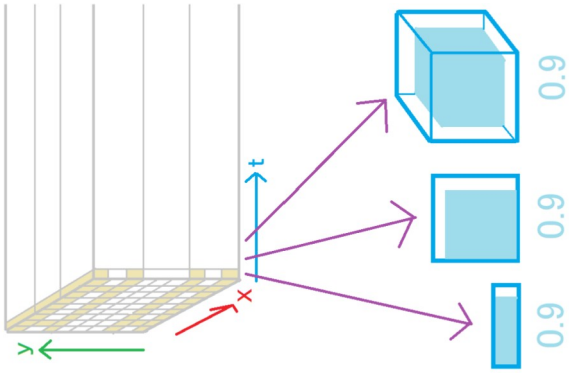
「わたしたちには
ロガリザン　が必要だよな☆」

// Logarithm.



「ロガリザン のような アルジェブラ を
将棋のような ジィオメトリー に
マァッピン できるのかだぜ☆？」





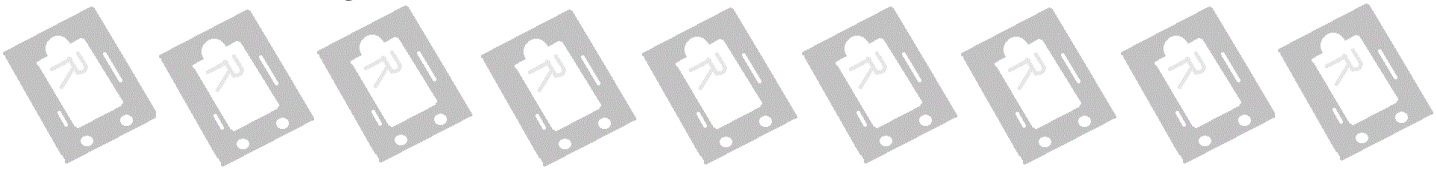
// Mapping.

KAKEZAN　TASHIZAN

「掛け算　を　足し算　に変換する方法を
思いつけば　いける☆」

「それが　ロガリザン　なのに……☆」

「一手　指すごとに
スレッドリッパー　を
３０倍
投入すればいいのよ。
簡単なことよ」



「原子の個数が足りないぜ☆」

TSUKURIDASE

「しかし、それを　創り出せ　ということだぜ☆
将棋の平均合法手数が　８０　として
８０、６４００、５１２０００…　と増える　容積　を
１、２、３…　の　長さ　にしか感じないゲームを☆」

FRAME　MONDAI

「フレーム問題　を解決すれば　いけるわねぇ。
玉を囲うのに　５１２０００手　も読んでいる人は　いませんからね」

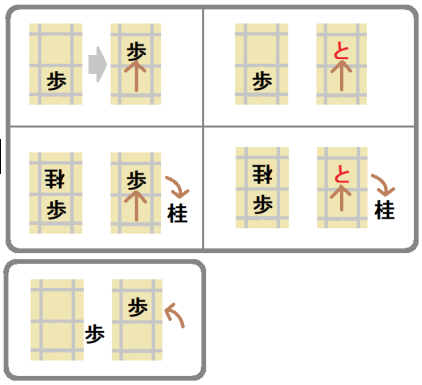
「将棋は　相手より　ちょっとだけ強ければ　圧倒的な差になるのだから、
探索空間の　完全解析　を目指さなくてもいいのよ」

TASHIZAN

駒を繰り出す指の運びは　足し算　だぜ☆

NORANAI

棋譜には　載らない　わねぇ

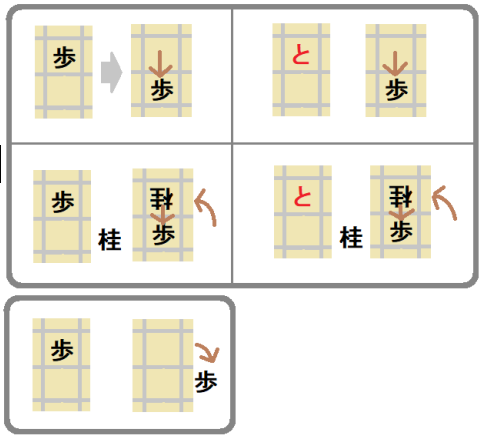


make_move( )

「創造の７番目は
ITTE NO KEISEI
一手の形成☆」

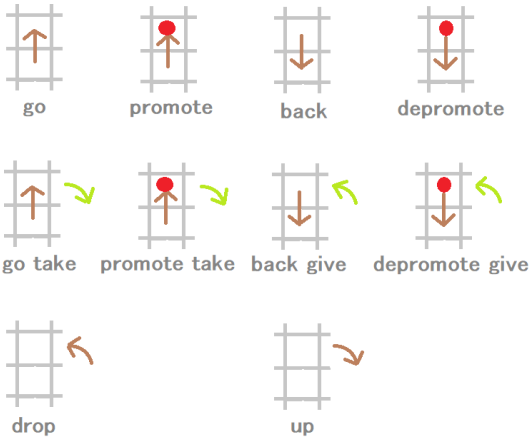
「もう用語にする気もない☆
（＾～＾）」

「駒の動き　じゃないの？」



unmake_move( )

「指したり、戻したり　したときに
人間将棋の　盤上のみなさんが
何をしてるかだぜ☆」



// 10 events.

「網羅すると　紙面が足りないんで、
抜粋すると　１０イベントある☆」

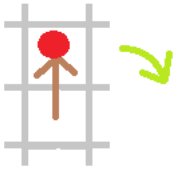
「ｕｐ　わらう☆」

「こうやって見ると、その場で成る　とか
無いな☆

わたしはチェスの キャスリング にびっくりしたが
将棋に無いイベントで見慣れなかったからだな☆」

// No these events.

「頭が　おかしくなる！」

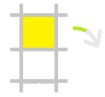


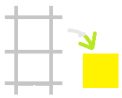
// 8h2b Promoted bishop.

「 ３手目、２二角成　で
足し算　を定義しよう☆」

「 この絵は　２二角成　だったのか……☆」

// This formation has 5 steps.


pickup the opponent
-2bβ


put out the opponent
+(B*)δ


pickup the friend
-8hα

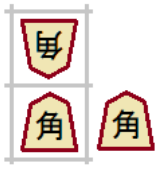

put on the friend
+2bβ


turn over to be promoted
+(+)γ

「 目に見えない変数
アルファ　ベータ　ガンマ　デルタ
α　β　γ　δ　があると思ってくれだぜ☆」

// 1 opponent piece and 2 friend pieces.

「 USIプロトコルと合わせるための調整を入れる☆」



+2bβ +8hα +8hα

-2bβ +(B*)δ +-8hα +2bβ +(+)γ +2bβ +8hα +8hα
= +-8hα +8hα +8hα -2bβ +2bβ +2bβ +(+)γ +(B*)δ
= +8hα +2bβ +(+)γ +(B*)δ
Answer.　8h 2b + (take B*)

8h2b+

「 これで足し算は定義された☆」

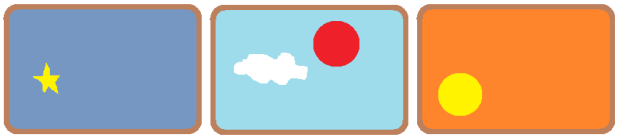
「 実質、何も　足してないですけどね」

SHISOU

足し算で　思想　して、　掛け算　で動くの？

HYOUGEN

表現 は思想の卵だぜ☆



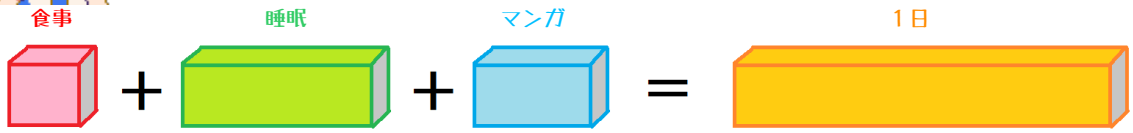
// One day.

「 創造の８番目は
ICHINICHI NO SHISOU
１日の思想　だぜ☆」

「 序・中・終盤を　１日に例えたとか
そんなのか☆」

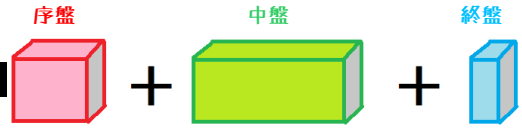
「 相手の思考を読み始めたわね」

「 明日は　今日と違う１日にしたい、
そういう構想を立てて　次の日　朝から　寝ている……☆」



「 ディオファントスの方程式でも　やるかと思ったのに……☆」

「 わたしも　誰も　思ったようには　できていないのに
終わってみれば　似たように　終わったと　心　に映る……☆
KEKKATEKI NI TASHIZAN
結果的に足し算、　足し算　とは　そういうものだぜ☆」



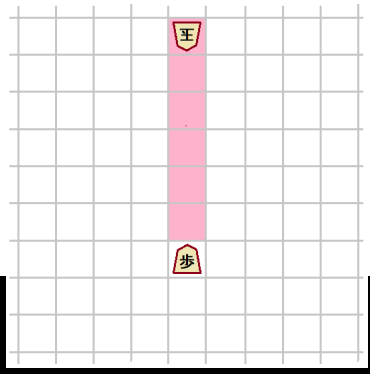
「 勝つときは　棚ぼた　で、
負けるときは　一手ばったり、終盤が無かったりする☆
そんなこと　しようと思ってないのに
そうなるんだぜ☆」

「 じゃあ　アルゴリズムには　棚ぼたで勝つ　を入れておかないとな☆
期待値は　高いぜ☆」

「 ウィンドウの広い結果論よね」

「 数　はあくまで　表現　だぜ☆
SOUWA
しかも　総和　には　次元の呪い　が無い☆
一次多項式の　お手ごろ感　を思想に使うのは　みんなやっている☆」

「 歩　を５回突けば　勝つ☆」



「 良かったな☆」

(5fβ -5gα)+(5eβ -5fα)+(5dβ -5eα)+(5cβ -5dα)
+(5bβ -5cα)+((5aβ +γ -5aβ +k*δ)-5bα)
= (5fβ +5eβ +5dβ +5cβ +5bβ +(5aβ +γ))
-(5gα +5fα +5eα +5dα +5cα +5bα)

「 α、β　はどっちも 連続する主体である歩 なんで 1 でも入れて消そうぜ☆」

= 5a+γ +k*δ -5g

「 ＋は成る、＊は打つ　で数式と被っているが、この１次式を　説明すると、
指の運び的には　５一に何かを置き、
どこかを裏返して　玉が駒台に乗り、５七の何かが消えるな☆」

「 盤面が見えない視聴者の叫びみたいね」

「 USIプロトコル用に調整☆」

= (γ +k*δ -5g) + (5a +5g + 5g)
= γ +k*δ + 5g +5a
Answer. 5g5a+ (take k*)
or 5a5g+ (take k*)

１日目　先手、後手、どちらでもない
２日目　盤
３日目　棋譜
４日目　駒のシープ
５日目　一対多
６日目　符号
７日目　一手の形成
８日目　１日の思想

GENESIS

「 これが　お父んの　ジェネスィス　かだぜ☆」

ジェネレイッ　ムーブ

「 というか　指し手生成　しか並んでないわね……

スァーチ　　エヴァーウエイション

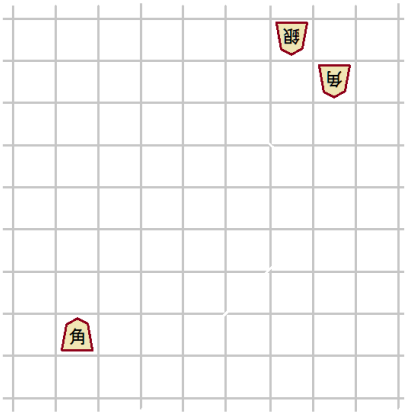
探索部　と　　評価部　　は　どうするのかしら？」

KENSHOU

筋違い角戦法で　検証　してみようぜ☆？

足し算だけで　手筋（TESUJI）　が作れるの？

// Wrong diagonal bishop.



「わたしの棋力は　アマ５級　なんで、
とりあえず　３手でできる
SUJI CHIGAI KAKU
筋違い角　の手筋を
足し算　で表してみようぜ☆？」

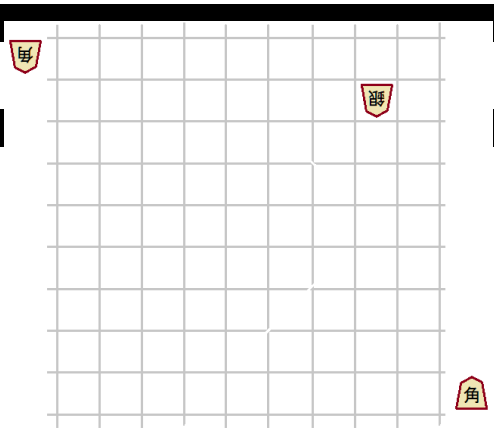
「これで役に立たなかったら、
ＰＲ文書を読んだ時間を返せよ☆」

-2bβ　+(B*)α　-8hα　+2bβ　+(+)β

「藤井猛のような　馬の裏返し方　は
今は　考えないものとする☆」

「うなぎの売っていない
うなぎ屋さんに入った気分ね」

-2bβ　+(B*)β　-3aα　+2bβ



// Turn horse.

「将棋で表現できない　駒の運び方が
３回続けて起こる☆
しかも私の理論より短手数☆」

「全パターン網羅した方が
いいんじゃないか☆？」

-(B*)α + 4eβ

「忘れた頃に　やられたら
反則手かと思って　ビックリしてしまう☆
合理的とは そういうことなのかもしれない☆」

-2bβ +(B*)α -8hα +2bβ +(+)β -2bβ +(B*)β -3aα +2bβ -(B*)α +4eβ
= +(B*)α -8hα -3aα -(B*)α -2bβ +2bβ +(+)β -2bβ +(B*)β +2bβ +4eβ
= -8hα -3aα +(+)β +(B*)β +4eβ

「 これを説明すると、 ８八 と ３一 から何かが消え、
どこかで何かが成っていて、 どこかに先手角が打ち込まれていて、
４五に何かが出てきてるぜ☆」

「 駒を取った時に 相殺して動きが消えてしまうのは 調整 するしかないのかしら。
成り駒を 駒台に置くときの 裏返し直す動き も必要ね」

「 駒が その場にあったことを表す定数が いるのでは☆？」

「 まだ理論の 調整 が要るが、
方針は 足し算☆ 以上の通りだぜ☆」

「 こんなんで 将棋が指せるのか☆？」

「 三駒関係が 局面近似 なら、
足し算 は 駒運びの近似 だぜ☆
それが 役に立つかどうか は知らん☆」

「 なんか名前は無いの？
こまはこびわ
駒運び和 とか」

「 じゃあ それで☆」

「 …… …… 」

FORMULA

感情のフォーミラを創り出せだぜ☆

MAKEZUGIRAI

負けず嫌い　が良いらしいわねぇ

// Purchase a fan at a shop.

「部屋を掃除していたら
糸谷元竜王の

SOUZOU
創造

が出てきたので
将棋の数式を
創造することにした☆」

「場当たり的に生きてるなぁ……☆」

KOMA HAKOBI WA
「駒運び和　は、
盤上の　ほこり　を取るぐらいにしか
使えなくない？」

「何も考えていない☆
まだ無い何かを　創造　するだけだぜ☆
創造　に　それ以外のことは　不要☆」

「スレッドリッパーに　へんちくりん　な足し算を　やらせる気か……☆
やはり　ただの箱　なのでは……☆？」

「自分の持ち駒を　相手の駒台に載せて
反則負け　することを考えた仕様なの？」

「残念ながら　ＵＳＩプロトコル　では
相手の駒台を　指定できない☆
しかし　駒運び和　の思考の中では　やるはず　だぜ☆」

「指し手生成部の開発者にしか通じない　お父んの言う　一種のダジャレ　だろ☆
指し手ジェネレーション～☆」

>>> Does kifuwarabe's Genesis overcome the curse of dimensionality?!

第29回
世界コンピュータ将棋選手権
Claire アピール文書

2019年3月31日

開発者　上原 大輔

自己紹介

- ✓ プログラミング歴は5年
- ✓ 2015年10月頃から開発開始
- ✓ 職業はプロデューサー（アイドルマスター）

過去の戦績

- ✓ 第26回世界コンピュータ将棋選手権
  やねうら王ライブラリを使用して出場
  1次予選27位（2勝4敗1分）
- ✓ 第27回世界コンピュータ将棋選手権
  オリジナル勢に転向して出場
  1次予選16位（4勝3敗）
- ✓ 第28回世界コンピュータ将棋選手権
  Rust言語で書き直したが完成せず
  0次予選敗退

今年も0次予選敗退の予感……

特徴(1) 評価関数

- ✓ 駒割、3駒関係、ディープラーニング評価関数を使わない。
- ✓ 評価関数は駒の利きのみを使用。

特徴(2) 探索・定跡

- ✓ 定跡はfloodgateの棋譜約8万局より抽出。
  局面の出現数・勝率を考慮し指し手を選択します。
- ✓ 飛・角・歩の不成も探索します。

現在の状況

- ✓ 前回大会キャンセル後、開発をサボっていたのに
  ゼロから作り直した結果、開発が間に合っていない
  （1年ぶり3回目）
- ✓ Rust完全に理解した（してない）
- ✓ 百合子と灯織のプロデュースが忙しい

本大会での目標

✓ 完成させる

最後に

- ✓ 最後まで読んで頂き、ありがとうございました。
- ✓ （完成したとしても）一昨年よりも弱いかもしれませんが、お手柔らかにお願いします。

Scherzo のアピール文書

2019年

Scherzo

Scherzoは、通常の全幅探索を行っています。

ルート近くの評価には neural network を用いています。

少し前まではNNを使っているのがプログラムの売りだったのですが、

AlphaZero 登場以降、ごく普通の技術となってしまったのが寂しいところです。

他に新機軸がないか考えた末、チェスでいうところの retrograde が将棋でもできないか検討しておりますが、

大会当日までに間に合うかどうかわかりません…。

第29回世界コンピュータ将棋選手権
にこあ将棋　アピール文書

- 名称「にこあ将棋」
  - 最初は Core○○の名称で他のソフト等も開発しているので、それらと合わせて、「こあ将棋」としました。
    その後、電王 T 参加時に、開発のきっかけを与えてくれたニコニコ動画をリスペクトの意もかねて「にこあ将棋」で参戦。その後「にこあ将棋」で通すことにしました。

- 開発経緯
  - 第2回将棋電王戦より興味を持ったものの、開発する余裕がなく保留にしていたのを、将棋電王戦 FINAL 第4局終了後より、無理やり理由をつけて開発を開始しました。

- 特徴
  - 他のソフト・ライブラリをまったくと言っていいほど参考にしていません。
    探索、評価、その他もろもろ全て自前で実装しています。
  - パラメータ調整 GUI を実装。
    自動学習と手動調整でユーザー独自の思考エンジンが可能です。
  - αβ 探索とモンテカルロ探索を併用（たぶんうまくいっていない）
  - 複数の評価関数をミックス、状況に応じて重みを変化でき…たらいいなぁ。

- その他実装したいこと
  - ユーザーの調整した AI で対戦できるようにし、ランキング1位の AI で参戦。
  - にこあ将棋ユーザーのマシン全体でグリッド化。
  - ディープラーニングによる評価関数もいつの日か…

- 使用言語
  - C++

- 開発者 HP

  AnotherProject

- 開発者 Twitter

  https://twitter.com/dr_neeton

第 29 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（よみ）じゅうろくしきいろはかい

【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。
改の部分は、改装……開発言語を変更して新たに一から作り直したので。
そして「・」を抜きました。まぁ、由来はやっぱり 2016 年のときとほとんど変わらずです。

【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。最近指してないんでもっと弱くなっている気が……
（「どうぶつしょうぎウォーズ」では4級です……ずっと同じ……あれ？？？ Σ(ﾟдﾟlll)w）

【ソフトについての大きな特徴】
「いろはは待たせませんっ！ Σ(ノ∀`*)ﾍﾟﾁｯw」

【ソフトについて】
・ライブラリ不使用です。
・一昨年出場したときから変わらず Lua です。（ホントは再度作り直そうと思ったりしたのですが）
・ Lua の特長は、軽量なスクリプト言語。さらに luvi（発音は、らび？）を利用して、
スクリプト言語の弱点である実行速度の遅さを、ある程度克服しています。
（参考） GitHub - luvit/luvi: https://github.com/luvit/luvi
・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**
・**以前よりもまして弱くなったりした**ことがあるといういう不思議さん。
（でも、今回はちゃんと強くなっている！……はずw）

【ツイッターアカウントやブログ】
https://twitter.com/16shiki168
http://16siki168.seesaa.net/

【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
~~序盤は居飛車党で「箱入り娘」を一心不乱に目指し、その後は~~合法手をすべて探し、後述の評価方法により、指す手を選びます。~~だから、十六式いろは改は「娘（女の子）」だそうです。~~
~~あと、王手されるとそれを逃れる手を選ぼうとするはずです…… ※というはずだったんですが、~~
~~なんか箱入り娘を目指さなくてもいいかなという思いがありますが~~
娘なのは間違いないはずです、**天然な方向で**w

・評価方法や探索方法
駒の損得、~~駒の効率、玉の硬さ、利き、ひも付き、手の強さ、厚み・位取り~~を評価します。
（何かひとつくらいは評価を追加したいナぁ）
それぞれの評価値は、手作業で入力し調整します。機械学習はしません、というか今回は~~も~~
そこまで手が回りません。
※そのときの**最低限の駒の損得は判断できるようになりました**（**評価値を出せるようになっています！**）が、次の手の探索にまったく活かしません……
駒を得しているなぁとか、損しているなぁとか思うだけです……えっ！？[2019/3 現在]
（選手権当日までには、活かせるようになっているはず！？）

探索は、基本的にランダムです。時の運です。~~反復深化法とミニマックス(アルファベータ)法ですが、基本的に「せまい」範囲を探索します……深くいけばいいのですが、たぶん浅い(3手先とか)です。……つまり、せまく浅く(！？)探すので、静的評価の精度をよくすることを目指しています。~~(※と思っていたときもありました)
以下のモードを搭載しています。もっと機能を載せたかったのですがたぶん、搭載する時間がなく……[2019/3 現在]

・搭載モード
~~局地深読みモード:相手の打った位置を中心に9～16マスのみを深く探索し、できるだけ王手する。~~
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。
(↑できるようになりました！(*ﾟ▽ﾟ)ﾉ)
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！？(昨年からの**安定の平常運転**です(*￣m￣)ﾌﾟｯw)

【独自性・希少性】
開発言語は「Lua」です。変えようと思っていましたが、時間がないと思うのでそのままです。その Lua 特長は以下の通り。
「Lua は、C 言語のホストプログラムに組み込まれることを目的に設計されており、高速な動作と、高い移植性、組み込みの容易さが特徴である。」(Wikipedia より https://ja.wikipedia.org/wiki/Lua)
でも C 言語は使いません。さらに LuaJIT を含む luvi(発音は、らび？)を利用して、スクリプト言語の弱点である実行速度の遅さを、ある程度克服しています。
(参考) GitHub - luvit/luvi: https://github.com/luvit/luvi Lua(luvi)を採用した最大の理由は、簡単に実行形式のファイルを出力でき、ソースの量も C 言語よりも 20～30%くらい減りました。
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)
1 勝を実力でもぎ取るくらいの強さはあるかもしれませんw

私の知る限りでは「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】
昨年はちょっとわけがあったんで傘下ができませんでしたが、今年もやっぱりずっと
完全にエンジョイ勢です、すみません Σ(ﾉ∀`*)ﾍﾟﾁｯw
(今までエンジョイできたので満足中です)

……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

とりあえずいつもどおり**予選当日の朝のギリギリまで**ガンバる予定です。
今年のゴールデンウィークは長いのでやりたいことを少しは実装したいですネ。

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。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！どれもいつものことです Σ(ﾉ∀`*)ﾍﾟﾁｯw
みなさまの応援うえるかむです。
それに応えられるようガンバります！想像もつかない方向へ……えっ！？
(ヤバい、変な方向にハードル上げすぎた。普通な指し方になったらどうしよう(;´Д`))

あー……前の PR 文章とほとんど変わってなくてごめんなさい。

2019.3.31

st34 アピール文書

田中　智

2019 年 3 月 27 日

1　出場大会

第 29 回世界コンピュータ将棋選手権：初参加

2　プログラム情報

2.1　プログラム名

st34

2.2　開発言語

python

2.3　アピール・工夫点

- ライブラリ不使用
- ニューラルネットワークを用いた評価関数を使用
- 局面の評価と詰みの評価で，異なる評価関数を使用予定
- 読みが深くなると，読みの幅を小さくする

第２９回コンピュータ将棋選手権
FTS3　PR 文

・作成動機
　以前より将棋に興味がありました。将棋を最初に始めた７，８年まえ頃では
トッププロの棋士が負けるわけはないと言われていたのに数年前に名人が破れ、
到底、人間が勝てるレベルではなくなってしまいました。
　そういう過程を見ている中で、自分も AI ソフトを一から作成をして、
プロのレベルを超えて、かつ、強いソフトと渡り合えるようなものを作成してみたい
と思ったことが作成動機です。

・FTS 3 の特徴
　ディープラーニングを用いた評価関数によって思考するソフトです。Alpha Go Zero を
参考にしています。評価関数には TensorFlow を用いて教師あり学習を行っています。
学習にはプロ棋士の棋譜を用いています。また、高 PC スペックでなくても動作することを
前提とした構成になっています。

・大会に向けての抱負
　初参加で一から作成をしたため、至らない点も多いと思いますが、
最低限、バグや反則負けは起こらないように作成をすることを心掛けました。
皆さんの胸を借りるつもりで少しでも上を目指して頑張ります。

WCSC29 アピール文章

ソフト名 Daigorilla　作者　田中大吾

名前の由来：　自分の名前を兼ねて面白いソフト名にした

Daigorilla の特徴：　新米長玉をモチーフに作者の指して・研究手順を学習させた自分の分身のようなソフト。
また新米長玉をモチーフとすることで受けのパラメータに優れることも判明。

工夫したおおまかな開発手順：より特徴的なソフトに仕上げるために教師作成時に書き出す手数を調整し、序盤専用と終盤専用にわけてそれらを学習させた。教師は Depth6 から Depth8 程度。量は約 300 億を追加学習に使用。

ソフトの型：　やね先生が配布した NNUE-K-P-256-32-32 を追加学習したもの。

使用した（教師作成も含む）ライブラリと選定理由

・Qhapaq ・・・ライブラリの中で最も強力な関数。特に中盤以降の強化に利用。

・elmo ・・・elmo 特有のエルモ絞りにより序盤の正確性がうかがえる。序盤のパターン付けの強化に利用。

・Apery・・・詰み探索に優れているほか終盤の正確性がうかがえる。終盤の確実性を強化するのに利用。

・tanuki・・・ やね先生が配布した魚沼産やねうらおうを使用するために申請が必須。

・やねうら王・・・ソースコードが読みやすく且つ余分な部分を省いて NNUE-K-P-256-32-32 用に高速化をかけてビルドした。

今回ライブラリを使わせてもらうことに開発者のみなさまに感謝します。

※定跡などは空の SBK ファイルから手動で入力

だるま将棋　アピール文

初参加なので反則をせずにきちんと投了することを目標にしています。

このプログラムはディープラーニングを使って学習（予定）させ、モンテカルロ木探索を使って探索させます（予定）。

色々な棋譜を学習させ自由な棋風にしたいと思っています。

ライブラリも使わせて頂くつもりですが、全てが予定なので使わない（使えない）かもしれません。

うさぴょん外伝アピール文書

2019/04/16

うさぴょん外伝開発チーム

1. 全体像

今回は、評価関数部を塩崎が、探索部を池が担当しています。

評価関数については、いわゆるDeepLearningにてGPUで処理を行い、探索部はCPUが処理を行う形で、DL勢としては、いたって普通の構成だと思います。
「うさぴょん」シリーズとしては、今までの資産をかなりの部分捨てた感じとなっています。（合法手生成や詰将棋位は「うさぴょん」から流用しますが…。なお、「うさぴょん2」から流用する部分はありません。）

2. 評価関数について

基本設計はmEsshiahのSDT5版と同じくCNNモデルをDQNで強化学習するという形です。
AlphaZero等とは違い、PolicyNetworkに該当するものはなく、ValueNetworkに該当するものだけとなります。
今回重点的に行ったのは以下の2点です。

・入力データの精査
・最適なネットワークサイズ、形の探求

2－1. 入力データの精査

入力データのうち、本当に必要なものはなんなのか？というのを以下の方法を用いて精査しました。

基本はやねうら王教師局面を用いての教師あり学習で実験を行いました。
学習方法はelmo式ではなく、単純な2乗誤差で行いました。

実験を重ねていくうちに、学習Iteration1万程度のときの一致率と一致率が伸びなくなるときの一致率とに相関関係があることがわかりました。
そこで、一致率が伸びなくなるまで学習させて比較するのではなく、学習Iteration1万程度の一致率をどれだけあげられるか？というコンセプトで特徴入力の精査を行いました。

まず、基本の駒位置と持ち駒の数。
普通に9*9*28+36のものをベースとし、以下のものと比較しました。

9*9の盤面に追加して各手番2*9の駒台要素の追加
マイナスの入力を使っての先手駒と後手駒のチャンネルの統合
駒が成り駒であるかどうかだけを入力するチャンネルの追加

これを行うことにより、従来の形の次元数の半分まで次元数を削減することができ、
また、先手の駒と後手の駒、成ってない駒、成り駒、持ち駒はすべて同じ種類の駒である、という情報を入力することを目指しました。

これをベースと比較した場合、2%程度の一致率の向上が認められました。
さらに以下のような特徴を追加しました。

単純効き（そのマスに駒の効きがあるかどうか）の追加　向上幅2%程度
方向別効きの追加　向上幅2%程度
ひもと当たり（盤上の駒に効きがついてるもの）の追加　向上幅1%程度

上記特徴で 1 万 Iteration（バッチサイズ 1024）で一致率 23%程度、最終一致率 38%程度となりました。

２－２．ネットワークサイズおよび形の精査

上記、入力データの精査では、教師あり学習での実験は 6 層 32 フィルターのベースネットワークで行いました。
これを以下の条件に変更して実験してみました。

12 層 32 フィルター
12 層 256 フィルター
24 層 256 フィルター
48 層 128 フィルター（GPU メモリの都合で 256 フィルターにはできなかった)

上記は 1 万 Iteration 時の一致率に 1%程度の向上はみられるものの、最終的な一致率は変わらなかった。

3 層 32 フィルター

上記は 1 万 Iteration 時、最終的な一致率とも 2～4%程度の低下が認められた。

1 層 32 フィルター

上記は 1 万 Iteration 時、最終的な一致率とも 8%～10%程度の低下が認められた。

FullConnect3 層 1024

上記は 1 万 Iteration 時に著しい低下が認められ、最終的な一致率は 10%程度の低下が認められた。

上記結果より、

現在の学習方法（教師あり学習）では一致率が 38%程度で打ち止め？
ネットワークを大きくしてもさほど精度は変わらなかった。

２－３．そして新しいネットワークへ…

根本的な改良が必要であると判断したためいろいろ試行錯誤した結果、1回の forword で全合法手の評価値を出すネットワークを学習できる方法を発見した。
その結果、従来のものより合法手の数分高速となり、より大きなネットワークを使うことが可能となった。
ネットワークサイズに関しては 3/30 時点で 13 層 256 フィルターのものを学習中である。

＊2019/04/16 追記　ここから：

２－４．全合法手の評価値を 1 回の forword で出力する為の新手法

今回の mEssiah 評価関数は新開発の方式を採用している。
具体的には入力、ネットワーク構成は従来のものを使用しているが、出力が大きく異なっている。
出力に One Hot Vector（後述）から生成した分散ベクトルを使用することにより、1 回の forword でその局面全ての合法手の評価値を出すことが可能となっている。

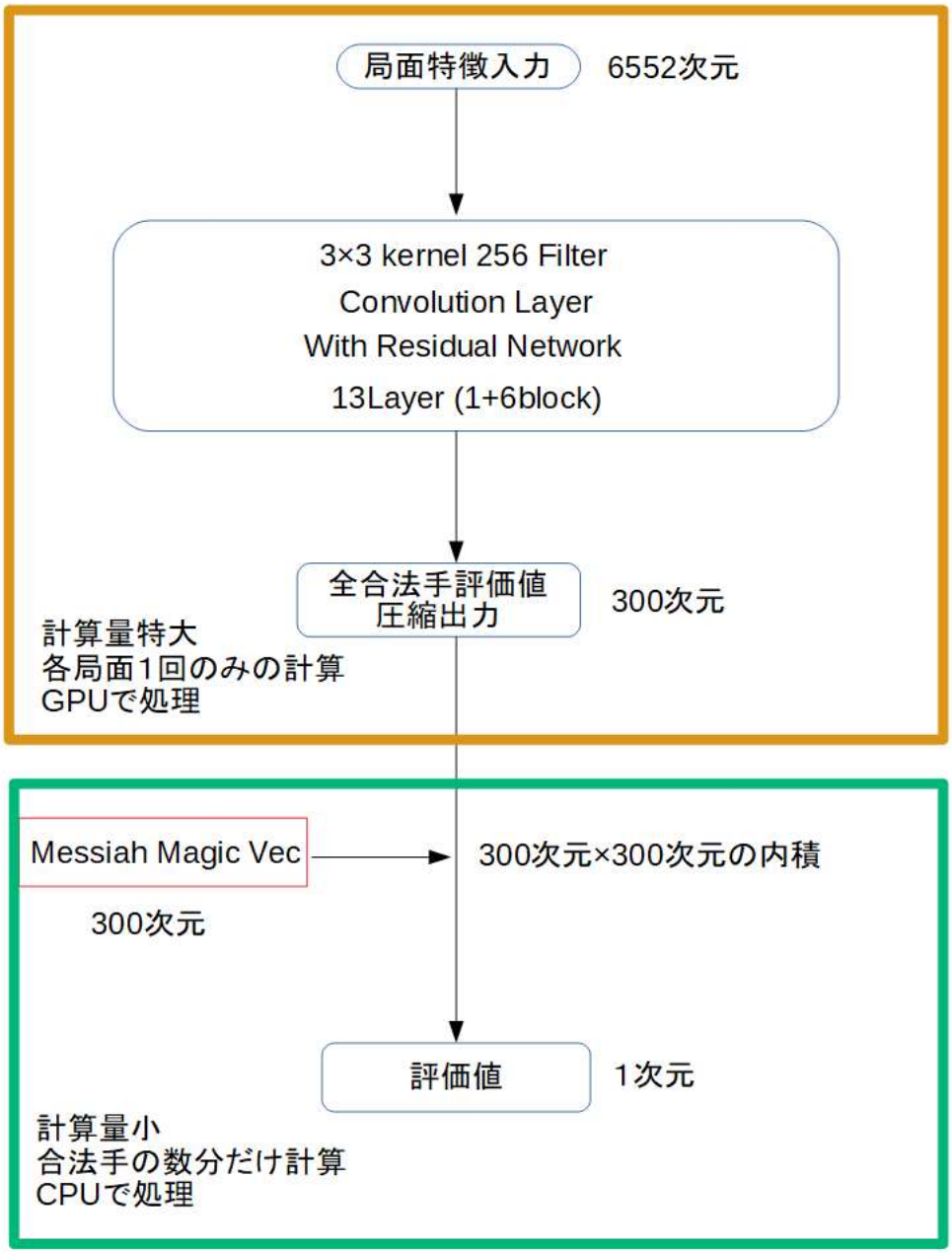


図1　実行時のネットワークと計算量

従来のネットワークでは出力は 1 次元で評価値そのものを出力していたが、今回のネットワークは合法手全ての評価値を圧縮した分散ベクトルとして出力する。
評価値として使用する時には後述する「学習」で作成した、合法手 One Hot Vector を分散ベクトル化したもの（Messhia Magic Vec）との内積をとることにより、その合法手の評価値とする。
こうすることにより、ネットワークの入力から出力までは一度しか計算しないが、その局面の合法手がたとえ 600 種類あろうとも、全ての合法手の評価値を圧縮した分散ベクトルが出力される。圧縮された評価値を解凍する計算量は従来のネットワーク（1 次元の評価値を出力するもの）を合法手分計算するよりもはるかに小さい。
よって、局面によっては従来の 100 倍速以上の評価速度となる。
また、合法手がどんなに多くとも圧縮評価値を出力するまでの時間は一定であり、圧縮された評価値の解凍は十分に速いことから、どんな局面でもほぼ一定時間で全ての合法手の評価値を出力することができる。

学習方法

合法手の One Hot Vector 化は以下のように行った。

移動元　81 マス + 駒台駒種ごとの 7 = 88
移動先　81 マス
この移動で駒が成ったかどうか 1

これを1次元配列化し[88*81*成り+88*移動後マス+移動元マス]を要素番号とする One Hot Vector とする。

学習時にはネットワークは図 2 のようになる。

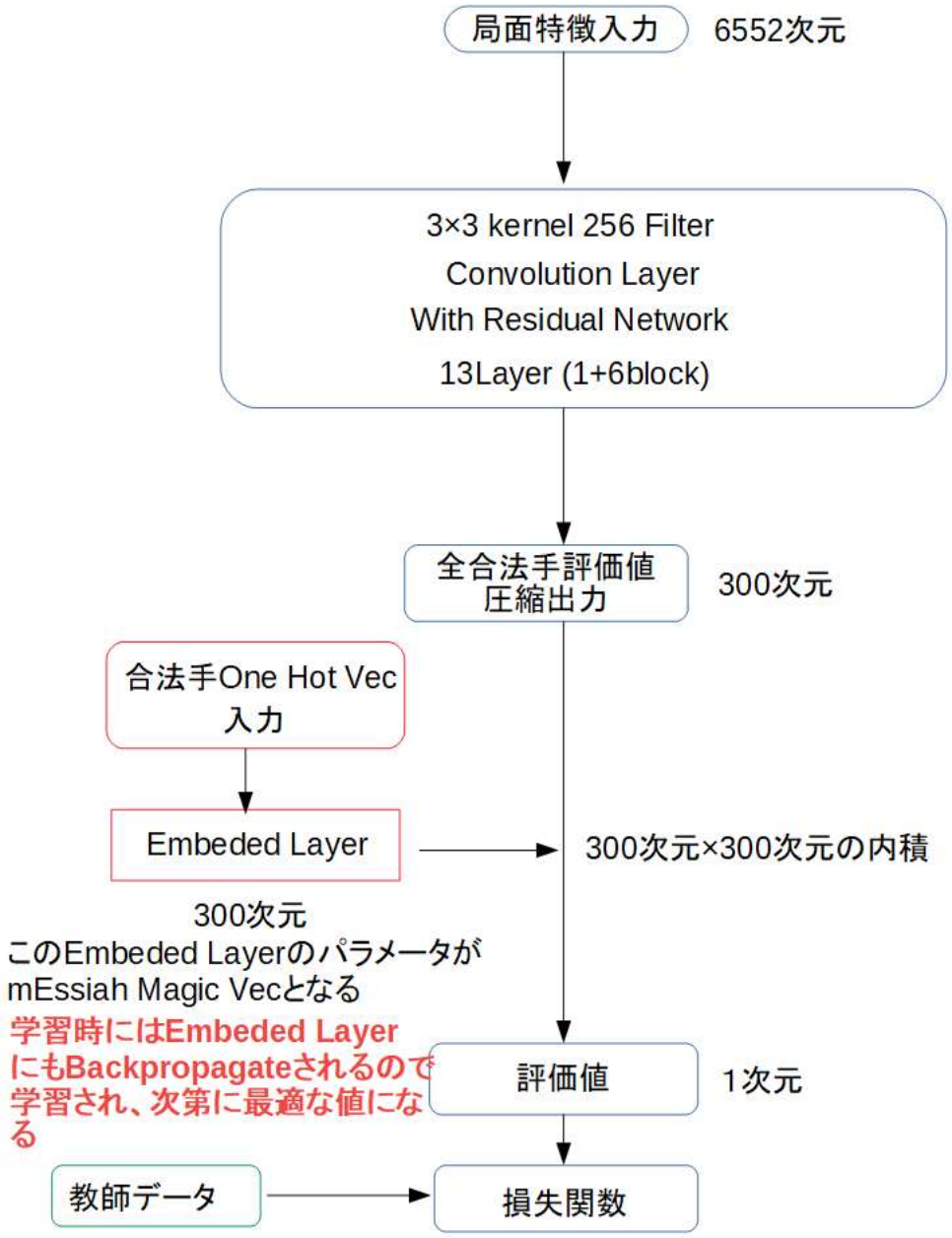


図 2　学習時のネットワーク構成と損失関数

図 2 における教師データは、既存データによる教師あり学習でも、強化学習での Q 値でも同じように使用ができる。学習完了後は Embeded Layer の Parameter を別途書き出し、実行時に使用できるようにする。
Embeded Layer の Parameter(mEssiah Magic Vec と命名)は(88*81*2)*300 次元のベクトルとなっており、使用時には図 1 で示した通り、該当合法手の分の 300 次元を取り出して使用する。

学習アルゴリズム

実際の学習ではまず Pre Training としてやねうら王教師局面での教師あり学習を行った。その後、強化学習に mEssiah DQN Stage3（DQN に Q(λ)の適用を行った独自のもの）を用いた。

本当は最初から強化学習のみでやりたかったのだが、Magic Vec が初期値に近い状態だと学習中の自己対局の 9 割前後が手数 MAX の引き分けとなってしまう。一方、強化学習の際には、報酬として勝敗の情報が欲しい。
そのため、軽い Pre Training を行って Magic Vec に少し値を付けた状態から DQN で強化学習を行った。

＊2019/04/16　追記、ここまで。

WCSC29(第29回 世界コンピュータ将棋選手権)ではアピール文書を3月末までにいったん提出しないといけないらしいので、ざっと書いて提出しておきました。全文掲載しておきます。

『やねうら王 with お多福ラボ 2019』 アピール文書
http://yaneuraou.yaneu.com/2019/03/20/%E3%80%8E%E3%82%84%E3%81%AD%E3%81%86%E3%82%89%E7%8E%8B-with-%E3%81%8A%E5%A4%9A%E7%A6%8F%E3%83%A9%E3%83%9C-2019%E3%80%8F-%E3%82%A2%E3%83%94%E3%83%BC%E3%83%AB%E6%96%87%E6%9B%B8/

■ 教師局面の生成効率の改善

『やねうら王 with お多福ラボ 2019』(以下、「やねうら王」と記す)では、教師局面の生成部に工夫を施した。

やねうら王では、自己対局を行い、それを教師として機械学習により学習させる。このとき学習に用いるのは、WCSC27でelmoが採用したelmo式である。

※ 『elmoがもたらしたオーパーツについて』:
http://yaneuraou.yaneu.com/2017/05/23/elmo%E3%81%8C%E3%82%82%E3%81%9F%E3%82%89%E3%81%97%E3%81%9F%E3%82%AA%E3%83%BC%E3%83%91%E3%83%BC%E3%83%84%E3%81%AB%E3%81%A4%E3%81%84%E3%81%A6/

elmo式には勝敗項と勝率項がある。勝敗項は、その局面から同じぐらいの棋力のプレイヤーが対局を引き継いだ時にどちらが勝つかという情報であり、勝率項は、その局面で深い探索を行ったときの評価値から計算される期待勝率である。

いま、質の良い教師をなるべく小さな計算コストで生成したいわけであるが、質の良い教師というのは、この勝敗項と勝率項のどちらの精度も上げなければならない。そこでそれぞれ個別に考察を行った。

■ 質の良い勝敗項について

勝敗項は、「その局面から同じぐらいの棋力のプレイヤーが対局を引き継いだ時にどちらが勝つかという情報」だと上で書いた。本当は、どちらが勝つかは100%先手勝ち/100%後手勝ちというような確定的な情報ではなく、「(同じぐらいの棋力のプレイヤー同士だと)先手側が8割ぐらい勝つ」のように確率的な情報であるが、その部分を掘り下げても話がややこしくなるだけなのでここでは細かい議論はしない。

ともかく、この「同じぐらいの棋力のプレイヤー」というところが問題である。例えば先手だけ神様レベルのプレイヤーで後手は並程度の棋力のプレイヤーであれば、先手ばかりが勝ってしまうことになる。これではelmo式の勝敗項はノイズにしかならない。だから「同じぐらいの棋力」であることは必要不可欠な条件である。

そして、このときの棋力は高ければ高いほうがより正確な局面の情報を反映すると考えられる。このため、深い探索深さ(high depth)での教師局面の生成を必要とする。(ここで言う「深い」とはdepth 6〜14ぐらいのことを指す。これは実際の対局時よりはかなり低い数値が、大量の教師局面を生成する都合、仕方がない。以下では「低ノード」「低いdepth」などと言う言葉が出てくるが、実際の対局時よりは相対的に低いという意味であり、これらはすべて教師生成時の探索のことを指している。)

探索深さが深いと指数関数的に探索時間が増加するので、high depthでの教師生成にはすこぶる計算資源を使う。やねうら王では、まず、これを少ない計算資源で抑えることができるかという工夫をした。

そのために、まず教師を生成するときの対局シミュレーションでの思考深さにおける勝率が上がるように探索パラメーターのチューニングを行った。簡単に言ってしまえば、短い時間で最強となるようにチューニングを施した。

1) 短い時間で最強
2) 少ない探索ノード数で最強
3) 低いdepth固定で最強

1)と2)はほとんど同じ意味だが、3)だけは大きく意味が異なる。3)でチューニングしてはならない。なぜなら、探索の時の枝刈りを甘くすれば、強くはなるが、思考時間は相対的に増えるからである。そのようなチューニングをしても何ら意味がない。よって、1)か2)の方法でチューニングすべきである。

やねうら王では、短い時間で最強の状態にチューニングすることにより、質の良い勝敗項(勝敗情報)を得ることに注力した。

同じ探索部で比較すると、0.1秒でその棋力を比較したときに探索パラメーターのチューニングだけで+R160程度の棋力上昇を確認した。その他、低depth(depth8や10)に向け、各種枝刈りの適用/非適用を調整することにより、+R100程度の棋力上昇を確認した。合計で+R260である。

つまり、従来と同じ計算資源で教師局面の生成時に+R260強いプレイヤーでの対局シミュレーションが行えるようになった。これは計算資源を2倍使うのと同等以上の効果があると考えられる。

■ 質の良い勝率項について

勝率項は、深い探索の評価値から計算されるものである。しかし、評価値が“ぶれる”と学習しにくい。ここで言う、ぶれるとは、同一局面、もしくは類似局面なのに評価値に差がありすぎることを言う。

例えば、近年、探索部は枝刈りをどんどん激しく調整する傾向にある。やねうら王の場合、2年前の探索部と1年前の探索部を比較するとdepth 8で同士で比較したときに平均思考時間が6,7割程度少ない。つまり、枝刈りがそれだけ激しくなってきているということである。このように枝刈りを激しくして、見込みのなさそうな指し手を早い段階で切り捨て、そして深くまで探索して“お宝”(評価値の高い局面)を発見し、現局面からそこに到達する指し手を選んだほうが、強くなる傾向がある。

確かに強くするという観点からはそうするのが正しいのであろうが、同じ局面を探索した場合であっても、たまたま“お宝”を見つけられた時は比較的高くなり、“お宝”を見つけられなかった時は比較的低くなってしまう傾向がある。このことは、同じ局面を持ってきて、探索パラメーターを少しだけ変更したときの探索の評価値の分散が、昔の探索部より大きくなっているということから示すことができる。

このような評価値のぶれは、教師として好ましい性質ではない。

そこで、評価値を平滑化するためにメディアンフィルタのようなものを適用する(前後の5手の評価値の平均をその局面の評価値とする)ことも考えられるが、平滑化したいのは、本来は対局シミュレーションのその一局に対してではなく、同一局面、もしくは類似局面の評価値に対してである。

一つの解決策として、枝刈りを甘くする方法が考えられる。つまり、そのdepth圏内の局面に存在する“お宝”を確実に見つけるのである。こうすることにより、評価値のぶれは多少抑えることができる。ただし、枝刈りを甘くすると、棋力自体は弱くなるのが普通であるから、教師データの勝敗項の精度が下がる。

また別の解決策として、評価値がぶれにくい評価関数を用いるというのがある。きちんとしたデータを取ったわけではないが、例えば、NNUE型の評価関数は、非線形な評価関数であるためか、(体感的には)評価値の分散が大きいように思う。(これも示せるようなデータは取っていない。) この意味で、KPPT型の評価関数から教師を生成するのはアリだと思う。やねうら王では、やねうら王2018(2018年にマイナビから発売した『将棋神やねうら王』に収録している、やねうら王2018の評価関数)から教師を作成する予定である。

■ 質の良い勝敗項・質の良い勝率項

勝敗項の精度を上げるには短い時間で最強にする必要があった。これは枝刈りを激しくすることを意味しているのだろうか？

『将棋神やねうら王』の開発の時の実験において、1スレッド3000ノードと2スレッド1500ノードとを対局させると後者のほうが勝率が高かった。普通、並列探索にすると探索効率が下がるので弱くなるはずなのだが、やねうら王の場合、低ノード(低ノード数に固定しての対局)ではそうではなく、枝刈りが相対的に甘くなるので強くなる傾向があるようである。やねうら王にとって、低ノードでは枝刈りが激しすぎる意味があるようだ。

そこで短い時間で最強に調整すれば自動的に枝刈りは甘くなり、評価値のぶれがなくなると思われるであろうが、実はそうでもない。

以下に一つの実験結果を示す。

これは、futility pruningという枝刈り(初期のBonanzaにも採用されている)のパラメーターをどのように調整すれば勝率が上がるかということを別のソフトと対局させてgrid searchしたものである。なお、わずかなRの差まで検出したいため、0.1秒で30万対局以上させている。

PARAM_FUTILITY_MARGIN_ALPHA1: // 元の値は172
  162 : 35451 - 391 - 14546(70.91% R154.75)
  164 : 35584 - 417 - 14807(70.62% R152.32)
  166 : 34531 - 445 - 14599(70.28% R149.55)
  168 : 34481 - 414 - 14749(70.04% R147.53)
  170 : 34955 - 392 - 15052(69.9% R146.37)
  172 : 34926 - 404 - 14913(70.08% R147.83)
  174 : 34363 - 387 - 14958(69.67% R144.49)
PARAM_FUTILITY_MARGIN_ALPHA2: // 元の値は50
  48 : 34508 - 398 - 14876(69.88% R146.17)
  50 : 34622 - 436 - 14851(69.98% R147.04)
  52 : 34331 - 400 - 14833(69.83% R145.78)
  54 : 35110 - 370 - 14924(70.17% R148.62)
  56 : 34920 - 401 - 14728(70.34% R149.97)
  58 : 35114 - 429 - 14682(70.52% R151.48)
  60 : 35686 - 416 - 14730(70.78% R153.72)

また、やねうら王のfutility marginは以下の計算式になっている。(C++で書かれた実際のコード)

```
// depth(残り探索深さ)に応じたfutility margin。
Value futility_margin(Depth d, bool improving) {
	return Value((PARAM_FUTILITY_MARGIN_ALPHA1 - PARAM_FUTILITY_MARGIN_ALPHA2 *
improving) * d / ONE_PLY);
}
```

futility pruningは以下のように、その局面の評価値(eval)が、beta値 + marginを超えていれば、枝刈りするというものなので、このfutility marginの値が小さければ小さいほど適用されやすくなる。

```
if (!PvNode
	&&  depth < PARAM_FUTILITY_RETURN_DEPTH/*7*/ * ONE_PLY
	&&  eval - futility_margin(depth, improving) >= beta
	&&  eval < VALUE_KNOWN_WIN) // 詰み絡み等だとmate distance pruningで枝刈りされるはずで、
ここでは枝刈りしない。
	return eval;
```

上のgrid searchの結果は、勝率
がPARAM_FUTILITY_MARGIN_ALPHA1,PARAM_FUTILITY_MARGIN_ALPHA2の関数だとして、関数の単峰性が見てとれる。
また、この結果は、0.1秒で強くするためには、PARAM_FUTILITY_MARGIN_ALPHA1をもっと下げよ(枝刈りを激しくせよ)、PARAM_FUTILITY_MARGIN_ALPHA2をもっと上げよ(枝刈りをさらに激しくせよ)と訴えかけている。(ちなみに、この２つの定数は、長い持ち時間で棋力が最大になるように調整したときのそれぞれのベストな値は、172と50であった。)

つまり、このgrid searchの結果に素直に従い、0.1秒で最強に調整していくと、枝刈りが激しくなり、評

価値のぶれが大きくなり、勝率項の精度が下がると考えられる。

そこで、やねうら王では、無作為に抽出された局面の評価値の分散を著しく上げないという拘束条件を追加しつつ、探索パラメーターを0.1秒で最強に調整する。

■ 結論

以上により、質の良い勝敗項と質の良い勝率項という相矛盾する要素を持つ教師生成を従来より2倍以上低い計算コストで実現が可能になった。

しかし何故かまだ昨年のものから強くならない。ゲロ吐きそうである。大会当日までにこのPR文書を書き直す。とりあえず、現時点[2019/03/19]ではこんな感じで取り組んでいますよ、ということで。

[2019/03/26追記] とりあえず、新しい評価関数が、SDT5(第5回 将棋電王トーナメント[2017])のときと同じぐらいの強さになった。まだマイナビから発売した『将棋神やねうら王』に収録したtanuki-(2018年度版)にR60ぐらい負けている。理由はわからない。禿げそう。

[2019/03/31追記] 使用しているライブラリについて書き忘れていたので以下に追記。

- やねうら王ライブラリ
  選定理由：自作のライブラリですが、申請が必要なのかどうか理解できていないのでルール上の地雷回避のために申請しておきます。

- tanuki-ライブラリ
  選定理由：やねうら王のGitHubにNNUE評価関数のプルリクエストをもらったのでマージしたのですが、
  これについて申請が必要なのかどうか理解できていないのでルール上の地雷回避のために申請しておきます。
  tanuki-ライブラリに関して、やねうら王のGitHubにあるコード以外は使用しておりません。

第２９回世界コンピュータ将棋選手権
参加プログラム　水匠

水匠アピール文書

平成３１年３月２１日

参加者　杉　村　達　也

第１　はじめに

１　本文書の目的

本文書は、第２９回世界コンピュータ将棋選手権（以下「WCSC29」といい、前回選手権を「WCSC28」といいます。）の参加プログラムである水匠の紹介及び独自工夫の要点等を記述するものです。

２　参加者及び参加プログラムの紹介

（１）本参加者[1]は、WCSC28 において tanuki-製作委員会が作成した、NNUE 評価関数に追加学習をさせた野良評価関数（NNUEkai）[2]を作成した者であり、当該野良評価関数が一部の方から評価されたことから、無謀にも WCSC29 の初参加を決意した者です。

（２）参加プログラムは、水匠といいます（「すいしょう」と読みます。）。現在、インターネット上で公開されている NNUE 評価関数の名称が、イルカ、シャチ、貝など、水にまつわるものが多いことから、水という文字を使用し、「水匠」と名付けました。

第２　使用ライブラリ

１　使用するライブラリの内容

水匠においては、世界コンピュータ将棋選手権ライブラリのうち、①tanuki-（wcsc28 版）[3]、②やねうら王コンピューター将棋フレームワーク[4]、及び③Apery[5]を使用します。

２　ライブラリ使用理由

①tanuki-（wcsc28 版）で採用された NNUE 評価関数は、現在、教師データによる強化が最も容易・有効な評価関数であり、本参加者が作成した野良評価関数も NNUE 評価関数であったため、使用させていただきます。

②やねうら王コンピューター将棋フレームワークは、NNUE 評価関数を動作させるた

---

[1] 本参加者の Twitter アカウントは、https://twitter.com/tayayan_ts
[2] https://1drv.ms/f/s!AjMACEoJwb5ngpY6NcU9qhdV5XS4UQ
[3] https://github.com/nodchip/tanuki-
[4] https://github.com/yaneurao/YaneuraOu
[5] https://github.com/HiraokaTakuya/apery

めの探索部として使用させていただきます。

③Apery は、その評価関数が、世界コンピュータ将棋選手権ライブラリの中で特に強い評価関数であるため、教師データの作成において使用させていただきます。

第３　参加プログラムの独自工夫

１　NNUE 評価関数の特徴

（１）NNUE 評価関数は、ニュートラルネットワークを利用した非線形評価関数であって、将棋における戦型把握とその戦型に対する駒の配置の学習能力について大変優れていると推測されており[6]、現在、従来の三駒関係を利用した評価関数に比べて、より強い評価関数が作成できると考えられています。

（２）ただし、NNUE 評価関数は、局面の理解力が極めて高いものの、そのネットワークサイズの小ささが影響している結果、多数の局面を踏まえた学習をすることを苦手としており、本参加者の実験によれば、学習限界局面数（それ以上の局面を用意しても、評価関数を強くできない局面数のことを指します。）が、三駒関係評価関数に比べて少ないことがわかっています。

（３）また、NNUE 評価関数は、局面の理解度が高いことが影響して、少ない学習局面数であっても、その評価関数を大きく変容させてしまうという特徴ももっており[7]、繊細な学習が求められることもわかっています。

２　教師データ自体の変更

（１）前項で指摘したとおり、NNUE 評価関数は、学習限界局面数が少なく、かつ、少ない学習局面数でも大きな変化が生じるといった特徴を持っていることから、NNUE 評価関数の学習においては、その教師データの質が、三駒関係評価関数以上に求められます。

（２）NNUE 評価関数を学習させるための教師データは、現在、既存の評価関数の自己対局による局面及び評価値を主な内容としており、自己対局における探索深度を高めれば、より質のいい教師データを作成することが可能です。しかし、探索深度を高めれば、教師データの作成速度は遅くなってしまうため、必要な量の教師データの作成する場合の探索深度には限界があります。

（３）そこで、探索深度を高める以外の方法によって、教師データの質を高めることが可能であれば、必要な量の教師データを、より高速に作成することが可能であると考えられます。

本参加者が WCSC29 で採用する手法は、自己対局によって作成された教師データの情報を、事後的に変更することによって、教師データの質を高め、その結果、より質

---

[6] http://yaneuraou.yaneu.com/2019/02/05/

[7] 評価関数 illqha 作者のめきっと氏（https://twitter.com/_illqha）も同旨を言及しています。

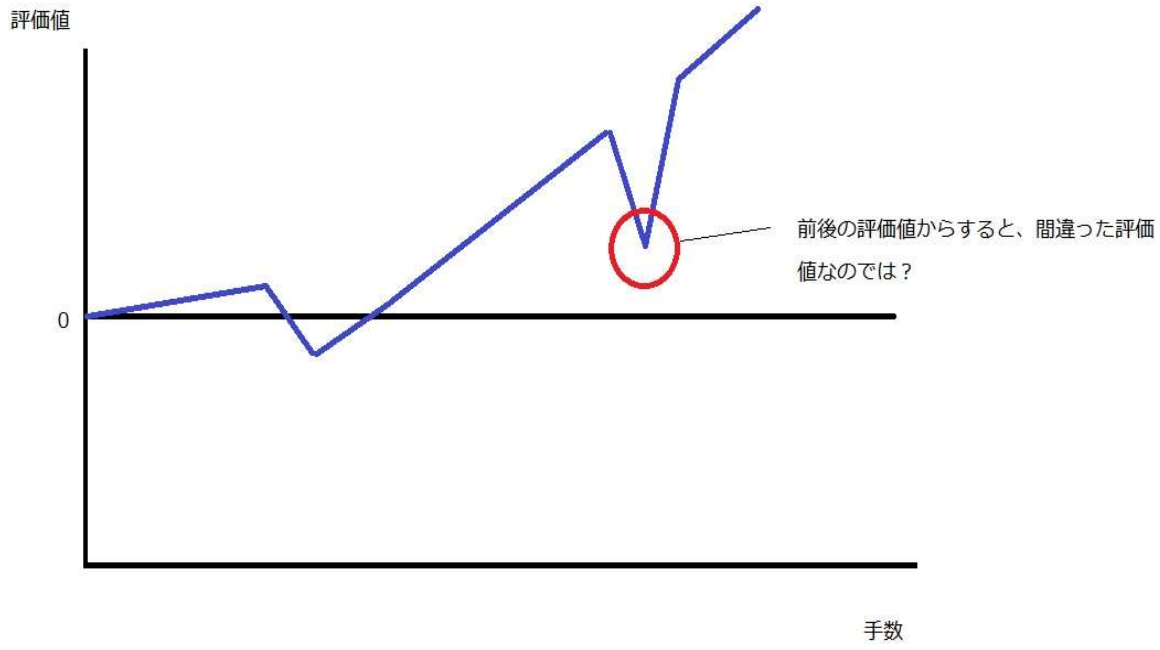
の高い評価関数を作成するという手法です。

3　教師データ自体を変更する利点

NNUE 評価関数（及びやねうら王）における教師データの内容は①手番、②局面、③評価値、④最善手、⑤勝敗、⑥手数によって構成されています。そして、教師データ作成時において、教師データの手数は一対局毎にまとまった順列となっています。

したがって、教師データ自体の変更という手法を採用すれば、前後の局面における評価値を考慮した上で、教師データを変更することが可能です（下記画像参照）。

なお、前後の局面を考慮する形で、強化学習をすることは、学習部の改造によっては困難であると考えられます（学習時にはシャッフルされた局面を利用するのが原則であるため。）。この点で、教師データ自体の変更という手法には、一定程度の有用性・利点があると考えられます。



また、例として、Depth8 の 5 億局面の教師データのみを使用して評価関数を作成した場合、教師データ変更の有無によって、評価関数の勝率に有意差が存在するため（下記画像参照）、教師データ変更の手法には一定程度の効果があると考えられます。なお、現在は、教師データ変更の手法がより緻密なものとなったため、下記画像時点のものよりも、データ変更の効果は上がっています。

4　教師データ変更の他の利用方法

（1）教師データ自体を変更するという手法は、評価関数強化以外にも利用ができ、例えば、対抗形による自己対局を実施させた教師データのうち、振り飛車側が敗北した対局を一定程度減らすことによって、教師データにおける、振り飛車の勝率を高めることができます。

そして、現在、NNUE 評価関数（及びやねうら王）で使用されている学習手法は elmo 式（局面の評価値及び当該対局の勝敗を考慮した学習手法）[8]であるため、意図的に振り飛車の勝率を上げた教師データを使用することにより、振り飛車を指す評価関数を作成することが可能となるのです（当該手法により作成された評価関数が拙作 NNUEkaiXF です。）。

（2）従来、振り飛車を指す評価関数の作成方法は、飛車の位置によって、評価値を増減させるという手法が採られていました[9]が、新たに、振り飛車側の勝率を変更するという選択肢も増えたことによって、振り飛車評価関数をより強化するための基礎が出来上がりつつあると考えられます。

5　学習パラメータ及び定跡に関する工夫

（1）NNUE 評価関数は、第３・第１項（３）で示したとおり、繊細な学習が求められるものであり、その学習パラメータの設定も、ある程度の重要性を有しています。

本参加者は、今までの野良評価関数（NNUEkai）の作成において、NNUE 評価関数の学習パラメータについて試行錯誤を繰り返してきており、その試行錯誤は、水匠にも活かされることとなります。

（2）また、定跡の作成については、「どういう展開にすれば自分が勝ちやすいかという点も踏まえた（中村太地七段による豊島将之二冠評）[10]」定跡作成が有効であると考

[8] http://www2.computer-shogi.org/wcsc27/appeal/elmo/elmo_wcsc27_appeal_r2_0.txt
[9] https://www.apply.computer-shogi.org/wcsc28/appeal/HoneyWaffle/appeal.pdf
[10] https://bizgate.nikkei.co.jp/article/DGXMZO34862020310820180000000

えています。本参加者が今まで作成してきた定跡（白黒定跡）は、単なる勝率ではなく、圧勝率（先手番評価値及び後手番評価値が、一度も敗者側に振れないまま全く紛れなく勝ち切った対局の勝率）を考慮して作成したものです。このような定跡作成手法によって作成された定跡は、評価関数単体よりも勝率を高くすることが可能で、かつ、定跡の穴を狙われにくいものであると考えています（実際はどうかわかりません。）。

WCSC29 においても、同じ手法で作成した定跡を採用する予定です。

第４　おわりに

以上が、水匠の紹介及び独自工夫の内容になります。

本参加者は、全くと言っていいほどプログラミングに関して初心者であり、上記独自工夫についても、プログラミングを初めて一日で習得できるような簡易な技術しか使われておりません。学会等に出没するといわれている、素人を名乗る者ではなく、ガチ素人です。

また、本参加者の日常業務も計算や機械とかけ離れた、文系職業の最たるものであるので、今回の WCSC29 への参加は、いわば、最近流行りの異世界転生ものであるといえるでしょう。

小説においては、凡そ成功する異世界転生の主人公ですが、現実はそう甘くないのか、それとも何かしらの爪痕を残せるのか、ご期待ください…ではなく、全く期待せず見守ってください。

以上

3. 探索部について

Policy Network がない為、AlphaZero 等の現在の最強と考えられる DL 系のプログラムの探索方法はそのままでは流用する事が出来ません。
探索部として、2019 年 3 月 30 日現在、いくつかの案を試してみています。

3－1. Policy の代わりに「うさぴょん」の手の評価を用いてみる

元々、うさぴょんの手の評価は（駒打ち以外は）相当正確でしたので、これを Policy の代わりに用いる事を試みています。今の所、手の評価が重過ぎる感じと、手の評価値は直接指し手の確率として使うことは出来ないので色々と小細工をしていますが、有効に働いているかどうかが疑問です。また、これを使うなら、うさぴょんの後継では…という感じが拭えませんw

3－2. 評価関数から全ての合法手の評価が返って来るので Policy の代わりに使って見る

性質の違う PolicyNet と ValueNet の組み合わせの故、AlphaZero 等の探索が上手く動いている可能性が高いのですが、Value だけから同じような事を試みる方法です。理論的に破綻している可能性が高いので、あまり深く追わない予定です。

3－3. 単純に UCB1 等の多腕バンディット問題向けのアルゴリズムを用いる

工夫がなくて面白くないのですが、選手権で実際に用いる可能性が今の所一番高いです。（池が囲碁の方で過去に経験している為。）

3－4. さらに単純に αβ 法などをそのまま用いる

もしもこれで強かったらこの方法を使いますが、これで上記に挙げたアルゴリズムのいずれよりも有意に強かったら…何を悩んでいたのか、という話になります。

4. 最後に

今のままだと探索部がイモい為に全体が弱くなる可能性が高いです。
*2019/04/16 追記：探索部、HDD 故障により、ロストしました。バックアップはあったのですが、3 週間分位巻き戻ってしまった感じです（爆）。

なお、評価関数単体では 1 手読みでもそれなりに強い事が確認出来ています。

ですので、ここからの探索部の頑張りにご期待ください！

AobaZero は、AlphaZero の将棋の実験の追試を行うことを最終目的とした将棋人工知能プロジェクトです。予備実験では、ゼロから開始する強化学習の初期段階で Elo レーティングが毎日 100 以上上昇し続けていることを確認しています。残念ながらこの実験の規模は桁違いに小さく、ニューラルネットワークの大きさは約 16 分の1、棋譜生成スピードは約 1000 分の 1 です。

そこで、Leela Zero (https://zero.sjeng.org) のようなユーザ参加型のコンテンツも作成して、AobaZero の実験の規模を大きくする計画を立てています。選手権までにはこのコンテンツ（棋譜や学習モデルと実験に必要なコードファイル一式を含む）を公開予定です。elmo を凌駕するような棋力を獲得する強化学習の過程を、我々と一緒に観察していきませんか？

将棋の知識を獲得していく過程が面白いので棋譜だけ先に公開しました(2019/03/31)。
http://www.yss-aya.com/aobazero/

github の予定地：
https://github.com/kobanium/aoba_zero

使用予定のソースコード：
Bonanzaの盤面構造を用いてモンテカルロ木探索を実装し、
OpenCL で GPU を利用する LeelaZero のコードを変更してニューラルネットの計算を行っています。

AobaZeroは以下の4つに分かれます：
1. 棋譜を作成する aobaz
2. aobazを 動かし、棋譜をサーバに送る autousi
3. ネット上のマシンから棋譜を集め、ネットワークの重みを配布するサーバ
4. 棋譜をまとめて学習し、ネットワークの重みを作る学習部

予備実験の強化学習の初期段階の観察結果：

1. 駒の動かし方(将棋のルール)を覚える(minibatch 64、最初の2000回の学習で)。

2. 金銀が2段目、王の近くにいると負けにくい、というのを覚える。まだ駒の損得は分かってない(13万棋譜)。

3. 駒を取れば勝ちやすいのがぼんやり分かってる(21万棋譜)。

4. 駒を取られたら取り返すのは少し理解(25万棋譜)。

GCT (Google Colab TPU) 将棋

基本方針
* 将棋AIで学ぶディープラーニング (山岡忠夫著) [1] の手法を参考にする
* 将棋以外にも活用するため、 Python を利用する
* 高速化が見込まれる場合は、 Cython を利用する
* Google の AlphaZero の手法をリスペクトするため、Google Colab (GCP) の TPU を利用する
* TPU のバグを回避するため、ローカルPCのGPU も併用する

使用ライブラリ

python-shogi
* 学習データの前処理（floodgate の棋譜）に使用予定
* 対局時に使用予定

elmo
* 自己対局データを学習データのバリエーションに使用予定

dlshogi/Apery
* C++の実装を参考にする予定

Tensorflow/Keras で以下のネットワークを作成
* 方策ネットワーク(policy network)
* 価値ネットワーク(value network)
* マルチタスク学習
* Residual Network

Google Colab の参考資料
* 【秒速で無料GPUを使う】TensorFow(Keras)/PyTorch/Chainer環境構築 on Colaboratory [2]
* 【秒速で無料GPUを使う】深層学習実践Tips on Colaboratory[3]

TPU 対応の参考資料
* 公式の notebook がある[4]
* Cloud TPU チュートリアルがある[5]
* Tensorflow/Keras の TPU 対応モデルがGitHubで公開されているが、Tensorflow 1.x では experiment[6]
* Tensorflow 2.x からクラウド TPU での Keras の利用をサポートするロードマップになっている[7]
* TPU で学習したモデルの学習／推論には不具合があるため、必要に応じてローカルPCのGPUで推論をする[8]
* Google Colab から Google Drive をマウントして、学習データ／学習済みモデルのローカル／クラウド間のやり取りができる[9]
* Google Colab に SSH接続する場合は、ngrok 経由で Connecting to instance over sshができる[10]
* 日本語情報が少ない。Shikoan's ML Blog が参考になる[11]

TBD
* Residual Networkのブロック数を増やす
* Network構成は、AlphaGo/AlphaZeroクローンごとに差異があるため、比較検討
* C++による高速化
* 大規模学習
* 強化学習
* ...etc

備考
* Google Colab がTPU対応した！ TPU パワーで手軽に強くなるんじゃね？っと思ったら、そんなうまい話はなかった。
* Tensorflow/Keras のバージョンで TPU の挙動がよく変わる。
* GPU で動くコードが TPU で動かないことが多い。デバッグが辛い。
* とはいえ、Google Colab 上で、TPU が無料で遊べるのは魅力的。使えるところには積極的に使いたい。
* 山岡ジェバンニ [12] が同様の手法を進めてくれた。

参考文献

* [1] https://book.mynavi.jp/ec/products/detail/id=88752
* [2] https://qiita.com/tomo_makes/items/f70fe48c428d3a61e131
* [3] https://qiita.com/tomo_makes/items/b3c60b10f7b25a0a5935
* [4] https://colab.research.google.com/notebooks/tpu.ipynb
* [5] https://cloud.google.com/tpu/docs/tutorials
* [6] https://github.com/tensorflow/tpu/tree/master/models/experimental
* [7] https://www.tensorflow.org/community/roadmap
* [8] https://www.tensorflow.org/api_docs/python/tf/contrib/tpu/TPUEstimator#prediction
* [9] https://colab.research.google.com/notebooks/io.ipynb#scrollTo=c2W5A2px3doP
* [10] https://medium.com/machine-learning-world/useful-snippets-for-google-colaboratory-free-gpu-included-d976d6b3e6de
* [11] https://blog.shikoan.com/tag/tpu/
* [12] http://tadaoyamaoka.hatenablog.com/entry/2019/03/13/001515